REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFI

UNIVERSITE MENTOURI CONSTANTINE

FACULTE DES SCIENCES DE L'INGENIEUR

DEPARTEMENT D'ELECTRONIQUE

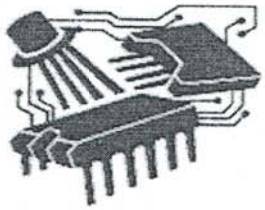
POLYCOPIE DE COURS

TRAVAUX DIRIGES ET TRAVAUX PRATIQUES

T. LAROUSSI

M. T. BENHABILLES

A. FARROUKI



MICROPROCESSEUR INTEL 8086 ET SES PERIPHERIQUES

ARCHITECTURE ET PROGRAMMATION

ANNEE UNIVERSITAIRE 2009-2010

TABLE DES MATIERES

CHAPITRE 8

INTERRUPTIONS DANS LE 8086 105

CHAPITRE 9

CIRCUIT D'ENTREES ET DE SORTIES PARALLELE ET PROGRAMMABLE 128

CHAPITRE 10

CIRCUIT D'ENTREES ET DE SORTIES SERIEL ET PROGRAMMABLE 144

ANNEXES

PRÉFACE

Lorsque Gordon Moore, co-fondateur d'Intel, avait prédit en 1965 que la densité en transistors des circuits intégrés allait doubler tous les deux ans, savait-il que sa prédiction allait se confirmer avec une précision inouïe jusqu'à aujourd'hui? Nous constaterons qu'entre temps la vitesse des calculateurs a augmenté à peu prés au même rythme que l'accroissement du taux d'intégration.

Bien que les machines algorithmiques ne semblent pas avoir de lien direct avoir l'architecture des PC, elles demeurent quand même une source introductrice du microprocesseur dont l'unité de commande matérialise l'algorithme d'interprétation du répertoire d'instructions. De part ce fait, les automates de commande tels l'incrémental est celui de Wilkes, que nous avons étudiés dans le chapitre 1, peuvent convenir à introduire les architectures de Von Neumann et Harvard et les technologies CISC, RISC et WISC. Il n'en demeure pas moins que l'enseignant en charge de ce cours soit libre des ses choix.

S'il nous sera reproché à juste titre que l'exposé du chapitre 2 reste incomplet non pas relativement au phénomène de mode, mais du fait des limites théoriques imposées par la structure même de l'IBM PC-XT (absence des concepts des mémoires cache, virtuelle et paginée, des RAM vidéo et shadowing, interfaces multimédia, chipsets, etc. ...), nous renvoyons à ce sujet le lecteur intéressé à la littérature spécialisée. Néanmoins, pour orienter les curieux et assouvir la faim des insatiables, nous présentons, dans le chapitre 3, un aperçu succinct de l'évolution vertigineuse de la technologie des micro-ordinateurs de la famille Intel et des différents chipset, bus, interfaces, mémoires et disques durs actuels.

En 1971, Intel surprend toute la planète en lançant le premier microprocesseur 4 bits. Il s'agissait du 4004 (2300 transistors et 60000 instructions par seconde). Depuis, on ne le répétera jamais assez, Intel n'arrête pas de créer l'événement en annonçant régulièrement de nouveaux produits toujours plus performants et en honorant ses annonces.

Pour une technologie en perpétuel mouvement nous ne pouvons nous donner la prétention de rendre fait de l'état de l'art et notre exposé se veut avant tout didactique. Notre choix s'est donc délibérément porté sur le microprocesseur 8086 de Intel, chapitres 4-6. En effet celui-ci, au même titre que les machines IBM PC-XT qui en sont équipées, réunit dans une large mesure toute la théorie moderne des machines universelles telle qu'elle à été édictée par Johannes Von

Neumann (1945). Par ailleurs, de part sa compatibilité ascendante, il constituera un outil d'apprentissage idéal où l'application pratique peut se réaliser sur n'importe quelle machine de fabrication ultérieure dans un parc très largement doté.

Sans montrer beaucoup de détails, le chapitre 7 reprend succinctement les modes de transferts des entrées en sorties dans les systèmes à microprocesseurs. Au vu de l'évolution vertigineuse du taux d'intégration dans les microprocesseurs, l'accent a été mis sur le mode des entrées et sorties par les interruptions.

Pour ce faire, nous savons, chapitre 8, que c'est précisément, en présence d'un nombre important de périphériques que les interruptions étalent toute leur efficacité en libérant le microprocesseur de la fastidieuse tâche de scrutation, améliorant du coup, d'une façon significative, les performances du système. Le PC est un exemple typique dans lequel les périphériques de base sont exclusivement pris en charge concomitamment par les interruptions BIOS et MS-DOS et les interruptions matérielles (Clavier, horloge cyclique, interface parallèle, interface sérielle, contrôleur disque dur, contrôleur de lecteur disquettes et RTC/ RAM CMOS dans les PC AT).

Ce qui apporte le plus de complication dans la conception d'un système, en logique programmée c'est la communication microprocesseur-périphérie. La raison tient essentiellement au fait qu'il n'y ait aucune standardisation dans les périphériques de sorte qu'un microprocesseur ne peut pas, en général, commander directement les périphériques. Un circuit intégré d'adaptation appelé interface s'avère nécessaire entre le microprocesseur et les périphériques. A de très rares exceptions près. L'interface établit donc une compatibilité entre les entrés et sorties du microprocesseur et celles des périphériques, tant au niveau du type de transmission (série ou parallèle), qu'au niveau du code (ASCII, EBCDIC, ISO, ...) ou alors au niveau de la vitesse de transmission. Pour ce faire, les chapitres 9-10 étudient, respectivement, un exemple d'interface parallèle et celui d'un interface série.

Enfin, le chapitre 11 introduit les deux types d'interfaces de communication qui existent sur les PC, à savoir, les interfaces parallèle (Centronix) et série (RS232 et USB). Remarquons, cependant que cette dernière décennie a connu une montée en puissance de l'interface série USB. A l'heure actuelle, tous les PC en sont dotés en nombre important, provocant ainsi la disparition de l'interface parallèle.

A l'exception des chapitres 7 et 11, tous les autres sont ponctués chacun par une série d'exercices et de manipulations sur micro-ordinateur, organisées

respectivement, sous forme de travaux dirigés et de travaux pratiques. Ces derniers ont été élaborés à l'issu de plusieurs années d'enseignement des systèmes à microprocesseurs.

Enfin, à la fin du manuscrit, les annexes constituent un support pour la compréhension de certaines sections abordées aux différents chapitres, travaux dirigés et travaux pratiques.

CHAPITRE 1

MACHINES ALGORITHMIQUES

I Introduction

Tout processus de traitement d'information numérique est considéré comme un calcul au sens algébrique du terme. L'exécution d'un calcul repose sur la mise en œuvre d'un algorithme, défini de la manière suivante:

1- Un algorithme est l'assemblage de primitives de calcul selon un ensemble de règles de construction admissibles.

2- Pour un concepteur de processeur, la matérialisation de la primitive de calcul devient l'idée centrale autour de laquelle sera élaboré l'ensemble du schéma architectural.

II Notion d'automate fini

Un automate fini peut se définir comme un système algébrique $A = < \Sigma, \Omega, Q, M, N >$, où

Σ : Ensemble fini des variables d'entrée.
Ω : Ensemble fini des variables de sortie.
Q : Ensemble fini des variables d'état interne.
M : $\Sigma \times Q \rightarrow Q$ Fonction de transition, associe l'état actuel à l'état suivant.
N : $Q \rightarrow \Omega$ Fonction de sortie, associe l'état interne à la sortie.

Fig. 1 Schématisation d'un automate fini

Un automate ainsi défini s'appelle automate de MOORE. Il existe un second type d'automate appelé, automate de MEALY, qui se différencie du premier par le fait que la fonction de sortie dépend non seulement de l'état interne, mais aussi des variables d'entrée tel que N : $\Sigma \times Q \rightarrow \Omega$.

III Matérialisation de la primitive de calcul par un circuit logique combinatoire

Les opérateurs réalisant les primitives de calcul peuvent être des circuits logiques, des réseaux logiques programmables (PLA), ou des mémoires mortes (ROM). Ils seront appelés indifféremment ressources. On les symbolisera par une boite noire notée Π_i.

Fig. 2 Schématisation d'une ressource

Le circuit logique combinatoire consiste en une interconnexion sans boucle des ressources. Il sera noté C. C.

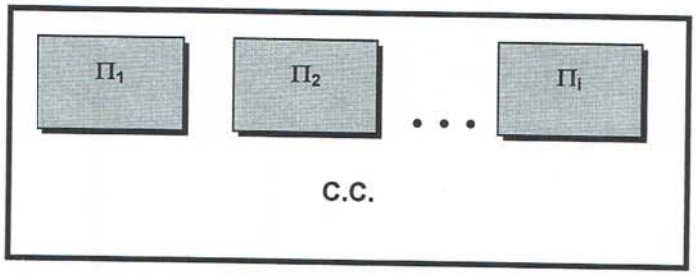


Fig. 3 Schéma synoptique d'un circuit combinatoire

En introduisant un registre mémoire de donnée M_D et un registre mémoire résultat M_R, la primitive de calcul sera représentée par un transfert fonctionnel $[M_R] := f([M_D])$, où f est une fonction arithmétique ou logique. On aboutit à l'organisation schématisée de la Figure 4.

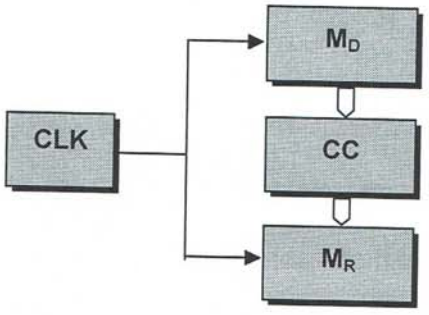


Fig. 4 Schématisation d'un transfert fonctionnel

Cette réalisation comporte, outre les mémoires M_R, M_D et la ressource CC, l'embryon d'un circuit de commande CLK, représenté par exemple par l'impulsion électrique qui décide du moment où s'effectue le transfert.

Une telle réalisation est appelée circuit combinatoire parce que la valeur mémorisée en M_R lors d'un cycle d'évolution ne dépend que du contenu de M_D lors de ce cycle et non de l'histoire antérieure du circuit. Nous l'appellerons système à temps d'exécution unité.

IV Modèle de Glushkov

Dans ce qui précède, nous avons vu la décomposition naturelle d'un système de calcul en deux sous-ensembles autonomes: l'unité de commande et l'unité de traitement. Dans le modèle de Glushkov, l'unité de commande reçoit un retour d'information par le biais de la réaction (a).

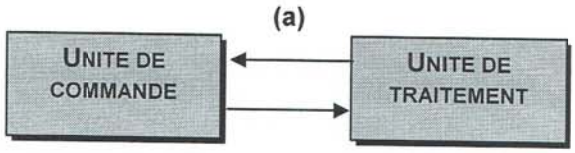


Fig. 5 Schéma synoptique du modèle de Glushkov

IV.1 Unité de traitement

Dans le cas le plus général, l'unité de traitement d'un processeur de calcul possède une structure assez régulière. Elle est constituée d'un ensemble {R} de registres permettant la mémorisation des grandeurs traitées, un ensemble {CC} de ressources matérialisant les primitives de calcul, et un réseau de connexion contrôlé permettant de mettre en rapport les registres et les ressources.

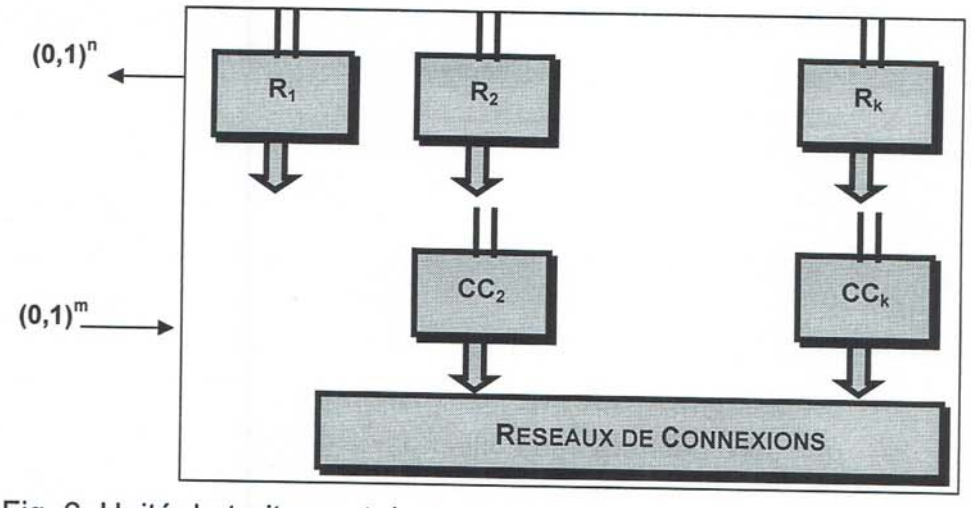


Fig. 6 Unité de traitement du modèle de Glushkov

Dans le modèle de Glushkov, l'unité de traitement répond à la définition d'un automate fini. Pour cette raison on l'appellera automate opérationnel, noté $< \Sigma, \Omega, Q, M, N >$.

Les éléments de l'alphabet d'entrée Σ sont appelés COMMANDES. Chacune des commandes est représentée par un vecteur à m composantes binaires matérialisées par:

Les lignes d'activation (Enable) ou d'inhibition (Inhibit) des registres.
Les lignes de programmation des ressources.
Les lignes de contrôle du réseau de connexions.

Les éléments de l'alphabet de sortie Ω sont codés au moyen de n variables binaires qui seront appelées variables de condition (Status Variables, Flags). Elles renseignent l'unité de commande sur les résultats partiels obtenus au cours de l'exécution d'un calcul.

La fonction de sortie est une application $N: Q \rightarrow \Omega$ car les variables de condition de l'automate opérationnel ne dépendent que de l'état interne et non de la commande. L'automate opérationnel est donc un automate de Moore.

Il reste à définir la fonction de transition M de l'automate opérationnel. La taille de l'alphabet d'entrée Σ et de l'ensemble des états internes Q exclut la description de cette fonction par une table des phases ou un graphe d'états. Nous représenterons la fonction de transition par un langage de transfert, défini comme suit:

A chaque commande $\sigma \in \Sigma$ s'associe une transformation $Q \rightarrow Q$ de l'ensemble des états internes de l'automate opérationnel. Cette transformation sera décrite par l'ensemble d'équations:

$$\sigma \rightarrow L\,[R_i] := f_i\,([R_1], [R_2], \ldots, [R_k]) \qquad i=1,k$$

Comme la notation [R] représente le contenu du registre R, les $[R_i]$ au membre de droite sont relatifs au contenu précédant l'application de la commande σ, ceux de gauche sont les nouveaux contenus qui s'inscrivent dans les registres après l'application de cette commande.

IV.2 Unité de commande

L'unité de commande apparaissant dans le modèle de Glushkov est, au même titre que l'unité de traitement, un automate fini. Ceci permettra de la synthétiser matériellement par des circuits combinatoires ou des circuits séquentiels synchrones.

A la lumière de la description de l'automate opérationnel, il devient évident que l'automate de commande va essentiellement compléter la matérialisation de l'algorithme de calcul à mettre en œuvre, compte tenu du langage de transfert de l'automate opérationnel.

Mishenko décrit les algorithmes de calcul par un programme, qui est constitué, par une suite d'instructions. Si les instructions représentent les fonctions de transition de l'automate de commande, nous pouvons sans ambiguïté associer programme et automate de commande.

La structure la plus simple a été proposée par WILKES en 1951, Figure 7. Le programme réside dans une mémoire morte, dans laquelle s'établit une relation biunivoque entre les fonctions de transition de l'automate de commande et ses variables de sortie (Commandes). L'automate de Wilkes représente surtout un intérêt historique.

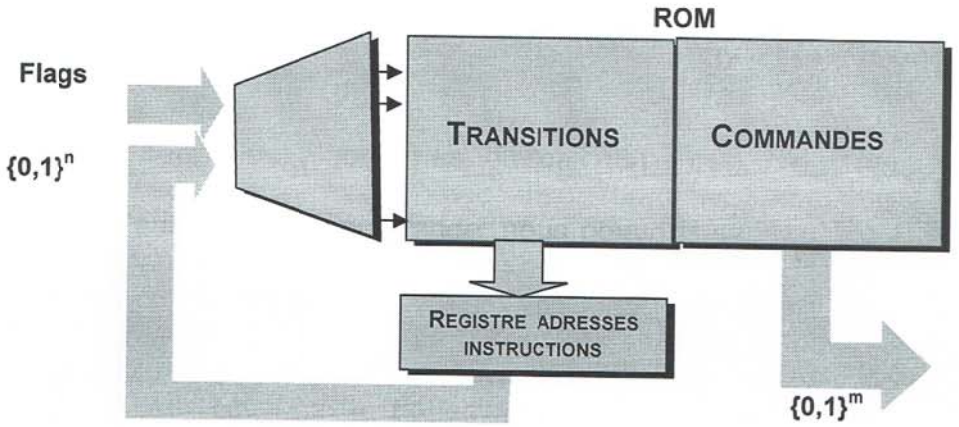


Fig. 7 Automate de WILKES

D'autres automates de commande ont été développés tel que l'automate incrémental, qui est nettement moins coûteux en espace mémoire et plus simple à programmer, moyennant une légère modification du circuit, Figure 8.

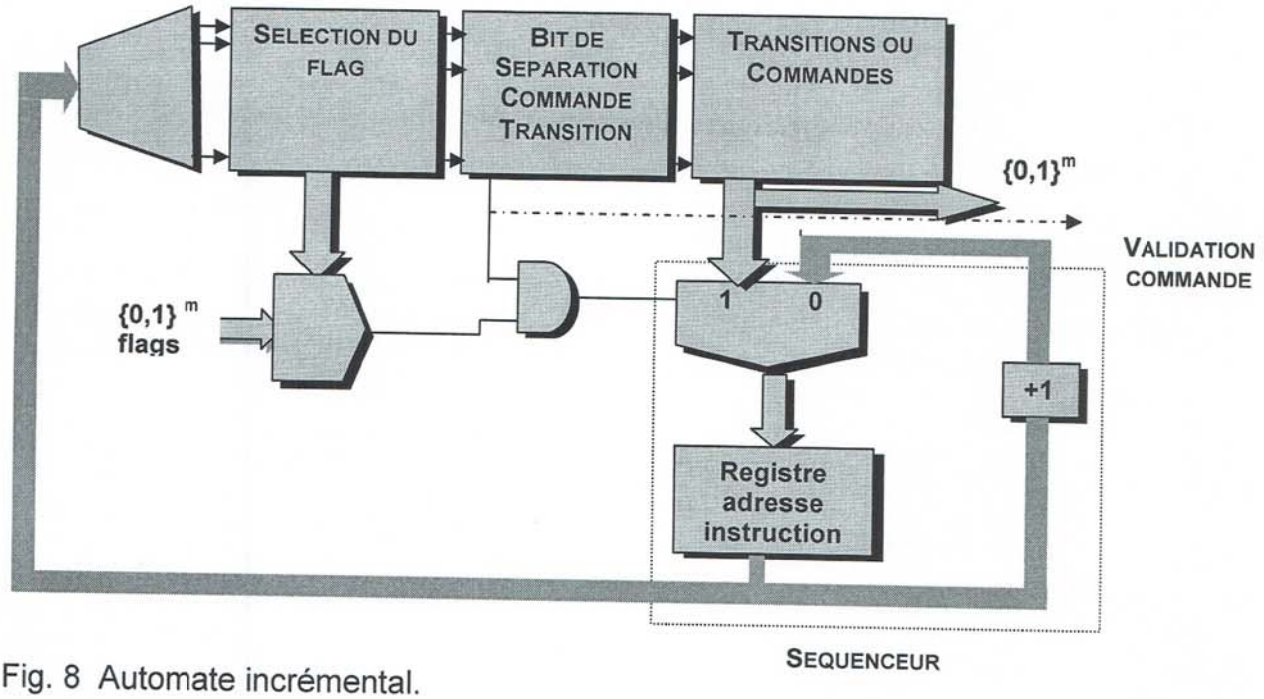


Fig. 8 Automate incrémental.

L'entrée de l'automate de commande, représentée par les variables de condition, contribue amplement à la détermination de la fonction de sortie. L'automate de commande est un automate de Mealy.

V Microprocesseur

Le nombre de registres contenus dans l'unité de traitement peut s'avérer insuffisant, ou du moins technologiquement limité, pour certaines applications. Ceci nécessite un apport extérieur de mémoire vive (RAM), appelé dans un premier temps à absorber le surplus de données utilisées par l'unité de traitement. On aboutit de la sorte à une structure à trois sous-ensembles, donnée à la Figure 9.

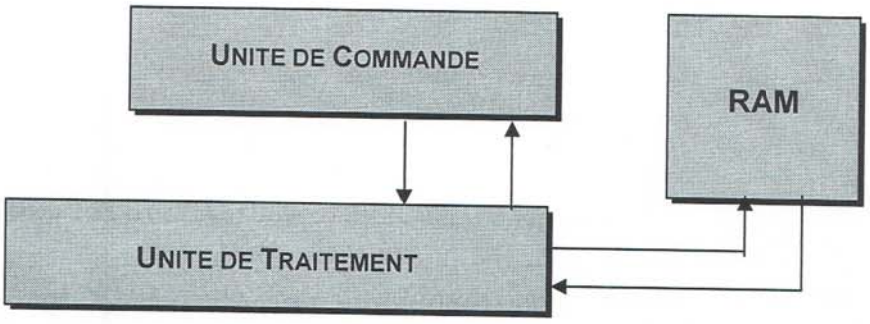


Fig. 9 Blocs fonctionnels d'un microprocesseur

Une idée assez simple, attribuée à Johannes VON NEUMANN (1945), est la suivante:

Disposant d'une mémoire volatile destinée à l'origine à contenir des données, il serait avantageux de pouvoir y implanter des algorithmes, accordant de par ce fait un caractère d'adaptabilité et surtout d'universalité au processeur de calcul. C'est le concept de machine à programme mémorisé ou interne (Stored Program Device). Ceci à l'inverse des premiers ordinateurs dans lesquels le programme était extérieur à la machine et lui était présenté sur un ruban codé.

Le prix à payer est la conception du format optimal sous lequel seront mémorisées les instructions du programme-mémoire, et de prendre en charge leur impact sur les unités de commande et de traitement. Il faudrait aussi établir tout un système d'aiguillage des connexions, appelé chemin de données (Data Path), afin de gérer la communication entre les trois sous-ensembles.

Si la contrainte d'optimalité est d'allier une relative rapidité du calcul à une occupation minimale de la mémoire volatile, nous pouvons disposer d'un alphabet assez complet en adoptant un langage qu'on appellera code machine, dans lequel toute instruction du programme-mémoire sera mémorisée sous l'un des trois formats de la Figure 10.

Fig. 10 Formats d'une instruction dans un microprocesseur

L'exécution d'un calcul dans un tel système consiste à établir une correspondance entre les instructions en code machine apparaissant dans la mémoire, et le déroulement des opérations au niveau de l'unité de traitement.

Comme à l'accoutumée une transition dans l'unité de traitement est régie par la commande que lui délivre l'unité de commande, cette dernière apparaît comme un passage obligé. Il est alors nécessaire d'introduire un registre appelé registre code instruction, qui représente une entrée supplémentaire à l'automate de commande, et qui contiendra le code opératoire de l'instruction en code machine à exécuter.

Le code opératoire, contenu dans le registre code instruction, devra être converti en une séquence de commandes appropriées appliquées à l'unité de traitement. Cette conversion est accomplie par le biais d'un algorithme d'interprétation, dont l'unité de commande est le siège. Le microprocesseur se présente ainsi comme une structure où cohabitent au minimum deux niveaux de langage:

1- Un langage apparenté à un langage de transfert, décrivant l'algorithme d'interprétation. Il est habituellement appelé Microlangage ou Microprogramme.

2- Un langage d'instructions appelé code machine, apparaissant dans la mémoire vive.

Le principe fondamental qui doit être retenu dans un tel système est: L'unité de commande matérialise l'algorithme d'interprétation du répertoire d'instructions. Les automates de commande étudiés auparavant peuvent convenir (automate de Wilkes, automate incrémental, ...).

L'unité de traitement du microprocesseur apparaît habituellement, sur le plan synoptique, sous la forme d'un bloc fonctionnel compact qui réunit toutes les ressources, l'unité arithmétique et logique (ALU), auprès de laquelle on fait figurer seulement les registres spécialisés (accumulateur, registre d'adresse-mémoire, compteur de programme- mémoire,...).

Cette structure d'algorithme d'interprétation est le propre de la technologie classique CISC (Complex Instruction Set Component). Il est nécessaire de citer d'autres types de machines existantes.

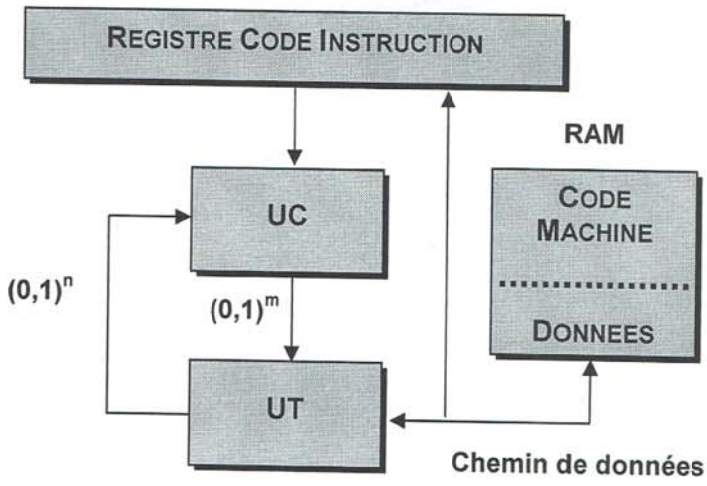


Fig. 11 Le programme-mémoire dans un microprocesseur

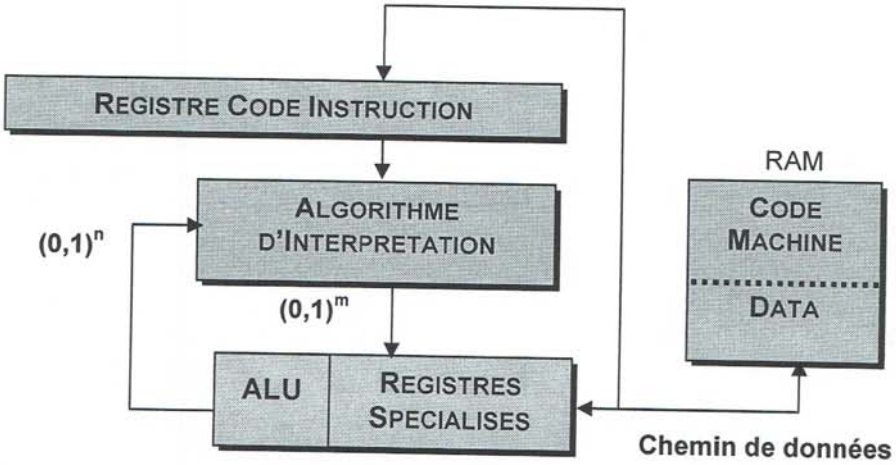


Fig. 12 Le microprocesseur représenté par ses deux niveaux de langage

Les machines RISC (Reduced Instruction Set Component), dans lesquelles le micro-code est réduit à l'extrême, ainsi que les machines WISC (Without Instruction Set Component), dans lesquelles il est inexistant. Une architecture autre que celle de Von Neumann est généralement présente sur ces dernières. Il s'agit de l'architecture Harvard qui se distingue par une séparation da la RAM en un champ code instruction et un champ opérande avec chacun son chemin de données propre permettant un gain de temps appréciable dans les accès RAM.

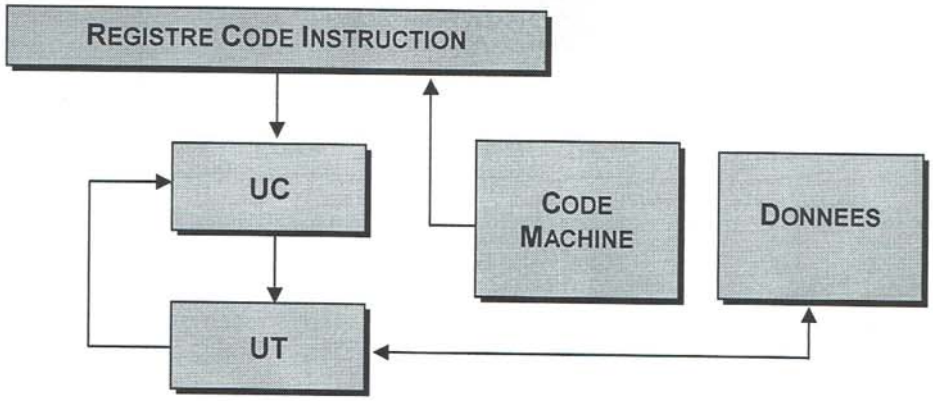


Fig. 13 Architecture Harvard

VI Appel de sous-programme

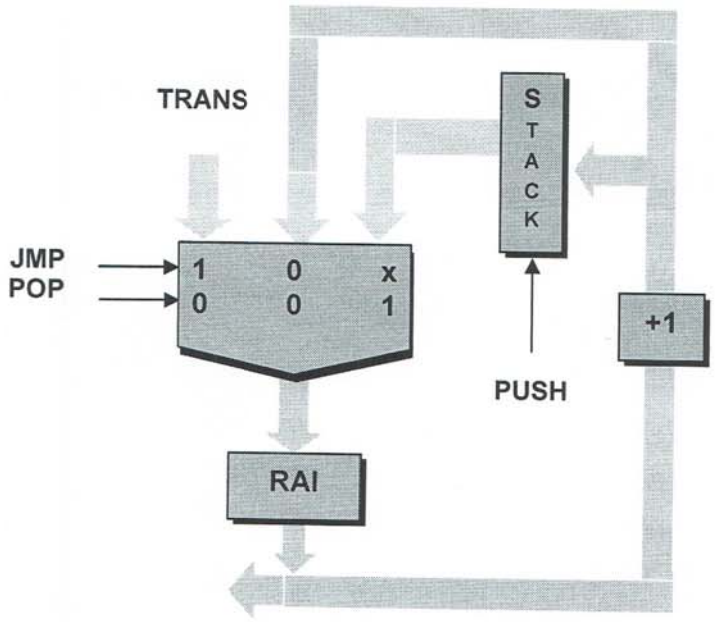


Fig. 14 Séquenceur modifié permettant de faire un appel de sous programme

Le multiplexeur est contrôlé par deux bits notés JMP et POP.
Le chargement du registre STACK s'effectue lorsque le bit PUSH est à 1.
La donnée provenant de la ROM est notée TRANS.

[RAI]:= [RAI]+1→incrémentation du RAI.
[RAI]:=TRANS→transition.
[stack]:=[RAI]+1→appel de sous programme avec sauvegarde de l'adresse RAI.
[RAI]:= [STACK]→retour de sous programme avec restitution de l'adresse.

Le séquenceur d'automate précédent ne permet de faire qu'un seul appel de sous programme valide. Afin de pouvoir effectuer des appels de sous

programme imbriqués, nous sauvegardons les adresses de retour respectives dans une mémoire vive de type FILO, appelée PILE.

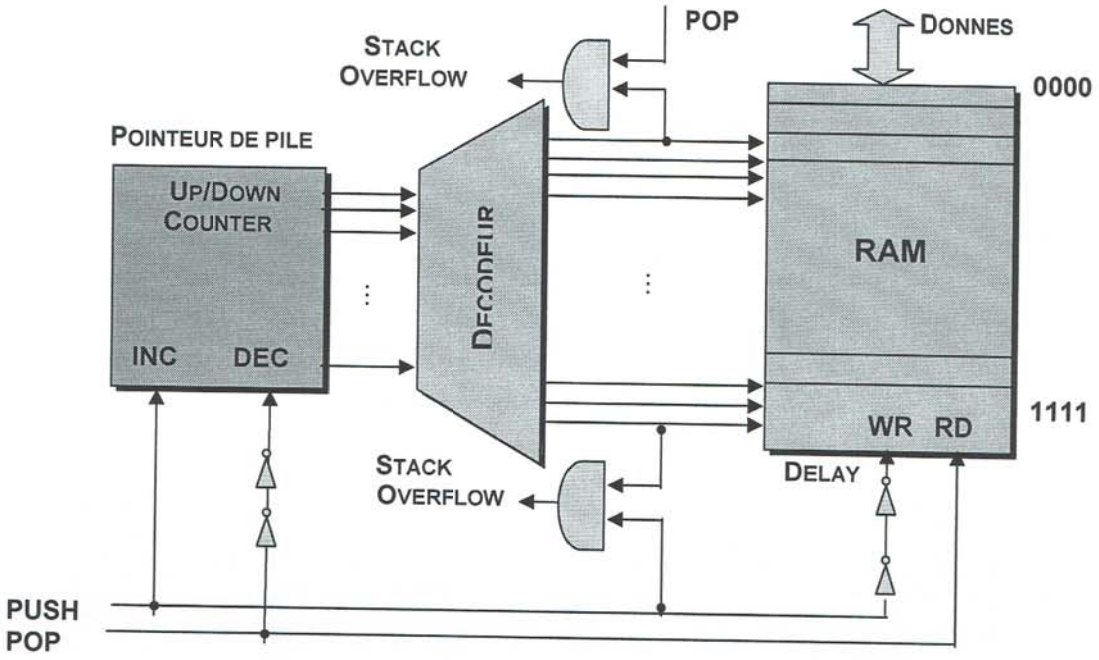


Fig. 15 Schéma de principe d'une Mémoire FILO

VI.1 Mémoire FILO utilisée comme pile de programme

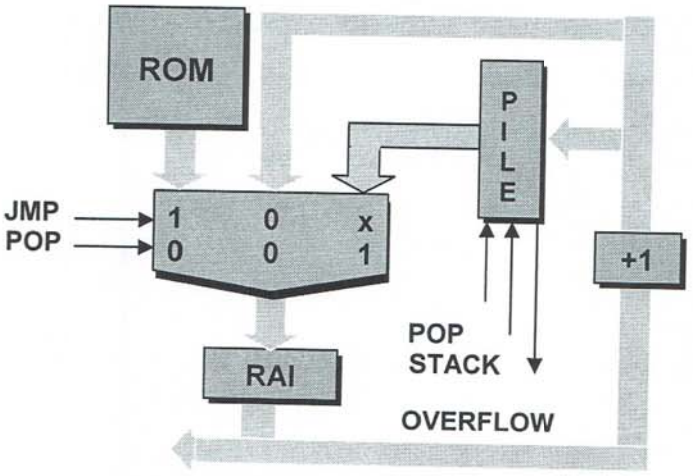


Fig. 16 Séquenceur modifié permettant de faire plusieurs appels de sous programmes imbriqués

L'opération d'empilement obéit à la séquence: Incrémentation du pointeur de pile, ensuite écriture en RAM.

L'opération de dépilement: Lecture en RAM, puis décrémentation du pointeur.

VII Exemple d'application

Algorithme de calcul:

Sans rien charger dans OP1¨, faire la soustraction OP1-OP2.
Si M=0 fin de calcul (division par zéro).
Sinon charger A dans OP1.
Faire la soustraction OP1-OP2.
Si M<0 fin de calcul (avec Q=C et R=OP1)
Sinon incrémenter C et charger M dans OP1
Allez à 4.
Fin de calcul (automate bloqué dans cette étape).

Il vous est demandé de respecter la structure du mot de commande qui doit être comme suit:

b_0, b_1 : Commande du multiplexeur.
b_2=1 : Activation du soustracteur.
b_3=1 : Activation du compteur.
OP1, S, M et C ont tous pour valeur initiale 0.

En respectant toutes les données de l'énoncé, réaliser la partie commande de l'automate incrémental qui matérialise cet algorithme. Donner un schéma complet de l'automate ainsi réalisé (partie traitement, partie commande et connexion.

Solution:

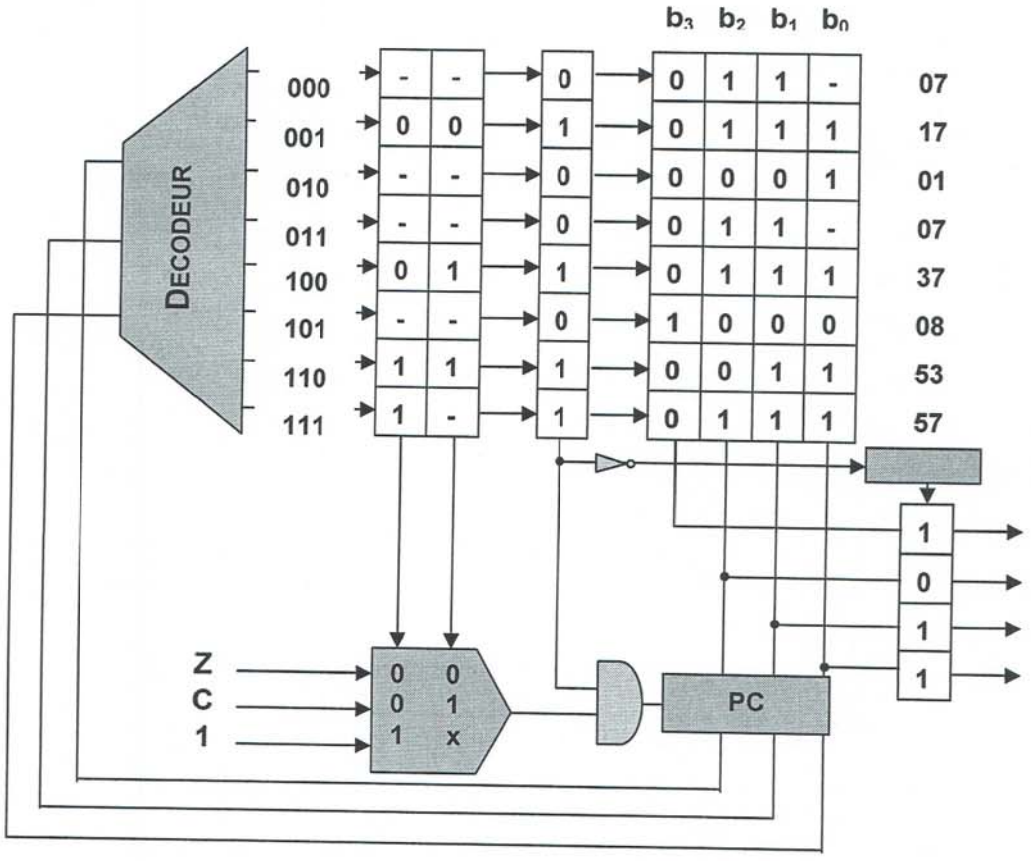


Fig. 17 Unité de commande du circuit de la division

CHAPITRE 2

MICRO-ORDINATEUR IBM PC-XT

I Introduction

La plupart des utilisateurs d'ordinateurs compatibles IBM (Industry Business Machine) n'exploitent sur leur système que les logiciels d'applications. Ils restent ainsi limités à un stade logiciel sans jamais passer au stade matériel. Pourtant, connaître la structure interne d'un ordinateur peut être bénéfique à plus d'un titre à tous ceux qui souhaitent intervenir au niveau matériel sur leur ordinateur que ce soit pour améliorer les performances de la machine ou bien pour concevoir de nouvelles cartes d'extension.

Le premier ordinateur personnel d'IBM mis sur le marché en 1981 était livré avec un microprocesseur 8088 de la firme Intel, une mémoire centrale (mémoire vive ou RAM et mémoire morte ou ROM) allant de 64 K-octets pour le PC junior à 256 K-octets pour les PC-XT, un bus de données de 8 bits, un bus d'adresses de 20 bits, un lecteur de disquettes de 360 K-octets, un BIOS (Basic Input Ouput System) et un interpréteur BASIC (Beginner's All-purpose Symbolic Instruction Code), tous deux, résidants en mémoire morte et un système d'exploitation MS-DOS (Microsoft Disk Operating System), fourni sur un disque souple. Ce chapitre est consacré à la description succincte des principaux composants du micro-ordinateur IBM PC-XT.

II Circuits de la carte mère

La carte mère est le cœur de l'ordinateur. Elle se trouve généralement en bas du boîtier de l'IBM PC ou bien montée contre le côté dans le boîtier tour. La fonction essentielle de la carte mère est d'assurer harmonieusement la liaison entre tous les éléments qu'elle contient. Chaque élément assure une tâche bien précise. En général, nous y trouvons ce qui suit:

II.1 Microprocesseur

C'est l'organe principal du micro-ordinateur. Il exécute les logiciels (programmes) et envoie des ordres à son environnement. Les performances globales d'un système dépendent essentiellement du microprocesseur utilisé. Nous consacrerons plusieurs chapitres à l'architecture et à la programmation du microprocesseur 8086 d'Intel.

II.2 Coprocesseur mathématique

Le coprocesseur mathématique est spécialisé dans le traitement des nombres à virgule flottante. Il permet donc d'accélérer les opérations arithmétiques en virgules flottantes, les fonctions trigonométriques et logarithmiques. Notons qu'il ne nous est pas possible d'étudier le coprocesseur mathématique du 8086 (architecture et programmation), en l'occurrence le 8087. Néanmoins, nous consacrerons une série de travaux dirigés à l'étude des différentes représentations des nombres dans le 8087.

II.3 Buffer de données bidirectionnel 8286/8287

Ce circuit permet d'amplifier, et le cas échéant d'isoler les signaux de données échangés entre le microprocesseur et les mémoires ou entre le microprocesseur et les périphériques d'entrées et sorties. C'est très souvent le circuit équivalent TTL 74/245 qui le remplace sur les carte-mères courantes.

II.4 Latch d'adresses 8282

Ce circuit assure le verrouillage de l'adresse et, par une exploitation judicieuse, le démultiplexage des bus d'adresses et données. C'est très souvent le circuit équivalent TTL 74/373 qui le remplace sur les carte-mères courantes.

II.5 Contrôleur de bus 8288

Ce circuit prend à sa charge les signaux de contrôle lors de l'utilisation du microprocesseur en mode maximum. Il assure l'arbitrage de l'utilisation du bus (les bus d'adresses de données et de commandes) dans le cas des configurations à plusieurs maîtres.

II.6 Circuit d'horloge 8284

C'est ce circuit qui génère l'impulsion d'horloge du microprocesseur 8086. Outre cette fonction, ce circuit joue un rôle prépondérant dans le RESET et la gestion des temps d'attente, comme il sera montré plus loin.

II.7 Contrôleur d'accès direct mémoire DMAC 8237

Le concept d'accès direct mémoire désigne la possibilité de transférer des données entre la mémoire et un périphérique, ou entre deux adresses différentes de la mémoire, sans transiter par les registres internes du microprocesseur. Ceci octroie au transfert des données une augmentation de vitesse considérable. Cette fonctionnalité est rendu possible grâce au concours de circuit contrôleurs DMA, tel

le 8237 d'Intel qui est constitué de quatre canaux identiques et indépendants, numérotés de 0 à 3. Ils sont répartis de la manière donnée par le Tableau 1.

Tableau 1 Répartition des canaux du DMAC 8237 sur la carte d'un PC-XT

Canaux DMAC	Fonction sur la carte d'un PC-XT
0	Utilisé pour l'indispensable opération cyclique de rafraîchissement de la mémoire dynamique.
1	Laissé libre pour une éventuelle application utilisateur.
2	Affecté aux transferts de données entre la mémoire RAM et les lecteurs de disquettes.
3	Affecté aux transfertx de données entre la mémoire RAM et le lecteur de disque dur.

II.8 Contrôleur d'interruptions programmable PIC 8259

Ce circuit sert à l'identification du périphérique qui sollicite une interruption matérielle du microprocesseur. En cas de demandes simultanées, le 8259 peut arbitrer jusqu'à huit niveaux de priorité, qui sont numérotés par ordre décroissant de IRQ_0, IRQ_1, IRQ_2 ..., IRQ_7. Les périphériques reliés aux entrées du PIC 8259 et leurs fonctions respectives dans un PC-XT sont résumés dans le Tableau 2.

Tableau 2 Périphériques reliés aux entrées du PIC 8259 d'un PC-XT

Broche IRQ_i i=0,1, 2, ... 7	Périphérique relié à IRQ_i
IRQ_O	Temporisateur-Compteur 8253 (Horloge)
IRQ_1	Clavier
IRQ_2	Libre
IRQ_3	Port série 2 (COM_1)
IRQ_4	Port série 1 (COM_2)
IRQ_5	Disque dur
IRQ_7	Imprimante (LPT_1)

Nous reviendrons plus en détail sur le fonctionnement de ce circuit dans le chapitre consacré aux interruptions.

II.9 Temporisateur-compteur programmable CTC 8253

Ce circuit spécialité rassemble en même temps les caractéristiques des compteurs et des générateurs de signaux, alliées à une grande souplesse d'utilisation du fait de sa commande par logiciel. Il possède trois canaux distincts, numérotés de 0 à 2, pouvant être utilisés séparément en générateur d'impulsions à fréquence programmable, en monostable universel, en compteurs

d'événements, ou en compteur d'intervalle. Ils sont répartis de la manière donnée par le Tableau 3.

Tableau 3 Répartition des canaux du CTC 8253 sur la carte d'un PC-XT

Canaux CTC	Fonction sur la carte d'un PC-XT
0	Utilisé pour générer, toutes les 1/18.2^{ème} de seconde, l'interruption cyclique de l'horloge système via le PIC.
1	Affecté à déclencher périodiquement au niveau du DMAC le cycle de rafraîchissement de la RAM dynamique.
2	Donne la tonalité pour la génération des sons au niveau du haut-parleur.

II.10 Interface parallèle de périphérique PPI 8255

Ce circuit fait l'objet d'un chapitre consacré aux périphériques à transfert de données en parallèle. Nous nous limiterons dans cette partie introductive à indiquer sont utilisation sur un IBM PC-XT en tant qu'interface de communication avec le clavier en premier lieu, et en tant que tampon de lecture des indicateurs de configuration en second lieu. En outre, il assure l'activation ou l'inhibition de la ligne de connexion entre le temporisateur-compteur 8253 et le haut-parleur.

II.11 Mémoire centrale

Les PC-XT (Extended Technology), à base du 8086, pouvaient gérer jusqu'à 1 Mo mémoire, réparti selon le Tableau 4. Le MS-DOS (Micro Soft Disk Operating System) utilisait 640 Ko de mémoire conventionnelle (Base Memory) uniquement. Les 384 Ko de mémoire supérieure LIM (Lotus Intel Motorola en circuits intégrés) ou EMS (Expanded Memory System) étaient utilisés par la mémoire écran, le BIOS (Basic Input Output System), le BIOS du disque dur et la carte graphique EGA (Enhanced Graphic Adapter).

Les PC-AT (Advanced Technology), à partir du microprocesseur 80286 gèrent au minimum 16 Mo mémoire. La zone supérieure à 1 Mo est appelée mémoire étendue ou XMS (Extended Memory System), car elle correspond aux cases mémoire accessibles grâce au bus d'adresses. La mémoire EMS en circuits intégrés a, quant à elle, disparue.

II.11.1 Mémoire RAM dynamique

Le PC est sorti avec une mémoire vive de 64 K-octets en 1981 sur un total de 640 K-octets atteint ultérieurement. La RAM dynamique contient les programmes et les données en cours d'exécution et une partie du système d'exploitation.

II.11.2 Mémoire morte ROM

Elle contient le BIOS: fonctions primitives nécessaires au système d'exploitation. C'est aussi le support physique du programme qui assure la prise en charge des opérations de démarrage du système, y compris la vérification de la configuration matérielle et le diagnostique.

II.12 Indicateurs de configuration matérielle

Il s'agit d'un ensemble de huit interrupteurs dont la position représente une information codée concernant le nombre d'unités de disquettes présentes sur la machine, la présence du coprocesseur arithmétique 8087, la quantité de RAM et le type de carte vidéo. Le système d'exploitation fait une lecture automatique de ces indicateurs lors de la mise sous tension.

Très rapidement, ces interrupteurs ont été supplantés par la mémoire MC6818 ou RTC/CMOS RAM (Real Time Clock/Static Complementary Metal Oxyde Semi-Conductor), circuit qui réunit sur la même puce un calendrier en temps réel et une RAM statique pour la conservation des paramètres de configuration matérielle. Le programme "SETUP" d'exploitation de la RTC/CMOS se trouve en ROM-BIOS.

Tableau 4 Organisation de la mémoire d'un PC

Bloc	Adresse	Contenu
15	F000 : 0000 – F000 : FFFF	BIOS
14	E000 : 0000 – E000 : FFFF	Libre pour insertion d'une ROM
13	D000 : 0000 – D000 : FFFF	Libre pour insertion d'une ROM
12	C000 : 0000 – C000 : FFFF	ROM BIOS supplémentaire
11	B000 : 0000 – B000 : FFFF	RAM d'écran
10	A000 : 0000 – A000 : FFFF	RAM d'écran EGA / VGA
9	9000 : 0000 – 9000 : FFFF	RAM de 576 K octets à 640 K octets
8	8000 : 0000 – 8000 : FFFF	RAM de 512 K octets à 576 K octets
7	7000 : 0000 – 7000 : FFFF	RAM de 448 K octets à 512 K octets
6	6000 : 0000 – 6000 : FFFF	RAM de 384 K octets à 448 K octets
5	5000 : 0000 – 5000 : FFFF	RAM de 320 K octets à 384 K octets
4	4000 : 0000 – 4000 : FFFF	RAM de 256 K octets à 320 K octets
3	3000 : 0000 – 3000 : FFFF	RAM de 192 K octets à 256 K octets
2	2000 : 0000 – 2000 : FFFF	RAM de 128 K octets à 192 K octets
1	1000 : 0000 – 1000 : FFFF	RAM de 64 K octets à 128 K octets
0	0000 : 0000 – 0000 : FFFF	RAM de 0 K octets à 64 K octets

III Cartes d'extensions

Les cartes d'extension jouent un rôle important dans la gestion des périphériques. Dans un contexte PC-XT, les adresses de leurs ports d'entrées et sorties sont résumées dans le Tableau 5. Dans ce qui suit, nous les décrivons laconiquement.

III.1 Carte graphique

Son rôle consiste à lire les données envoyées sous forme digitale par le processeur central, à les convertir en signaux analogiques, et à les transmettre par un câble au moniteur qui les traduira en image. Son élément central est le contrôleur d'écran CRTC MC 6845 (Cathod Ray Tube Controller) circuit qui permet les deux modes d'utilisation de l'écran, le mode texte et le mode graphique.

III.2 Contrôleur de lecteurs de disquettes FDC 8272

Le 8272 d'Intel est un contrôleur de disquettes (FDC, Floppy Disk Controller) en technologie LSI (Large Scale Integration, pouvant prendre entre 100 et 5000 circuits élémentaires) contenant les circuits et les fonctions de contrôle nécessaires pour réaliser l'interface entre un processeur et quatre lecteurs de disquettes. IL est capable de supporter soit le format simple densité (FM, Frequency Modulation) d'IBM 3740 (Data Entry System), ou le format double densité (MFM, Modified Frequency Modulation) d'IBM système 34 (minicomputer d'IBM de 1978 à 1983) avec enregistrement double faces. Les 8272 donnes les signaux qui simplifient la conception d'une boucle à verrouillage de phase (PLL, Phase Locked Loop) externe et le circuit de précompensation en écriture. La PLL a pour utilité l'asservissement de la fréquence des signaux dans le cas de perturbation de la vitesse de rotation du moteur du lecteur de disquettes. La première disquette d'IBM (1971) était de capacité 81.6 K Octets (32 pistes). Six ans plus tard, une disquette de capacité 1.2 M Octets (154 pistes) a vu le jour.

La précompensation, quant à elle désigne une mesure introduite afin de pallier à la distorsion magnétique qui affecte les signaux enregistrés sur les pistes les plus internes de la disquette.

III.3 Port série et le port parallèle

C'est souvent sur la même carte que sont présents l'UART 8250, circuit d'interface avec le port sériel RS-232, et l'interface parallèle Centronics dont l'utilisation principale sur l'IBM-PC est la connexion avec l'imprimante. Ces deux standards d'interfaces feront l'objet de chapitres séparés.

III.4 Clavier

Le clavier fournit les données en série. Il possède son propre microprocesseur 8042 qui permet de diagnostiquer l'état du clavier à la mise sous tension, de générer le code des touches du clavier (scan code) et de le transmettre au tampon mémoire de la carte mère qui peut contenir un maximum de 14 caractères.

Tableau 5 Standardisation des adresses des ports d'entrées et sorties

Circuit	PC-XT	PC-AT
Contrôleur DMA n° 1 (8237A-5)	000 – 00F	000 – 01F
Contrôleur d'interruption n° 1 PIC (8259A)	020 – 021	020 – 03F
Temporisateur CTC (8253)	040 – 043	040 – 05F
Interface parallèle de périphérique PPI (8255)	060 – 063	Néant
Clavier (8042)	Néant	060 – 06F
Horloge temps réel RTC/CMOS (MC6818)	Néant	070 – 07F
Registre de page DMA	080 – 083	080 – 09F
Contrôleur d'interruption n° 2 PIC (8259A)	Néant	0A0 – 0BF
Contrôleur DMA n° 2 (8237A-5)	Néant	0C0 – 0DF
Coprocesseur arithmétique	Néant	0F0 – 0F1
Coprocesseur arithmétique	Néant	0F8 – 0FF
Contrôleur de disque dur	032 – 32F	1F0 – 1F8
Manette de jeux	200 – 20F	200 – 207
Unité d'extension	210 – 217	Néant
Imprimante parallèle n° 2	Néant	278 – 27F
Interface série n° 2	2F8 – 2FF	2F8 – 2FF
Carte prototype	300 – 31F	300 – 31F
Carte réseau	Néant	360 – 36F
Imprimante parallèle n° 1	378 – 37F	378 – 37F
Carte écran monochrome	3B0 – 3BF	3B0 – 3BF
Carte vidéo couleur graphique	0D0 – 0DF	0D0 – 3DF
Contrôleur de disquettes	3F0 – 3F7	3F0 – 3F7
Interface série n° 1	3F8 – 3FF	3F8 – 3FF

III.5 Connecteurs du bus ou slots d'extensions

Ces connecteurs ou slots sont mis à la disposition de l'utilisateur pour d'éventuels ajouts de périphériques. Ils représentent un support visible du bus système, avec le bus d'adresses et le bus de données démultiplexés, ainsi que toutes les lignes du bus de contrôle séparées. Sur les IBM PC-XT nous avons des bus sur le standard ISA (Industry Standard Architecture). Le Tableau 6 résume la fonction de

chacun des signaux constituant le slot ou connecteur à 62 broches de ce bus d'extension pour les PC-XT.

Tableau 6 Connecteur d'extension à 62 broches

Broche	Signal	E/S	Fonction
A_1	$\overline{\text{I/O CH CK}}$	E	I/O Channel Check
A_2 à A_9	D_7 à D_0	E/S	Data Bus
A_{10}	$\overline{\text{I/O CH RDY}}$	E	I/O Channel Ready
A_{11}	AEN	S	Address Enable (for DMA)
A_{12} à A_{31}	A_{19} à A_0	S	Address Bus
B_1	GND	E	Ground
B_2	RESET DRV	S	Reset Driver
B_3	+5V	S	Power
B_4	IRQ_2	E	Interrupt Request N° 2
B_5	-5V	E	Power
B_6	DRQ_2	E	DMA Request N° 2
B_7	-12V	S	Power
B_8	Reserved	S	Reserved
B_9	+12V	S	Power
B_{10}	GND	-	Ground
B_{11}	$\overline{\text{MEMW}}$	S	Memory Write
B_{12}	$\overline{\text{MEMR}}$	S	Memory Read
B_{13}	$\overline{\text{IOW}}$	S	I/O Write
B_{14}	$\overline{\text{IOR}}$	S	I/O Read
B_{15}	$DACK_3$	S	DMA Acknowledge N° 3
B_{16}	DRQ_3	E	DMA Request N° 3
B_{17}	$DACK_1$	S	DMA Acknowledge N° 1
B_{18}	DRQ_1	E	DMA Request N° 1
B_{19}	$DACK_0$	S	DMA Acknowledge N° 0
B_{20}	CLOCK	S	4.77 MHz Clock
B_{21}	IRQ_7	E	Interrupt Request N° 7
B_{22}	IRQ_6	E	Interrupt Request N° 6
B_{23}	IRQ_5	E	Interrupt Request N° 5
B_{24}	IRQ_4	E	Interrupt Request N° 4
B_{25}	IRQ_3	E	Interrupt Request N° 3
B_{26}	$DACK_2$	S	DMA Acknowledge N° 2
B_{27}	T / C	S	Terminal Count (for DMA)
B_{28}	ALE	S	Address Latch Enable
B_{29}	+5V	S	Power
B_{30}	OSC	S	Oscillator
B_{31}	GND	-	Ground

Afin d'assurer une compatibilité ascendante, l'IBM PC-AT est équipé du même connecteur ou slot d'extension, et ce pour garantir la connectivité des cartes à 8 bits conçues pour les PC-XT. Il est complété par un second connecteur de 36 broches, Tableau 7, incluant 8 bits de données supplémentaires (portant

ainsi le nombre de fils de données à 16), 4 bits d'adresses supplémentaires (portant ainsi le nombre de fils d'adresses à 24) et certaines autres broches de contrôle propres aux PC-AT.

Tableau 7 Connecteur d'extension à 36 broches

Broche	Signal	E/S	Fonction
C_1	$\overline{SBHE}$	E/S	System Bus High Enable
C_2 à C_8	LA_{23} à LA_{17}	E/S	Address Bus
C_9	$\overline{MEMR}$	E/S	Memory Read
C_{19}	$\overline{MEMW}$	E/S	Memory Write
C_{11} à C_{18}	SD_8 à SD_{15}	E/S	Data Bus
D_1	MEM-CS16	E	MEM CS16
D_2	$\overline{I/O\text{-}CS16}$	E	I/O_CS16
D_3	IRQ_{10}	E	Interrupt Request N°10
D_4	IRQ_{11}	E	Interrupt Request N°11
D_5	IRQ_{12}	E	Interrupt Request N°12
D_6	IRQ_{13}	E	Interrupt Request N°13
D_7	IRQ_{14}	E	Interrupt Request N°14
D_8	$DACK_0$	S	DMA Acknowledge N°0
D_9	DRQ_0	E	DMA Request N°0
D_{10}	$DACK_5$	S	DMA Acknowledge N°5
D_{11}	DRQ_5	E	DMA Request N°5
D_{12}	$DACK_6$	S	DMA Acknowledge N°6
D_{13}	DRQ_6	E	DMA Request N°6
D_{14}	$DACK_7$	S	DMA Acknowledge N°7
D_{15}	DRQ_7	E	DMA Request N°7
D_{16}	+5V	-	Power
D_{17}	$\overline{MASTER}$	E	Master
D_{18}	GND	-	Ground

IV Alimentation

L'alimentation transforme les 220 Volts du secteur en 5 Volts et 12 Volts. Les 5 Volts sont destinés aux circuits de l'ordinateur, alors que les 12 Volts servent à alimenter les moteurs de lecteurs de disquettes et de disques durs. La puissance de sortie de l'alimentation est fonction du nombre de périphériques connectés.

Ainsi, les premiers PC (jusqu'au 8086) disposaient d'une alimentation d'une puissance de 65 Watts permettant d'alimenter l'ensemble des circuits de la carte mère et deux lecteurs de disquettes. Les PC-XT (à partir du 80286) étaient dotés d'une alimentation de type AT d'une puissance de 133 Watts pouvant alimenter, en outre, un disque dur. Avec l'avènement du Pentium II, les PC-AT, ont bénéficié, quant à eux, d'une alimentation de type ATX (Advanced Technology Extended) d'une puissance allant de 220 Volts à 230 Volts. Cette dernière servait aussi à alimenter l'écran et toutes les cartes d'extension.

CHAPITRE 3

EVOLUTION DES MICRO-ORDINATEURS PC

I Introduction

Lorsque Gordon Moore, co-fondateur d'Intel, avait prédit en 1965 que la densité en transistors des circuits intégrés allait doubler tous les deux ans, savait-il que sa prédiction allait se confirmer avec une précision inouïe jusqu'à aujourd'hui? Nous constaterons qu'entre temps la vitesse des calculateurs a augmenté à peu prés au même rythme que l'accroissement du taux d'intégration.

En 1971, Intel surprend toute la planète en lançant le premier microprocesseur 4 bits. Il s'agissait du 4004 (2300 transistors et 60000 instructions par seconde). Depuis, on ne le répétera jamais assez, Intel n'arrête pas de créer l'événement en annonçant régulièrement de nouveaux produits toujours plus performants et en honorant ses annonces.

Pour une technologie en perpétuel mouvement nous ne pouvons nous donner la prétention de rendre fait de l'état de l'art, et notre exposé se veut avant tout didactique. Notre choix s'est donc délibérément porté sur le microprocesseur 8086 d'Intel. En effet celui-ci, au même titre que les machines IBM PC-XT qui en sont équipées, réunit dans une large mesure toute la théorie moderne des machines universelles telle qu'elle à été formulée par Johannes Von Neumann (1945).

Par ailleurs, de part sa compatibilité ascendante, il constituera un outil d'apprentissage idéal où l'application pratique peut se réaliser sur n'importe quelle machine de fabrication ultérieure dans un parc très largement doté. S'il nous sera reproché à juste titre que l'exposé reste incomplet non pas relativement au phénomène de mode, mais du fait des limites théoriques imposées par la structure même de l'IBM PC-XT (absence du concept de la mémoire cache, des Vidéo RAM et shadowing, interfaces multimédia, ...) nous renvoyons à ce sujet le lecteur intéressé à la littérature spécialisée. Néanmoins, pour orienter les curieux et assouvir la faim des insatiables, nous présentons, dans ce qui suit, un aperçu succinct de l'évolution vertigineuse de la technologie des micro-ordinateurs de la famille Intel et des différents chipset, bus, interfaces, mémoires et disques durs actuels.

II Bus

Le bus ISA reste l'un des standards les plus répandus en matière de bus. Ce bus est apparu sous sa première version en 1984 avec le microordinateur IBM PC-AT. Le microprocesseur était alors le 80286 fonctionnant à 8 MHz. Les bus, comme tous les autres composants ont également suivi une évolution historique, passant ainsi du bus ISA du PC-AT aux bus EISA, MCA, VLB, PCI, AGP, USB, PCI, Express, etc.

Actuellement, le PCI Express ou PCIe dans sa deuxième version est suffisamment rapide pour remplacer non seulement le PCI classique mais aussi l'AGP, un port rapide exclusivement utilisé pour les cartes graphiques. Dans un avenir proche, il serait même envisagé d'y connecter des périphériques externes. Contrairement au PCI qui est relié au southbridge de la carte mère, le PCIe est relié au northbridge. De plus, alors que le PCI 2.3 (2002) utilise un unique bus de 64 bits bidirectionnel alterné (half duplex) pour tous les périphériques, le PCIe utilise une interface série de 1 bit, à base de lignes bidirectionnelles. Nous pouvons, par exemple, parler d'une carte mère possédant 20 lignes PCIe. Une ligne permet des échanges par ligne et par direction à 250 Mo/s pour le PCIe première génération (versions 1.0 et 1.1). Sur les micro-ordinateurs, nous distinguons des ports PCIe x1, x2, x4, x8, x16 et x32 pour différencier les ports en fonction du nombres de connecteurs de lignes dont il dispose i. e. 1, 2, 4, 8, 16 et 32 lignes , respectivement. De ce fait, un port x32 permet d'atteindre un débit de 8 Go/s, soit 4 fois le débit des ports AGP. 2007 a vu la naissance de la deuxième génération du PCIe. Cette version (PCIe 2.0 et PCIe 2.1) permet d'atteindre un débit de 500 Mo/s par ligne et par sens. Ce qui représente le double du débit de la première génération. Enfin, la troisième version (PCIe 3.0) devrait être commercialisée dès le deuxième semestre 2010 pour atteindre un débit de 1Go/ par ligne et par sens.

II.1 Anciens bus

Tableau 1 Caractéristiques des anciens bus

Type	Largeur bits	Fréquence MHz	Débit Mo/s	Nombre max de connecteurs et configuration des cartes d'extension
ISA	8-16	8-12.5	8	8 Configuration matérielle
EISA	8-16-32	8-12.5-20	50	18 Configuration logicielle
MCA	16-32	8-12.5	50	8 Configuration logicielle
VLB	16-32	33	132	3 Configuration Plug & Play
PCI 1.0	16-32	33	132	10 Configuration Plug & Play

ISA : Industry Standard Architecture.
EISA : Extended ISA.
MCA : Micro Channel Architecture.
VLB : VESA (Video Electronic Standard Association) Local bus.
PCI : Peripheral Component Interconnect.

II.2 Nouveaux bus

Tableau 2 Caractéristiques des nouveaux bus

Type	Débit max Mo/s	Max de périphs.	Utilisation
PCI 2.0 64 bits	132	10	Compatible EISA, Plug & Play (pont mémoire PCI, ...)
PCI-X 2.0 64 bits	4264	10	Compatible EISA, Plug & Play (pont mémoire PCI, ...)
PCMCIA	150	1	Cartes à mémoires flash, modems & DD (portables)
DMA/33/66/100/133	33/66/100/133	4	Disque Dur IDE
UDMA/33/66/100/133	33/66/100/133	4	Disques Durs ATA et UATA
USB 1.0 et 1.1	1.5 Haute Vitesse	127	Périphériques Plug & Play à faible débit
USB 2.0	60	-	Périphériques Plug & Play à haut débit
USB 3.0	400	-	Périphériques Plug & Play à très haut débit (Disques durs)
Fire Wire IEEE 1394	50-400	63	Périphériques rapides (Vidéo numérique)
AGP 1X/2X/4X/8X	266/533/1000/2000	1	Cartes graphiques ultra rapides
PCI Express PCIe 1.0 et 1.1 (2004)	*250	**	Cartes graphiques, cartes mémoires, Compact flash, etc.
PCI Express PCIe 2.0 et 2.1 (2007)	*500	**	Cartes graphiques, cartes mémoires, Compact flash, etc.
PCI Express PCIe 3.0 (2010)	*1000	**	Cartes graphiques, cartes mémoires, Compact flash, etc.

* Débit par ligne et par sens.

** Dépend du slot existant, i.e. x1, x2, x4, x8, x16 ou x32 et de la carte d'extension utilisée.

PCI-X : PCI-SIG (Special Interests Group).
PCMCIA : Personal Computer Memory Card International Association.
UDMA : Ultra Direct Memory Access.
USB : Universal Serial Bus.
IEEE : Institute of Electrical & Electronics Engineering.
AGP : Accelerated Graphic Port.

III Evolution des iAPX86

Avec l'évolution de la gravure, nous ne pouvions que penser à de nouvelles technologies telles que celles du superscalaire, superpipeline, MMX (Multi Media eXtension), MCPU (Multi Central Processing Unit) et MFPU (Multi Floatting Processing Unit), de mémoires caches niveaux 1, 2 et 3 intégrées,...

De 2000 à 2004, Pentium, le processeur emblématique de la famille Intel est né avec ses différentes versions II, III et IV. C'est un processeur qui fonctionne en technologie EPIC (Explicitly Parallel Computing). En effet, alors que le pentium II dispose de 2 CPU accompagnés de 8 registres et d'une unité FPU également accompagnées de 8 registres, le Pentium III à 1GHz dispose de plein de CPU associés à 128 registres et plusieurs FPU utilisant également 128 registres. Enfin, les Pentium IV et Prescott sont montés en fréquence (2.8 GHz et 4 GHz) atteignant ainsi un taux de gravure de 0,09µm.

Grâce à la technologie multi core (multi cœur), le dernier lustre a également vu la naissance de processeurs 64 bits très puissants tels que le Core 2 Duo, Core 2 Quad, Core i3, Core i5, Core i7 et le Sandy Bridge dont la technologie est SSE5 (Steraming SIMD Single Instruction on Multiple Data Extension version, par opposition au fonctionnement du jeu d'instructions traditionnel, i.e., SISD: Single Instruction Single Data). Ces processeurs à très faible consommation d'énergie sont passés à des modes de fonctionnement très complexes tel que celui du 'out-of-order' qui exulte les plus grands acteurs de l'informatique. La loi Moore ne veut pas s'arrêter en si bon chemin. Elle compte aller jusqu'à 2014 pour atteindre une gravure de 6nm i.e. taux d'intégration au-delà de 1 G transistors et des performances excédant 1 GIPS, voire même au delà. Le Tableau 3 synthétise la chronologie des différentes architectures de microprocesseurs et coprocesseurs de la firme Intel.

Tableau 3 Différentes architectures des microprocesseurs Intel

Architecture	Microprocesseur/Coprocesseur
Pre-x86	4004 - 4040 - 8008 - 8080 - 8085
x86-16	8086 - 8088 - 80186 - 80188 - 80286
X87 (ext. FPU)	8087 (8/16) - 80287 (16) - 80387 - 80487 (32)
X86-32/IA-32	80386 - 80486 (SX-DX2-DX4-SL) - Pentium (Original - Pro - II - III - 4 - M) - Core - Celeron M - Celeron D
X86-64/EM64	Pentium 4 - Pentium D - Pentium Extreme - Celeron D
Core-based Microarchitecture	Core Solo - Core Duo - Core 2 Duo - Core 2 Quad - Core i3 - Core i5 - Core i7

FPU : Floatting Point Unit.

III.1 Microprocesseurs

Trans.	1975	1980	1985	1990	1995	2000	2005	2010	MIPS
1000 M								Core i Core 2	150000 50000
100 M						Prescott P IV			10000
10 M					P Pro P MMX	P III PII			1000 500
1 M			80486	Pentium					100 25
300K 140 K		80286	80386						1
30 K	8086								0.5
6.K 2.3 K	8085 4004								0.4 0.06

Fig. 1 Evolution des microprocesseurs Intel

Le Tableau 4 illustre quelques exemples du parc microprocesseurs Intel, partant de la plus de la deuxième génération (80286) à la onzième génération (Sandy Bridge).

Tableau 4 Caractéristiques de quelques microprocesseurs Intel (Ancienne et nouvelle générations)

Type	An. Fab.	Fréq. MHz	Bus @ (bits)	Bus Data (bits)	Cache L_1 (KO)	Cache L_2 (KO)	Alim. (V)
286	1982	8-20	24	16	-	-	5
386DX	1985	16-33	32	32	-	-	5
486DX	1989	33-100	32	32	8	Ext.	5 - 3.3
Pent.	1993	60-200	32	2 x 32	2 x 8	Ext.	3.3 - 2.9
P MMX	1995	200-266	32	2 x 32	2 x 16	Ext.	2.8
P Pro.	1995	150-200	32	2 x 32	2 x 8	256	3.1 - 2.9
P II	1997	266-450	32	2 x 32	2 x 8	256	2.8
P III	1999	450-1000	32	32	2 x 16	512	2.8
P 4	2002	1300-2800	32	32	2 x 16	512	2.8
Prescott	2004	3400-4000	32	32	16	2048	2.8
***Core 2**	2006	1860-3200	32	64	64	4096	1 - 1.15
***Core i**	2008	2600-3600	48	64	64	512	1.2 -1.3
Sandy Bridge	2010	Technologie SSE5/AVX : Conception hautement modulaire Caractéristiques inconnues à jour					

*Caractéristiques par core.
Versions de 2 à 6 cores et mémoire cache L_3 de 4 MO à 12 MO.

III.2 Chipsets ou jeux de composants

Le chipset joue un rôle prépondérant dans le choix d'une carte mère. Il gère la coopération entre le microprocesseur, les bus, la mémoire et les ports d'extensions. Les premiers chipsets commercialisés par Intel furent utilisés avec les microprocesseurs 80486 tels que le 420 TX (1992) et 420 EX (1994). Ils possédaient des FSB travaillant à des fréquences de 33 MHz et 50 MHz, respectivement, et gérant des capacités mémoire allant jusqu'à 128 MO. Actuellement, ces chipsets permettent de piloter des bus systèmes reliant microprocesseur et mémoires à des fréquences de plus de 1.6 GHz. Ainsi, comme le montre les Tableaux 5-7, ils sont capables de gérer les nouveaux bus PCI Express versions 2.0 et 2.1. Par ailleurs, Ils ont une technologie SLI qui permet de coupler deux cartes graphiques identiques en utilisant une carte mère compatible PCIe et intègrent des pare-feu de type Active Armor qui libère le processeur d'environ 80% des calculs concernant la sécurité. Enfin, notons, Figure. 2, que du point de vue connective, le chipset se distingue par une architecture à deux chips, en l'occurrence le Northbridge (bus processeur) et le Southbridge (bus d'entrées et sorties). Alors que le premier est exclusivement conçu pour être connecté à la mémoire et aux cartes graphiques compatibles AGP/PCIe, le second, quant à lui est exclusivement conçu pour être connecté au power management (ACPI), aux bus PCI/PCIe/USB/LAN, aux systèmes audio (HDA), aux disques durs SATA, Bus, etc.

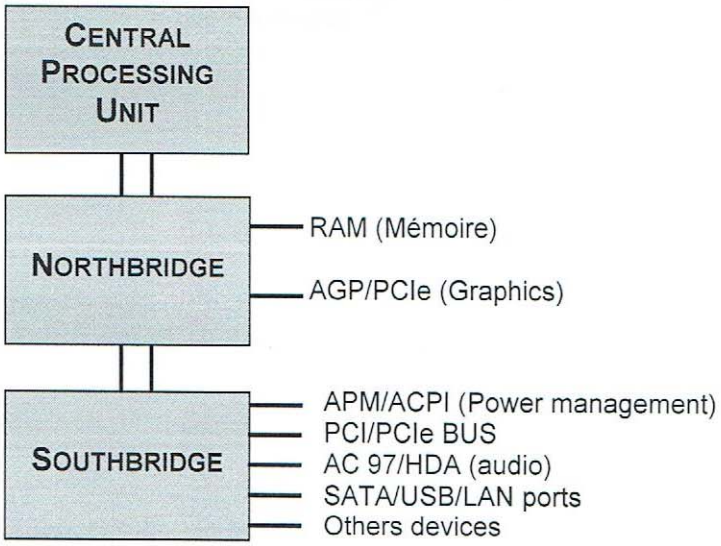


Fig. 2 Schématisation de l'architecture d'un chipset

Tableau 5 Exemples de chipsets pour microprocesseurs en technologies Core 2 et Core i7 (Nehalem)

Chipset	South Bridge	µp	FSB (MHz)	Mém. (MHz)	Max Mém.	Graphics
82975x MCH (2005)	ICH7R	Core 2 Duo	800/1066	DDR2 800	8 Go	1 PCIe 2.0 x16 2 PCIe x8
P31 (2007)	ICH7	Dual Core Core2 Duo Core2 Quad	800/ 1066	DDR2 667/800	4 Go	1 PCIe x16 1 PCIe x4
G41 (2008)	ICH7	Core2 Duo Core2 Quad	800/ /1066 /1333	DDR3 800/1066 DDR2 667/800	8 Go	1 PCIe 2.0 x16 1 Integrated GMA X4500 graphics
B43 (2009)	ICH10D	Core2Duo Core2 Quad	800/ 1066/ 1333	DDR3 800/1066 DDR2 667/800	16 Go	1 PCIe 2.0 x16 1 Integrated GMAX4500 graphics
X58 IOH Express Chipset (2009)	ICH10 et ICH10R	Core i7	QPI (25.6 GB/s)	Intel Turbo Memory	-	1 PCIe 2.0 Graphics

Tableau 6 Exemples de chipsets (Southbridge)

Chipset	ATA	SATA	RAID Level	USB
ICH7	UDMA 100/66/33	3.0 Gbits/s, 4 ports	No	2.0, 8 ports
ICH10D	No	3.0 Gbits/s, 6	0, 1, 5, 10 and Matrix	2., 12

		ports	RAID	ports

Tableau 7 Exemples de chipsets pour microprocesseurs en technologie Core i7

Chipset	Broches	Interface de Bus	Débit Bus	PCIe	SATA	USB	FDI
Q57 (2010)	LGA 1156	DMI	2 Go/s	8 PCIe 2.0	3.0 Gbits/s, 6 ports	2.0, 14 ports	Yes
P67 (2010)	LGA 1155	DMI 2.0	5 Go/s	8 PCIe 2.0	6 Gbits/s, 2 ports et 3 Gbits/s, 4 ports	2.0, 14 ports	No

ICH : I/O Controller Hub.
MCH : Memory Controller Hub
APM : Advanced Power Management Advanced.
ACPI : Configuration and Power Interface.
AC 97/HDA : Audio Coded '97 and High Definition Audio.
LAN : Local Area Network.
GMA : Graphic Media Accelerator.
DMI : Desktop Management Interface.
FDI : Flash Data Integrator.
FSB/QPI : Front Side Bus/Quick Path Interconnect (Core i7)
SLI : Scalable Link Interface.
LGA : Land Grid Array

IV Mémoires

Dans les micro-ordinateurs, la mémoire vive se présente généralement sous la forme de barrettes SIMM (Single Inline Memory Module) de 8 ou 32 bits ou bien encore 9 ou 36 bits si la barrette doit gérer un bit de parité. Avec l'avènement du Pentium, ces barrettes étaient montées par paires formant des bancs de mémoire dits 'banks memory'. Le format SIMM a, ensuite été supplanté par des barrettes au format DIMM (Dual Inline Memory Module) qui pouvait gérer des accès à la mémoire par blocs de 64 ou 72 bits quand les bits de contrôle sont comptabilisés. Ainsi, une barrette DIMM pouvait remplacer deux barrettes SIMM. C'était le format utilisé avec les SDRAM (Sychronous DRAM). Précisons que les DRAM et EDO RAM ont également existés dans des versions plus performantes et plus chères dites ECC (Error Correcting Code) qui intégraient des bits supplémentaires à gérer les codes de contrôle des erreurs. La mémoire EDO qui a connu un bel essor a été détrônée par la SDRAM. Avant l'apparition de la SDRAM, les DRAM étaient asynchrones. Cela signifie qu'il fallait attendre un temps donné pour obtenir une modification de l'état des entrées. Une mémoire SDRAM, quant à elle, attend un front d'horloge pour prendre en compte l'état des signaux d'entrées. Cette horloge qui est habituellement synchrone au FSB (Front Side Bus), permet de pipeliner les

instructions entrantes. Le pipelining permet de commencer une opération avant que l'opération précédente ne soit terminée. Ce qui résulte en un gain considérable de temps de lecture ou écriture mémoire.

En 2003, une nouvelle technologie de mémoire à vue le jour. La DDR-SDRAM ou simplement DDR, fournit une meilleure bande passante que l'ordinaire SDRAM. Alors que la SDRAM transfère les données sur le front montant des impulsions d'horloge, la DDR effectue les transferts sur le front montant et sur le front descendant des impulsions d'horloge (technique dite Dual Pumping). Ainsi, une carte dotée de mémoire DDR-SDRAM avec un bus mémoire cadencé à 133 MHz est équivalente en débit de données à de la mémoire SDRAM à 266 MHz. Depuis 2005, ces mémoires ont eu des difficultés à aller au-delà de 400 MHz, elles ont donc été remplacées par la DDR2-SDRAM ou DDR2. Cette dernière fonctionne selon le même principe que la DDR, mais plus simple à produire et atteignant des fréquences d'horloge plus élevées. La différence majeure entre la DDR2 et la DDR est que la fréquence du bus est maintenant égale au double de celle du groupe de cellules mémoires. A fréquence des cellules mémoires égale, la DDR2 a un débit deux fois plus élevés que celui de la DDR.

Les PC commercialisés en 2007 étaient équipés de mémoire DDR3-SDRAM ou DDR3. Depuis, ce standard défini par l'organisation de standardisation des semi-conducteurs JEDEC (Joint Electron Device Engineering Council) a progressivement remplacé la DDR2. Cette nouvelle technologie de mémoire a permis de réduire la consommation énergétique de la DDR3 de 40% par rapport à celle de la DDR2. Ceci a donc permis une baisse de tension allant de 2.5 V sur la DDR, à 1.8 V sur la DDR2 et 1.5 V sur la DDR3. Quant à la bande passante maximale par canal, elle est passée de 4.8 Go/s pour la DDR, à 9.6 Go/s pour la DDR2 et 14.4 Go/s pour la DDR3.

Enfin, notons que les mémoires DDR, DDR2 et DDR3 ne sont ni compatibles ni rétrocompatibles. Autrement dit, un module DIMM-DDR ne peut être enfiché ni dans le slot d'un module DIMM-DDR2 ni dans celui d'un module DIMM-DDR3 et inversement.

Tableau 8 Caractéristiques de quelques circuits intégrés RAM

Type	Signification	Fréq. Max (MHz)	Temps d'accès (ns)
SRAM	Static Random Access Memory	166-66	6-15
DRAM	Dynamic RAM	33-28-25	50-60-70
VRAM/WRAM	Video RAM/ Window RAM	-	-
EDO RAM	Extended Data Out DRAM	50-40-33	50-60-70
BEDO RAM	Burst EDO DRAM	66-60-50	52-60-70
SDRAM	Synchronous DRAM	150-100-133	7-10-12
RDRAM	Rambus DRAM	800-200-166	5-6-7

Tableau 9 Caractéristiques des circuits intégrés DDR

Type	An. Fab.	Signification	Fréq. Max (MHz)	Débit Max (MT/s)	Temps d'accès (ns)
DDR	2000	Double Data Rate 1 SDRAM	100-200	200-400	10-5
DDR2	2003	Double Data Rate 2 SDRAM	200-533	400-1066	10-3.75
DDR3	2007	Double Data Rate 3 SDRAM	400-800	800-1600	10-5
DDR4	2012	Double Data Rate 4 SDRAM	-	2800-3600	-

1 MT/s : 10^6 Transferts/seconde. Pour un bus de données de 8 octets et un taux de transfert de 1 MT/s, le débit serait équivalent à 8 Mo/s.

V Connecteurs ou interfaces disques durs

La fonction de pilotage des moteurs de rotation et de déplacement des têtes de lecture/enregistrement et d'interprétation des signaux électriques reçus de ces têtes pour les convertir en bits ou la réalisation de l'opération inverse afin d'enregistrer des données à un endroit particulier du disque dur, confère au contrôleur de disque le rôle de gestion des échanges de données et leur encodage entre le disque et le système.

Sur les premiers disques durs, les contrôleurs étaient indépendants de la mécanique. Le volumineux câblage a rapidement favorisé la recherche d'une solution plus compacte. De nos jours, les interfaces se trouvent accolés aux disques durs, donnant naissance aux standards SCSI et IDE. Le contrôleur SATA sous ses versions II et III, constitue le standard qui équipe le plus les micro-ordinateurs actuels. De plus, pour réduire le temps d'accès aux données, les contrôleurs de disques durs actuels intègrent le mode NCQ.

La capacité de stockage des disques durs a augmenté plus rapidement que leur temps d'accès, limité par la mécanique. Ainsi, alors que le standard en 1997 était de 2 Go pour les disques durs 3,5 pouces, 1 To (Téra octets) ont été atteints par Hitachi en 2007 et respectivement, 2 To et 3 To par Western Digital en 2009 et 2010.

Les disques durs hybrides sont des disques magnétiques classiques accompagnés d'un petit module de mémoire flash (128 Mo à 256 Mo). L'utilisation de la mémoire flash permet d'améliorer de 20% les chargements et le temps de démarrage des PC, de réduire leur consommation et d'augmenter leur durée de vie. Notons que l'ajout de mémoire vive (mémoire cache) sur le contrôleur de

disque augmente ses performances. Cependant, comme tout système de cache, cela pose un problème de cohérence de données.

Enfin, la technologie RAID qui était souvent réservée aux serveurs professionnels peut, de nos jours, équiper les PC. Celle-ci permet de stocker de données en utilisant plusieurs disques et en optimisant leur complémentarité afin d'obtenir, selon les cas, plus de sécurité et de fiabilité ou plus de performance. Le système RAID existe sous plusieurs variantes. Pour ce faire, citons, par exemple les RAID 0, 1 et 5. Ces chiffres désignent les modes de fonctionnement de la technologie RAID. De ce fait, alors que le système RAID 0 (striping) fait fonctionner plusieurs disques en une seule unité logique, le RAID 1 (mirroring) protège des erreurs dues au stockage et le RAID 5 réalise de bonnes performances tout en assurant une sécurité confortable des données i. e. combinaison des systèmes RAID 0 et RAID 1.

Tableau 10 Caractéristiques de quelques contrôleurs de disques durs

Type d'interface	Débit (Mo/s)	Mémoire Cache (Mo)	Vitesse de rotation (trs/mn)
EIDE, ATA 100-133 et UDMA 100-133	100-133	8-16	7200
USB 2.0	60	-	7200
USB 3.0	625	-	7200
eSATA	3000	-	-
SATA II	375	32-64	5400
SATA III	750	64	7200
Fire Wire 800	165	-	7200
Ultra-320/Ultra 640 SCSI	320/640	16	15000
SAS 1.0/SAS 2.0	375/750		
SSD	250	128	-

EIDE : Enhanced Integrated Drive Electronic.
SCSI : Small Computer System Interface.
ATA : Advanced Technology Attachment
SATA : Serial ATA.
eSATA : Express SATA.
SAS : Serial Attachment SCSI.
SSD : Solid State Drive or Disk
RAID : Redundant Array of Independent Disks.
NCQ : Native Command Queuing Reduction.

VI Affichage graphique

Un écran ou moniteur est le périphérique d'affichage de l'ordinateur. On distingue habituellement deux familles d'écrans. D'une part, les anciens écrans à tube cathodique (CRT, Cathod Ray Tube), équipant la majorité des ordinateurs de bureau. Il s'agit de moniteurs volumineux et lourds, possédant une consommation électrique élevée. D'autre part, les écrans plats (Dalles TFT LCD, Thin Film Transistor, Liquid Cristal Display) équipant la totalité des ordinateurs portables, les assistants personnels (PDA, Personal Digital Assitant), les appareils photos numériques, ainsi qu'un nombre de plus en plus grand d'ordinateurs de bureau. Il s'agit d'écrans peu encombrants en épaisseur, légers et possédant une faible consommation électrique.

Un moniteur quel qu'il soit est d'abord caractérisé par sa définition exprimée en nombre de pixel (picture elements). Ce nombre de points que peut contenir l'écran est généralement compris entre 640 x 480 (640 pixels en colonne, 480 pixels en ligne) et 2048 x 1536. Des résolutions supérieures sont techniquement possibles. La taille qui se calcule en mesurant la diagonale de l'écran et est exprimée en pouces (1 pouce=2,54 cm). Le pas de masque (dot pitch) qui représente la distance qui sépare deux luminophores. Plus celle-ci est petite plus l'image est précise. Ainsi un pas de masque inférieur ou égal à 0,25 mm procurera un bon confort d'utilisation, tandis que les écrans possédant des pas de masque supérieurs ou égaux à 0,28 mm sont à proscrire. Enfin, la résolution qui détermine le nombre de pixels par unité de surface ou pixels par pouce linéaire (DPI, Dots Per Inch). Une résolution de 72 dpi signifie 72 colonnes et 72 lignes de pixels sur un pouce carré ce qui donnerait donc 5184 pixels sur un pouce carré. Soit donc 0.353mm par pixel.

Un autre aspect important de l'écran est le mode graphique ou mode d'affichage des informations à l'écran, en termes de définition et de nombre de couleurs. Ainsi, il représente la capacité d'une carte graphique à gérer des détails ou celle d'un écran de les afficher. Le contrôle du moniteur s'effectue grâce à un contrôleur (carte écran, carte vidéo, etc.) De nombreux standards se sont succédés.

VI.1 Cartes et modes graphiques

En 1987, IBM a lancé le mode VGA qui émule entièrement les modes CGA et EGA, et offre des définitions supplémentaires dont 640 x 480 pixels en mode graphique WYSIWYG (What You See Is What You Get) et 720 x 400 en mode caractère. Soit 30 lignes de 80 caractères (matrice 9 x 16), en 16 couleurs.

Le mode SVGA est une extension du mode VGA apparue vers 1989 dont la définition dépend du constructeur. Le Tableau 11 donne des exemples de cartes graphiques de l'ancienne génération.

Tableau 11 Exemples de cartes graphiques

Type	Année de fabrication	Abréviation
MDA	1970	Monochrome Dispaly Adapter
CGA	1981	Color Graphic Adapter
HGA	1982	Hercule Graphic Adapter
EGA	1984	Enhanced Graphic Adapter
PGA	1985	Professional Graphic Adapter
VGA	1987	Video Graphic Array
SVGA	1989	Super VGA
XGA	1990	Extended Graphic Array
UXGA	-	Ulra Extended Graphic Array
QSXGA	-	Quad Super Extended Graphic Array
QUXGA	-	Quad Ultra Extended Graphic Array

Tableau 12 Exemples de modes graphiques

Type	Définition	Bits/pixel	Couleurs	Mémoire min (Mo)
VGA	640 x 480	4/8/16/24	16/256/64Ko/16Mo	0.25/0.5/1/1
SVGA	800 x 600	4/8/16/24	16/256/64Ko/16Mo	0.25/0.5/1/1.5
XGA	1024 x 768	4/8/16/24	16/256/64Ko/16Mo	0.25/1/1.5/2.5
SWGA	1024 x 768	4/8/16/24	16/256/64Ko/16Mo	1/1.5/2.5/4
UXGA	1600 x 1200	4/8/16/24	16/256/64Ko/16Mo	1/2/4/6
QSXGA	2560 x 2048	4/8/16/24	16/256/64Ko/16Mo	3/5.5/10.5/16
QUXGA	3200 x 2400	4/8/16/24	16/256/64Ko/16Mo	4/8/15.5/23.5

VI.2 Affichage haute définition

Pour des écrans de micro-ordinateurs PC haute définition conventionnels, les tailles varient entre 13 et 20 pouces (mesure de la diagonale) à une définition allant de celle de la XGA (768 lignes) à celle de la UXGA (1200 lignes). Cependant, les applications super haute définition, telles que celles

des images médicales et du contrôle aérien, exigent des définitions de 2000 lignes et plus (QSXGA et QUXGA) et des tailles d'écrans allant de 20 à 30 pouces. Les solutions avec les écrans CRT les plus performants (résolution de 110 dpi) sont ainsi abandonnées au profit de celles qui utilisent les dalles TFT LCD et plasma, pouvant dépasser une résolution de 200 dpi. A cet effet, des connecteurs DVI (Digital Visual Interface), qui transmettent des signaux vidéo numériques sans conversion analogique, sont présents sur toutes les cartes graphiques récentes. Ces liaisons améliorent la qualité de l'affichage par rapport à la connexion VGA. Par ailleurs, les interfaces HDMI (High Definition Multimedia Interface), qui transmettent des signaux audio vidéo numériques, sont présentes sur les DVD (Digital Versatile Disc) haut de gamme et cartes graphiques de micro-ordinateurs PC. Celles-ci, quant à elles, permettent d'exploiter les différents formats vidéo numériques, dont la définition standard (SD, Standard Defintion), la définition améliorée (ED, Enhanced Definition) et la haute définition (HD, High Definition) ainsi que le son multi-canal, en véhiculant toutes les données grâce à un seul câble. Il existe plusieurs niveaux HDMI, i. e. HDMI 1.0 (2003), HDMI 1.1 (2004), ..., HDMI 1.4 (2009).

VII Synoptique d'une carte mère dernière génération

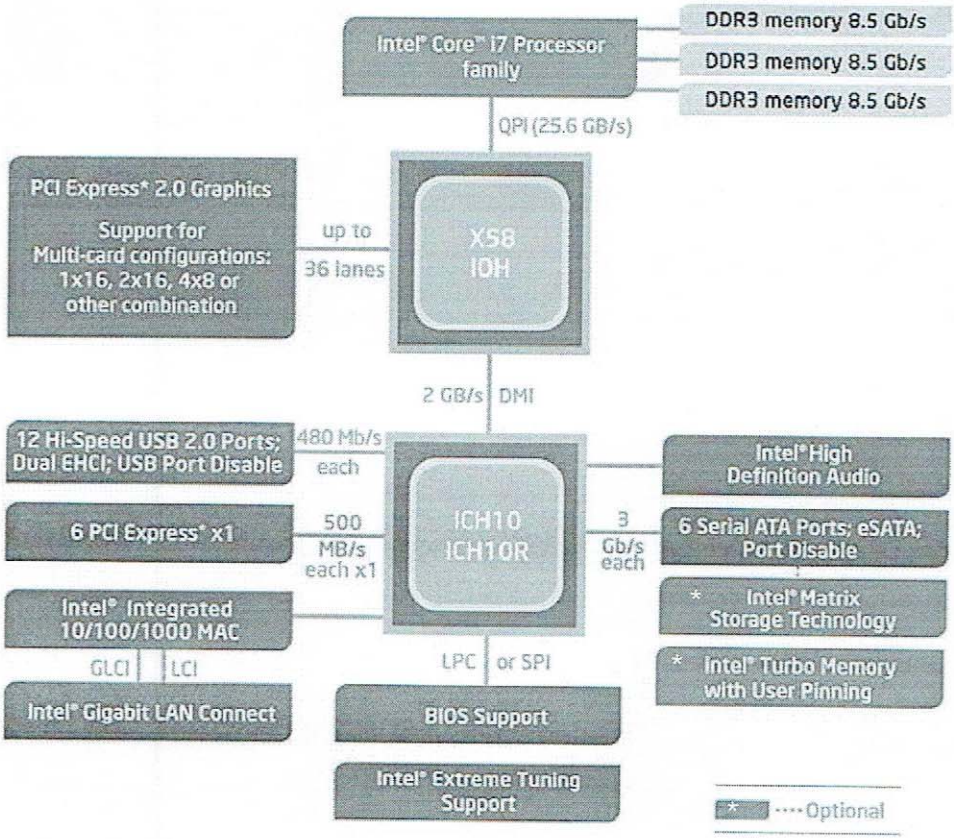


Fig. 1 Synoptique carte mère Core i7

CHAPITRE 4

ARCHITECTURE DU 8086

I Introduction

Le cœur de l'IBM PCXT est le 8086 (ou le 8088). C'est un microprocesseur de la troisième génération dont l'architecture interne est à 16 bits de données. Succinctement, ses principales caractéristiques sont les suivantes:

Il utilise le même jeu l'instruction que le 8088 mais les exécute en deux fois plus vite que le 8088 vu que ce dernier possède un bus de données de 8 bits seulement. Il adresse jusqu'à un million d'octets mémoire grâce à un bus d'adresses de 20 bits (00000H-FFFFFH). De même, il adresse jusqu'à 64 K Octets d'entrées et sorties (0000H-FFFFH) à travers les 16 bits les moins significatifs de ce même bus d'adresses.

Il travaille en mode dit pipe-line ou prefetch (prérecherche). Cela signifie que lorsqu'une instruction est en cours d'exécution le 8086 accède à la prochaine afin de supprimer les délais dus à la recherche d'une instruction avant son exécution. Ce procédé est rendu possible grâce à la technique de la file d'attente d'instructions (mémoire de 6 Octets) qui travaille selon le principe FIFO (First In First Out).

II Principe de la segmentation

La structure interne du 8086 (8088) étant limitée à 16 bits, il doit donc combiner deux mots (2 registres) de 16 bits chacun pour adresser une case mémoire de 1Mo d'adresses physiques (adresse sur 20 bits). Cette adresse est formée à partir d'un segment et d'un offset. Elle est notée segment:offset et est appelée adresse logique.

II.1 Quelques définitions

II.1.1 Segment

Le segment est une zone mémoire adressable. Il est géré par un mot (registre) segment de 16 bits et un mot (registre) offset de 16 bits.

II.1.2 Offset

L'offset est la deuxième partie de l'adresse segment. C'est une adresse de 16 bits qui indique la position d'une case mémoire dans un segment.

L'adresse physique se calcule à partir de l'adresse logique segment:offset comme suit: Adresse physique = 16 x segment + offset. Ce calcul est schématisé en Figure 1.

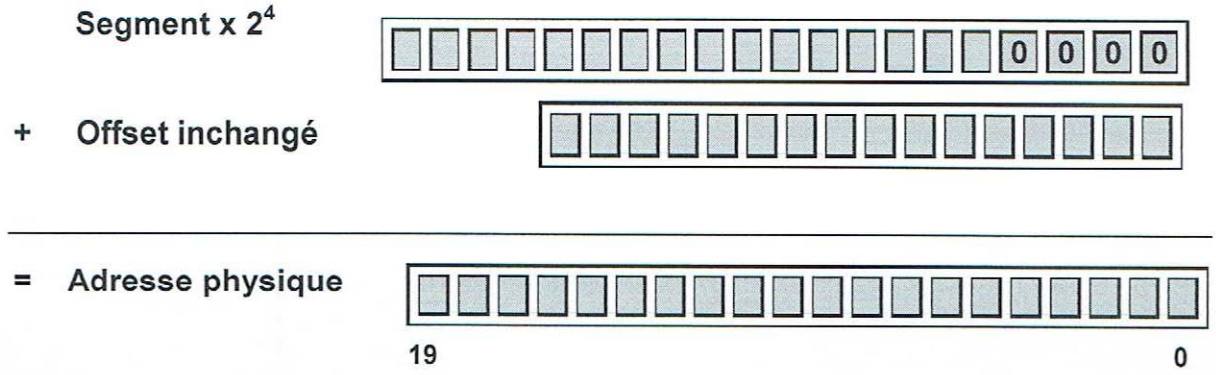


Fig. 1 Calcul de l'adresse physique

II.1.3 Exemple

Soient les 2 couples d'adresses logiques segment:offset de valeurs respectives 0F0FH:1234H et 1011H:0214H. En se référant à la Figure 1, il est facile de voir que ces 2 couples produisent la même adresse physique 10324H. De la même manière, il est possible de trouver d'autres exemples de couples qui engendrent la même adresse physique. Par conséquent, une infinité de combinaisons segment:offset peuvent produire une même adresse physique.

Le 8086 compte quatre registres segment comme le montre la Figure 2.

Fig. 2 Les registres segments du 8086

CS : Code Segment
DS : Data Segment
ES : Extra Segment
SS : Stack Segment

III Architecture du 8086

III.1 Architecture externe

Le 8086 dispose de deux modes de fonctionnement. Un mode dit minimum lorsque la broche 33 est mise à '1' et un mode dit maximum lorsque celle-ci est (Broche 33) est mise à '0'.

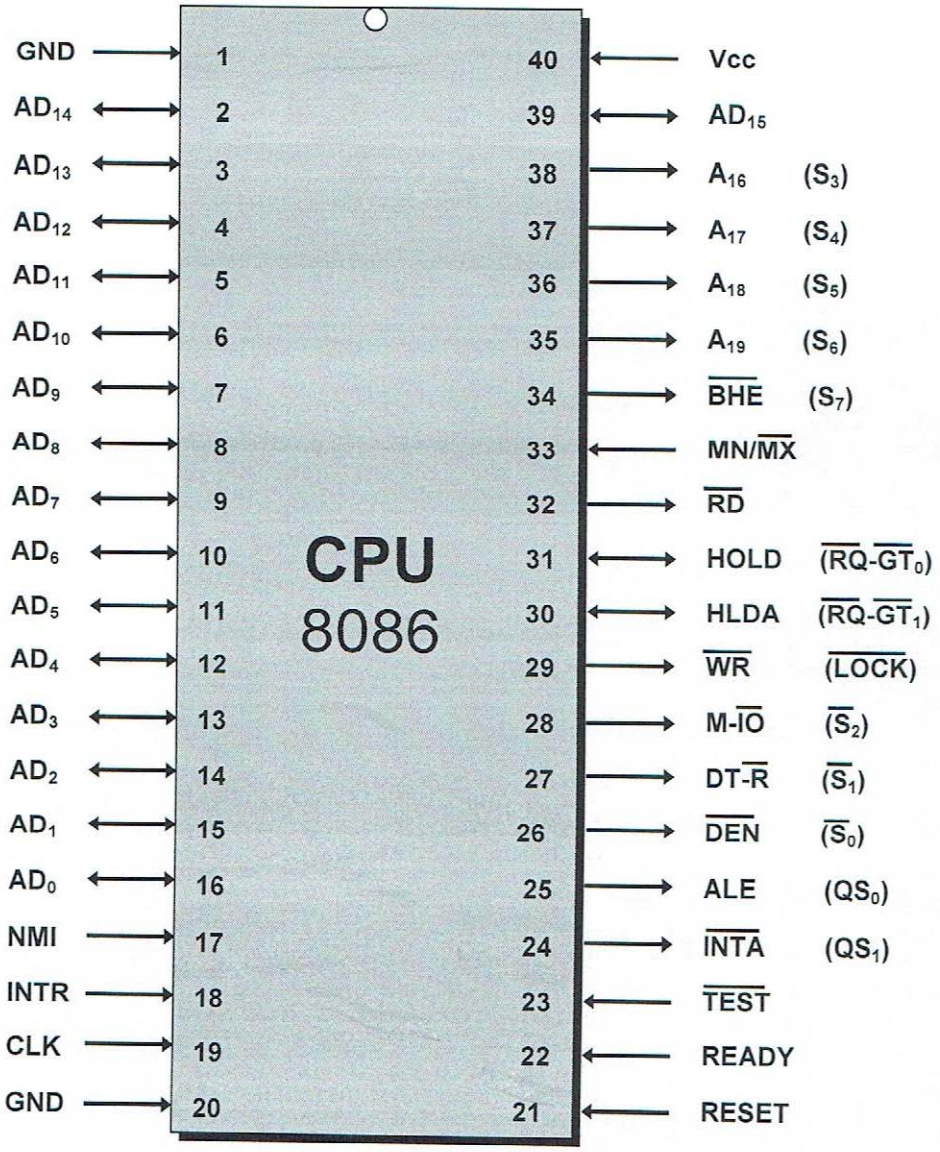


Fig. 3 Brochage du 8086

III.1.1 Mode minimum

Dans ce mode, le 8086 conserve sa pleine capacité d'adressage et gère directement le bus de contrôle, de commande et les signaux d'acquittement DMA. Ce mode est utilisé dans le cas des petits systèmes, c'est à dire les systèmes où le nombre de circuits externes est réduit.

III.1.2 Mode maximum

Ce mode étend l'architecture du 8086 aux configurations multiprocesseurs et permet ainsi l'extension du jeu d'instructions par celui du coprocesseur de calcul de type 8087. Dans ce mode un contrôleur de bus de type 8288 est utilisé pour décharger le 8086 de quelques signaux de contrôle et de commande. Les signaux du 8086 peuvent être classés suivant 3 catégories.

III.1.3 Broches communes aux deux modes maximum et minimum

Alimentation et masse (VCC et GND, entrées): Comme le 8086 est un circuit TTL (Transistor Transistor Logic), alors il admet une alimentation mono tension (VCC, GND)= (0V, 5V)

Sélection du mode maximum ou minimum (MN/$\overline{MX}$, entrée):

$$MN/\overline{MX} = \begin{cases} 1 & \text{Mode minimum ou multi-maîtres} \\ 0 & \text{Mode maximum ou mono-maître} \end{cases}$$

Demande d'interruption masquable (INTR, Interrupt Request, sortie):
Cette broche dépend de l'état du bit I (Interrupt Flag ou drapeau d'interruption)

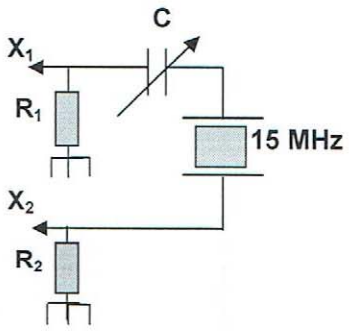
$$I = \begin{cases} 1 & \text{Interruptions autorisées} \\ 0 & \text{Interruptions masquées} \end{cases}$$

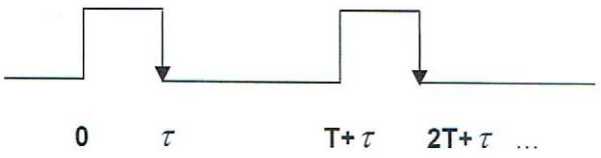
Remarque: Cette demande d'interruption émane, généralement, du PIC (8259). Avant de l'honorer, le 8086 teste d'abord l'état du bit IF d'où le nom de masquable.

Demande d'interruption non masquable (NMI, Non Maskable Interrupt, sortie): Elle est utilisée pour détecter les coupures de courant dans les PC. Elle peut interrompre une boucle infinie lorsque l'interruption INTR ne peut le faire. L'interruption NMI est honorée quel que soit l'état de IF d'où le nom de non masquable. Cependant, elle peut être masquée par une circuiterie externe.

Horloge système: (CLK, entrée): Le 8086 travaille avec une horloge de 4,77 MHz, de rapport cyclique égal à 1/3. Le rapport cyclique désigne, pour un phénomène périodique, le ratio entre la durée du phénomène sur une période et la durée de cette même période. La Figure 3 donne une illustration générale du rapport cyclique donné par $\alpha = \frac{\tau}{T}$ où $0 \leq \alpha \leq 1$.

L'horloge système est généralement fournie par le circuit 8284. Le 8284 ne peut délivrer 4,77 MHz que si on lui fournit une fréquence de base triple de celle-ci.





Le rapport cyclique est $\alpha = \frac{\tau}{T}$, $0 \leq \alpha \leq 1$

α : Temps à l'état haut pendant une période.
T : Période du signal d'horloge

R_1 = R_2 = 510 Ω et C= 5 à 30 PF

Fig. 4 Horloge système du 8086

Remise à zéro du 8086 (RESET, entrée): Ce signal permet d'initialiser le 8086. Il doit rester actif pendant au moins quatre cycles d'horloge pour laisser au 8086 le temps d'initialiser tous ses registres. Cette durée est automatiquement réalisée par le 8284. En effet, après la mise sous tension, tous les registres du 8086 sont mis à 0 sauf le registre code segment CS qui est mis à FFFFH et dont le calcul d'adresse physique donne FFFF0H, adresse où doit se trouver une instruction de saut; Dans un contexte PC, il s'agit du BIOS qui va aller chercher le système d'exploitation pour lui passer le contrôle.

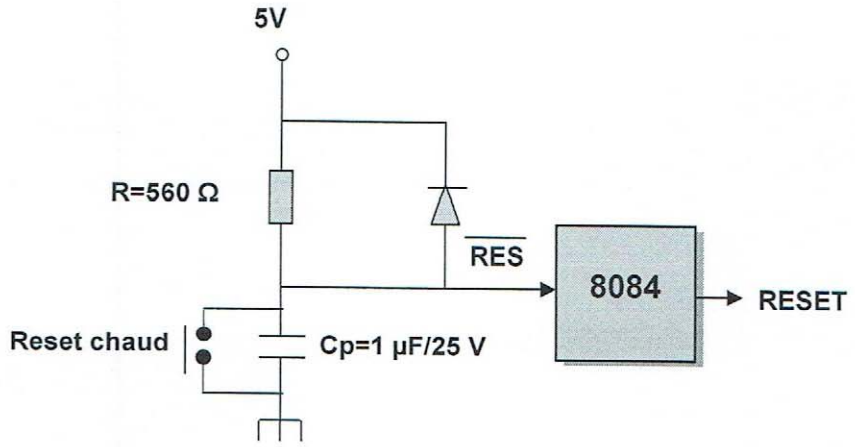


Fig. 5 Remize à zéro du 8086

Signal de synchronisation (READY, Entrée): Dès qu'un circuit mémoire et/ou périphérique est (sont) adressé (s) et qu'il (s) n'a (ont) pas la possibilité d'exécuter le transfert à la vitesse du 8086, il (s) l'indique (ent) au 8284 à travers les signaux RDY_1 et/ou RDY2. A ce moment, le 8084 met à zéro READY en synchronisation avec l'horloge système.

Signal de synchronisation ($\overline{TEST}$, Entrée): Ce signal (entrée) permet également de synchroniser le 8086 sur une information extérieure. Lorsque le 8086, décode une instruction *WAIT*, il teste l'état de $\overline{TEST}$ et ne permet l'exécution du programme que si $\overline{TEST}$ passe à 0. Cette ligne correspond à la broche BUSY du 8087. Elle permet ainsi de faire attendre le 8086 lorsqu'il a besoin d'un résultat numérique en cours d'exécution dans le 8087 pour pouvoir poursuivre sa tâche.

Bus d'adresses et de données multiplexés (AD_0 - AD_{15}, sorties): Bus d'adresses et de données bidirectionnels en logique 3 états multiplexés. Il se comporte en sortie au moment du dépôt de l'adresse et en entrée/sortie au moment de l'échange de données.

Bus d'aresses et d'états du 8086 multiplexés (A_{19} (S_6)-A_{16} (S_3), sorties): Bus d'adresses et d'états en logique 3 états. Ces signaux sont également multiplexés. Durant la sortie de l'adresse, ils constituent les quatres bits de poids forts pour les opérations mémoires. Pendant l'échange de données, S_3 et S_4 permettent de connaître le registre segment qui a permis le calcul de l'adresse physique. S_5 permet de connaître l'état du flag ou drapeau d'interruption I. S_6 est toujours à zéro.

Tableau 1 Signification des quatre bits de poids forts du bus d'adresses pendant l'échange de données

S_4	S_3	Registre Segment
0	0	ES
0	1	SS
1	0	CS ou aucun
1	1	DS

$S_5 = I =$ { 1 Interruptions validées ; 0 Interruptions inhibées } et S_6 est toujours à 0

Validation de l'octet fort du bus de données ($\overline{BHE}$ (S7) Bus High Enable): Durant la sortie de l'adresse, $\overline{BHE}$ et A_0 permettent de valider la partie haute et la partie basse du bus de données, respectivement. Cependant, durant

l'activation du bus de données, S_7 indique si le 8086 procéde à une opération de reconnaissance d'interruption.

III.1.4 Broches en mode maximum

Signaux de gestion des resources mémoire et E/S ($\overline{S_2}$, $\overline{S_1}$ et $\overline{S_0}$, sorties): Dans le mode maximum, l'utilisation du 8288 est indispensable déchargeant ainsi le 8086 de la gestion de quelques signaux de commande.

Tableau 2 Signaux de gestion des ressources mémoire et E/S du 8086

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Cycle d'opération
0	0	0	Reconnaissance d'une interruption
0	0	1	Lecture entrée/Sortie
0	1	0	Ecriture Entrée/Sortie
0	1	1	HALT
1	0	0	Recherche d'une instruction
1	0	1	Lecture mémoire
1	1	0	Ecriture mémoire
1	1	1	Etat de repos

A l'état de repos toutes les trois lignes $\overline{S_2}$, $\overline{S_1}$ et $\overline{S_0}$ sont à '1'. Le passage à '0' de l'une d'entre elles déclenche un nouveau cycle d'opération et indique au 8288 la nature du cycle en cours pour lui permettre de générer les signaux correspondant.

Demande des bus ($\overline{RQ}$-$\overline{GT_{0/1}}$, Request/Grant, entrées et sorties)

Chacune de ces deux broches reçoit les commandes $\overline{RQ}$ (request) d'utilisation du bus local par le coprocesseur pour transférer une donnée à un autre maître du bus. Le 8086 répond à travers sa ligne $\overline{GT}$ (Grant) à la fin du cycle d'opération en cours. Le coprocesseur ayant pris le contrôle du bus local indique enfin au 8086 par la même broche (Request) qu'il peut reprendre l'usage de son bus. Au cours de l'utilisation du bus local par un autre maître, le 8086 est dans un état de haute impédance. Enfin, notons que $\overline{RQ}$ ($\overline{GT_0}$) est plus prioritaire que $\overline{RQ}$ ($\overline{GT_1}$).

Verouillage des bus ($\overline{LOCK}$, sortie): Ce signal en logique 3 états est contrôlé par logiciel. En écrivant le préfixe LOCK en tête d'une instruction, ce signal indique aux autres maîtres des bus qu'ils n'ont pas le droit d'en prendre le contrôle. Il reste actif jusqu'à l'exécution complète de l'instruction.

Remarque: Le verrouillge est très important dans les opérations de lecture et d'écriture répétitives. En effet, il faut empêcher qu'une zône mémoire ne soit

modifiée par un autre processeur avant la fin d'une opération de lecture ou écriture. Le signal $\overline{LOCK}$ passe à l'état de haute impédance lorsque le 8086 perd l'usage de son bus. A ce moment, un autre processeur (maître) du bus peut utiliser cette ligne à son profit.

Etat de la file d'attente d'instructions (QS_1 et QS_0, Queue State, sorties): Signaux relatifs à la file d'attente d'instructions. Ils permettent au coprocesseur ou à d'autres circuits de suivre l'utilisation que fait le 8086 de sa file d'attente d'instructions.

Tableau 3 Signaux relatifs à l'état de la file d'attente d'instructions

QS_1	QS_0	Opération
0	1	Pas d'opération
0	1	Recherche du 1er octet dans la queue
1	0	Effacement de la queue
1	1	Recherche de l'octet suivant

III.1.5 Broches en mode minimum

Adressage mémoire ou E/S (M-$\overline{IO}$, Memory/ Input/Output, sortie): Ce signal en logique 3 états, sert à distinguer un accès mémoire de celui des entrées et sorties.

$$M\text{-}\overline{IO} = \begin{cases} 1 & \text{Mémoire} \\ 0 & \text{E/S} \end{cases}$$

Transmission ou réception des données (DT-$\overline{R}$, Data Transmit Data Receive, sortie): Ce signal en logique 3 états, sert à contrôler le sens de la transmission de données des transmetteurs (8286/87) sur le bus de données.

$$DT(\overline{R}) = \begin{cases} 1 & \text{Transmission de données} \\ 0 & \text{Reception de données} \end{cases}$$

Remarque: M($\overline{IO}$) et DT($\overline{R}$) sont valides durant tous le cycle d'opération. Ils se mettent à l'état de haute impédance à la demande d'un autre processeur comme le DMA, par exemple.

Signal de lecture ($\overline{RD}$, Read, sortie): Ce signal en logique 3 états, est actif à l'état bas. Après la sortie de l'adresse, il indique l'exécution d'une opération de lecture par le 8086. Il passe à l'état de haute impédance quand le 8086 perd l' usage de ses bus.

Signal d'écriture ($\overline{WR}$, Write, sortie): Ce signal en logique 3 états est actif à l'état bas. Après la sortie de l'adresse, il indique l'exécution d'une opération d'écriture par le 8086. Il passe à l'état de haute impédance quand le 8086 perd l' usage de ses bus.

Reconnaissance des interruptions ($\overline{INTA}$, Interrupt Acknowledge, sortie): Ce signal (sortie) joue le même rôle que $\overline{RD}$ durant la reconnaissance d'une interruption. Le 8086 lit ainsi le numéro de l'interruption placée sur son bus de données par le circuit 8259.

Capture d'adresses (ALE, Address Latch Enable, sortie): Actif à l'état haut, ce signal (sortie) sert à capturer l'adresse dans les latches 8282/83 au début du cycle d'opération avant la transformation du bus d'adresses en bus de données.

Demande des bus (HOLD et HLDA, Hold and Hold Acknowledge, entrée et Sortie): Lorsqu'il est actif, le signal HOLD (entrée) indique au 8086 qu'un autre maître demande l'usage du bus. Le 8086 répond par un HLDA (sortie) à la fin du cycle d'opération en cours et met simultanément tous ses bus à l'état de haute impédance. Lorsque HOLD redevient à l'état bas à l'initiative de l'utilisateur du bus, HLDA redescend et le 8086 reprend le contrôle du bus.

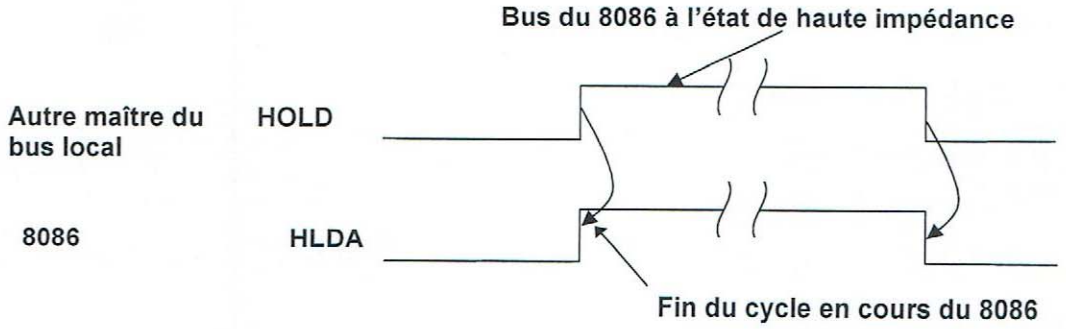


Fig. 6 Chronogramme d'un cycle DMA

Validation du bus de données ($\overline{DEN}$, Data Enable, sortie): *DEN* autorise les transferts de données aussi bien en entrée qu'en sortie. Agissant au niveau des tranmetteurs bidirectionnels (8286/87), il peut isoler le bus local pendant qu'un autre maître l'utilise (le DMA par exemple).

III.2 Chronogrammes de Lecture/Ecriture dans le 8086

III.2.1 Cycle d'écriture et de lecture dans le 8086

Pour effectuer une lecture/écriture, le 8086 exécute un cycle d'opération comptant au minimum 4 périodes d'horloge (T_1,T_2,T_3 et T_4). L'adresse sur 20 bits est émise durant la période T_1, le signal *ALE* permet alors à l'aide de son front descendant de capturer l'adresse et de la maintenir durant tout le cycle d'opération.

Les signaux $\overline{RD}$ et $\overline{WR}$ sont activés à partir de T_2. Le transfert de données (8 ou 16 bits) a lieu durant T_3 et T_4. Si la mémoire ou circuit d'entées et sorties n'est pas capable d'exécuter le transfert de données à la vitesse du 8086, elle (il) doit le faire savoir au 8086 avant le front descendant de T_3 ou pendant T_2. A ce moment, le 8086 insère un ou plusieurs cycles supplémentaires T_W à la suite de T_2 (READY = 0) jusqu'à ce que la mémoire ou le circuit des entrées et sorties soit prêt. A ce moment, le 8086 termine alors le cycle de l'opération.

III.2.2 Gestion des temps d'attente (Wait/State)

La Figure 8 montre l'insertion d'un cycle supplémentaire T_W à la suite de T_2. Dans ce cas, si READY = 1 pendant le deuxième front déscendant de T_2, alors le 8086 termine le cycle de l'opération (T_3,T_4). Sinon, il insère un autre cycle supplémentaire à la suite de T_2 et ainsi de suite. Pour mieux comprendre le concept du Wait State (WS) et plus précisement le calcul du nombre de périodes supplémentaires qu'il faut insérer, nous avons pris quelques exemples de fréquences d'horloges dans le Tableau 4. Si, d'une part, nous notons par $T\mu$ la période élémentaire du 8086 tel que T_1=T_2=T_3=T_4= $T\mu$ alors, la lecture ou l'écriture d'une donnée par celui-ci nécessite $2T\mu$. Si, d'autre part, nous notons par T_A, le temps d'accès mémoire ou entrées et sorties (E/S), alors il est possible de trouver une relation empirique donnant le nombre de WS en fonction de $T\mu$ et T_A comme suit:

$$\textbf{Nombre de WS} = \frac{T_A \quad 2T_\mu}{T_\mu} \textbf{ majoré à l'entier immédiatement supérieur}$$

A tirtre d'exemple, pour un 8086 travaillant à une horloge de 8 MHz, nous avons $2T\mu$= 250 ns. Si nous disponsons d'un circuit mémoire ayant un temps d'accès T_A= 400 ns, alors en utilisant la formule ci-dessus, nous obtenons le Nombre de WS=1.2. En majorant à l'entier immédiatement supérieur, nous trouvons le Nombre de WS= 2. Ce qui est conforme au résultat trouvé dans le Tableau 4. Par ailleurs, nous pouvons clairement remarquer que pour des temps d'accés mémoire ou circuits d'entrées et sorties (E/S) constants, plus l'horloge du 8086 est élevée, plus le Nombre de WS est élevé.

Tableau 4 Quelques exemples de nombre de Wait State

CLK (MHz)	Période (ns)	TA (ns)	Nombre de WS
4.77	200 x 2 = 400	200 400 450	0 0 1
8	125 x 2 = 250	200 400 450	0 2 2
16	62.5 x 2 = 125	200 400	2 5

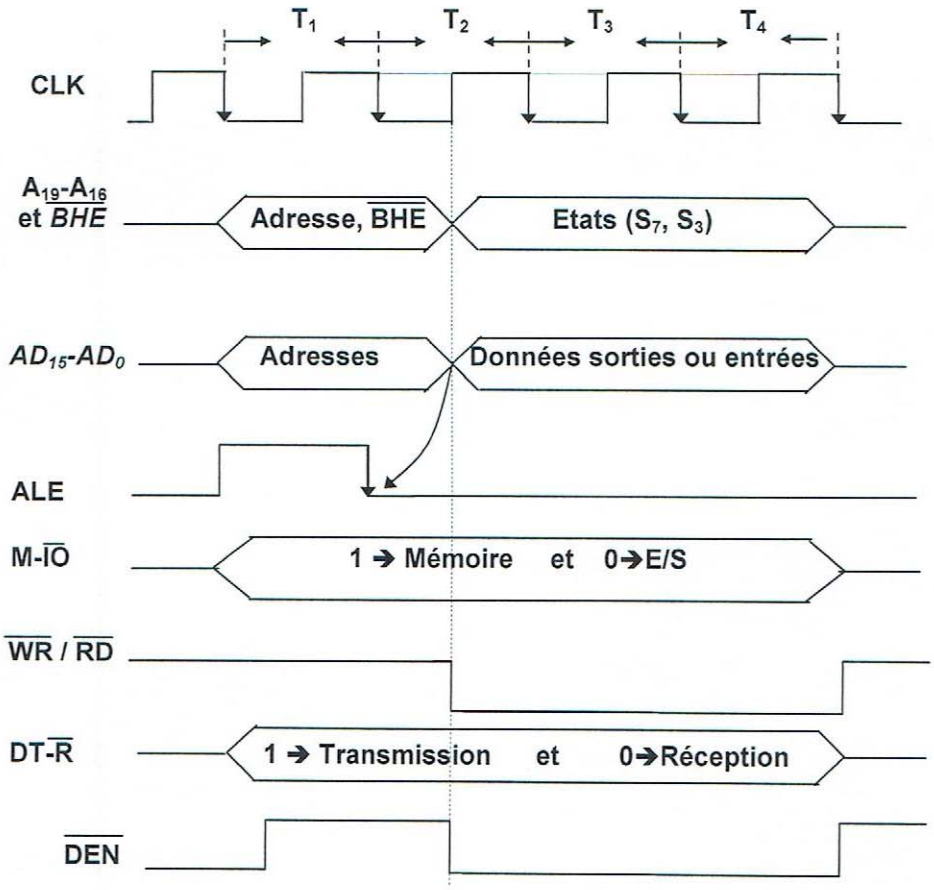


Fig. 7 Chronogramme de lecture/écriture dans le 8086

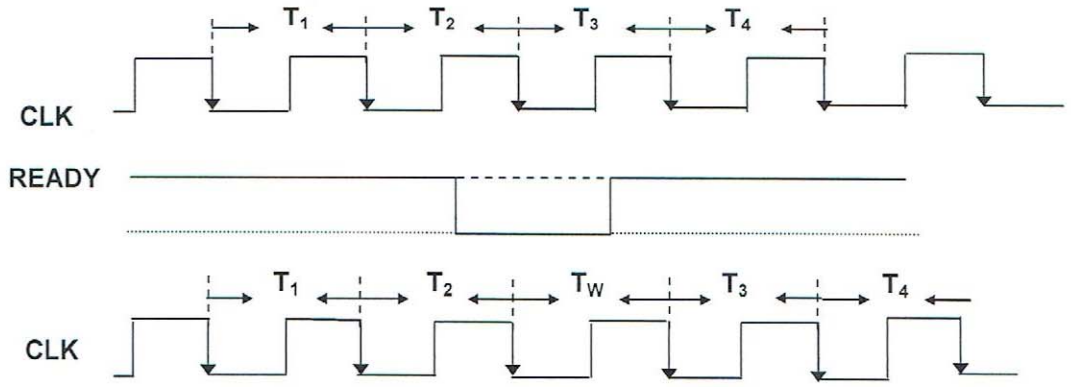


Fig. 8 Chronogramme de gestion des temps d'attente

III.2.3 Organisation de l'espace mémoire dans le 8086

Comme nous l'avons stipulé précédement, le 8086 peut adresser jusqu'à 1 Méga octets d'emplacements mémoire à l'aide de son bus d'adresses A_0 - A_{19}. Il est important de savoir qu'i ne s'agit pas de 1 Méga mots car par conception, comme le montre la Figure 9, la mémoire du 8086 est organisée en octets et non pas en mots et ce à cause de la longueur du bus d'adresses qui n'est en fait que 19 fils ($A_1 - A_{19}$). Quant à A_0, il sert à selectionner la partie basse du bus de données ($D_0 - D_7$), alors que la partie haute est selectionnée par $\overline{BHE}$.

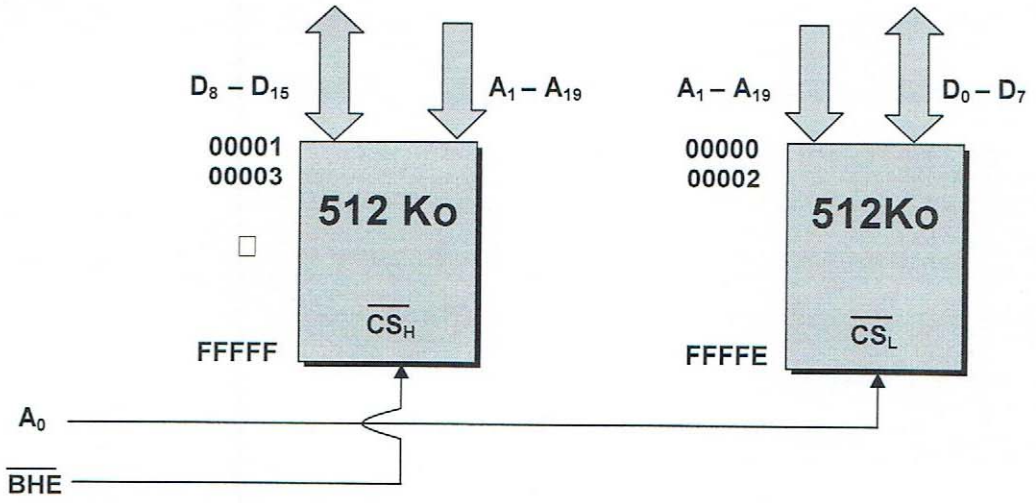


Fig. 9 Organisation de l'espace mémoire du 8086

Tableau 5 Selection d'un mot ou d'un octet dans le 8086

$\overline{BHE}$	A_0	Ecriture/Lecture
0	0	Mot adresse paire
0	1	Octet adresse impaire
1	0	Octet adresse paire
1	1	Pas d'opération

Une simple lecture du Tableau 5, nous conduit à poser les trois questions suivantes:

1- Comment écrire/lire un mot situé à une adresse impaire?
2- Comment différencier une écriture/lecture d'un mot de celle d'un octet, tous deux situés à une adresse paire?
3- Comment éviter qu'une suite de données ne soit située à des adresses impaires?

Les réponses peuvent être résumées, respectivement, comme suit:

1- Un mot situé à une adresse impaire est lu/écrit en deux cycles d'opération.
2- $\overline{BHE}$ établit la différence entre une écriture/lecture d'un mot et celle d'un octet situeé à une adresse paire (Cf. Tableau 5).
3- Utiliser la directive d'assemblage EVEN (PAIR).

Mot/Octet situé à une adresse impaire: La lecture/écriture d'un mot situé à une adresse impaire est schématisée par la Figure 9. En effet, comme le montre le Tableau 5 une telle opération nécessite deux cycles d'opération. Le 1er sert à transférer l'octet d'adresse X+1 et le 2ème à transférer l'octet d'adresse X.

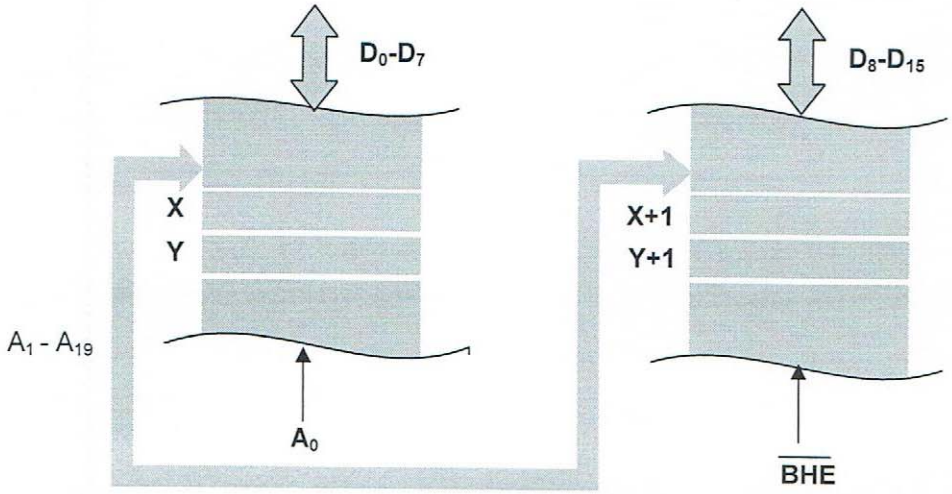


Fig. 10 Lecture/écriture d'un mot situé à une adresse impaire

Mot situé à une adresse paire: Contrairement au mot situé à une adresse impaire, le 8086 lit et écrit un mot situé à adresse paire A_0 (bas) et $\overline{BHE}$ (bas) en un seul cycle d'opération. Par conséquent, les transferts des octets d'adresses X+1 et X s'effectuent en même temps.

Octet situé à une adresse paire: Toujours à partir du Tableau 5, il est facile de remarquer que pour lire ou écrire un octet situé à une adresse paire A_0 (bas) et $\overline{BHE}$ (haut), il nous faut un cycle d'opération c'est à dire le transfert de l'octet d'adresse X.

Octet situé à une adresse impaire: La lecture/écriture d'un octet situé à une impaire A_0 (haut) et $\overline{BHE}$ (bas), s'effectue aussi en un cycle d'opération donnant lieu au transfert de l'octet d'adresse X+1.

III.3 Architecture interne du 8086

Le 8086 dispose de deux unités distinctes. L'unité d'exécution (UE) et l'unité d'interface de bus (UIB).

III.3.1 Unité d'exécution

Cette unité contient essentiellement l'unité arithmétique et logique (UAL). Elle contient aussi le mot d'état et de contrôle du 8086 et manipule les opérandes et les registres généraux (AX, BX, CX et DX). Enfin, à l'exception du registre IP, elle contient tous les pointeurs et index. Il faut noter que tous les registres et bus internes de l'unité d'exécution sont à 16 bits. Par ailleurs, elle est complètement déconnectée de l'environnement extérieur. Les opérandes et les instructions lui sont fournis par une mémoire interne appelée file d'attente d'instructions de capacité 6 octets. Cette mémoire fonctionne en FIFO (First In First Out) et elle est gérée par l'unité d'interface de bus.

III.3.2 Unité d'interface de bus

Cette unité effectue toutes les opérations pour le compte de l'unité d'exécution. Ainsi, c'est elle qui contient tous les registres segments et c'est elle qui calcule l'adresse physique pour le compte du 8087. L'unité d'interface de bus contient aussi le registre IP qui lui permet par incrémentation simple d'aller chercher les opérandes et les instructions en mémoire et de les mettre dans la file d'attente d'instructions (FAI) quand l'accès au bus le lui permet et quand celle-ci n'est pas pleine. Il faut noter qu'une instruction de saut entraîne le vidage de la file d'attente d'instructions.

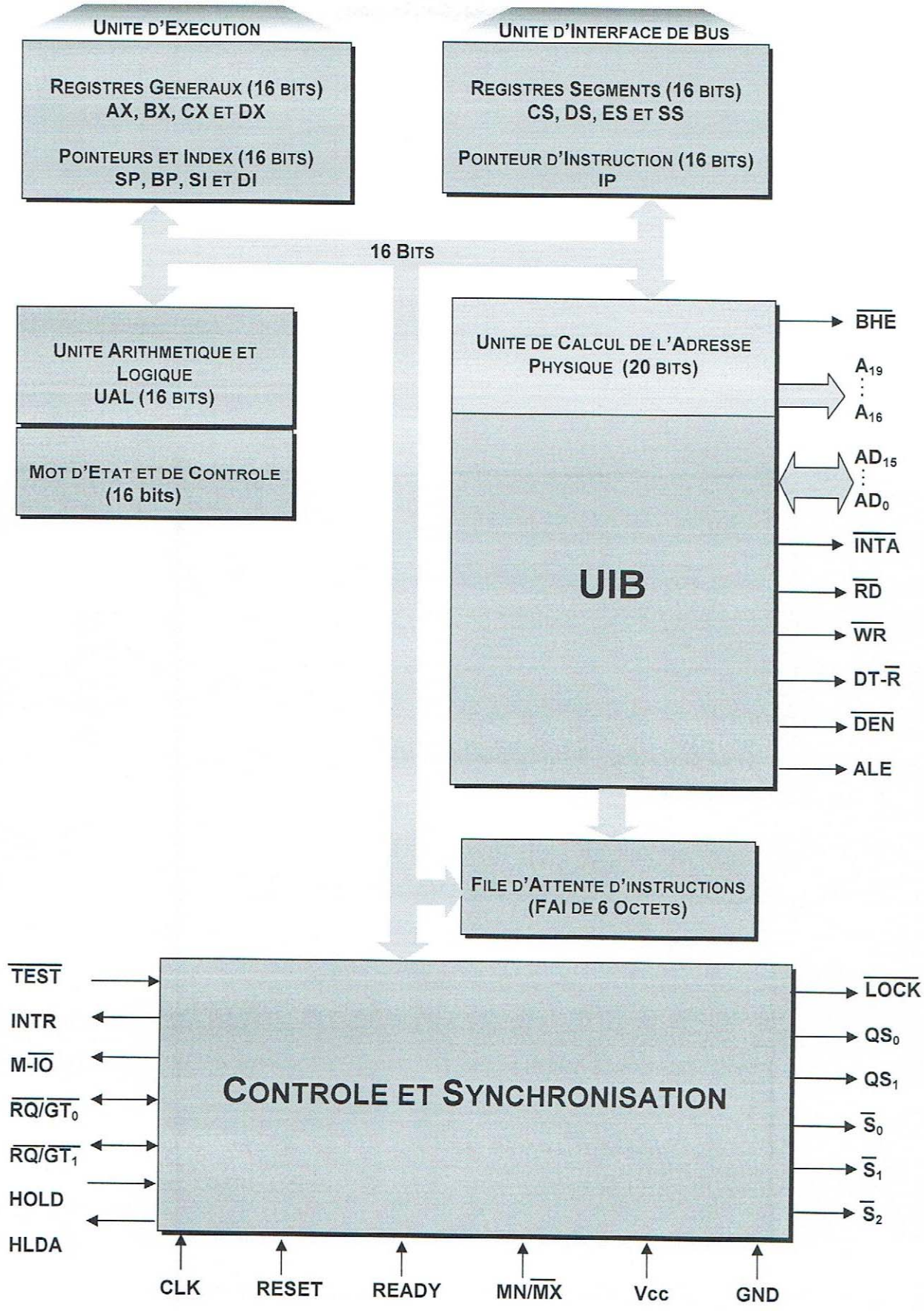
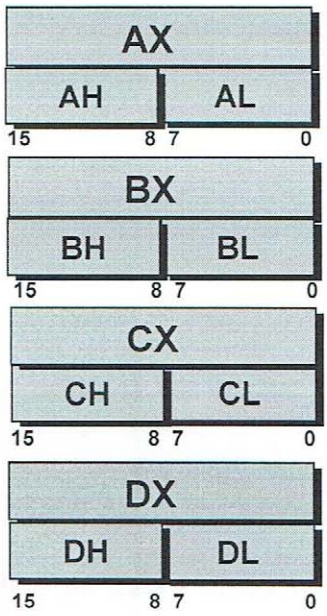


Fig. 11 Architecture interne du 8086

III.3.3 Registres du 8086

Le 8086 compte quatre catégories de registres répartis sur l'UE) et l'UIB.

Registres généraux 8 et 16 bits



Accumulator ou accumulateur 8 ou 16 bits

AL et AH: Registres de 8 bits chacun

Base register ou registre de base 16 bits

BL et BH: Registres de 8 bits chacun

Counter ou compteur 8 ou 16 bits

CL et CH : Registres de 8bits chacun

Data register ou register data 16 bits

DL et DH : Registres de 8 bits chacun

Fig. 12 Les registres généraux du 8086

Registres généraux 16 bits

Registre AX: Ce registre peut être utilisé dans n'importe qu'elle opération mais son emploi est obligatoire dans les opérations de multiplication et de division ainsi que les transferts entre 8086 et la périphérie et certaines opérations répétitives.

Registre BX: Ce registre est utilisé comme pointeur et sert de registre auxiliaire dans certaines opérations.

Registre CX: Ce registre est obligatoirement utilisé dans les opérations de comptage et les opérations répétitives.

Registre DX: Ce registre est associé à AX pour former un mot de 32 bits dans les opérations de multiplication et de division.

Registres généraux 8 bits: Tous les registres généraux du 8086 peuvent être adressés comme des registres de 8 bits tel que RX = RH o RL où R = A ou B ou C ou D. Le 'o' est appelé symbole de concaténation. Par exemple AL s'occupe de tous les transferts 8 bits avec la périphérie et joue le rôle de l'accumulateur dans

les opérations arithmétiques sur 8 bits. CL quant à lui, est employé comme compteur dans les opérations de décalage et de rotation.

Exemples:

```
ADD  AL, BL   (AL← AL+BL)
IN   AL, DX   (AL← (DX))
MOV  CL, 02
SHL  AL, CL
```

Registres pointeurs et index: BP, SP et SI, DI

Un pointeur ou un index est une variable contenant l'offset seul d'une étiquette d'instruction ou de l'adresse d'une variable. Les deux pointeurs servent à la génération des adresses des données en particulier en pile pour SP. Les deux index permettent la gestion de suite de mots.

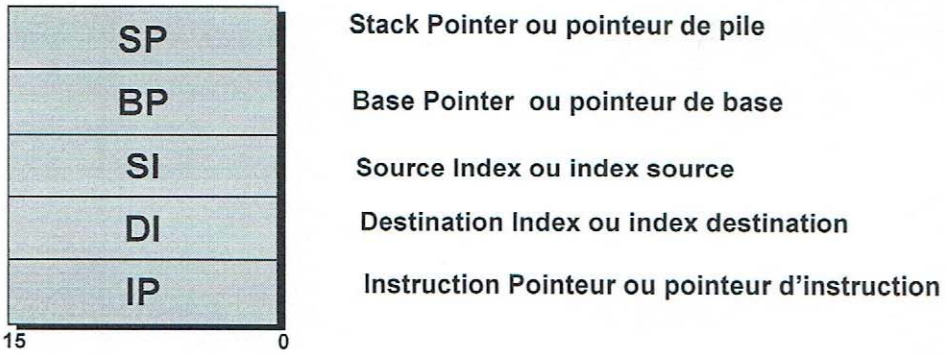


Fig. 13 Les registres pointeurs et index du 8086

Remarque: En plus de son rôle de registre général à 16 bits, BX peut aussi être utilisé comme pointeur.

Registre segments

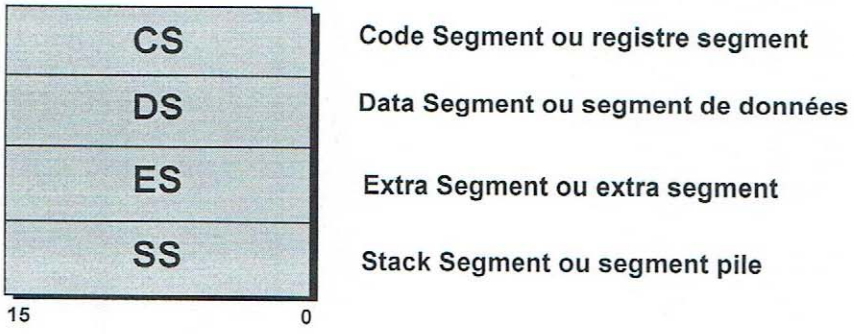


Fig. 14 Les registres segments du 8086

Registre CS: Ce registre est utilisé exclusivement pour les instructions de programme.

Registre SS: *Ce registre* est utilisé exclusivement pour les opérations avec la pile.

Registres DS et ES: Ces deux registres sont utilisés pour les variables et les données.

Combinaisons Segment:Offset

Toute combinaison segment:offset où l'offset désigne un registre pointeur ou un registre index, doit satisfaire l'une des conditions suivantes:

IP: Travaille exclusivement avec CS pour adresser les instructions du programme.
SP: Travaille exclusivement avec SS pour les opérations avec la pile.
BP: Travaille de préférence avec SS. Il peut être associé à DS, CS et ES.
BX: Travaille de préférence avec DS. Il peut être associé à SS, ES et CS.
DI: Travaille de préférence avec DS mais obligatoirement avec ES pour les instructions répétitives. Il peut être associé à SS, ES et CS.
SI: Travaille de préférence avec DS. Il peut être associé à SS, ES et CS.

Cas implicite

```
MOV  AX, [BP]        Segment =SS et Offset = BP
MOV  AX, [BX] [SI]   Segment =DS et Offset = BX+SI.
MOV  AX, [BP] [SI]   Segment =SS et Offset = BP + SI
```

Cas explicite

```
MOV  AX,CS : [BP]      ; Segment =CS et Offset = BP
MOV  ES :[BX],AX       ; Segment =ES et Offset = BX
MOV  AX, DS :[BP][SI]  ; Segment =DS et Offset = BP+SI
```

Mot d'état et de contrôle PSW (Program Status Word)

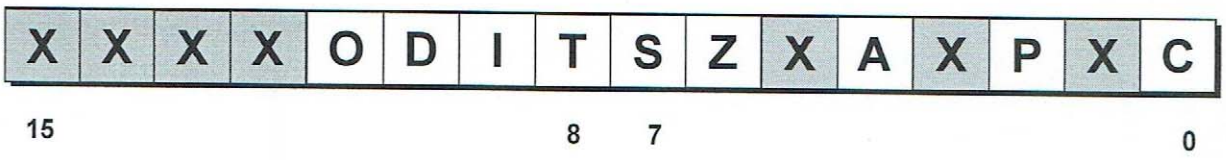


Fig. 15 Le registre d'état du 8086

A (Half Carry) = { 1 Si opération sur 8 ou 16 bits a généré une demi retenue ; 0 Sinon.

C (Carry) = { 1 Si opération sur 8 ou 16 bits a généré une retenue ; 0 Sinon.

Z (Zéro) = { 1 Si le contenu d'un registre est nul ; 0 Sinon.

S (Sign) = { 1 Si le contenu d'un registre est négatif ; 0 Sinon.

O (Overflow) = { 1 Si opération sur 8 ou 16 bits a produit un dépassement ; 0 Sinon.

P (Parity) = { 1 Si le nombre de 1 est pair dans un mot de 8 ou 16 bits ; 0 Sinon.

D (Direction) = { 0 Incrémentation de l'index ; 1 Décrémentation de l'index

I (Interrupt) = { 0 Interruptions masquées ; 1 Interruptions autorisées

T (Trap) = { 1 Exécution pas à pas ; 0 Sinon

Remarques:

1- Le bit Half Carry est principalement utilisé dans les ajustements décimaux.
2- Le bit Sign est principalement utilisé dans les branchements conditionnels.

3- Le bit Parity est principalement utilisé dans les transmissions série et parallèle.
4- Le bit Direction est principalement utilisé dans les opérations répétitives.

IV Concept de la pré-recherche (prefetch)

L'exemple suivant illustre le gain en temps réalisé grâce à la séparation des taches entre l'unité d'exécution et l'unité d'interface de bus et à l'utilisation de la file d'attente d'instructions ou FAI. Soit donc la suite d'instructions:

1- MOV [BX], AL	(DS :[BX]← AL)	Avec accès externe
2- MUL BX	(DX x AX← AX x BX)	Sans accès externe
3- MOV CX, 2	(CX← 0002)	Avec accès externe
4- ADD SI, CX	(SI← SI+CX)	Sans accès externe
5- ADD DI, DX	(DI← DI+DX)	Sans accès externe

Dans ce qui suit, nous allons comparer les performances de deux microprocesseurs. Le premier appartient à la deuxième génération et par conséquent il ne possède pas de file d'attente d'instructions et le deuxième à une génération ultérieure comportant une file d'attente d'instructions.

IV.1 Microprocesseur de la 2ème génération

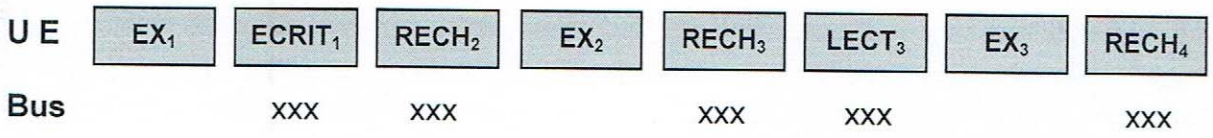


Fig. 16 Occupation des bus pour un microprocesseur de la 2ème génération, 'xxx' représente l'occupation des bus.

IV.2 Microprocesseur avec file d'attente d'instructions (prefetch)

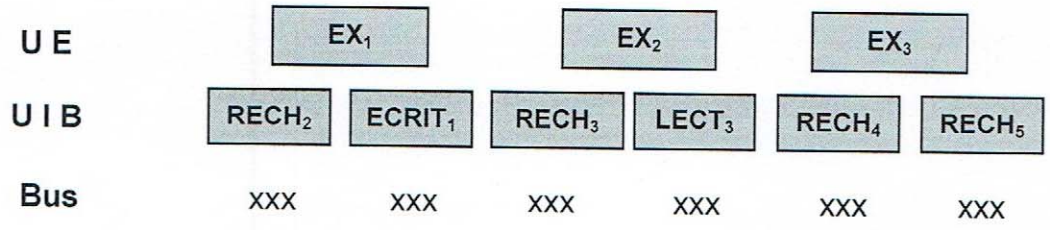


Fig. 17 Occupation des bus pour un microprocesseur avec file d'attente d'instructions, 'xxx' représente l'occupation des bus.

Si nous regardons l'utilisation que fait chacun des deux microprocesseurs de ses bus respectifs, nous remarquons que celui qui utilise une file d'attente d'instructions occupe beaucoup plus les bus que l'autre et ce surtout pendant l'exécution des instructions. Par conséquent, le gain en temps est réalisé grâce à la séparation des taches des UE et UIB et de l'utilisation FAI.

CHAPITRE 5

ASSEMBLEUR DU 8086

I Introduction

L'assembleur du 8086 constitue un outil logiciel de conversion du code source en code machine. Tout microprocesseur possède son propre assembleur. Avant d'introduire l'assembleur du 8086, nous devons passer en revue tous les types d'instructions de celui-ci. Ces instructions sont aussi appelées mnémoniques. Chaque mnémonique constitue une abréviation de l'opération que peut exécuter un microprocesseur et qui fait donc partie de son jeu d'instructions.

II Jeu d'instructions du 8086

Il existe six catégories de types d'instructions dans le 8086.

Instructions de transfert de données.
Instructions arithmétiques.
Instructions logiques.
Instructions répétitives (manipulation des chaînes de caractères).
Instructions de branchements.
Instructions de contrôle du 8086.

II.1 Instructions de transferts de données

II.1.1 Transferts explicites

Tableau 1 Instructions de transferts explicites

Mnémoniques	Description	O	S	Z	A	P	C
MOV AX,SI	AX ← SI	Aucun n'est affecté					
MOV AX,3	AX ← 0003						
MOV AL,5	AX ← 0005						
PUSH BX Empilement	(1) SP← SP-1 ; (2) SS :[SP]← BH (3) SP← SP-1 ; (4) SS :[SP]← BL						
POP DI Désempilement	(1) SS :[SP]→DI_L ; (2) SP← SP+1 (3) SS :[SP]→DI_H ; (4) SP← SP+1						
XCHG AX,DI	AX ↔ DI						
XCHG [BP],CX	SS :[BP] ↔ CX						

II.1.2 Transferts implicites

Tableau 2 Instructions de transferts implicites

Mnémoniques	Description	O	S	Z	A	P	C
IN AX,DX	AX ← (DX)	Aucun n'est affecté					
IN AL,28H	AL ← (28H)						
IN AX,FFH	AX ← (FFH)						
IN AL,DX	AX ← (DX)						
OUT FFH,AX	AX → (FFH)						
OUT DX,AL	AX→ (DX)						
XLAT Translate byte to AL Table indexée par AL	DS:Segment ; BX + AL : Offset AL ← DS :[BX+AL]						

II.1.3 Transferts d'adresses

Tableau 3 Instructions de transferts d'adresses

Mnémoniques	Description	O	S	Z	A	P	C
LEA DI, [SI] [BP]	DI ← SI+BP	Aucun n'est affecté					
LEA BX,TAB	BX ← Offset de TAB						
LEA BX, [1234]	BX ← 1234						
LDS BX,TAB	BX ← DS:[TAB] DS← DS:[TAB+2]						
LES BX,TAB	BX ← ES:[TAB] ES ← ES:[TAB+2]						

II.1.4 Transferts du PSW mot d'états

Tableau 4 Instructions de transferts du mot d'état

Mnémoniques	Description	O	S	Z	A	P	C
LAHF	AH← Flow	Aucun n'est affecté					
SAHF	Flow ← AH						
PUSHF	SP←SP-1; SS:[SP]←FH SP←SP-1; SS:[SP]←FL						
POPF	SS :[SP]→FL ; SP←SP+1 SS :[SP]→FH ; SP←SP+1						

II.2 Instructions arithmétiques

II.2.1 Addition

Tableau 5 Instructions d'addition

Mnémoniques	Description	O	S	Z	A	P	C
ADD SI,AX	SI ← SI + AX	X	X	X	X	X	X
ADD SI,[DI]	SI ← SI + DS:[DI]	X	X	X	X	X	X
INC CX	CX ← CX + 1	X	X	X	X	X	X
INC [BX]	INC BPTR[BX] INC WPTR[BX]	X	X	X	X	X	X
ADC SI,AX	SI ← SI + AX + C	X	X	X	X	X	X
DAA (*)	Decimal Adjust for Addition	?	X	X	X	X	X

•	**Non défini**
X	**Modifié en fonction du résultat**
?	**Indéfini**

II.2.3 Soustraction

Tableau 6 Instructions de soustraction

Mnémoniques	Description	O	S	Z	A	P	C
SUB AX,DI	AX ← AX – DI	X	X	X	X	X	X
SUB [SI][BP],AX	[SI+BP] ← SS:[BP + SI] - AX	X	X	X	X	X	X
SBB AX,CX	AX ← AX – CX - C	X	X	X	X	X	X
DEC [BX]	DEC BPTR[BX] DEC WPTR[BX]	X	X	X	X	X	.
DEC [SI+1]	DS :[SI + 1] ← DS:[SI + 1] - 1	X	X	X	X	X	.
DAS(**)	Decimal Adjust for Substraction	?	X	X	X	X	X
CMP AX,4	AX – 4	X	X	X	X	X	X
NEG BX	BX← 0-BX	X	X	X	X	X	X

Remarques:

1- (*) Concerne AL uniquement. Cette instruction est utilisée pour ajuster les résultats issus de l'addition de nombres BCD (Binary Coded Decimal). Dans ce cas, nous ajoutons 6 à un quartet s'il est supérieur à 9 ou si C = 1ou A = 1.
2- (**) Concerne AL uniquement. Cette instruction est utilisée pour ajuster les résultats issus de soustraction de nombres BCD (Binary Coded Decimal).

Dans ce cas, nous soustrayons 6 à un quartet s'il est supérieur à 9 ou si C = 1ou A = 1.

Exemples:

1) (*)

$$\begin{array}{rr} & 29\ 0010\ 1001 \\ + & 35\ 0011\ 0101 \\ \hline & 5E\ 0101\ 1110 \\ E>9 + & 06\ 0000\ 0110 \\ \hline & 64\ 0110\ 0100 \end{array}$$

2) (**)

$$\begin{array}{rr} & 65\ 0110\ 0101 \\ - & 27\ 1101\ 1001 \\ \hline & 3E\ 0011\ 1110 \\ E>9+ & 06\ 1111\ 1010 \\ \hline & 38\ 0011\ 1000 \end{array}$$

II.2.4 Multiplication

Tableau 7 Instructions de multiplication

Mnémoniques	Description	O	S	Z	A	P	C
MUL BX (non signée)	DX o AX←AX x BX	X	.	.	X	.	X
IMUL CX (signée)	DX o AX ← AX x CX	X	X	.	X	.	X
MUL [SI]	MUL BPTR[SI] AX ← AL x DS:[SI] MUL WPTR DS:[SI] DX o AX ← AX x DS:[SI]	X	.	.	X	.	X

II.2.5 Division

Tableau 8 Instructions de division

Mnémoniques	Description	O	S	Z	A	P	C
DIV BX (non signée)	DX o AX ÷ BX→DX o AX	Indéfinis					
IDIV BX (signée)	DX o AX ÷ BX→ DX o AX						
DIV [DI]	DIV BPTR[DI] AH o AL ÷ DS:[DI] →AH o AL DIV WPTR DS:[DI] DX o AX ÷ [DI] → DX o AX						

II.2.6 Conversion

Tableau 9 Instructions de conversion

Mnémoniques	Description	O	S	Z	A	P	C
CBW	AL → AH o AL	Aucun n'est affecté					
CWD	AX →DX o AX						

II.3 Instructions logiques

II.3.1 Complément à 1 et décalage

Tableau 10 Instructions de complément à 1 et de décalage

Mnémoniques	Description	O	S	Z	A	P	C
NOT AX	AX ← $\overline{AX}$	.	.	.	.	.	.
SHL AL,1	C ← AL ← 0	X	X	X	.	X	X
SHR AX,1	0 → AX → C	X	X	X	.	X	X
SAR DX,1	→ S AX → C	X	X	X	.	X	X
SAR BX,CL	Si le nombre de décalage >1	X	X	X	.	X	X
SHR [DI],1	0 → DS :[DI] → C	X	X	X	.	X	X

II.3.2 Rotation

Tableau 11 Instructions de rotation

Mnémoniques	Description	O	S	Z	A	P	C
ROL AH,1	C ← AH ←	X	.	.	.	.	X
ROR BH,1	→ BH → C	X	.	.	.	.	X
RCL AX,1	← C ← AX ←	X	.	.	.	.	X
RCR [SI],1	→ DS :[SI] → C	X	.	.	.	.	X
RCR DX,CL	Si le nombre de rotation est supérieur à 1	X	.	.	.	.	X

II.3.3 Opérations logiques fondamentales

Tableau 12 Instructions logiques

Mnémoniques	Description	O	S	Z	A	P	C
AND CL,DL	CL ← CL AND DL	0	X	X	.	X	0
AND AX,FE00H	AX ← AX AND FE00H	0	X	X	.	X	0
OR AL,AL	AL ← AL OR AL	0	X	X	.	X	0
XOR AX,AX	AX ← AX ⊕ AX	0	?	1	.	X	0
XOR BX,[SI]	BX ← BX ⊕ DS:[SI]	0	X	X	.	X	0
TEST AL,80H	Si bit 7 de AL =1⇒ Z=0 Si bit 7 de AL =0⇒ Z=1	0	X	X	.	X	0
AND AL,80H	AL AND 80 → AL	0	X	X	.	X	0

II.4 Instructions de manipulation des chaînes de caractères

Tableau 13 Instructions de manipulation des chaînes de caractères

Mnémoniques	Description	O	S	Z	A	P	C
LODSB	AL ← DS:[SI] D= { 0 SI ← SI + 1 ; 1 SI ← SI - 1 }	Aucun n'est affecté					
LODSW	AX ← DS:[SI] D= { 0 SI ← SI + 2 ; 1 SI ← SI - 2 }						
STOSB STOSW	AL → ES:[DI] AX → ES:[DI] D= { 0 DI ← DI + 1 (ou + 2) ; 1 DI ← DI – 1 (ou +2) }						
MOVSB MOVSW	DS:[SI] →ES:[DI] D = { 0 { SI ← SI + 1(ou +2) ; DI ← DI + 1(ou +2) } ; 1 { SI ← SI - 1 (ou +2) ; DI ← DI – 1 (ou+ 2) } }						
SCASB SCASW	AL – ES:[DI] AX – ES:[DI] D= { 0 DI ← DI + 1 (ou +2) ; 1 DI ← DI - 1 (ou +2) }						

Mnémoniques	Description	O	S	Z	A	P	C
CMPSB CMPSW	DS :[SI] - ES :[DI] D = 0 { SI ← SI + 1(ou +2) ; DI ← DI + 1(ou +2) } D = 1 { SI ← SI - 1 (ou +2) ; DI ← DI – 1 (ou +2) }						

II.4.1 Préfixes

Tableau 14 Préfixes

Mnémoniques	Description du préfixe
REP	Répète tant que CX ≠0
REPE/REPZ	Répète tant qu'il y a égalité et CX≠0
REPNE/REPNZ	Répète tant qu'il n'y a pas égalité et CX≠0

Remarques:

1- Les préfixes ne peuvent exister par eux-mêmes. Ils peuvent être utilisés uniquement avant les instructions de manipulation de chaines de caractères.
2- Selon le Tableau 14, l'exécution de chaque préfixe décrémente d'une unité CX.

II.5 Instructions de branchements

Il existe trois types de branchements dans le 8086.

1- Les branchements inconditionnels.
2- Les branchements conditionnels.
3- Les appels de sous programmes.

II.5.1 Branchements inconditionnels

Définitions des branchements inconditionnels: Il y a trois catégories de branchements ou sauts inconditionnels.

JUMP SHORT: Lors d'un saut court de type intra segment, le déplacement δ est défini par l'intervalle suivant:

$-2^7 \leq \varnothing \leq 2^7-1$ ou $-128 \leq \varnothing \leq 127$ (IP ← IP + 2 + $\varnothing$)

JUMP NEAR: *Lors d'un* saut rapproché de type intra segment, le déplacement $\varnothing$ est défini par l'intervalle suivant:

$-2^{15} \leq \varnothing \leq 2^{15}-1$ ou $-32768 \leq \varnothing \leq 32767$ (IP ← IP + 3 + $\varnothing$)

JUMP FAR: Saut éloigné de type inter segment. *Dans ce cas,* le couple CS:IP est affecté et prennent de nouvelles valeurs.

Tableau 15 Instructions de branchements inconditionnels

Mnémoniques	Type de saut	Description
JMP FFH	Direct relatif à IP: NEAR ou SHORT	IP ← 1D00H
JMP DI	Direct intra segment: NEAR	IP ← DI
JMP [DI] ou JMP WORD PTR [DI]	Indirect intra segment: NEAR	IP ← DS:[DI]
JMP 2345:6789	Direct inter segment: FAR	CS ← 2345 IP ← 6789
JMP FAR [SI] Ou JMP DWORD PTR [SI]	Indirect inter segment: FAR	IP ← DS:[SI] CS ← DS:[SI+2]

Remarque: Le Tableau 16 résume tous les cas d'ambiguïtés pouvant survenir lors du calcul du déplacement $\varnothing$. La grande police de caractères représente les quartets, la petite les bits et x=Don't care.

Tableau 16 Levées d'ambiguïtés dans le calcul du déplacement $\varnothing$

Longueur du déplacement $\varnothing$ (bits)		Type de saut
0 0 1xxx xxxx	16 bits	NEAR
0 0 0xxx xxxx	8 bits	SHORT
F F 1xxx xxxx	8 bits	SHORT
F F 0xxx xxxx	16 bits	NEAR

Exemple: Soit à calculer le déplacement $\varnothing$ résultant de l'instruction JMP FFH. Comme ce type d'instruction est relatif au contenu du registre IP, il est donc nécessaire de connaitre la valeur de ce dernier.

Pour ce faire, le Tableau 17 résume quelques exemples de calcul de $\varnothing$ en fonction de IP. Notons que pour IP= 0000H et 0100H, nous avons eu recours au Tableau 16 pour lever les ambiguïtés.

Tableau 17 Exemples de calcul du déplacement ∅

Instruction	IP	Résultats	Ambiguïté	∅	Type de saut
JMP FFH	0000H	00FCH	Oui	00FCH	NEAR
	0100H	FFFDH	Oui	FDH	SHORT
	1000H	F0FCH	Non	F0FCH	NEAR

II.5.2 Branchements conditionnels

Rappel sur l'arithmétique non signée et l'arithmétique signée: La comparaison de deux nombres non signés entraîne l'utilisation des termes 'Supérieur et Inférieur' (Above and Bellow), tandis que la comparaison de deux nombres signés entraîne l'utilisation des termes 'Plus grand et Plus petit' (Greater and Less). Nous obtenons donc le Tableau 18.

Tableau 18 Comparaison de 2 nombres en arithmétique signée et non signée

Condition	Non signée				Signée				
	O	S	Z	C	O	S	Z	C	S⊕O
Op1 > Op2	.	.	0	0	X	X	0	.	0 (s=0)
Op1 = Op2	.	.	1	0	0	0	1	.	0 (s=0)
Op1 < Op2	.	.	0	1	X	X	0	.	1 (s≠0)

Mnémoniques de branchements conditionnels

Tableau 19 Branchements conditionnels

Condition	Non signée	Signée
Op1 >Op2	JA/JNBE/JNC	JG/JNLE
Op1 ≥O p2	JAE/JNB	JGE/JNL
Op1 = Op2	JE/JZ	JE/JZ
Op1 < Op2	JB/JNAE/JC	JL/JNGE
Op1 ≤ Op2	JBE/JNA	JLE/JNG
Op1 ≠ Op2	JNE/JNZ	JNE/JNZ

Remarques:

1- Les instructions de sauts conditionnels suivent généralement une instruction de comparaison (CMP) de deux nombres signés ou non signés.
2- Un branchement conditionnel est de type short uniquement. Son mot instruction s'écrit toujours sur deux octets (IP← IP+2+∅). Le premier constitue le code opératoire et le second, la valeur du saut.

Instructions de branchements conditionnels

Tableau 20 Instructions de branchements conditionnels

Condition de branchement	Flag	non signé	signé	O	S	Z	P	C	Opération logique
Op1 = Op2	JE/JZ					1			Test de Z
Op1 ≠ Op2	JNE/JNZ					0			Test de Z
Overflow	JO			1					Test de O
No overflow	JNO			0					Test de O
Signe < 0	JS				1				Test de S
Signe >0	JNS				0				Test de S
Parité paire	JP/JPE						1		Test de P
Parité impaire	JNP/JPO						0		Test de P
Op1 > Op2		JA/JNBE/JNC				0		0	$\overline{C}$ et $\overline{Z}$
			JG/JNLE	X	X	0			O=S et $\overline{Z}$
Op1 ≥ Op2		JAE/JNB				1		0	$\overline{C}$ ou Z
			JGE/JNL	X	X	1			O=S ou Z
Op1 < Op2		JB/JNAE/JC				0		1	C et $\overline{Z}$
			JL/JNGE	X	X	0			O≠S et $\overline{Z}$
Op1 ≤ Op2		JBE/JNA				1		1	C ou Z
			JLE/JNG	X	X	1			O≠S ou Z

II.5.3 Instructions de contrôle des itérations

Tableau 21 Instructions de contrôle des itérations

Mnémoniques	Description
LOOP adresse	CX ← CX-1 et saut tant que CX ≠ 0 IP← IP + déplacement étendu sur 16 bits
LOOPE/LOOPZ adresse	CX ← CX-1 et saut tant que CX ≠ 0 et Z=1 à IP← IP + déplacement étendu sur 16 bits
LOOPNE/LOOPNZ adresse	CX ← CX-1 et saut tant que CX≠0 et Z=0 à IP←IP + déplacement étendu sur 16 bits
JCXZ adresse	Saut si CX = 0 à IP← IP+ déplacement sur 16 bits

Remarque: Si initialement CX = 0, alors une décrémentation de '1' le fait passer à CX=FFFFH.

II.5.4 Les appels de sous programmes

Appels indirects (interruptions): Dans un contexte PC (MS-DOS), La table des vecteurs d'interruptions (TVI) occupe le premier Kilo octets de la mémoire vive i.e., les adresses 00000H-0003FFH. Comme cela est illustré en Figure 1, la TVI

est organisée en 256 couples CS:IP correspondant chacun à un vecteur d'interruption (VI) totalisant ainsi 256. La syntaxe d'une instruction d'interruption est donnée par:

INT N° VI $0 \leq$ N° VI ≤ 255, où N° VI est le Numéro du Vecteur d'Interruption

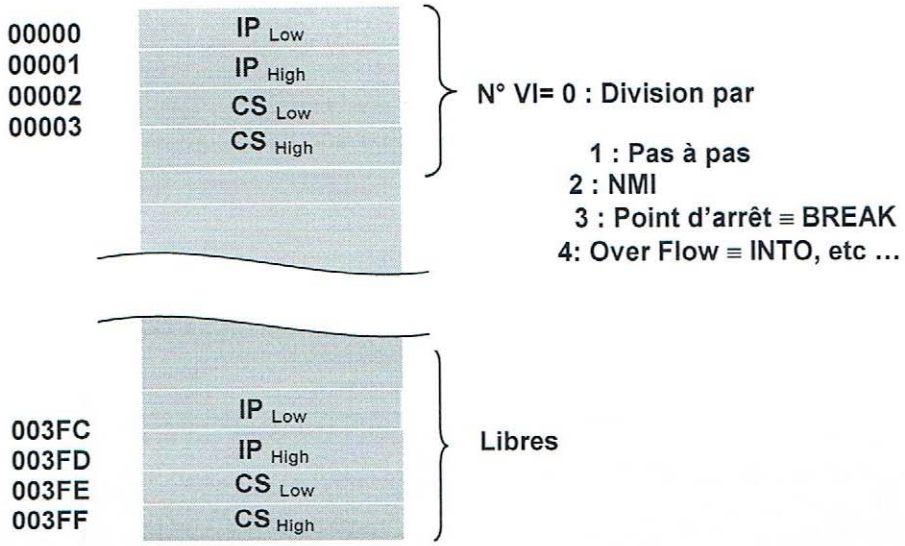


Fig. 1 Table des vecteurs d'interruption dans le 8086

Mécanisme des interruptions: Avant l'exécution d'un sous programme (appelé aussi routine ou procédure) d'interruption, le 8086 doit d'abord sauvegarder la valeur courante de CS:IP dans le segment pile et ce afin d'utiliser la valeur du VI pour retrouver et charger dans CS:IP la nouvelle valeur contenue dans la TVI. Pour cela, le 8086 exécute, par exemple, pour l'instruction INT 04H, le mécanisme suivant:

INT 04H (Type=N° VI=04H)

1- Type x 4 = 10H; calcul de l'adresse 00010H de la TVI.
2- Sauvegarde du registre F dans la pile, Cf. Figure 1.
3- T= I = 0 ; Inhibition des interruptions et du mode d'exécution pas à pas.
4- Sauvegarde du registre CS dans la pile, Cf. Figure 1.
5- CS ← (type x 4 + 2 = 12H) ; chargement de CS avec la valeur pointée par 00012H dans la TVI.
6- Sauvegarde du regitre IP dans la pile, Cf. Figure 1.
7- IP ← (type x 4 = 10H) ; chargement de IP avec la valeur pointée par 00010H dans la TVI.

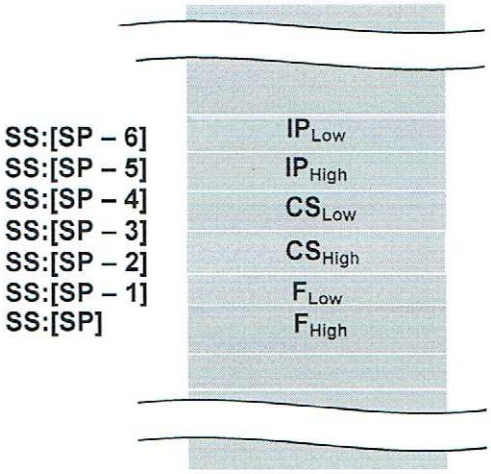


Fig. 2 Empilements de F, CS et IP dans le segment pile

Remarque: Tout sous programme d'interruption doit se déterminer par une instruction IRET. Après l'exécution de l'IRET, les registres IP, CS et F sont restitués et l'exécution du programme redevient normale.

Appels direct de sous programme: Les appels de sous programmes se font, généralement, à travers l'instruction CALL. Cette dernière peut effectuer un appel de type NEAR intra segment ou de type FAR inter segment. Le Tableau 22 décrit quelques exemples d'appels de sous programmes.

Tableau 22 Instructions d'appels de sous programmes

Mnémoniques	Type de saut	Description
CALL 1234	Direct intra segment: NEAR	IP ← 1234
CALL 1234:ABCD	Direct inter segment: FAR	CS ← 1234 IP ← ABCD
CALL [SI] ou CALL WORD PTR [SI]	Indirect intra segment: NEAR	IP ← DS :[SI]
CALL SI	Direct intra segment: NEAR	IP ← SI
CALL WORD PTR[BX] ou CALL [BX]	Indirect intra segment: NEAR	IP ← DS :[BX]
CALL DWORD PTR [BX] Ou CALL FAR [BX]	Indirect inter segment: FAR	CS ← DS :[BX] IP ← DS :[BX+2]

Remarques: L'instruction CALL permet la sauvegarde automatique du registre IP dans le cas d'un CALL NEAR et des registres CS et IP, dans le cas d'un CALL FAR.

Tout sous programme doit se déterminer par une instruction RET dans le cas d'un CALL NEAR et par RETF ou RET FAR dans le cas d'un CALL FAR.

Après l'exécution de RET (RETF), le registre IP (IP et CS) est (sont) restitué (s) et l'exécution du programme redevient normale.

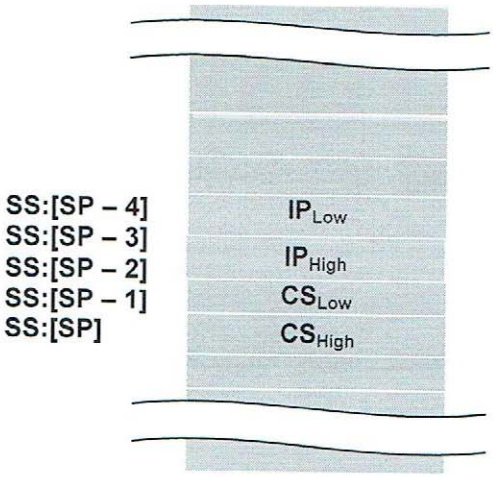


Fig. 3 Empilements de CS et IP dans le segment pile à la suite d'un CALL FAR

II.6 *Instructions de contrôle du 8086*

II.6.1 Opération sur les flags

Tableau 23 Instructions de contrôle des flags

Mnémoniques	Description
CLC	C ← 0
CMC	C ← $\overline{C}$
STC	C ← 1
CLD	D ← 0
STD	D ← 1
CLI	I ← 0
STI	I ← 1

II.6.2 Arrêt du 8086: HLT

Pour sortir d'un état d'arrêt HALT, il faut un RESET ou une interruption matérielle NMI (type 2) ou INTR (I = 1)

II.6.3 Attente du 8086: WAIT

Après l'exécution de l'instruction WAIT, le 8086 teste $\overline{TEST}$. Tant que $\overline{TEST}$=1, le 8086 reste dans un état d'attente.

II.6.4 Echappement du 8086: ESCape to an external device

L'instruction ESC dont le code opératoire débute par 1BH (11011xxx) désigne toutes les instructions destinées au coprocesseur mathématique 8087. Les mnémoniques de ces instructions comment par F (FAST). Dans ce cas, si l'opérande est un registre de 16 bits, l'instruction ESC est équivalente à une instruction NOP (No Operation). Sinon, si l'opérande est une case mémoire, le 8086 calcule l'adresse physique pour le compte du 8087 du 8086.

Enfin, l'instruction ESC équivalente à une instruction NOP pour un système à base de 8086 ne disposant pas 8087. Notons que 30 coups d'horloge équivalent à 10 NOP.

Exemples:

ESC 28,AX ≡ FNENI = M = 0 du 8087; validation des interruptions du 8087.
ESC 28,CX ≡ FDISI = M = 1 du 8087 ; inhibition des interruptions du 8087.
ESC 01,[SI] ≡ FMUL DWORD PTR [SI]
ESC 15,DX ≡ FSQRT
ESC 06,[DI] ≡ FDIV DWORD PTR[DI]

Où '≡' désigne l'opérateur équivalence. Comme le 8086 peut lire les instructions plus vite que le 8087 ne peut les exécuter, nous devons maintenir le 8086 en synchronisation avec son coprocesseur comme le montre l'exemple suivant:

```
FWAIT ; Si l'instruction précédente est destinée au coprocesseur.
FMUL  DWORD PTR [SI]
FWAIT                    ≡ WAIT
FDIV  DWORD PTR [DI]
FWAIT ≡ WAIT
```

II.6.5 Préfixe LOCK

Comme nous l'avons déjà dit précédemment, ce préfixe agit sur le signal $\overline{\text{LOCK}}$ du 8086. Il ne peut pas exister de lui-même. Il doit être associé à une instruction permettant un accès mémoire ou E/S dans une configuration multi-maître dont les processeurs partagent les mêmes ressources mémoire et E/S.

Exemple:

```
LOCK   XCHG  AX,[DI]
```

III Modes d'adressages dans le 8086

Le 8086 dispose des modes d'adressages suivants:

1- Immédiat
2- Direct
3- Basé
4- Indexé
5- Indexé et basé
6- Basé et déplacement
7- Indexé et déplacement
8- Basé, indexé et déplacement
9- Registre

III. 1 Mode d'adressage immédiat

III.1.1 Définition

Registre/Mémoire ← valeur sur 8 ou 16 bits.

Exemples:

MOV AL,08H	AL ← 08H
MOV BX,F0ABH	BX ← F0ABH

III.2 Mode d'adressage direct

III.2.1 Définition

Registre ⇔ Mémoire

Exemples:

ID1	DB 02H	DB: Define Byte 02H
ID2	DW 3000H	DW: Define Word 3000H
ID3	DB ?	Define a Byte
ID4	DD 42FF5678H	DD: Define Double Word 42FF5678H
ID5	DB 05H DUP (?)	Define 5 Bytes

où IDi est une adresse sur 20 bits qui s'écrit sous la forme segment:offset

```
MOV  AL,ID1        AL← (ID1)
MOV  BP,ID2        BP← (ID2)
MOV  ID3,AH        (ID3)← AH
MOV  ID5,BX        (ID5)← BX
```

III.3 Mode d'adressage basé

III.3.1 Définition 1: Adressage basé sur DS

Registre ⇔ DS:[BX]

Exemples:

```
MOV  BX,08H        BX ← 0008H
MOV  AX,DS :[BX]   AX ← DS:[0008H]
LEA  BX,ID2        BX ← OFFSETID2
MOV  AX,DS:[BX]    AX ← DS:OFFSET ID2=3000H
```

III.3.2 Définition 2: Adressage basé sur SS

Registre ⇔SS: [BP]

Exemples:

```
MOV  BP,2020H      BP ← 2020H
MOV  DX,[BP]       DX ← SS:[2020H]
```

III.4 Mode d'adressage indexé

III.4.1 Définition

Registre ⇔ [SI] ou [DI]

Exemples:

```
MOV  SI,0000H      SI  ← 0000H
MOV  AX,[SI]       AX← DS:[SI]
MOV DI,04H         DI  ← 04H
MOV ES:[DI],AH     ES:[0004H] ← AH
```

III.5 Mode d'adressage indexé et basé

III.5.1 Définition 1: Adressage basé sur DS

Registre ⇔ [BX] [SI] ou [BX] [DI]

Exemples:

```
MOV  BX,2F1EH        BX ← 2F1EH
MOV  SI,0000H        SI ← 0000H
MOV  DX, [BX][SI]    DX ← DS:[BX+SI]
```

III.5.2 Définition 2: Adressage basé sur SS

Registre ⇔ [BP] [SI] ou [DI]

Exemples:

```
MOV  BP,212BH        SI ← 212BH
MOV  DI, 3000H       DI ← 3000H
MOV  AX,[BP] [DI]    AX ← SS:[BP+DI]
```

III.6 Mode d'adressage basé et déplacement

III.6.1 Définition 1: Adressage basé sur DS

Registre ⇔ Déplacement [BX]

Exemples:

```
ID1   DB   02H          Define Byte 02H
ID2   DW   5000H         Define Word  5000H
      MOV  BX,2121H      BX ← 2121H
      MOV  CX,ID2[BX]    CX ← DS:[OFFSET ID2+BX]
```

III.6.2 Définition 2: Adressage basé sur SS

Registre ⇔ Déplacement [BP]

Exemples:

```
ID1   DB   05H DUP (?)      Define 5 bytes
      MOV  BP,0002H         BP ← 0002H
      MOV  ID1[BP],AX       AX → le 3ème et le 4ème éléments de id1
```

III.7 Mode d'adressage indexé et déplacement

III.7.1 Définition

Registre ⇔ déplacement [SI] ou [DI]

Exemples:

```
MOV  SI,0001H              SI ← 0001H
MOV  AX, 0005H[SI]         AX ← DS :[SI+5]
```

III. 8 Mode d'adressage basé, indexé et déplacement

III.8.1 Définition 1: Adressage basé sur DS

Registre ⇔ déplacement [BX] [SI] ou [DI]

Exemples:

```
ID1   DB   02H              Define byte 02H
ID2   DW   5000H            Define word 5000H
      MOV  BX, 0005H        BX ← 0005H
      MOV  SI, 0004H        SI ← 0004H
      MOV  AX,ID2 [BX][SI+1]  AX ← DS :[5+4+1+Offset ID2]
```

III.8.2 Définition 2: Adressage basé sur SS

Registre ⇔ déplacement [BP] [SI] ou [DI]

Exemples:

```
ID1   DB   02H              Define byte 02H
ID2   DW   5000H            Define word 5000H
      MOV  BP,0005H         BP ← 0005H
```

```
MOV  SI,0004H          SI ← 0004H
MOV  CX,ID2 [BP][SI+1] CX ← SS :[5+4+1+Offset ID2]
```

III.9 Mode d'adressage registre-registre

III.9.1 Définition

Registre 8 ou 16 bits ⇔ Registre 8 ou 16 bits

Exemples:

```
MOV  AX,BX     AX ← BX
MOV  DH,AL     DH ← AL
```

IV Accès à des segments multiples: Segment Override

La manière dont nous avons présenté les modes d'adressages utilise implicitement les registres segments DS ou SS. A titre d'exemple, lorsque BX, DI ou SI joue le rôle d'offset, le segment est extrait du registre DS, sauf dans le cas des instructions répétitives où le segment est extrait de ES pour l'offset DI; au contraire, lorsque BP joue le rôle d'offset, l'adresse physique est calculée à partir du segment SS. Cette manière de procéder entraîne que le registre ES et CS ne sont jamais pris en compte. Il arrive que ce besoin se fasse sentir lors d'un transfert entre deux segments autre que DS et SS.

Les programmes TSR: Terminate and Stay Resident, font souvent appel à un segment override ou dépassement de segment ou l'accès se fait à travers le registre code segment CS. Les pilotes résidants, drivers résidants ou TSR, sont des programmes qui, une fois installés, demeurent en mémoire et peuvent à tout moment être activés. Ils sont appelables par des interruptions matérielles ou logicielles tels que PRINT (Spooler d'impression en DOS), GRAPHICS (Recopie d'écran) et MOUSE (Gestion de la souris)

De ce fait, Intel a traité et résolu de manière simple ce problème sous la forme de ce qu'on nomme par un segment override. Ce dépassement de segment se matérialise par un préfixe de segment, c'est à dire un code que nous pouvons mettre devant toute instruction assembleur et qui désigne explicitement le registre de segment à utiliser pour former l'adresse physique.

Exemples:

```
MOV  AX,DS:[BP+DI]
MOV  AL,ID1
MOV  AX,ES :[BX+DI]
MOV  AL,SS:[BP]
MOV AL,CS:[BX]
```

L'octet du préfixe du segment override s'écrit sous la forme: 001 reg 110 où reg prend les valeurs suivantes:

00	pour	ES	10	pour	SS
01	pour	CS	11	pour	DS

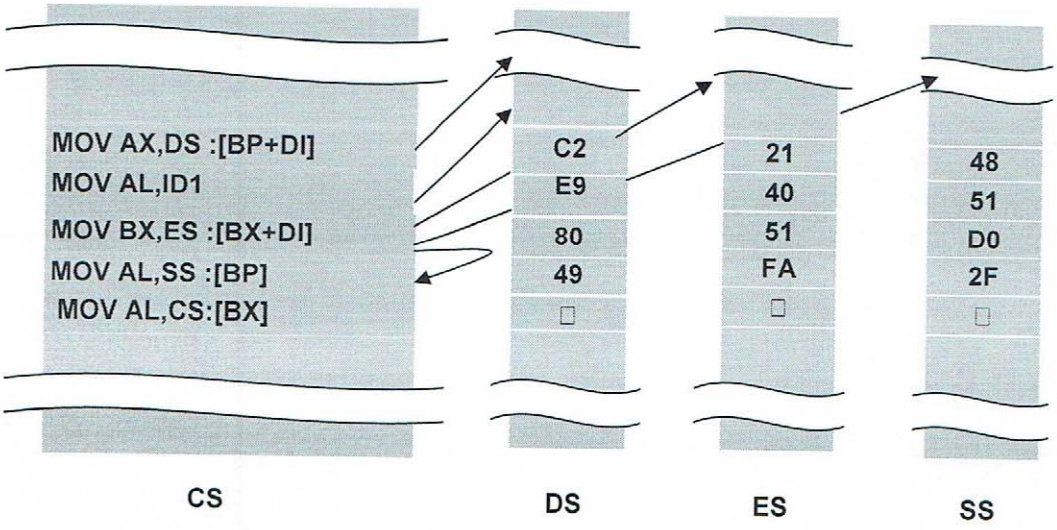


Fig. 4 Dépassement de segment

V Assembleur du 8086

V.1 Définitions

Un programme assembleur joue le rôle de traducteur du code source en code machine.

Fig. 5 Transposition code ASCII →HEXA

Dans la transposition code source en code machine, il existe deux formes d'assembleurs. Le premier travaille en un seul passage : Assembleur monopasse ou assembleur ligne. L'autre travaille en deux passes (assembleur 2 passes). L'assembleur monopasse parcourt une seule fois le code source tandis que l'assembleur deux passes utilise le 1er passage pour créer en RAM une table contenant toutes les étiquettes, variables et données avec leurs adresses respectives. Ceci est accompli en comptant le nombre d'octets de chaque instruction. Le code objet est crée au moment de la 2ème passe.

Remarques:

1- Dans un assembleur 2 passes, toutes les erreurs sont signalées à la fin de la $2^{ème}$ passe.
2- Dans un assembleur 2 passes, une instruction de saut réserve deux octets pour le saut. S'il s'agit d'un saut SHORT alors le $2^{ème}$ octet est remplacé par un NOP. S'il s'agit d'un saut NEAR l'assembleur ne fait rien. S'il s'git d'un saut far l'assembleur détecte une erreur. Auquel cas il faut spécifier un JMP FAR.

3- Un listing peut être fourni par l'assembleur (assembleur 3 passes), contenant en plus des instructions, leurs codes machine respectifs et les différentes informations concernant le nombre d'erreurs et leurs types, le nombre de lignes sources et le nombre d'étiquettes avec leurs adresses respectives.

V.2 Exemples d'assembleurs

Il existe plusieurs types d'assembleurs sur le marché. Citons, par exemple le débogueur sous MSDOS, le turbo assembleur, le turbo débogueur, l'ASM (Assembleur), le MASM (Macro assembleur) et les mulateurs et simulateurs. Dans ce qui suit, nous prendrons le cas de l'ASM pour montrer les différentes étapes par lesquelles nous devons faire passer un fichier source pour en faire un fichier exécutable à la manière de Microsoft.

1- Le code source est crée à partir d'un éditeur de texte. Il est noté essai.asm. L'assemblage de essai.asm se fait comme suit:

C: \> asm essai ↵

L'assemblage de essai.asm donne un fichier.obj. Dans notre cas essai.obj. En option, nous pouvons demander la création du fichier essai.lst sinon Null.lst sera crée par défaut. Ce fichier contient le listing du programme source et celui du code machine correspondant. Si le programme ne contient pas d'erreurs le message suivant s'affiche sur l'écran: Warning Errors 0 and Severe Errors 0.

2- Comme essai.obj est inutilisable par le DOS, il faut donc utiliser un éditeur de liens ou LINKER. Celui-ci transforme un fichier.obj en un fichier.exe comme suit :

C : \> link essai.obj essai1.obj essai2.obj ..... ↵

L'éditeur de liens crée donc essai.exe à partir un programme principal et de plusieurs sous programmes.

3- Création d'un fichier.com, commençant à l'offset 0100H qui s'exécute à la manière de Microsoft. L'utilitaire exe2bin crée donc essai.com de la manière suivante:

C : \> exe2bin essai essai.com ↵

Enfin, l'exécution de essai.com à la manière de Microsoft se fait comme suit:

C : \> essai ↵

Les étapes précédentes peuvent être résumées comme suit:

Fig. 6 Etapes pour le passage du code source au code exécutable

Afin de ne pas reprendre toutes ces étapes à chaque fois qu'il y a création d'un fichier source, nous créons un fichier batch de la manière suivante:

alc.bat = { asm %1;
link %1.obj;
exe2bin %1 %1.com

où %1 représente le nom du fichier. Soit donc:

C : \> alc essai ↵

Remarques:

1- Les sous programmes sont automatiquement lus au cours du 'LINK'.
2- Comme nous l'avons dit précédemment, un programme dont l'offset à la valeur 0100H a une extension '.com' (CS:0100H). La terminaison d'un tel programme s'effectue toujours avec l'interruption INT 21H, fonction 4CH ou INT 20H, avec

retour au MSDOS. Les 100 H premiers octets, d'offset 0000H à 00FFH (CS:0000H à CS:00FFH), sont réservés au PSP (Prefix Segment Program).

VI Structure d'une ligne assembleur

Une ligne assembleur comporte 4 champs. Ils sont définis dans le Tableau 22.

Tableau 22 Structure d'une ligne d'assembleur

Champ Etiquette	Champ opération	Champ opérande	Champ commentaire
Tous les Caractères Alphanumériques de '0 à 9' et de 'a à z', tous les caractères spéciaux '-, £, $, §', ... Il faut éviter: Les noms de registres, les noms significatifs à l'assembleur et les directives d'assemblage Exemple: ADR-1	Opérations à Effectuer MOV ADD MUL ⋮ Exemple: MOV	Registre et Mémoire Exemple: AX,BX	Tout commentaire doit précédé par un ' ;' Exemple: ; Chargement de AX par BX

VII Exemples de programmation du 8086

VII.1 Premier exemple

Soit à réaliser un programme qui traduit la valeur reçue en Hexadécimal par le port 1 en code Gray sur 4 bits et l'émet par le port 2 (Cf... Figure 7). Les adresses des ports 1 et 2 sont 01H et 02H, respectivement.

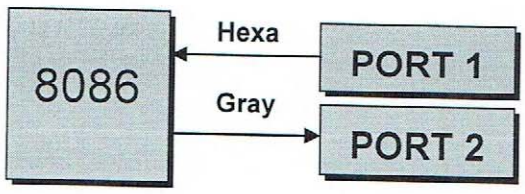


Fig. 7 Synoptique d'un dialogue entre deux ports d'entrées et sorties

Définition du code Gray: Soient g_n, g_{n-1}, ..., g_2, g_1 et g_0 la séquence Gray de la séquence binaire b_n, b_{n-1}, ..., b_2, b_1 et b_0. Les relations qui donnent la séquence en code Gray en fonction de celle en code binaire sont définies par:

$$\begin{cases} g_i = b_i \oplus b_i + 1 & 0 \le i \le n\text{-}2, \text{ où n représente le nombre de bits.} \\ g_n = b_n \end{cases}$$

A titre d'exemple, soit la séquence binaire 101101 (n=6). La séquence correspondante en code Gray s'obtient de la manière suivante:

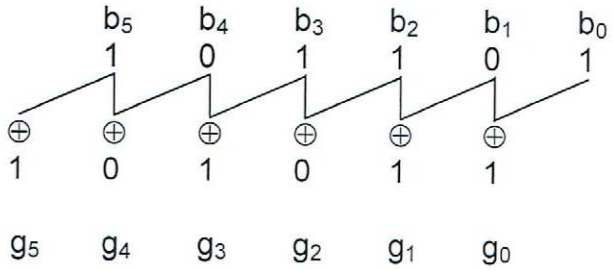


Fig. 9 Exemple de conversion binaire/Gray sur 6 bits

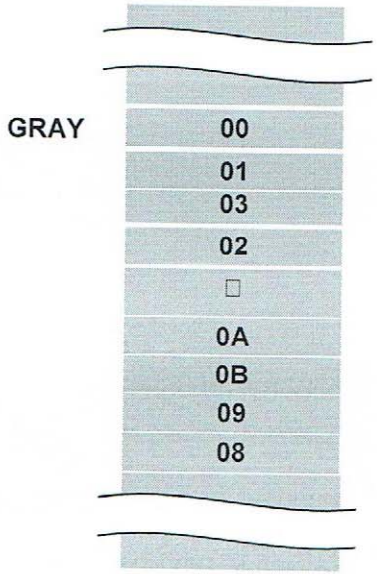


Fig. 10 Table de conversion en mémoire Binaire/Gray sur 4 bits

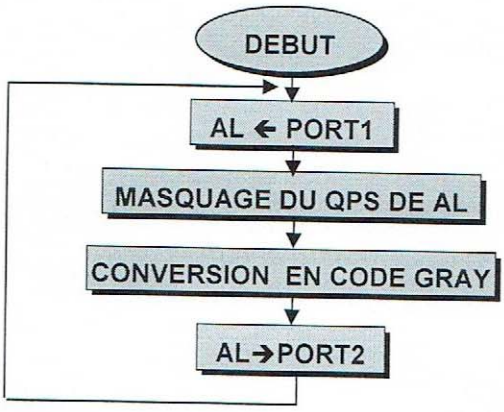


Fig. 11 Organigramme de conversion Binaire/Gray sur 4 bits

```
GRAY    DB   00H,01H,03H,02H,06H,07H,05H,04H,0CH,0DH,0FH, ...
        MOV  AX, DATA ; DATA est le segment de données
        MOV  DS,AX
        LEA  BX,GRAY
NEXT:   IN   AL,01H
        AND  AL,0FH
        XLAT GRAY; AL←DS :[BX+AL]
        OUT  02H,AL
        JMP  NEXT
        HLT
```

Fig. 12. Programme en mnémoniques 8086 de l'organigramme de la fig. 11

VII.2 Deuxième exemple

Soit à réaliser l'addition décimale de deux nombres entrés en ASCII et rangés en mémoire à partir de STRING-1 et STRING-2. Le résultat est rangé à partir de l'adresse du 2ème nombre (STRING-2).

Addition faite manuellement

```
  2571D
+ 4183D
= 6754D
```

Addition faite par le 8086

```
  2571D
+ 4183D
= 66F4H
```

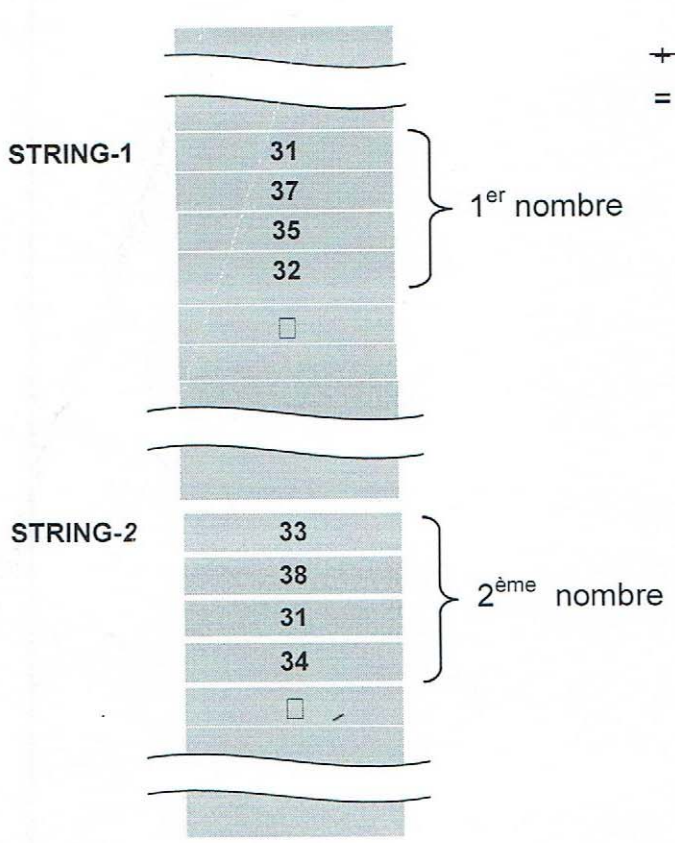


Fig. 13 Représentation en mémoire du 8086 (ASCII) des nombres 2571D et 4183D

Comme le 8086 additionne les nombres en Hexadécimal, le résultat donné par celui-ci est incorrect. Par conséquent, nous devons procéder à une correction décimale. L'instruction indiquée est AAA (ASCII Adjust for Addition). Cette instruction convertit le contenu de l'accumulateur AL en un nombre décimal étendu (unpacked number). Les 4 bits de poids forts du registre AL sont alors mis à zéro. La règle qui corrige AL, après une opération d'addition uniquement, est la suivante:

Si QMS de AL est > 9 ou AF=1 alors: { AL ←AL+06H ; AH←AH+01H ; AL ←AL AND 0FH

OF, SF, ZF et PF sont indéfinis et AF=CF=1.

Si l'opération est sur un octet unique, AH doit être préalablement remis à zéro.

En tenant compte de la règle de correction ci-dessus, l'opération 2571D+4183D se fait de la manière suivante:

```
0   0   1   0   0
    ↖   ↖   ↖   ↖
   32  35  37  31        Addition par octet en hexa avec propagation de carry
+  34  31  38  33
-------------------
=  66  67  6F  64
+  00  00  06  00        Correction ASCII-BCD de chaque octet
-------------------
   66  67  75  64
AND 0F 0F  0F  0F        Masquage du poids fort de chaque octet
-------------------
=  06  07  05  04        Nombre décimal étendu représentant 6754
```

Remarques:

1- Les Nombres décimaux étendus (unpacked numbers) s'écrivent de 0-1-2- ...99 comme suit: 00 00-00 01-00 02 ...01 00 01 01 ... 09 09.
2- Les nombres décimaux compactés (packed numbers) s'écrivent de 0-1-2-...99 comme suit: 00-01-02 ...99.

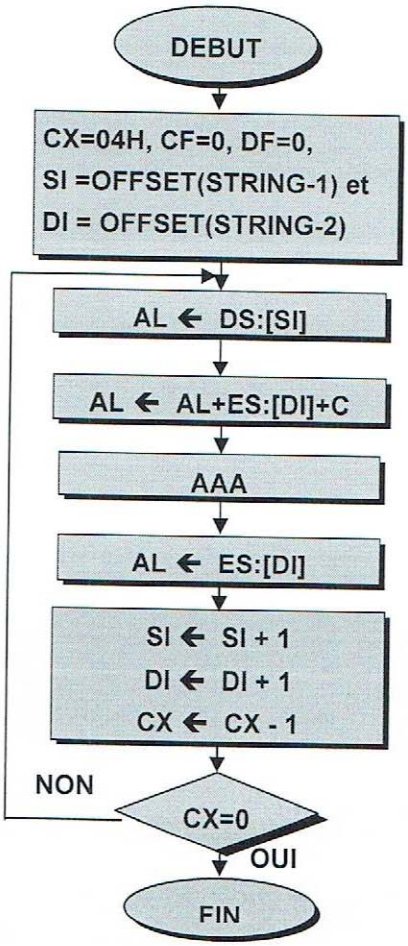


Fig. 14 Organigramme de l'addition de 2 nombres décimaux entrés en ASCII

```
STRING-1   DB    '1','7','5',2'
STRING-2   DB    '3','8','1','4'
           MOV AX,DATA
           MOV DS,AX
           MOV ES,AX
           MOV  CX,04H
           LEA   SI,STRING-1
           LEA   DI,STRING-2
           CLC
           CLD
SUIVANT:   LODSB  STRING-1
           ADC   AL,ES:[DI]
           AAA
           STOSB  STRING-2
           LOOP SUIVANT
           HLT
```

Fig.15 Programme en mnémoniques 8086 de l'organigramme de la Figure 14.

CHAPITRE 6

ORGANISATION LOGICIELLE DANS LE 8086

I Introduction

Dans les deux exemples étudiés dans le chapitre précédent, nous avons transcrit ou converti les organigrammes en mnémoniques sans même se soucier de leur l'emplacement ni celui des données en mémoire. Par essence, la conception du 8086 préconise l'utilisation de segments où chacun d'eux est géré par un registre segment CS, DS, ES ou SS. Une première approche, suggère une organisation telle que nous devons disposer d'un segment unique pour les instructions, un segment unique pour les variables et les constantes et un segment pour la pile. Une telle organisation est dite minimale et est présentée en Figure 1.

II Organisation minimale

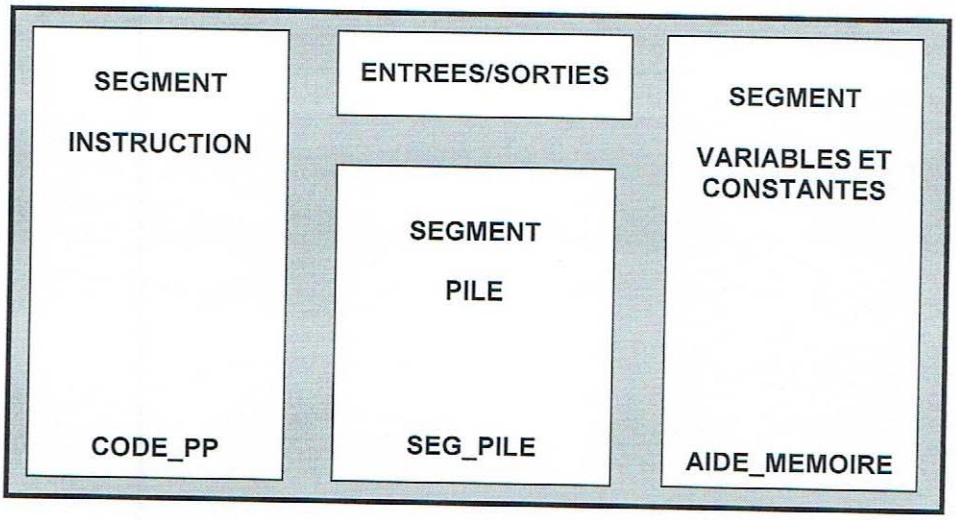


Fig. 1 Organisation minimale

III Organisation structurée

Pour sortir du cadre d'un segment unique pour les instructions et un seul pour les variables et pour les constantes, nous faisons appel à la notion de segments multiples car dans ce cas l'ensemble des instructions et l'ensemble des variables et constantes peuvent dépasser 64 Ko, respectivement.

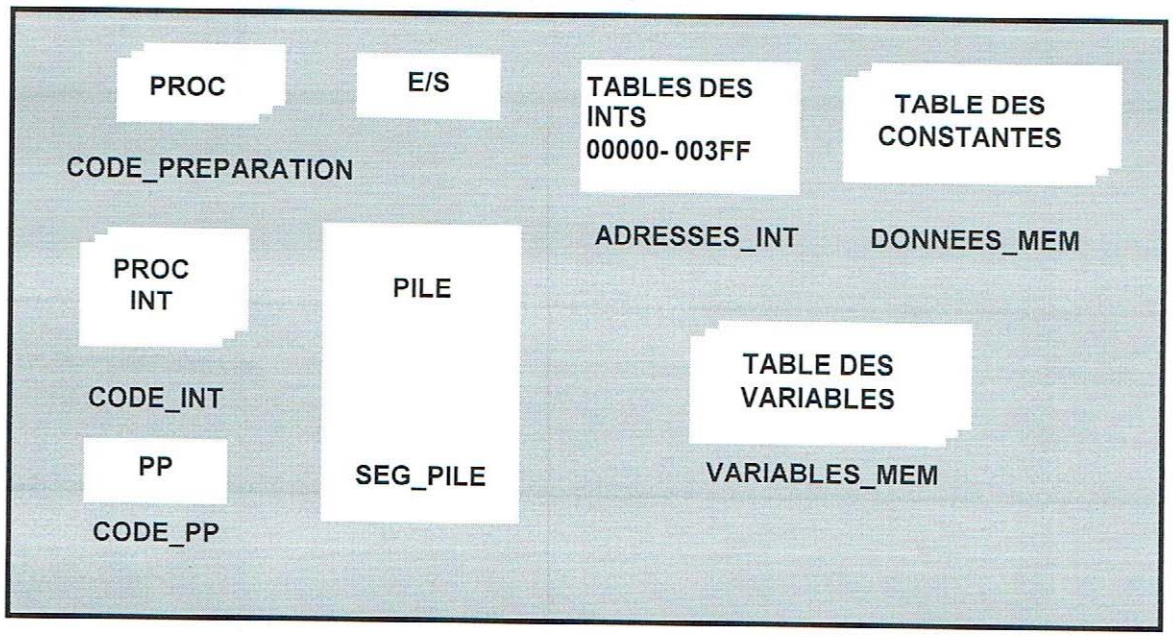


Fig. 2 Organisation structurée: Segments multiples

IV Modules multiples

Pour sortir du cadre du module unique nous devons répartir les différents morceaux du logiciel en plusieurs modules. D'un module à un autre, il est possible de rouvrir un segment déjà créé ou d'en ouvrir un autre.

V Regroupement

Il est clair que l'éparpillement des informations peut créer un problème au programmeur. Pour y répondre l'assembleur propose deux solutions: l'une à vocation logicielle (le groupe) et l'autre à vocation matérielle (la classe).

V.1 Groupe

Chaque segment a son adresse segment propre et pour passer d'un segment à un autre, le 8086 doit modifier l'un de ses registres segments (segment override); Opération peu coûteuse en temps mais elle constitue un handicap inutile quand la somme de tous les segments ne dépasse pas 64 Ko. Dans ce cas, l'assembleur accepte de gérer tous les segments dans un groupe travaillant avec une seule valeur segment pour tous les segments.

V.2 Classe

Indépendamment des considérations logicielles, il peut s'avérer indispensable de grouper les éléments d'information en un ensemble homogène du point de vue matériel. A savoir la mémoire non volatile (ROM) pour les instructions et les

constantes et la mémoire volatile (RAM) pour les tables de paramètres et les variables.

L'ensemble des segments de code réunis en groupe ou non est situé à un emplacement physiquement imposé. Le segment de la pile est généralement placé aux adresses les plus élevées de la mémoire vive. Enfin, les variables sont chargées aux adresses basses de la mémoire vive couvrant ainsi la table des vecteurs d'interruption.

VI Structure d'un programme

Avant de passer en revue quelques pseudo-instructions, nous pouvons déjà faire remarquer que tout ce qui est entre [.] est optionnel et que tout ce qui est entre <.> est obligatoire.

VI.1 Lien entre modules

Pour référencer un module, il existe 3 directives: NAME, PUBLIC et EXTRN

VI.1.1 NAME

Affecte un nom à un module. Ainsi, il permet à l'assembleur d'identifier la table des symboles en cours.

Syntaxe: NAME <NOM DU MODULE>

Exemple: NAME RACINE-CARREE

VI.1.2 PUBLIC

Déclaration en tant que "GLOBAL" d'un symbole ou d'une variable.

Syntaxe: PUBLIC <NOM1>[,<NOM2>,<NOM3>,...]

Tous les symboles et variables utilisés par d'autres modules doivent apparaître dans une directive PUBLIC, sinon ils sont considérés comme locaux et inexistants pour les autres modules.

Exemple:

```
NAME MODULE-A
PUBLIC    SOMME, VALEUR, ADDIT
```

```
        ASSUME   DS: DATA
SOMME   DW       ?
VALEUR  DB       200 DUP ( ?)
ADDIT   PROC     NEAR
```

VI.1.3 EXTRN

Définit les symboles dans le programme source.

Syntaxe: EXTRN <NOM1>:<TYPE1>[:<NOMBRE1>]
[,<NOM2>:<TYPE2>[:<NOMBRE2>]]...

Donne à l'assembleur le type et éventuellement le nombre des symboles suivant l'endroit où ils sont définis. Avant d'employer un symbole pour la première fois en tant que PUBLIC dans un autre module, vous devez le déclarer en tant que EXTRN dans le module en cours.

<NOM>: Nom de la variable ou du symbole.

<TYPE>: QWORD, PWOR,. TBYTE, PROC NEAR, PROC FAR...

[<NOMBRE>]: Nombre de variables.

Exemple:

```
        EXTRN   MEMO:BYTE:100
        EXTRN   ADDIT:ABS
        EXTRN   ROOT:NEAR, SQR:NEAR, MULT:FAR
        EXTRN   ROOT:PROC
        EXTRN   DEMO:WORD
                ADD  AX, DEMO
                CALL SQR
```

Exemple récapitulatif:

```
                NAME    MODULE-A
                PUBLIC  SOMME, VALEUR, ADDIT
DATA            SEGMENT
SOMME           DW      ?
VALEUR          DB      200 DUP(?)
ADDIT           PROC    NEAR
                :
ADDIT           ENDP
DATA            ENDS
CODE            SEGMENT PUBLIC
```

```
                ASSUME  CS:CODE, DS:DATA
                MOV  AX, DATA
                MOV  DS,AX
CODE            ENDS
                NAME     MODULE-B
                EXTRN    SOMME:WORD,VALEUR:BYTE:200, ADDIT:NEAR
CODE            SEGMENT  PUBLIC
                :                .
CODE            ENDS
                END
```

VI.2 Segment

Toutes les instructions et toutes les variables sont contenues dans un segment. S'il en existe pas un, l'assembleur en crée un appelé " ?? SEG". C'est dans celui-ci que seront rangées toutes les instructions ou les variables "égarées" en dehors des segments.

VI.3 Compteur d'emplacements ou compteur d'adresses

Dès qu'un segment est créé, un compteur d'emplacement (CE) lui est alloué par l'assembleur. Ce compteur permet de compter les bytes attribués à chaque information dans le segment. Le comptage s'arrête dès qu'une pseudo-instruction ENDS est rencontrée. Le comptage peut reprendre à la rencontre du segment dans un module ou dans un autre s'il a été déclaré public.

VI.4 Pseudo-instructions ou directives d'assemblage

Les pseudo-instructions constituent un jeu d'instructions complémentaire au jeu d'instruction du 8086. Leur utilisation facilite l'écriture des programmes. Cependant, elles sont reconnues par l'assembleur mais ne sont pas traduites en code objet.

VI.4.1 SEGMENT/ENDS

Définit le début et la fin d'un segment.

Syntaxe:

Structure d'un segment

```
<NOM>     SEGMENT  [<?>][<?>][<?>]
                   .
<NOM>     ENDS
```

Exemple:

```
CODE    SEGMENT
        :
CODE    ENDS
```

Remarque: Le NOM sert à associer une valeur pour le calcul des adresses physiques.

VI.4.2 Directive 'ASSUME'

Affecte un segment à un registre segment

Syntaxe:

ASSUME <NOM DU REGISTRE SEGMENT> :<NOM DU SEGMENT>

<NOM DU REGISTRE SEGMENT> : Un des 4 registres segments CS, DS, SS ou ES

<NOM DU SEGMENT> : Nom du segment ou du groupe

Exemple:

```
ASSUME  CS:CODE, DS:DATA, ES:DATA, SS:PILE
ASSUME  NOTHING (Annule l'affectation)
ASSUME  SS:NOTHING
ASSUME  CS:CGROUP, DS:DGROUP, ES:DGROUP, SS:SGROUP
```

VI.4.3 Directive 'ORIGINE'

A la création du segment CE=0. ORIGINE détermine alors l'emplacement mémoire dans lequel le code assemblé (code objet) sera déposé.

Syntaxe: ORG <DEPLACEMENT SUR 16 BITS>

Exemples:

```
ORG  100H      (CE=0100H)
ORG  $+3       (Valeur courante de CE+3)
ORG  21 H*4    (Interruption 21H,CE=84H)
ORG  DEBUT     (CE=DEBUT désigné par une étiquette)
```

VI.4.4 Directive 'END'

Fin d'un programme. Toute instruction située après END est ignorée durant l'assemblage.

Syntaxe: END [ADRESSE]

Exemples:

```
END
END  START (START est une étiquette à partir de laquelle le programme
            commence à s'exécuter)
```

Exemple récapitulatif

```
00000 CODE      SEGMENT
                ORG  0100H
                ASSUME  CS:CODE, DS:CODE, ES:CODE, SS:CODE
      TIMER     DB  42
                :
      START:    CALL GETKEY
                :
                "PROGRAMME"
                :
      CODE      ENDS
                END  START
```

VI.4.5 Structure d'un segment

<NOM>: Indique le nom du segment auquel est associé la valeur segment pour le calcul des adresses physiques de tous les éléments inscrits entre <NOM> SEGMENT et <NOM> ENDS.

Syntaxe:

```
<NOM>    SEGMENT  [<ALIGNEMENT>] [<TYPE>] ['<CLASSE>']
         :
<NOM>    ENDS
```

[<ALIGNEMENT>]: Optionnel, il indique l'adresse de début d'un segment. Par défaut, l'alignement PARA est utilisé lors de l'édition des liens. Les différents types d'alignements sont: BYTE, WORD, DWORD, PARAGRAPH, PAGE et INPAGE

Tableau 1 Les différents alignements

Alignement	Le segment commence
BYTE	à l'emplacement mémoire suivant. Le segment est donc collé au segment précédent sans espace vide.
WORD	à l'emplacement mémoire divisible par 2 suivant. Le segment commence donc à une adresse paire; c'est à dire que le dernier quartet de l'adresse est de la forme xxx0 où x≡bit.
DWORD	à l'emplacement mémoire divisible par 4 suivant; c'est à dire que le dernier quartet de l'adresse est de la forme xx00 où x ≡ bit
PARAGRAPH 'PARA'	à l'emplacement mémoire divisible par 16 suivant c'est à dire que l'adresse est de la forme XXXX0 où X≡Quartet. En l'absence d'alignement, PARA est pris par défaut.
PAGE	à l'emplacement mémoire divisible par 256 suivant c'est à dire que l'adresse s'écrit sous la forme XXX00 où X≡Quartet.
INPAGE	Tout le segment est contenu dans une page (256 Octets).

[<TYPE>]: Optionnel, il décrit la manière dont l'éditeur de liens traite les segments de même nom. Les types de segments dont nous disposons sont: AT, PRIVATE, PUBLIC, MEMORY, STACK et COMMON. Sans précision, les segments seront de type PRIVATE (LOCAL) déposés en mémoire dans l'ordre de leur définition. Dans ce cas, si un segment de même nom est écrit dans un autre module, il sera considéré comme différent.

AT: Suivi d'une valeur ≤ 64 Ko (valeur sur 16 bits). La valeur segment est alors imposée à l'assembleur; c'est celle indiquée après AT (non symbolique)

PRIVATE: Le segment ne peut être rouvert ou prolongé dans d'autres modules. Il considéré comme local et inexistant dans les autres modules.
PUBLIC: Le segment peut être rouvert et prolongé dans ou plusieurs autres modules. C'est l'utilitaire LINK qui s'en charge.

MEMORY: Le segment est localisé au-dessus de tous les autres.

STACK: Le segment pile peut être rouvert par plusieurs modules.

COMMON: Les segments de même nom sont regroupés. Ils ne s'ajoutent pas les uns aux autres; ils se superposent.

[<CLASSE>]: Elle est optionnelle. Tous les segments d'une même classe sont regroupés en blocs même s'ils diffèrent par leurs noms avant tous les autres segments. Le nom doit figurer entre guillemets. C'est l'utilitaire LINK qui rassemble ces segments. L'absence de cette option implique une classe "nulle".

Exemples:

```
PILE1   SEGMENT  WORD     PUBLIC    'STACK'
        DB    100 DUP(?)
PILE1   ENDS
PILE2   SEGMENT  WORD     PUBLIC    'STACK'
        DB    200 DUP (?)
PILE2   ENDS
CODE    SEGMENT  PARA     PUBLIC    'CODE'
DATA    SEGMENT  BYTE     PUBLIC    'DATA'
MEMO    DW    100
DATA    ENDS
CODE    ENDS
TRICK   SEGMENT  AT    0000H
        ORG 21H*4
VECTOR  EQU $
TRICK   ENDS
```

Remarque: Un segment de type AT peut contenir ni codes ni données. Il s'agit plutôt d'un moyen pour affecter une adresse précise à des étiquettes, par exemple dans le cas de vecteurs d'interruption ou d'adresses de mémoire vidéo.

VI.4.6 Directive 'PROC/ENDP'

Définit le début et la fin d'une procédure.

Syntaxe:

```
<NOM>     PROC [<DISTANCE>][USES<REGISTERS>...,]
           [<ARGUMENTS>][RETURNS <ARGUMENTS>]

[<NOM>]   ENDP
```

<NOM>: Il est utilisé pour appeler une procédure au moyen d'un CALL.

[<DISTANCE>]: Procédure de type NEAR ou FAR (Absence: NEAR)

[USES<REGISTERS>]: Registres utilisés par la procédure et sauvegardés automatiquement au moment de l'exécution de la dite procédure.

[<ARGUMENTS>]: Transmission par pile d'arguments à une procédure.
[RETURNS <arguments>]: Réception par la pile d'arguments du programme appelant.

Exemples

```
DEMO    PROC  NEAR  USES  SI  DI
        XOR  SI,SI
        XOR  DI,DI      .          .
        RET
        DEMO     ENDP

TEST    PROC     FAR I:BYTE, J:BYTE  RETURNS POINTER:DWORD
        :
        RETF
TEST    ENDP
```

VI.4.7 Directive 'GROUP'

Définit un groupe de plusieurs segments.

Syntaxe: <NOM> GROUP <NOMS DES SEGMENTS>

Exemples:

```
ALL_GROUP   GROUP    SEG_1, SEG_2, SEG_3
SEG_1       SEGMENT  PARA      PUBLIC    'DATA'
            ASSUME   DS:ALL_GROUP
SEG_1       ENDS
CODE        SEGMENT  WORD      PUBLIC    'CODE'
            ASSUME   CS:CODE
SEG_2       SEGMENT  PARA
            ASSUME   DS:ALL_GROUP
SEG_2       ENDS
SEG_3       SEGMENT  PARA      PUBLIC    'DATA'
            ASSUME   DS:ALL_GROUP
            DEMO          DW  ?
            SEG_3         ENDS
            CODE          ENDS
```

VI.4.8 Directive 'OFFSET'

Permet d'acquérir l'offset d'une étiquette.

Syntaxe: OFFSET <EXPRESSION>

Exemple:

```
DEMO    DB   'MESSAGE'
        MOV  AX, OFFSET DEMO
        MOV  SI, OFFSET ALL_GROUP:DEMO
```

VI.4.9 Directive 'EQU' (Equate)

Définit un symbole mais ne réserve pas de case mémoire.

Syntaxe: <NOM> EQU <EXPRESSION>

Exemples

```
ENTIER      EQU  1234        ; Valeur entière
EXPRESSION  EQU  12*ENTIER   ; Expression
TEXTE       EQU  "BONJOUR"   ; Chaîne
GETSP       EQU  MOV BP,SP   ; Instruction
LOIN        EQU  FAR PTR     ; Déclaration
PRES        EQU  NEAR PTR    ; Déclaration
```

VI.4.10 Directive 'DB' (Define Byte)

Définit un tableau d'octets.

Syntaxe: <NOM> DB <TABLEAU>

Exemples

```
DEMO1    DB   1                  ; Valeur
DEMO2    DB   1,2,3              ; Plusieurs valeurs
CHAINE   DB   'VOICI UNE CHAINE' ; Une chaîne
MIXTE    DB   1,2,'CHAINE',3,4   ; Mélange de genres
FLAG     DB   3                  ; Valeur
CALCUL   DB   49*3-FLAG          ; Une expression
VIDE     DB   ?                  ; Réservation d'un octet
```

```
ENCORE   DB   100 DUP (?)        ; Réservation de 100 octets
DATA     DB   30 DUP (1,2,3)     ; Remplissage de 30 octets
```

VI.4.11 Directive 'DW' (Define Word)

Définit une valeur de type mot.

Syntaxe: <NOM> DW [<TYPE POINTEUR>]<TABLEAU>

Exemples:

```
INT          DW   1234             ; Entier sur 16 bits
ADR          DW   ARRAY            ; Pointeur sur ARRAY
ARRAY        DW   20 DUP (?)       ; 20 valeurs sur 16 bits
ADR_ARRAY    DW   BYTE PTR ARRAY   ; Un  BYTE PTR sur ARRAY
```

VI.4.12 Directive 'EVEN'

Initialise CE à une adresse paire

Syntaxe: EVEN

Exemple:

```
             EVEN
BOUCLE :     CALL QUELQUE_CHOSE
             :
             LOOP BOUCLE
             :
             EVEN
MOT          DW  FADC
```

VII Canevas d'écriture d'un programme

VII.1 Première manière d'écrire un programme

Dans un premier temps, nous pouvons supposer que l'ensemble des segments ne dépassent pas 64 Ko. Dans ce cas, nous définissons un seul segment appelé CODE qui répond au canevas suivant:

```
;NOM DU PROGRAMME

CODE    SEGMENT
        ORG     0100H
        ASSUME  CS:CODE, DS:CODE, ES:CODE, SS:CODE
                "PLACER TOUS LES EQUATES ICI"
START:  JMP     BEGIN
                "PLACER TOUS LES DEFINE ICI"
                "PLACER TOUTES LES PROCEDURES ICI"
BEGIN:          "PLACER VOTRE PROGRAMME ICI"
CODE            ENDS
                END  START
```

VII.1.1 Code Gray

```
;GRAY_CODE

CODE    SEGMENT
        ORG  0100H
        ASSUME  CS:CODE, DS:CODE, ES:CODE, SS:CODE
START:  JMP  BEGIN
GRAY    DB    00H  01H 03H ...
BEGIN:  MOV  AX, CODE
        MOV  DS, AX
        MOV   BX,OFFSET GRAY
NEXT:   IN     AL, 01H
        AND   AL, 0FH
        XLAT  GRAY
        OUT   02H, AL
        JMP   NEXT
        HLT
CODE    ENDS
        END  START
```

VII.2 Deuxième manière d'écrire un programme

La deuxième manière d'écrire un programme suppose que la somme des segments dépasse 64 Ko. Dans cet exemple, nous définissons deux segments; un segment DATA et un segment CODE comme suit:

```
;NOM DU PROGRAMME

DATA        SEGMENT
            ORG  F100H
DATA_1      DB     20 DUP(?)
DATA_2      DB     80 DUP(0,1,2)
            :
DATA        ENDS
CODE        SEGMENT
            ORG  0100H
ASSUME      CS:CODE, DS:DATA, ES:DATA, SS:CODE
            "PLACER TOUS LES EQUATES ICI"
            "PLACER TOUTES LES PROCEDURES ICI"
START:      "PLACER VOTRE PROGRAMME ICI"
CODE        ENDS
            END   START
```

VII.2.1 Addition des nombres ASCII en BCD

```
;ADDITION ASCII
ALL_GROUP   GROUP      CODE, DATA
DATA        SEGMENT
            ORG  0100H
STRING_1    DB     '1752'
STRING-2    DB     '3814'
DATA        ENDS
CODE        SEGMENT
            ORG  F100H
ASSUME      CS:ALL_GROUP, DS:ALL_GROUP, ES: ALL_GROUP,
            SS: ALL_GROUP
COMPT       EQU  04H
START:      MOV  AX,DATA
            MOV  DS,AX
            MOV  ES,AX
            MOV  CX,COMPT
            CLC
            CLD
            MOV  SI,OFFSET STRNG_1
            MOV  DI,OFFSET STRING_2
SUIVANT:    LODS STRING_1
            ADC  AL,ES:[DI]
            AAA
            STOS STRING_2
```

```
                LOOP SUIVANT
                HLT
CODE            ENDS
                END START
```

CHAPITRE 7

MODES DE TRANSFERTS DES ENTREES ET SORTIES

I Introduction

Un microprocesseur n'est pas fait pour dialoguer avec sa mémoire de données et de programmes seulement, il doit pouvoir communiquer avec la périphérie et recevoir ou transmettre les informations de ou vers l'environnement extérieur.

Cet environnement dépend évidemment de l'application. Cela peut être en entrée un clavier ou une unité de disque souple et en sortie une imprimante ou un écran ou cela peut être tout simplement un ensemble de signaux électriques provenant de capteurs.

Pour réaliser un tel transfert, il faut que le microprocesseur travaille efficacement avec sa périphérie de manière à ne pas dégrader ses performances de traitement. Il existe alors trois modes de dialogue entre le microprocesseur et sa périphérie.

II Mode programmé avec ou sans test du mot d'état

Dans ce cas, le microprocesseur prend entièrement en charge le dialogue avec la périphérie.

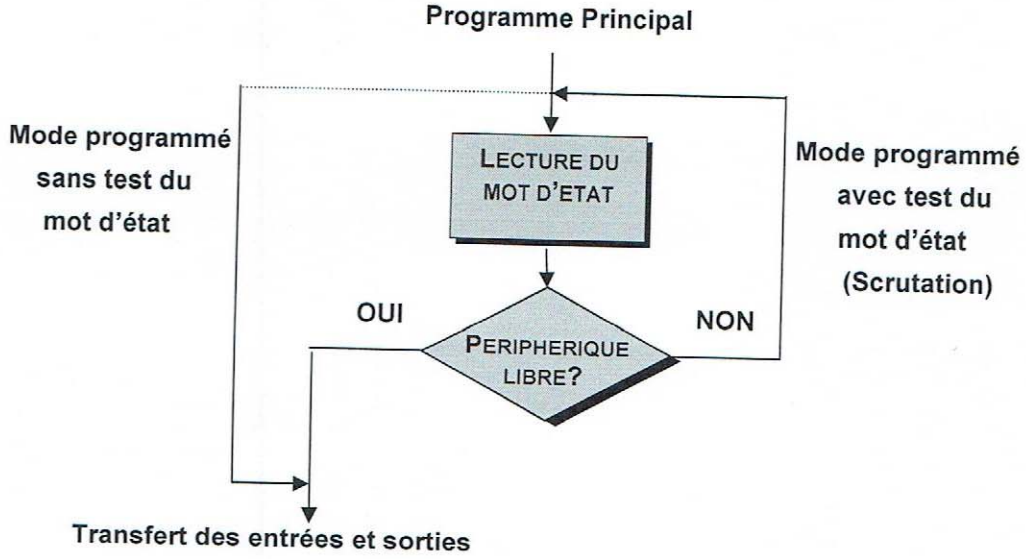


Fig. 1 Mode programmé avec ou sans test du mot d'état

Cette méthode est coûteuse en temps et réduit les performances globales du système. Nous préférons effectuer les transferts en interrompant le microprocesseur chaque fois qu'un périphérique souhaite échanger des informations. Le microprocesseur exécute alors normalement son programme et ne s'arrête que si on le lui demande.

III Mode interruptible

III.1 Définition

Une interruption est une commutation du contexte d'un processeur déclenchée par une cause extérieure au déroulement de l'instruction en cours. Physiquement, l'interruption se traduit par un signal envoyé au microprocesseur. Ce signal provoque le changement d'état d'un indicateur consulté au cours de chaque instruction.

Remarque: Contrairement à une interruption, un déroutement ou un appel à un superviseur est provoqué par une cause directement liée au déroulement de l'instruction en cours.

III.2 Concept des interruptions

Dans ce cas l'initiative de transfert est laissée au périphérique et non plus au microprocesseur. Ainsi, par exemple, pour le clavier le programmeur ne peut savoir à quel moment l'utilisateur viendra appuyer sur une touche du clavier pour demander ou donner une information. Par conséquent, l'appui sur une touche fait une demande d'interruption au microprocesseur.

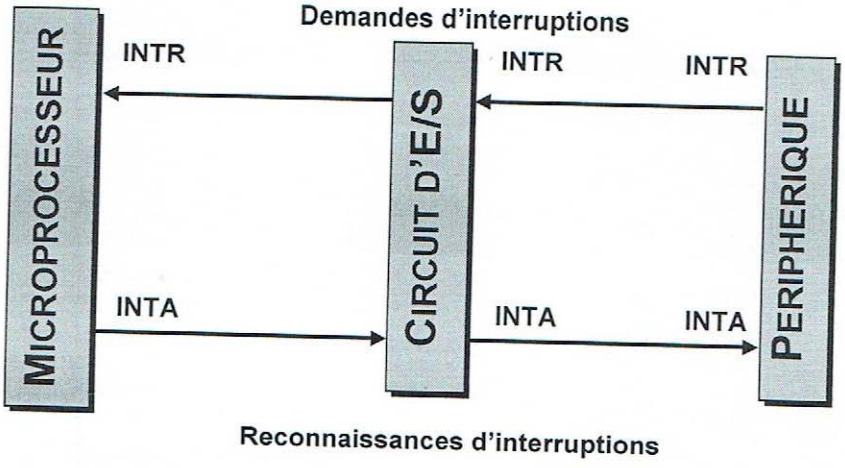


Fig. 2 Schématisation du concept des interruptions

Dans le cas où plusieurs périphériques demandent l'attention du microprocesseur, deux problèmes apparaissent.

1- La hiérarchisation des priorités en cas de sollicitations simultanées.

2- La recherche du ou des périphériques ayant sollicité le transfert.

III.3 Gestion des priorités

En général, une seule entrée d'interruption doit supporter plusieurs circuits d'entrée et de sorties. Tous ces circuits possèdent des niveaux de priorités égaux. Pour définir la priorité d'un circuit d'Entrées/Sorties par rapport à un autre, nous disposons de deux méthodes.

III.3.1 Méthode logicielle

Toutes les lignes de demandes d'interruptions sont câblées en 'OU' logique. Par conséquent, Tous les circuits d'entrée et sortie ont un vecteur d'interruption commun.

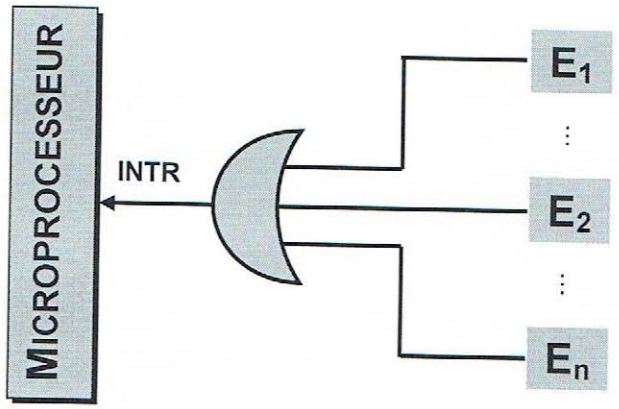


Fig. 3 Câblage des lignes d'interruptions à travers un 'OU' logique

Par ailleurs, comme il s'agit d'une méthode logicielle, le programme associé à toute demande d'interruption doit répondre aux tests de l'organigramme de la Figure 4.

III.3.2 Méthode matérielle

On peut établir la hiérarchisation entre les différents circuits d'entrées et de sorties en utilisant une logique externe permettant d'identifier directement l'origine de la demande. Cette logique existe sous forme de circuit intégré (contrôleur de

priorité). Il permet de gérer jusqu'à huit niveaux d'interruption. Le numéro de la ligne d'interruption active est codé sur trois bits et la demande d'interruption est transmise au microprocesseur à travers la ligne INTR.

Dans le cas d'interruptions simultanées, c'est celle qui a le plus petit numéro en décimal qui est sélectionnée.

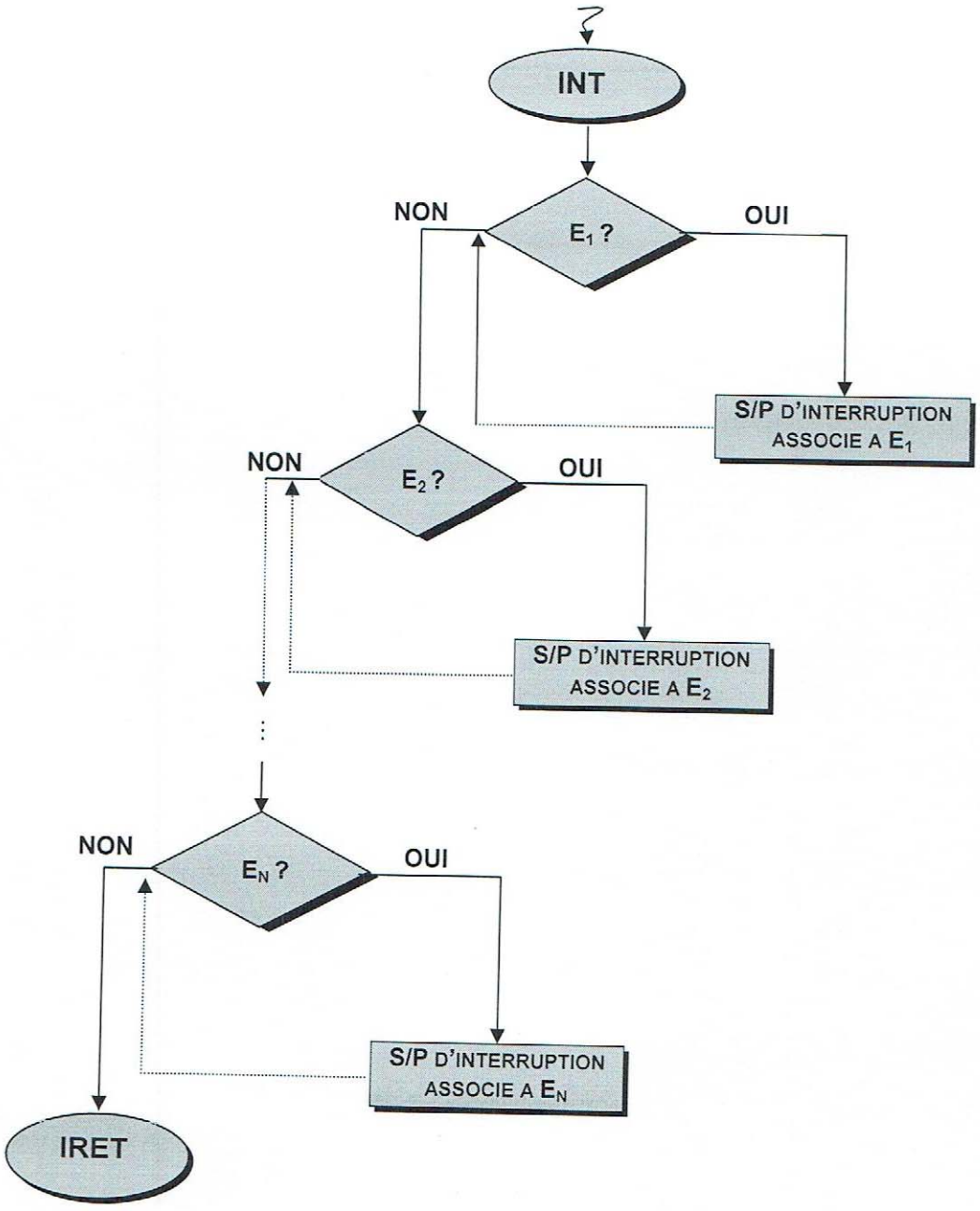


Fig. 4 Organigramme de hiérarchisation et de scrutation des interruptions

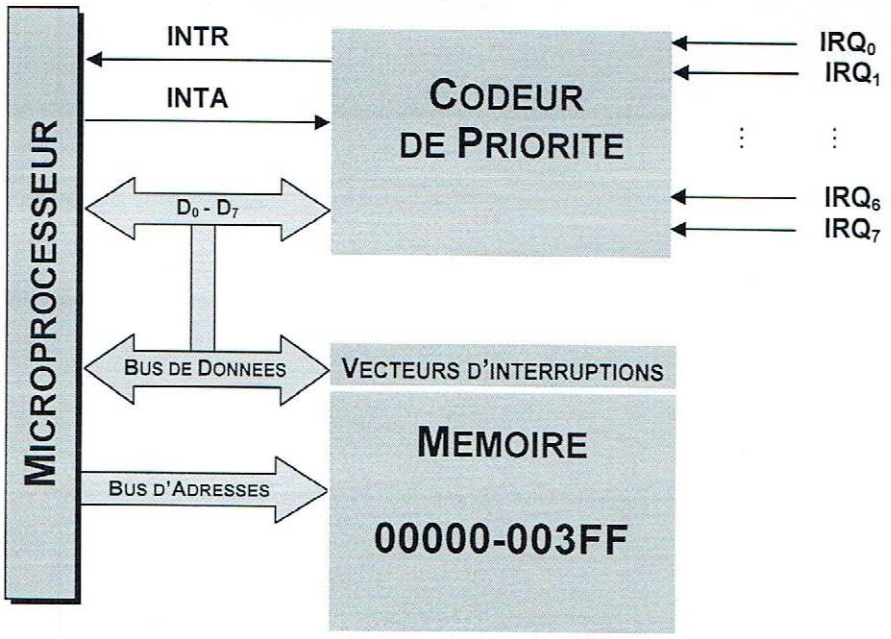


Fig. 5 Le codeur de priorité dans une structure à base de microprocesseur

IV Mode DMA (Direct Memory Access)

Lorsqu'un périphérique a un assez grand nombre d'informations à faire rentrer en mémoire ou lorsqu'il doit en recevoir une quantité importante de la mémoire, le transfert par interruption est lent, un autre mode de transfert des entrées et sorties est alors possible, c'est le mode DMA. Dans ce mode, les échanges périphérique-mémoire ou mémoire-périphérique sont effectués directement tout en maintenant les bus du microprocesseur à l'état de haute impédance. Cette nouvelle possibilité entraîne l'utilisation de deux nouveaux signaux DMARQ (DMA Request) et DMAACK (DMA Acknowledge).

Il existe des DMA en circuit intégré qui font fonction d'un processeur spécialisé dans la gestion de la mémoire en mode DMA. Dans ce cas, l'initiative de transfert est laissée au périphérique.

Dans les PC le DMA à deux fonctions principales d'échange de données, d'une part, entre disque dur & Mémoire et disque souple & Mémoire. D'autre part, le rafraîchissement des mémoires dynamiques.

Remarques:

1- Dans le cas d'un transfert de la périphérie vers la mémoire, il y a activation simultanée de MEMW, IOR et DMAACK.
2- Dans le cas d'un transfert de la mémoire vers la périphérie, il y a activation simultanée de MEMR, IOW et DMAACK

3- Pour le MOVS (Move String), par exemple, les transferts mémoire-mémoire nécessitent 8 cycles avec le DMA contre 17 cycles sans le DMA.
4- Dans les systèmes ISA-DMA, il existe 16 canaux dont 7 sont utilisés par les CPU des PC.
5- Dans les nouveaux systèmes PCI-BUS MASTERING DMA, ce sont les périphériques eux-mêmes qui contrôlent les bus et effectuent les transferts.
6- Les systèmes 'on-chip' ou 'embedded systems' utilisent, quant à eux, les protocoles de bus AHB (Advanced High Performance Bus) introduit par AMBA (Advanced Microcontroller Bus Architecture). Ce type de protocole est utilisé dans les architectures ARM (Advanced RISC Machines).

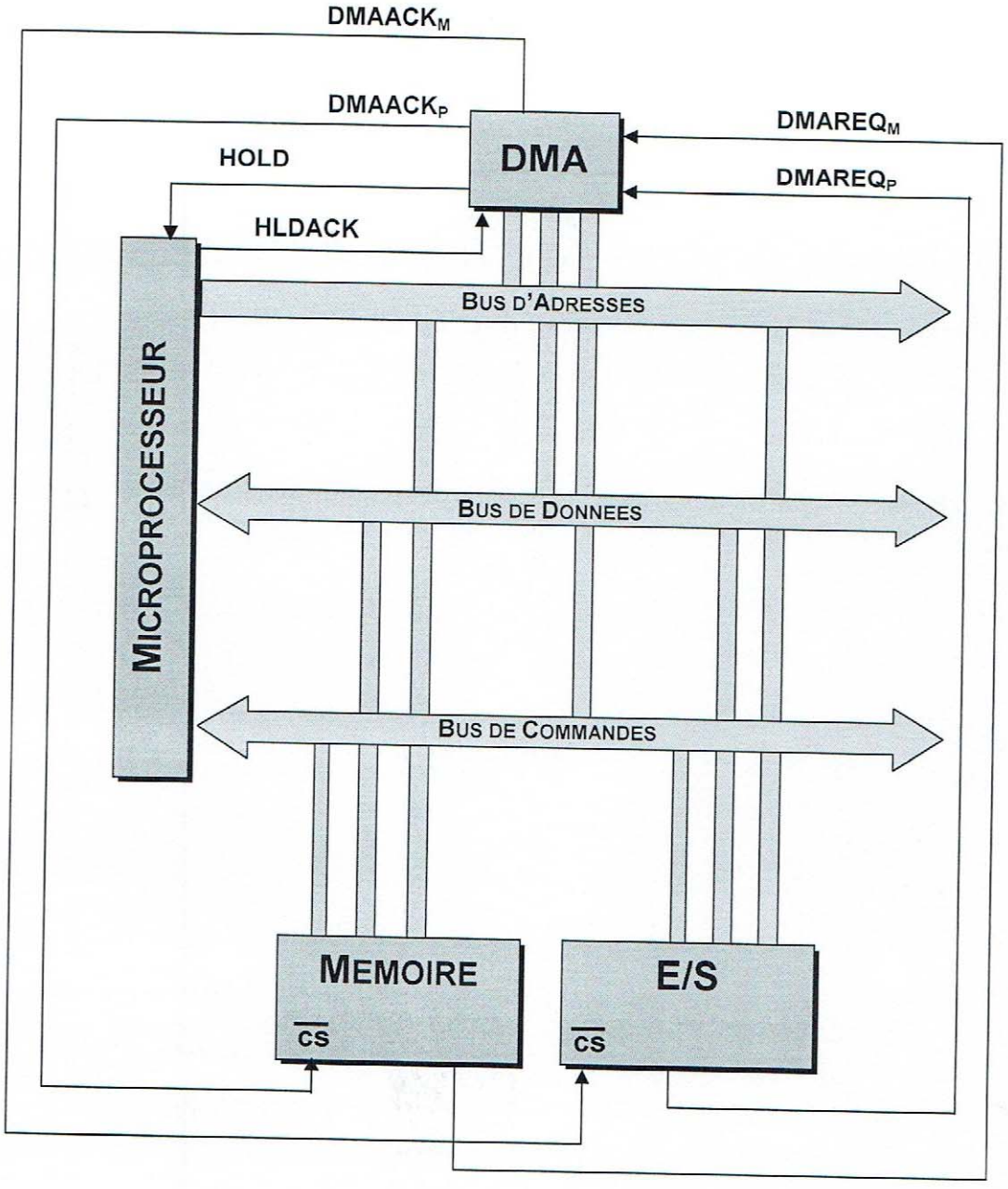


Fig. 6 Le circuit DMA dans une structure à base de microprocesseur

V Tableau récapitulatif

Enfin, nous pouvons conclure par une comparaison des trois modes de transferts sous forme de tableau récapitulatif.

Tableau 1 Caractéristiques des trois modes de transferts

Mode	Initiative	Rapidité
Programmé	Microprocesseur	Lent
Interruptible	Périphérique	Rapide
DMA	Périphérique	Très rapide

Ainsi, nous remarquons à travers le Tableau 1, la rapidité du mode DMA par rapport aux deux autres modes. Cependant, le mode interruptible constitue un moyen puissant de transfert de données car les microprocesseurs travaillent, de nos jours, à des horloges excessivement élevées.

CHAPITRE 8

INTERRUPTIONS DANS LE 8086

I Introduction

Le mécanisme des interruptions est une technique très répandue dans les systèmes à microprocesseurs. De par sa souplesse et son efficacité, elle offre de larges possibilités d'exploitation, notamment pour ce qui est de l'accès aux organes périphériques.

La scrutation périodique représente le moyen commun et évident pour l'accès aux entrées et sorties. En contrepartie, elle nécessite un temps de réponse substantiel et devient carrément inopérante quand il s'agit de contrôler un grand nombre de dispositifs d'entrées et sorties.

C'est précisément, en présence d'un nombre important de périphériques que les interruptions étalent toute leur efficacité en libérant le microprocesseur de la fastidieuse tâche de scrutation, améliorant du coup, d'une façon significative, les performances du système. Le PC est un exemple typique dans lequel les périphériques de base sont exclusivement pris en charge par les interruptions : Clavier, horloge cyclique, interface parallèle, interface sérielle, contrôleur disque dur, contrôleur de lecteur disquettes et RTC/ RAM CMOS dans les PC-AT.

II Principe général des interruptions

D'une manière simplifiée, le mécanisme des interruptions matérielles repose sur le principe suivant : le microprocesseur est interrompu par un périphérique lorsque celui-ci sollicite un service particulier. Ce n'est qu'à partir de cet instant, sous certaines conditions liées à la priorité et au masquage des interruptions, que le microprocesseur quitte le programme en cours pour exécuter la routine spécifique au service demandé. Un circuit spécialisé appelé communément contrôleur des interruptions joue un rôle important en assurant certaines tâches essentielles dans ce mécanisme.

III Description sommaire du contrôleur des interruptions

Dans le cas des micro-ordinateurs PC-XT et PC-AT, le contrôle des interruptions est assuré par un circuit programmable de la famille Intel appelé PIC (Priority Interrupt Controller): le 8259 pour le PC-XT et le 8259 A pour le PC-AT.

La Figure 1, montre le PIC dans une structure à base de 8086. Nous remarquons que lorsque le 8086 est configuré en mode maximum, les lignes INTR et $\overline{INTA}$ sont connectés au 8288 ou contrôleur de bus. Les principales fonctions du PIC peuvent être résumées comme suit:

1- La mémorisation des demandes d'interruptions émanants des différents périphériques.
2- La hiérarchisation des priorités. Autrement dit, le PIC permet de classer les lignes d'interruption IRQ_i i=0,1,..., 7 par ordre de priorité. Ce qui donne la possibilité d'examiner la priorité d'une requête relativement à l'interruption en service.
3- Enfin, l'identification du périphérique demandeur. En effet, le PIC envoie un numéro spécifique à chaque périphérique au microprocesseur afin de lui permettre de déterminer le type de l'interruption demandée.

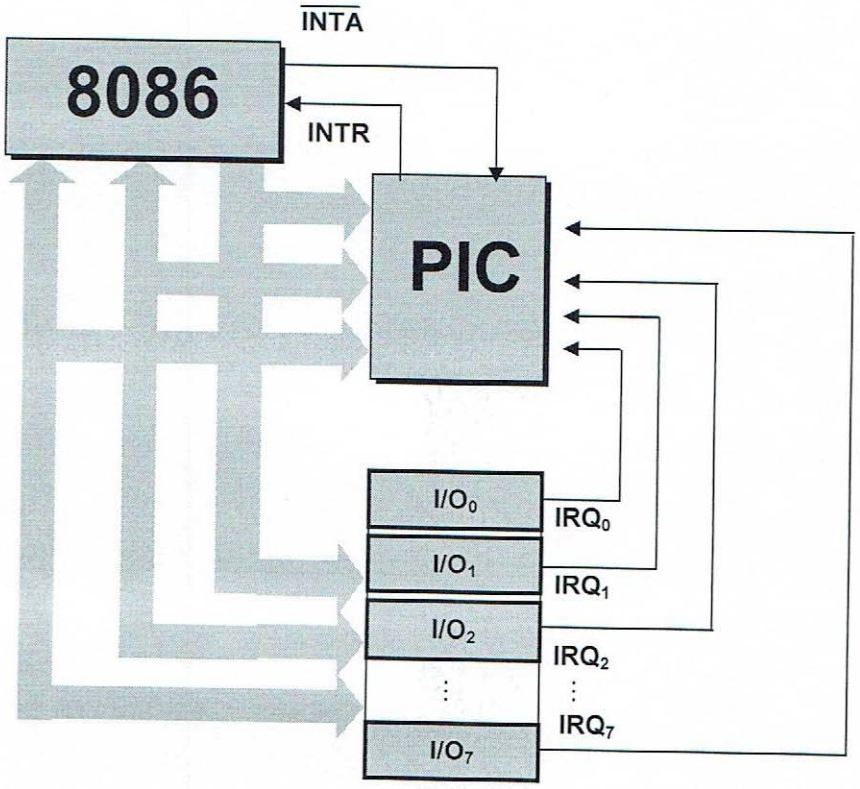


Fig. 1 Le PIC dans une structure à base de 8086

III.1 Architecture Interne du PIC

Vue à partir du 8086, l'architecture du PIC est essentiellement basée sur les registres de la Figure 2.

III.1.1 Registres IRR et ISR (Interrupt Request Register et In Service Register)

Les entrées IRQ_i sont prises en charge par deux registres en cascade IRR et ISR respectivement registre de demandes des interruptions pour stocker les requêtes des interruptions et registre des interruptions en service pour stocker le rang des interruptions en cours de traitement.

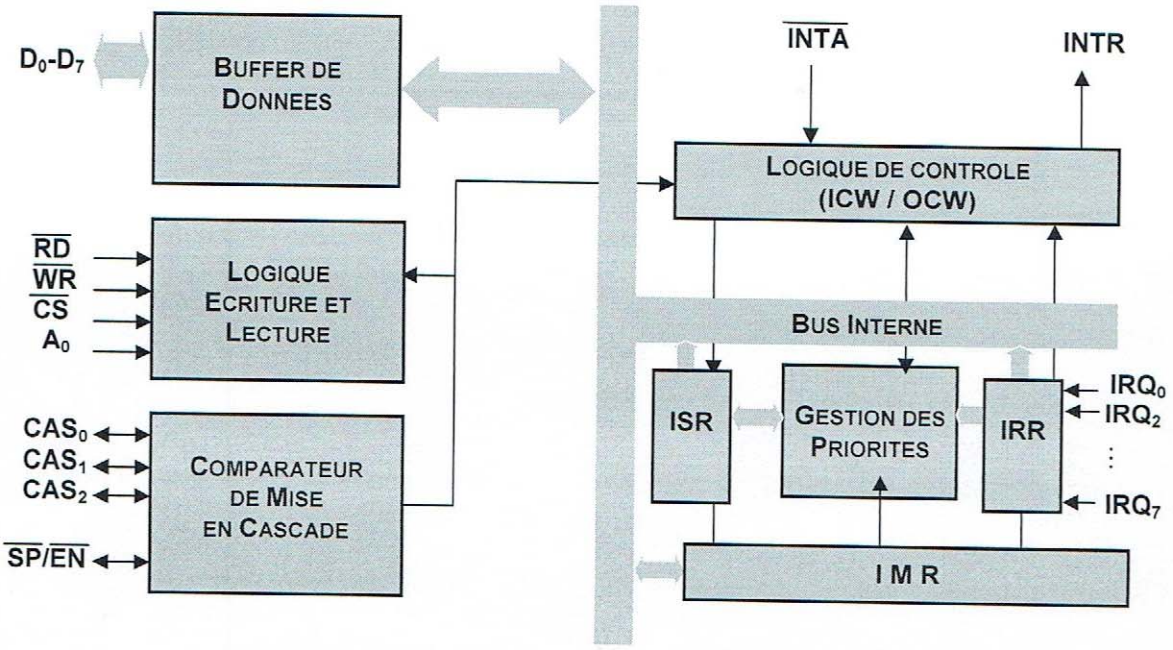


Fig. 2. Architecture interne du PIC

III.1.2 Registre gestionnaire des priorités

Ce block détermine les priorités des bits mis à '1' dans l'IRR (requêtes mémorisées). La plus grande priorité est sélectionnée et le bit correspondant est mis à '1' dans le registre ISR (Interruption en service) durant le cycle de reconnaissance INTA ce qui permet au PIC de fournir le type de l'interruption correspondante en déposant son numéro sur le bus de données.

III.1.3 Registre IMR ou OCW_1 (Interrupt Mask Register)

L'interruption est validée si le bit correspondent dans le registre IMR est mis à '0'. L'interruption est inhibée si le bit correspondant dans le registre IMR est à '1'.

III.1.4 Logique de lecture et d'écriture

En fonction des signaux $\overline{CS}$, $\overline{WR}$, $\overline{RD}$ et A_0, comme le montre le Tableau 1, ce bloc permet au 8086 d'écrire des mots de commande d'initialisation et des mots de commande d'opération dans les registres ICW (ICW_1-ICW_4): Initialization Command Register et OCW (OCW_1-OCW_3): Operation Command Register. Il permet également la lecture des registres IRR, ISR, le masque IMR ou le niveau des interruptions.

Tableau 1 Adressage des registres du PIC

$\overline{RD}$	$\overline{WR}$	A_0	$\overline{CS}$	Registre sélectionné	Fonction
0	1	1	0	OCW_1	Lecture des masques
1	0	0	0	ICW_1, OCW_2 ou OCW_3	Ecriture initialisation / opération
0	1	0	0	IRR ou ISR	Lecture des requêtes / services
1	0	1	0	ICW_2, ICW_3, ICW_4 ou OCW_1	Ecriture initialisation / opération
X	X	X	1	Aucun	Haute impédance

III.1.5 Logique de contrôle

Dans le cas où le PIC 8259 est seul, le 8086 a besoin de 3 écritures pour l'initialiser à travers les registres ICW_1, ICW_2 et ICW_4. Dans le cas où il y a plus d'un PIC 8259, le 8086 a besoin de 4 écritures pour initialiser chacun des PIC à travers les registres ICW_1, ICW_2, ICW_3 et ICW_4. Les registres OCW_1, OCW_2 et OCW_3 peuvent être générés à tout moment au cours du programme.

Description des registres ICW_1-ICW_4 : Initialization Command Write

Registre ICW_1: Ce registre décrit si le PIC est seul ou non.

X=Do not care LTIM SNGL

X	X	X	X	1/0	X	1/0	X

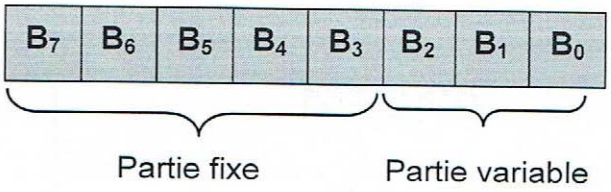
SNGL:Single= { 1 Un seul PIC ; 0 Plusieurs PIC }

LTIM= Level Triggered Input Mode

LTIM = { 1 Détection d'un niveau haut stable sur IRQ_i ; 0 Détection d'un changement d'état sur IRQ_i

Remarque: Le bus ISA ne supporte pas le cas LTIM=1. Par conséquent, les PC/XT et PC/AT et systèmes compatibles doivent être programmés en LTIM=0 (Edge Triggered Input Mode). Le bus MCA, quant à lui, supporte le cas LTIM=1. Sur les nouveaux bus tels que EISA, PCI, etc...., un registre appelé Edge/Level Control Register (ELCR) contrôle le mode par rapport à l'entrée IRQi. Ce qui rend le mode du 8259 inadéquat pour de tels systèmes ayant le bus ISA. Le registre ELCR est programmé par le BIOS au démarrage du système, i.e., $ELCR_i$= 1 pour LTIM et 0 pour ETIM.

Registre ICW_2: Ce registre décrit le vecteur d'interruption.

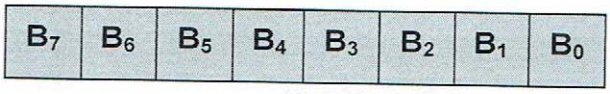


B_3 - B_7 : Concernent la partie haute (fixe) du vecteur d'interruption.

B_0 - B_2 : Représentent le code de IRQ_i i=0, 1, 2, ..., 7 (ajouté automatiquement).

Registre ICW_3: Ce registre n'est utilisé que dans le cas d'une configuration à deux ou plusieurs PIC i.e., SNGL=0. Par conséquent, il existe deux configurations pour ICW_3.

ICW_3 pour le PIC maître :



B_i= { 1 Un PIC esclave est branché sur IRQ_i i=0, 1, 2, ..., 7 ; 0 Demande ordinaire sur IRQ_i

ICW_3 pour le ou les PIC esclaves:

0	0	0	0	0	B_2	B_1	B_0

B_0-B_2 : Permettent l'identification de l'esclave branché sur IRQ_i i=0, 1, 2, ..., 7.

Registre ICW_4: Ce registre décrit le mode spécial d'imbrication de plusieurs PIC.

			SFNM	BUF	M/S	EOI	
0	0	0	1/0	1/0	1/0	1/0	1

EOI: End Of Interrupt $= \begin{cases} 1 & \text{AEIO} \\ 0 & \text{SEOI.} \end{cases}$

AEOI: Le bit correspondant dans ISR est remis à 0 à la fin du 2ème cycle $\overline{INTA}$.
EOI: Un mot de commande OCW_2 doit être envoyé avant le retour de la procédure interruption.

Les différents modes Master/Slave avec ou sans buffer, associés au positionnement des bits BUF et M/S, sont résumés dans le Tableau 2.

Tableau 2 Modes Master/Slave avec ou sans buffer

BUF=Buffer	M/S=Master/Slave	Mode M/S
0	0	Sans buffer esclave
0	1	Sans buffer maître
1	0	Buffer esclave
1	1	Buffer maître

SFNM: Special Fully Nested Mode $= \begin{cases} 1 & \text{Mode SFNM} \\ 0 & \text{Mode FNM} \end{cases}$

SFNM=1: Lorsque plusieurs PIC 8259 sont cascadés, le maître peur être programmé dans ce mode à travers ICW_4. Dans ce mode, un esclave qui a déjà une interruption en service pourra en faire prendre en compte d'autres par le maître, si elles sont de plus forte priorité.

SFNM=0: Dans ce mode, le maître ignore les interruptions suivantes du même esclave, jusqu'à la commande EOI correspondant à l'interruption en service. Conséquemment, le 8086 doit vérifier avant de sortir du programme, que l'esclave qui l'a interrompu n'a pas une seconde interruption en service. Pour cela, il envoie un EOI non spécifique à l'esclave et lit son ISR (les bits R S L = 001 de OCW_2). Si celui-ci est vide, il envoie aussi un EOI au maître. Sinon, i.e., s'il y a encore une interruption en service, il ne doit pas envoyer de commande EOI au maître. L'automatique EOI (AEOI) ne doit pas être utilisé dans ce cas.

Description des registres $0CW_1$-OCW_3: Operation Command Write

Registre OCW_1=IMR: Ce registre permet se valider les interruptions associées aux lignes IRQ_0, IRQ_1, ... et IRQ_7.

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

B_i (i=0, 1, 2, ..., 7)= { 1 Inhibition de l'interruption correspondante à IRQ_i ; 0 Validation de l'interruption correspondante à IRQ_i

Registre OCW_2: Ce registre permet la rotation des priorités afin d'éviter qu'une interruption n'accapare l'attention du 8086. Cette rotation peut s'effectuer au moment de l'envoi d'un AEOI ou EOI spécifique ou non.

R, SL et EOI indiquent les choix possibles de modification des priorités par rotation.

R	SL	EOI		L_2	L_1	L_0	
1/0	1/0	1/0	0	1/0	1/0	1/0	1

Tableau 3 Rotation des priorités au moment d'un AEOI ou EOI spécifique ou non

R	SL	EOI	Fonction
0	0	1	Fin d'interruption non spécifique
0	1	1	*Fin d'interruption spécifique
1	0	1	Rotation des priorités au moment d'un EOI non spécifique
1	1	1	*Rotation sur chaque EOI spécifique
0	0	0	Arrêt de la rotation sur un EOI automatique
0	1	0	Pas d'opération
1	0	0	Rotation autorisée à chaque EOI automatique
1	1	0	*Fixe le bit de priorité la plus élevée

Remarque: Les bits L_2, L_1 et L_0 fixent éventuellement le numéro de l'interruption concernée par l'action de OCW_2. L'indication (*) tient compte de ces bits. Les autres les ignorent.

R: Rotation = { 1 Rotation autorisée ; 0 Sinon }

SL: Specific Leve l = { 1 EOI spécifique ; 0 EOI non spécifique }

EOI = { 1 EOI spécifique ou non spécifique ; 0 EOI automatique }

Registre OCW_3: Ce registre décrit, d'une part, les modes de masquage spécial et celui de la scrutation et permet, d'autre part, de lire les registres ISR et IRR.

		ESMM	SMM			P	RR	RIS
0		1/0	0/1	0	1	1/0	1/0	1/0

Le mode spécial de masquage résout les problèmes d'occupation abusive du 8086 par une interruption.

ESMM : Enable Special Masked Mode = { 1 Autorise l'effet de SMM ; 0 Inhibe l'effet de SMM }

SMM : Special Masked Mode = { 1 Mode SMM sélectionné ; 0 Mode SMM non sélectionné }

Dans le mode SMM, il faut exécuter la séquence d'opérations suivantes:

1- Masquer, à travers OCW_1=IMR, uniquement le bit correspondant à la routine en cours d'exécution.
2- Mettre le PIC en mode spécial de masquage à l'aide de OCW_3, i.e., ESMM=SMM=1.
3- Toujours à travers OCW_1=IMR, autoriser les interruptions sans tenir compte de

leur priorité.

Enfin, avant de sortir de la routine, il faut exécuter la séquence inverse. Autrement dit:

1- Bloquer les interruptions, validées à l'étape 3 précédente.
2- Rétablir un masquage normal, validé à l'étape 2 précédente.
3- Valider le bit de l'interruption, masquée à l'étape 1 précédente.

Le mode de scrutation est prévu dans le cas où l'application requiert l'utilisation de plus de 64 PIC.

P: Poll Command = { 1 Commande de scrutation ; 0 Commande par interruption }

Dans ce mode, c'est le 8086 qui a l'initiative pour déclencher les interruptions. En effet, celles-ci sont traitées normalement par le 8259. Cependant, le 8259 ne peut interrompre le 8086 (IFF=0). A l'initiative du 8086, une commande OCW_3 contenant P=1 est émise. Le 8259 traite alors la prochaine lecture comme une reconnaissance d'interruption. Durant cette lecture, il émet sur le bus de données le mot (octet) suivant:

	X=Do not care				W_2	W_1	W_0
I	X	X	X	X	1/0	1/0	1/0

I: Interrupt = { 1 Interruption demandée ; 0 Sinon }

Les bits W_2, W_1 et W_0 désignent l'interruption IRQ_i de plus haute priorité.

Le mode de scrutation présente beaucoup d'avantages. Il permet notamment de:

1- Traiter les interruptions au moment convenant le mieux pour le logiciel. IFF=1 ne permet pas un tel contrôle.
2- N'écrire qu'une seul fois une procédure identique pour plusieurs niveaux d'interruptions.
3- D'étendre le nombre de PIC au-delà de 64. Le 8086 procédant par la commande Poll pour tous.

Lecture des registres ISR et RR.

$$\text{RR: Read Register} = \begin{cases} 1 & \text{Autorise la lecture de ISR et IRR} \\ 0 & \text{Sinon} \end{cases}$$

$$\text{RIS: Read In Service} = \begin{cases} 1 & \text{Lecture de ISR} \\ 0 & \text{Lecture de IRR} \end{cases}$$

La lecture de ISR ou de IRR doit s'effectuer en deux étapes. La première consiste à envoyer un mot de commande au PIC à travers OCW_3 dans laquelle RR=1 et RIS=1(pour une lecture de ISR) ou 0 (pour une lecture de IRR), suivie d'une lecture de OCW_3. Cette deuxième étape permettra de lire le contenu de ISR ou IRR.

III.1.6 Buffer de données: D_0-D_7

Ce bus bidirectionnel en logique 3 états permet le transfert en écriture et en lecture des mots de contrôle et d'état du PIC.

III.2 Architecture Externe du PIC

Le PIC se présente sous la forme d'un boîtier DIL 28 broches monotension (0, 5V). Nous distinguons les signaux d'interface avec le microprocesseur et les signaux d'interface avec la périphérie.

III.2.1 Liaisons avec le 8086

Bus de données (D_0-D_7, entrées et sorties): Ces 8 lignes bidirectionnelles en logique 3 états, sont directement connectées au bus de données du 8086 (AD_0-AD_7). Elles assurent l'échange des données entre le 8086 et le PIC.

Bus d'adresses ($\overline{CS}$, entrée et A_0, sortie): Un état bas sur la broche $\overline{CS}$, permet la sélection d'un boîtier PIC pour les échanges avec le 8086.

$$\overline{CS} = \begin{cases} 0 & \text{Sélection d'un boîtier PIC} \\ 1 & \text{Sinon} \end{cases}$$

Combinée à $\overline{CS}$, $\overline{WR}$ et $\overline{RD}$, la ligne d'adresse A_0 permet de sélectionner un des registres du PIC (Cf. Tableau 1).

Bus de contrôle (WR et RD): La lecture/écriture des registres du PIC s'effectue sous contrôle des signaux $\overline{WR}$ et $\overline{RD}$.

Demande d'interruption (INTR): Connecté à la broche INTR du 8086, ce signal est généré par le PIC lorsque celui-ci confirme une demande d'interruption. Le bit correspondant est mis à '1' dans ISR.

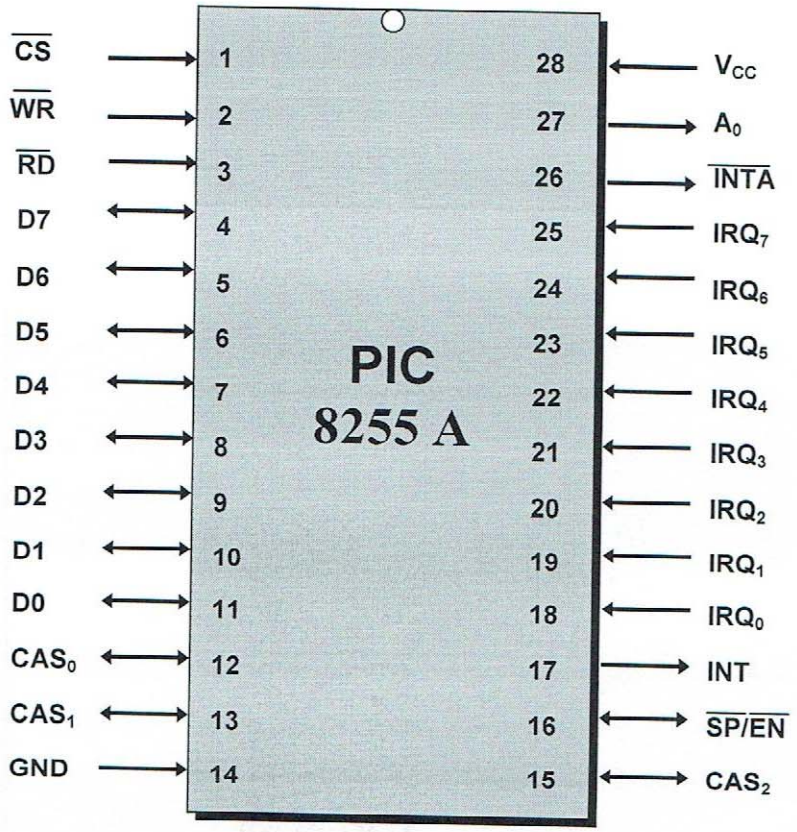


Fig.3 Brochage du PIC

Reconnaissance d'interruption *($\overline{INTA}$):* Ce signal est émis par le 8086.Une séquence d'impulsions sur cette broche signale au PIC qu'il doit déposer le vecteur d'interruption sur le bus de données.

III.2.2 Liaisons avec la périphérie

Broches de requêtes des interruptions (IRQ_0, IRQ_1, ..., IRQ_7): En mode activation par front, le périphérique connecté à une entrée IRQ_i (i= 0,1...,7) doit fournir et maintenir une impulsion positive jusqu'à ce qu'elle soit reconnue par un signal $\overline{INTA}$. Cependant, dans le mode activation par palier, l'activation de IRQ_i se fait par un niveau haut. Dans ce cas, le niveau haut doit être enlevé afin que le 8086 ne le confonde pas avec une autre requête.

Broches de mise en cascade (CAS$_0$, CAS$_1$ et CAS$_2$): Ces lignes servent à contrôler la mise en cascade de deux ou plusieurs PIC. Ce sont des sorties pour le maître et des entrées pour les esclaves.

Broche M/S ou mode buffer ($\overline{SP}/\overline{EN}$, Slave Program/Enable): En mode bufferisé, c'est une sortie qui contrôle le sens de la transmission ($\overline{EN}$). Dans le cas d'une mise en cascade, cette broche est une sortie. Elle désigne le maître quand elle est à l'état haut ou l'esclave quand elle est à l'état bas.

III.2.3 *Fonctionnement du PIC*

Le PIC 8259A peut prendre en charge jusqu'à 8 demandes d'interruptions (IRQ_0-IRQ_7). Comme nous l'avons dit précédemment, Il peut aussi détecter soit une impulsion positive ou un état haut stable sur l'une de ces broches. Cette demande est mémorisée, suivant son rang, dans le registre IRR.

Pendant l'initialisation et en fonction du type de priorité choisi, le PIC vérifie que la dernière requête enregistrée dans le registre IRR n'est pas masquée et qu'elle est plus prioritaire que celle qui est en service. Dans ce cas, la ligne INTR est activée. Si les interruptions sont autorisées au niveau du CPU, celui-ci envoie un cycle de reconnaissance $\overline{INTA}$ en réponse à INTR. A cet instant, l'interruption est considérée comme étant en service et le PIC met à '1' le bit correspondant à cette interruption dans le registre ISR, et remet à '0' le bit correspondant dans le registre IRR.

Dans un système à base de 8086, le cycle de reconnaissance est constitué de deux cycles $\overline{INTA}$. Lors de la réception du 2ème cycle $\overline{INTA}$, le PIC dépose sur le bus de données, le type de l'interruption honorée sur huit bits. Cet octet est formé de cinq bits fixes chargés dans le registre ICW_2. Les trois autres bits les moins significatifs correspondant au rang de la ligne IRQ_i ayant provoqué l'interruption.

Une fois le type d'interruption déposé sur le bus de données, deux situations sont alors possibles.

1- Si le mode de fin d'interruption automatique (AEOI) a été préalablement sélectionné alors, le bit correspondant dans le registre ISR est automatiquement remis à '0' à la fin du 2ème cycle $\overline{INTA}$.
2- Sinon, si le mode de fin d'interruption spécifique (SEOI) a été préalablement choisi, alors le programmeur doit prévoir, dans la routine d'interruption, l'envoi d'un mot de commande (E0I) vers le PIC afin de lui signaler que cette interruption a été servie. Ceci va donc servir à effacer le bit concerné dans le registre ISR.

Mise en cascade du PIC: Le PIC est prévu pour être mis en cascade dans une structure Maître/Esclave. L'initialisation de ce circuit prévoit de préciser au maître si un ou plusieurs esclaves sont branchés sur ces lignes IRQ_i. L'initialisation précise également au PIC esclave son numéro d'identification qui lui permettra de se reconnaître lorsque ce numéro lui est envoyé sur ces lignes de mise en cascade par le PIC maître.

Lorsqu'une ligne IRQ_i du maître sur laquelle est branché un esclave, est activée pour interrompre le 8086, le maître place sur les lignes CAS_0-CAS_2 le rang correspondant à cette ligne. L'esclave est alors en mesure de se reconnaître car sa propre initialisation lui a précisé sur quelle ligne il est branché. Cet échange s'effectue pendant la 1er impulsion du cycle de reconnaissance $\overline{INTA}$ générée par le 8086 ou par le Contrôleur de Bus. Comme nous l'avons dit précédemment, la $2^{ème}$ impulsion $\overline{INTA}$ est utilisée par le PIC esclave pour déposer, sur le bus de données, son vecteur d'interruption codé sur huit bits. La Figure 4 permet de bien assimiler le mécanisme de mise en cascade de 2 PIC..

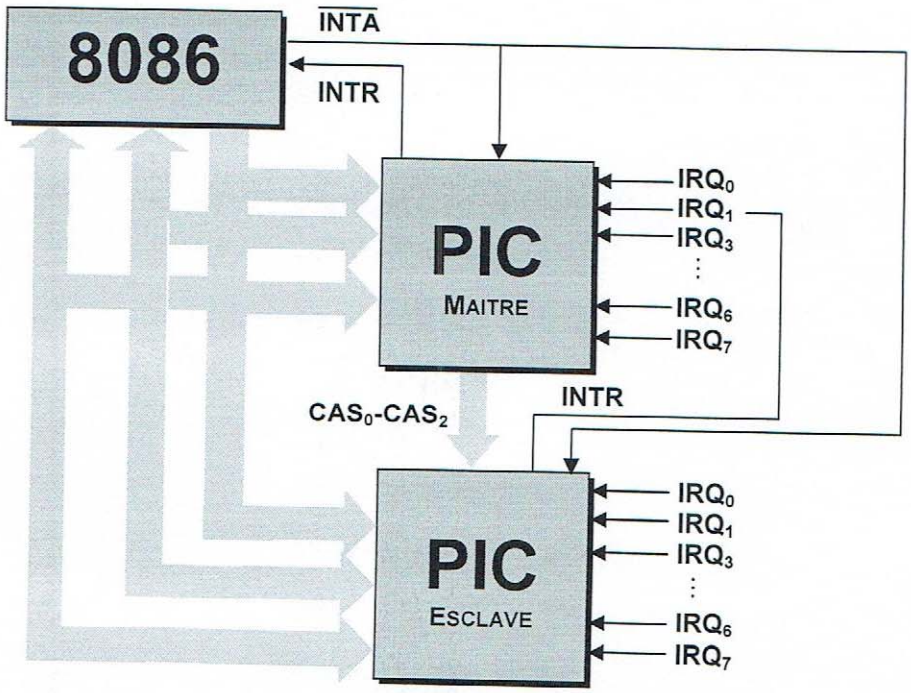


Fig. 4 Exemple de mise en cascade de deux PIC

PIC dans les PC: Le PC-XT dispose d'un seul PIC 8259. L'initialisation de ce circuit par le BIOS configure le registre ICW_2 avec la donnée 08H. Ce qui a pour effet d'attribuer aux interruptions matérielles les numéros 08H, 09H, 0AH,..., 0FH car les trois bits les moins significatifs de ICW_2 sont tous à '0'. Les périphériques pris en charge par le PIC dans les PC-XT sont énumérés dans la Figure 5.

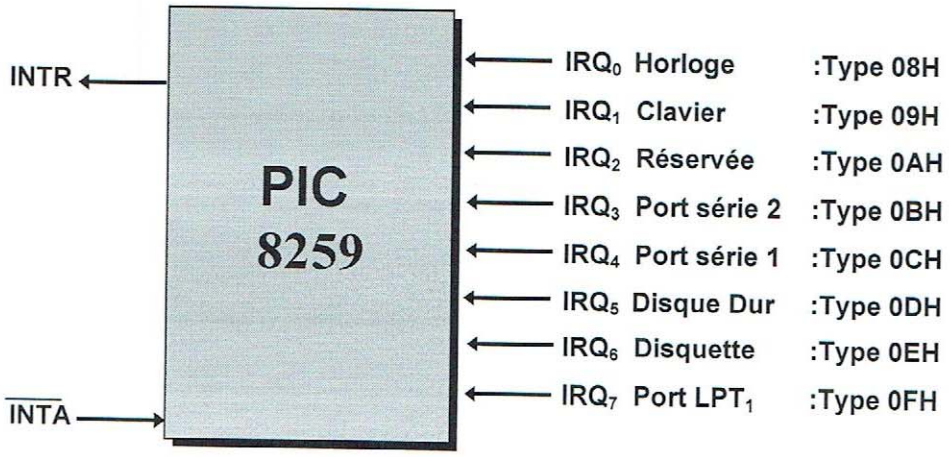


Fig. 5 PIC dans les PC-XT

Remaques:

1- COM_1 et COM_2 désignent les ports série 1 et 2, respectivement.
2- LPT_1 désigne le port parallèle.
4- NDP désigne Numeric Data Processor: Coprocesseur mathématique.

Les PC-AT renferment deux circuits 8259 montés en Maître/Esclave et offrant seize niveaux d'interruptions matérielles. L'initialisation du PIC Maître inscrit l'octet 08H dans le registre ICW_2, ce qui assure une compatibilité entre le PC-XT et le PC-AT. Le registre ICW_2 du PIC esclave, reçoit quant à lui, l'octet 70H pendant la phase d'initialisation, ce qui fixe les types des interruptions matérielles qui lui sont rattachées à 70H, 71H, ..., 77H. Le PIC esclave est branché à la broche IRQ_2 du PIC Maître. La Figure 6 , montre l'affectation des lignes d'interruption dasn une strcture Maître-Esclave.

Remarques:

1- Pour les PC-AT à base de 80286 (Bus ISA), la broche IRQ_2 était utilisée pour réaliser la mise en cascade des 2 PIC (8259A). Cependant, il est clair que la broche IRQ_2 est considérée comme perdue.
2- Grace à l'avènement des bus PCI, il est devenu possible de partager les broches IRQ_i (i=0, 1, 2, ..., 15) par plusieurs périphérique. Cette caractéristique des bus PCI rend les systèmes qui en sont dotés plus généreux en nombre de broches d'interruptions (Cf. Tableaux 4-5).

Tableau 4 PIC maître dans un système PC-AT-PCI

Broche	IRQ_0	IRQ_1	IRQ_2	IRQ_3	IRQ_4	IRQ_5	IRQ_6	IRQ_7
Utilisation	HS	PS/2	IRQ_9	COM2, COM4	COM1, COM3	Son	CDS	LPT_1

Tableau 5 PIC esclave dans un système PC-AT-PCI

Broche	IRQ_8	IRQ_9	IRQ_{10}	IRQ_{11}	IRQ_{12}	IRQ_{13}	IRQ_{14}	IRQ_{15}
Utilisation	RTC	VGA Emul.	Libre	Libre	PS/2	NDP.	IDE (P)	IDE (S)

HS : Horloge Système
Emul.: Emulateur
P : Primaire

PS/2 : Personal System 2
CDS : Contrôleur Disque Souple
S : Secondaire

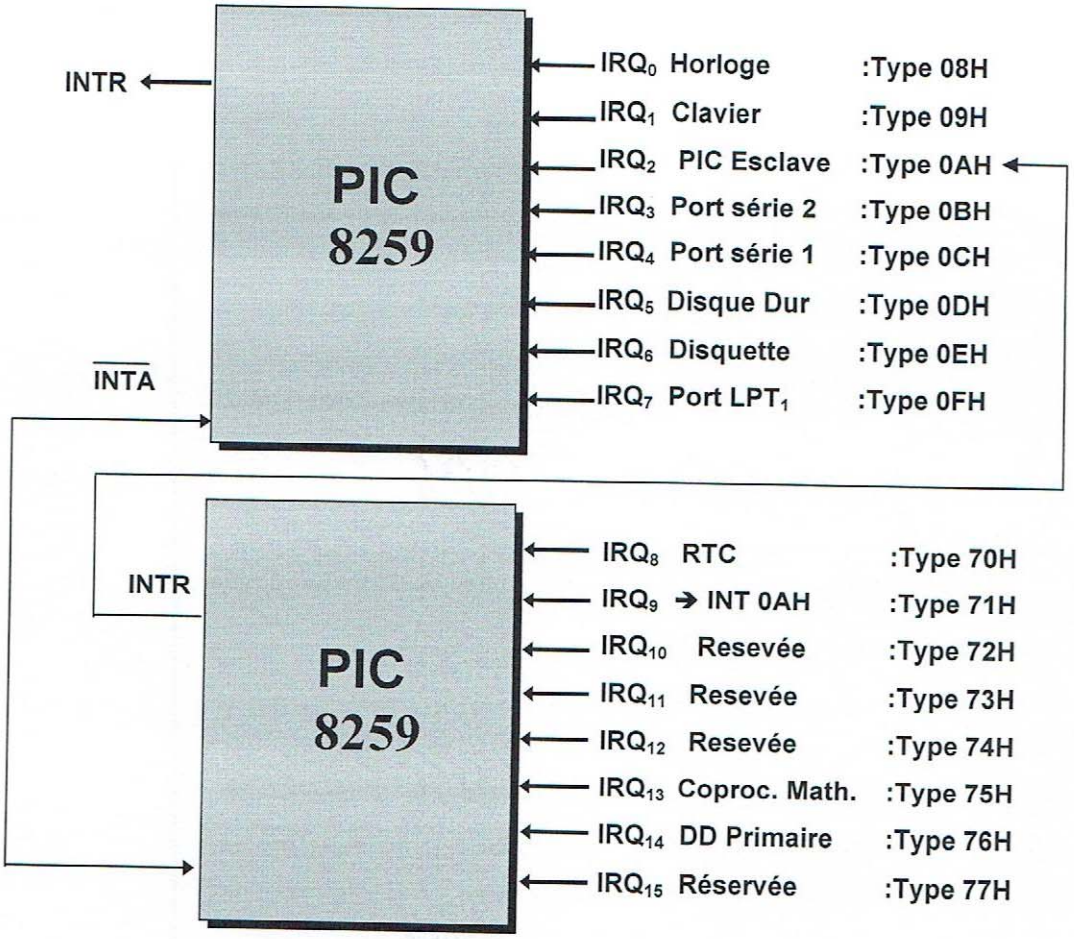


Fig. 6 Mise en cascade des PIC dans les PC-AT

IV Interruptions dans les PC

L'importance des interruptions ne provient pas uniquement du fait qu'elles jouent un grand rôle sur le plan logiciel, mais également du rôle qu'elles jouent dans l'exploitation des requêtes matérielles. L'activation des routines d'interruption s'effectue soit par des événement externes tels que l'unité de disquettes, le clavier, la souris..., par des événements internes tels que la division par zéro,

le fonctionnement du procésseur en mode pas à pas, le point d'arrêt...ou alors lorsque le logiciel fait appel à des instructions logicielles de type INT n.

Dans la famille INTEL (8086, 8088, 80286), il existe 256 niveaux d'interruptions, toutes de type FAR, pouvant être classées selon deux catégories: matérielles et logicielles (Cf. Figure 7).

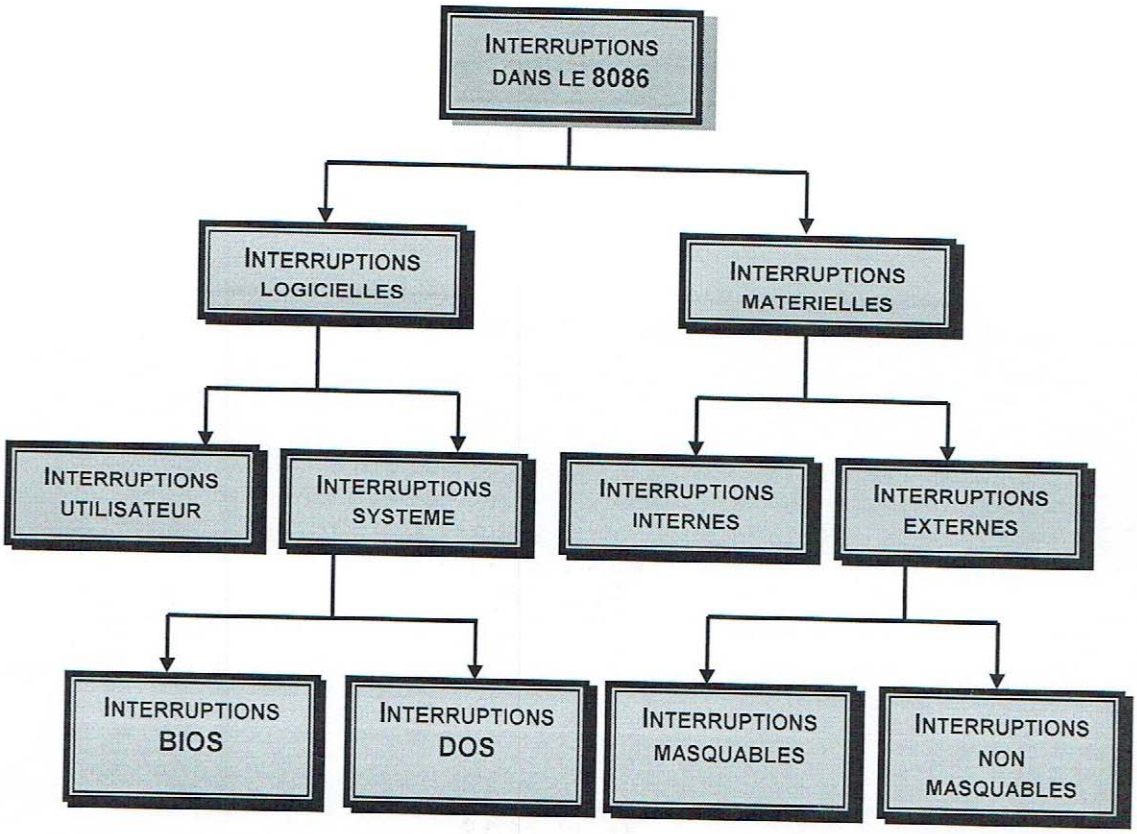


Fig. 7 Interruptions dans le 8086

IV.1 Interruptions matérielles

Comme leur nom l'indique, ce type d'interruptions est déclenché par un événement matériel. On distingue les interruptions internes et externes.

IV.1.1 Interruptions internes

Ce type d'interruptions est automatiquement généré par la rencontre d'un événement lié au déroulement du programme en cours d'exécution. Les causes de ces interruptions ainsi que leurs vecteurs respectifs sont figés et n'acceptent aucune modification (Interruptions câblées et spécifiques au processeur).

Tableau 5 Interruptions internes

N° du vecteur	Adresse du vecteur	Natute de l'interruption
00H	00H – 03H	Division par zéro
01H	04H – 07H	Pas à pas
03H	0CH – 0FH	Point d'arrêt
04H	10 H – 13H	Dépassement

IV.1.2 Interruptions externes

Contrairement aux Interruptions internes les interruptions externes sont générées par des périphériques reliés au microprocesseur via le PIC sauf la NMI et le RESET. On distingue trois types d'interruptions externes.

Interruption non masquable (NMI, Non Maskable Interrupt): Comme son nom l'indique, l'interruption non masquable ne dépend pas de l'état du flag I.

Tableau 6 Interruption non masquable (NMI)

N° du vecteur	Adresse du vecteur	Natute de l'interruption
02H	08H – 0BH	Interruption non masquable

Interruptions masquables (INTR, Interrupt Request): Comme nous l'avons précédemment expliqué, le numéro du vecteur d'interruption est fourni par le PIC. Contrairement à la NMI, ce type d'interruption dépend de l'état du flag I.le déroulement de ce type d'interruption est résumé en Figure 8.

Tableau 7 Interruptions masquables (INTR)

N° du vecteur	Adresse du vecteur	Natute de l'interruption
08H	20H – 23H	Horloge (18.2 Hz)
09H	24H – 27H	Clavier
0BH	2CH – 2FH	COM2 (RS 232)
0CH	30 H – 33H	COM1 (RS 232)
0DH	34H – 37H	Contrôleur de disque dur
0EH	38H – 3BH	Contrôleur de disquette
0FH	3CH – 3FH	LPT1 (Imprimante)

Les interruptions masquables obéissent toutes à une même séquence d'opérations que le microprocesseur doit effectuer avant et après l'exécution du sous programme de l'interruption qui a demandé l'attention du microprocesseur. Cette séquence peut se résumer sous la forme d'un organigramme comme suit:

Reset (Remise à zéro du micrprocesseur): Lors de la mise à '0' de la broche RESET du microprocesseur (Reset Froid), le couple CS:IP prend la valeur FFFFH:0000H dont le contenu désigne une instruction de saut au programme

BIOS: Branchement à l'adresse FFFFH:0000H

IV.2 Interruptions logicielles

IV.2.1 Interruptions système

Interruptions BIOS: Ce type d'interruptions est implanté en mémoire ROM. Elles permettent, notamment, de prendre en charge la gestion des périphériques, les Interruptions gérées par programme et Pointeurs des tables des paramètres.

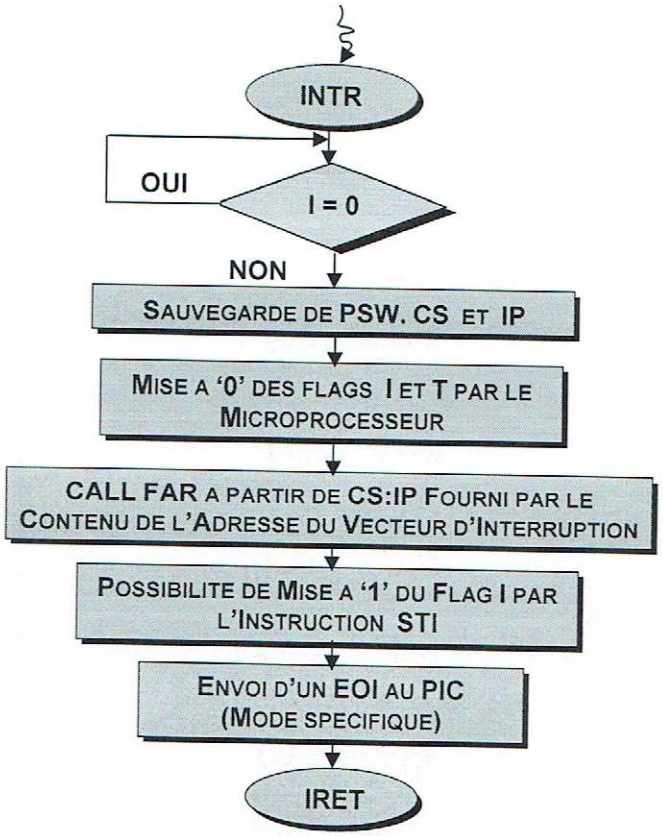


Fig. 8 Déroulement d'une interruption INTR

Tableau 8 Gestion des périphériques

N° du vecteur	Adresse du vecteur	Natute de l'interruption
05H	14H – 17H	Impression de l'écran
10H	28H – 31H	Affichage (gestion de l'écran)
11H	32H – 35H	Description de la configuration
12H	36 H – 39H	Taille de la RAM
13H	3AH – 3DH	Accès à la disquette/Disque dur
14H	3EH – 41H	Accès à l'interface série
16H	46H – 49H	Gestion du clavier
17H	4AH – 4DH	Sortie imprimante
18H	4EH – 51H	Bootstrap à partir de la disquette

Tableau 9 Interruptions gérées par programme

N° du vecteur	Adresse du vecteur	Natute de l'interruption
1BH	6CH – 6FH	Interruption de programme par Ctrl-Brk
1CH	70H – 73H	Contrôle du compteur d'horloge

Tableau 10 Accès aux tables des paramètres

N° du vecteur	Adresse du vecteur	Natute de l'interruption
1DH	74H – 77H	Table des paramètres VIDEO
1EH	78H – 7BH	Table des paramètres unité de disquettes
1FH	7CH – 7FH	Table des caractères supplémentaires

Interruptions MS DOS: Ces interruptions sont gérées par le DOS et occupent les numéros des vecteurs 20H-3FH. Le Dos permet ainsi de gérer, particulièrement les Interruptions et les fonctions du DOS.

Tableau 11 Interruptions et les fonctions du DOS

N° du vecteur	Adresse du vecteur	Natute de l'interruption
20H	80H – 83H	Terminaison de programme
21H	84H – 87H	Fonctions du DOS (00H - 09H)
23H	8CH – 8FH	Annulation de l'effet Ctrl-Brk

IV.2.2 Interruptions utilisateur

Ce type d'interruptions est réservé aux programmes d'application et à toutes les extensions que l'utilisateur peut envisager. A titre d'exemple, prenons celui de la gestion de la mémoire EMS (Expanded Memory System)

Tableau 12 Un exemple des interruptions utilisateur

N° du vecteur	Adresse du vecteur	Natute de l'interruption
33H	CCH - CFH	Gestion de la souris
67H	25CH - 25FH	EMS Manager

IV.3 Vectorisation des interruptions

IV.3.1 Table des vecteurs d'interruptions

Nous rappelons que les interruptions sont organisées en vecteurs d'interruptions, mémorisés dans une table, dite table des vecteurs d'interruptions (TVI), occupant

le premier Kilo octets du plan mémoire du PC..

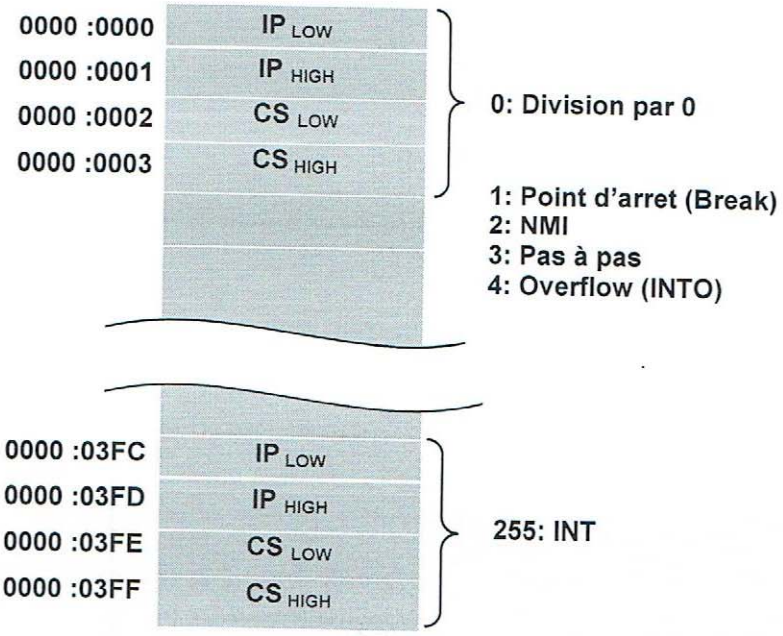


Fig. 9 Les 256 types d'interruptions dans le 8086

IV.3.2 Branchement à une routine d'interruption

Après la lecture du type d'interruption, le 8086 effectue les opération suivantes:

1- Calcul de l'adresse physique du pointeur de la table des vecteurs d'interruptions (TVI), et ce en multipliant le type par 4.
2- Sauvegarde du registre d'état dans la pile.
3- Mise à zéro des flags I et T pour bloquer les interruptions et le fonctionnement en mode pas à pas.
4- Sauvegarde du registre CS dans la pile et chargement du nouveau CS à l'aide de la partie haute du pointeur de la TVI.
5- Sauvegarde ddu registre IP dans la pile et chargement du nouvel IP à l'aide de la partie basse du pointeur de la TVI.

L'instruction IRET permet la restitution des registres IP,CS et PSW à partir de la pile pour remettre le processeur dans son contexte initial.

Remarque: Dans le cas où le PIC a été configuré pour travailler en mode spécifique, la fin du traitement de l'interruption lui est signifiée par l'envoi d'un EOI.

IV.4 Exemples d'applications des interruptions

IV.4.1 Interruption matérielle cyclique INT 08 H

Dans le cas des PC, PC-XT et PC-AT, le CTC (Counter Timer Circuit) fournit un appel à l'interruption INT 08H qui est transmise via le PIC au 8086, 18.2 fois par seconde. Etant fixe, 18.2 Hz, cette fréquence convient parfaitement pour la génération des temporisations. La principale utilisation que fait la BIOS de cette interruption est l'arrêt du moteur du lecteur de disquette si aucun accès ne s'est produit au bout d'un certain délai. Le BIOS dirige le vecteur de cette interruption vers une routine qui incrémente de '1' le mot situé à l'adresse 0040:006C. A titre d'exemple, une temporisation de 14 secondes correspondrait à environ FF H appels par le CTC.

IV.4.2 Lecture et modification de l'heure au moyen de l'interruption INT 1A H du BIOS

Tous les PC-AT disposent d'une horloge temps réel alimentée par une pile qui leur permet de charger le compteur horaire à partir de la RTC lors du lancement de la machine. Quant aux PC-XT, lesquels sont dépourvus d'une RTC, ils fixent leur compteur horaire à '0' lors de chaque mise sous tension. Dans les deux cas, une valeur compteur est renvoyée par le BIOS dans la paire de registre CX o DX qui est incrémenté de '1' par celui-ci toutes les 65536 fois. Ce compteur ne représente pas l'heure en format heure et minute. Pour cela, nous devons d'abord écrire le compteur d'horaire en nombre d'impulsions sous la forme:

Compteur = CX * 65536+DX

Puis en secondes tel que:

Compteur (Secondes) = Compteur / 18.2 secondes

Fonctions 00 H et 01 H de l'interruption BIOS INT 1A H

Fonction 00 H: Elle permet de lire l'heure du système en hexadécimal.

Entrée : AH = 00 H
Sortie : CX o DX en Hexadécimal

Fonction 01 H: Elle permet de régler l'heure du système en hexadécimal.

Entrées : AH = 01 H
: CX o DX en Hexadécimal
Sortie : Néant

Fonctions 02 H - 07 H: Ces fonctions ne sont disponibles que sur les PC-AT.

Fonction 02 H: Elle permet de lire l'heure du système en format BCD

Entrée : AH = 02 H
Sortie : CH = Heures, CL = Min., DH = Sec et DL = 1/100 Sec

Fonction 03 H: Elle permet de régler l'heure du système en format BCD

Entrée : AH = 03 H
: CH = Heures, CL = Min., DH = Sec et DL = 1/100 Sec
Sortie : Néant

Fonction 04 H: Elle permet de lire la date du système en format BCD

Entrée : AH = 04 H
Sortie : CX = Années, DH = Mois et DL = Jours

Fonction 05 H: Elle permet de régler la date du système en format BCD

Entrée : AH = 05 H
: CX = Années, DH = Mois et DL = Jours
Sortie : Néant

Fonction 06 H: Elle permet de régler l'alarme du système en format BCD

Entrée : AH = 06 H
: CH = Heures, CL = Min., DH = Sec et DL = 1/100 Sec
Sortie : Néant

Fonction 07 H: Elle permet d'annuler l'alarme du système en format BCD

Entrée : AH = 07 H
Sortie : Néant

IV.4.3 Lecture et modification de l'heure au moyen de l'interruption INT 21 H du DOS: Fonctions 2C H et 2D H

Fonction 2C H: Elle permet de lire l'heure du système en format heure. Elle utilise la fonction 00 H de l'interruption 1AH du BIOS pour convertir le compteur horaire en format heure. Elle permet ainsi de lire l'heure actuelle du système en Hexadécimal.

Entrée : AH = 2C H
Sortie : CH = Heures, CL = Min., DH = Sec et DL = 1/100 Sec

Fonction 2D H: Elle permet de régler l'heure du système en format heure. Elle utilise la fonction 01 H de l'interruption 1AH du BIOS pour convertir le compteur horaire en format heure. Elle permet ainsi de modifier l'heure actuelle du système en Hexadécimal.

Entrées : AH = 2D H
: CX = Heures et DX = Minutes en Hexadécimal pour les PC-AT.
Sorties : AL = 00 H Tout est bon
: AL = FF H Heure impossible.

CHAPITRE 9

CIRCUIT D'ENTREES ET DE SORTIES PARALLELE ET PROGRAMMABLE

I Introduction

Ce qui apporte le plus de complication dans la conception d'un système en logique programmée, c'est la communication microprocesseur-périphérie. La raison tient essentiellement au fait qu'il n'y a aucune standardisation dans les périphériques de sorte qu'un microprocesseur ne peut pas, en général, commander directement les périphériques. Un circuit intégré d'adaptation appelé interface s'avère nécessaire entre le microprocesseur et les périphériques. A de très rares exceptions près, l'interface établit donc une compatibilité entre les E/S du microprocesseur et celle des périphériques et ceci à plusieurs niveaux:

1- Au niveau du type de transmission: série ou parallèle. Le microprocesseur délivre et reçoit les données sur 8 ou 16 fils parallèles, alors qu'il existe des périphériques tel qu'un téléimprimeur qui véhicule les données un bit après l'autre sur un seul fil (transmission série).

2- Au niveau du code: vu qu'il existe plusieurs codes pour la représentation des chiffres, des lettres et des symboles tels que les codes ASCII (American Standard Code for Information Interchange), EBCDIC (Extended Binary Coded Decimal Interchange Code), ISO (International Organization for Standardization), ..., il est nécessaire de procéder à un transcodage si le microprocesseur et les périphériques ne travaillent pas dans le même code.

3- Au niveau de la vitesse de transmission: étant électromécaniques, les périphériques sont lents en comparaison avec le microprocesseur. Ainsi celui-ci doit faire transiter les données par un registre appartenant à l'interface qui sert de mémoire de verrouillage, adressé soit comme une case mémoire soit séparément de celle-ci.

II PPI (Programmable Peripheral Interface)

Le PPI est un circuit d'Entrées/Sorties de la famille Intel. Il permet de réaliser la liaison parallèle entre le 8086 et ses périphériques. Le PPI communique avec le

microprocesseur par l'intermédiaire du bus de données, d'adresses et de contrôle. Les dialogues avec la périphérie sont assurés par deux groupes de contrôles A et B.

Groupe A – Le port A (PA_0 -PA_7) et le port C (PC_7-PC_4).
Groupe B – Le port B (PB_0 -PB_7) et le port C (PC_3-PC_0).

La programmation interne du PPI est fonction du périphérique connecté.

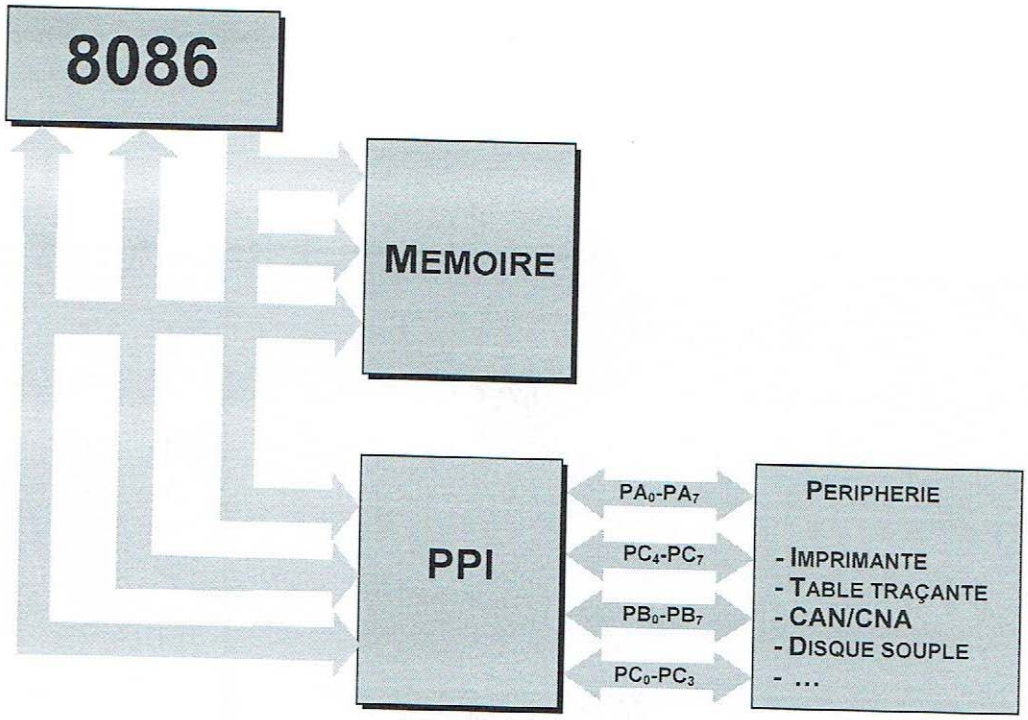


Fig. 1 PPI dans une structure 8086

III Description du PPI

Le PPI se présente sous la forme d'un boîtier DIL 40 broches mono tension (0, +5V). Nous distinguons les signaux d'interface avec le microprocesseur et les signaux d'interface avec le périphérique.

III.1 Liaisons avec le 8086

III.1.1 Bus de données D_0-D_7

Les 8 lignes bidirectionnelles, et à logiques trois états sont directement connectés au bus de données du 8086 (AD_0-AD_7). Elles assurent l'échange bidirectionnel des données entre le 8086 et le PPI.

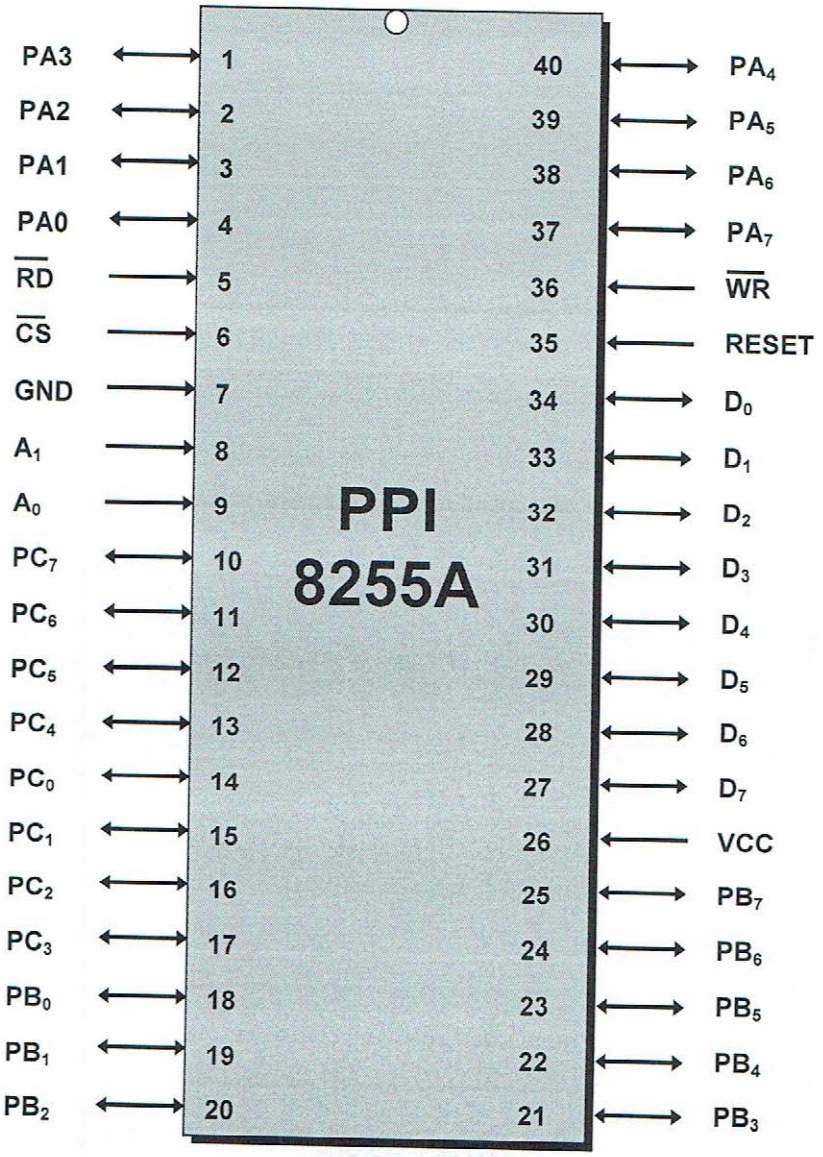


Fig. 2 Brochage du PPI

III.1.2 Bus d'adresses $\overline{CS}$, A_0 et A_1

Les broches A_0 et A_1 permettent la sélection de l'un des trois registres de données associés aux trois ports A, B et C et le registre de contrôle. La broche $\overline{CS}$ set à Sélectionner un boîtier PPI.

$\overline{CS}$ = { 0 boîtier sélectionné ; 1 boîtier non sélectionné.

Tableau 1 Sélection des ports/ registres du PPI

A_1	A_0	Port ou registre sélectionné
0	0	Port A ou RDA
0	1	Port B ou RDB
1	0	Port C ou RDC
1	1	Registre de contrôle

Remarque: Les deux lignes A_0 et A_1 peuvent être connectées aux AD_0 et AD_1 du bus d'adresses du 8086.

III.1.3 Alimentation

Etant un circuit TTL, le PPI accepte une alimentation unique (0, 5v).

III.1.4 RESET

Cette broche sert à initialiser le PPI. Doit être connecté au RESET du 8284.

III.1.5 Bus de contrôle

Comme le montre le Tableau 2, l'écriture et la lecture des registres du PPI s'effectuent sous contrôle des signaux $\overline{WR}$ et $\overline{RD}$.

Tableau 2 Lecture/Ecriture des ports/registres du PPI

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Opération effectuée
0	0	0	1	0	Lecture port A
0	1	0	1	0	Lecture port B
1	0	0	1	0	Lecture port C
1	1	0	1	0	Condition illégale
0	0	1	0	0	Ecriture port A
0	1	1	0	0	Ecriture port B
1	0	1	0	0	Ecriture port C
1	1	1	0	0	Ecriture registre de contrôle
X	X	X	X	1	Haute impédance
X	X	1	1	0	Condition illégale

III.2 Liaisons avec la périphérie

Le PPI possède 03 ports d'Entrées/Sorties de 8 bits chacun. Ils peuvent être configurés pour un large éventail d'applications. Ils possèdent chacun des caractéristiques qui lui sont propres.

III.2.1 Port A (PA_0 -PA_7)

En sortie, ce port joue le rôle d'un latch/buffer. En entrée, il joue le rôle d'un latch uniquement.

III.2.2 Port B (PB_0 -PB_7)

En sortie, ce port joue le rôle d'un latch/buffer. En entrée, il joue le rôle de latch uniquement. Les lignes PB_0 -PB_7 peuvent aussi driver des transistors Darlington (3 mA).

III.2.3 Port C (PC_0 –PC_7)

En sortie, ce port joue le rôle d'un latch/buffer. En entrée, il joue le rôle d'un buffer uniquement. Les lignes PC_0 – PC_7 peuvent aussi driver des transistors Darlington (3 mA). Le port C peut être divisé en deux ports de 4 bits chacun. Les 4 bits de chaque port, peuvent être utilisés comme signaux de contrôles et d'état en conjonction avec les ports A et B formant ainsi les groupes A et B.

IV Description interne du PPI

Le PPI vu du 8086 se présente sous la forme de 4 registres adressables.

1- Registre de données du port A (RDA)
2- Registre de données du port B (RDB)
3- Registre de données du port C (RDC)
4- Registre de contrôle (RC)

IV.1 Registres de données

IV.1.1 Registre de données du port A (RDA)

En sortie ce registre sert de latch et de buffer. En entrée, il sert de latch uniquement.

IV.1.2 Registre de données du port B (RDB)

En sortie ce registre sert de latch et de buffer. En entrée, il sert de buffer uniquement.

IV.1.3 Registre de données du port C (RDC)

En sortie ce registre sert de latch et de buffer. En entrée, il sert de buffer uniquement.

IV.2 Registre de contrôle (RC)

Ce registre sert à configurer le PPI. Il constitue deux configurations distinctes dites première et deuxième configurations.

IV.2.1 Première configuration du registre de contrôle

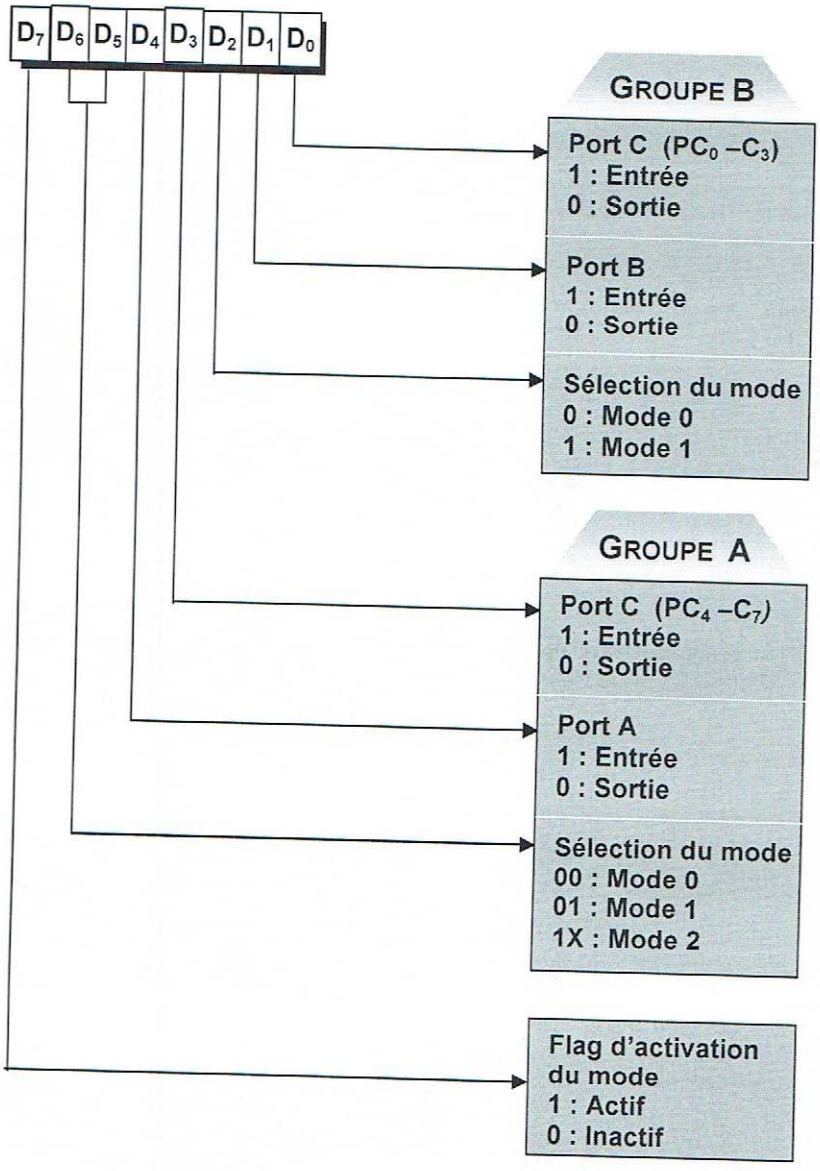


Fig. 3 Première configuration du registre de contrôle du PPI

IV.2.2 Deuxième configuration du registre de contrôle

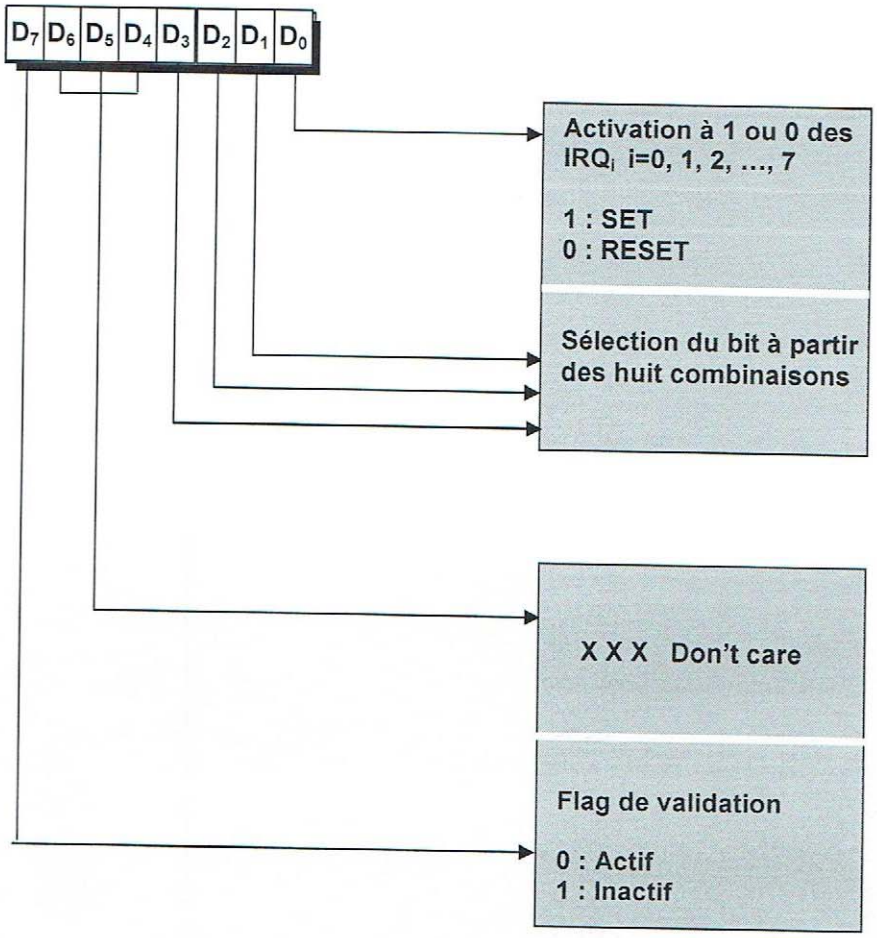


Fig. 4 Deuxième configuration du registre de contrôle du PPI

V Description opérationnelle du PPI

Il existe trois modes d'opération qui peuvent être sélectionnés par programme. On peut les réunir dans le Tableau 3.

V.1 Mode 0 (M_0): Mode d'Entrées/Sorties de base

Le mode offre une configuration simple d'Entrées/Sorties sans signaux de dialogue.

V.1.1 Caractéristiques

1- Deux ports de 8 bits et deux de 4 bits.
2- N'importe quel port peut avoir une configuration en entrée ou en sortie.
3- Les sorties sont verrouillées (latched).
4- Les entrées ne sont pas verrouillées.
5- 16 combinaisons d'Entrées/Sorties sont possibles dans ce mode.

Tableau 3 Description opérationnelle du PPI

Broches	Mode 0		Mode 1		Mode 2
	Entrée	Sortie	Entrée	Sortie	Groupe A
PA_0	E	S	E	S	↔
PA_1	E	S	E	S	↔
PA_2	E	S	E	S	↔
PA_3	E	S	E	S	↔
PA_4	E	S	E	S	↔
PA_5	E	S	E	S	↔
PA_6	E	S	E	S	↔
PA_7	E	S	E	S	↔
PB_0	E	S	E	S	Mode 0 ou Mode 1 uniquement
PB_1	E	S	E	S	
PB_2	E	S	E	S	
PB_3	E	S	E	S	
PB_4	E	S	E	S	
PB_5	E	S	E	S	
PB_6	E	S	E	S	
PB_7	E	S	E	S	
PC_0	E	S	$INTR_B$ (S)	$\overline{INTR_B}$ (S)	↔
PC_1	E	S	$\overline{IBF_B}$ (S)	$\overline{OBF_B}$ (S)	↔
PC_2	E	S	STB_B (E)	ACK_B (E)	↔
PC_3	E	S	$\overline{INTR_A}$ (S)	$INTR_A$ (S)	$\overline{INTR_A}$ (S)
PC_4	E	S	$\overline{STB_A}$ (E)	↔	STB_A (E)
PC_5	E	S	IBF_A (S)	↔	$\overline{IBF_A}$ (S)
PC_6	E	S	↔	$\overline{ACK_A}$ (E)	$\overline{ACK_A}$ (E)
PC_7	E	S	↔	$\overline{OBF_A}$ (S)	$\overline{OBF_A}$ (S)

V.2 Mode 1 (M1): Mode d'Entrées/Sorties avec signaux de dialogue

Ce mode offre une configuration d'Entrées/Sorties avec signaux de dialogue.

V.2.1 Caractéristiques

1- Deux groupes A et B.
2- Les 8 bits du port A ou B peuvent être en entrée ou en sortie et sont tous verrouillés (latched) en entrée et sortie.
3- Les 8 bits du port C sont utilisés comme signaux de contrôle et d'état des ports A et B.

V.2.2 Définitions des signaux de contrôle et d'état en entrée

$\overline{STB}_{A/B}$ (Strobe): Un état bas sur ce signal indique que le latch d'entrée est entrain de se remplir.

$IBF_{A/B}$ (Input Buffer Full): Un état haut sur ce signal indique que le latch d'entrée est plein. $\overline{IBF}_{A/B}$ est mis à '1' quand $\overline{STB}_{A/B}$ est à '0'. Il passe à '0' sur le front montant de RD.

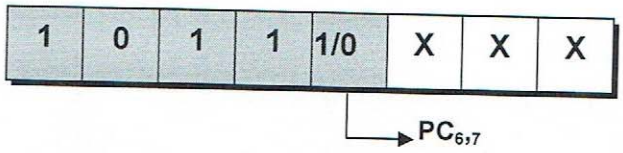
$INTR_{A/B}$ (Interrupt Request): Un état haut sur ce signal est utilisé pour interrompre le 8086 pour l'entrée d'une donnée.

Remarques:

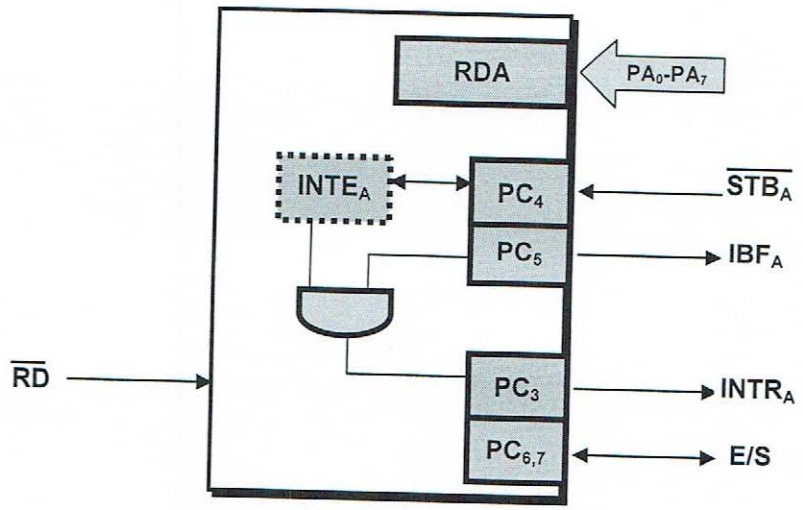
1- Quand le 8255A est en mode M_1 ou M_2, les signaux de contrôle sont utilisés pour interrompre le 8086.
2- La validation des interruptions pour le groupe A en M_1 et en sortie se fait à partir de PC_6. Par ailleurs, la mise à 1/0 de la bascule $INTE_{A/B}$ pour les groupes A et B obéit à la règle suivante:

$$INTE_{A/B} = \begin{cases} 1 & \text{Interruption validée} \\ 0 & \text{Interruption inhibée} \end{cases}$$

Exemple de chronogramme du groupe A en M_1 et en entrée: Avant de donner le chronogramme des signaux de ce groupe, nous devons d'abord montrer leur configuration en mode M_1 et en entrée.



(a) Première configuration du registre de contrôle pour le groupe A en M_1



(b) Etats interne et externe du PPI pour le groupe A en M_1

0	X	X	X	1	0	0	1

(c) Deuxième configuration du registre de contrôle pour le groupe A; validation des interruptions à travers PC_4

Fig. 5 Configuration du groupe A en M_1 et en entrée à travers (a), (b) et (c)

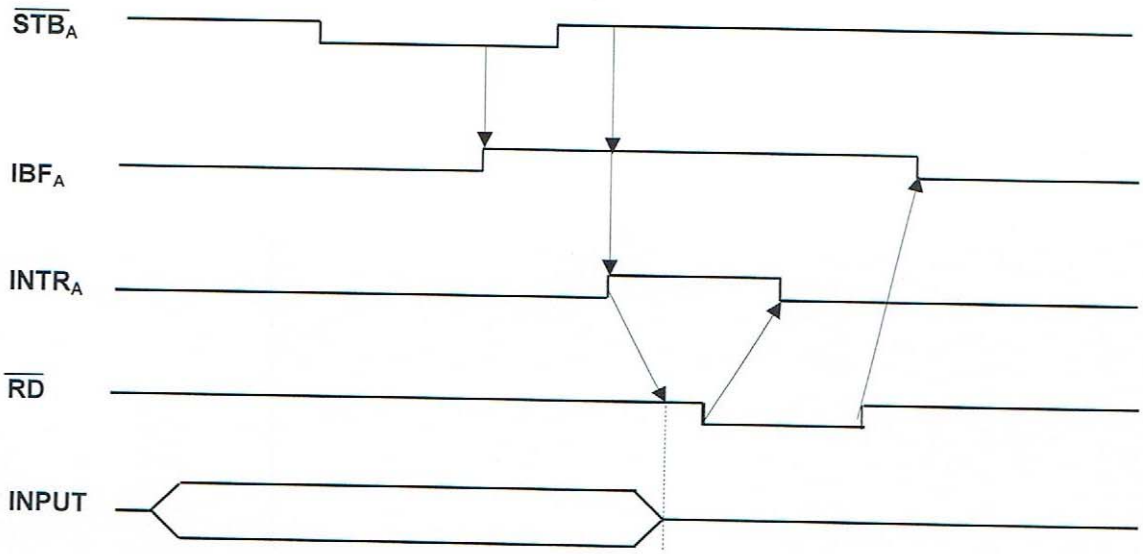


Fig. 6 Chronogramme du groupe A en mode, entrée

V.2.3 Définitions des signaux de contrôle et d'état en sortie

$\overline{OBF_{A/B}}$ (Output Buffer full): *Un* état bas sur ce signal indique que le 8086 a effectué une opération de sortie sur le port en question. $\overline{OBF_{A/B}}$ est mis à 0 par le front montant de $\overline{WR}$ et passe à '1' quand $\overline{ACK_{A/B}}$ passe à '0'.

$\overline{ACK_{A/B}}$ (Acknowledge): Un état bas sur ce signal informe le 8255A que la donnée envoyée par le port en question (port A ou B) a été lue correctement par le périphérique.

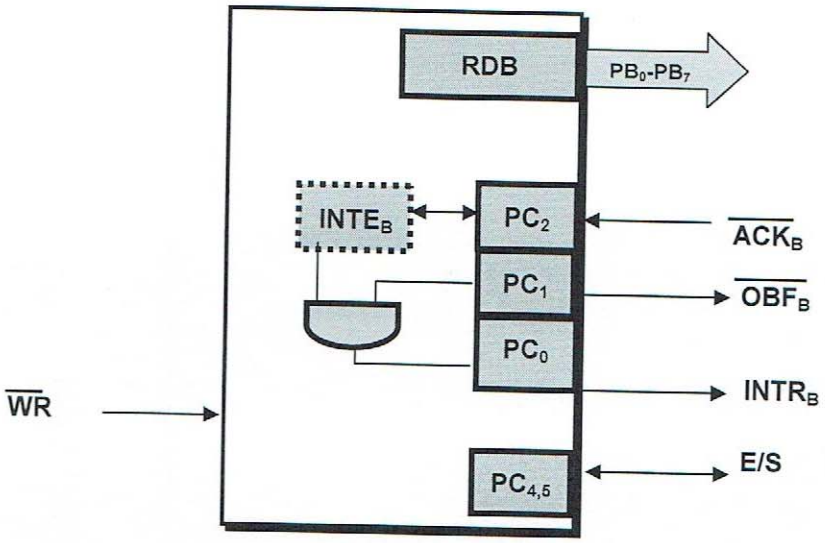
$INTR_{A/B}$ (Interrupt Request): Un état haut sur ce signal est utilisé pour interrompre le 8086 pour l'envoi de la prochaine donnée.

Remarque: La validation des interruptions pour le groupe B en mode 1 entrée se fait à partir de PC_2.

Exemple de chronogramme du groupe B en mode 1 et en sortie: Avant de donner le chronogramme des signaux de ce groupe, nous devons d'abord montrer leur configuration en mode M_1 et en sortie.

1	X	X	X	X	1/0	0	X

(a) Première configuration du registre de contrôle pour le groupe B en M_1



(b) Etats interne et externe du PPI pour le groupe B en M_1

0	X	X	X	0	1	0	1

(c) Deuxième configuration du registre de contrôle pour le groupe B; validation des interruptions à travers PC_2

Fig. 7 Configuration du groupe B en M_1 et en sortie à travers (a), (b) et (c)

V.3 Mode 2 (M_2): Mode bidirectionnel avec signaux de dialogue

Ce mode bidirectionnel utilise lui aussi les signaux d'asservissement. On rappelle qu'il n'est valable que pour le groupe A uniquement.

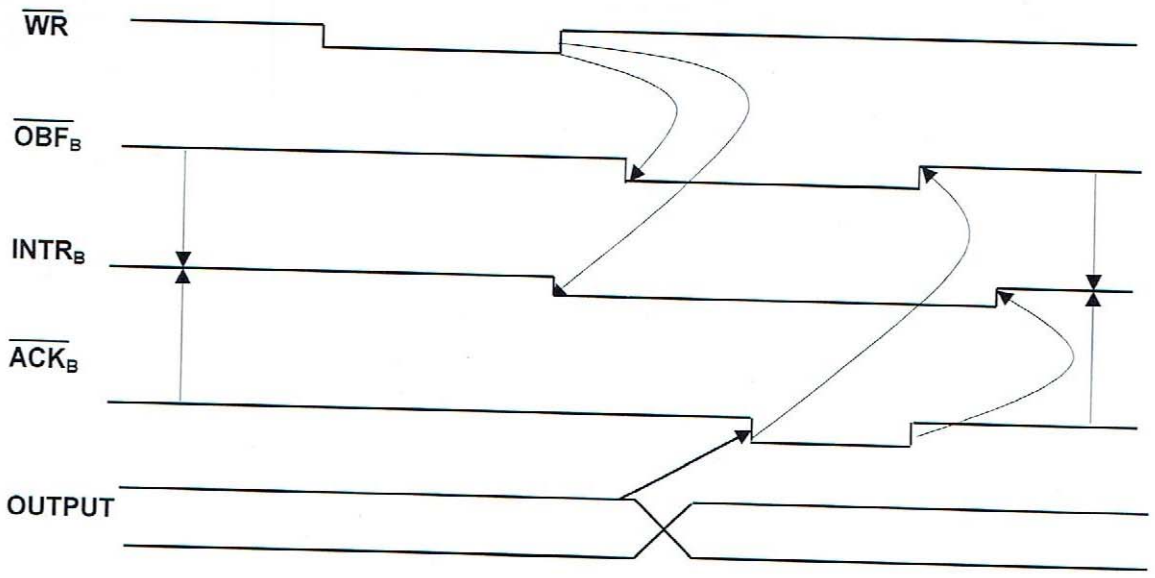


Fig. 8 Chronogramme pour le groupe B en mode 1 et en sortie

V.3.2 Définitions des signaux de contrôle et d'état

En sortie:

$INTR_A$ (Interrupt Request): Un état haut sur ce signal sert à interrompre le 8086 aussi bien en entrée qu'en sortie.

$\overline{OBF}_A$ (Output Buffer Full): Un état bas sur ce signal indique que le 8086 a effectué une opération de sortie sur le port A.

$\overline{ACK}_A$ (Acknowledge): Un état bas sur ce signal informe le 8255A que la donnée envoyée par le port A a été lue correctement par le périphérique.

$INTE_{A1}$ (Interrupt Enable): Cette bascule représente l'état de la bascule INTE associée à $\overline{OBF}_A$ et contrôlée par PC_6.

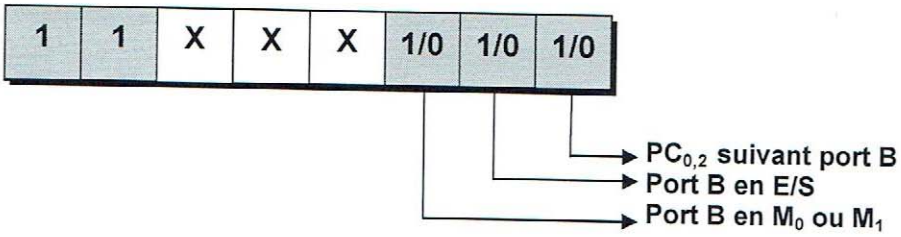
En entrée:

$\overline{STB}_A$ (Strobe): Un état bas sur ce signal indique que le latch d'entrée du port A est entrain de se remplir.

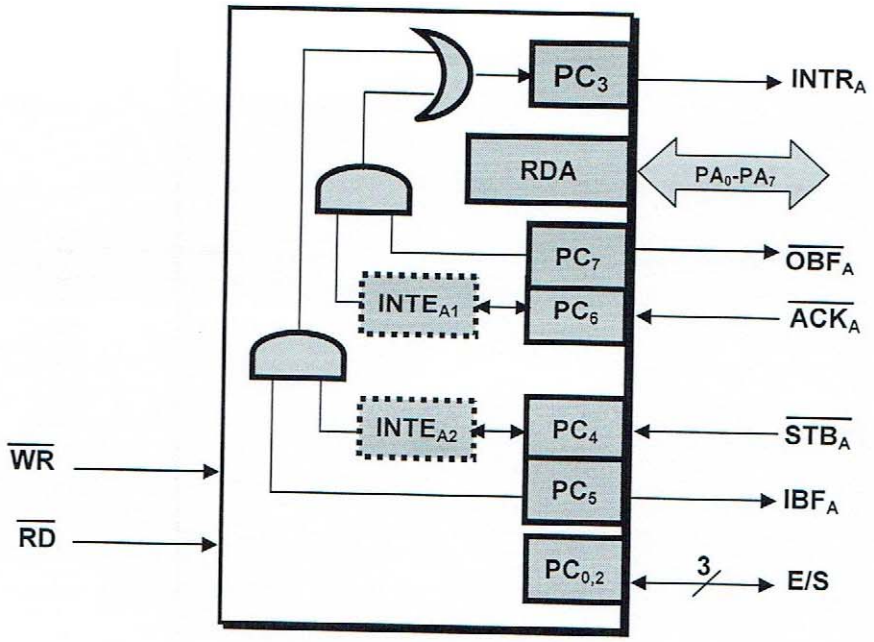
IBF_A (Input Buffer Full): Un état haut sur ce signal indique que le latch d'entrée du port A est plein.

$INTE_{A2}$ (Interrupt Enable): Cette bascule représente l'état de la bascule $INTE_A$ associée à IBF_A et contrôlée par PC_4.

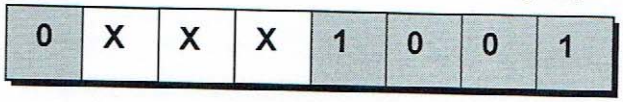
V.3.3 Configuration du PPI en mode 2



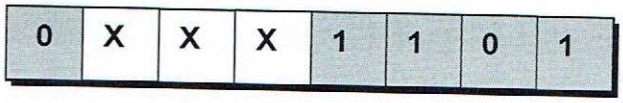
(a) Première configuration du registre de contrôle en mode 2



(b) Etats interne et externe du PPI pour le groupe A en mode 2



c) Deuxième configuration du registre de contrôle pour le groupe A en M_2 (entrée); validation des interruptions à travers PC_4



d) Deuxième configuration du registre de contrôle pour le groupe A en M_2 (sortie); validation des interruptions à travers PC_6

Fig. 9 Configuration du groupe A en M_2 à travers (a), (b), (c) et (d)

Remarque: Il est à noter que le chronogramme pour le mode 2 est similaire à celui du mode 1 combiné en entrée et en sortie.

VI Programmation du PPI

La programmation du PPI se fait à travers le registre de contrôle commun aux trois ports. Lors de l'initialisation, le registre de contrôle reçoit un, deux, trois ou quatre mots suivant:

1- Le mode de fonctionnement des ports A, B et C.
2- Le masque des interruptions en mode M_1 ou M_2 du port C.

Remarques:

1- A l'initialisation, tous les ports du PPI sont en mode M_0 et en entrées.
2- Un '1' valide les interruptions et un '0' les invalide.

IV.1 Exemple de programmation du PPI

IV.1.1 Premier exemple

Le port A se comporte comme 8 sorties, le port B comme 8 entrées. Les registres de données des ports A et B occupent les adresses 00H et 01H, respectivement. Le registre de contrôle, quant à lui, occupe la position 03H. Il n'y a pas de signaux de dialogue ou d'asservissement.

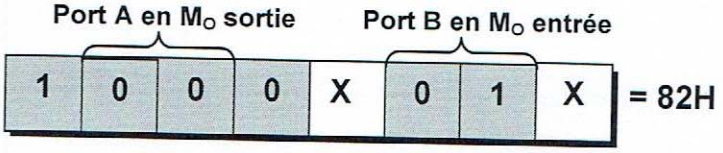


Fig. 10 Sélection du mode de fonctionnement

Initialiser le PPI revient donc à envoyer la valeur 82H vers le registre de contrôle dont l'adresse est 03H de la manière suivante:

```
MOV  AL, 82H
OUT  03H, AL
```

IV.1.2 Deuxième exemple

Le port A se comporte en bidirectionnel. Le port B en M1 et en sortie. Les registres de données des ports A et B occupent les adresses 00H et 01H, respectivement. Le registre de contrôle, quant à lui, occupe la position 03H. Les broches $PC_{0,2}$ sont des signaux d'asservissement du port B. Les interruptions sont validées pour les deux ports.

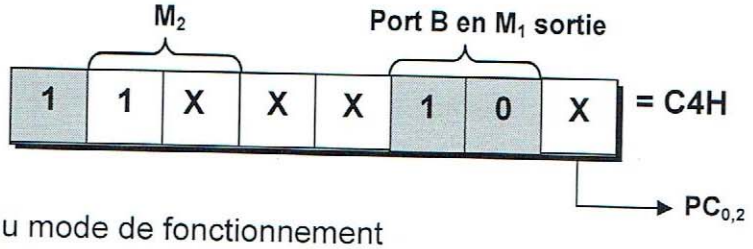


Fig. 11 Sélection du mode de fonctionnement

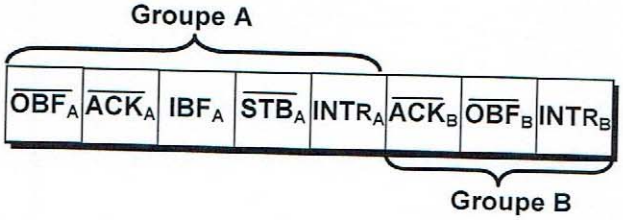
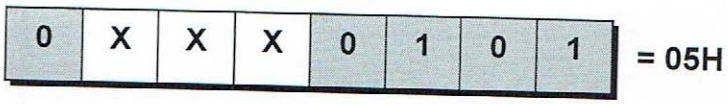
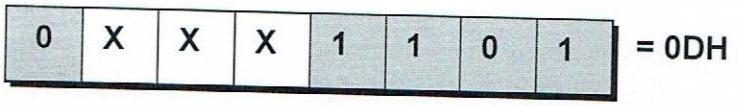


Fig. 12 Etats des signaux d'asservissement du port C



(a) Validation des interruptions à travers PC_2

(b) Validation des interruptions à travers PC_4



(c) Validation des interruptions à travers PC_6

Fig. 13 Validation des interruptions à travers le port C à travers (a), (b) et (c)

Initialiser le PPI revient donc à envoyer la séquence C4H, 05H, 09H et 0DH vers le registre de contrôle dont l'adresse est 03H de la manière suivante :

```
MOV  AL,C4H
OUT  03H, AL
MOV  AL,05H
OUT  03H, AL
MOV  AL,09H
OUT  03H,AL
MOV  AL,0D
OUT  03H, AL
```

CHAPITRE 10

CIRCUIT D'ENTREES ET DE SORTIES SERIEL ET PROGRAMMABLE

I Introduction

Rappelons qu'au niveau du type de transmission, le 8086 délivre et reçoit les données sur 8 ou 16 fils parallèles alors qu'il existe des périphériques tel qu'un téléimprimeur qui véhicule les données un bit après l'autre sur un seul fils. Il est donc nécessaire d'introduire un circuit d'Entrées/Sorties capable d'établir une telle adaptation.

II Principes de base de la communication sérielle

Il existe deux problèmes dans la communication sérielle.

1- Comme les données changent d'état d'une manière asynchrone, il faut trouver une méthode pour capturer le signal à des intervalles correspondant à des bits de données individuels.

2- Les bits (données) sériels doivent être cadrés de manière correcte pour éviter toute confusion, tel que ce bit appartient à tel ou tel autre caractère.

II.1 Communication sérielle synchrone

Une communication synchrone se dit d'un mode de transmission de données lorsque l'émetteur et le récepteur fonctionnent au rythme d'une horloge commune, réglée au début de la communication.

La communication sérielle synchrone requiert le transfert d'un bit sur chaque front actif du signal d'horloge. Ceci n'est réalisable que si, à quelques erreurs prés, les fréquences des horloges des deux extrémités sont les mêmes. D'où le fait que l'émetteur génère un signal qui doit permettre au récepteur de se synchroniser à chaque bit.

D'autre part, dans ce type de communication sérielle les caractères sont délimités par un caractère dit de synchronisation. Typiquement, on peut avoir un ou deux caractères de synchronisation précédant un bloc de données. En transmission, le transmetteur doit transférer un bit sur chaque front actif de

l'horloge de transmission, sinon un caractère de synchronisation est aussitôt inséré.

En réception, le récepteur reconnaît et ignore les caractères de synchronisation. Cependant, à l'intérieur du bloc de données, tous les caractères synchronisation sont assemblés comme le reste des données

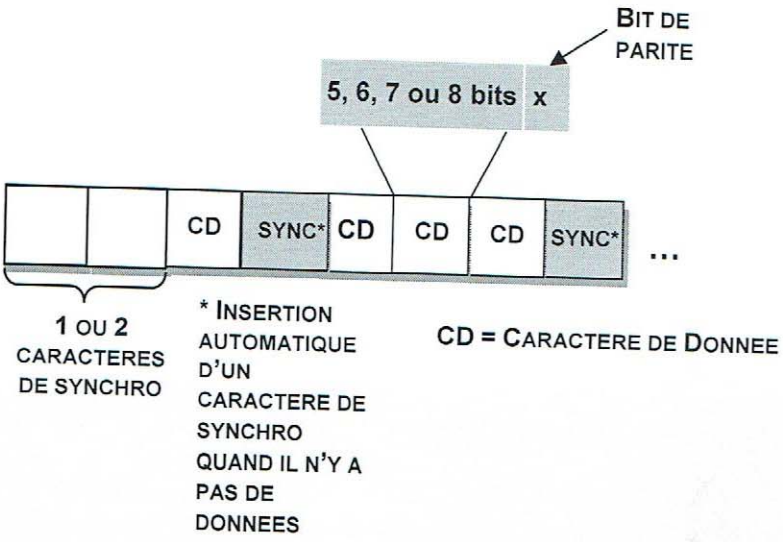


Fig. 1 Format d'une trame de données synchrone

II.2 Communication sérielle asynchrone

Une communication asynchrone se dit d'un mode de transmission de données d'une machine à une autre dans lequel chaque caractère est précédé d'un bit de départ et suivi d'un bit de fin. On l'appelle aussi transmission START/STOP.

Dans ce cas, les caractères sont transmis lorsqu'ils sont disponibles. Entre les différents caractères, la ligne de transmission est maintenue à l'état haut (MARKED). En l'absence de données, la ligne de transmission est forcée à l'état bas (BREAK) afin d'éviter l'envoi d'informations intempestives. D'autre part, chaque caractère de données est délimité par 1 bit START, 1,1 1/2 ou 2 bits STOP. Le bit start est un '0' et le bit stop est un '1', entre les deux on peut avoir 5, 6, 7 ou 8 bits de données et éventuellement 1 bit de parité. Un signal de synchronisation (bit START) est généré par l'émetteur au début seulement d'une séquence de bits de données plus au moins longue (un octet par exemple)

Au début de chaque caractère asynchrone, la logique de réception doit identifier le bit START pour assembler le caractère donnée. Cependant il y a un risque d'erreur car les horloges de transmission et de réception peuvent ne pas

être en phase. Ce problème est résolu par l'introduction d'un multiplicateur × 16, x 32 ou × 64 fois la fréquence de transfert ou de décalage (Baud Rate).

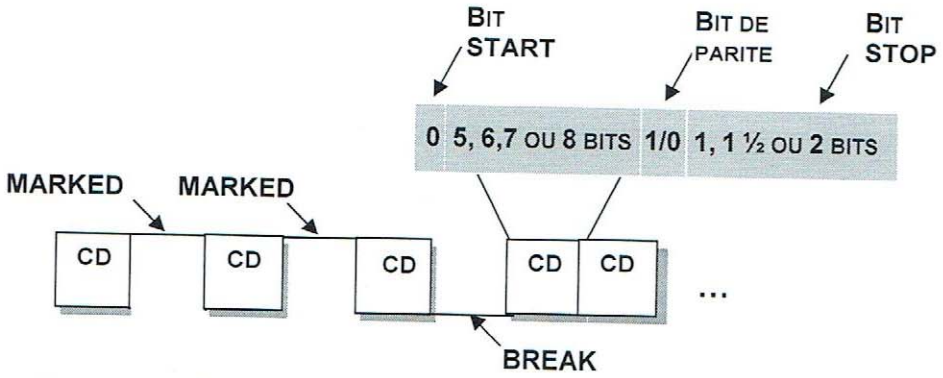


Fig. 2. Format d'une trame de données asynchrone

II.3 Bit Rate et Baud Rate

La notion de Bit Rate et Baud Rate est compliquée et entrelacée. Le Bit Rate est le nombre de bits de données transmis par seconde. Cependant, le Baud Rate est la mesure du nombre de fois par seconde qu'un signal change d'état dans un canal de communication. Il porte le nom de son inventeur Emile Baudot (Code Baudot dans la télégraphie). Le Bit Rate et Baud rate définissent donc la vitesse de connexion dans les liaisons séries. Ils sont liés par la relation suivante:

Bit Rate = Baud Rate x log2 (M)

Remarque: Pour le binaire M=2. Par conséquent, Bit Rate = Baud Rate=Nombre de bits/Seconde. Dans ce cas, 1 Baud = 1 bit/s=1 Hz.

III USART (8251)

L'USART est un circuit d'Entrées/Sorties de la famille Intel. Il permet de réaliser la liaison série entre le 8086 et ses périphériques (conversion parallèle série et série parallèle programmable).

L'USART communique avec le 8086 à travers les bus de données d'adresses et de commande. Les dialogues avec la périphérie sont assurés par un canal disposant d'une ligne de transmission et d'une ligne de réception et des signaux de contrôle et de synchronisation. La programmation de l'USART est fonction du périphérique connecté.

III.1 Caractéristiques de l'USART

L'USART a les caractéristiques suivantes:

1- Transfert sériel synchrone et asynchrone.
2- Détection des erreurs de parité, de surcharge et de format.
3- Communication Half et Full duplex.
4- Longueur d'un caractère 5, 6, 7ou 8 bits.
5- En mode asynchrone, il peut avoir 1, 1 1/2 ou 2 bits stop.
6- En mode synchrone, il peut d'atteindre un Baud Rate de 56 K Baud.
7- En mode asynchrone, il peut atteindre un Baud Rate de 9.6 K Baud.
8- En mode asynchrone, il peut travailler suivant les horloges $\times 1$, $\times$ 16 et $\times$ 64. A titre d'exemple, soient les fréquences de décalage bit à bit (Baud Rate fourni par le MODEM) $V_{TX} = V_{RX}$ = 110 Baud, alors les horloges de T_X et R_X:

$T_XC = R_XC$ = 110 Hz (x1)
$T_XC = R_XC$ = 1.76 KHz (x16)
$T_XC = R_XC$ = 7.04 KHz (x64)

La 8ème caractéristique est illustrée pour la transmission et la réception par les chronogrammes des Figures 3-4, respectivement.

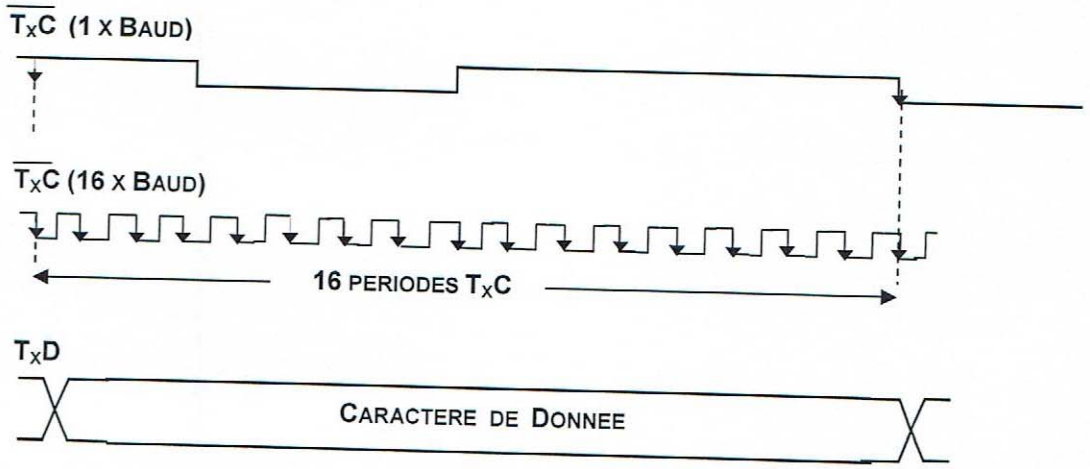


Fig. 3 Horloge et transmission et données

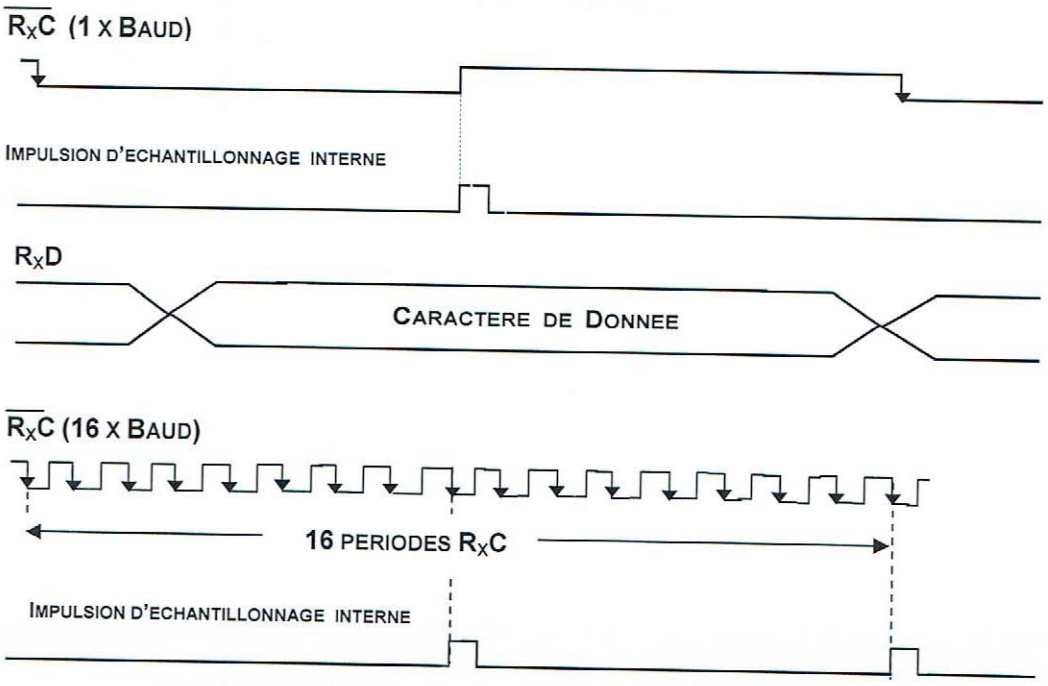


Fig. 4 Horloge et réception de données

III.2 Description externe de l'USART

L'USART se présente sous la forme d'un boîtier DIL 28 broches mono-tension (0, 5V). Nous distinguons les signaux d'interface avec le microprocesseur et les signaux d'interface avec le périphérique.

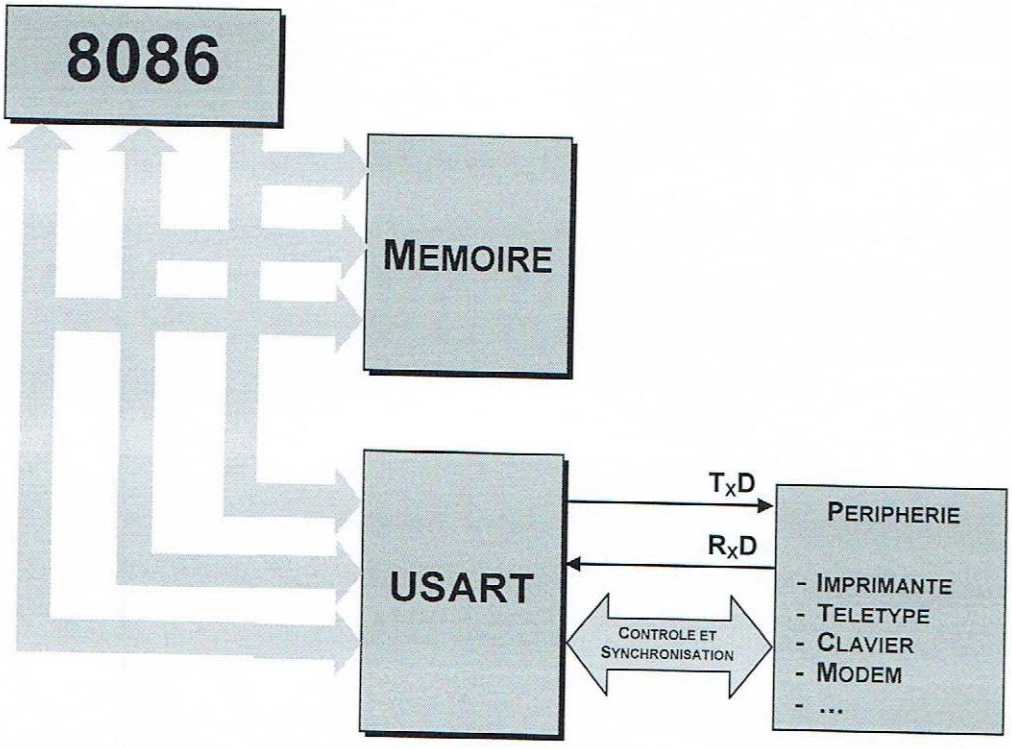


Fig. 5 L' USART dans une structure à base de 8086

III.2.1 Liaisons avec le 8086

Bus de données D_0-D_7 : Les 8 lignes bidirectionnelles et en logique 3 états, sont directement connectées au bus de données du 8086 (AD_0 - AD_7). Elles assurent l'échange des données entre le 8086 et l'USART.

Bus d'adresses $\overline{CS}$ et C/$\overline{D}$: Ces broches servent, respectivement, à la sélection d'un boîtier USART et à la sélection de son registre de contrôle ou d'état ou de son registre de données en transmission ou en réception.

$\overline{CS}$ = { 0 boîtier USART sélectionné ; 1 boîtier USART non sélectionné.

C/$\overline{D}$= { 1 Sélection du registre de contrôle ou du registre d'état ; 0 Sélection du registre de données en T_X et R_x.

Vcc ou Alimentation unique: Etant un circuit TTL, le PPI travaille avec une alimentation unique (0, 5v).

CLOCK ou Horloge système: Cette entrée est directement reliée à celle du 8086 pour synchroniser les signaux internes de l'USART. Par conséquent, pour assurer une bonne synchronisation du système, celle-ci est reliée au même circuit générateur d'horloge (8284) que le 8086. Enfin, l'horloge système vaut 30 fois $\overline{T_XC}$ et $\overline{R_XC}$ en mode synchrone et 4.5 fois $\overline{T_XC}$ et $\overline{R_XC}$ en mode asynchrone.

RESET: Sert à initialiser l'USART. Dans ce cas, il n y a ni transmission ni réception.

Bus de contrôle: La lecture/écriture des registres de l'USART s'effectue sous contrôle des signaux $\overline{WR}$ et $\overline{RD}$.

III.2.2 Liaisons avec le périphérique

Broches T_XD et R_XD (Transmit Data et Receive Data): Ces lignes servent à la transmission et à la réception bit à bit, respectivement. Elles changent d'état sur le front descendant et sur le front montant des horloges $\overline{T\times C}$ et $\overline{R\times C}$, respectivement.

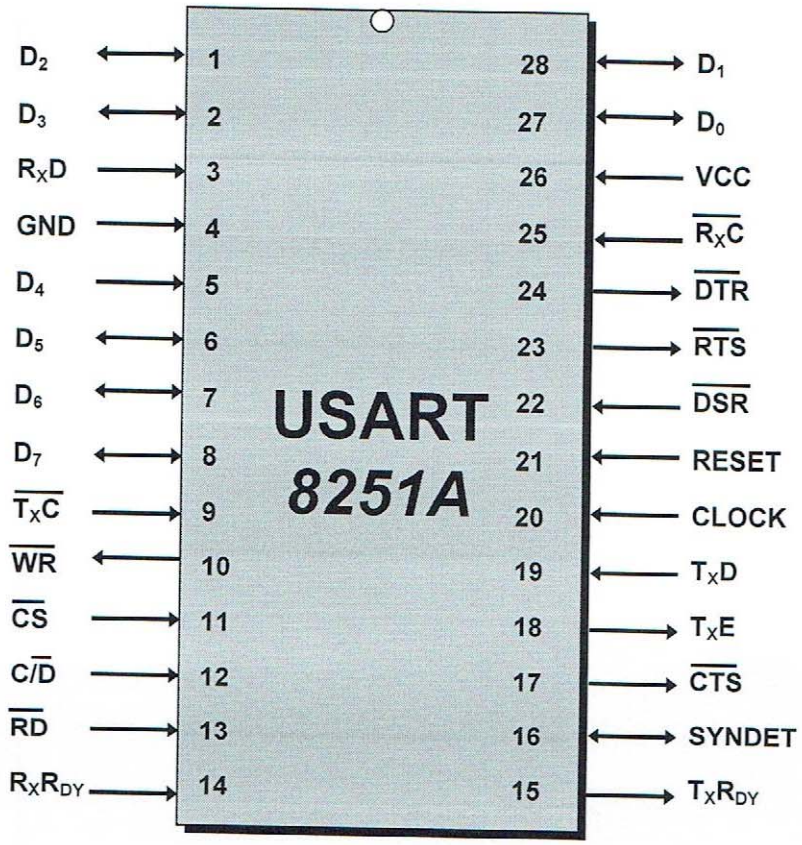


Fig. 6 Brochage de l'USART

Tableau 1 Adressage des registres de l'USART

$\overline{RD}$	$\overline{WR}$	C/$\overline{D}$	$\overline{CS}$	Registre sélectionné	Fonction
0	1	0	0	Registre de données	Réception de données
1	0	0	0	Registre de données	Transmission de données
0	1	1	0	Registre d'état	Lecture du registre d'état
1	0	1	0	Registre de contrôle	Ecriture registre de contrôle
X	X	X	1		Haute Impédance

Broches $\overline{T_XC}$ et $\overline{R_XC}$ (Transmit Clock et Receive Clock): En mode synchrone les fréquences $\overline{T\times C}$ et $\overline{R\times C}$ sont les mêmes que les fréquences de décalage bit à bit (Baud Rate fourni par le MODEM) en transmission et en réception. En mode asynchrone, la fréquence de décalage bit à bit est une fois, 16 fois ou 64 fois moindre que les fréquences de $\overline{T\times C}$ et $\overline{R\times C}$.

Remarques:

1- Les multiplicateurs x 1, x 16 et x 64 doivent être les mêmes pour la transmission et la réception. Notons que le mode x 1 est appelé isosynchrone.
2- $\overline{T\times C}$ et $\overline{R\times C}$ peuvent être reliées à un CTC (8253) de façon à pouvoir programmer les fréquences de décalage bit à bit.

Broche de synchronisation SYNDET (Synchronous Detect): Cette broche n'est utilisée que dans le mode synchrone. Elle peut être soit en sortie soit en entrée. Ceci dépend de la programmation de l'USART en synchronisation interne ou externe, respectivement.

En mode de synchronisation externe, cette ligne est une entrée. Quand la synchronisation externe est terminée, SYNDET doit être forcée au niveau haut après que le dernier bit du dernier caractère de synchronisation ait été reçu. Autrement dit, quand la séquence de synchronisation est détectée, la logique externe attend alors une période d'horloge $\overline{R_XC}$ avant d'activer SYNDET. Une fois celle-ci activée, elle doit être maintenue à l'état haut pendant au moins 1 période R×C pour accomplir une bonne synchronisation. C'est à partir de ce moment que l'assemblage des caractères doit commencer.

En mode de synchronisation interne (Monosynchronous ou Bisynchronous) SYNDET est une sortie. Elle est activée lorsque le dernier bit du 2ème caractère de synchronisation est reconnu. Elle est donc activée chaque fois qu'un caractère de synchronisation est reconnu. Elle est remise à zéro par la lecture du registre d'état par le microprocesseur.

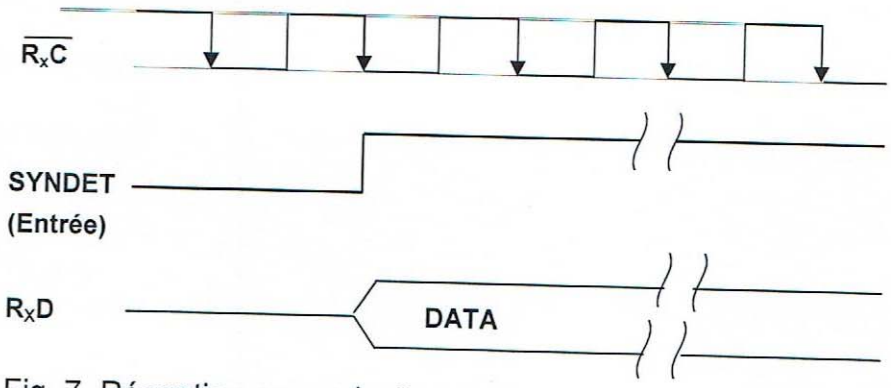


Fig. 7 Réception en mode de synchronisation externe

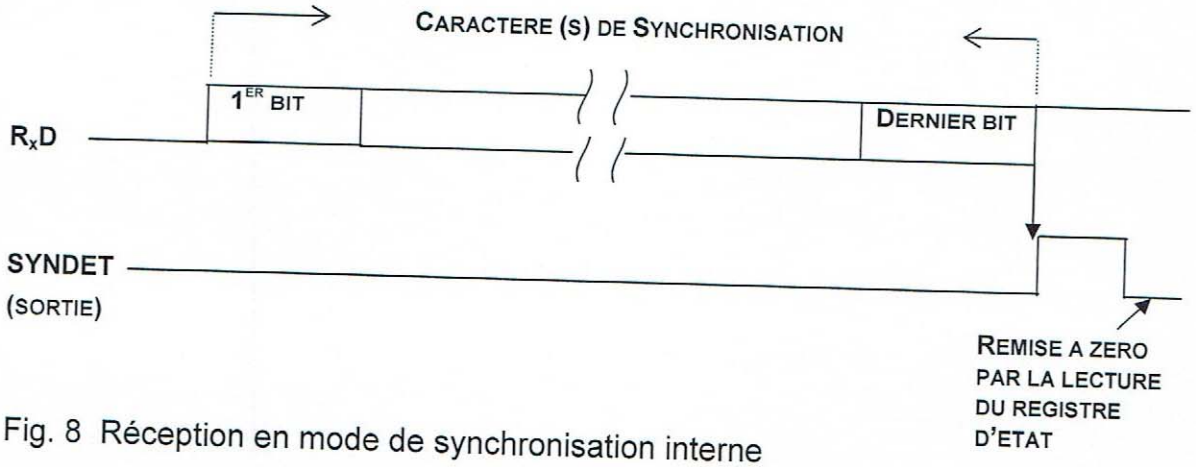


Fig. 8 Réception en mode de synchronisation interne

Broche R_X R_{DY} (Receiver Ready): Active, cette broche indique qu'une donnée a été transférée de la logique de réception vers le registre de réception et par conséquent, elle peut être lue. Cette ligne demeure à l'état haut jusqu'à ce que le processeur finisse de lire la donnée. Elle peut être inhibée par la mise à zéro de la bascule R×E (bit 2) du registre de commande. Cependant, la bascule $R\times R_{DY}$ (bit 1) du registre d'état est toujours mise à 1 lorsque le registre de réception est plein et ce quel que soit l'état de la bascule R×E.

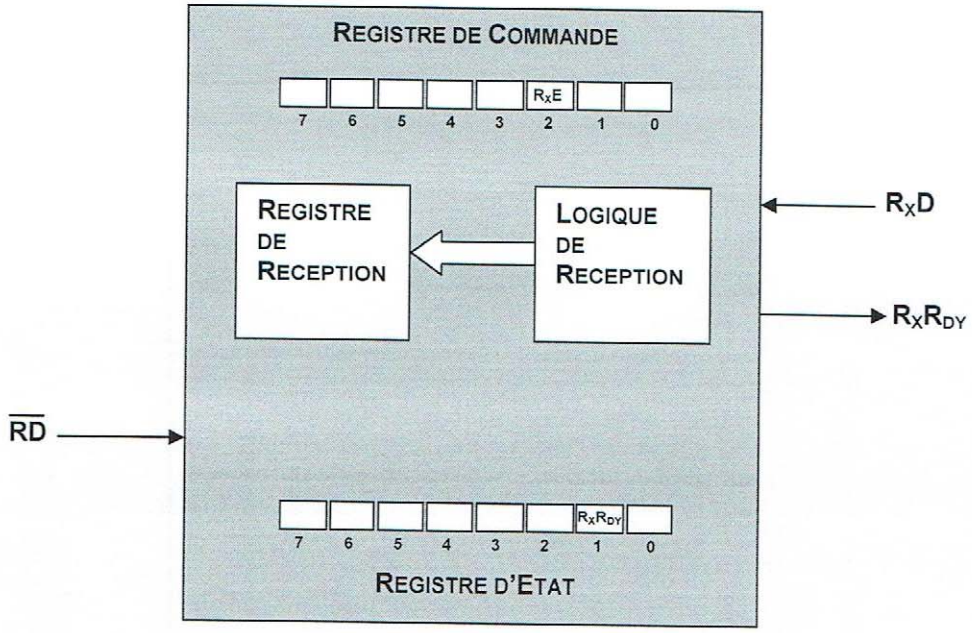


Fig. 9 Réception asynchrone

Remarques:

1- Si le processeur ne lit pas le registre de réception avant que le prochain caractère ne soit transféré de la logique de réception alors une erreur de surcharge est aussitôt détectée par l'USART.
2- R_XR_{DY} peut être utilisée pour interrompre le 8086.

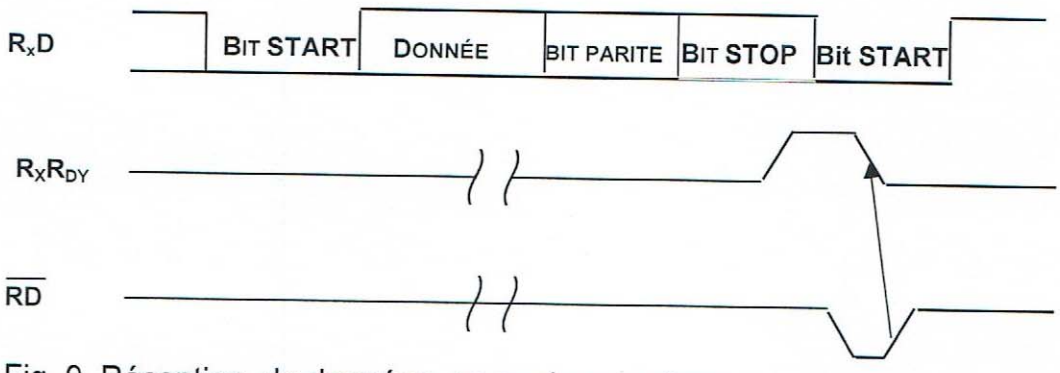


Fig. 9 Réception de données en mode asynchrone

Broche T_XR_{DY} (Transmitter Ready: Active, cette broche indique que le contenu du registre de transmission a été transféré dans la logique de transmission. Par conséquent, le registre de transmission peut accepter le prochain caractère. T_XR_{DY} passe à l'état bas quand l'information est inscrite dans le registre de transmission. $\overline{T_XR_{DY}}$ peut être inhibée par la mise à '0' de T_XEN (bit 0) du registre de commande et/ou la mise à '1'de la broche $\overline{CTS}$. Cependant, la bascule T_XR_{DY} (bit 0) du registre d'état est toujours mis à '1' lorsque le registre de transmission est vide et ce quel que soit l'état de T_XEN et $\overline{CTS}$.

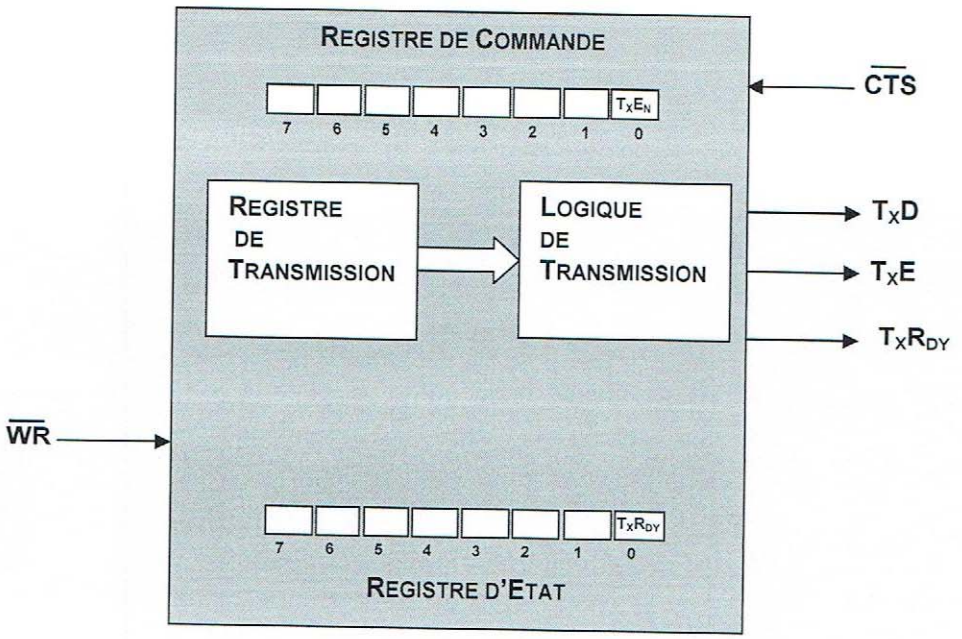


Fig. 10 Transmission asynchrone

Broche T_XE (Transmitter Empty): Active, cette broche indique que la logique de transmission a transféré la donnée et par conséquent, elle est vide. Elle demeure à l'état haut jusqu'à ce qu'une autre donnée soit transférée du registre de transmission vers la logique de transmission.

Remarques:

1- Le chargement du registre de contrôle remet T_XR_{DY} à zéro.
2- T_XR_{DY} peut être utilisée pour interrompre le 8086.
3- En mode synchrone, la broche T_XEN est mise à '1' afin d'indiquer que l'USART est vide et qu'il est uniquement entrain de transmettre des caractères de synchronisation.

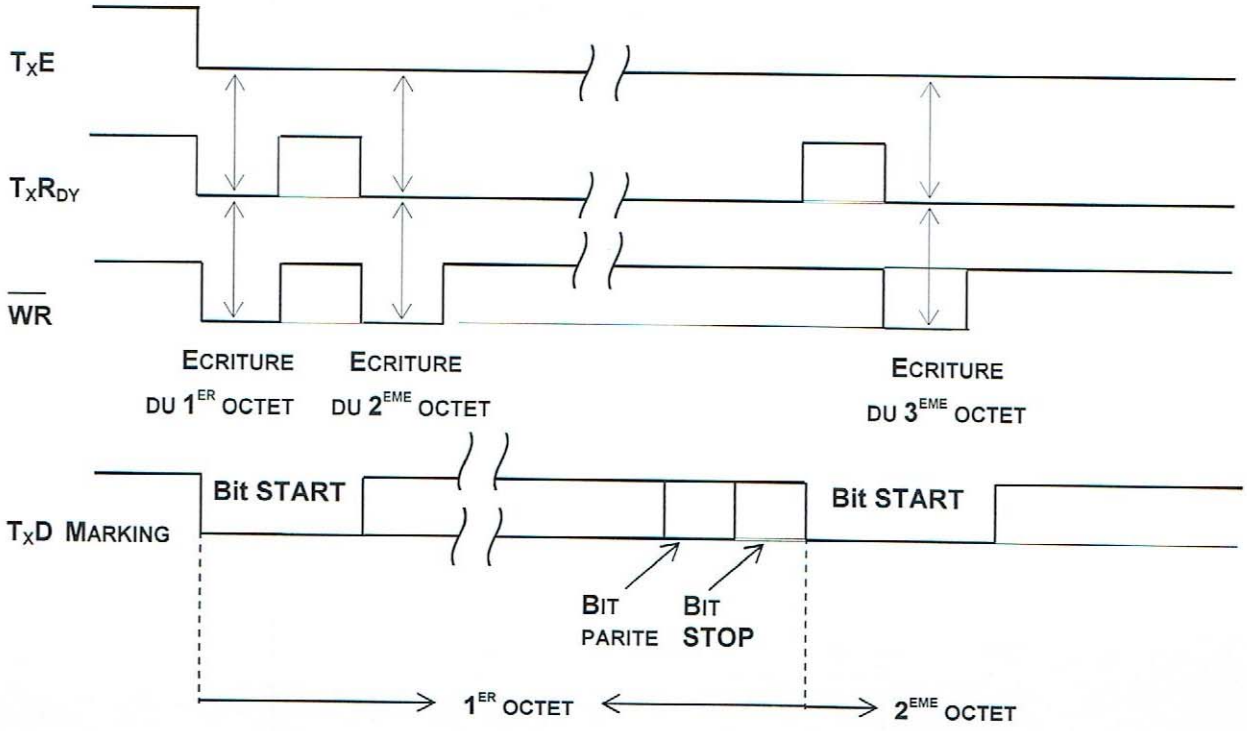
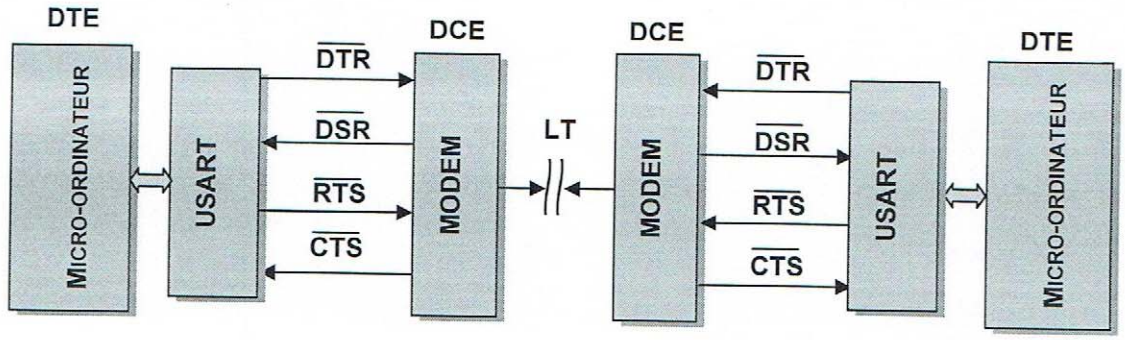


Fig. 11 Transmission de données en mode asynchrone

Broches d'interface avec le MODEM

Ces broches d'interface avec le MODEM sont au nombre de quatre. Comme il est montré en Figure 12, ils permettent de connecter un micro-ordinateur PC (Data Terminal Equipment, DTE) à un MODEM (Data Communication Equipment, DCE) à travers un USART. Pour pouvoir faire communiquer deux micro-ordinateurs PC, il faut utiliser un MODEM et un USART, de part et d'autre d'une ligne de transmission.



LT: LIGNE DE TRANSMISSION
DTE: DATA TERMINAL EQUIPMENT
DCE: DATA COMMUNICATION EQUIPMENT

Fig. 12 Connexion de deux micro-ordinateurs PC

$\overline{DTR}$ (Data Terminal Ready): Connecté habituellement à un MODEM, ce signal indique que l'USART est prêt.

$\overline{DSR}$ (Data Set Ready): Généralement, ce signal est utilisé pour répondre à un $\overline{DTR}$ par le MODEM pour signifier qu'il est prêt.

$\overline{RTS}$ (Request To Send): Demande d'émission. Actif, ce signal initie une transmission de données en demandant au MODEM de se préparer à transmettre.

$\overline{CTS}$ (Clear To Send): Inhibition du transmetteur par le MODEM. Ce signal est utilisé pour répondre à un $\overline{RTS}$ par le MODEM. Actif, Ce signal indique que la transmission peut commencer. Non connecté $\overline{CTS}$=0.

III.3 Description interne de l'USART

L'USART, vu du 8086 se présente sous la forme des quatre registres adressables suivants:

III.3.1 Registre de données en transmission

Le registre de transmission de longueur 8 bits est à écriture seule. Il sert uniquement lors d'une transmission.

III.3.2 Registre de données en réception

Le registre de réception de longueur 8 bits est à lecture seule. Il sert uniquement lors d'une réception.

III.3.3 Registre de contrôle

Le registre de contrôle dépend de la sélection du mode synchrone ou asynchrone. Il joue ainsi le rôle des registres suivants:

Registre de contrôle en mode synchrone ou asynchrone: La première configuration de ce registre constitue la sélection du mode synchrone ou asynchrone.

Registres identifiant le(s) caractère(s) de synchronisation: La deuxième configuration de ce registre doit suivre impérativement la première configuration si cette dernière a choisi le mode synchrone comme mode de fonctionnement de l'USART. Selon le cas, un ou deux caractère(s) de synchronisation peut (peuvent) être envoyé(s) et ce en fonction du mode mono-synchrone ou bi-synchrone.

Registre de commande pour le mode synchrone ou asynchrone: La configuration de ce registre est valable aussi bien pour le mode synchrone que

pour le mode asynchrone. Pour la première fois et dans les deux modes, le registre de commande doit être immédiatement envoyé après le ou les 2 caractères de synchronisation pour le mode synchrone et immédiatement après l'envoi du registre de contrôle pour le mode asynchrone. Ensuite, ce registre de commande peut être modifié et envoyé à n'importe quel endroit du programme

Registre d'état: Le registre d'état sert à connaitre l'état de l'USART, notamment celui des erreurs de parité, de surcharge et de format.

Remarques:

1- Les registres de données en transmission et en réception sont adressés de manière unique.
2- Les registres de contrôle, (celui) ceux identifiant(s) le ou les caractères de synchronisation, le registre de commande et le registre d'état sont adressés de manière unique.

Registre de contrôle = Mode synchrone

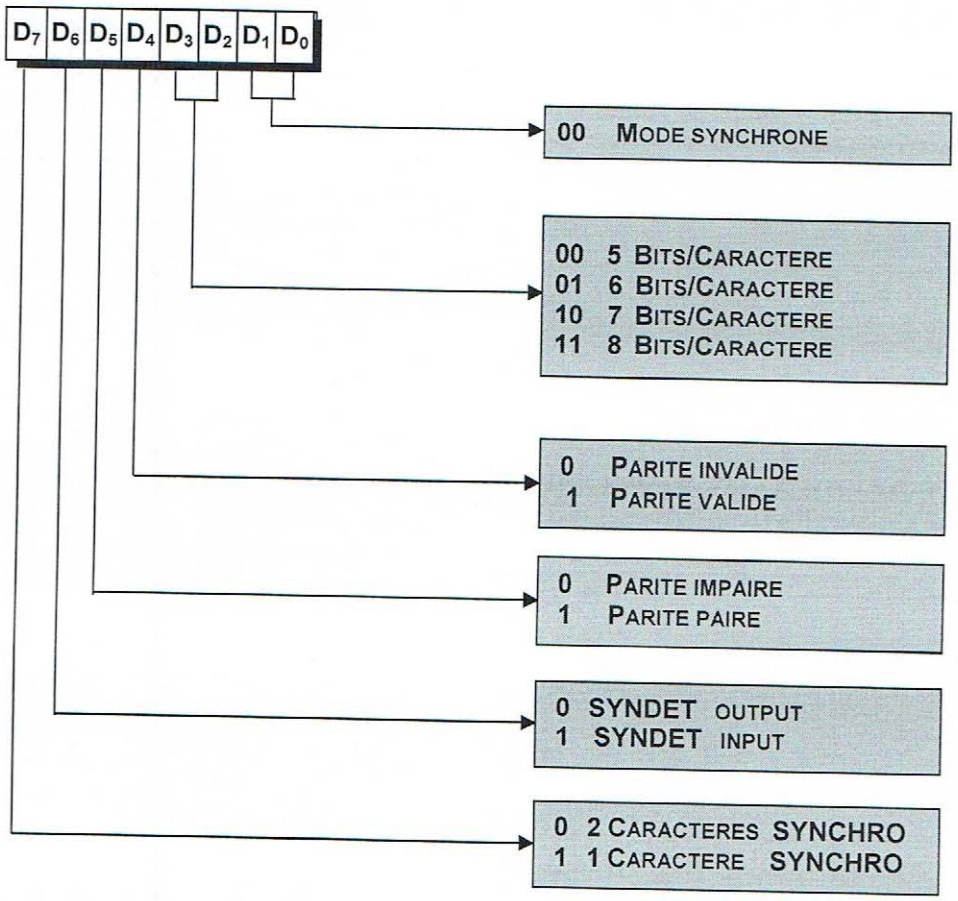


Fig. 13 Configuration du registre de contrôle en mode synchrone

Remarque: Si le format 5, 6 ou 7 bits est choisi, le restant des bits pour atteindre 8 bits doit être mis à '0' avant toute transmission en mode synchrone ou asynchrone.

Registre de contrôle = Mode asynchrone

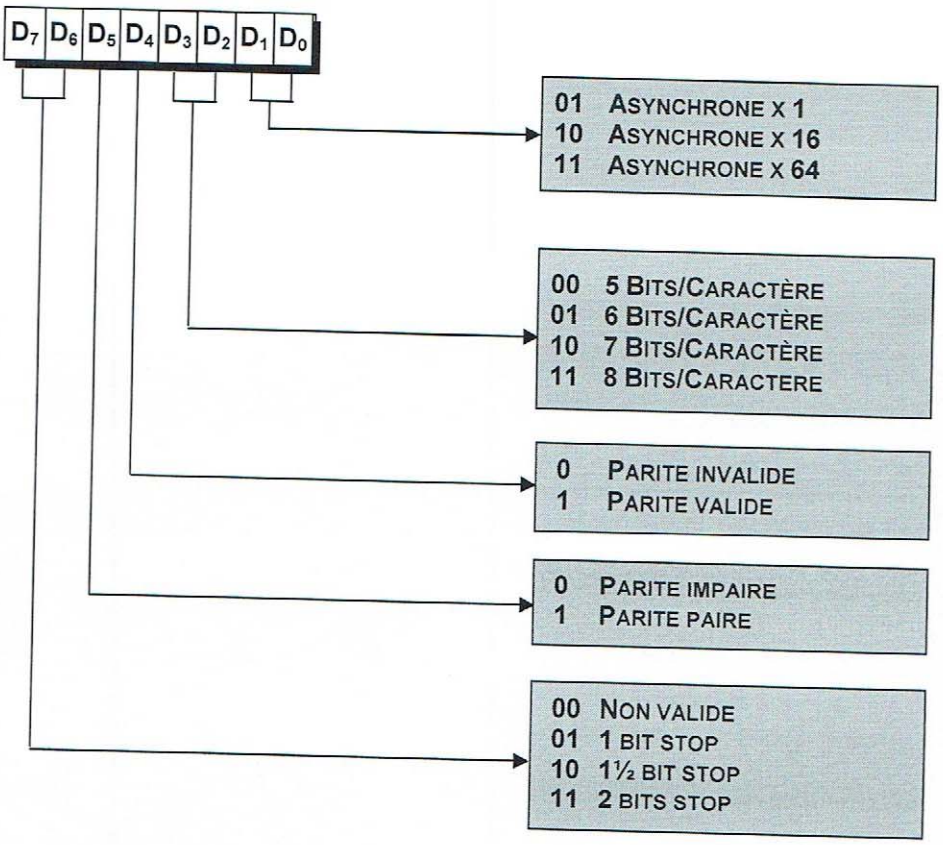
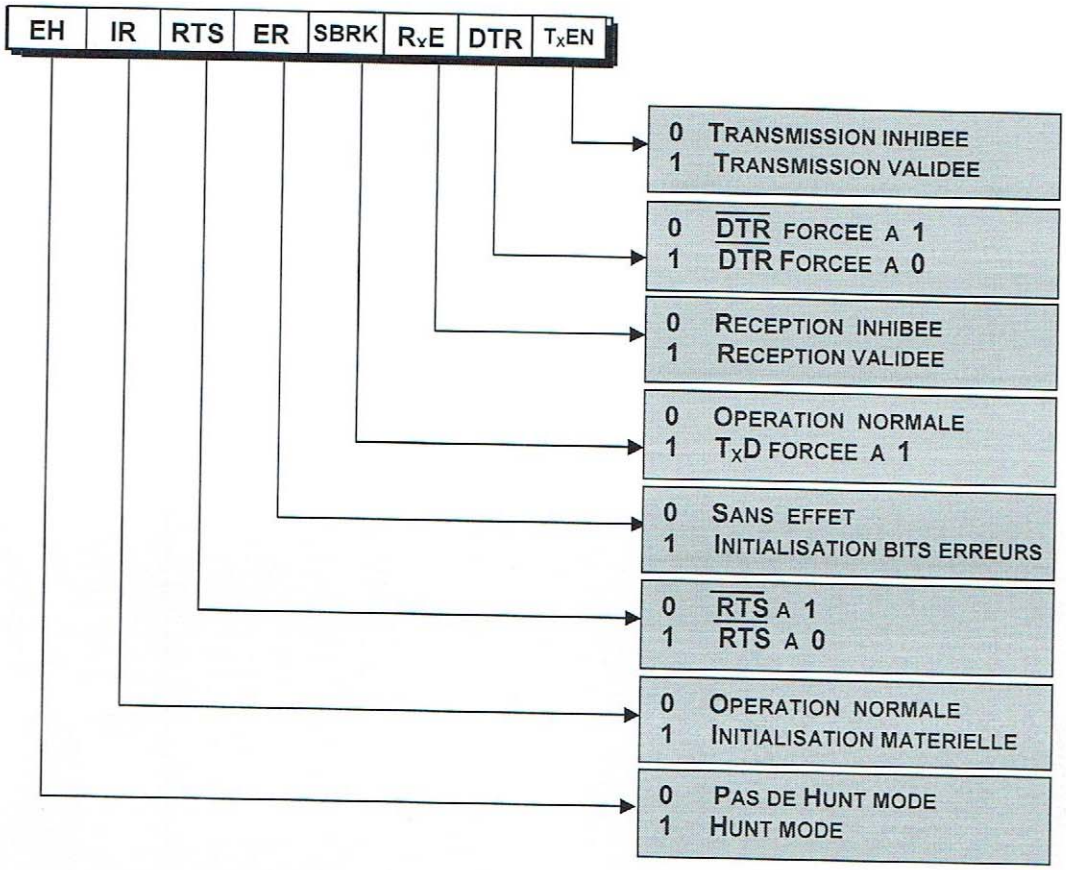


Fig. 14 Configuration du registre de contrôle en mode asynchrone

Caractère(s) de synchronisation

Le ou les caractères de synchronisation sont considérés comme des registres de contrôle. Si le mode synchrone est choisi, le registre de contrôle doit être immédiatement suivi par le ou les caractères de synchronisation (1 ou 2 octets).

Registre de contrôle = Registre de commande



EH: ENTER HUNT MODE
IR: INTERNAL RESET
ER: ERROR RESET
SBRK: SEND A BREAK
R_XE: RECEIVER EMPTY
T_XEN: TRANMITTER ENABLE

Fig. 15 Configuration du registre de commande

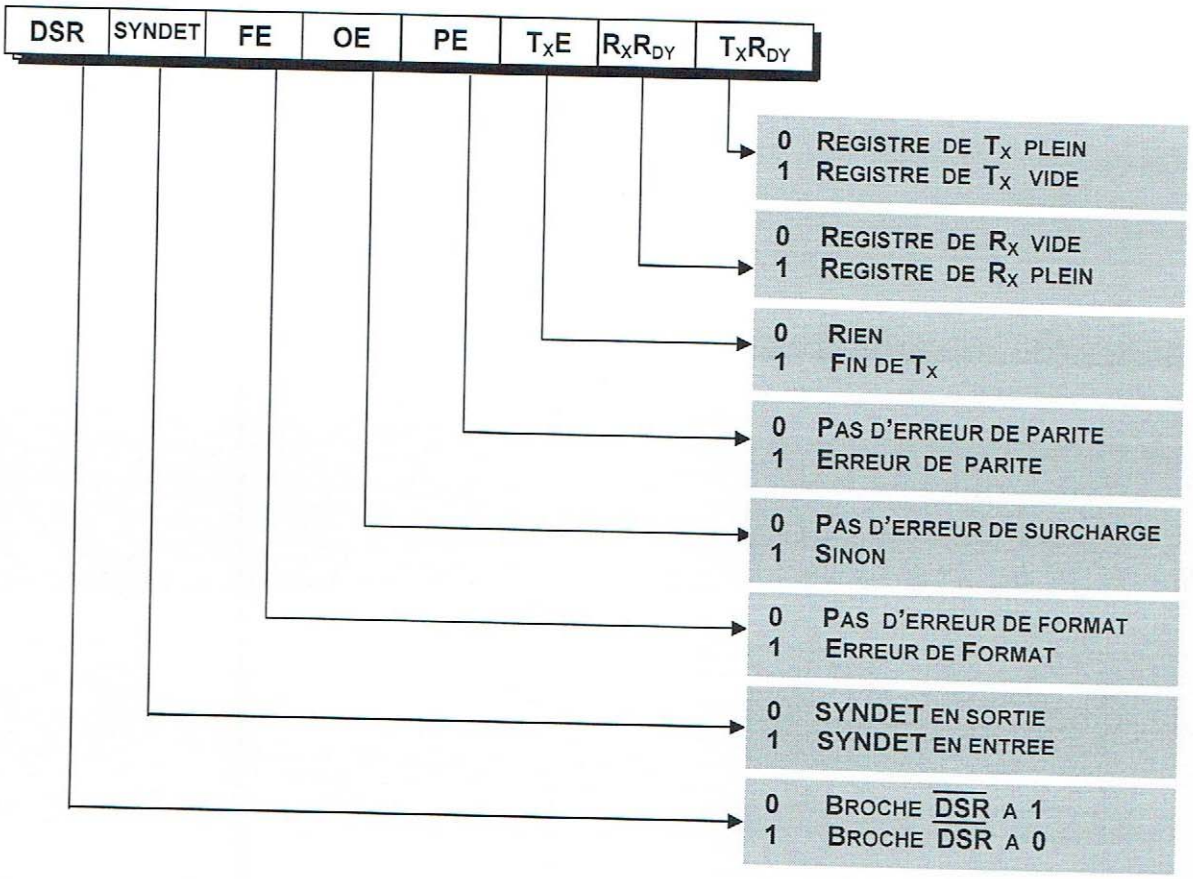
Remarque: Le mode de chasse (Hunt Mode) du ou des caractères de synchronisation est mise à '1' quand il y a une perte de synchronisation ou au moment de la réception des données.

III.3.4 Registre d'état

Le registre d'état sert informer le programmeur de l'état de fonctionnement de l'USART. Ainsi, il peut en mode sans interruption, à travers les broche T_XR_{DY} et R_XR_{DY}, nous renseigner sur l'état (vide ou plein) des registres de données en

transmission et en réception. Aussi, il peut nous renseigner sur l'état des bascules d'erreurs en réception (erreur de format, erreur de parité et erreur de surcharge).

Registre de contrôle = Registre d'état



FE: FORMAT ERROR
OE: OVER-RUN ERROR
PE: PARITY ERROR
T_XE: TRANSMITTER EMPTY
R_XR_{DY}: RECEIVER READY
T_XR_{DY}: TRANMITTER READY

Fig. 16 Configuration du registre d'état

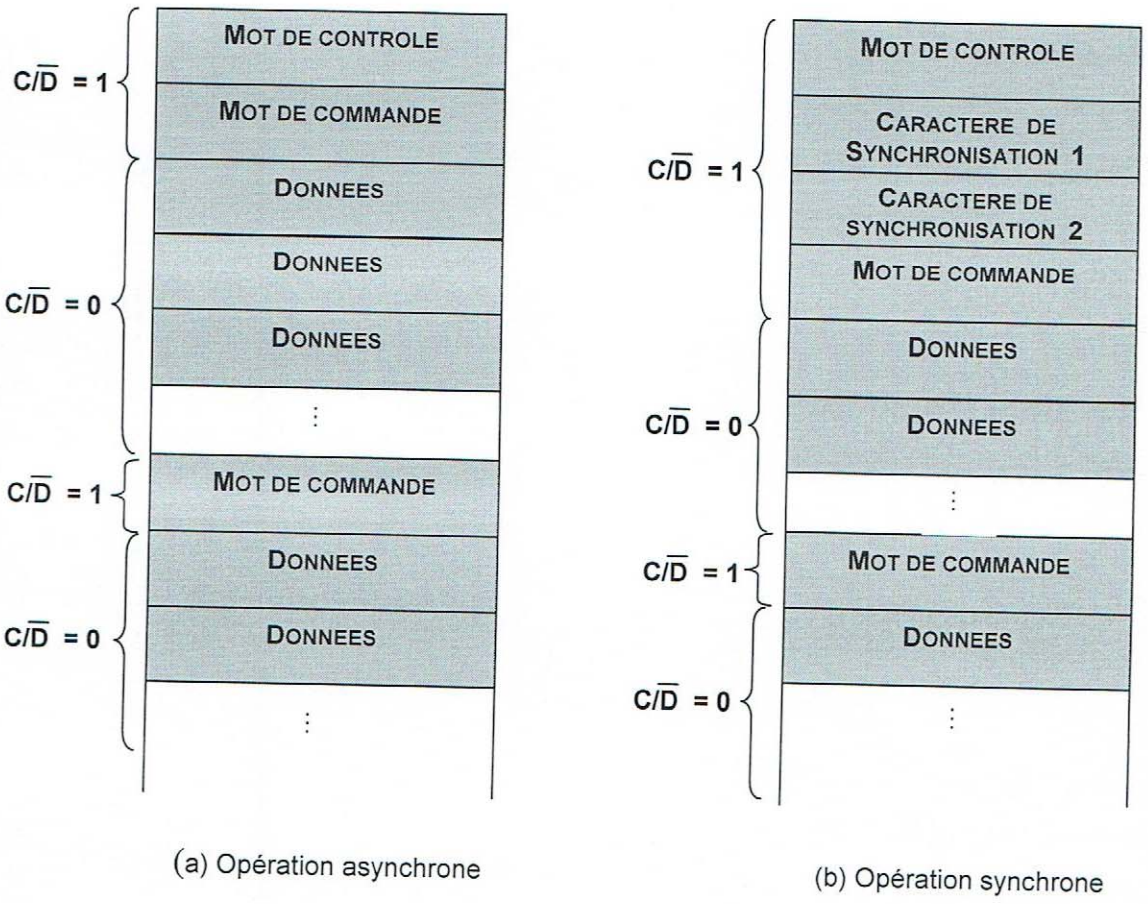
Remarques:

1- Les registres de contrôle et d'état sont adressables de manière unique. Sauf que le premier utilise une instruction de type 'OUT' et le second une instruction de type 'IN'.

2- L'état du bit T_XEN= 0 (bit 0) du registre de commande et/ou la broche CTS valide (ent) ou invalide (ent) les interruptions dues à la broche T_X R_{DY}.
3- L'état du bit T_XEN du registre de commande et/ou de la broche $\overline{CTS}$ n'affecte en rien celui du bit T_X R_{DY} du registre d'état.
4- L'état du bit R_XE (bit 2) du registre de commande valide ou invalide les interruptions dues à la broche R_X R_{DY}.
5- L'état de R_XE n'affecte en rien celui du bit R_X R_{DY} (bit 1) du registre d'état.

IV Programmation de l'USART

A l'initialisation, l'USART peut prendre 2, 3 ou 4 octets à travers le registre de contrôle.



(a) Opération asynchrone

(b) Opération synchrone

Fig. 17 Les deux modes de fonctionnement de l'USART

IV.1 Exemple de programmation de l'USART

IV.1.1 Premier exemple

Il s'agit de programmer l'USART en interface asynchrone. Les adresses des registres sont 00H pour les registres de données et 01H pour le registre de contrôle. L'USART travaille donc en asynchrone, sur un format de 7 bits/Caractère avec parité paire et 1 bit stop. La fréquence de décalage bit à bit se fait en transmission et en réception à une horloge 16 fois moindre que celle des horloges de transmission et de réception (T_X et R_X). D'autre part, aussi bien la transmission que la réception se font en interrompant le microprocesseur.

Registre de contrôle

0	1	1	1	1	0	1	0

= 7AH

Registre de commande

X	0	1	1	0	1	1	1

= 37H

L'initialisation de l'USART consiste donc à écrire les données 7AH et 37H, dans cet ordre, dans le registre de contrôle de celui-ci de la manière suivante:

```
MOV AL, 7AH
OUT 01H, AL
MOV AL, 37H
OUT 01H, AL
```

IV.1.2 Deuxième exemple

Il s'agit de programmer l'USART en interface synchrone. Les adresses des registres sont 00H pour les registres de données et 01H pour le registre de contrôle. L'USART fonctionne suivant le protocole BYSINC d'IBM, sans parité sur un format de 8 bits/caractère. La Transmission et la Réception se font en interruption. L'USART travaille en synchronisation externe. Le 1^{er} caractère de synchronisation est égal à 4BH et le $2^{ème}$ caractère de synchronisation est égal à 6AH.

Registre de contrôle

0	1	X	0	1	1	0	0

= 4CH

1er Caractère de SYNCHRO

0	1	0	0	1	0	1	1

= 4BH

2ème Caractère de SYNCHRO

0	1	1	0	1	0	1	0

= 6AH

Registre de commande

0	0	1	1	0	1	1	1

= 3BH

L'initialisation de l'USART consiste donc à écrire les données 4CH, 4BH, 6AH et B7H, dans cet ordre, dans le registre de contrôle de celui-ci de la manière suivante :

```
MOV AL, 4CH
OUT 01H, AL
MOV AL, 4BH
OUT 01H, AL
MOV AL, 6AHH
OUT 01H, AL
MOV AL, 3BH
OUT 01H, AL
```

CHAPITRE 11

INTERFACES SUR PC

I Introduction

Lorsqu'un grand nombre de données doivent circuler en peu de temps entre deux ordinateurs ou entre un ordinateur et un périphérique, il est vital de contrôler le flux d'information. Ce contrôle, est appelé "dialogue" ou "handshaking". Sans dialogue entre PC et PC-périphériques, le nombre de caractères se perdent car la mémoire d'un ordinateur est loin d'être infinie.

Les deux types d'interfaces de communication qui existent sur les PC sont parallèle et série. La liaison parallèle est unidirectionnelle, c'est une sortie uniquement (elle est bidirectionnelle dans la nouvelle génération de PC). Elle est très rapide mais limitée aux faibles distances. La liaison série est, plus lente, et bidirectionnelle et supporte des distances importantes, jusqu'à plusieurs Km sans amplification.

Cette dernière décennie a connu une montée en puissance de l'interface série USB. A l'heure actuelle, tous les PC en sont dotés en nombre important, provocant ainsi la disparition de l'interface parallèle. Bien que toujours présente sur les PC, la RS-232 continue à être utilisée dans l'industrie pour connecter différents appareils électroniques tel que les automates, appareils de mesure, etc. En cas d'absence de port RS-232, il existe des adaptateurs USB/port série. Le code d'échange entre PC est le code ASCII (American Standard Code for Information Interchange).

II Interface parallèle Centronics

Les interfaces parallèles des PC répondent au standard édicté par le constructeur d'imprimantes Centronics. Elle possède 8 bits de données qui sont transmis simultanément en plus de divers signaux de dialogues circulant dans les deux sens. La vitesse peut atteindre environ 200.000 octets/s si le périphérique récepteur le veut bien.

II.1 Fonctionnement de l'interface parallèle Centronics

Le fonctionnement de l'interface parallèle Centronics est géré par 3 signaux de dialogues $\overline{\text{STROBE}}$, $\overline{\text{ACK}}$ et BUSY. Leur chronogramme de fonctionnement est représenté en Figure 1.

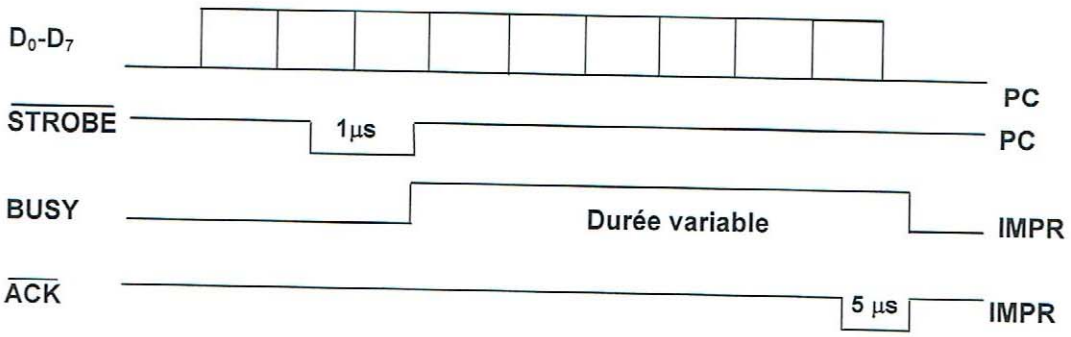


Fig. 1 Chronogramme de dialogue d'un interface parallèle Centronics

II.2 Brochage de l'interface parallèle Centronics

Les interfaces parallèles sont disponibles sur la face arrière du PC sous forme d'un connecteur à 25 broches (DB-25 femelles). Les PC-XT et AT en possèdent deux, reconnus par le MS-DOS comme étant LPT_1 et LPT_2 (Line PrinTer).

Tableau 1 Brochage de l'interface parallèle Centronics (DB-25)

Broche	Signal généré par	Nom
1	***PC***	$\overline{\text{STROBE}}$: Octet envoyé
2	***PC***	D_0
3	PC	D_1
4	PC	D_2
5	PC	D_3
6	PC	D_4
7	PC	D_5
8	PC	D_6
9	PC	D_7
10	IMPR	$\overline{\text{ACK}}$: Octet reçu
11	IMPR	BUSY: Occupée
12	IMPR	PE: Paper End
13	IMPR	SELECT: Imprimante sélectionnée
14	IMPR	$\overline{\text{Autofeed}}$: line feed automatique
15	IMPR	Error: Imprimante en erreur
16	PC	$\overline{\text{RESET}}$: Remise à zéro
17	PC	SelectIN: Sélection
18 à 25	PC	GND: Masse

Remarques:

1- $\overline{\text{Autofeed}}$: avance papier à la réception d'un RETURN (ODH)
2- SelectIN : à la masse pour que l'imprimante imprime.
3- Rappelons que dans les PC, les adresses utilisées par les ports LPT_1 et LPT_2 sont, respectivement, 378H-37FH pour les PC-XT et PC-AT et 278H-27FH pour les PC-AT, uniquement.
4- La gestion de l'imprimante dans les PC-AT-PCI est effectuée par scrutation (polling). Il s'agit d'une boucle de programmation qui teste en permanence l'état de Busy et agit en conséquence.

III Interface série RS-232 (1969)

Au delà de quelques mètres, la transmission parallèle et ses trop nombreux fils deviennent inutilisables. L'interface série entre en jeu par la transmission de données entre PC et ordinateurs. Les informations transmises passent par le même fils, successivement. L'interface série implique un nombre de fils beaucoup moins important, avec un minimum de trois. Comme la vitesse de transmission est un paramètre important pour tout support de communication, la spécification RS-232 exige des débits (vitesses) pratiques allant de 75 bps à 115200 bps.

Tableau 2 Gamme de vitesses de transmission les plus usuelles

Débit (bps)	Longueur (mètres)	Utilisation
50	-	T_X Radio 'HAM'
110	-	Télétype (10 caractères/s)
300	-	T_X ligne téléphoniques
1200	-	MODEM téléphonique sur PC
2400	≤ 3000	Liaisons entre PC
4800	≤ 1000	Imprimante
9600	≤ 500	PC utilisés en terminaux
19200	≤ 50	Liaisons entre PC

Remarque: Pour les applications exigeant de plus hauts débits, il est possible d'utiliser les spécifications RS-422/485. Ces dernières peuvent atteindre des débits de 2 Mbps (Bus OTN, Oracle Technology Network, de largeur 96 bits).

III.1 Définition de la norme RS-232

Nous avons étudié le fonctionnement de l'USART lequel convertit les données parallèles en données séries et réciproquement. Toute fois, l'USART comme tous les circuits LSI (Large Scale Integration) fonctionne en logique TTL, c'est-à-dire une tension entre 0 et 0,8 Volts représente le '0' en logique TTL et une tension entre 2,4 et 5 Volts représente le '1' en logique TTL. Ce type de signal ne se prête pas à la transmission longue distance. En effet, après quelques

centaines de mètres de fils, les 5 Volts de départ ne sont plus que 4 ou 3 voire 2 Volts et dans ce cas, le niveau logique '1' devient le niveau logique '0'. C'est pour éviter ce genre de problème qu'est né le standard RS-232.

En RS-232, les niveaux logiques sont représentés en codage NRZ (Non Return to Zero) entre –25 Volts et –3 Volts pour le niveau logique '1' (MARK) et entre +3 Volts et +25 Volts pour le niveau logique '0' (SPACE). Avec cette convention, Le niveau 25 Volts affaibli sera lu correctement i.e., SPACE, même s'il l'est fortement. Cependant, s'il est trop, le niveau correct ne sera pas présent, mais son inverse ne le sera pas non plus. Ce qui est préférable à une donnée fausse qui n'est remarquée par personne. Ordinairement, des niveaux de +12V et -12V sont utilisés.

Remarques:

1- La conversion TTL en RS-232 utilise le circuit intégré MC 1488 i.e., +5 Volts → -12 Volts et 0 Volt → +12 Volts
2- La conversion RS-232 en TTL utilise, quant à elle, un circuit intégré MC 1489 i.e., -12 Volts → +5 Volts et +12 Volts → 0 Volt
3- La boucle de courant (20mA) est tombée en désuétude. Elle n'est plus utilisée de nos jours, car elle ne fonctionne qu'aux basses vitesses.

III.2 Brochage de la RS-232

Les PC-XT et AT supportent en standard quatre interfaces RS-232 reconnus par le MS-DOS comme étant COM_1-COM_4 (COMmunication). Ils sont disponibles sur la face arrière du PC sous forme d'un connecteur 9 broches (DB-9 Mâle). Seule la version DB-25 est vraiment standardisée, la DB-9 est une adaptation d'IBM adoptée lors de la création du PC-AT.

Tableau 3 Brochage de la RS-232 (DB-9)

Nom du signal		DB-9	DTE (PC)	DCE (MODEM)
$\overline{DCD}$	Data Carrier Detect	1	Entrée	Sortie
Rx	Receive data	2	Entrée	Sortie
Tx	Transmit data	3	Sortie	Entrée
GND	Ground	4	Sortie	Entrée
$\overline{DTR}$	Data Terminal Ready	5	Châssis	Châssis
$\overline{DSR}$	Data Set Ready	6	Entrée	Sortie
$\overline{RTS}$	Request To Send	7	Sortie	Entrée
$\overline{CTS}$	Clear To Send	8	Entrée	Sortie
$\overline{RI}$	Phone RInger	9	Entrée	Sortie

Tableau 4 Correspondance DB-25 et DB-9 pour le PC-AT

Nom du signal	DB-25	DB-9
$\overline{DCD}$ (UART)	8	1
R_X	3	2
T_X	2	3
$\overline{DTR}$	20	4
GND	7	6
$\overline{DSR}$	6	6
$\overline{RTS}$	4	7
$\overline{RI}$ (UART)	22	9

III.3 Liaison NULL-MODEM (MOdulation-DEModulation)

Dans le cas où une liaison sérielle ne nécessite pas de MODEM (quelques centaines de mètres, il est possible d'utiliser tous les signaux de dialogue et on parlera aussi d'un câble dit NULL-MODEM (car il ne nécessite pas de MODEM). Il existe deux types de liaisons NULL-MODEM.

1- Câble NULL-MODEM complet.
2- Câble NULL-MODEM simplifié.

III.3.1 Câble NULL-MODEM complet

Il existe deux types de configuration de câble NULL-MODEM complet. La première est donnée par la Figure 2 et la deuxième par la Figure 3.

Fig. 2 Premier type de liaison NULL-MODEM complet

Remarques:

1- Le signal RI sert à lancer un appel téléphonique à partir du PC.
2- Les signaux RI et $\overline{DCD}$ existent dans l'UART uniquement.
3- Le dialogue s'effectue d'une manière HARD.

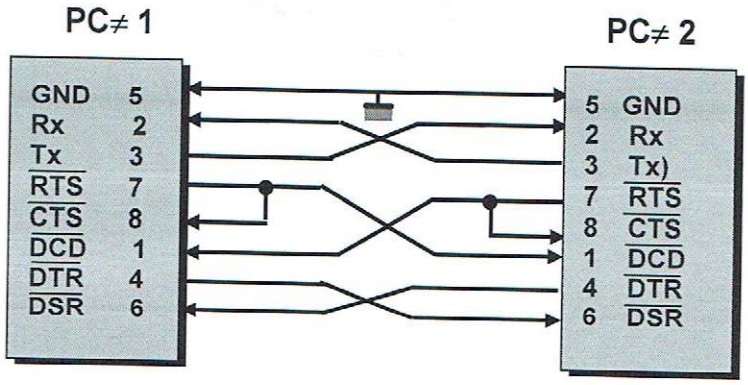


Fig. 3 Deuxième type de liaison NULL-MODEM complet

III.3.2 Câble NULL-MODEM simplifié

Si la vitesse est faible ou si les PC sont assez rapides, pour se passer du dialogue, un câble plus simple à trois fils peut suffir. C'est là où l'utilisation du protocole XON-XOFF est possible.

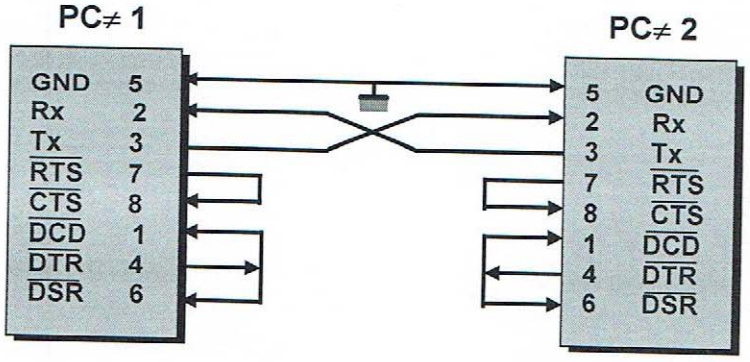


Fig. 4 Liaison NULL-MODEM simplifiée

Remarque: Le dialogue s'effectue d'une manière soft selon le protocole XON-XOFF, où XON (CTRL-S) désigne la demande de lancement de données et XOFF (CTRL-Q) l'arrêt du lancement de données.

IV Interface ou bus USB (Universal Serial Bus)

L'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'un interface parallèle, car celle-ci ne supporte pas les fréquences trop élevées. Dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des décalages provocant des erreurs. De plus, les câbles série coûtent moins chers que les câbles parallèles.

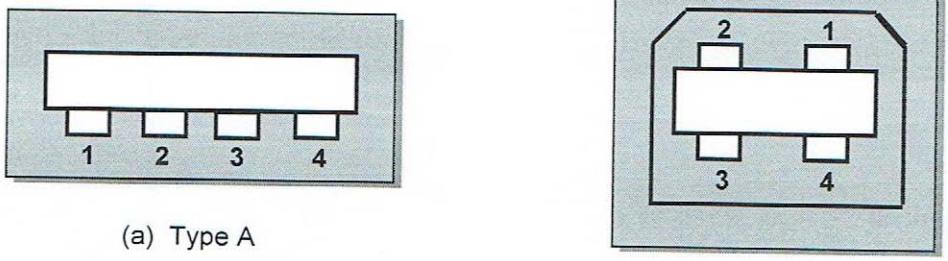
Ainsi, dès 1995, le standard USB a été mis au point pour la connexion d'une grande variété de périphériques. Le standard USB 1.0 propose 2 modes de

communication 12 Mbps en mode haute vitesse et 1.5 Mbps en mode basse vitesse. Le standard USB 1.1 apporte quelques modifications aux constructeurs de périphériques USB mais ne modifie en rien le débit. La norme USB 2.0 permet d'atteindre des débits de 480 Mbps. Cependant, la compatibilité entre les périphériques USB 1.0, 1.1 et 2.0 est assurée. Néanmoins, l'utilisation d'un périphérique USB 2.0 sur un port USB à bas débit (1.0 ou 1.1), limite le débit à 1.5 Mbps au maximum. De pus, le système d'exploitation est susceptible d'afficher un message expliquant que le débit est bridé.

En 2008, l'USB 3.0 a vu le jour. D'une part, il permet d'obtenir 4.8 Gbps contre 12 Mbps pour l'USB 2.0. D'autre part, il contient 6 contacts contre 4 pour les anciennes versions. Il est attendu pour le grand public en 2010.

IV.1 Types de connecteurs USB

Le standard USB 1.0, 1.1 et 2.0 est constitué de 4 broches, définies comme suit:
1- Alimentation de +5 V (V Bus) développant un courant de 100 mA au maximum.
2- Données (D^-)
3- Données (D^+)
4- GND (Masse)



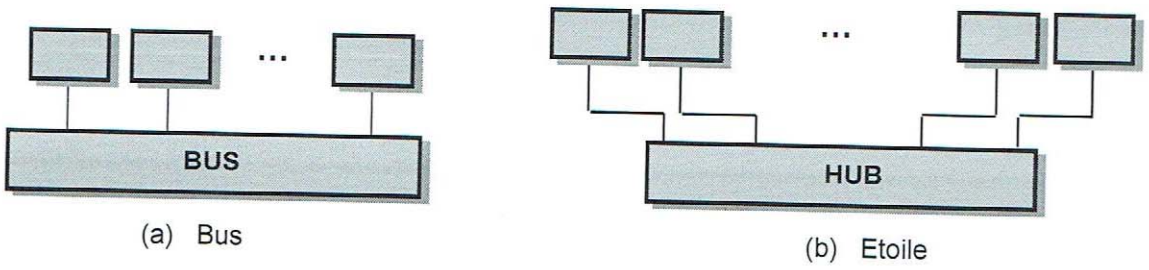
(a) Type A

(b) Type B

Fig. 5 Connecteurs USB normes 1.0, 1.1 et 2.0 (a) Type A et (b) Type B

IV.2 Fonctionnement du bus USB

L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie, dans la limite de 15 V maxi/périphérique. La norme USB permet le chainage des périphériques, en utilisant une topologie en Bus ou en Etoile. Les périphériques pouvant alors être soit connectés les uns à la suite des autres, soit ramifiés.



(a) Bus (b) Etoile

Fig. 6 Topologie en Bus ou en Etoile de la norme USB 1.0, 1.1 et 2.0

La communication entre l'hôte (l'ordinateur) et les périphériques se fait selon un protocole ou langage de communication basé sur le principe de l'anneau à jeton (token ring). Cela signifie que la bande passante est partagée temporellement entre tous les périphériques connectés. L'hôte (l'ordinateur) émet un signal de début de séquence chaque milliseconde (ms), intervalle de temps pendant lequel il va donner simultanément la 'parole' à chacun d'entre eux. Lorsque l'hôte désire communiquer avec un périphérique, il émet un 'jeton' (un paquet de données, contenant l'adresse du périphérique, codée sur 7 bits) désignant un périphérique. C'est donc l'hôte qui décide du 'dialogue' avec ses périphériques. Si le périphérique reconnait son adresse dans le jeton, il envoie un paquet de données (de 8 à 255 octets) en réponse, sinon il fait suivre le paquet aux autres périphériques connectés. Les données ainsi échangées sont codées sur 7 bits. Par conséquent 128 périphériques (2^7) peuvent être connectés simultanément çà un port de ce type. Il convient en réalité de ramener ce chiffre à 127 car l'adresse '0' est réservée.

A raison d'une longueur de câble de 5 mètres au maximum entre 2 périphériques et d'un nombre maximal de 5 Hubs (alimentés), il est possible de créer une chaine de 25 mètres. Le port USB supporte le 'Hot Plug and Play'. Ainsi, les périphériques peuvent être connectés sans éteindre l'ordinateur.

Enfin, l'interface WUSB (Wireless USB) a une portée maximale de 10 mètres et un débit de 110 Mbps contre une portée de 5 mètres et un débit de 480 Mbps pour le l'USB câblé. Il faut savoir que l'interface WUSB ne perturbe pas les liaisons WIFI ou Bluetooth à 2.4 GHz, car elles reposent sur la technologie radio courte Ultra Wide Band (UWB). Cette technologie traverse mieux les obstacles et exploite les fréquences allant de 3.1 GH à 10.6 GHz. Cependant, les industriels ont abandonné l'usage de cette norme pour se consacrer au standard USB 3.0.

EXERCICES

TRAVAUX DIRIGES N°1

Exercice 1: La division entière de deux nombres positifs A et B donne pour résultat un quotient Q et un reste R strictement inférieur au diviseur. On note A div B=Q avec R < B, ces quatre nombres entiers vérifient la relation A=Q.B+R. L'algorithme de la division entière consiste à répéter les soustractions successives ((A-B)-B)-B..., jusqu'à ce que le résultat soit inférieur à B. Le quotient Q sera égal au nombre de soustractions effectuées, et R au résultat final des soustractions. Le schéma de calcul, figure 1, est matérialisé par un circuit soustracteur qui réalise $M=OP_1-OP_2$ et un compteur C. Le registre M est doté d'un bit de signe S et d'un bit de zéro Z qui vérifient les conditions S=1 si M < 0 et Z=1 si M=0.

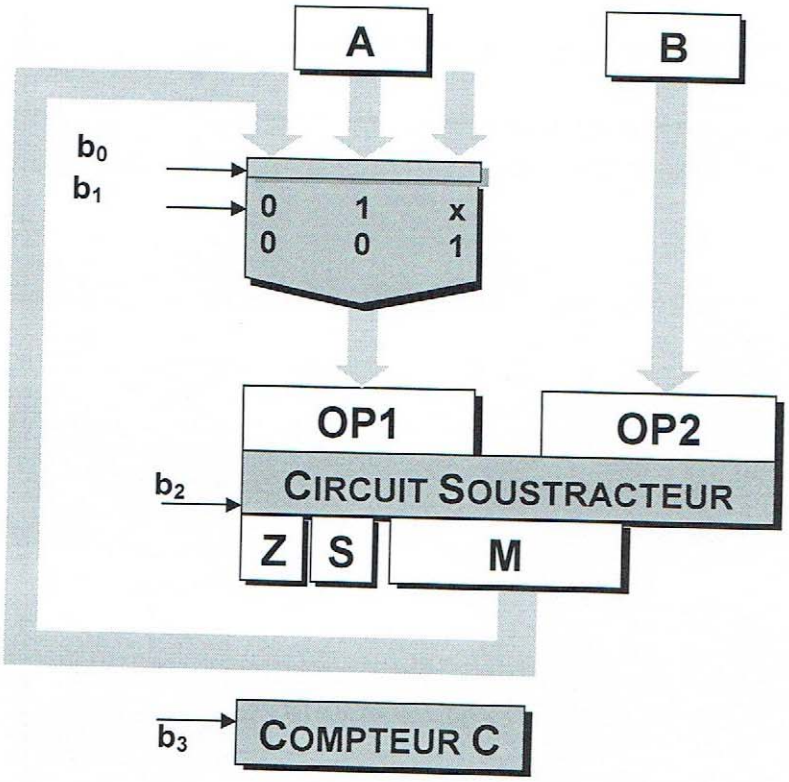


Fig. 1 Schéma de calcul d'une division entière de deux nombres positifs

Algorithme de calcul:

1. Sans rien charger dans OP1", faire la soustraction OP1-OP2.
2. Si M=0 fin de calcul (division par zéro).
3. Sinon charger A dans OP1.
4. Faire la soustraction OP1-OP2.

5. Si M < 0 fin de calcul (avec Q = C et R = OP1)
6. Sinon incrémenter C et charger M dans OP1
7. Allez à 4.
8. Fin de calcul (automate bloqué dans cette étape).

Il vous est demandé de respecter la structure du mot de commande qui doit être comme suit:

b_3	b_2	b_1	b_0

b_0 et b_1: Commande du multiplexeur.
b_2 = 1: Activation du soustracteur.
b_3 = 1: Activation du compteur.
OP1, S, M et C ont tous pour valeur initiale 0.

1- En respectant toutes les données de l'énoncé, réaliser la partie commande de l'automate incrémental qui matérialise cet algorithme.
2- Donner un schéma complet de l'automate ainsi réalisé (partie traitement, partie commande et connexion).

Exercice 2: Apporter une modification au séquenceur (compteur de programme) de l'automate incrémental afin de le rendre capable de prendre en charge un appel de sous programme.

Exercice 3: Soit un automatisme dont la partie opérationnelle est décrite par le schéma de la Figure 2.

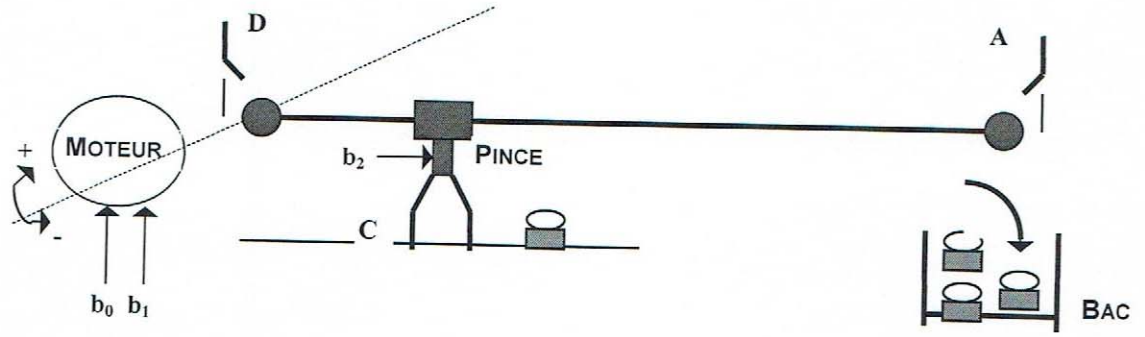


Fig. 2 Exemple d'un automate

Une pince, entraînée par un moteur, effectue un mouvement d'aller - retour le long du segment DA. La fonction de cet automatisme étant de ramasser des pièces éventuellement présentes sur le trajet de la pince et de les déposer au niveau d'un bac situé au niveau de l'extrémité droite.

Description des flags:

A: Contacteur de détection de fin de course droite
La pince est arrivée à l'extrémité droite (contact fermé) A=0
La pince n'a pas atteint l'extrémité droite (contact ouvert) A=1

D : Contacteur de détection de fin de course gauche

La pince arrive à l'extrémité gauche (contact fermé) D=0
La pince n'a pas atteint l'extrémité gauche (contact ouvert) D=1

C : Détecteur de présence de la pièce sous la pince.

Si la pièce est absente C=1.
Si la pièce est présente sous la pince, C bascule à 0 et reste maintenu à 0 jusqu'à l'ouverture de la pince. (C'est l'ouverture de la pince qui provoque la remise à 1 du flag C).

Description et fonctionnement des actionneurs:

b_1, b_0: Commande du moteur de déplacement.
b_2 : Commande d'ouverture/fermeture de la pince.

b_1 b_0	Action
0 1	Rotation du moteur dans le sens positif
1 0	Rotation du moteur dans le sens négatif

b_2	Action
0	Ouverture de la pince
1	Fermeture de la pince et remise à zéro du flag C

Algorithme de fonctionnement:

1. Actionner le moteur dans le sens positif et ouvrir la pince.
2. Si $(C \cdot A = 1)$ aller à 2.
3. Si $(\bar{A}=1)$, aller à 5.
4. Fermer la pince.
5. Si (A=1) Aller à 5.
6. Actionner le moteur dans le sens négatif et ouvrir la pince.
7. Si (D=1) aller à 7.
8. aller à 1

1- Donner l'organigramme correspondant à l'algorithme de la Figure 3.
2- Programmer l'automate incrémental représentant l'unité de commande du dispositif de la Figure 4.

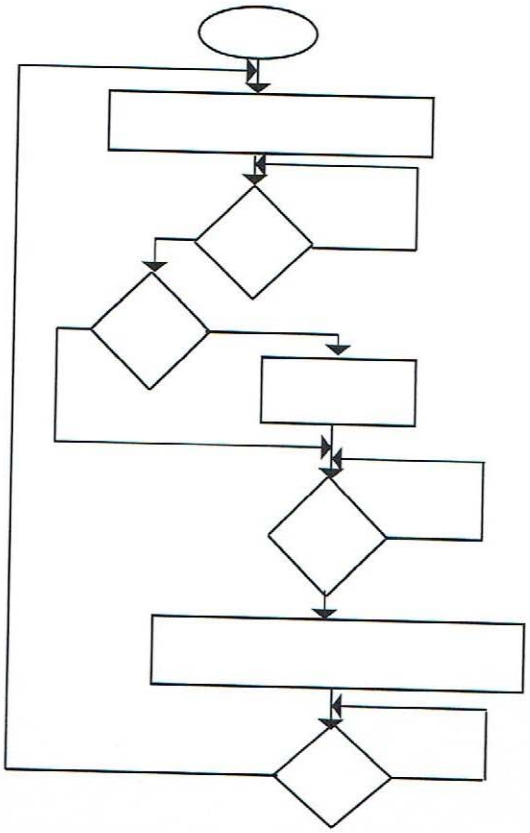


Fig. 3 Organigramme

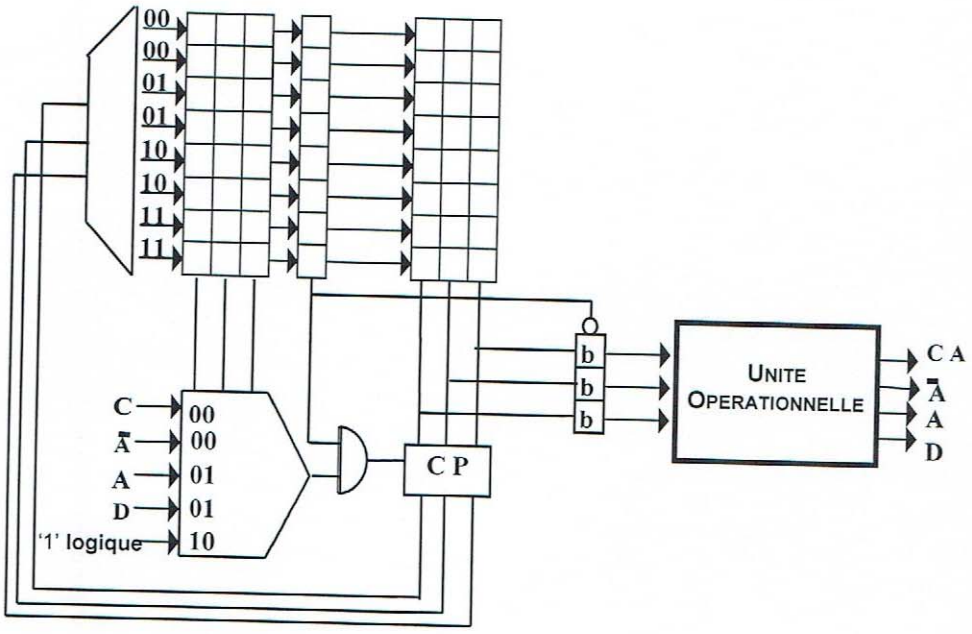


Fig. 4 Automate

Travaux Dirigés N°2

Exercice 1: Représentation des nombres binaires

1- Donner les différentes représentations binaires d'un entier codé sur 8 bits (Binaire naturel, Complément à 1, Complément à 2). Spécifier les valeurs maximales et minimales pour chaque type de représentation.
2- Etablir la relation de passage de la représentation binaire naturelle à la représentation complément à 2.
3- Généraliser pour un nombre entier codé sur N bits.

Exercice 2: Changement de bases

1- Convertir en hexadécimal les nombres décimaux suivants: 205, 2988, 49374 et 51966.
2- Convertir en décimal les nombres hexadécimaux suivants: 400, 7CD, C350 et 7FFF.

Exercice 3:

1- Effectuer les additions suivantes en compléments à 2 sur 8 bits et donner l'état du flag 'overflow' pour chacune d'elles.

a- (+90) + (+50)
b- (+90) + (-50)
c- (-90) + (+50)
d- (-90) + (-50)

2- En déduire l'équation logique qui réalise le flag 'overflow'

Exercice 4: Circuit d'addition et de soustraction parallèle

1- Donner les équations logiques et le circuit qui réalise l'addition binaire élémentaire.
2- Proposer un circuit à structure parallèle pour l'addition et la soustraction de deux nombres représentés en complément à 2 sur 8 bits.

Exercice 5: Circuit d'addition et de soustraction série

1- Donner les principaux inconvénients du circuit proposé dans l'exercice
2- Proposer un algorithme pour l'addition de deux nombres codés sur N bits.
3- En déduire le circuit qui le matérialise.

Exercice 5: On dispose d'un système microprogrammé dont les deux fonctions élémentaires (addition 8 bits avec et sans retenue) sont disponibles. Faire l'organigramme qui effectue l'addition multi-octets.

Exercice 6: Soit une représentation de nombres réels à champ fixe caractérisée par une partie entière à 8 bits et une partie fractionnaire à 4 bits.

1- Représenter les nombres réels 126.56, 12.09, 255.55, -61.20 et -128. Calculer l'erreur d'arrondi pour chaque cas.
2- En déduire, pour une représentation en virgule fixe renferment 1 bit pour la partie entière et k bits pour la partie fractionnaire, l'erreur d'arrondi et les valeurs limites (nombres réels signés).

Exercice 7: Tout nombre peut être représenté sous forme d'un réel et d'un facteur multiplicatif (puissance de 2). Cette représentation est dite flottante, la partie réelle constitue la mantisse et la puissance de la base constitue l'exposant.

1- Prendre une représentation en virgule flottante définit par:

Exposant

Mantisse

2- Représenter le nombre π = 3.1416. A l'aide de ce format, vérifier que l'erreur d'arrondi est moindre pour une représentation normalisée, calculer l'erreur d'arrondi commise.
3- Donner la procédure de normalisation après une opération qui dénormalise la mantisse.

Exercice 8: Représentation des nombres en virgule flottante

1- Etablir les limites d'un nombre représenté dans le format virgule flottante suivant: sm se exposant (Kbits) mantisse (Nbits)
2- Proposer une représentation du nombre zéro (zéro propre et zéro corrigé)

Exercice 9: Norme IEEE 754 simple précision

Ce format standard formalise la représentation de nombres particuliers ($+\infty$,$-\infty$, 0) et permet de représenter les objets qui ne sont pas des nombres appelés NaN (Not a Number). Toute opération sur ces objets interrompt l'exécution du processus de calcul. Ce format code l'exposant sur 8 bits et la mantisse sur 24 bits dont un de signe. La mantisse étant purement fractionnaire et normalisée.

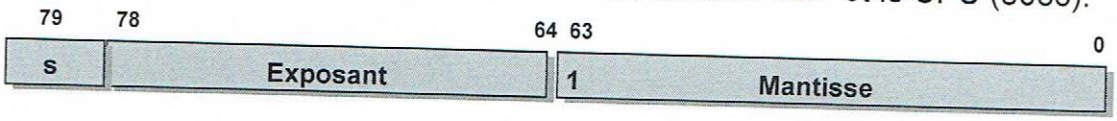
1- Donner l'expression d'un nombre N dans cette représentation.

2- Définir l'étendue de variation de ce format.
3- Donner la représentation des nombre particuliers +0, -0, +∞, -∞ et NaN.

Exercice 10: Soit un format en virgule flottante définit par un exposant codé en excédant à 2^7 et une mantisse codée sur 24 bits dont un bit de signe. Reprendre l'algorithme de normalisation de la mantisse en utilisant les registres du 8086. La mantisse est stockée à partir de l'adresse MADR et l'exposant à partir de l'adresse EXPADR.

Exercice 11: Norme IEEE 754 double précision

Le NDP:8087 conditionne plusieurs formats d'entrées et de sorties. Le formalisme qui représente le format TBYTE (IEEE) en Fortran est dit 'format réel temporaire'. Ce format n'est utilisé en mémoire que pour garder des résultats intermédiaires, lorsque la pile interne du 8087 est pleine. C'est sous cette forme que toutes les opérations internes sont effectuées. Les autres formats ne sont utilisés que pour les échanges de valeurs, à travers la mémoire, entre le NDP et le CPU (8086).



1- D'après vous, que désigne la lettre 'T' de TBYTE. Justifiez votre réponse.
2- D'après la figure ci-dessus, quelle est la longueur de l'exposant et celle de la mantisse. Que représentent '1' et 's' .
3- Sachant que tout nombre réel X s'écrit dans ce format en excédent à 214, donnez la formule littérale de ce réel dans ce format.
4- Définissez l'étendue de variation de X dans ce format (littérale et numérique approximée en puissance de 10).
5- Donnez la représentation des nombres réels particuliers +0, -0, +∞, -∞ et NaN (Not a Number).

Exercice 12: Soit un nombre réel X négatif. On se propose de le représenter dans un format de nombres réels à champ fixe caractérisé par une partie entière de 6 bits dont un bits de signe et une partie fractionnaire de 4 bits, d'une part et dans un format en virgule flottante défini par un exposant codé sur 6 bits (en excédent à 2^5) et une mantisse normalisée sur 3 bits et un bit signe, d'autre part.

1- Donner l'expression littérale du nombre X dans le format à champ fixe.
2- Représenter le nombre –7.45 dans ce format.
3- Définir l'étendue de variation de X dans ce format.
4- Donner l'expression littérale du nombre X dans le format à champ flottant.
5- Représenter le nombre –7.45 dans ce format.
6- Définir l'étendue de variation de X dans ce format.

Travaux Dirigés N°3

Cours: Evaluation des connaissances sur les PC. Il s'agit d'un QCM (Questions choix multiples). Il n'y a qu'une réponse juste et une seule par question. Cercler la lettre correspondant à la réponse juste.

Question 1: Que désigne t-on par bus local.

a- un bus dont les signaux sont pris directement sur le processeur 8086.
b- un bus dont les signaux ne sont pas pris directement du processeur 8086.
c- un bus dont les signaux sont issus des circuits 8282, 8286 et 8288.
d- un bus dont les signaux sont issus des circuits 8253, 8237 et 8289.

Question 2: Que désigne t-on par bus système.

a- un bus dont les signaux sont issus des circuits 8253, 8237, 8259 et 8289.
b- un bus dont les signaux sont issus du circuit 8087.
c- un bus dont les signaux du 8086 sont passés à travers les circuits 8282, 8286 et 8288.
d- a et c.

Question 3: La carte mère d'un PC XT inclut le processeur

a- MOTOROLA 6845.
b- NEC PD 765.
c- c- 8237.
d- 8042.

Question 4: Laquelle des fonctions suivantes, n'est pas réalisée par un CRTC (MOTOROLA 6845)

a- conversion numérique/analogique.
b- gestion de la RAM vidéo.
c- génération des données (image) sous forme digitale.
d- clignotement du curseur.

Question 5: Laquelle des fonctions suivantes peut-être réalisée par un FDC (NEC PD 765).

a- Le contrôle du crayon optique.
b- Le contrôle des moteurs de disquettes (marche /arrêt).
c- La gestion des codes de recherche (scan codes).
d- L'adaptation monochrome.

Question 6: Lesquelles des fonctions suivantes peuvent être réalisées par un processeur de clavier (8042).

a- La gestion de la communication sérielle entre le clavier et la carte mère.
b- Le diagnostic du clavier lors de la mise sous tension du PC XT.
c- La gestion de la zone tampon.
d- Toutes les réponses ci-dessus.

Question 7: Parmi les bus suivants lesquels sont des bus locaux .

a- SCSI et SATA
b- IDE et MCA
c- VLB et PCI
d- EISA et PCMCIA

Question 8: Que désigna t-on par mémoire cache d'un disque dur.

a- Une mémoire qui sert à améliorer les performances d'accès au disque dur.
b- Une mémoire qui sert à améliorer les performances d'accès au lecteur de disquettes.
c- Une mémoire qui sert à améliorer les performances de la carte graphique.
d- Une mémoire qui sert à améliorer les performances du microprocesseur.

Question 9: Dans l'architecture de Johannes Von Neumann, la structure d'algorithme d'interprétation est le propre de la technologie

a- WISC.
b- CISC.
c- RISC.
d- Aucune des réponses ci-dessus.

Question 10: L'adressage d'une mémoire ayant une taille de N octets, nécessite un registre de

a- N/2 bits.
b- N½ bits.
c- Log_2 (N) bits.
d- 2*N-1 bits.

Question 11: Un chipset est un

a- microprocesseur.
b- circuit d'entrées et de sorties.
c- circuit permettant de gérer les communications entre les composants d'un PC.
d- pare-feu.

Question 12: Un chipset SLI assure

a- le couplage de deux cartes d'entrées et de sorties identiques.
b- le couplage de deux cartes mémoires identiques.
c- le couplage de deux cartes graphiques identiques.
d- le couplage de deux cartes mémoires caches identiques.

Question 13: Un bus PCI express permettant un débit de 4 GO/s est appelé

a- x 1 PCI Express.
b- x 2 PCI Express.
c- x 4 PCI Express.
d- x 16 PCI Express.

Question 14: Parmi les caractéristiques suivantes laquelle n'est pas celle d'un contrôleur de disque dur.

a- Capacité de transfert.
b- Temps d'accès.
c- Taux de transfert.
d- Résolution.

Question 15: Le mode d'affichage VGA émule entièrement les modes

a- MDA et HGA.
b- CGA et EGA.
c- EGA et PGA.
d- MDA et PGA.

Question 16: La norme HDMI transmet un

a- signal audio-vidéo analogique.
b- signal vidéo numérique.
c- signal audio-vidéo numérique.
d- signal vidéo analogique.

Question 17: Parmi les fonctions suivantes, laquelle n'est pas réalisée par le contrôleur de lecteur de disquette FDC 8272.

a- précompensation
b- Phase Locked Loop (PLL)
c- Conversion série parallèle
d- Enregistrement double face

Question 18: Un contrôleur de disque dur SATA 2 peut atteindre une capacité de stockage de

a- 150 GO
b- 250 GO
c- 500 GO
d- 750 GO

Question 19: La technologie qui assure l'assemblage de plusieurs disques durs en une seule unité logique est dite:

a- Redundant Array of Independent Disks
b- Redundant Association of Independent Disks
c- Redundant Array of Inexpensive Disks
d- a et b

Question 20: Dans les modes d'affichage, la résolution désigne

a- le nombre de couleurs.
b- le nombre de bits par pixel.
c- le nombre de pixels.
d- la longueur de l'écran.

Question 21: Le langage que peut interpréter directement un microprocesseur est:

a- le langage C
b- le langage BASIC
c- le langage machine
d- le langage assembleur

Question 22: L'unité de mesure de la puissance d'un ordinateur est

a- le kilo-octet
a- le bit par seconde
b- le million d'instructions par seconde
c- le cycle d'horloge par seconde

Question 23: Un buffer est un circuit digital qui

a- convertit le niveau de tension d'un signal quelconque au niveau TTL
b- augmente le fan-out du circuit aval
c- rend les fronts des signaux plus abrupts
d- réduit la consommation en courant du circuit aval

Question 24: Quelle mémoire nécessite un rafraîchissement périodique des données.

a- La RAM dynamique
b- La ROM
c- La RAM statique
d- L'EPROM

Question 25: Un bus de données est

a- une mémoire qui conserve momentanément les données
b- un réseau de connexions entre deux ordinateurs
c- un câble à plusieurs conducteurs
d- un support physique de transmission de données en parallèle

Question 26: Dans un microprocesseur, les entiers sont codés en

a- code BCD
b- code ASCII
c- complément à 2
d- virgule flottante

Question 27: Le ROM BIOS loge aux adresses mémoire suivantes

a- 0000:0000-0000:FFFF
b- F000:0000-0000:FFFF
c- F000:0000-F000:FFFF
d- B000:0000-B000:FFFF

Question 28: Parmi les fonctions suivantes, laquelle n'est pas réalisée par le contrôleur de clavier 8042.

a- Diagnostique de l'état du clavier à la mise sous tension
b- Génération du scan code
c- Transmission en série des données
d- Génération du code ASCII

Question 29: Parmi les circuits suivants, lesquels sont des 'buffers' bidirectionnels.

a- 8286
b- 8283
c- 8287
d- a et c

Question 30: Parmi les circuits suivants, lesquels sont des 'latches' d'adresses.
a- 8283
b- 8282
c- 8288
d- a et b

Question 31: Parmi les circuits suivants, lequel est un contrôleur de bus.

a- 8283
b- 8284
c- 8288
d- 8286

Question 32: Dans un PC, le clavier interrompt le microprocesseur à travers la ligne.

a- IRQ_2
b- IRQ_1
c- IRQ_0
d- IRQ3

Question 33: Laquelle des caractéristiques suivantes ne correspond, pas à celle d'un bus.

a- Fréquence
b- Largeur
c- Temps d'accès
d- Débit

Question 34: Laquelle des ressources, n'est pas gérée par un chipset.

a- Mémoire
b- Générateur d'horloge
c- Hot plug PCI
d- Bus USB

Question 35: Lequel des interfaces suivantes n'est pas celle d'un disque dur.

a- IDE
b- Fire Wire
c- SCSI
d- SATA

Question 36: Lesquels des modes d'affichages suivants correspondent à un affichage WYSIWYG.

a- SVGA
b- HGA
c- VGA
d- a et c

Question 37: Lequel des bus suivants, correspond à celui d'un contrôleur graphique.

a- PCMCIA
b- USB
c- AGP
d- PCI

Question 38: L'ordre chronologique d'apparition des mémoires suivantes est:

a- DDR-EDORAM-SDRAM-BEDORAM-DRAM-DDR2-DDR3
b- EDORAM-DRAM-DDR-SDRAM-BEDORAM-DDR2-DDR3
c- DRAM-EDORAM-BEDORAM-SDRAM-DDR-DDR2-DDR3
d- SDRAM-BEDORAM-EDORAM-DDR-DRAM-DDR2-DDR

TRAVAUX DIRIGES N°4

Cours 1: Evaluation des connaissances sur le 8086. Il s'agit d'un QCM (Questions choix multiples). Il n'y a qu'une réponse juste et une seule par question. Cercler la lettre correspondant à la réponse juste.

Question 1: Pour initialiser un 8086, il faut

a- 2 cycles d'horloge.
b- 3 cycles d'horloge.
c- 4 cycles d'horloge.
d- 5 cycles d'horloge.

Question 2: Le transmetteur bidirectionnel 8286/8287, travaille en transmission ou en réception à la l'aide de la broche

a- $\overline{OE}$
b- Vcc
c- T
d- GND

Questions 3: Le latch d'adresses 8282/83 permet de capturer l'adresse à l'aide de la broche

a- $\overline{OE}$
b- STB
c- GND
d- T

Questions 4: Le contrôleur de bus 8288 est utilisé exclusivement en

a- mode minimum
b- mode minimum et maximum
c- mode maximum
d- ni a ni b

Questions 5: Les connexions entre le 8288 et les 8286/87 et 8282/83 sont:

a- DT/$\overline{R}$ avec STB pour le 8286/87 et ALE avec T pour le 8282/83.
b- ALE avec STB pour le 8282/83 et DT/$\overline{R}$ avec T pour le 8286/87.
c- ALE avec T pour 8286/87 et DT/$\overline{R}$ avec STB pour le 8282/83.
d- Aucune des réponses ci-dessus.

Questions 6: Lequel des signaux suivants du 8086 est utilisé en mode minimum.

a- $\overline{S_0}$
b- $\overline{S_1}$
c- QS_0
d- HOLD

Questions 7: Lequel des signaux suivants du 8086 est utilisé en mode minimum.

a- $\overline{WR}$
b- $\overline{LOCK}$
c- $\overline{RD}$
d- $\overline{INTA}$

Questions 8: Lequel des signaux suivants du 8086 est commun aux modes maximum et minimum.

a- HOLD
b- QS_0
c- $\overline{BHE}$
d- $\overline{DEN}$

Questions 9: Laquelle des combinaisons registre segment/offset n'est pas réalisable.

a- DS:BP
b- SS:DI
c- CS:BX
d- ES:IP

Questions 10: Quel est le registre segment associé à l'instruction MOV AX,[SP][DI].

a- SS
b- CS
c- ES
d- DS

Cours 2: Compréhension du cours

1- Donner (en K octets) la longueur de l'espace mémoire directement adressable par le 8086.

2- Donner l'adresse physique générée sur 20 bits par le 8086 lorsque, d'une part, le registre segment contient la valeur F460 et le registre offset correspondant contient la valeur 2E20 H et d'autre part, le registre segment contient EA71 H et le registre offset CD10 H.
3- Que peut-on conclure.
4- Donner (en K Octets) la longueur de l'espace mémoire adressable par le registre offset lorsque le registre segment correspondant est maintenu à une valeur fixe
5- Donner la longueur du saut d'adresse effectuée lorsque le registre offset est maintenu à une valeur fixe et que le registre segment correspondant est incrémenté d'une unité.
6- Donner les différentes zones mémoire d'un PC-AT et le registre segment correspondant.
7- Expliquer brièvement les signaux: HOLD, HLDA, $\overline{RQ}$ /$\overline{GT}$. Spécifier le mode dans lequel ils sont fonctionnels.
8- Expliquer brièvement les signaux $\overline{TEST}$ et $\overline{LOCK}$. Spécifier le mode dont lequel ils sont fonctionnels.
9- Donner la définition d'un bus local et d'un bus système.
10- Expliquer brièvement les signaux $\overline{INTR}$ et $\overline{INTA}$. Spécifier le mode dans lequel ils sont fonctionnels.

Cours 3: Compréhension du cours

1- Sous l'effet de quel évènement est sollicitée la broche NMI du microprocesseur Intel 8086 dans une carte PC. Expliquer.
2- Expliquer le mécanisme d'insertion des temps d'attente dans le cycle de lecture/écriture mémoire du microprocesseur Intel 8086.
3- La broche $\overline{BHE}$ est fondamentalement une ligne d'adresse qui, ajoutée à A_{19}-A_{16} et AD_0-AD_{15}, ramène à 21 le nombre de bits d'adressage et donc 2 méga-octets le champ adressable par le 8086. Expliquer pourquoi seul 1 méga-octets de mémoire est effectivement adressable.
4- Expliquer le mécanisme du Prefetch dans le microprocesseur Intel 8086. Donner la combinaison des broches $\overline{S_0}, \overline{S_1}$ et $\overline{S_2}$ correspondante.

Exercice 1: On désire réaliser les deux configurations types (minimum et maximum) d'un système à base du microprocesseur Intel 8086. On dispose pour cela, des circuits suivants:

- microprocesseur du type Intel 8086 fonctionnant à 4.77 MHz,
- générateur d'horloge du type Intel 8284,
- quartz de 15 Mhz,
- latch d'adresse du type 8282,
- transmetteur bidirectionnels du type 8286,

- contrôleur de bus du type 8288 (IOB = 0),
- boîtiers mémoires RAM, organisés selon la boite noire (Voir Figure 1.a),
- boîtiers mémoires PROM, organisés selon la boite noire (Voir Figure 1.b),
- boîtiers d'Entrées/Sorties, organisés selon la boite noire (Voir Figure 1.c),
- divers circuits tels que résistances, capacités, diodes, portes logiques, etc.

1- Donner le schéma du circuit 8086/8284. On précisera les circuits reset et horloge. Justifier les connexions utilisées.
2- Donner le schéma synoptique de la configuration 'minimum' du 8086. Indiquer clairement les signaux impliqués dans cette configuration et justifier les connexions utilisées (pas de signaux d'interruption ni de signaux de demande de bus).
3- Donner le schéma synoptique de la configuration 'maximum' du 8086. Indiquer clairement les signaux impliqué dans cette configuration et justifier les connexions utilisées (pas de signaux d'interruption ni de signaux de demande de bus).
4- On peut imaginer que les boites noires RAM et PROM soient en fait constituées de 4 x 1Ko de boîtiers mémoires RAM de type 2142 et de 2 x 2 K octets de boîtiers mémoires ROM de type 2716, respectivement. Par ailleurs, on suppose que les adresses RAM et ROM commencent, respectivement, à partir de 00000H et F0000H. Donner pour cela, l'organisation d'un tel espace mémoire. Utiliser un décodage linéaire, c'est à dire, un décodage d'adresse à l'aide de portes logiques uniquement.
5- En tenant compte des circuits suscités, dans quel cas l'utilisation d'un circuit gestionnaire de temps d'attente s'avère telle nécessaire.

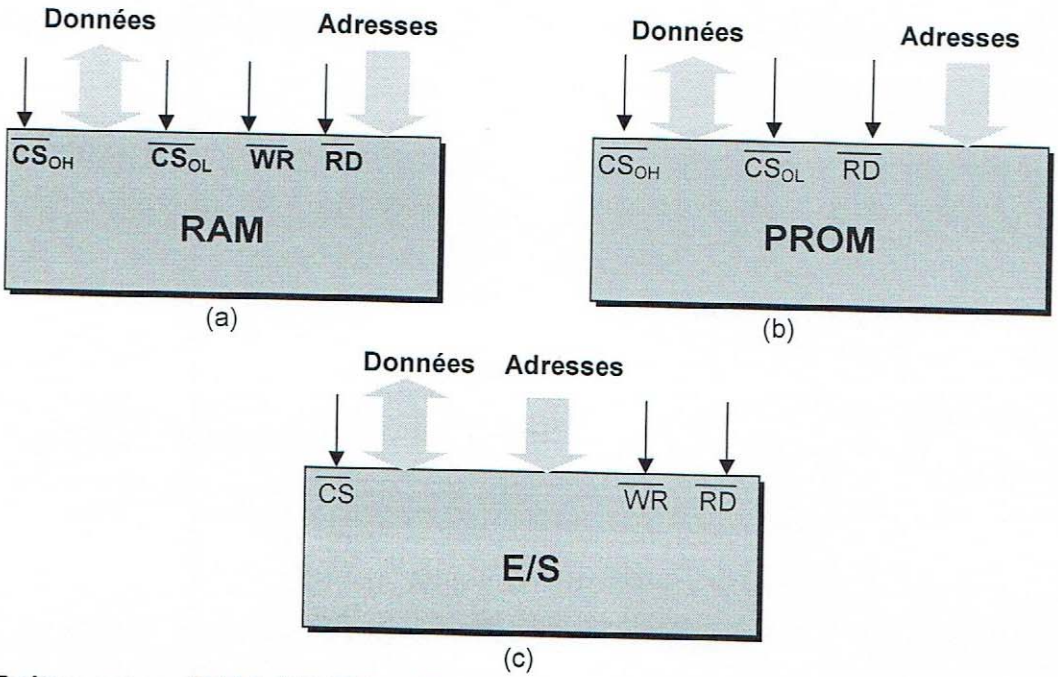


Fig. 1 Boîtes noires RAM, PROM et E/S

Exercice 2: Dans une carte à base de microprocesseur Intel 8086 fonctionnant en mode minimum, on s'intéresse uniquement aux quatre Kilos Octets de mémoire vive situés aux adresses physiques 00000H à 00FFCH, et à un périphérique d'entrée/sortie occupant les ports d'adresses 378H à 37FH. Il vous est demandé de câbler cette partie de la mémoire et des périphériques d'entrées/sorties.
Pour cela vous disposez des circuits suivants:

- Deux RAM statiques de 2 K octet dotées chacune d'une broche $\overline{CS}$, une broche $\overline{WR}$ (Write) et une broche $\overline{RD}$ (Read).
- Un périphérique d'entrées/sorties constitué de huit registres internes de huit bits chacun ayant comme broches communes $\overline{CS}$, $\overline{RD}$ et $\overline{WR}$. Cependant leurs lignes d'adresses respectives sont distinctes.
- Deux latches 8082.
- Deux transmetteurs bidirectionnels 8286.
- Diverses portes logiques AND, NAND, NOR, NOT, XOR.

Indication*:* Générez les $\overline{CS}$ et par analogie au mode maximum, réalisez les connexions de $\overline{MEMRD}$, $\overline{MEMWR}$, $\overline{IORD}$ et $\overline{IOWR}$ et des lignes d'adresses et de données. Attention, il faudrait tenir compte des octets hauts et bas du bus de données, ainsi que de l'accès mémoire et celui des entrées/sorties.

Exercice 3: Les mémoires RAM dynamiques parmi les plus utilisées dans les cartes PC-XT sont du type 4164, de capacité 64 Kilo-bits. Leur diagramme interne montre quelles sont constituées de 64 lignes x 64 colonnes de cellules mémoire représentant chacune 1 bit d'information. Deux bus multiplexés dans le temps doivent être prévus pour les broches A_0-A_7. Toujours, à partir de ce diagramme, les adresses de sélection ligne et adresses de sélection colonne sont celles-là même qui se présentent sur les broches A_0-A_7. Ce mécanisme permet en association avec les broches $\overline{RAS}$ (Row Address Strobe ou sélection ligne) et $\overline{CAS}$ (Column Address Strobe ou sélection colonne) de sélectionner un bit précis à l'intérieur de la matrice, et offre en outre l'avantage d'utiliser uniquement 9 broches pour adresser 65536 bits. Le bit adressé est accessible sur la broche D_{in} en écriture et D_{out} en lecture. Une même broche $\overline{W}$ est utilisée pour l'activation du bit aussi bien en écriture qu'en lecture. Voir brochage, diagramme interne et chronogrammes de la Figure 2.

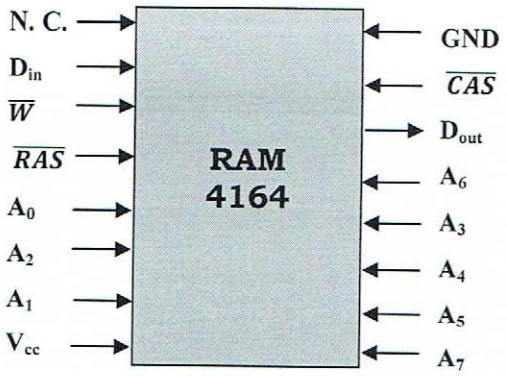
1- Donner le nombre de boîtiers 4164 nécessaires pour réaliser les 128 Kilo-octets de mémoire dynamique et l'adresse finale si nous supposons que l'adresse physique de départ est 50000 H - ?H, d'une carte à base de microprocesseur Intel 8086.
2- Donner le schéma synoptique global pour l'interfaçage de l'ensemble de ces boîtiers avec le microprocesseur Intel 8086 en mode maximum, où l'on fera apparaître clairement les lignes du bus d'adresses utilisé, ainsi que la partie basse

et la partie haute du bus de données uniquement. Pour ne pas encombrer le schéma, il ne faut pas faire apparaitre le contrôleur de bus (8288), les buffers bidirectionnels (8286/87) et les latches d'adresses (8282/83).

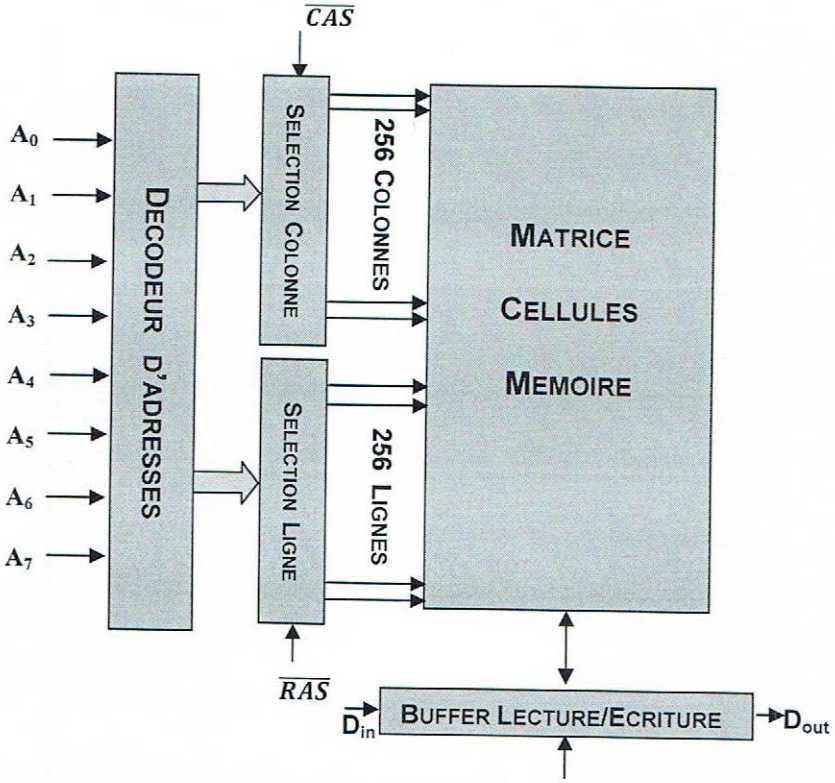
3- Donner le détail de la connexion des broches Din et Dout d'un boîtier 4164 de votre choix. Vous devez utiliser uniquement les signaux D_i (i=0, 1, 2, ..., 15) du bus de données du 8086 issus des 8286/87, le signal $DT - \overline{R}$ du 8288 et des buffers tri-state.

4- Donner le détail de la connexion des broches $\overline{RAS}$ et $\overline{CAS}$ d'un boîtier 4164 de votre choix. Ces signaux, considérés comme des broches de sélection de boîtier, ne doivent différer que par une bascule D (Delay Flip Flop).

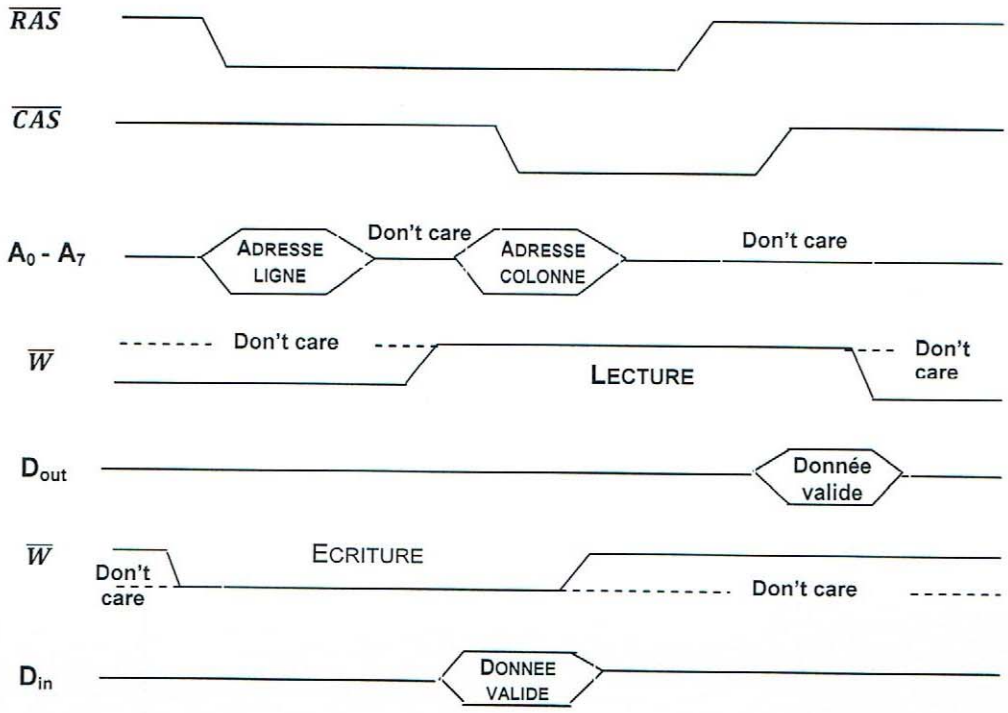
5- Donner le détail de la connexion de la broche $\overline{W}$ d'un boîtier 4164 de votre choix. Ce signal est l'association des deux signaux $\overline{MWTC}$ et $\overline{MRDC}$ du 8288.



(a) Brochage



(b) Diagramme interne



(c) Chronogramme

Fig.2 Caractéristiques d'une mémoire RAM de type 4164

TRAVAUX DIRIGES N°5

Exercice 1: Donner pour chacune des instructions suivantes, si elle figure dans le jeu d'instructions du 8086.

MOV AX,DX; MOV AL,[BX+SI]; PUSH AX; PUSH ES:[BP+SI]; SUB [SI+BP],AX; INC WPTR[BX]; DAA; TEST AL,40H; DIV [DI] ; MOVSB; RCR [SI],CL; LOOP 200H; JMP FFH ; CALL F000H; CALL FAR[SI]; ESC 15,DX.

1- Le groupe auquel elle appartient.
2- Le nombre d'octets qu'elle occupe.
3- Le contenu de chacun des octets avec leur disposition en mémoire.
4- Le mode d'adressage qu'elle utilise.
5- L'état des flags après son exécution.

Etats initiaux en hexadécimal des registres du 8086:

AX = 0000, BX = 1000, CX = 00FF, DX = 0010, CS = FFFF, SS = FFFF, DS = FFFF, ES = FFFF, IP = 0100, SP = 0FFF, BP = 0000, SI = 0080 et PSW = 0000.

Exercice 2: Instruction de manipulation de chaine de caractère

1- Donner l'équivalent de REP SCASB à l'aide de l'instruction CMP et d'une instruction de saut de votre choix.
2- Donner le nombre d'octets qu'occupe l'instruction rep scasb et la séquence équivalente.
3- Donner le contenu de chacun des octets de rep scasb et la séquence équivalente.

Exercice 3: A la suite d'une instruction d'addition, il est possible à 4 parmi les 8 valeurs suivantes de figurer dans le registre des flags du 8086.

F02A F06A F06E F0AB F0AE F0EE F82A F8AB

Dites lesquelles et pourquoi.

Exercice 4: Ecrire une séquence d'instruction en assembleur 8086 totalisant 3 octets en code machine, qui permet de charger la valeur zéro "0" dans les registres DX et AX.

Exercice 5: Soit la séquence d'instructions suivantes

```
0E17:0100   MOV  AX,25B6H
0E17:0103   PUSH AX
0E17:0104   MOV  AX,0133H
0E17:0107   MOV PUSH AX
0E17:0108   RET  FAR
```

1- Donner l'adresse (sous forme SEGMENT:OFFSET) de l'instruction qui sera exécutée juste après l'instruction située à l'adresse 0E17:0108.
2- Si MOV AX,25B6H était située à l'adresse 0E17:FFFF, quelle serait la prochaine valeur du couple adresse CS:IP.

Exercice 6: Etablir un programme en assembleur 8086 permettant d'effectuer l'addition de deux nombres N1 et N2. On suppose que les deux nombres sont stockés en mémoire dans le même segment à partir des offset ADR1et ADR2, respectivement. Le résultat en Hexadécimal sera stocké dans le même segment à partir de l'offset ADR. Utiliser la directive DEFB ou DEFW pour définir les variables. Que devient ce programme si l'on désire travailler en BCD. Reprendre le même exercice pour deux nombres multi-mots.

Exercice 7: Recherche du maximum (minimum) dans un tableau de N éléments.

Donner l'organigramme et le programme en assembleur 8086 correspondant à la recherche du maximum (minimum) d'un tableau de N éléments. Les éléments du tableau sont placés à partir de la case DS:[200] et la case DS:[300] contient le nombre d'éléments nuls du tableau.

Exercice 8: Calcul du nombre d'éléments nuls dans un tableau de N éléments.

Donner l'organigramme et le programme en assembleur 8086 correspondant au calcul du nombre d'éléments nuls dans un tableau de N éléments. Les éléments du tableau sont placés à partir de la case DS:[200] et la case DS:[300] contient la valeur maximale (minimale) du tableau.

Exercice 9: Calcul du nombre de bits égaux à '1' dans un mot de 16 bits.

Donner l'organigramme et le programme en assembleur 8086 correspondant au calcul du nombre de '1' contenu dans un mot de 16 bits. Le mot est situé à partir de l'adresse DS:[200] et le nombre de '1' est situé à l'adresse [202].

Exercice 10: Calcul du PGDC et du PPMC de deux nombres strictement positifs.

Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de calculer le plus grand diviseur commun (PGDC) et le plus petit multiple commun (PPMC) de deux nombres strictement positifs sur 8 bits stockés, respectivement, aux adresses NBRE1 et NBRE2. Le PGDC et le PPMC seront stockés, respectivement, aux adresses PGDC et PPMC. Utiliser pour cela l'algorithme d'Euclide.

$\forall$ m ≥ n entiers strictement positifs où m ≥ n,
PGDC (m,n)=PGDC(n,reste[m/n]) et PGDC(n,0)=n $\forall$ n
Par ailleurs, PPMC (m,n)= $\frac{\mathbf{m \times n}}{\mathbf{PGDC(m,n)}}$

Exercice 11: Insertion d'un bloc de données.

Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant d'insérer un bloc de M octets situé à partir de l'adresse BLOC à l'intérieur d'une zone mémoire de N octets située à partir de l'adresse ZONE. L'insertion du bloc doit commencer à partir de l'adresse d'offset INSERT. Prendre N=1000H, M=100H et INSERT=200H.

Indication: L'utilisation des pseudos instructions est obligatoire dans les trois exercices.

Exercice 12: Calcul des codes ASCII relatifs à un tableau de N nombres BCD.

Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant calculer les codes ASCII relatifs à un tableau de N nombres BCD. Chaque nombre BCD étant codé sur 4 bits (1digit). Les digits sont groupés deux à deux et stockés à des adresses contigües. Pour la conversion, voir la table des codes ci-dessous. L'emplacement des digits (nombres BCD) est situé à partir de DS:[200] et celui des codes ASCII est situé à partir de DS:[300].

HEXA	ASCII
0	30H
1	31H
⋮	⋮
9	39H

Exercice 13: Calcul de la moyenne arithmétique.

Il s'agit d'écrire un programme en langage assembleur 8086 en assembleur 8086 (Organigramme et programme) permettant de calculer la moyenne arithmétique

exprimée aux dixièmes prés (1 chiffre après la virgule) de N octets non signés à partir de l'adresse TAB. Le résultat, quant à lui, sera stocké à partir de l'adresse MEAN. Rappelons à ce titre que l'expression de la moyenne arithmétique est: $M = \frac{(x_0 + x_1 + ... + x_{N-1})}{N}$

Exercice 14: Recherche d'une séquence binaire dans une trame de 16 bits.

Il s'agit d'écrire un programme en langage assembleur 8086 en assembleur 8086 (Organigramme et programme) permettant de rechercher une séquence binaire de longueur fixe de 4 bits dans une trame de 16 bits situé e à partir de l'adresse DS:[200]. A l'initialisation, la séquence binaire doit-être rangée au quartet le moins significatif de l'adresse DS:[202]. Le rang en hexadécimal (0, 1, 2, ..., E ou F) du bit le moins significatif de cette séquence binaire doit-être, quant à lui, rangé au quartet le plus significatif de l'adresse DS:[200].

Exercice 15: Test de primalité.

Soit un nombre entier strictement positif codé sur 16 bits en Hexadécimal et stocké dans à partir l'adresse NOMBRE. Il s'agit de vérifier par une méthode de votre choix si ce nombre est premier. Etablir pour cela l'organigramme et le programme correspondant en mnémoniques 8086 permettant de réaliser ce test de primalité. Si le nombre est premier, il suffira de mettre à '1' le flag carry. Dans le cas contraire, le flag carry est mis à '0'.

Exercice 16: Multiplication d'un nombre multi-octets par 4

Il s'agit d'écrire l'organigramme et le programme correspondant en assembleur 8086 permettant de multiplier un nombre multi-octets par 4. Ce nombre est de longueur 100H. Les octets sont rangés (du moins significatif au plus significatif) à partir de l'adresse DS:[200]. Les octets résultats seront stockés (du moins significatif au plus significatif) à partir de l'adresse DS:[300].

Exercice 17: Calcul de la Taille d'un programme.

Sachant qu'un programme commence à l'adresse CS:IP et se termine par l'instruction INT 03H dont le code machine est CCH, donner la routine qui calcule la taille de ce programme. Dans notre cas, le programme constitue la routine elle-même. On prendra IP=100H. La taille du programme devra être trouvée à partir de l'adresse DS:[200].

Exercice 18: Positionnement du flag T (Trap).

Ecrire un programme permettant de positionner le flag T.

Exercice 19: Transfert de zone mémoire.

Ecrire un organigramme et son programme correspondant en assembleur 8086 qui recopie la zone mémoire située à partir de l'adresse DS:[200] vers la zone mémoire située à partir de l'adresse DS:[202].

Exercice 20: Insertion d'un bloc d'octets dans une zône mémoire.

Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant d'insérer un bloc de M octets situé à partir de l'adresse BLOC à l'intérieur d'une zone mémoire de N octets située à partir de l'adresse ZONE. L'insertion du bloc doit commencer à partir de l'adresse d'offset INSERT. On prendra N=1000H, M=100H et INSERT=200H.

Exercice 21: Inversion d'un bloc mémoire.

Ecrire l'organigramme et le programme correspondant en mnémoniques 8086 qui permet de recopier un bloc mémoire de N octets situé à partir de l'adresse ZONE en l'inversant et en l'insérant dans la même zône mémoire. On prendra N=1000H.

Exercice 22: Puissance de deux entiers (octets).

Ecrire l'organigramme et le programme correspondant permettant de calculer A^B. Les nombres a et b sont situés, respectivement, aux adresses NOMBREA et NOMBREB. Le résultat est rangé à partir de l'adresse RESULTAT.

Exercice 23: Inversion des bits dans un mot de 16 bits.

Etablir l'organigramme et le programme correspondant en assembleur 8086 consistant à inverser totalement l'ordre des bits d'un mot situé à partir de l'adresse DS:[200].

Exercice 24: Somme des produits.

Etablir l'organigramme et le programme qui calcule le produit des éléments d'un tableau de longueur N. Ces éléments sont en fait des octets notés x_i (i=1, 2, …, N).

La somme des produits de ces éléments s'écrit: $\Phi(k) = \sum_{i=1}^{N} x_i x_{i+k}$ où k=0, 1, …. Les

éléments du tableau sont placés à partir de l'adresse TAB. L'indice k est stocké à partir de l'adresse INDICEK et la grandeur $\Phi(k)$ à partir de l'adresse SOMMEP.

Exercice 25: Checksum d'une ROM de 128 K Octets

Utiliser la méthode du checksum qui consiste en l'addition de tous les octets contenus dans toutes les cases mémoires d'une ROM de 128 K Octets située à partir de l'adresse D000:0000 du mapping mémoire d'un PC. Le diagnostic consiste en la comparaison du résultat de l'addition au contenu de la dernière case mémoire. Si ce dernier vaut le résultat de l'addition, la ROM est déclarée 'BONNE'. Sinon, elle est déclarée 'DEFECTUEUSE. Ecrire pour cela l'organigramme et le programme correspondant en mnémoniques 8086 permettant un tel checksum.

Indication: Il ne s'agit pas d'écrire la procédure d'affichage mais de l'appeler par son nom ROM-STATE pour l'affichage de 'ROM DEFECTUEUSE' ou de 'ROM BONNE'.

Exercice 26: Désassemblage d'une séquence de codes machine.

Désassembler la séquence de codes machine 8086 suivante:
C7 07 F7 FF FF 07 F7 FF FF F7 FF E7

Exercice 27: Tri en ordre croissant (décroissant) d'un tableau.

Donner l'algorithme et l'organigramme correspondant au tri en ordre croissant (et décroissant) d'un tableau de N éléments, situé à partir de l'adresse TAB.

Exercice 28: Recherche d'une chaîne de caractères dans un bloc de taille N.

Donner l'organigramme et le programme en assembleur 8086 correspondant à la recherche d'une chaîne de caractères dans un bloc de taille N. Un caractère étant codé sur 8bit. Il s'agit de calculer l'adresse (rang) du premier caractère de la chaine si celle-ci existe et de le mettre dans le registre DI. Par contre, si la chaine n'existe pas, le registre DI doit-être mis à zéro. L'emplacement de la chaine de caractères se trouve à partir de la case DS:[300]. L'emplacement du tableau pouvant contenir la chaine en question est à partir de DS:[200]. On prendra comme exemple celui de la chaîne 'TEC 586'.

Exercice 29: Diagnostic d'une RAM de 64 K Octets.

Ecrire l'organigramme et le programme correspondant en assembleur 8086 permettant de diagnostiquer une mémoire RAM de 64 K Octets située à partir de

l'adresse 2000:0000 du mapping mémoire d'un PC. Pour cela, utiliser la méthode qui consiste en l'écriture de la valeur 00H, suivie d'une lecture de celle-ci. Répéter la même chose pour la valeur FFH et ce pour toutes les cases mémoires de cette RAM. Si au cours d'une lecture, la valeur lue est différente de celle initialement inscrite alors, la RAM est déclarée 'défectueuse'. Auquel cas, une routine appelée RAM-STATE affiche 'RAM DEFECTUEUSE'. Sinon cette même routine affiche 'RAM BONNE'.

Indication: Il ne s'agit pas d'écrire la procédure d'affichage mais de l'appeler par son nom RAM-STATE pour l'affichage de 'RAM DEFECTUEUSE' ou de 'RAM BONNE'.

Exercice 30: Calcul du rang du premier nombre positif dans un tableau.

Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de calculer le rang du premier nombre positif dans un tableau de N Octets situé à partir l'adresse TAB. Si un nombre positif existe alors rangez son rang à partir de l'adresse PNBREP. S'il en existe aucun, alors rangez '0' à l'adresse PNBREP. Le rang du premier nombre positif doit être 1, 2, 3 ... ou N.

Exercice 31: Décrémentation d'un nombre multi-octets.

Ecrire un programme en assembleur 8086 permettant d'incrémenter de 01 un nombre multi-octets situé à partir de l'adresse TAB.

Exercice 32: Les arguments de lignes de commandes sont des caractères ajoutés après le nom d'un fichier *.com ou *.exe lors de son lancement. Pour comprendre ceci, prenons l'exemple de la ligne de commande KEYB. □FR est dit argument de ligne de commande. Le □ étant le blanc inséré entre KEYB et FR. La chaîne de caractères représentant les arguments de ligne de commande est mémorisée à partir de l'adresse SEGPSP:0081H. L'octet situé à l'adresse SEGPSP:0080H contient la longueur de la chaîne ligne de commande (en nombre de caractères, sans compter le caractère ↵ final). La valeur du segment SEGPSP est celle du registre BX après l'appel d'une fonction MS-DOS associé à la ligne de commande en question. On voudrait concevoir un programme KEYB.COM qui réagit aux arguments de ligne de commande selon le tableau suivant:

Ligne de commande	Action
□US	Installation du gestionnaire de clavier Américain
□GR	Installation du gestionnaire de clavier Allemand
□FR	Installation du gestionnaire de clavier Français
Pas d'argument	Affichage du gestionnaire de clavier courant

Donner L'organigramme et le programme, en exploitant le contenu des adresses SEGPSP:0080H et les suivantes.

Exercice 33: Soit le programme en mnémoniques 8086 utilisant une routine récursive. Le code objet du programme appelant est assemblé à partir de la paire CS:IP = 25FO:0100 et celui de la routine récursive est assemblé à partir de la paire CS:IP=25FO:0200. On prendra CS = DS = SS.

1- Quelle est la taille de la pile (en octets) utilisée par ce programme.
2- Quel est le contenu des quatre premiers octets empilés. Représenter la pile et le contenu des quatre premiers octets.
3- Réécrire l'ensemble du programme sachant que la routine récursive est assemblée à partir de la paire CS:IP=4000:0200.

```
PROGRAMME          25FO:0100   MOV  CL, 08
APPELLANT                      CALL 0200
                               INT  3
                               :
ROUTINE            25FO:0200   DEC  CL
RECURSIVE                      JZ   FIN
                               CALL 0200
                         FIN:  RET
```

Exercice 34: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de:

1- Rechercher le rang du premier élément négatif dans un tableau de N=1025D éléments situé à partir de l'adresse TAB. S'il existe un ou plusieurs nombres négatifs, alors ranger le rang du premier trouvé à l'adresse RPNBREN. S'il en existe aucun alors ranger '0' à l'adresse PNBREN.
2- Compter tous les éléments négatifs et de ranger leur nombre à partir de l'adresse NBREN.

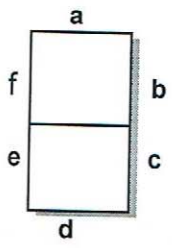
Exercice 35: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de déclarer un nombre multioctets (N octets) de deux rangs à gauche avec un remplissage de '0'. Un nombre binaire multioctets est rangé en mémoire à des adresses contiguës. L'octet le moins significatif étant rangé à l'adresse la plus basse notée TAB. On prendra N=100H.

Exercice 36: Soit à convertir un nombre BCD compris entre 0 et 9 en code 7 segments. Il s'agit pour cela d'établir l'organigramme et le programme correspondant en mnémoniques 8086 permettant de tester et d'effectuer la conversion de ce nombre dont l'emplacement mémoire est noté DIGIT en son

équivalent en code 7 segments dont l'emplacement mémoire est noté C7SEG. Si le nombre situé à l'adresse DIGIT n'est pas compris entre 00 et 09 il faut afficher 'ERROR'.

Indications:

1- Utiliser les correspondances bit-segment suivantes:



Bit	Segment correspondant	Bit	Segment correspondant
B_0	a	B_4	e
B_1	b	B_5	f
B_2	c	B_6	g
B_3	d	B_7	Validation de l'afficheur

2- Un segment est allumé quand le bit correspondant est à '1'. La validation de l'afficheur étant à '0' (anode commune).
3- En vous basant sur la correspondance bit-segment ci-dessus, créer un tableau contenant toutes les configurations de l'afficheur correspondant à tous les nombres BCD compris entre 0 et 9.
4- Tous les nombres doivent être exprimés en Hexadécimal.

Exercice 37: Soient N nombres entiers strictement positifs codés chacun sur 8 bits en Hexadécimal stockés dans un tableau à partir de l'adresse NTAB. Il s'agit de créer à partir de ce tableau, un nouveau tableau commençant à l'adresse PTAB, dans lequel seront stockés tous les nombres premiers. Etablir pour cela l'organigramme et le programme correspondant en mnémoniques 8086 permettant de créer un tel tableau. Pour ce faire, on prendra N = 1024D.

Indication: Tous les nombres doivent être exprimés en Hexadécimal.

Exercice 38: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de convertir un nombre décimal à 5 chiffres codé en binaire (c'est à dire un nombre BCD), en un nombre binaire. Le résultat sera stocké dans deux cases mémoire adjacentes. L'octet le moins significatif du nombre BCD est logé à partir de NBCD et celui du nombre binaire est logé à partir de NBIN. La valeur maximale du nombre BCD étant de 64535.

Exercice 39: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de convertir un nombre binaire en un nombre BCD. Utiliser les mêmes hypothèses que celle de l'exercice précédent.

Exercice 40: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de faire la multiplication binaire de deux nombres de 32 bits. Le multiplicande est stocké à partir de l'adresse MD, le multiplicateur à partir de l'adresse MC et le résultat à partir de l'adresse RT.

Exercice 41: Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de faire la division binaire de deux nombres de 32 bits. Le dividende est stocké à partir de l'adresse DV, le quotient à partir de l'adresse QT et le reste à partir de l'adresse RT.

Exercice 42 : On désire contrôler l'exactitude du contenu d'une zone mémoire de 4 K Octets située à partir de l'adresse MEM. Pour ce faire, on cherche à détecter les erreurs en se basant sur un contrôle de parité verticale. Autrement dit, les bits de même poids (même colonne) et dont le nombre est pair, donnent lieu à un bit de parité égale à '1' qui sera sauvegardé dans un octet situé à l'adresse OCPARITY (Octet parité) à la même colonne. Si au contraire, le nombre de '1' est impair, le bit de parité correspondant est mis à '0 '. Donner l'organigramme ainsi que le programme en assembleur 8086 permettant de calculer l'octet de parité à partir des données de la mémoire à contrôler.

Exercice 43: Donner l'organigramme et le programme correspondant en assembleur 8086 permettant de complémenter à 2 un nombre multi-octets (N octets) situé à partir de l'adresse MBYTE. Le résultat sera, quant à lui, mémorisé à l'adresse CMBYTE. On prendra N= 400H.

Exercice 44: Réaliser l'organigramme et le programme correspondant en assembleur 8086, sous forme de procédure, permettant de faire la traduction code Gray - code Hexadécimal d'un nombre codé sur 4 bits. Etant donné une séquence en code Gray, on rappelle que les bits binaires correspondants sont définis comme suit:

$b_i = g_i$ si lenombre de '1' qui précède ce bit (à gauche de ce bit) est pair
$b_i = \overline{g_i}$ si le nombre de '1' qui précède ce bit (à gauche de ce bit) est impair

Indications:

1- i=0, 1, 2, 3.
2- Commencer par reproduire le bit b_i le plus significatif de la séquence binaire.
3- Un nombre de '1' égal à '0', correspond à un nombre pair.
4- A titre d'exemple, la séquence Gray 1001 représente la séquence binaire 1110.
5- Aussi bien la séquence en code Gray que celle en code Hexadécimal doivent être contenues dans un même registre de 8 bits que vous définirez.

6- Avant d'écrire le programme, il s'agira d'abord de créer en mémoire une table débutant à l'adresse BIN, dans laquelle il faudra placer toutes les séquences binaires correspondant aux séquences Gray 0000, 0001, ..., 1111.

Exercice 45: Il s'agit de concevoir un programme permettant de remplacer les lettres contenues dans un texte écrit en minuscules par les lettres correspondantes selon le schéma de cryptage dit code de César ou de décalage alphabétique. Dans la version numérique de ce schéma de cryptage, on fait d'abord, correspondre à la lettre 'a' le chiffre '0', à la lettre 'b' le chiffre '1', ainsi de suite jusqu'à faire correspondre à la lettre 'z' le chiffre 25. Puis, pour un chiffrement à clé 9 (clé=9), on additionne 9 à chaque nombre m (m=0, 1, ..., 25). Si m+9>25 alors on retranche 26 et on fait correspondre au résultat une lettre de l'alphabet. Pour ce faire, on suppose que le texte avant et après cryptage est mémorisé à partir de la même adresse string DB 'THAT IS ALL FOLKS','$' où $ représente la fin de texte (code ASCII de $ est égal à 24H). La table de cryptage est mémorisée, quant à elle, à partir de l'adresse table DB 61H (' '), 'alphabet latin crypté' où DB 61H représente 97D cases mémoires remplies par des 'space' (code ASCII de space est égal à 20H). Il est à noter que si un symbole ne correspond pas (en dehors) à l'alphabet a b c d e f g h i j k l m n o p q r s t u v w x y z, il n'est pas reconnu par l'opération de cryptage.

1- En vous aidant du schéma de cryptage donné ci-dessus, trouvez manuellement l'alphabet crypté celui qui débute à l'adresse TABLE.
2- En déduire le texte qui crypte celui qui débute à l'adresse string.
3- Donner alors l'organigramme et le programme correspondant en mnémoniques 8086 permettant de crypter automatiquement le texte de la question 2- .

Les codes ASCII de l'alphabet latin en minuscule sont les suivants:

ASCII	a	b	c	d	...	x	y	z
HEXADECIMAL	61	62	63	64	...	78	79	7a
DECIMAL	97	98	99	100	...	120	121	122

Indication: Les pseudo-instructions sont obligatoires pour les quatre exercices.

Exercice 46: Comptage du nombre de caractères dans une chaîne
Donner l'organigramme et le programme correspondant en mnémoniques 8086 permettant de compter le nombre de caractères contenus dans une chaîne terminée par un zéro. Pour ce faire, nous prenons comme exemple celui de l'alphabet latin (a b c ... x y z). Nous supposons que cette chaîne est stockée sous forme ASCII à partir de l'adresse string.

Exercice 47: Commande de la température d'un four.

Nous désirons maintenir constante la température d'un four. Pour cela, nous commandons le système de chauffage par une carte à base de microprocesseur 8086. En supposant que la température de l'air est inférieure à 60°, nous nous proposons de maintenir à l'aide d'un programme la température du four entre 60° et 80°. Le système est tel que la température du four est constamment mesurée à la sortie d'un capteur de température dont l'adresse du port est 7DH. Si celle-ci est inférieure à 60° alors, la mise en marche du chauffage est établie en envoyant un '1' sur le port d'adresse FEH. Au contraire, si celle-ci est supérieure à 80°, la mise à l'arrêt du chauffage est alors établie en envoyant un '0' sur le même port et ainsi de suite.

1- Donner le schéma synoptique de commande de la température du four.
2- Donner l'organigramme et le programme en assembleur 8086 permettant de maintenir la température du four entre 60° et 80°.

Exercice 48: Cryptage d'un texte.

Il s'agit d'écrire un programme en assembleur 8086 permettant de remplacer les lettres minuscules contenues dans un texte par les lettres correspondantes selon le schéma de cryptage suivant: La lettre 'a' est codée (cryptée) par la lettre 'h', la lettre 'b' par la lettre 'i' etc., de telle manière à obtenir une correspondance un à un de l'alphabet latin: a b c d e f g h i j k l m n o p q r s t v u w x y z et de l'alphabet de cryptage: h i j t u v w x y z a b c d k l m n o p r q s e f g. Pour ce faire, nous supposons que le texte avant et après cryptage est mémorisé à partir de la même adresse: STRING DB 'BONNE CHANCE', '$' où '$' (code ASCII de $ est égal à 24H) représente la fin de texte. La table de cryptage est mémorisée, quant à elle, à partir de l'adresse: table DB 61H (' '), 'h i j t u v w x y z a b c d k l m n o p r q s e f g', où DB 61H (' ') représente 97 (en décimal) cases mémoires remplies par des 'space' (code ASCII de space est égal à 20H). Il est à noter que si un symbole ne correspond pas à l'alphabet a b c d e f g h i j k l m n o p q r s t v u w x y z, il n'est pas reconnu pas l'opération de cryptage.

1- En vous aidant du schéma de cryptage donné ci-dessus, trouver manuellement le texte qui crypte celui qui débute à l'adresse STRING.
2- En déduire l'organigramme et le programme correspondant en mnémoniques 8086 permettant de crypter automatiquement le texte de la question 1.

Indications: Le tableau des codes ASCII de l'alphabet latin en minuscule en Annexe 6.

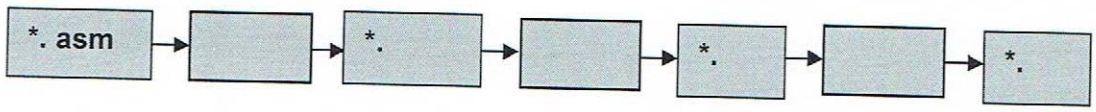
Exercice 49: Soit le programme en mnémoniques 8086

```
CODESEGMENT
            ASSUME    CS:CODE, DS:CODE, SS:CODE
            ORG       0100H
COMPT       EQU       0200H
TAB  DW          200  DUP(.)
START:      MOV       AX,CODE
            MOV       DS,AX
            MOV       SI,OFFSET TAB
            MOV       CX,COMPT
            MOV       AX,0002H
CONT:       CMP       AX,[SI]
            JZ        FIN
            ADD       SI,0002H
            LOOP      CONT
FIN:        NOP
CODE        ENDS
            END START
```

1- Quelle est la fonction de ce programme.
2- Donner le mode d'adressage pour chacune des instructions de ce programme.
3- Optimiser ce programme en utilisant une instruction de manipulation de chaîne.
4- Assembler les deux versions de ce programme. Comparer les nombres d'octets qu'occupent chacune d'elles. on supposera que la directive END possède un code objet égal à 01H et CODE = 1ADFH.

Exercice 50: Pour chacune des questions, répondre brièvement à l'intérieur du cadre réservé à cet effet.

1- Compléter les étapes par lesquelles doit passer n'importe quel fichier dont l'extension est .asm pour aboutir au même fichier dont l'extension est .obj



2- Donner les instructions de sauts conditionnels correspondants aux états des flags suivants:

S=O=0 et Z=0	C=1 ou Z=1	S=O et S=1 ou Z=1	C=1 et Z=0

Instructions	AL(H) Avant	AL (H) Après	AF	CF	ZF	SF	OF	PF
MOV AL,123H								
CMP AL,124H								
AND AL,02H								
TEST AL,02H								
SAR AL,CL								

3- Soit le contenu de la zone mémoire:

0000:0500	9E	0F	CA	00	65	04	70	00
0000:0508	16	00	66	07	65	04	70	00

et soit le programme en assembleur suivant:

CS:IP = 2A16:0100 MOV AX , 0000H (1)
MOV DS,AX (2)
MOV SI,0500H (3)
CALL FAR [SI] (4)

Donner sous forme de Segment:Offset:

le couple CS:IP après l'instruction (1) CS : ☐ IP : ☐
le couple CS:IP après l'instruction (2) CS : ☐ IP : ☐
le couple CS:IP après l'instruction (3) CS : ☐ IP : ☐
le couple CS:IP après l'instruction (4) CS : ☐ IP : ☐

4- Assembler les instructions 8086 suivantes:

CS:0100 DIV BPTR ES:[BP+SI+F100] ☐☐☐☐☐☐

CS:0105 RET 0026 ☐☐☐

5- Désassembler le code machine suivant:

CS:0100 82 3C 00 73 FB 74 14

CS : 100

CS :

CS :

Exercice 51: Sachant qu'avant l'exécution de chacune des instructions suivantes AL= - 5, CL= 2 et AF=CF=ZF=SF=OF=PF=0, compléter le tableau ci-dessous.

Exercice 52: Donner les codes machine des instructions suivantes:

Instructions	Codes machine en Hexadécimal
MOV ES:[SI],SI	
TEST CL, [81]	
SUB [BP+DI],SI	
RCR BPTR [DI+ABCD], CL	
REP MOVSB	

Exercice 53 : Donner pour chacun des fragments l'instruction JMP équivalente.

Fragments de codes	Instruction JMP équivalente
CS:100 Call 120	
CS:100 MOV SI,0120 CS:103 MOV [SI],100 CS:107 LEA AX, [SI] CS:109 CALL AX	
CS: 100 MOV AX,0112 CS: 103 PUSH AX CS: 104 RET	
CS:100 MOV AX, 120 CS:103 SUB SP, 2 CS:106 MOV BP,SP CS:108 MOV [BP], AX CS:10B RET	
CS:100 MOV AX, 15EB CS:103 JMP 101	

Exercice 54: Donner le contenu des registres IP et CL après l'exécution du fragment de codes suivant:

Fragment de codes	IP		CL	
	Avant	Après	Avant	Après
CS:100 MOV CL,CC			AF	
CS:102 INC CL				
CS:104 JMP 101				

Quelles sont alors l'adresse et l'instruction qui sera exécutée après l'instruction JMP 101.

CS :

Exercice 55: Donner le fragment de codes (Au plus 3 instructions) qui permet de charger le registre IP par la valeur 0100.

Fragments de codes	Contenu du registre IP après l'exécution du fragment de codes ci-contre
CS : 010B CS : CS :	

Exercice 56: Le codage de Huffman consiste à coder un texte en transformant un alphabet à longueur fixe en un alphabet à longueur variable. Ce codage nécessite la connaissance de la probabilité d'occurrence de chaque caractère i. e., probabilité d'apparition de chaque caractère dans le texte original).
Afin d'illustrer la technique utilisée pour le calcul du code de chaque caractère (Il s'agira de la construction d'un graphe sous forme d'arbre), nous donnons l'exemple simplifié suivant :

Alphabet	a	e	t	r	S
Probabilité	40%	20%	20%	10%	10%
Code caractère	0	10	111	1111	1100

Le graphe (arbre) est obtenu selon les règles suivantes (Cf. Figure 1):

1- A chaque étape, l'ordre décroissant des probabilités doit être rétabli du haut vers le bas.

2- A chaque étape, un nœud est formé par l'association des deux probabilités les plus basses. Un bit égal à '1' est inscrit en haut du nœud et un '0' est inscrit en bas du nœud.
3- Le code de chaque caractère est obtenu de droite à gauche en parcourant le chemin approprié dans l'arbre.

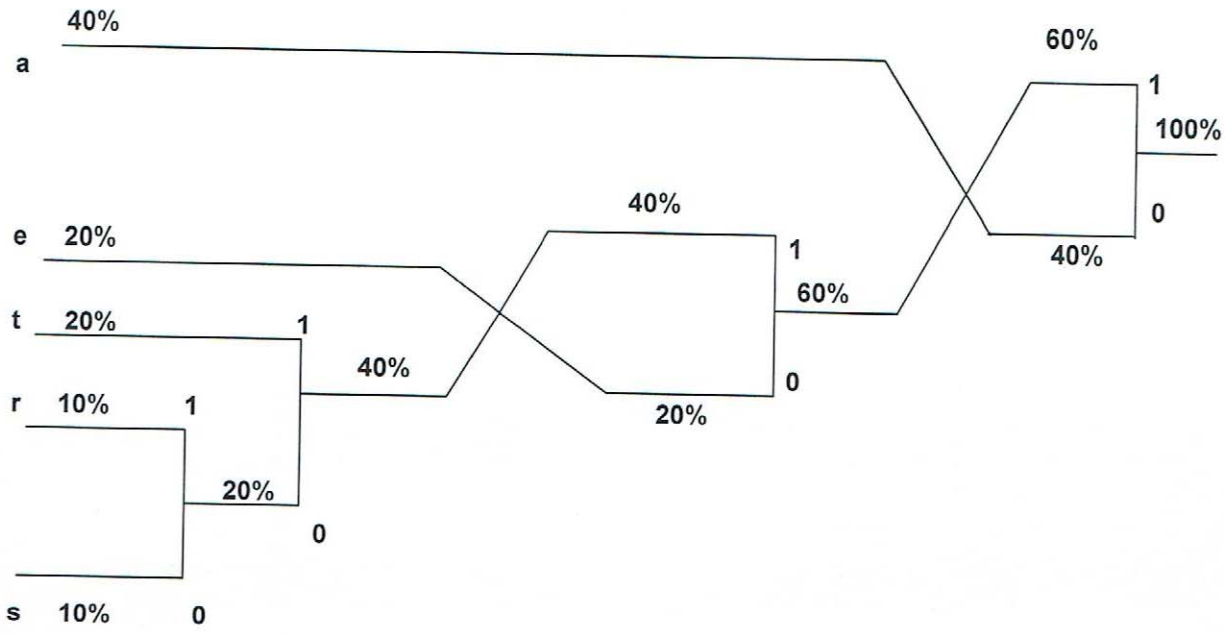


Fig. 1 Arbre représentatif

Faire l'organigramme et le programme correspondant en assembleur 8086 permettant de calculer le code Huffman de chaque caractère d'un alphabet constitué de N caractères, tel que:

Caractère	1er caractère	2ème caractère	...	Nème caractère
Probabilité	P_1	P_2	...	P_N

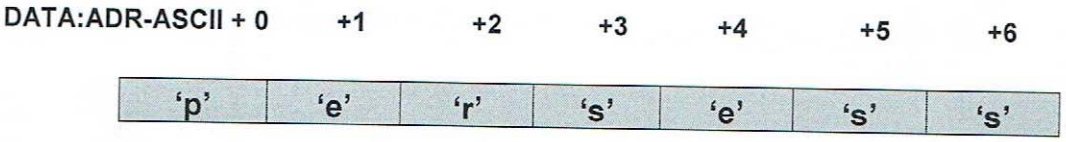
Exercice 57: Il existe un système de codage appelé code de Huffman dans lequel les caractères ont une longueur variable. On veut convertir une zone mémoire de 2 K octets représentée en ASCII en une autre zone mémoire représentée en code Huffman. Le contenu de la zone mémoire origine que l'on désire convertir est formée exclusivement des 5 caractères 's', 'e', 'p', 'a' et 'r', codage ASCII de longueur fixe. La transformation se fait conformément au tableau ci-après:

Caractère en code ASCII Longueur fixe (8 bits)	Code Huffman Longueur variable
's'	0
'e'	11
'p'	101
'a'	1001
'r'	1000

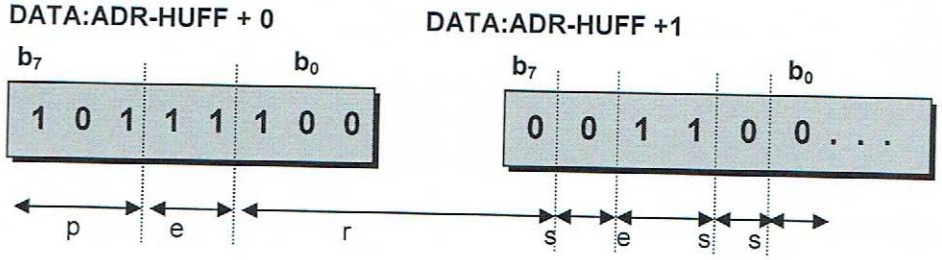
L'adresse de début de la mémoire origine est notée DATA :ADR-ASCII. L'adresse de début de la mémoire destination est notée DATA :ADR-HUFF

L'exemple suivant aide à bien assimiler cette technique de codage.

Zone mémoire initiale



Zone mémoire codée



1- Donner l'organigramme et le programme correspondant en assembleur 8086 permettant de réaliser la conversion du code ASCII de la zone mémoire originale en code Huffman de la zone destination.
2- La zone mémoire destination a automatiquement une taille inférieure à la zone mémoire origine. On désigne par taux de compression le rapport entre la taille de la mémoire destination et la taille de la mémoire origine exprimé en pourcentage (%).
3- Etablir l'organigramme et le programme permettant de calculer la partie entière du taux de compression.

Exercice 58: Programme utilisant une routine récursive.

Soit le programme en mnémoniques 8086 utilisant une routine récursive. Le code objet du programme appelant est assemblé à partir de CS:IP = 4000H:0100H et celui de la routine récursive est assemblé à partir de CS:IP=4000H:0400H. Prendre CS=DS=SS.

4- Quelle est la taille en hexadécimal de la pile (en octets) utilisée par ce programme.
5- En déduire la définition du segment pile et donner la valeur de départ de SP. Prendre un segment d'alignement PARA, de type PUBLIC et de classe STACK.
6- Réécrire l'ensemble du programme dans le cas où la routine récursive est assemblée à partir de CS:IP=6000H:0600H.

```
;PROGRAMME
CODE    SEGMENT        PARA    PUBLIC   'CODE'
        ORG 0100H
        ASSUME CS:CODE, DS:CODE, SS:CODE

        SEGMENT PILE À DEFINIR ET À FERMER.

RECUR  PROC  NEAR
       ORG 0400H
          DEC  CL
          JZ   FIN
          CALL  0400H
FIN:      RET
       ENDP
```

```
START:  MOV AX,CODE
        MOV DS,AX
        MOV SS,AX
        MOV SP, ?H
        MOV CL,20H
        CALL RECURSIVE
        INT 03H
CODE    ENDS
        END START
```

Exercice 59: Tri en ordre décroissant de nombres réels

On considère un ensemble de nombres réels représentés en virgule flottante selon le format mantisse en complément à 2 sur 8 bits et exposant en complément à 2 sur 8 bits. Des paires d'octets mantisse-exposant représentant chacune un nombre réel sont stockées en mémoire RAM dans un PC à partir de l'adresse SOURCE, de telle manière qu'une zone mémoire équivalent à N nombres réels forme l'ensemble des données qui seront traitées dans ce problème. On précise que les mantisses ne sont pas forcément normalisées, et que les exposants peuvent avoir n'importe quelle valeur comprise entre –128 et +127. On désire classer ces données en ordre décroissant. Pour cela il faudra procéder au schéma proposé selon les étapes suivantes :

1- En premier lieu, élaborer une procédure de comparaison entre deux nombres réels répondant au format spécifié plus haut, en vue d'établir une relation d'ordre <, =, ou >. Présenter l'organigramme, ainsi que la procédure en langage assembleur 8086 permettant de réaliser cette comparaison. On insiste pour que la procédure ainsi développée obéisse à une stratégie de rapidité d'exécution.
2- Le programme principal ou l'algorithme de tri proprement dit, devra reprendre la technique classique du tri à bulles, Comparaisons récursives avec échange de données si l'ordre décroissant n'est pas observé. A partir de ce programme, il sera fait appel à la procédure de comparaison développée auparavant par une simple instruction CALL. Etablir l'organigramme, ainsi que le programme principal en langage assembleur 8086, dans lequel les règles syntaxiques et structurelles prévues pour l'utilisation des pseudo-instructions et des directives d'assemblage seront respectées en toute rigueur.
3- A supposer que vous ayez réussi le tri d'un ensemble de nombres réels selon le schéma global suggéré dans cet énoncé, auriez vous une meilleure approche à proposer comme solution, en vue de favoriser au choix l'un des 3 critères de performance fondamentaux dans l'art de la programmation :

- Vitesse d'exécution
- Réduction de la taille de la mémoire programme
- Précision des résultats (notamment dans la procédure de comparaison)

Nous rappelons qu'en théorie ces trois critères sont contradictoires, il s'agira d'abord de bien choisir lequel d'entre eux vous semble susceptible d'apporter une amélioration de performance pour l'ensemble du programme.

TRAVAUX DIRIGES N°6

Exercice 1: Le programme suivant représente la procédure de gestion d'une interruption BIOS.

```
PUSH DS
PUSH BX
MOV BX, 0040H
MOV DS, BX
OR AH, AH
JZ READ
DEC AH
JZ SET
JMP FIN
```

```
READ:  MOV CX, [006EH]
       MOV DX, [006CH]
       JMP FIN
SET:   MOV [006CH], DX
       MOV [006EH], CX
FIN:   POP BX
       POP DS
       IRET
```

1- Donner les paramètres d'appel (entrée) de cette interruption .
2- Donner les paramètres de retour (sortie) de cette interruption .
3- Etes-vous en mesure d'attribuer une signification à ces paramètres .
4- Si oui, préciser la valeur du paramètre qui définit chaque fonction réalisée par cette interruption.
5- Donner le numéro de cette interruption.

Exercice 2: Pour chacune des questions, répondre brièvement à l'intérieur du cadre réservé à cet effet.

1- Quels sont, dans l'ordre, les registres empilés et désempilés automatiquement, respectivement lors de l'appel et du retour d'une interruption (Instruction INT N°)

Appel

Retour

2- Quelles sont les cinq premières interruptions contenues dans la table des interruptions du 8086. Donner pour chacune d'elle le type (N°) et la fonction.

Type (N°)	Signification

3- Soit l'interruption INT 80 H dont le sous programme associé commence à l'adresse 2D00:0100. A quel endroit de la mémoire se trouve cette adresse . Donner le contenu de cette zone mémoire.

Segment = [ ] : Offset = [ ]

Exercice 3: Soit un PIC disposant des adresses suivantes : @OCW_1= 21H et @OCW_2 =20H, où @ désigne adresse.

1- Quelles sont les fonctions principales du PIC (Priority Interrupt Controller) 8259.
2- Quelles sont, sous forme d'organigramme, les différentes étapes qui permettent l'initialisation du PIC 8259.
3- Donner la séquence d'instructions qui permet de valider l'interruption sur la broche IRQ3 sans pour autant toucher autres interruptions sur les broches IRQi i=0, 1, 2, 4, 5, 6 et 7 .
4- Donner la séquence d'instructions qui permet d'envoyer un NSEOI à ce PIC.
5- Donner la séquence d'instructions qui permet de lire le registre IRR de ce PIC.
6- Donner la séquence d'instructions qui permet de lire le registre ISR de ce PIC.
7- Sachant qu'un PIC 8259 est dans le mode SMM, dire si oui ou non, le bit dans ISR dont le bit correspondant est masqué dans IMR peut être remis à zéro par l'envoie d'un NSEOI.
8- Sachant qu'un PIC 8259 maître a son bit AEOI=1 de ICW4, dire si oui ou non, le mode SFNM est recommandé dans ce cas.
9- Sachant que ISR=A0H. Quel est l'ordre de priorité des broches IRQ_i i=0,1, 2, 3, 4, 5, 6, 7 après que l'interruption la plus prioritaire ai été servie et au moment de l'établissement d'une rotation automatique à travers OCW_2.

10- Soient deux PIC, comptant au total 16 broches d'interruption (IRQ_0-IRQ_{15}), cascadés à travers la broche IRQ2. Si nous masquons cette broche IRQ2, dire si oui ou non, les broches IRQ8 à IRQ15 sont automatiquement masquées.

Exercice 4: Le PC-XT renferme un seul PIC (Priority Interrupt Controller) de type 8259A de la famille Intel qui n'est pas sur le bus local. Il est noté PIC et est relié à sept (07) périphériques à préciser. Nous supposons les hypothèses suivantes : Les broches IRQ_i du PIC travaillent sur des niveaux hauts stables. A l'initialisation toutes les interruptions sont validées. La fin d'une interruption se fait automatiquement. Il s'agit de:

1- Donner le schéma du PIC dans cette structure. Préciser toutes les liaisons de celui-ci avec le 8086 et ses périphériques. Préciser le circuit qui assure l'ouverture et la fermeture des buffers bidirectionnels (8286/87) aussi bien par le PIC que par le 8086. Nous supposerons, dans ce cas, que le 8086 est en mode minimum.
2- En fonction des registres ICW_{1-4}, le PIC effectue, en général, un certain nombre de tests qui lui permettent de s'initialiser correctement. Donner sous forme d'organigramme ces différents tests.
3- Quels sont les contenus des registres (ICW_{1-4} et OCW_{1-3}) utilisés lors de l'initialisation du PIC. Justifier les états 'don't care' des bits non mentionnés dans les énoncés.
4- Dans le cas où nous voudrions cascader un autre PIC au PIC existant, quels seraient les signaux et les registres (bits) du PIC maître qui doivent faire l'objet d'une modification.

Exercice 5: Le PC-AT renferme deux PIC de type 8259A de la famille INTEL. Ils sont montés en Maître/Esclave. Le maître est relié à huit (08) périphériques, dont un PIC esclave, connecté à travers la broche IRQ_2. A cet esclave, sont reliés trois (03) périphériques seulement à travers les broches IRQ_8, IRQ_{13} et IRQ_{14}. Les deux PIC sont sur le bus ISA multi-maîtres. Nous travaillons avec les hypothèses suivantes. Les vecteurs d'interruptions du maître débutent à partir de 08H et ceux de l'esclave débutent à partir de 70H. Le PIC maître n'est pas programmé en SFNM. A l'initialisation, à l'exception des broches auxquelles aucun périphérique n'est connecté, toutes les interruptions sont validées. La fin d'une interruption se fait automatiquement à la fin de la deuxième impulsion INTA pour les deux PIC. Il s'agit de:

1- Donner le circuit qui réalise l'ouverture et la fermeture des buffers bidirectionnels (8286/87) aussi bien par les deux PIC que par le 8086.
2- Donner le contenu des registres ICW_1-ICW_4 et IMR des deux PIC à l'initialisation.

Exercice 6: Le PC-AT renferme deux PIC (Priority Interrupt Controller) de type 8259A de la famille INTEL. Ils sont montés en Maître/Esclave. Le maître est relié à huit (08) périphériques, dont un PIC esclave, connecté à travers la broche IRQ_2. A cet esclave, sont reliés trois (03) périphériques seulement. Les deux PIC ne sont pas sur le bus local. Nous travaillons avec les hypothèses suivantes. Les broches IRQ_i des deux PIC travaillent sur des niveaux hauts stables. Les vecteurs d'interruptions du maître débutent à partir de 08H et ceux de l'esclave débutent à partir de 70H. Le PIC maître n'est pas programmé en SFNM. A l'initialisation, toutes les interruptions sont validées. La fin d'une interruption se fait automatiquement à la fin de la deuxième impulsion $\overline{INTA}$ pour les deux PIC. Il s'agit de:

1- Donner le schéma des deux PIC dans cette structure. Donner toutes leurs liaisons avec le CPU et avec les périphériques. Préciser le circuit qui réalise l'ouverture et la fermeture des buffers bidirectionnels (8286/87) aussi bien par les deux PIC que par le CPU.
2- Donner sous forme d'organigramme le programme d'initialisation des registres des deux PIC. Préciser les bits concernés par une telle initialisation.
3- Donner les étapes qui permettent de lire les registres ISR et IRR.
4- Pouvez-vous imaginer la mise en cascade de plusieurs PIC au PIC maître à travers la même broche IRQ_2. Si oui, comment. Sinon, pourquoi.

Exercice 7: Il s'agit de concevoir un programme qui permet l'affichage binaire d'une chaine de caractères saisie au clavier. Une chaine de caractère située à l'adresse 'buffer' représente votre variable d'entrée. La chaine est constituée uniquement des codes ASCII correspondant aux caractères à signification numérique. C'est-à-dire les nombres 30H à 39H correspondent respectivement aux chiffres '0' à '9'. La longueur totale de cette chaine est la même pour tous les cas de figure, et égale à dix (10) caractères ASCII. La variable de sortie doit correspondre à la représentation binaire de ces nombres.
Votre programme doit être composé d'une procédure de saisie de données ainsi qu'une procédure d'affichage de la valeur en binaire. Pour ce faire:

1- Utiliser INT 16H du BIOS pour saisir par le clavier les 10 caractères numériques représentant la variable d'entrée.
2- Utiliser INT 10H du BIOS pour afficher sur écran la représentation binaire des '1' et '0'. L'affichage doit s'effectuer de gauche à droite en commençant par l'octet le plus significatif.

Exercice 8: On dispose d'une interface PC pour piloter une lampe UV dont la durée d'allumage doit être contrôlée de manière précise. La valeur FFH envoyée sur le port 378H allume la lampe, tandis que FEH sur ce même port provoque son extinction. Ecrire un programme en assembleur 8086 permettant d'allumer cette

lampe et de la maintenir allumée pendant 14 secondes. Pour cela, utiliser l'interruption INT 08H pour réaliser une telle temporisation.

Exercice 9: Ecrire un programme en assembleur 8086 utilisant INT 21H du MS-DOS pour afficher la chaîne 'TEC 586' sur écran à l'enduit du curseur.

Exercice 10: Ecrire un programme en assembleur 8086 utilisant INT 10H du BIOS pour afficher la chaîne 'TEC 586' sur écran à l'endroit du curseur.

Exercice 11: Ecrire un programme en assembleur 8086 utilisant INT 13 H du BIOS pour transférer le contenu du secteur 1, piste 0, face 1, de la disquette présente dans le lecteur B, vers le tampon de mémoire commençant à l'adresse ES:0200H.

Exercice 12: On dispose d'un interface PC pour piloter un moteur dont la durée de marche doit être contrôlée de manière précise. La valeur 01H envoyée sur le port 378H met en marche le moteur, tandis que 00H sur ce même port provoque sa mise à l'arrêt. Sachant que le PIC (8259), supporte sur sa broche IRQ_0 une horloge d'une fréquence de 18.2 Hz, et sachant que l'interruption matérielle, INT8, associée à cette broche, provoque l'incrémentation du mot situé à l'adresse 0040:006C toutes les 1/18.2 secondes, écrire un programme en assembleur 8086 qui permet de mettre en marche ce moteur et doit le maintenir ainsi pendant une (01) heure et le mettre à l'arrêt pendant le reste du temps.

Exercice 13: Ecrire un programme en assembleur 8086 permettant d'effectuer l'affichage binaire de l'octet contenu dans la case DS:0200. L'affichage doit s'effectuer de gauche à droite en commençant par le bit le plus significatif.
Exemple: DS:[0200]=5F → affichage de 01011111

Indications:

1- On utilisera INT 10 du BIOS.
2- La correspondance Hexa → ASCII est:
0 → 30H
1 → 31H

Exercice 14: Ecrire un programme en assembleur 8086 qui imprime votre nom dès que vous appuyer sur la touche 'P' en utilisant les interruptions BIOS INT 16H (clavier) et INT 17H (imprimante). Ce programme permettra:

3- de lire le clavier au moyen de INT 16H et vérifier la réception de la lettre 'P'.
4- de vérifier l'état de l'imprimante et imprimer votre nom au moyen de INT 17H.

Exercice 15: Tout réseau informatique identifie les PC qui lui sont connectés par une adresse IP (Internet Protocol), définie par le format BCD de type (X.Y.Z.W). Les champs X, Y, Z et W sont codés en BCD sur 3 digits chacun, pouvant varier de 0.0.0.0 à 255.255.255.255. Dans ce qui suit, nous nous proposons d'écrire la procédure de saisie et d'impression sur écran de ces champs, stockés à partir de l'adresse DS:200 H. Pour cela:

1- Donner le contenu des octets saisis au clavier à partir de DS:200 H pour le cas de l'adresse IP 209.191.192.119. Les points qui séparent les champs X, Y, Z et W sont considérés comme virtuels.
2- Etablir l'organigramme permettant de saisir au clavier, une adresse IP quelconque en BCD à l'aide de l'interruption INT 16H du BIOS et de l'imprimer sur écran à l'aide de l'interruption INT 10H. Nous supposons que la conversion ASCII-BCD d'un caractère se fait à l'aide d'une routine notée ASCIIBCD que nous ne vous demandons pas d'écrire mais tout simplement de l'appeler à chaque fois que cela est nécessaire.
3- Traduire cet organigramme sous forme de programme au moyen de mnémoniques et pseudo instructions du 8086.

Exercice 16: Donner l'organigramme d'une procédure permettant, lors de l'exécution d'un programme quelconque, de sauvegarder sur disque dur l'image présente sur écran, si n'importe quelle touche est enfoncée (capture d'image).

Exercice 17: Concevoir un programme qui imprime le message 'TEST D'IMPRESSION' dès que vous appuyer sur la touche 'I'. Votre programme doit être en mesure de détecter l'adresse du port CENTRONICS, d'attendre la réception du 'scan code' de la touche 'I' sur le port du PPI (ligne du clavier) et de réaliser la vérification du port CENTRONICS et l'envoie des caractères de contrôle et les noms des membres du trinôme vers l'imprimante.

Exercice 18: Ecrire un programme en assembleur 8086 permettant de classer par ordre alphabétique des chaînes de caractères de longueurs variables. On considère pour cela les hypothèses suivantes:

1- Chaque chaîne ne peut dépasser une longueur de 16 caractères.
2- Chaque chaîne est formée uniquement de caractères alphabétiques majuscules.
3- La fin d'une chaîne est détectée par un caractère de fin de ligne (Carriage Return, ASCII=0DH).

Prévoir une partie du programme qui interrompt le traitement (classification) et qui affiche le message "Données incohérentes" si l'une des chaînes contient des

caractères numériques, minuscules ou spéciaux ou l'une d'elles dépasse la longueur de champs autorisée.

Indications:

1- Ecrire votre programme de façon structurée (utilisation des procédures).
2- Prendre le cas de dix chaînes de caractères stockés respectivement à partir de: DS:[200], DS:[210], DS:[220]....
3- Utiliser le service 9 de l'interruption 21H du MS-DOS pour l'affichage des chaînes de caractères. L'interruption 21H (service 09 permet d'afficher la chaîne située à partir de DS:DX, jusqu'à la rencontre d'un caractère $. Ces paramètres d'entrées sont DS, DX et AH (N° de service).

Exercice 19: Il vous est demandé d'écrire votre programme de gestion de l'interruption 8 (déclenchée par hardware toutes les 1/18.2 sec) qui consiste à afficher en permanence l'heure sous le format HH :MM :SS :NN en bas de l'écran.

Indications:

1- Utiliser quatre variables HH, MM, SS et NN de type octet.
2-
   NN: Compteur modulo 18, incrémenté à chaque occurrence de INT 08 H.
   SS : Compteur modulo 60, incrémenté lorsque NN passe à 0.
   MM : Compteur modulo 60, incrémenté lorsque SS passe à 0.
   HH : Compteur modulo 24, incrémenté lorsque MM passe à 0.
3- Adopter, de préférence, le format BCD pour faciliter l'affichage par l'INT 10H (l'affichage se fera par quartet). On rappelle que le code ASCII de 0 est 30H, 1 est 31H,.....9 est 39H et ' :' est 3AH.
4- HH:MM:SS:NN sera affichée à la ligne 24, colonne 64 sans affecter le contenu du reste de l'écran.

Exercice 20: Il s'agit de concevoir un programme de conversion ASCII-Binaire dont l'utilité est la gestion de nombres réels dans un contexte PC. Une chaine de caractère située à l'adresse 'buffer' représente votre variable d'entrée. La chaine est constituée uniquement des codes ASCII correspondant aux caractères à signification numérique décimale. C'est-à-dire les nombres 30H à 39H correspondent respectivement aux chiffres '0' à '9' et 2E correspond au point. La longueur totale de cette chaine est la même pour tous les cas de figure, et égale à dix (10) caractères ASCII, y compris le point. Il est important de noter que ce dernier, en vue d'une exploitation la plus générale de votre programme, peut ne pas figurer dans la variable d'entrée. Exemples: 456.767819, 2593411812, .236543210, 0.12347890, 123456789. etc.

La variable de sortie doit correspondre à la représentation binaire des nombres réels dans le format à virgule fixe dont la taille (nombre de bits) reste à définir.
Votre programme doit être composé d'une procédure de saisie de données, une procédure de conversion ASCII-Binaire ainsi qu'une procédure d'affichage de la valeur convertie. Pour ce faire:

1- Utiliser INT 16H du BIOS pour saisir par le clavier les 10 caractères décimaux représentant la variable d'entrée.
2- La conversion réalisée, utiliser INT 10H du BIOS pour afficher sur écran la représentation réelle binaire. Une conversion ASCII-Binaire répondant uniquement au besoin d'afficher les '1' et '0' au moyen de INT 10H doit être prévue dans cette étape.

Exercice 21: Economiseur d'écran

Considérons les hypothèses et informations suivantes:

1- Lorsqu'une touche du clavier est enfoncée, le BIOS lit le code de la touche (Scan Code) et détermine son code ASCII associé. Ces deux octets sont mémorisés dans un buffer de 15 mots situés aux adresses 40H:1E – 40H:3D appelé buffer clavier. Deux pointeurs gèrent ce buffer. La variable HB (Head Buffer) mémorisée à l'adresse 40H:1A pointe le premier mot du buffer et la variable TB (Tail Buffer) mémorisée à l'adresse 40H:1C, pointe le dernier mot du buffer. Tel qu'il est illustré par la Figure 1, le buffer clavier possède une structure cyclique selon laquelle TB est incrémenté à chaque nouvelle touche entrée tandis que HB est incrémenté chaque fois qu'un caractère est traité par le BIOS. Si HB ou TB arrive à la fin du buffer (40H:3D), ils se repositionnent au début (40H:1E).

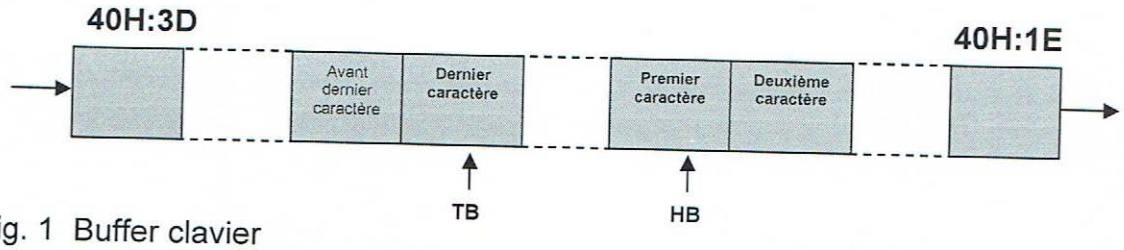


Fig. 1 Buffer clavier

Les deux variables HB et TB sont égales uniquement dans les deux cas suivants:

Buffer plein. Dans ce cas, aucun caractère n'a été traité. Par conséquent, émission d'un signal sonore et tout caractère en provenance du clavier est ignoré.
Buffer vide. Dans ce cas, tous les caractères ont été traités et il n'y a aucun caractère en provenance du clavier.

2- L'interruption BIOS INT 1CH est appelée automatiquement par l'interruption matérielle INT 08H. Elle est mise à la disposition de l'utilisateur de telle manière qu'il puisse programmer une action qui s'exécute toutes les 1/18.2^{ème} de seconde.
3- La mémoire vidéo du PC est organisée en pages, correspondant chacune au contenu d'une image écran. A chaque instant, il y a une seule page (page active) qui est affichée. Les autres pages sont présentes dans la mémoire sans qu'elles soient visibles à l'écran. Si nous voulons visualiser une autre page que celle qui est sur l'écran, il suffit de la choisir comme page active.

Quand l'INT 10H est appelée avec le registre AH=0FH, elle retourne le numéro de la page active dans le registre BH. Par contre, quand l'INT 10H est appelée avec AH=05H, nous pouvons sélectionner une nouvelle page active dont le numéro, compris entre 0 et 7, doit figurer dans le registre AL.

Au vu de ces hypothèses et informations, il vous est demandé de concevoir l'organigramme correspondant à la méthodologie d'approche d'un ECONOMISEUR D'ECRAN et le programme associé tout en respectant l'orthodoxie de l'écriture en assembleur 8086. Ce programme, géré par l'INT 1CH, fonctionne selon le schéma suivant:

1- Si aucune touche du clavier n'a été enfoncée durant 5 minutes, remplacer la page vidéo active par une autre laquelle est considérée comme écran de veille.
2- Si une touche du clavier est enfoncée pendant l'écran de veille, restituer la page active précédente.

Exercice 22: Editeur de page écran

Dans ce programme, nous voulons réaliser un éditeur de page écran en utilisant la RAM vidéo.

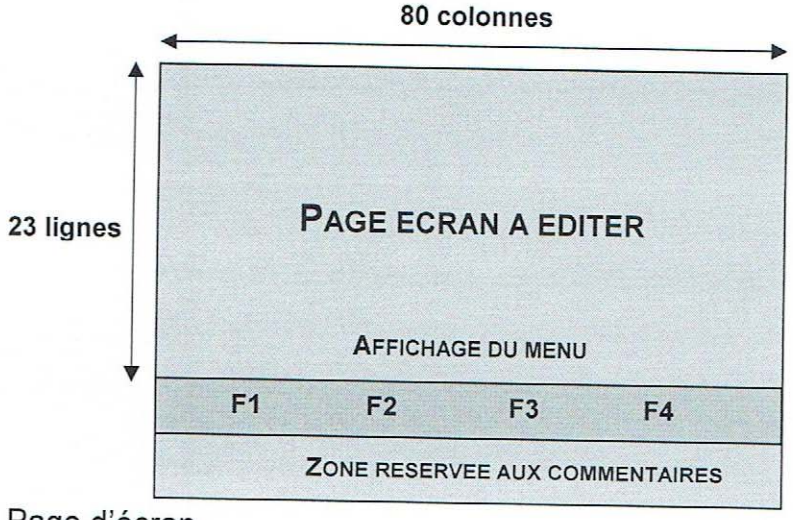


Fig. 2 Page d'écran

Données	Adresses
1er caractère	B800:0000
Attribut	B800:0001
2ème caractère	B800:0002
Attribut	B800:0003
. . .	. . .
2000ème caractère	B800:0F9E
Attribut	B300:0F9F

La page écran occupe 80*25=2000 caractères, ce qui correspond à 2*2000=4000 octets dans la RAM vidéo.
L'éditeur de page dispose d'un menu dont le fonctionnement est contrôlé par la touche de fonction appuyée sur le clavier tel que :

F1: Sauvegarde le contenu de la page écran sur le disque.
F2: Chargement d'un fichier vers l'écran.
F3: Impression du contenu de la page écran.
F4: Sortie de l'éditeur, $.
Les caractères édités sont mémorisés dans la RAM vidéo. Lorsque les touches de déplacement du curseur sont enfoncées. Le curseur sera positionné automatiquement suivant la touche ($\rightarrow, \leftarrow, \uparrow, ou \downarrow$). Pour une meilleure utilisation de l'éditeur, les deux dernières lignes de l'écran sont réservées pour l'affichage du menu et les commentaires.

Les commentaires sont mémorisés aux adresses suivantes:

DS:[0223] « F1 F2 F3 F4 »
DS: [0250] « Donner nom de fichier à charger $ »
DS:[02A0] « Donner nom de fichier à sauvegarder $ »

Ecrire l'organigramme et le programme correspondant en mnémoniques 8086 qui commence par effacer l'écran, afficher le menu de l'éditeur et positionner le curseur en haut de l'écran, A partir de la, le programme principal attend un caractère à partir du clavier s'il s'agit d'une touche de fonction (F1, F2, F3), le sous programme correspondant sera exécuté. Si au contraire, c'est un caractère, il sera affiché sur l'écran.

Dans le cas d'un appuie sur les touches de direction, la position du curseur est lue, les registres DH et DL sont incrémentés ou décrémentés selon le sens de la flèches ($\rightarrow, \leftarrow, \uparrow, ou \downarrow$) et le curseur sera positionné. La nouvelle position du curseur est celle qui correspond aux nouveaux contenus de DH et DL (position verticale et horizontale).
Lorsque la touche F4 est enfoncée, l'écran est effacé, le curseur est positionné en haut de l'écran et le programme prend fin.

TRAVAUX DIRIGES N°7

Exercice 1: On désire effectuer la gestion de feux tricolores d'un croisement à l'aide d'une carte à microprocesseur de type INTEL 8086. Le carrefour est composé d'une voie principale et d'une voie secondaire (Figure 1). Le fait de choisir cette solution, plus onéreuse pour une telle application, permet de changer à volonté les différentes temporisations des feux et d'assurer ainsi une certaine fluidité de la circulation.

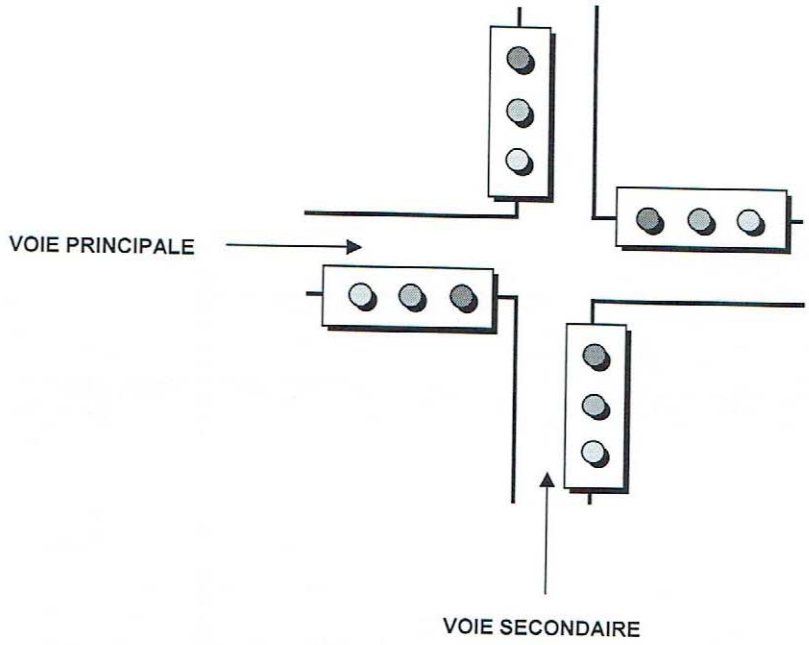
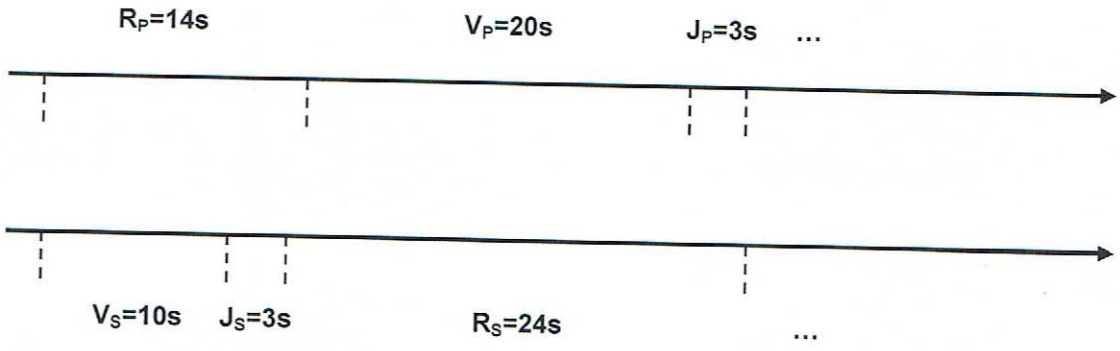


Fig. 1 Schématisation d'un feu tricolores



R_P: Feu rouge de la voie principale.
V_P: Feu vert de la voie principale.
J_P: Feu jaune de la voie principale.
J_S: Feu jaune de la voie secondaire.
R_S: Feu rouge de la voie secondaire.
V_S: Feu vert de la voie secondaire.

Fig. 2 Chronogramme de gestion des feux tricolores

Un exemple de fonctionnement de ces feux tricolores repose sur le chronogramme de la Figure 2. Il vous est demandé de réaliser:

1- La configuration et initialisation du PPI.
2- Le sous programme correspondant au fonctionnement jour.
3- Le sous programme correspondant au fonctionnement nuit.
4- Le sous programme d'interruption activé dès que l'interrupteur passe de '1' à '0', processus de commutation jours/nuit et inversement.
5- Le programme principal pour le fonctionnement global du système.

Exercice 2: Une imprimante de 60 caractères est associé au port A d'un PPI, programmé en mode M_1. Les lignes de dialogues sont $\overline{ACK}$, $\overline{OBF}$, et INTR. Chaque caractère est représenté en code ASCII par un octet. La ligne à imprimer est stockée à partir de l'adresse DS:0300H (CS = DS = SS = 1ADFH).

1- Donner le schéma synoptique d'une telle application, en précisant les lignes de dialogues utilisées.
2- Elaborer l'organigramme et le programme correspondant en assembleur 8086 permettant de réaliser le transfert de données du 8086 vers l'imprimante sous contrôle d'interruption. Le contrôle se compose de deux parties: Procédure d'initialisation du PPI et procédure d'interruption.

Indication: Les registres du port A ont les adresses suivantes: RDA = 00H et RC = 03H. Le vecteur d'interruption associé au port A est D4 et l'offset de début de la routine d'interruption est 01FFH. Le segment étant celui du programme appelant, débutant à CS = 0100H.

Exercice 3: Les micro-ordinateurs compatibles PC possèdent à l'arrière de leur unité centrale un port parallèle dit CENTRONICS. Dans notre application, nous l'utilisons pour y connecter une imprimante parallèle de 8 K octets caractères, comme le montre la figure2. Le texte à imprimer est stocké en code ASCII en mémoire à partir de l'adresse 1ADF:0200.

Exercice 4: On désire commander deux afficheurs du type 7 segments (comme le montre la Figure 3) via un PPI. Cette application permet notamment d'utiliser plusieurs modes d'affichage et de changer à volonté les temporisations prévues à cet effet. Dans votre exemple d'application, on se propose de réaliser deux modes de fonctionnement pour l'affichage de deux caractères: Fonctionnement clignotant: Dans ce cas, les deux afficheurs sont éteints pendant 0.5 sec et allumés pendant 0.5 sec et fonctionnement stable: Dans ce cas, les deux afficheurs sont allumés continuellement.

1- En vous inspirant de la figure, donner le schéma synoptique d'une telle application en précisant toutes les lignes de dialogues, le mode de fonctionnement du PPI ainsi que sa configuration interne.
2- Quelles sont en mnémoniques 8086 les différentes étapes d'initialisation des registres 8086 et ceux du PPI.
3- Elaborer l'organigramme et le programme correspondant en assembleur 8086 permettant de réaliser les deux fonctionnements suscités. Les registres du groupe A ont les adresses suivantes: RDA = 300H, RC = 303H. le vecteur d'interruption associé au groupe A est AAH, l'offset de début de la routine d'interruption est 02EEH et le segment est 4000H. Les segments du programme appelant étant CS = DS = 5000H. Il est recommandé d'utiliser l'interruption INT 08H pour réaliser la temporisation de 0,5 sec.

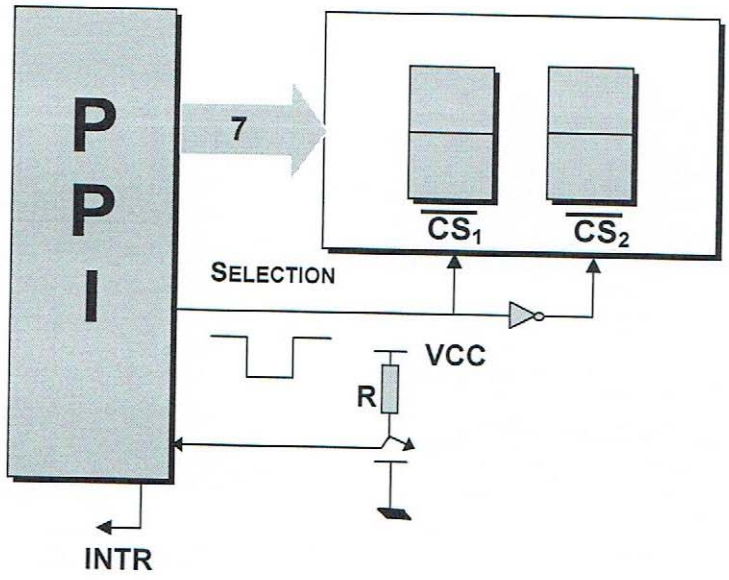


Fig.3 Commande de deux afficheurs par un PPI

Exercice 5: Comme cela est illustré en Figure 4, on dispose d'un KIT d'évaluation constitué d'une carte CPU 8086 (8MHz), de périphériques standards (clavier, barrette d'affichage) et d'interfaces d'E/S parallèle et sérielle. L'ensemble est géré par un moniteur offrant un ensemble de fonctionnalités classiques: accès aux registres, accès aux mémoires, exécution, fonction pas à pas, assembleur/désassembleur, éditeur ligne par ligne et autres utilitaires. On désire utiliser l'interface parallèle (PPI 8255A) de ce système pour réaliser l'acquisition de données provenant d'un convertisseur analogique/numérique à 8 bits (ADC 0804). Le rôle du CAN est de convertir sur 8 bits la tension analogique présente sur son entrée Vin(+), cette tension qui représente une grandeur physique (vitesse de rotation d'une machine) varie entre 0 et 5V; elle doit être convertie et acquise toute les 0.1 seconde (fréquence d'échantillonnage de 10 Hz fournie par une horloge externe). La donnée acquise doit être stockée en mémoire à partir de l'adresse

DS:D-Buffer; ce buffer possède une taille de 4 octets et doit contenir, à chaque instant, les 4 données les plus récentes.

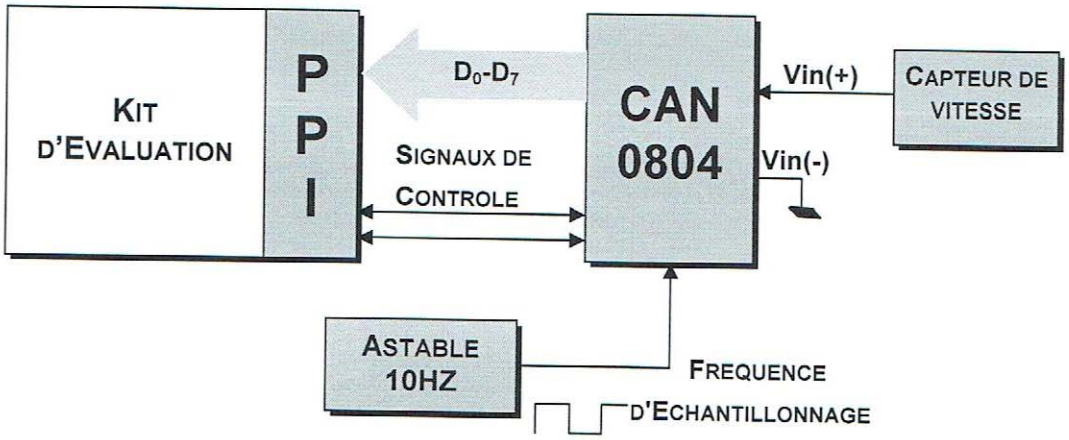


Fig. 4 Commande d'un CAN par un PPI

Le CAN convertit la tension analogique différentielle Vin(+) - Vin(-) sur 8 bits sous le contrôle des signaux $\overline{CS}$, $\overline{SOC}$ (Start Of Conversion), $\overline{EOC}$ (End Of Conversion) et $\overline{RD}$ selon le chronogramme de la Figure 5.

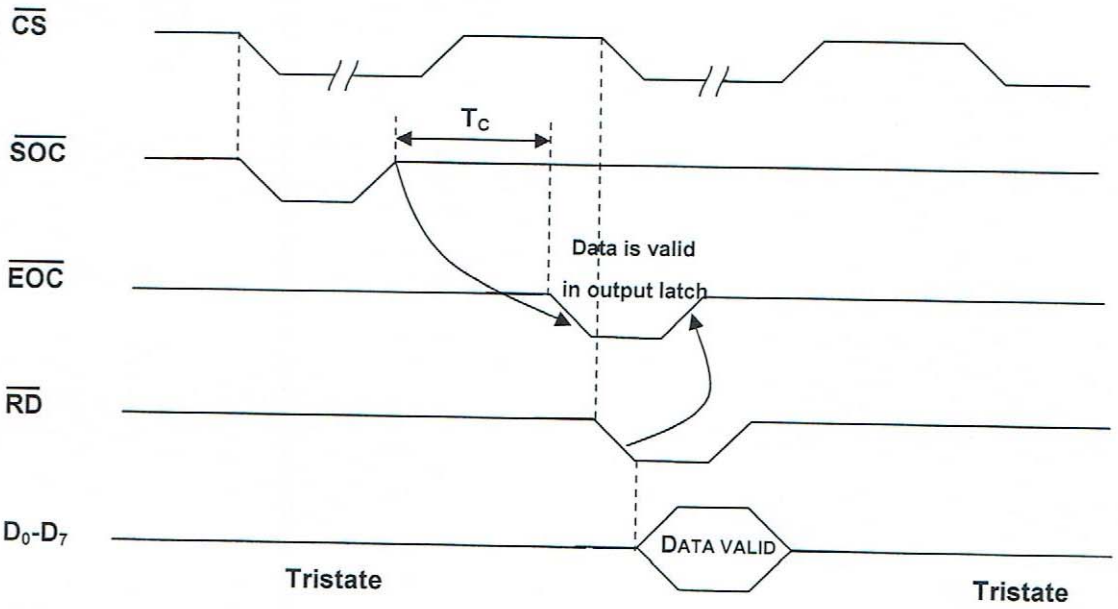


Fig. 5 Chronogramme de fonctionnement d'un CAN

Tc (temps de conversion) ≅ 100µs pour une horloge interne de 640 KHz (à ne pas confondre avec l'horloge externe d'échantillonnage).
L'interruption matérielle associée au PPI est l'interruption INTR-0AH. Les adresses des registres PPI sont: RC = 303H, RDA = 300H, RDC = 302H. On n'utilisera que

le groupe A pour les besoins de cette application. Utiliser le même segment pour le code et les données; CS=DS=F000. Cette zone n'étant pas utilisée par le moniteur. Le moniteur modifie en permanence tous les registres du 8086. Le buffer D-BUFFER sert à conserver, à tout instant d'échantillonnage k, les 4 données numériques les plus récemment acquises. Autrement dit, les données relatives aux instants k, (k-1), (k-2), (k-3). Le programme d'acquisition doit fonctionner en arrière plan; c'est à dire que le kit reste disponible (sous moniteur) pour l'exécution de n'importe quelle commande ou n'importe quel autre programme utilisateur. Le programme doit être scindé en 3 parties:

- une routine d'interruption, assemblée à partir de CS:INTR-0AH, laquelle constitue le corps principal du programme (pilote) et permet de réaliser l'acquisition de la donnée, son stockage dans D-BUFFER et la réorganisation du buffer.
- Une routine d'installation appelée INST-ON, permettant de rendre active la routine d'interruption INTR-0AH. Cette dernière ne réagira à l'interruption matérielle envoyée par le PPI que si le programme INST-ON a été exécuté (une seule exécution suffit).
- Une routine de désinstallation appelée INST-OFF permettant de désactiver le programme INTR-0AH.

1- Quel est le mode de fonctionnement du PPI qui convient le mieux à cette application. Donner les mots de configuration du PPI.
2- Proposer un schéma de câblage de ce système d'acquisition: PPI (Côté périphérie)-CAN-Astable, faire apparaître essentiellement les signaux du CAN:$\overline{CS}$, $\overline{SOC}$, $\overline{EOC}$, $\overline{RD}$, D_0-D_7.
3- Etablir l'organigramme et le programme de la routine INTR-0AH.
4- Donner le programme de la routine INST-ON.
5- Donner le programme de la routine INST-OFF.

Exercice 6: Le convertisseur de palette de couleurs vidéo ADV473 est un circuit doté de 3 octets en entrée R_0 –R_7, G_0 –G_7 et B_0 –B_7 représentant chacun la valeur digitalisée du niveau des couleurs de base vidéo R, G et B. Les IOR, IOG et IOB forment le pixel (point d'image) qui sera affiché à l'écran.

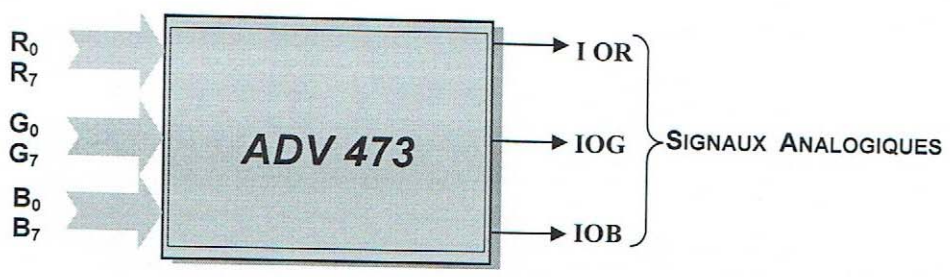


Fig. 6 Brochage d'un convertisseur de palette

On dispose d'une zone mémoire de 60 K octets située à l'adresse DS:VIDEO organisée selon le modèle suivant:

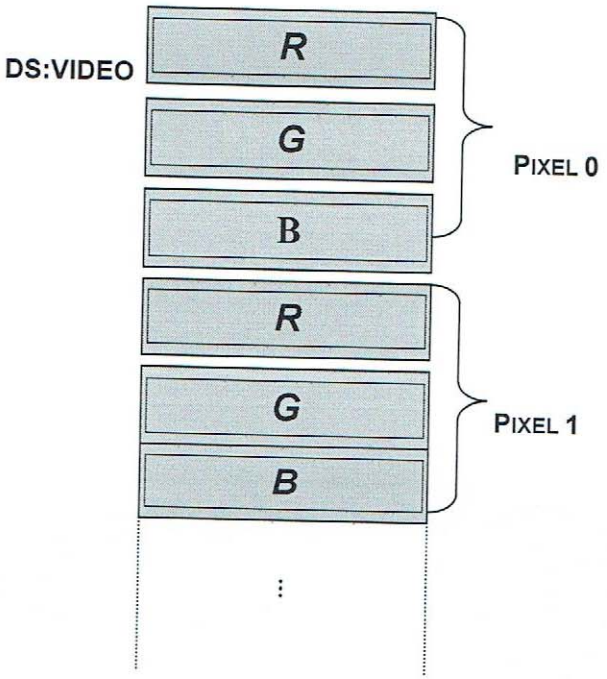


Fig.6 Organisation de la mémoire vidéo

Les pixels doivent être transmis séquentiellement à l'ADV 473 en utilisant une interface de type PPI 8255A dont les registres occupent les adresses d'entrées et de sorties RDA: Port 300H, RDB: Port 301H, RDC: Port 302H et RC: Port 303H.

1- Etablir les connexions entre le PPI 8255A et l'ADV 473, et trouver le mode qui convient à cette opération.
2- Donner l'organigramme et le programme correspondant en assembleur 8086 permettant d'initialiser le PPI 8255A et de transférer le contenu des 60 K octets à l'adresse DS:VIDEO vers l'ADV 473 en respectant l'organisation de l'image vidéo (ordre des couleurs).

Exercice 7: Soient trois interrupteurs (ON/OFF) notés K_0, K_1 et K_2, respectivement, reliés au port A d'un PPI 8255 à travers les lignes PA_0, PA_1 et PA_2. La ligne PB_0 du port B est reliée à une LED active à '1' (cathode à la masse à travers une résistance). Il s'agit de scruter cycliquement les interrupteurs K_0, K_1 et K_2 et d'allumer la LED lorsque la condition (K_0, K_1, K_2) = (1, 0, 1) est vérifiée. Pour cela, écrire un organigramme et son programme en mnémoniques 8086 réalisant une telle application. Les adresses des registres du PPI : RDA=300H, RDB=301H et RC=303H. L'adresse du programme principal est CS:IP=CS:0100H et CS=DS=ES=SS.

1- Donner le schéma fonctionnel d'une telle application. Faites apparaître tous les signaux impliqués dans la liaison du 8086 et du PPI, d'une part et ceux du PPI et des interrupteurs et la LED, d'autre part.
2- Donner sous forme de procédure INIT, la séquence d'initialisation du PPI.
3- Donner l'organigramme et le programme correspondant en mnémoniques et directives d'assemblage du 8086 permettant la gestion d'un telle tâche.

Indictions: On suppose que le 8086 ne fait rien d'autre en dehors de la gestion de cette tâche.

Exercice 8: Une imprimante de 64 K octets est reliée au port A d'un PPI programmé en mode M1. Les lignes de dialogues de l'imprimante sont le $\overline{\text{STROBE}}$ et le BUSY. Il s'agit d'établir un dialogue entre le 8086 et l'imprimante à travers le PPI sous contrôle d'une routine d'interruption. C'est le 8086 qui commence à envoyer le premier des 64 K octets. Dans ce cas, l'imprimante le reçoit et active son signal BUSY pendant la réception. La mise à '0' de BUSY signifie au 8086 la possibilité de l'envoi du prochain octet et ainsi de suite. On suppose que le 8086 ne fait rien d'autre en dehors de la gestion de l'imprimante. L'adresse du message à imprimer se trouve à partir de ADR-BUFFER. Les adresses des registres du PPI sont RDA=378H et RC=37BH. Le vecteur d'interruption=AAH. L'adresse du sous programme d'interruption est CS:IP = 2BCEH:0200H. L'adresse du programme principal est CS:IP = CS:0100H. (CS#2BCEH) et CS=DS=ES=SS.

1- Donner la configuration d'une telle application. Faites apparaître tous les signaux impliqués dans la liaison du 8086 et du PPI, d'une part et ceux du PPI et de l'imprimante, d'autre part . Le PPI doit apparaitre en interne et en externe.
2- Donner sous forme de procédure, la séquence d'initialisation du PPI .
3- Donner sous forme de procédure l'initialisation des registres du 8086 et le chargement du vecteur d'interruption.
4- Donner sous forme de procédure la routine d'interruption.
5- Donner l'organigramme et le programme correspondant en mnémoniques et directives 'assemblage du 8086 permettant la gestion d'un tel dialogue.

Exercice 9: Un contrôle type de processus industriel est donné par la figure 1. On considère, pour cela, le monitoring (contrôle) de ce processus par un PC à travers le bus système (ISA). L'occurrence de n'importe qu'elle anomalie (erreur ou défaut) est signalée au CPU-8086 à travers le PPI-8255. A cet effet, des lignes d'asservissement notées 0, 1,... et 7 sont reliées au système sous contrôle à travers ce dernier (PPI-8255). Les adresses des registres RDA=300H, RDB=301H, RDC=302Het RC=303H. L'adresse du programme principal CS:IP=CS : 0100H et CS=DS=ES=SS.

1- Compléter le schéma de la figure 1, en indiquant, d'une part, les signaux impliqués dans la liaison du CPU-8086 avec le PPI-8255 et, d'autre part, le port reliant le PPI-8255 au processus industriel.
2- En spécifiant le mode de fonctionnement du PPI-8255 le plus approprié à cette application, donner sous forme de procédure notée INIT-PPI-8255, la séquence d'initialisation du PPI-8255.

La séquence de monitoring du système par le PC est la suivante:

a- Initialisation du PPI-8255.
b- Mise en marche du système.
c- Attente d'une minute.
d- Début de la scrutation qui comporte les tests de l'alarme de l'alimentation du système, l'alarme de la température du système et de la pression du système.

Pour cela,

- Si le CPU-8086 détecte un défaut d'alimentation alors, il commande l'arrêt du système, déclenche un scan pour déceler et réparer la panne et retour à b.

- Si le test révèle un défaut de température et/ou un défaut de pression alors, le CPU-8086 déclenche le chauffage du système et/ou le pressurise.

- Sinon, c'est-à-dire si le CPU-8086 ne détecte aucune anomalie de fonctionnement, alors retour à d.

3- Donner sous forme de procédure notée TEMPO, la temporisation d'une minute.
4- Traduire la séquence de monitoring en organigramme.
5- Ecrire le programme, noté MONITOR, correspondant en mnémoniques et directives d'assemblage du CPU- 8086.

Description des lignes d'asservissement du système sous contrôle.

Ligne 0: Alarme pression du système
Ligne 1: Pressurisation du système
Ligne 2: Mise en marche du chauffage du système
Ligne 3: Alarme température du système
Ligne 4: Arrêt du système
Ligne 5: Alarme alimentation du système
Ligne 6: Alimentation du système
Ligne 7: Scan du système

Indications:

1- On suppose que le 8086 ne fait rien d'autre en dehors de la gestion de cette tâche.
2- Il ne vous est pas demandé d'écrire la routine SCAN mais de l'appeler à chaque fois que cela est nécessaire.

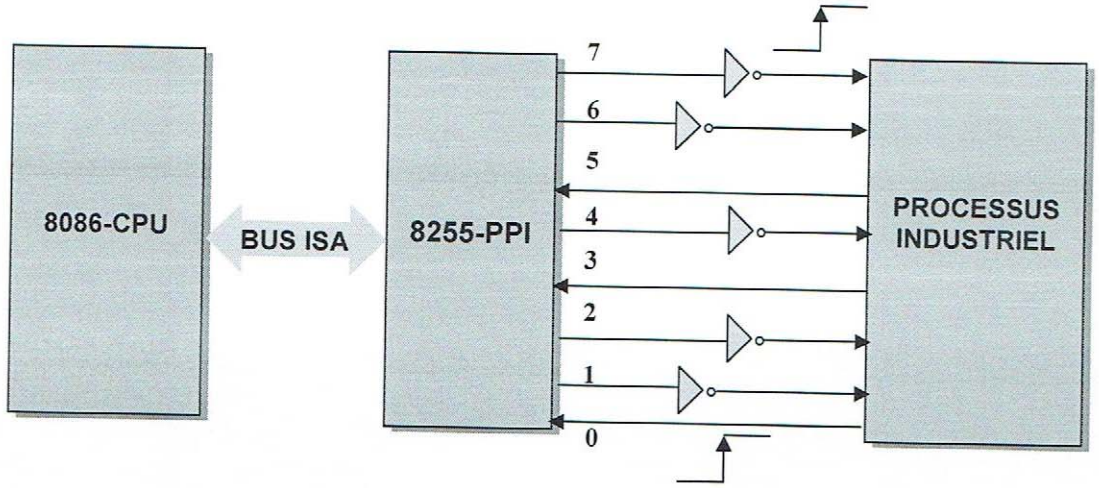


Fig. 7 Synoptique de contrôle d'un processus industriel

TRAVAUX DIRIGES N°8

Exercice 1: On veut recevoir d'un Télétype une donnée et la stocker en mémoire à l'adresse 4EA0:0000. On utilise pour cela un USART en interface asynchrone. On ne travaille pas en interruption. La fréquence R_XC est de 1760 Hz et on veut obtenir une vitesse de 110 Bauds. Le Télétype transmet des caractères de 7 bits avec parité impaire et deux bits stop. La ligne de transmission est inactive. Les interruptions du transmetteur sont inhibées. Les adresses du registre de données en R_X et du registre de contrôle sont, respectivement, 00H et 01H.

1- On suppose que le message reçu ne présente aucune erreur.
2- Dans certains cas, le message peut présenter une erreur de format, une erreur de surcharge et/ou une erreur de parité.

Etablir l'organigramme et le programme en assembleur 8086 correspondant dans chacun des deux cas 1 et 2.

Exercice 2: On veut envoyer une donnée stockée à l'adresse 4EA0:0000 vers un Télétype. Les conditions de fonctionnement de l'USART sont les suivantes: On ne travaille pas en interruption. La fréquence T_XC est de 14080 Hz et on veut obtenir une vitesse de 220 Bauds. Le Télétype reçoit des caractères de 8 bits avec parité impaire et 1 bit stop. La ligne de réception est inactive et les interruptions du récepteur sont inhibées. Etablir l'organigramme et le programme en assembleur 8086 permettant un tel transfert. Les adresses du registre de données en T_X et du registre de contrôle sont, respectivement, 00H et 01H.

Exercice 3: On désire établir une liaison sérielle du type RS 232C, comme le montre la Figure 1, entre une carte à base de 8086 et un terminal d'ENTREES/SORTIES. On s'intéresse à la liaison côté carte qui comporte en fait un USART fonctionnant en mode asynchrone. L'USART émet et reçoit des mots de longueur 8 bits avec parité paire et 1 bits stop. La fréquence de décalage pour la transmission et la réception est 1536 KHz. On veut obtenir une vitesse de 2400 Bauds. On ne travaille pas en interruption. Il s'agit d'écrire un programme (organigramme obligatoire) qui obéit à l'algorithme suivant:

1- Initialisation de l'USART
2- Envoi du texte 'TERMINAL SOUS CONTROLE DE LA CARTE' vers le terminal d'entrées/sorties, qui est automatiquement visualisé sur l'écran du terminal.
3- Test de la touche ↵ (code (ASCII=0DH). Si cette touche est enfoncée alors aller à 1, sinon aller à 4.

4- Test de la touche ↓ (code ASCII=69H). si cette touche est enfoncée alors envoi du texte 'TEMINAL LIBRE' vers le terminal et aller à 5, sinon aller à 3.
5- Dans ce cas le terminal peut être utilisé indépendamment de la carte. Toute touche enfoncée du clavier est automatiquement visualisée sur son écran sauf la touche ESC (code ASCII=1BH) est reconnue par la carte comme un retour à 3.

Indications:

1- On pourra utiliser un sous programme SCAN, supposé existant, dont la fonction est de scruter le clavier de la carte et de mettre le code ASCII de la touche enfoncée dans l'accumulateur.
2- On utilisera aussi un sous programme appelé CLEAR, la fonction et d'effacer l'écran du terminal.
3- Les adresses des registres de contrôle et de données en T_X/R_X sont, respectivement, 303H et 302H.

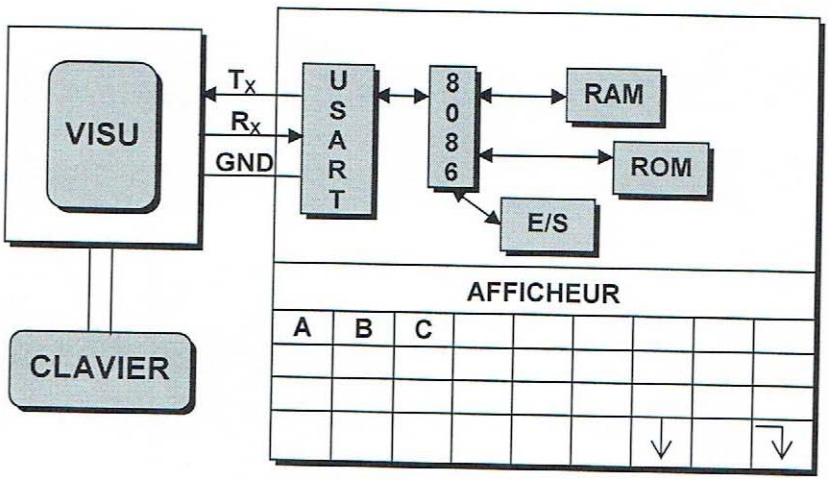


Fig. 1 Liaison type RS 232 Null-Modem

Exercice 4: On désire transférer des caractères provenant d'un périphérique noté P_1 vers un périphérique noté P_2. L'opération consiste à connecter un USART sur une structure à base de 8086, d'une part à P_1 via la ligne R_XD et d'autre part à P_2 via la ligne T_XD. On suppose qu'on travaille en NULL-MODEM simplifié.

1- Donner le schéma synoptique d'une telle application en précisant les lignes de dialogue utilisées dans le cas d'une réception en interruption et d'une transmission en scrutation.

2- Les adresses des registres de l'USART sont: RD T_X/R_X=00H, RC/RE=01H. L'USART travaille en interface asynchrone sur un format de caractère 8 bits avec parité paire et 2 bits STOP. Le décalage des bits en série se fait en transmission et en réception à une horloge 16 fois moindre que celle des horloges de transmission et de réception. Le vecteur d'interruption associé à la réception est D4H et l'offset de début de la routine est 01FFH (le segment étant celui du programme appelant, CS=DS=SS=1ADFH, débutant à IP=0100H).

3- La fonction du programme de gestion d'un tel système consiste à initialiser l'USART en ensuite à effectuer le transfert sous contrôle d'une procédure d'interruption. Il vous est demandé d'établir l'organigramme du programme principal de gestion, de donner la procédure d'interruption associée à la réception sans erreurs et de traduire l'organigramme du programme principal en mnémoniques 8086.

Exercice 5: On veut recevoir d'un Télétype plusieurs données et les stocker en mémoire à partir de l'adresse 8000:0300H. La fin de transmission est reconnue par la détection d'un EOT (End Of Text, code ASCII 04H). On utilise pour cela un USART en interface asynchrone. Le vecteur d'interruption associé à la réception est A0H et l'offset de début de la routine d'interruption est 0200H. Quant au segment de cette routine, c'est celui du programme appelant dont CS=DS=ES=SS=8000H et IP=0100H. La fréquence R_XC est de 1536 KHz et on veut obtenir une vitesse de 2400 Bauds. Le Télétype transmet des mots de 8 bits avec parité paire et un bit stop. La ligne de transmission est inactive. Les interruptions du transmetteur sont inhibées. On suppose que le message ne présente aucune erreur. Les adresses des registres de données et de contrôle sont 0210H et 0211H, respectivement.

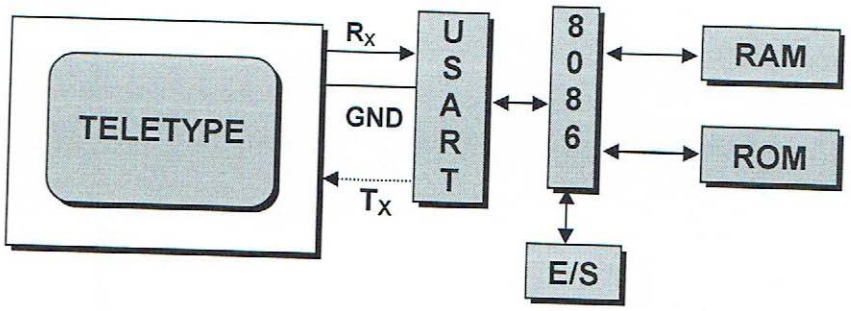


Fig. 2 Liaison télétype-USART

La fonction du programme consiste à initialiser l'USART et ensuite à effectuer la réception sous contrôle d'une routine d'interruption. Il vous est demandé:

1- d'établir l'organigramme du programme principal.
2- d'écrire la procédure d'interruption associée à la réception.
3- de traduire l'organigramme du programme principal en mnémoniques 8086.

Exercice 6: Soit un USART 8251 connecté à un terminal d'E/S et travaillant selon le protocole XON/XOFF. L'adresse des registres de données en T_X et en R_X est 200H et celle des registres de contrôle et d'état est fixée à 201H. La fréquence des horloges de transmission et de réception du 8251 est de 153.6 KHz. Il s'agit d'écrire:

1- Une procédure INIT, qui initialise le 8251 avec les paramètres suivant: 2400 bits/s, 8 bits/caractère, parité paire et 1 bit de STOP.
2- Une procédure d'émission ENVOI d'un message contenu dans la zone d'adresses MSG. L'émission s'arrête lorsque le caractère EOT (End Of Text, code ASCII=03H) est rencontré.
3- Une procédure de réception RECEPTION, qui reçoit une ligne de 80 caractères et la range dans une zone de données appelée LIGNE. En cas d'erreur de réception d'un caractère (format, parité ou surcharge), envoyer le caractère NAK (No AcKnowledge, code ASCII=15H), sinon, envoyer le caractère ACK (ACKnowledge, code ASCII=06H).

Indication: Pour les procédures ENVOI et RECEPTION, Il faut prévoir une procédure qui sera notée ENVOI-CARACTERE et dont la fonction principale est l'envoi d'un caractère unique.

Exercice 7: On désire connecter un terminal d'E/S à une carte à base de microprocesseur CPU-8086 à travers un USART-8251 par l'intermédiaire d'une liaison RS232C Null MODEM complet. L'USART-8251 transmet et reçoit des mots de 8 bits sans parité avec 1 bit STOP. Le décalage des bits se fait à une fréquence de 153.6 KHz et on veut obtenir une vitesse de 9600 Bauds. On suppose que seuls les états du transmetteur et du récepteur peuvent interrompre le CPU-8086. Les messages reçus ne présentent aucune erreur. Les interruptions du transmetteur et du récepteur ne sont jamais validées en même temps.
Il s'agit d'établir un dialogue entre le CPU-8086 et le terminal d'E/S sous contrôle des interruptions. On ne s'intéresse qu'à la gestion côté CPU-8086. La stratégie de gestion étant la suivante: C'est le CPU-8086 qui commence à émettre. Dans ce cas, le terminal d'E/S reçoit et inversement. La fin d'un message en transmission ou en réception est détecté par un ETX (End of TeXt, ASCII = 03H). Pour savoir si l'USART-8251 est en transmission ou en réception, on utilisera un BYTE-T_X-R_X (Un octet) en mémoire. Si BYTE-T_X-R_X=1 alors l'USART-2851 est en transmission, sinon il est en réception. L'adresse du message à transmettre à partir de D-BUFFER-TRANS. On réservera 100 H cases mémoire. L'adresse du message à

recevoir à partir de D-BUFFER-REC. On réservera 100 H cases mémoire. Les dresses RD T_X/R_X = 08H et RC/RE = 0AH. Le vecteur d'interruption commun à la transmission et à la réception = 2BH. L'adresse de la routine d'interruption CS:IP=2000H:F000H. L'adresse du programme principal CS:IP=CS:0100H et CS=D = ES=SS.

1- Donner la configuration d'une telle application. Faites apparaître tous les signaux de données, d'adresses et de commandes du CPU-8086 et de l'USART-8251. Préciser la configuration Full Modem complet USART-8251 et Terminal d'E/S.
2- Donner l'organigramme et le programme, noté DIALOGUE, de gestion d'un tel dialogue en assembleur et directives d'assemblage du CPU-8086. Préciser notamment: sous forme de procédures, la séquence d'initialisation de tous les registres du CPU-8086 sauf celle des registres segments (notée INIT-CPU-8086), la séquence d'initialisation de l'USART-8251 en transmission (notée INIT-USART-8251) et la routine d'interruption (notée ROUT-INT).

Indication: On suppose que le CPU-8086 n'est connecté à aucun autre périphérique.

MANIPULATIONS

TRAVAUX PRATIQUES N°1

INITIATION AU SYSTEME MS-DOS

I Généralités

MS-DOS est le système d'exploitation le plus utilisé sur les micro-ordinateurs compatible IBM PC. La version 6 offre un large éventail de nouveaux utilitaires pour transformer l'ordinateur en un puissant outil de travail. En plus des fonctions classiques disponible dans les anciennes versions du DOS, MS-DOS 6 renferme des outils principalement destinés à:

1- L'optimisation de la mémoire disponible: MemMaker.
2- La libération d'espace mémoire sur disque par compression de données: DoubleSpace.
3- La réduction du temps de lecture des données sur disque dur: SmartDrive.
4- La sauvegarde préventive des données, contre les effacements accidentels et les défaillances du disque dur: MS-Backup.
5- La protection contre les virus informatiques: MS-Av.

En plus MS-DOS 6 permet de gérer la connexion réseau et offre tous les drivers d'imprimantes disponibles.

II Objectifs

Utilisation du micro-ordinateur en tant qu'outil de travail pour les prochaines manipulations et maîtrise des notions fondamentales du système d'exploitation, notamment celles qui se rapportent à:

1- L'utilisation du programme Setup pour la modification de la configuration matérielle du système.
2- La création du fichier autoexec.bat pour le démarrage personnalisé du système.
3- La gestion des fichiers et sous répertoires.
4- Fonctions relatives aux disques et à la réaction de disquette système.

III Travail demandé

III.1 Identification de la configuration matérielle

Exécuter les tâches suivantes:

1- Démarrer la machine et activer le programme SETUP. Ce programme est implanté en ROM et agit sur les données de la RAM CMOS. Ces données définissent la configuration matérielle du système. Dans le menu de ce logiciel, utiliser l'option STANDARD CMOS SETUP. Identifier les différents équipements rattachés au système: paramètres HD, type FD, présence de coprocesseur mathématique, type de carte graphique, taille mémoire conventionnelle, taille mémoire XMS etc.
2- Changer un des paramètres, sauvegarder la nouvelle configuration et redémarrer le système. Conclusion?
3- Revenir à la configuration initiale.

III. 2 Commandes usuelles

Exécuter et commander quelques commandes de base MS-DOS telles que: DATE, TIME, VER, CLS, DIR, DIR/P, DIR/W, DIR*.EXT, DIR XXX.*, CD REP, CD.., CD\, RENAME, DEL, COPY, ATTRIB, MKDIR, RD, PATH, PROMPT...

III.3 Création d'une disquette amorçable

Utiliser une disquette haute densité (1.44Mo) que vous introduirez dans le lecteur A. procéder au formatage de la disquette avec transfert du système minimum (les deux fichiers cachés du dos et l'interpréteur de commande.

1- Examiner l'état physique de la disquette en comparant la taille de l'espace mémoire disponible à la taille des fichiers transférés. Conclure?
2- Vérifier le contenu de la disquette en comparant la taille de l'espace mémoire disponible à la taille des fichiers transférés. Conclure?
3- Redémarrer votre système à l'aide de cette disquette.

III.4 Répertoires et sous répertoires

Remarquez que la disquette ainsi obtenu ne permet pas une utilisation correcte de la machine; clavier non configuré, gestionnaire de périphériques non installés, convention nationales non définies tels que symboles monétaires, format des dates et heures, etc. Rétablir pour cela un minimum de confort pour votre système en utilisant, depuis le disque dur:

1- La commande KEYB pour rétablir une configuration française du clavier.

2- La commande COUNTRY associée aux paramètres français (code 033, fichier COUNTRY.SYS).
3- Le gestionnaire de la mémoire étendue (XMS gérée par HYMEM.SYS).

4- Créer dans votre disquette un sous répertoire A:\DOS> dans lequel seront copiés les fichiers nécessaires au fonctionnement correct de la machine. Pour cela prendre en compte les hypothèses suivantes: Le clavier doit être convenablement configuré. La mémoire étendue physiquement présente, doit être gérée. On suppose qu'on travaillera ultérieurement avec des applications DOS nécessitant l'utilisation de la mémoire paginée, les machines dont on dispose ne sont pas dotées de cartes de mémoire paginée; il vous est demandé à cet effet de SIMULER cette dernière au moyen du gestionnaire EMM386.
5- Les conventions nationales des formats d'affichage doivent être définies.

III.5 Personnalisation et configuration du système

Utiliser l'éditeur du DOS EDIT.COM pour créer et modifier les fichiers AUTOEXEC.BAT et CONFIG.SYS. En plus des hypothèses de travail déjà mentionnées, le système doit répondre aux exigences suivantes après le démarrage:

1- L'invité du DOS (Message d'attente) doit être personnalisé.
2- Le système doit reconnaître tous les chemins d'accès aux sous répertoire existants, non seulement sur la disquette de travail, mais également sur le disque dur.
3- Afin de libérer de la mémoire conventionnelle, la partie résidante du DOS doit être chargée en mémoire haute (HMA ou les 64 premier K Octets de la mémoire étendue). Enumérer pour cela, les commandes qui doivent être contenues dans le fichier AUTOEXEC.BAT et celle qui doivent être dans le fichier CONFIG.SYS. Que doit-on faire pour que les commandes introduites dans AUTOEXEC.BAT aient un effet sur le système ? Aussi, que doit-on faire pour que les commandes introduites dans CONFIG.SYS aient un effet sur le système? Utiliser la commande MEM pour examiner l'utilisation par le système des différentes parties de la mémoire RAM (conventionnelle, XMS, Paginée, HMA.

TRAVAUX PRATIQUES N°2

FAMILIARISATION AVEC LES COMMANDES DU SIMULATEUR DU 8086

I Objectifs

1- Familiarisation avec l'environnement intégré de SIM.
2- Utilisation de quelques lignes de commande de SIM.
3- Test de quelques signaux de commandes du 8086 en mode simulation.
4- Utilisation de SIM en mode privilégié (exécution en temps réel).

II Environnement de SIM

Le logiciel SIM comporte essentiellement le fichier exécutable SIM.EXE, les fichiers utilitaires à l'édition des liens, la conversion...etc. et quelques fichiers de démonstration. L'ensemble de ces fichiers se trouvent dans le répertoire C:\SIM86. Il vous est demandé de lancez le logiciel à partir de ce répertoire.

1- Tapez la commande DSCRN ↵. Observez les 5 fenêtres actives sur l'écran suivantes:

a- Une fenêtre "Micro étapes d'instructions"(vide au début)
b- Une fenêtre "Flag"
c- Une fenêtre "Etat du simulateur"
d- Une fenêtre "Processeur"
e- Une fenêtre "De assembleur"

Concomitamment, le symbole >> s'affiche en bas de l'écran.

Tapez BIG ↵. Que constatez-vous?
Tapez BIG ↵ encore une fois, Que constatez-vous?

2- A l'aide de la ligne de commande LOAD NOM DE FICHIER.*, chargez un fichier de démonstration LOOP1.DEM, par exemple. Quelles sont les modifications observées à l'écran. Tapez U 100 ↵. Quelle est la différence entre les lignes qui viennent de s'afficher et la fenêtre d'assemblage? Tapez MEM 100 ↵. Commentez? Tapez DMEM OFF ↵. Commentez?

3- Tapez la commande SIM 10 ↵ et observez les différentes fenêtres du logiciel. Relancez la simulation à partir de IP 100 en contrôlant la vitesse d'exécution à partir du pavé numérique (de 1 à 9). Que constatez-vous?
4- Choisissez le niveau de simulation par la commande STEP n (n=0, 1, 2, 3 ou 4). Observez attentivement la fenêtre "Micro étapes d'instruction", pour tous les modes de simulation. Commentez?
5- Reprenez 4 mais cette fois-ci, exécutez, au préalable, la commande T, (Trap ou exécution pas à pas). Comparer avec la commande STEP 1". Observez attentivement le message à l'écran. Que concluez-vous?

III Quelques lignes de commande du SIM

1- La commande "Registre" sert au chargement des registres sous la ligne de commande du simulateur. La syntaxe est: NOM DU REGISTRE VALEUR. Valeur est la donnée qui devra être chargée dans le registre. Exemple: AX FFFF.
2- Rechargez les valeurs de départ (restauration). Si vous les avez omises, utiliser la commande NEW.
3- La commande "E" sert à écrire à une adresse mémoire sous la ligne de commande du simulateur. La syntaxe existe selon deux formats: E adresse ou E adresse liste. Exemple: E DS:1234 ou E DS:1234 11 22 33 FF (dans le cas d'une liste).
4- Essayer d'écrire diverses valeurs de votre choix dans les adresse à partir de DS:OFFSET (DS et OFFSET= valeurs de votre choix). Visualisez la mémoire en utilisant la commande DMEM. Reprenez la même commande mais cette fois-ci avec le deuxième format.

IV Fonctionnalité de quelques signaux du 8086 en simulation

Tapez >> A adresse ↵. Ecrire un programme à partir de l'adresse CS:0100 (IP=0100). Pour ce faire, écrire un programme contenant la séquence d'instructions suivante:

```
WAIT
MOV AX, 456FH
MOV SI, 123AH
LOCK  MOV AX,DS:[SI]
BRK ou INT 03H
```

Valider le programme avec Ctrl-C.

1- Lancez la simulation. Pour mieux apprécier l'exécution du programme, vous pouvez, au préalable, commencer par exécuter la commande STEP n avec n=2. Que remarquez-vous?
2- Pour rompre l'état d'attente appuyer sur la touche "t". Que concluez-vous?

3- Reprenez la valeur IP 100, essayer de rompre l'état d'attente en appuyant sur la touche "i". Dites pourquoi il n'y a pas eu de rupture de l'état d'attente. Essayer la touche "n". Que concluez-vous?
4- Continuer l'exécution du programme jusqu'à l'instruction ayant le préfixe LOCK. Que constatez-vous?

V Exécution en temps réel

1- L'exécution en temps réel du programme commençant à CS:IP ne peut se faire qu'en mode privilégié. Pour passer à ce mode, utilisez la commande PRIV. Une fois dans le mode privilégié, pour lancer l'exécution, tapez G ↵. Comparez avec la Section 2.
2- Pour lancer l'exécution pas à pas, tapez T 100 ↵. Que remarquez-vous. Continuez l'exécution du programme en agissant sur 'la barre espace'.
3- Chargez CS par FFFF et IP par 0000 (chargement direct). Lancez l'exécution en temps réel (G ↵). Commentez.

TRAVAUX PRATIQUES N°3

INITIATION A LA PROGRAMMATION AVEC LE SIMULATEUR DU 8086

I Assemblage

Vérifier l'assemblage des instructions suivantes en spécifiant les champs w, d, mod, r/m, data, addr low, addr high (IR doit être affiché en binaire).

```
MOV AX, 0200
MOV BX, AX
MOV SI, BX
MOV [SI], BX
MOV [0200], BX
MOV ES:[SI], 0200
```

Simuler l'exécution détaillée de ces instructions en observant les registres et mémoires concernées.

II Désassemblage

1- Désassembler les codes instructions suivantes: 8B C2 89 D0 8B C6 89 F0 CC. Donner une explication de ce que vous observez.
2- Désassembler le code instructions suivant: 90 64 66 6D D8 04 CC

Exécutez cette séquence d'instructions en mode simulation puis en mode privilégié. Que concluez-vous?

III Affectation des flags

1- Donner toutes les instructions de positionnement des flags.
2- Exécuter les instructions suivantes en observant les flags affectés.

```
MOV AX, 7FFF
ADD AX, 0001
MOV AX, 7FFF
INC AX
MOV AX, FFFF
ADD AX, 0001
MOV AX, FFFF
```

INC AX
SUB AX, AX

Proposez une séquence d'instructions pour positionner le flag T?

IV Utilisation de pile

1- Assembler d'abord RET à CS:0200 et RET FAR à 3000:0400.
2- Simuler ensuite l'exécution de séquence suivante. Bien observer SP et [SP].

CALL 200
CALL FAR 3000:0400
INT 1CH

V Ajustements décimaux

1- Comme le 8086 effectue des ajustements décimaux sur 8 bits uniquement. Etablir un programme en assembleur 8086 permettant d'effectuer des ajustements décimaux sur 16 bits.
2- Etablir un programme en assembleur 8086 permettant d'effectuer l'addition de deux nombres double-word (32 bits chacun) N1 et N2 codés en BCD. Le résultat de l'addition doit être en BCD.

VI Conversion Hexadécimal-Octal

Etablir un programme en assembleur 8086 permettant d'effectuer la conversion Hexadécimal-Octal d'un nombre multi-octets N, en utilisant:

1- Le principe des décalages (Shift)
2- Le principe de masquage

VII Adressage IP (Internet Protocol)

Dans le réseau Internet, les machines sont identifiées par une adresse appelée IP (Internet Protocol) définie par le format BCD X.Y.Z.W. Les champs X, Y, Z et W sont codés en BCD sur 3 digits chacun. A titre d'exemple, l'une des adresses IP du CERIST (Fournisseur d'accès ou Provider en Algérie) est 193.194.64.11. Les points sont virtuels. Cette représentation permet de constituer toutes les adresses. Autrement dit, toutes les combinaisons de 0.0.0.0 à 255.255.255.255, en 5 classes appelées A, B, C, D et E.

La classe A contient les adresses dont le premier champ X converti en binaire commence par 0.

La classe B contient les adresses dont le premier champ X converti en binaire commence par 10.
La classe C contient les adresses dont le premier champ X converti en binaire commence par 110.
La classe D contient les adresses dont le premier champ X converti en binaire commence par 1110.
La classe E contient les adresses dont le premier champ X converti en binaire commence par 11110.

1- A quelle classe appartient l'adresse IP du CERIST ?
2- Etablir l'organigramme permettant de convertir une adresse IP quelconque en binaire (codé sur 32 bits : 1 octet pour chacun des champs X, Y, Z et W) et de trouver la classe à laquelle il appartient.
3- Traduire cet organigramme en utilisant les mnémoniques et les pseudos instructions du 8086.

Données:

1- L'adresse IP en BCD est stockée à partir de DS:200
2- L'adresse IP en binaire stockée à partir de DS:210
3- Le registre DL contient la classe de cette adresse IP. C'est à dire, le code ASCII de A, B, C, D ou E.

TRAVAUX PRATIQUES N°4

PROGRAMMATION EN ASSEMBLEUR

I Objectifs

1- Utiliser M (T) ASM pour la première fois familiariser avec toutes les opérations nécessaires à la création d'un programme en Assembleur: édition, assemblage et édition de liens.
2- Suivre l'exécution d'un programme pas à pas en suivant l'évolution des registres et de la mémoire avec un débogueur DOS comme le Code View (CV) ou Turbo Débogueur (TD).
3- Faire fonctionner des programmes en Assembleur simple comme ceux vus en travaux dirigés.
4- Programmer un générateur d'une suite pseudo-aléatoire LSFR (Linear Feedback Shift Register).

II Programmation du générateur

La suite d'une séquence de nombres aléatoires est générée selon un schéma de substitutions de bits et de décalages itératifs à partir d'un mot $A=a_{15}\,a_{14}\,\ldots\,a_2\,a_1\,a_0$ initial de votre choix et d'un bit 'b' comme le montre la Figure 1.

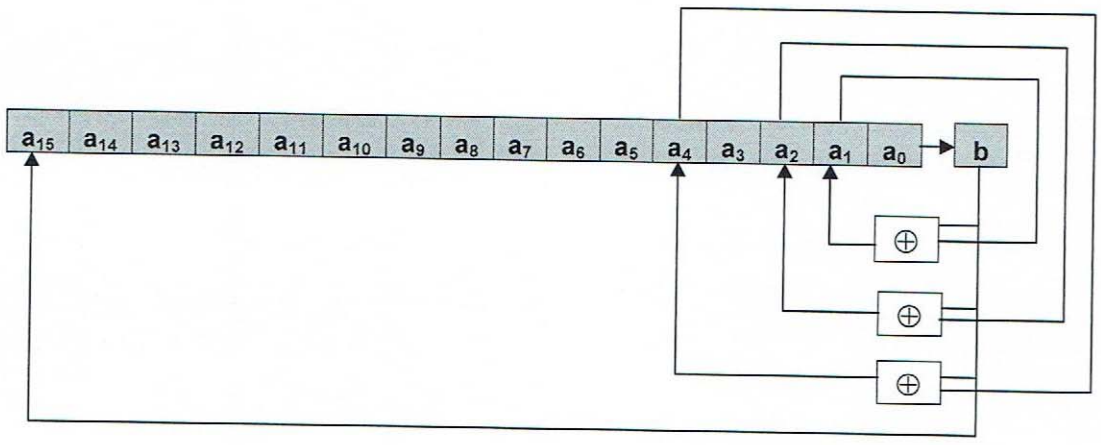


Fig. 1 Générateur d'une suite pseudo-aléatoire LSFR

1- L'algorithme effectue la substitution des bits $a_1=a_1\oplus b$, $a_2=a_2\oplus b$, $a_4=a_4\oplus b$ et $a_{15}=b$. Les autres bits de A restent inchangés. Le symbole $\oplus$ désigne l'opérateur OU EXCULSIF.

2- Le résultat de 1- est considéré comme un nouveau mot aléatoire et est déplacé de A vers son adresse de destination dans la mémoire de données.
3- Décaler A de 1 bit vers la droite avec $b=a_0$.

Ces trois étapes sont répétées 32768 fois. Les registres DS et ES devront être choisis suffisamment éloignés des registres CS et SS afin de ne pas faire chevaucher vos propres programmes et la pile.
En vous aidant de l'algorithme ci-dessus, il vous est demandé d'établir l'organigramme et le programme en Assembleur 8086 permettant la réalisation d'un tel générateur de suite pseudo-aléatoire.

III Vérification de la non répétition des mots

Le programme vient de s'exécuter. Comment vérifier qu'il a bien fait ce qu'il devait faire? Une manière de valider l'exactitude du programme précédent consiste à s'assurer qu'aucun mot ne se répète le long des 32 K Mots formant la suite. Pour ce faire, faites le calcul à la main et donnez les 10 premiers mots résultats en commençant par b=1 et A =XX YY. Sachant que votre date de naissance est: XX/YY/19... Donnez le contenu des registres et de leurs évolutions pour la première itération du programme.

Pour les plus forts, il est possible d'établir un organigramme et d'écrire le programme correspondant en Assembleur 8086 qui permet la vérification de la non répétition des mots.

1- Etablir les probabilités d'occurrences des quartets 0 et F pour la zone mémoire de 2 K Octets à partir de DS:ADR-ASCII.
2- A partir du graphe, établir le code Huffman de chaque quartet au choix, à la main ou par programme.
3- Convertir la zone mémoire DORG en code de Huffman DS:ADR-HUFF
4- vérifier le taux de compression.

IV Utilisation des programmes MASM (TASM) et V (ou TD)

Ecrie le programme dans un fichier *.asm avec l'éditeur edit.exe du DOS. Il faut ouvrir un nouveau fichier avec 'New'. Ecrire le programme et le sauvegarder avec 'Save'. Assemblez-le avec M(T) asm pour vérifier que vous n'avez pas d'erreurs de syntaxe. Pour produire un fichier exécutable, utilisez (T) link.

CV (ou TD) est un programme qui vous permettra de faire les choses suivantes:

1- Exécuter un programme pas à pas.
2- Observer le contenu des registres.

3- Observer le contenu de la mémoire.
Pour cela, il faut encore exécuter la commande CV (ou TD) programme.exe. En utilisant la commande STEP, exécutez le programme en pas à pas et vérifiez le contenu des différents registres. Initialisez A à XX YY et comparez vos résultats avec ceux du programme.

VII Procédure d'affichage

Vous devez ajouter la procédure suivante à votre programme pour afficher le résultat à l'écran, si le programme est exécuté sans le débogueur.

```
AFFICHE     PROC NEAR
            PUSH SI
            PUSH AX
; GET CURRENT BYTE
            MOV AL,[Si]
; PRINT CHARACTER IN TELETYPE MODE
            MOV AH,0EH
            INT 10H
; UPDATE INDEX REGISTER BY 1
            INC SI
; GET SECOND BYTE
            MOV AL,[SI]
; PRINT CHARACTER IN TELETYPE MODE
            MOV AH,0EH
            INT 10H
            POP SI
            POS AX
            RET
AFFICHE     ENDP
```

Adaptez la procédure à votre programme pour afficher le contenu du résultat dans DS:[SI] et appeler au moyen de l'instruction CALL sans même se soucier de son fonctionnement. Cependant, il est important de l'appeler directement après l'instruction de sauvegarde et avant l'incrémentation de SI.

Travaux Pratiques N°5

INTERRUPTIONS

PARTIE I

I Objectifs

1- Comprendre le mécanisme des interruptions tels que la sauvegarde et la restitution du contexte, vecteur d'interruption, adresses des routines d'interruptions, table des vecteurs d'interruptions, etc.
2- Maitrise des interruptions internes, des interruptions BIOS (INT 01H et INT 16H), de l'interruption matérielles masquable INT 08H et des caractères ASCII étendus.

II Utilisation de l'interruption interne INT 01H (Trap ou pas à pas)

Pour ce faire, travaillez en mode de simulation niveau 2.

1- Visualiser le contenu de la table des interruptions. Quelle est l'adresse du sous programme associé à l'interruption INT 01H? Quel est le contenu de cette routine?
2- Positionnez manuellement le flag T à '1'. Assemblez à partir de CS:100, 2 instructions simples de votre choix (en dehors du BRK). Lancez l'exécution et observez. Expliquez tous les détails affichés au niveau de la fenêtre micro- étapes. Expliquez le mécanisme de fonctionnement de l'interruption INT 01H.
3- Expliquez pourquoi il y a sans cesse exécution de l'instruction IRET ? Quelles sont les instructions qu'on doit rajouter dans le sous programme associé a l'interruption INT 01H (à la fin de la routine) pour éviter ce blocage. Peut-on rajouter ces instructions à partir de l'adresse initiale de l'interruption INT 01H (adresse trouvée en 2-) ?
4- Choisissez une nouvelle adresse pour la routine d'interruption INT 01H et assemblez la séquence d'instructions proposée en 3- (ceci s'appelle un déroutement de l'interruption INT 01H). Reprenez les hypothèses de la question 2- et lancez l'exécution. Conclusion?

III Utilisation des interruptions BIOS et de l'interruption INT 08H

Ecrivez un programme utilisant les services des interruptions INT 10H et INT 16H, qui permet:

1- D'afficher à l'écran, caractère ce qui est saisi au clavier.
2- De prendre en compte les caractères 'Imprimables' alphanumériques.
3- De prendre en compte certains caractères de contrôles tels que le déplacement droit, gauche, haut et bas, back space (pour effacement du caractère précédent le curseur), suppr (pour effacement du caractère à la position du curseur), Entr (pour saut de ligne simple sans décalage des lignes vers le bas)
4- Tous les autres caractères sont ignorés.
5- L'appui sur la touche ESC, efface l'écran, repositionne le curseur dans le coin supérieur gauche, temporise 5 secondes et rend la main à l'utilisateur afin d'effectuer une nouvelle saisie.
6- La touche Fin permet de quitter le programme.

PARTIE II

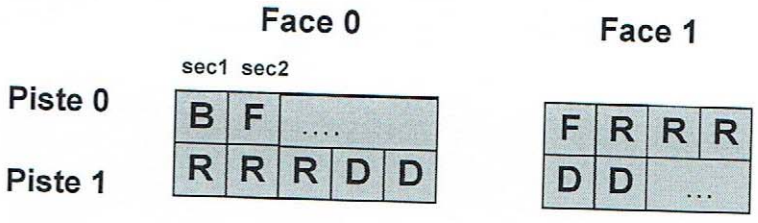
I Objectifs

1- Compréhension des interruptions dans le 8086.
2- Utilisation de quelques interruptions matérielles, BIOS, MS-DOS (paramètres d'entrées et de sorties) dans la génération des temporisations, la lecture d'un caractère à partir du clavier et la gestion de l'écran.

II Affichage du répertoire d'une disquette sur l'écran

Ecrire un programme en assembleur 8086 qui affiche sur l'écran le répertoire principal de votre disquette.

II.1 Structure logique de la disquette



B: secteur d'amorçage (bootstrap)
F: table d'allocation des fichiers (FAT, File Allocation Table)

R: répertoire principal
D: données

II.2 Paramètres disquette

Les paramètres de la disquette sont situées dans le bootstrap aux adresses suivantes:

Word B: nombre d'octets par secteur
Byte D: nombre de secteur par cluster
Word E: nombre de secteurs occupés par le bootstrap
Byte 10: nombre d'exemplaire de la FAT
Word 11: nombre de fichiers max répertoire principal
Word 13: nombre de secteur total utilisés par MS-DOS
Byte 15: type de disquette
Word 16: nombre de secteurs occupés par une FAT
Word 18: nombre de secteur par piste
Word 1A: nombre de faces
Word 1C: numéro du secteur précédant la FAT

II.3 INT 13H: fonction lecture (AH=02H)

Pour la définition de la fonction lecture de l'interruption INT 13H et ses différentes fonctions et entrées et sorties, se référer à l'annexe 7.

II.4 INT 10H fonctions d'affichage en mode texte (AH=00H et AL=03H)

Pour la définition de la fonction affichage en mode texte de l'interruption INT 10H et ses différentes fonctions et entrées et sorties, se référer à l'annexe 7.

III Utilisation des interruptions INT 08H et INT 21H

III.1 Définitions des interruptions à utiliser

III.1.2 Horloge temps réel (Real Time Clock Interrupt): INT 08H

Cette interruption est du type matériel. Le mot situé à l'adresse 0040:006C est incrémenté de 1 par l'interruption INT 08H toutes les 1/18.2 sec. Exemple: 14sec=256 x 1/18.2 sec.

III.1.2 Lecture d'une chaîne de caractères à partir du clavier: Interruption MS-DOS INT 231H

La fonction 0AH de l'interruption INT 21H permet d'installer les caractères tapés au clavier dans la zone d'adresse DS:DX. Ce buffer se remplit tant que le caractère RETURN n'est pas tapé et la longueur maximale de la chaîne n'est pas atteinte; sinon la fonction se termine.

AH ← 02 (N° de la fonction ou du service)

DS:[DX+0] ← Nombre de caractères à lire (paramètres d'entrées).
DS:[DX+1] ← Nombre de caractères lus sans le RETURN (paramètres de sortie).
DS:[DX+1] ← Code ASCII du premier caractère lu.
DS:[DX+ (n+1)] ← Code ASCII du $n^{ème}$ caractère lu.

III Travail demandé

1- Faire un sous programme permettant de réaliser une temporisation de 1 seconde.
2- Ecrire un sous programme permettant d'afficher, à la position du curseur, le message "LE SYSTEME VA ETRE INITIALISE"
3- Ecrire un sous programme permettant de saisir une chaîne de 03 caractères et de les stocker dans un emplacement mémoire de votre choix. En utilisant les sous programmes définis ci-dessus et les interruptions INT 10H et INT 21H, écrire un programme qui réalise les tâches de saisie d'une chaîne de caractères à partir du clavier et de test du contenu de la dite chaîne de la manière suivante:

   a- Si la chaîne ="OUI" alors afficher "LE SYSTEME VA ETRE INITIALISE" pendant 5 secondes puis rebouter la machine a-.
   b- Sinon si la chaîne ="NON" alors quitter le programme.
   c- Sinon aller à a-.

Indication: Utiliser les pseudos-instructions.

IV Manipulation

1- Testez séparément les différents sous programme en mode simulation.
2- Exécutez le programme complet en mode privilégié.

PARTIE III

I OBJECTIFS

Il s'agit de concevoir un programme de conversion ASCII →Binaire dont l'utilité est la gestion de nombres réels dans un contexte PC.

II Calcul de la valeur d'un nombre réel

Une chaine de caractères située à l'adresse BUFFER représente notre variable d'entrée. La chaine est constituée uniquement des codes ASCII correspondant aux caractères à signification numérique décimale. 30H à 39H correspondant respectivement aux chiffres '0' à '9'. 2EH correspondant au point. La longueur totale de cette chaine est la même pour tous les cas de figure, et égale à dix (10) caractères ASCII, y compris le point. Il est important de noter que ce dernier, en vue d'une exploitation la plus générale de votre programme, peut ne pas figurer dans la variable d'entrée. Exemples de cas:

456.767819,
2593411812,
.236543210,
0.12347890,
123456789., etc.

La variable de sortie doit correspondre à l'un des formats de représentation binaire des nombres réels de votre choix, dans les limites de 3 formats suivants:

- Format à virgule fixe dont la taille (nombre de bits) reste à définir.
- Format à virgule flottante dont la taille de l'exposant et mantisse restent à définir.
- Format IEEE 754 qui est aussi à définir.

Elle doit avoir pour valeur la chaine ASCII d'entrée et située à l'adresse VALEUR

III Travail demandé

Dans une vue globale de votre programme, ce dernier doit être composé d'une procédure de saisie des données, une procédure de conversion, ainsi qu'une procédure d'affichage de la valeur convertie, de manière que cette dernière offre un moyen immédiat d'accorder une signification au lecteur (il pourra reconnaitre la valeur affichée sans peine). Pour ce faire:

1- Utiliser l'interruption INT 16H du BIOS pour saisie par le clavier les 10 caractères décimaux représentant la variable d'entrée. La saisie selon votre choix peut se faire avec ou sans écho.
2- La conversion réalisée, utiliser l'interruption INT 10H du BIOS pour afficher sur écran la représentation réels binaire. Une conversion binaire → ASCII répondant uniquement au besoin d'afficher les '1' et '0' au moyen de l'interruption INT 10H doit être prévue dans cette étape.
3- Pour les plus forts. Êtes-vous en mesure de procéder à la conversion inverse? Format réel binaire → représentation Ascii du nombre réel afin d'offrir un moyen facile de vérification de la valeur convertie.

TRAVAUX PRATIQUES N°6

ENTREES ET SORTIES PARALLELES

PARTIE I: FEUX TRICOLORES

I Objectifs

Les objectifs escomptés de ce TP sont la compréhension du fonctionnement des feux tricolores et leur programmation à travers la gestion des entrées et sorties parallèles du microprocesseur 8086. Ce travail est basé sur les données de l'exercice 1 du TD N°7 et sur la solution de l'exercice 19 du TD N°5. Quant à sa mise en œuvre, elle fait appel à l'environnement de l'émulateur EMU8086.

II Introduction

L'émulateur EMU8086 a 7 périphériques virtuels (vous pouvez les voir dans le menu "Virtual Devices" de l'émulateur) que nous pouvons contrôler par programme à travers des ports d'E/S virtuels. Pour réaliser l'émulation des E/S, l'EMU8086 utilise le fichier c:\emu8086.io. Les données de ce fichier représentent les informations transitant par ces périphériques ou ports. Le Port d'adresse 0 correspond à l'octet 0 du fichier, le port d'adresse 4 correspond à l'octet 4, ...etc. Un de ces périphériques s'appelle "Traffic Lights" ou feux tricolores. Nous utiliserons ce périphérique pour réaliser la gestion des feux tricolores de croisement de l'exercice 1 du TD N° 6 qui est repris dans la Figure 1.

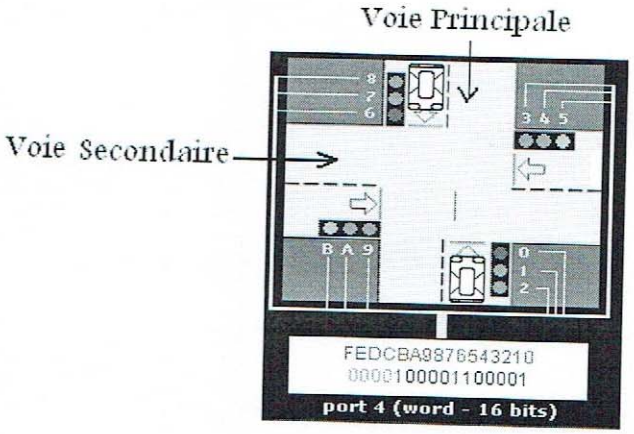


Fig.1 Gestion des feux tricolores d'un croisement

L'EMU8086 utilise le port 4 pour contrôler les feux de la figure ci-dessus avec un mot de 16 bits. Il y a 12 lampes codées: 0123456789AB comme le montre le tableau suivant:

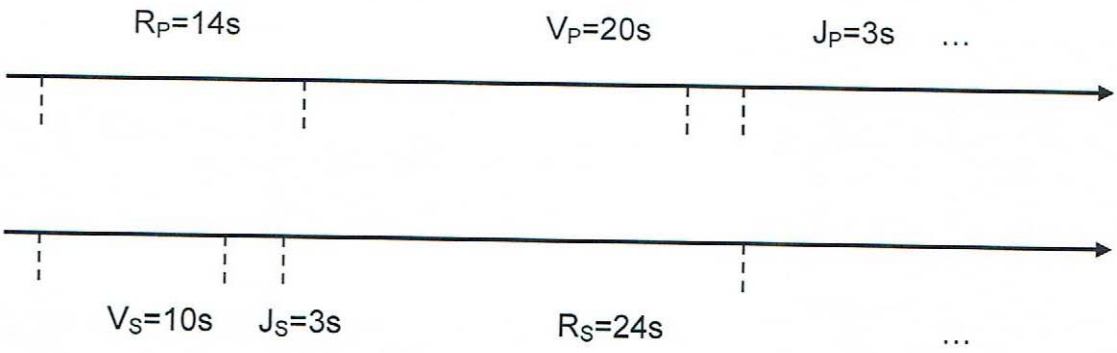
Couleur	Lampes			
Rouge	0	3	6	9
Jaune	1	4	7	A
Vert	2	5	8	B

Chaque lampe est contrôlée par un bit du port d'E/S. Les lampes sont allumées si leurs bits de commande sont à 1 e éteintes si leurs bits respectifs sont à 0. Par exemple, pour allumer toutes les lampes vertes seulement (ce qui constitue un problème pouvant entrainer des risques d'accidents), nous utilisons la séquence d'instructions suivante:

```
MOV AX, 0000100100100100B OU MOV AX, 0924H
OUT 04H, AX
```

Pour activer l'interface graphique des feux tricolores, il faut aussi insérer la commande suivante au début du programme: #start=traffic_lights.exe#
L'exercice 1 du TD N°7 stipule que les feux de même couleur appartenant à la même voie sont contrôlés par la même séquence (voir le chronogramme ci-dessous). Par exemple, les deux feux rouges de la voie principale s'allument en même temps. D'autre part, il y a aussi deux modes de fonctionnement: jour et nuit.

Le fonctionnement jour se fait selon le chronogramme suivant:



Avec:

Vp : feu vert de la voie principale.
Vs : feu vert de la voie secondaire.
Jp : feu jaune de la voie principale.
Js : feu jaune de la voie secondaire.
Rp : feu rouge de la voie principale.
Rs : feu rouge de la voie secondaire.

Dans le mode de fonctionnement nuit, seules les lampes jaunes clignotent.
Par ailleurs, nous disposons de deux modes de fonctionnement: Mode manuel et mode automatique. En fonctionnement normal, le système est en mode automatique. Dans ce cas, les feux travaillent, par exemple, en mode jour de 07 :00 à 18:59 et en mode nuit de 19:00 à 06:59. Sinon, il est possible de faire basculer leur fonctionnement (les feux) en mode manuel. Auquel cas, ils peuvent travailler continuellement en mode jour ou en mode nuit.

III Travail demandé

Donner l'organigramme et le programme en assembleur 8086 permettant de prendre en compte les modes de fonctionnement automatique/manuel et jour/nuit. Pour cela, vous utiliserez les touches a, m, j et n pour désigner les modes automatique, manuel, jour et nuit, respectivement.

Indication: Il est recommandé d'utiliser la fonction 86 h de l'interruption INT 15H (wait function ou délai) du BIOS car elle permet de programmer de façon indépendante du matériel, des temporisations plus fines que celles obtenues en utilisant l'interruption INT 08H qui renvoie le temps, dans le meilleur des cas, en centième de secondes. L'exemple suivant permet de générer une temporisation de 5 secondes.

```
MOV CX, 004CH   ; 004C4B40H = 5, 000,000 MICROSECONDES
MOV DX, 4B40H
MOV AH, 86H
INT 15H
```

Indication: Pour le mode automatique, il est recommandé d'utiliser la solution de l'exercice 12 du TD N°5 qui permet de générer et d'afficher l'heure sous le format HH:MM:SS:NN.

;PROGRAMME DE TEST DES 'TRAFFIC LIGHTS' (COPIER ET COLLER POUR TESTER)

```
#START=TRAFFIC_LIGHTS.EXE#

CODE SEGMENT
                ASSUME CS:CODE, DS:CODE, ES:CODE, SS:CODE
BEGIN:          JMP START
                ; AJOUTER LES  DEFINITIONS ICI
START:          MOV AX,CODE
                MOV DS,AX
                MOV ES,AX
```

```
        MOV SS,AX
        XOR AX,AX
        ; 2 GREEN 2RED
BOUCLE: MOV AX, 030CH; 0000 0011 0000 1100
        OUT 04H, AX
        ; WAIT 5 SECONDS (5 MILLIONS MICROSECONDS)
        MOV CX, 004CH
        MOV DX, 4B40H
        MOV AH, 86H
        INT 15H
        ; ALL YELLOW
        MOV AX, 0492H; 0000 0100 1001 0010
        OUT 04H, AX
        ; WAIT 5 SECONDS (5 MILLION MICROSECONDS)
        MOV CX, 004CH
        MOV DX, 4B40H
        MOV AH, 86H
        INT 15H
        ; 2 RED 2 GREEN
        MOV AX, 0861H; 0000 1000 0110 0001
        OUT 04H, AX
        ; WAIT 5 SECONDS (5 MILLION MICROSECONDS)
        MOV CX, 004CH
        MOV DX, 4B40H
        MOV AH, 86H
        INT 15H
        ; ALL YELLOW
        MOV AX, 0492H ; 0000 0100 1001 0010B
        OUT 04H, AX
        ; WAIT 5 SECONDS (5 MILLION MICROSECONDS)
        MOV CX, 004CH
        MOV DX, 4B40H
        MOV AH, 86H
        INT 15H
        LOOP BOUCLE
        HLT
CODE    ENDS
        END START
```

PARTIE II: INTERFACE PARALLELE CENTRONICS

I Objectifs

Comprendre et utiliser les interruptions BIOS INT 16H et INT 17H pour la gestion du clavier et celle de l'interface parallèle Centronics, respectivement.

II Programmation

Ecrire un programme en Assembleur 8086 qui imprime votre nom dès que vous appuyer sur la touche 'P' en utilisant les interruptions BIOS INT 16H (Clavier) et INT 17H (Imprimante). Ce programme doit permettre de lire le clavier au moyen de l'interruption INT 16H et vérifier la réception de la lettre 'P' et de vérifier l'état de l'imprimante et imprimer votre nom au moyen de l'interruption INT 17H.

TRAVAUX PRATIQUES N°7

ENTREES ET SORTIES SERIELLES

I Objectifs

Comprendre et utiliser l'interruption BIOS INT 14H pour la gestion de l'interface RS 232 C.

II Programmation de la RS 232

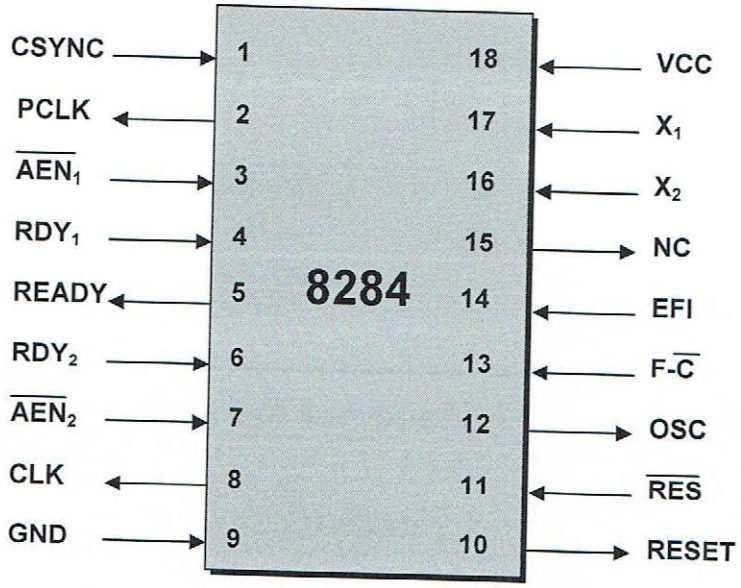
Ecrire un programme en Assembleur 8086 permettant l'émission et la réception d'un message personnalisé entre deux PC à travers l'interface sériel RS 232.

Indication: Utiliser l'interruption INT 14H.

ANNEXE N° 1

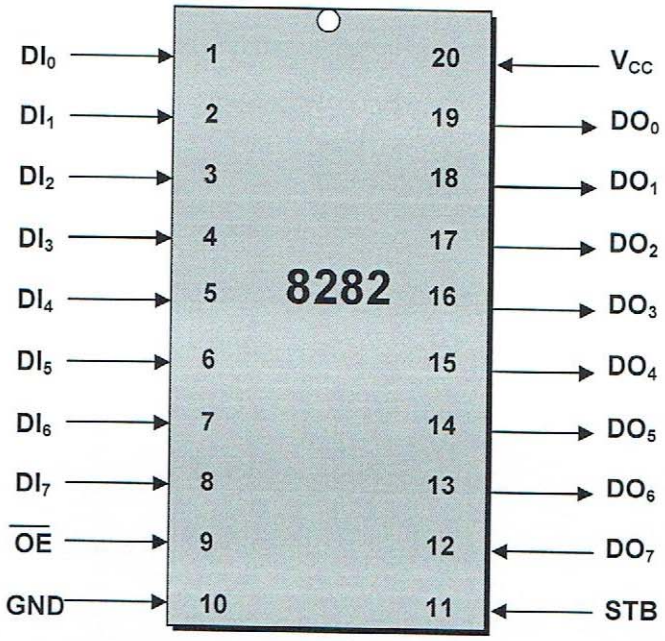
CIRCUITS ANNEXES DU 8086

A.I Circuit générateur d'horloge (clock generator): 8284



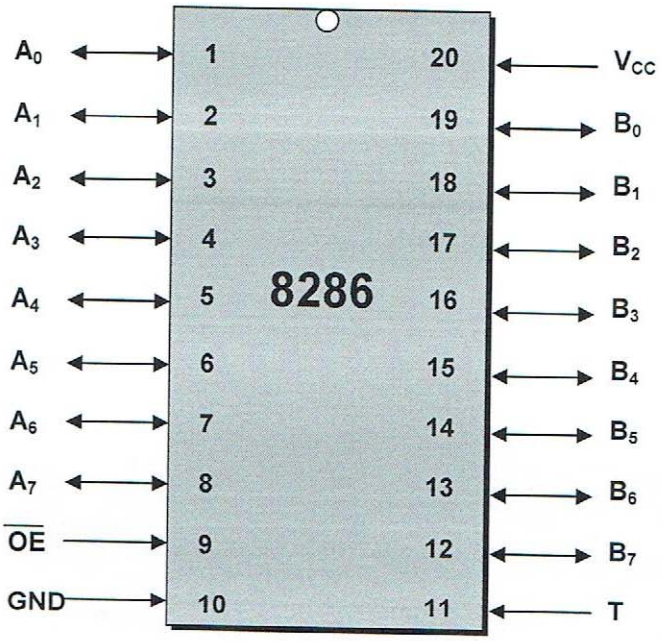
Broche	Direction	Description
F-$\overline{C}$	Entrée	0→Source de fréquence pilotée par un quartz branché sur X_1 et X_2 (3 x horloge du 8086) 1→Source signal TTL de cycle symétrique branché sur EFI (3 x horloge du 8086).
OSC	Sortie	Délivre une fréquence égale à celle du quartz pour alimenter les EFI d'autres 8284 (existe sur le connecteur d'extension).
CLK	Sortie	Horloge du 8086 et son coprocesseur du même bus local (rapport cyclique 33%).
PCLK	Sortie	CLK/2 (Rapport cyclique 50%) pour les périphériques comme le 8253, par exemple.
CSYNC	Entrée	Quand CSYNC passe de 1 à 0, les horloges CLK et PCLK démarrent en phase
$\overline{RES}$	Entrée	Commande la sortie RESET du 8086. Elle peut être utilisée comme RESET général du système.
READY	Sortie	Destinée aux périphériques ne pouvant pas effectuer de transfert à la cadence du 8086
RDY_1 et RDY_2	Entrées	Permettent l'utilisation de deux périphériques.
$\overline{AEN_1}$ et $\overline{AEN_2}$	Entrées	Autorisent l'accès aux entrées RDY_1 et RDY_2.

A.II Circuit capteur d'adresses 8 bits (8 bits address latch): 8282/8283



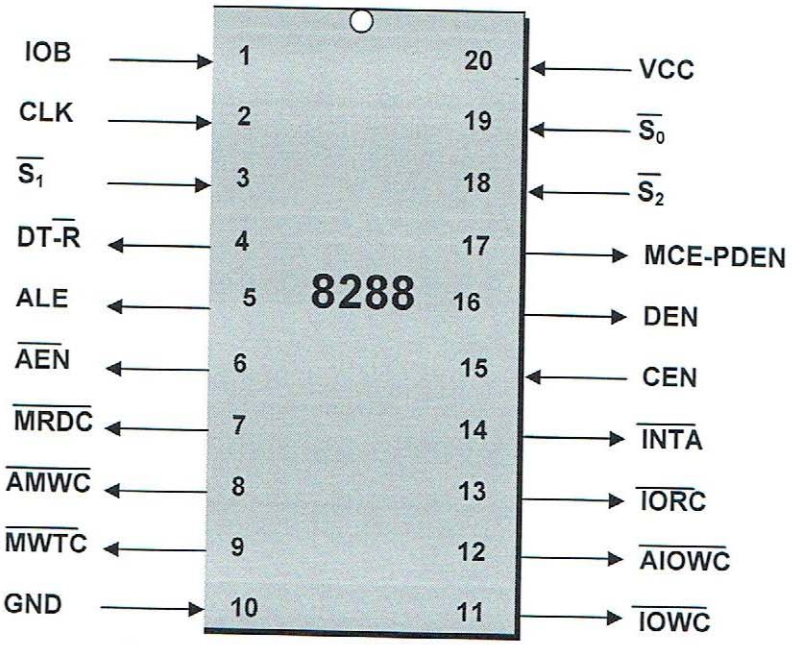
Broche	Direction	Description
$\overline{OE}$	Entrée	Permet de remplir la fonction d'isolement du bus local s'il y a plusieurs maîtres. Si le système ne comprend que le coprocesseur de calcul sur le bus local, elle est mise à la masse.
STB	Entrée	Reliée au ALE (8086 en mode minimum ou 8288 au mode maximum)

A. III Transmetteur bidirectionnel 8 bits (8 bits transceiver): 8286/8287



Broche	Direction	Description
$\overline{OE}$	Entrée	Permet de mettre le 8286/8287 à l'état de haute impédance (OE=1). Elle est reliée au $\overline{DEN}$ du 8086 en mode minimum et au DEN du 8288 en mode maximum.
T	Entrée	0 → A_0-A_7← B_0-B_7 Reception de données 1 → B_0-B_7←A_0-A_7 Transmission de données T est reliée à: DT-$\overline{R}$ du 8086 en mode minimum DT-$\overline{R}$ du 8288 en mode maximum

A. IV Contrôleur de bus (Bus controller): 8288



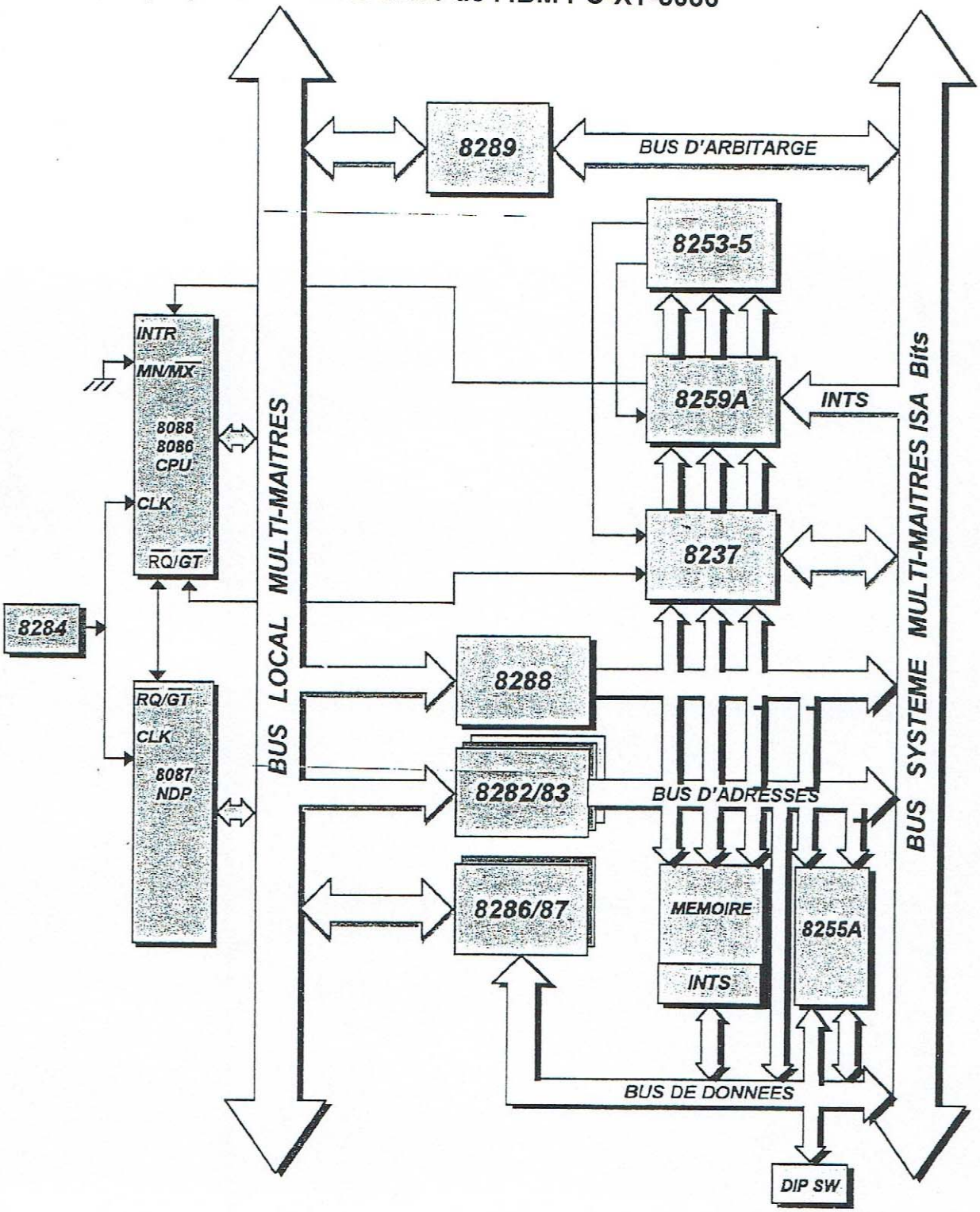
Broche	Direction	Description
CLK	Entrée	Reliée au CLK du 8284.
$\overline{S_0}$, $\overline{S_1}$ et $\overline{S_2}$	Entrées	Les trois lignes sont à l'état de haute impédance lorsque le bus local est utilisé par un autre maître. A la fin du cycle d'opération, elles reviennent toutes les trois à l'état haut. 000 → Reconnaissance d'une INT *001* → Lecture **I/O** 010 → Ecriture **I/O** 011 → Halt 100 → Recherche d'une instruction 101 → Lecture mémoire 110 → Ecriture mémoire 111 → Etat de repos (pas d'accès externe)
IOB	Entrée	Deux modes de fonctionnements sont possibles suivant IOB 0 → Mode de bus d'entrée/sortie action: - La configuration est multimaitre mais le 8086 partage des ressources communes dans le champ mémoire et le champ des entrées/sorties - L'ensemble des signaux de commandes est mis sous contrôle de $\overline{AEN}$. - $\overline{AEN}$=1 toutes les commandes sont à l'état de haute impédance - Le 8288 laisse écouler 105 ns avant de fournir les signaux d'entrées/sorties et mémoire. L'abritage est ainsi possible pour les deux champs - DEN est utilisée pour les entrées/sorties

Broche		Description
IOB (suite)		- Quand $\overline{AEN}$=1, MCE (Sortie) permet d'adresses un contrôleur d'interruption (8259A) esclave sur le bus local et le bus système 1➔ Mode de bus système Action: - La configuration est multimaitre mais le 8086 à son propre bus d'entrées/sorties - $\overline{PDEN}$ (sortie), combiné avec DT/$\overline{R}$ (sortie) est équivalent à $\overline{DEN}$ (sortie) pour les entrées/sorties - $\overline{AEN}$ (entrée): n'a pas d'effet sur $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$ et $\overline{INTA}$ (sorties) qui son toujours validées - Le 8288 laisse écouler 105 ns avant de fournir les signaux de commandes des mémoires $\overline{MRDC}$, $\overline{MRDC}$, $\overline{MWTC}$ et $\overline{AMWTC}$ (sorties)

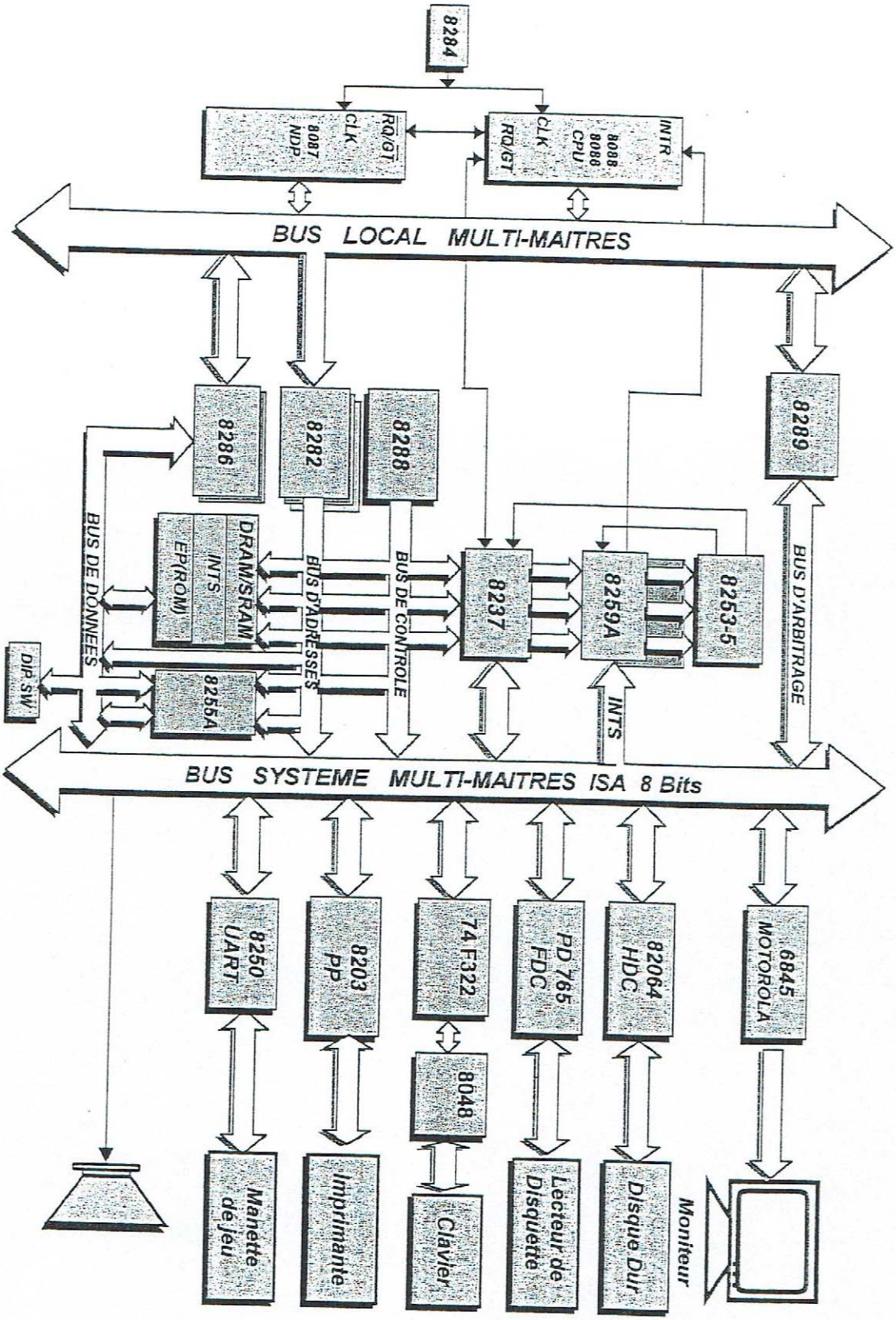
ANNEXE N° 2

SYNOPTIQUES DE QUELQUES IBM PC

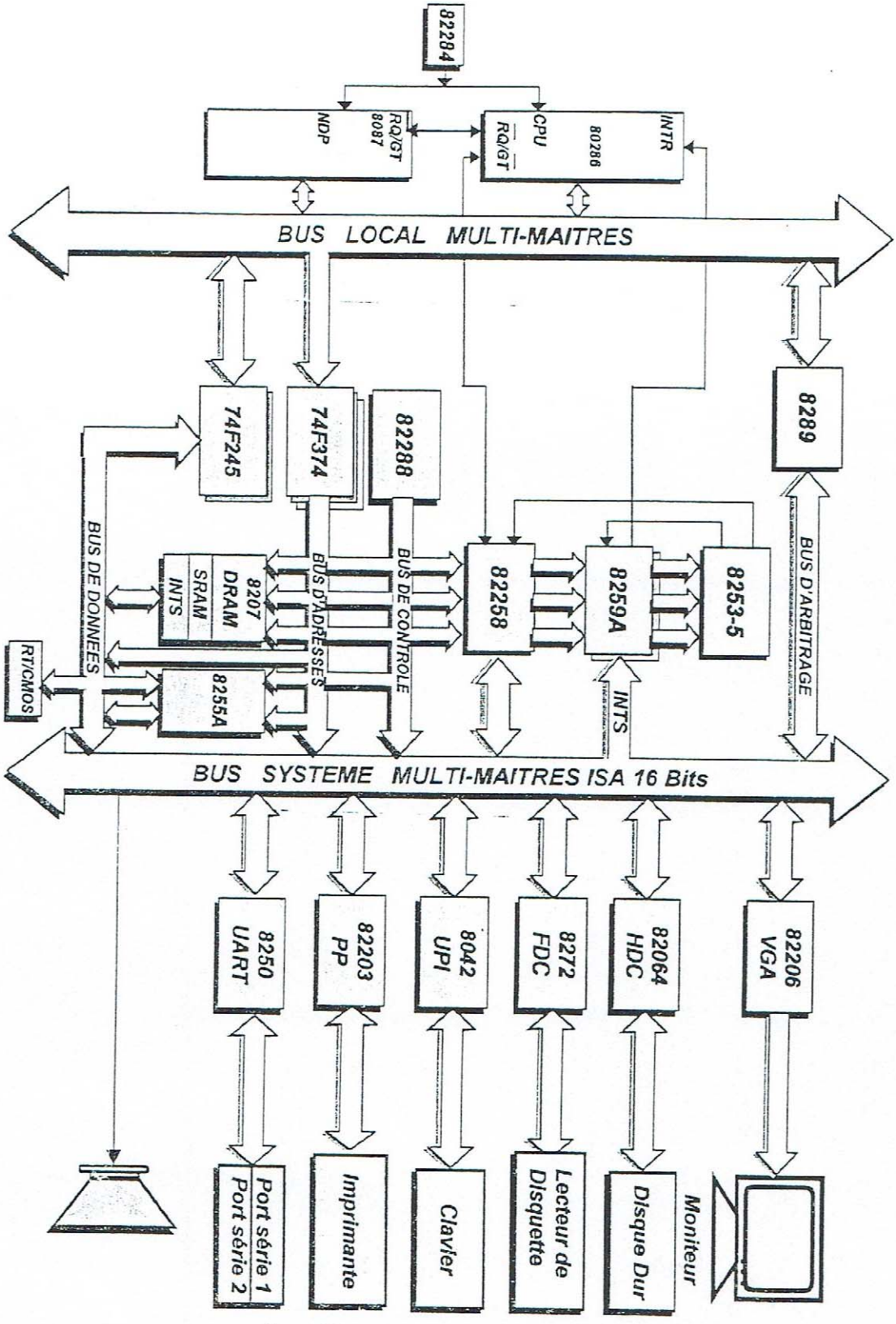
A.I Synoptique de la carte mère de l'IBM PC-XT-8086



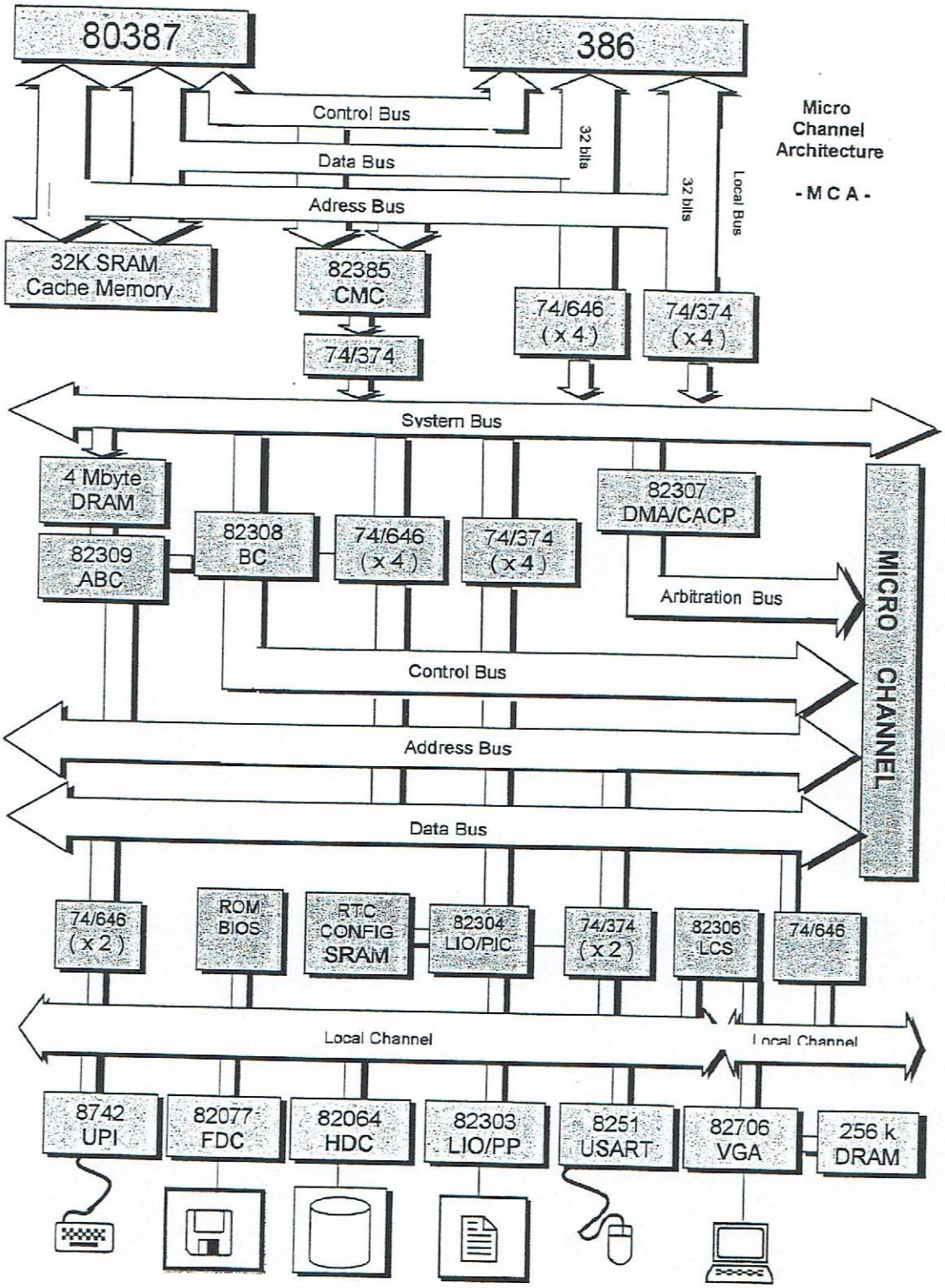
A.II Synoptique de l'IBM PC-XT-8086



A.III Synoptique de l'IBM PC-AT-80286



A.IV Synoptique de l'IBM PC-AT-80386 (Architecture MCA)



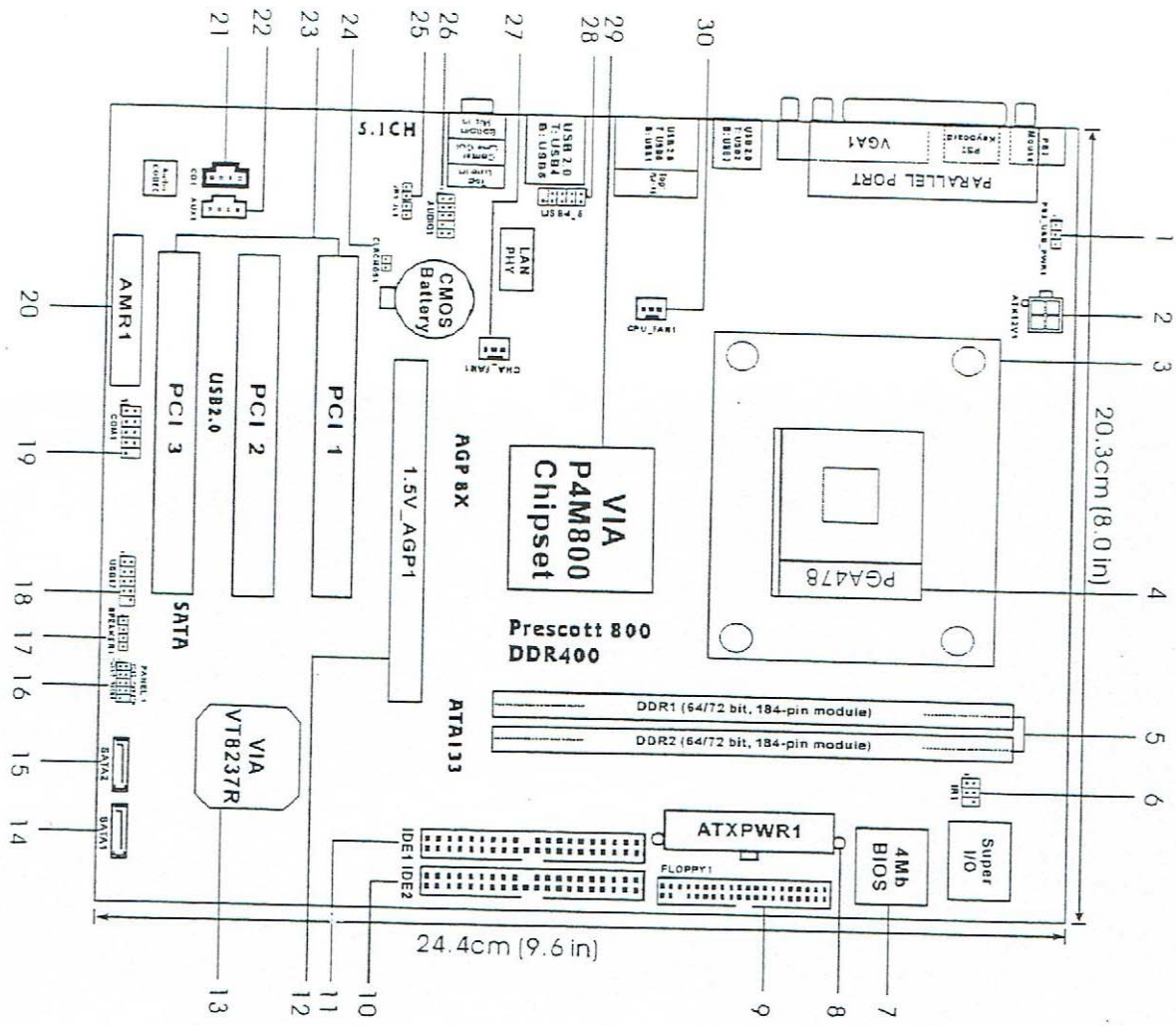
A.IV.1 Circuits de l'IBM PC-AT-80386 (Architecture MCA)

Le bus MCA (Micro Channel Architecture) était proposé par IBM en 1987 en amélioration du bus ISA. Il transmettait les informations par paquets de 32 bits à une fréquence de 33 MHz. Les cartes ISA (Industry Standard Architecture) n'étaient cependant pas compatibles avec le bus MCA. Ce bus surpassait alors les bus ISA et EISA et disposait de caractéristiques voisines du bus PCI (Peripheral Component Interconnect), qui ne verra le jour que quelques années plus tard. Le bus MCA, désavantagé par son incompatibilité avec les anciennes cartes ISA, par son prix et sa complexité technique, n'eut pas le succès qu'IBM espérait et dut abandonner la course au profit de ses bus rivaux ISA et EISA.

Tableau 1 Circuits intégrés de l'IBM PC-AT-386

Circuit	Fonction
80386	Central Processing Unit
80387	Arithmetic Coprocessor
82385	Cache Memory Controller
74/374	Octal D flip-flop with 3-state output
74/646	Octal Bus Transceiver
82309	Dynamic RAM Controller / Address Bus Controller
82307	DMA / Channel Bus Arbitration Controller
82306	Local Channel support
82304	Local I/O Support with 2-integrated 8259A PIC
82303	Local I/O Support with parallel port
8742	Universal Peripheral Port
82077	Floppy Disk Controller
82064	Hard Disk Controller
82706	Video Graphic Array
8251	Universal Synchronous Asynchronous Receiver Transmitter

A.V Synoptique de l'IBM PC-AT-Prescott-800



A.V.1 Composants de l'IBM PC-AT-Prescott-800

Tableau 2 Description des composants de la carte mère VIA P4M8000-P4

Numéro sur la carte	Fonction
1	PS2_USB-PWR1 Jumper
2	ATX 12 V Connector ((ATX12 V1)
3	CPU Heatsink Retention Module
4	CPU socket
5	2 x 184-pin DDR DIMM Slots (DDR1, DDR2; blue)
6	Infrared Module Header (IR1)
7	Flash Memory
8	ATX Power Connector (ATXPWR1)
9	Floppy Connector (FLOPPY1)
10	Secondary IDE Connector (IDE2, Blue)
11	Primary IDE Connector (IDE1, Blue)
12	AGP Slot (1.5V_AGP1)
13	South Bridge Controller
14	Primary Serial ATA Connector (SATA1)
15	Secondary Serial ATA Connector (SATA2)
16	System Panel Header (PANEL1)
17	Chassis Speaker Header (SPEAKER1)
18	USB 2.0 Header (USB67, Blue)
19	Serial Port Connector (COM1)
20	AMR Slot (AMR1)
21	Internal Audio Connector: CD1 (Black)
22	Internal Audio Connector: AUX1 (White)
23	3 x PCI Slots (PCI1-3)
24	Clear CMOS Jumper (CLRCMOS1)
25	JR1 / JL1 Jumpers
26	Front Pannel Audio Header (AUDIO1)
27	Chassis Fan Connector (CHA_FAN1)
28	Shared USB 2.0 Header (USB4-5, Blue)
29	North Bridge Controller
30	CPU Fan Connector (CPU-FAN1)

A.V.2 Connecteurs d'E/S visible sur le panneau arrière

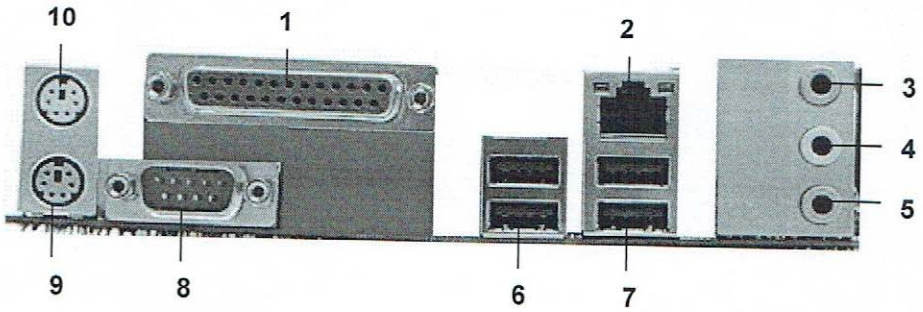


Fig. 1 Connecteurs des E/S (panneau arrière)

Tableau 3 Description des connecteurs des E/S du panneau arrière

Numéro	Fonction
1	Parallel Port
2	RJ-45 Port
3	Line In (Light Blue)
4	Line Out (Lime)
5	Microphone (Pink)
6	Shared USB 2.0 Ports (USB2-3)
7	USB Ports (USB0-1)
8	VGA Port
9	PS/2 Keyboard Port (Purple)
10	PS/2 Mouse (Green)

A.VI Synoptique de l'IBM PC-AT-Core 2 Duo

A.VI.1 Synoptique de la carte mère

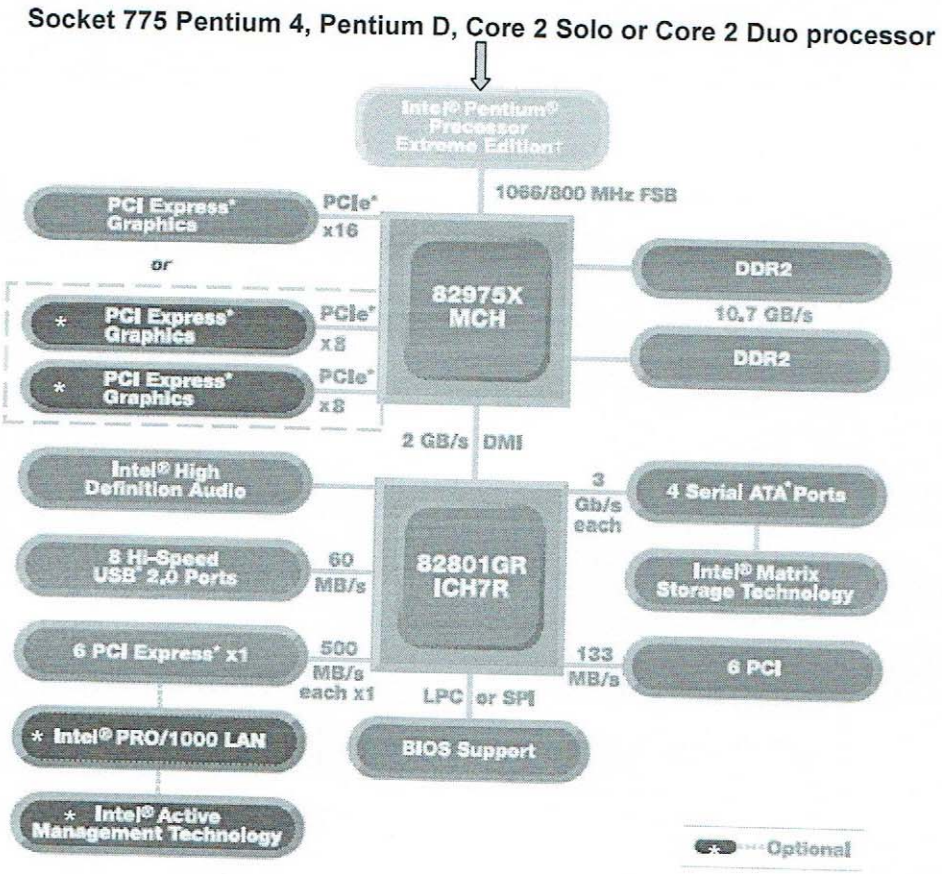


Fig. 2 Synoptique carte mère Core 2 Duo (Plateforme Intel 975X)

A.VI.2 Composants du lecteur de carte multimédia

Le lecteur de carte multimédia (Media Card Reader) est un périphérique optionnel disponible sur certains modèles uniquement. Le Tableau 4 résume l'identification des composants du lecteur de carte multimédia.

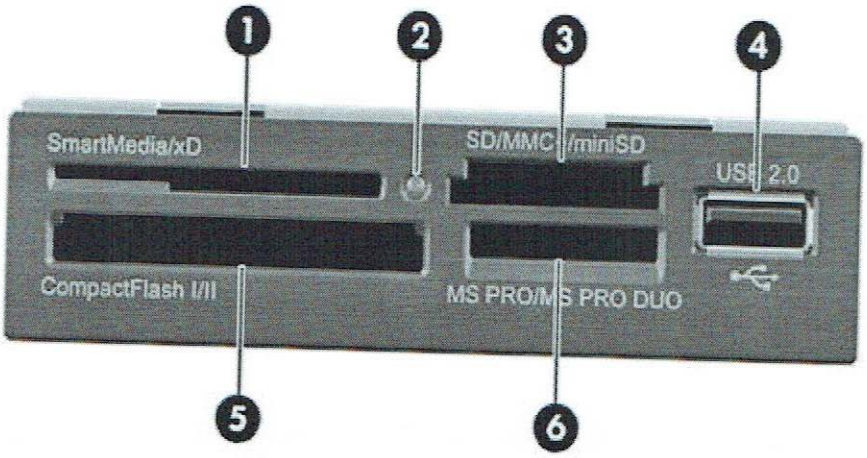


Fig. 3 Lecteur de carte multimédia (Panneau avant)

Tableau 4 Identification des composants du lecteur de carte multimédia

Numéro	Identification
1	Smart Media Card and D-picture Card
2	Media Card Reader Activity Light
3	Secure Digital Card, Multi Media Card+ and mini SD Card
4	USB Port
5	Compact Flash Type 1 and Type 2
6	Memory Stick PRO and Memory Stick PRO DUO

A.VI.3 Supports de module DIMM sur la carte mère

La carte mère est équipée de 4 supports DIMM DDR2-SDRAM, 2 par canal. Ces supports sont identifiés par XMMi (Extended Memory Module i=1, 2, 3 et 4). Les supports XMM1 et XMM2 correspondent au canal A; XMM3 et XMM4 correspondent au canal B. Le Tableau 5 résume la description des 4 supports. La carte mère peut-être équipée d'un maximum de 8 Go mémoire.

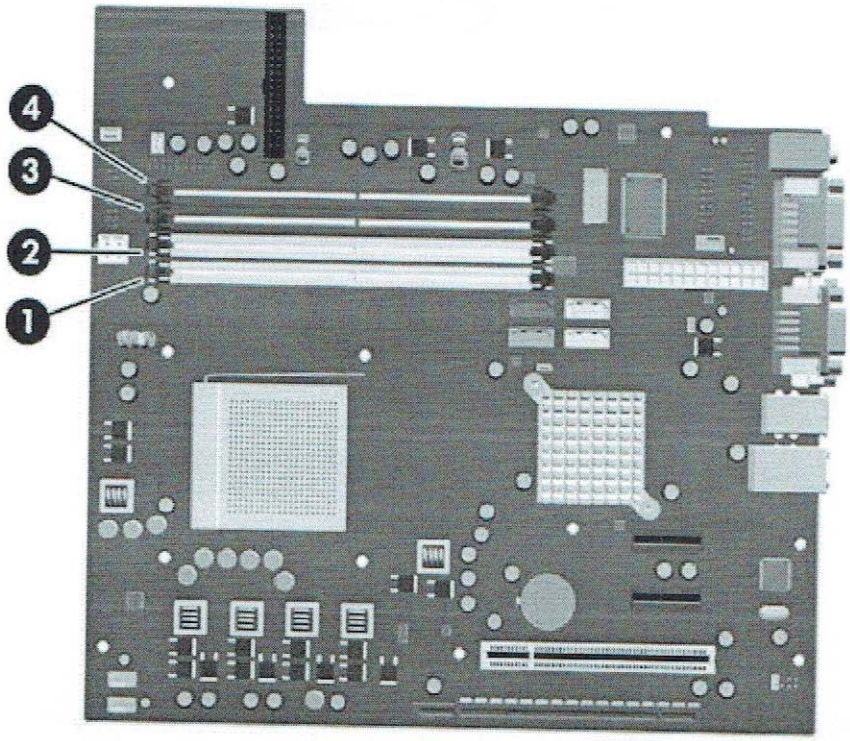


Fig. 4 Emplacement des supports DIMM

Tableau 5 Description des supports DIMM

Numéro	Description	Socket color
1	DIMM socket XMM1, channel A	White
2	DIMM socket XMM2, channel B	White
3	DIMM socket XMM3, channel A	Black
4	DIMM socket XMM4, channel B	Black

A.VI.4 Supports pour cartes d'extension sur la carte mère

Le PC possède un support pour cartes d'extension PCI standard. Il dispose également de 2 connecteurs d'extension PC Express x1 et d'un connecteur PCI Express x16. Il est possible d'installer une carte d'extension PCI Express x1, x4, x8 ou x16 dans le support d'extension x16. Le Tableau 6 résume les emplacements de ces supports.

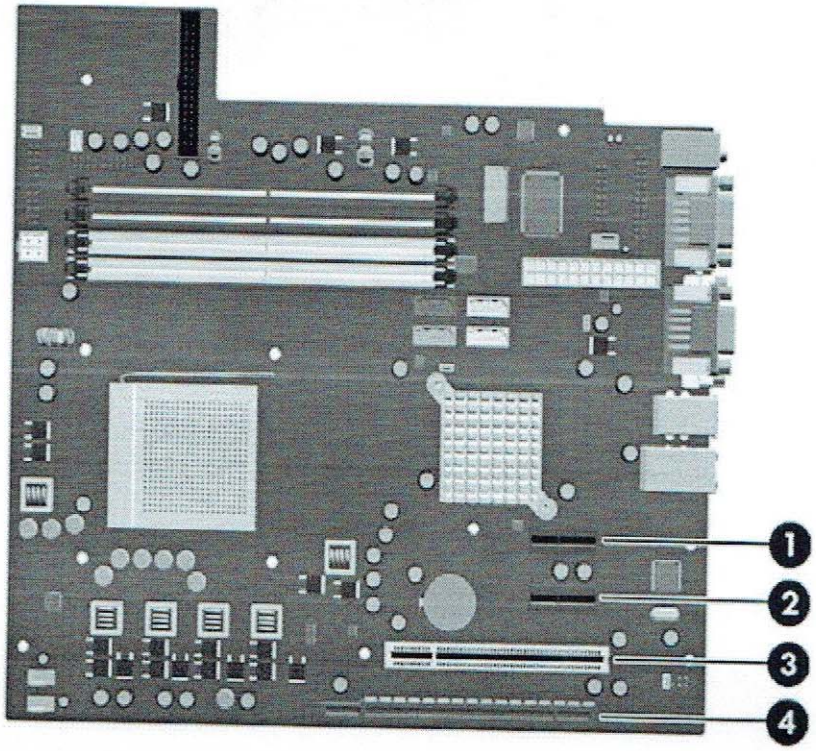


Fig. 5 Emplacements des supports PCI

Tableau 6 Description des supports PCI

Numéro	Description
1	PCI Express x1 expansion slot
2	PCI Express x1 expansion slot
3	PCI expansion slot
4	PCI Express x16 expansion slot

A.VI.5 Connexions de cartes supplémentaires sur la carte mère

Le PC dispose également de plusieurs connexions supplémentaires pouvant prendre en charge des unités de lecteurs de disquette, de carte multimédia et de disque dur. Le Tableau 7 résume ces connexions d'unités supplémentaires.

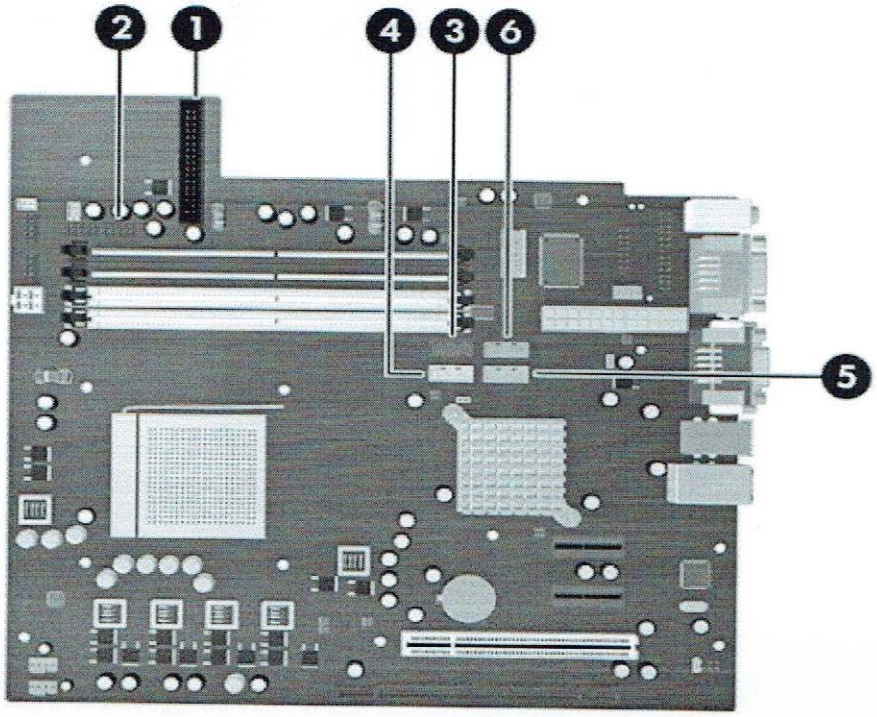


Fig. 6 Emplacements des connexions d'unités supplémentaires

Tableau 7 Description des connexions d'unités supplémentaires

Numéro	Identification	Inscription sur la carte mère	Couleur
1	Diskette drive	FLPY	black
2	Media card reader	MEDIA	black
3	SATA0	SATA0	Dark blue
4	SATA1	SATA1	White
5	SATA2	SATA2	Light blue
6	SATA3	SATA3	Orange

ANNEXE N°3

JEU D'INSTRUCTIONS DU 8086

NOTES

- AL = 8-bit accumulator, AX = 16-bit accumulator, CX = Count register, DS = Data segment and ES = Extra segment.
- Above and Below refer to unsigned values. Greater and Less refer to more positive and more negative signed values.

- If d = 1 then 'to' reg; if d = 0 then 'from' reg.
- If w = 1 then word instruction; if w = 0 then byte instruction.

- If mod = 11 then r/m is treated as a reg field.
- If mod = 00 then DISP = 0*; disp-low and disp-high are absent.
- If mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent.
- If mod = 10 then DISP = disp-low; disp-high.

- If r/m = 000 then EA = (BX) + (SI) + DISP.
- If r/m = 001 then EA = (BP) + (DI) + DISP.
- If r/m = 010 then EA = (BP) + (SI) + DISP.
- If r/m = 011 then EA = (BX) + (DI) + DISP.
- If r/m = 100 then EA = (SI) + DISP.
- If r/m = 101 then EA = (DI) + DISP.
- If r/m = 110 then EA = (BP) + DISP*.
- If r/m = 111 then EA = (BX) + DISP.

DISP follow 2nd byte of instruction (before data if required).
* except if mod = 00 and r/m = 110 then EA = disp-low; disp-high.

- If sw = 01 then 16 bits of immediate data from the operand.
- If sw = 11 then an immediate data byte is sign extended to from the 16-bit operand.
- If sw = 00 or 01 then 8 bits of immediate data from operand.

- If v = 0 then "count"=1; if v = 1 then "count" in (CL) x = don't care.
- Z is used for string primitives comparison with ZF FLAG.

- Segment override prefix defined as **001 reg 110** reg is assigned according to the Table 1.

Tableau 1 Register code assignments

16-bit (w=1)	8-bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Remark: Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS= X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF).

MNEMONIC	DESCRIPTION	INSTRUCTION CODE			
	DATA TRANSFER	76543210	76543210	76543210	76543210
MOV	**= Move**				
	Register/Memory to/ from register	100010dw	mod reg r/m		
	Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w=1
	Immediate to Register	1011wreg	data	data if w=1	
	Memory to Accumulator	1010000w	addr-low	addr-high	
	Accumulator to Memory	1010001w	addr-low	addr-high	
	Register/Memory to Segment Register	10001110	mod 0 regr/m		
	Segment Register to Register/Memory	10001100	mod 0 reg r/m		
POP	**= Pop**				
	Register/Memory	10001111	mod 000 r/m		
	Register	01011reg			
	Segment Register	000reg111			
PUSH	**= Push**				
	Register/Memory	11111111	mod 110 r/m		
	Register	01010reg			
	Segment Register	000reg110			
XCHG	**= Exchanger**				
	Register/Memory with register	1000011w	mod reg r/m		
	Register with accumulator	10010reg			
IN	**= Input from**				
	fixed port	1110010w	port		
	Variable port	1110110w			
OUT	**= Output to**				
	fixed port	1110011w	port		
	variable port	1110111w			
XLAT	**= Translate Byte to AL**	11010111			

DATA TRANSFER		76543210	76543210	76543210	76543210
LEA	= Load EA to register	10001101	mod reg r/m		
LDS	= Load pointer to DS	11000101	mod reg r/m		
LES	= Load pointer to ES	11000100	mod reg r/m		
LAHF	= Load AH with Flags	10011111			
SAHF	= Store AH into Flags	10011110			
PUSHF	= Push Flags	10011100			
POPF	= Pop Flags	10011101			

MNEMONIC	DESCRIPTION	INSTRUCTION CODE			
	ARITHMETIC	76543210	76543210	76543210	76543210
ADD	**= Add** Register/Memory with Register to Either Immediate to Register/Memory Immediate to Accumulator	000000dw	mod reg r/m		
		100000sw	mod 000 r/m	data	data if s: w=01
		0000010w	data	data if w=1	
ADC	**= Add with Carry** Register/Memory with Register to Either Immediate to Register/Memory Immediate to Accumulator	000100dw	mod reg r/m		
		100000sw	mod 010 r/m	data	data if s: w=01
		0001010w	data	data if w=1	
INC	**= Increment** Register/Memory Register	1111111w	mod 000 r/m		
		01000reg			
AAA	**= ASCII Adjust for Add**	00110111			
DAA	**= Decimal Adjust for Add**	00100111			
SUB	**= Subtract** Register/Memory with Register to Either Immediate from Register/Memory Immediate from Accumulator	001010dw	mod reg r/m		
		100000sw	mod 101 r/m	data	data if s: w=01
		0010110w	data	data if w=1	
SSB	**= Subtract with Borrow** Register/Memory with Register to Either Immediate to Register/Memory Immediate to Accumulator	000110dw	mod reg r/m		
		100000sw	mod 011 r/m	data	data if s: w=01
		0001110w	data	data if w=1	
DEC	**= Decrement** Register/Memory Register	1111111w	mod 001 r/m		
		01001reg			
NEG	**= Change sign**	1111011w	mod 011 r/m		

ARITHMETIC		76543210	76543210	76543210	76543210
CMP	= Compare Register/Memory and Register Immediate with Register/Memory Immediate with Accumulator	001110dw	mod reg r/m		
		100000sw	mod 111 r/m	data	data if s: w=01
		0011110w	data	data if w=1	
AAS	= ASCII Adjust for Subtract	00111111			
DAS	= Decimal Adjust for Subtract	00101111			
MUL	= Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL	= Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM	= ASCII Adjust for Multiply	11010100	00001010		
DIV	= Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV	= Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD	= ASCII Adjust for Divide	11010101	00001010		
CBW	= Convert Byte to Word	10011000			
CWD	= Convert Word to Double Word	10011001			

MNEMONIC	DESCRIPTION	INSTRUCTION CODE			
	LOGIC	76543210	76543210	76543210	76543210
NOT	= Invert	1111011w	mod 010 r/m		
SHL/SAL	= Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR	= Shift Logical Right	110100vw	mod 101 r/m		
SAR	= Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL	= Rotate Left	110100vw	mod 000 r/m		
ROR	= Rotate Right	110100vw	mod 001 r/m		
RCL	= Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR	= Rotate Through Carry Flag Right	110100vw	mod 011 r/m		
AND	= AND Register/Memory and Register to Either Immediate to Register/Memory Immediate to Accumulator	001000dw	mod reg r/m		
		1000000w	mod 100 r/m	data	data if w=1
		0010010w	data	data if w=1	
TEST	= And Function to Flags, No Result Register/Memory and Register Immediate Data and Register/Memory Immediate Data and Accumulator	1000010w	mod reg r/m		
		1111011w	mod 000 r/m	data	data if w=1
		1010100w	data	data if w=1	
OR	= Or Register/Memory and Register to Either Immediate to Register/Memory Immediate to Accumulator	000010dw	mod reg r/m		
		1000000w	mod 001 r/m	data	data if w=1
		0000110w	data	data if w=1	

LOGIC		76543210	76543210	76543210	76543210
XOR	**= Exclusive Or**				
	Register/Memory and Register to Either	001100dw	mod reg r/m		
	Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w=1
	Immediate to Accumulator	0011010w	data	data if w=1	

MNEMONIC	DESCRIPTION	INSTRUCTION CODE
	STRING MANIPULATION	76543210
REP	**= Repeat**	1111001z
MOVS	**= Move Byte/Word**	1010010w
CMPS	**= Compare Byte/Word**	1010011w
SCAS	**= Scan Byte/Word**	1010111w
LODS	**= Load Byte/Word to AL/AX**	1010110w
STOS	**= Store Byte/Word from AL/AX**	1010101w

MNEMONIC	DESCRIPTION	INSTRUCTION CODE		
	TRANSFER CONTROL	76543210	76543210	76543210
CALL	**= Call**			
	Direct within Segment	11101000	disp-low	disp-high
	Indirect within Segment	11111111	mod 010 r/m	
	Direct Intersegment	10011010	offset-low	offset-high
			seg-low	seg-high
	Indirect Intersegment	11111111	mod 011 r/m	
JMP	**= Unconditional Jump**			
	Direct within Segment	11101001	disp-low	disp-high
	Indirect within Segment-short	11101011	disp	
	Indirect within Segment	11111111	mod 100 r/m	
	Direct Intersegment	11101010	offset-low	offset-high
			seg-low	seg-high
	Indirect Intersegment	11111111	mod 101 r/m	
RET	**= Return from CALL**			
	Within Segment	11000011		
	Within Segment Adding Immediate to SP	11000010	data-low	data-high
	Intersegment	11001011		
	Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ	**= Jump on Equal/Zero**	01110100	disp	
JL/JNGE	**= Jump on Less/Not Greater or Equal**	01111100	disp	
JLE/JNG	**= Jump on Less or Equal/Not Greater**	01111110	disp	
JB/JNAE	**= Jump on Bellow/Not Above or Equal**	01110010	disp	
JBE/JNA	**= Jump on Bellow or Equal/Not Above**	01110110	disp	

TRANSFER CONTROL		
JP/JPE	= Jump on Parity/Parity Even	01111010 disp
JO	= Jump on Overflow	01110000 disp
JS	= Jump on Sign	01111000 disp
JNE/JNZ	= Jump on Not Equal/Not Zero	01110101 disp
JNL/JGE	= Jump on Not Less/Greater or Equal	01111101 disp
JNLE/JG	= Jump on Not Less or Equal/Greater	01111111 disp
JNB/JAE	= Jump on Not Below/Above or Equal	01110011 disp
JNBE/JA	= Jump on Not Below or Equal /Above	01110111 disp
JNP/JPO	= Jump on Not Par/Par Odd	01111011 disp
JNO	= Jump on Not Overflow	01110001 disp
JNS	= Jump on Not Sign	01111001 disp
LOOP	= Loop CX Times	11100010 disp

TRANSFER CONTROL		76543210 76543210
LOOPZ/LOOPE	**= Loop While Zero/Equal**	11100001 disp
LOOPNZ/LOOPNE	**= Loop While Not Zero/Equal**	11100000 disp
JCXZ	**= Jump on CX Zero**	11100011 disp
INT	**= Interrupt** Type Specified Type 3	11001101 type 11001100
INTO	**= Interrupt on Overflow**	11001110
IRET	**= Interrupt Return**	11001111

MNEMONIC	DESCRIPTION	INSTRUCTION CODE
	PROCESSOR CONTROL	76543210 76543210
CLC	= Clear Carry	11111000
CMC	= Complement Carry	11110101
STC	= Set Carry	11111001
CLD	= Clear Direction	11111100
STD	= Set Direction	11111101
CLI	= Clear Interrupt	11111010
STI	= Set Interrupt	11111011
HALT	= Halt	11110100
WAIT	= Wait	10011011
ESC	= Escape (to External Device)	11011xxx mod xxx r/m
LOCK	= Bus Lock Prefix	11110000
NOP	= No Operation	10010000

ANNEXE 4

COMMANDES DU SIMULATEUR 8086

Les premières séances de Travaux Pratiques sont effectuées au moyen du logiciel de simulation du 8086 dont nous présentons ci-après les principales commandes.

1- A : Assembler en mémoire

Syntaxe: A [adresse]

Appel de l'assembleur non-symbolique. Faire Ctrl-C pour sortir (quitter).

Remarque: [.] désigne un champ optionnel, c'est-à-dire non obligatoire.

Exemple: A CS:100

2- BASE: Modification de la base de calcul des registres

Syntaxe: BASE reg * HEX.BIN.DEC

Détermine si le contenu des registres sera affiché en hexadécimal, binaire ou décimal, au choix. Lorsqu'on a choisi la base hexadécimale pour les registres 16 bits, le simulateur ne les décomposera pas en deux fois 8 bits. Si l'on inscrit * à la place du "reg", la base sera modifiée pour tous les registres.

***Exemple*:** BASE CH BIN (CH est affiché en binaire).

3- *BIG*: Modifie la taille des fenêtres d'affichage du processeur et du désassemblage.

***Syntaxe*:** BIG

Il s'agit d'une commande de type "bascule", ainsi on alterne entre un affichage grand processeur/petit écran de travail et petit processeur/grand écran de travail.

***Exemple*:** BIG

4- *C* : Comparaison de deux zones mémoire

***Syntaxe*:** C zones adresse

Les différences sont données sous la forme:adr1:Contenuadr1, Contenuadr2:adr2.

***Exemple*:** C DS:100 L 300 CS:100
C 100 200 500

5- *CALC* : Active la calculette

***Syntaxe*:** CALC

Appel la calculette qui effectue les quatre opérations arithmétiques. La calculette peut également être mise en route au cours d'une simulation, grâce à la touche C, si l'on est en pause. La base sera hexadécimale, binaire ou décimale, le choix s'effectuant respectivement par les commandes ^H, ^B et ^D.

Remarque: Quel que soit l'affichage (décimal, hexadécimal ou binaire), la saisie des valeurs se fait toujours en mode décimal.

Les entrées peuvent être écrites sous deux formats:

Soit N1 opérateur N2, où N1 et N2 sont des entiers écrits dans la base de calcul choisie. et opérateur sera bien entendu, une de ces quatre opérations: +, -, /, *.

Soit N1, où N1 est un entier écrit dans la base choisie. Ce format est réservé aux conversions. Les valeurs négatives seront données sur 32 bits en complément à 2. La valeur positive maximale est: 7FFFFFFFH, la valeur négative maximale est 80000000H.

Erreurs:

Overflow (Dépassement de capacité): Le résultat d'une opération est supérieur à 7FFFFFFFH ou inférieur à 80000000H.

Divide by zero (Division par zéro): L'opérande N2 dans la division est égal à 0.

***Exemple*:** CALC

6- CASE: Modification minuscules/majuscules

***Syntaxe*:** CASE

Redessine l'écran et passe des majuscules aux minuscules ou vice-versa.

***Exemple*:** CASE

7- CLEAR: Vide les pointeurs

***Syntaxe*:** CLEAR pointeur

Vide un pointeur. La partie "F" du nom du pointeur peut être omise.

***Exemple*:** CLEAR TF

8- *D* : Affiche le contenu de la mémoire

***Syntaxes*:** D [adresse] ou
D zone

Format 1: Affiche 128 octets à partir de l'adresse X
Format 2: Affiche tous les octets dans la zone X

***Exemples*:** D CS:100
D CS:100 L 500

9- DSCRN : Redessine l'écran en fonction des états DMEM, BIG et CASE.

***Syntaxe*:** DSCRN

Exemple: DSCRN

10- DMEM: Afficher la mémoire

Syntaxe: DMEM adresse.OFF [W]

Ouvre ou ferme une fenêtre mémoire à l'adresse X. le paramètre OFF entraîne la fermeture de la fenêtre mémoire et l'agrandissement de la fenêtre de

désassemblage. Le paramètre W provoque l'affichage de la mémoire mot par mot, sinon elle sera affichée octet par octet. On peut afficher deux zones distinctes de 16 octets. Pour afficher la deuxième zone, retaper DMEM.

Exemples: DMEM SS:FFF0 W
DMEM OFF

11- E: Ecrire des données en mémoire

Syntaxes: E adresse
E adresse liste

Format 1: Entraine une écriture à une adresse mémoire. Le tiret (-) permet de décrémenter le compteur d'adresses. L'incrémentation se fait grâce à la touche ENTER, sans changer la valeur de l'adresse. Ctrl-C met fin à cette opération.

Format 2: Ecrit une liste d'octets à partir de l'adresse N.

Exemples: E CS:0100
E CS:0100 01 02 03 04

12- ERASE: Efface l'écran

Syntaxe: ERASE

Effacement de l'écran. Utiliser DSCRN pour rafraichir (redessiner) l'écran.

Exemple: ERASE

13- G: Go, donne la main au 8086

Syntaxe: G [adresse]

Donne la main au 8086 et permet de lancer vos programmes ou vos applications. La main est rendue au Simulateur lorsque CS:IP=adresse (que vous avez indiquée dans la commande Go) ou lorsque votre programme rencontre BRK (INT 03H, code opératoire CCH) ou encore lorsque votre programme prendra fin normalement (Appel système 20H). Le message

'8086 Control' s'affiche sur la ligne d'erreurs. Vous pouvez spécifier jusqu'à 1à points de rupture (BRK) dans la commande Go.

Remarque: la commande Go ne permet pas de transmettre un trap au registre pointeur du 8086.

Exemples: G 4000
G DS:3333 CS:0101

14- I/O: Input/Output (Entrée/Sortie) port

Syntaxe: I mot / O mot octet

Lit un octet sur le port 'mot'/ Envoie 'octet' dans le mot 'port'. Pour les utilisateurs privilégiés uniquement.

Exemple: I 3F8 / O 3F8 CC

15- INTR: Définition d'une priorité d'interruption

Syntaxe: INTR octet

Définit le type d'interruption qui sera provoquée lorsque le simulateur répond à un signal sur la broche INTR.

Exemple: INTR 9

16- LOAD: Chargement d'un de de plusieurs fichiers

Syntaxe: LOAD nom de fichier [nom de fichier]

Charge en mémoire le fichier nommé et crée un préfixe de segment programme en CS:0000. Si on spécifie un deuxième nom de fichier, ce nom est inscrit dans le FCB situé à CS:005C. BX:CX contient la longueur du fichier. Le DTA est mis à CS:0080. Si le nom de fichier a une extension .EXE, le fichier est alors rechargé et les registres CS, SS, IP et SP prennent les valeurs des entêtes. Les extensions des fichiers doivent être indiquées. les noms de fichiers peuvent être spécifiés au moment de charger SIM. C> SIM nom de fichier 1 nom de fichier 2. Le résultat est tel que le fichier 1 est placé en mémoire et le fichier 2 est tarité comme un FCB situé à CS:005C.

Exemples: LOAD PROG13.DEM
LOAD WS.COM TEST.COM
C> SIM WS.COM TEST.TXT

17- M: Déplace une zone en mémoire

Syntaxe: M zone adresse
Place les octets de 'zone' à partir de 'adresse'

Exemples: M CS:100 L 500 SS:1000
M 100 500 800

18- MODE: Mode de simulation

Syntaxe: STEP n où n=0, 1, 2, 3 ou 4

Les valeurs de n déterminent le nombre de pauses au cours de l'exécution d'une instruction.

Tableau 1 Effets de STEP n

STEP	Effets
4	Fait une pause après chaque instruction, chaque macro et micro step, chaque lecture/écriture et entre chaque phase de vérification des interruptions.
3	Pause entre les instructions, les macros et micro étapes.
2	Pause entre les instructions et les macros étapes (Mode idéal pour apprendre)
1	Pause entre les instructions uniquement.
0	Pas de pause

Exemple: STEP 2

19- NEW: Reprend les valeurs par défaut de SIM.

Syntaxe: NEW

Restaure les valeurs initiales de SIM.

Remarque: Si un nom de fichier a éét associé à SIM (lors du chargement sous DOS), celui-ci est aussi rechargé.

20- PRIV: Commande type bascule pour passer en mode privilégié.

Syntaxe: PRIV

Remarque: En mode privilégié, vous ne pouvez pas écrire en mémoire ni exécuter les commandes Go, SKIP, T et O.

Exemple: PRIV

21- SKIP: Saut d'une instruction CALL ou d'une interruption.
Pour les utilisateurs en mode privilégié uniquement, cette commande établit un break point après l'instruction en cours et donne la main à votre programme. La main est redonnée à SIM dès que CS:IP est égal à l'adresse de l'instruction suivante. Le message '8086 Control' apparait dans la ligne réservée aux messages d'erreur.

Exemple: SKIP

22- Q: Quitter

Syntaxe: Q

Retour au MS-DOS.

Exemple: Q

23- SAVE: Sauvegarde sur disque

Syntaxe: SAVE [nom du fichier]

La sauvegarde sur disque se fait à partir de BX:CX octets à partir de CS:0100. SI le nom de fichier est celui d'un fichier que vous venez de charger en mémoire et auquel vous avez apporté des modifications, il n'est pas nécessaire de spécifier un nom de fichier lors du SAVE.

Remarque: Vous ne pouvez pas faire ces opérations sur un fichier .EXE. Pour y remédier, vous pouvez changer le nom de l'extension avant de la charger en mémoire.

24- SET: Affecte une valeur à un pointeur.

Syntaxe: SET flag

Affecte la valeur '1' (binaire) au pointeur flag.

Exemple: SET I

25- SIM: Lance une simulation 8086

Syntaxe: SIM [mot]

Lance une simulation pour la ou les instructions ([mot] spécifie la zone mémoire à simuler). Si on omet 'mot', une seule et unique instruction est simulée. Si la fenêtre du processeur est réduite, SIM l'agrandit automatiquement pour les besoins de la simulation. Ci-dessous, les commandes actives en cours de simulation.

Tableau 2 Commandes actives en cours de l'exécution de SIM

Touche	Effet
Barre espace	Demande ou annulation de pause.
C	Active la calculette.
1-9 inclus	Détermine la vitesse de la simulation.
^P	Sorite imprimante (chaque instruction).
Escape, ^C	Fin de simulation, retour à l'écran.
T, N et I	Activation des signaux sur les broches d'entrée.

Exemple: SIM 0F00

26- T: (Trap) Trace l'instruction se trouvant dans CS:IP.

Syntaxe: T [mot]

Utilise le Trap flag du 8086 pour l'exécution pas-à-pas d'une seule instruction. Si 'mot' est supérieur à 1, la touche ENTER permet d'exécuter rapidement ce pas-à-pas. La barre espace permet elle, d'exécuter l'instruction pas-à-pas.

Exemple: T FFFF

27- U: Désassemblage en mémoire.

Syntaxe: U [adresse]
U zone

Format 1: Désassemblage, selon la taille de l'écran, de 18 ou 6 octets.
Format 2: Désassemblage de toutes les instructions dans une zone.

Remarques:

1- INT 03H donne BRK
2- LOONE donne LOOPN
3- La fenêtre de désassemblage tronque les instructions supérieures à 33 caractères et le cas échéant place un ( ) dans la colonne à l'extrême droite.

28- REG: Affectation d'un registre.

Syntaxe: REG mot

Affecte 'mot' à 'reg'. Un message 'Range error' apparait si 'reg' est un registre de 8 bits alors que 'mot est supérieur à 256.

Exemple: IP 100

ANNEXE 5

CODE ASCII

Le code ASCII est une norme d'encodage informatique des caractères alphanumériques de l'alphabet latin. Il établit une correspondance entre une représentation binaire des caractères de l'alphabet latin et les symboles et signes qui constituent cet alphabet. Par exemple, le caractère 'a' est associé à '01100001' et 'A' à '01000001'. La norme ASCII permet ainsi à toutes sortes de machines de stocker, analyser et communiquer de l'information textuelle. En particulier, la quasi totalité des ordinateurs personnels et des stations de travail utilisent l'encodage ASCII.

Le codage ASCII est souvent complété par des correspondances supplémentaires afin de permettre l'encodage informatique d'autres caractères, comme les caractères accentués par exemple. Il existe d'autres normes que l'ASCII, comme l'Unicode par exemple, qui présente l'avantage de proposer une version unifiée des différents encodages de caractères complétant l'ASCII mais aussi de permettre l'encodage de caractères autres que ceux de l'alphabet latin. Le codage UTF-8 (USC (Universal Character Set) Transformation Format) de l'Unicode est une extension d'ASCII utilisant le $8^{ème}$ bit.

Il existe deux modes de transmission des fichiers Informatiques. Le mode ASCII et le mode Binaire. Un mauvais choix dans le mode de transmission peut Rendre un fichier inexploitable. En mode ASCII, le logiciel de transmission adressera le code ASCII de chaque caractère. Ce mode est particulièrement destiné à la transmission des fichiers dits "texte" (HTML (Hypertext Markup Language), sources, scripts). En mode Binaire, le logiciel de transmission envoie les bits par paquets. Ce mode convient aux fichiers, comme les exécutables, les images, les sons, etc.

Le choix du mode ASCII / Binaire est souvent laissé à l'appréciation du logiciel de transmission qui déterminera automatiquement le mode approprié en fonction de l'extension ou du contenu du fichier à transmettre. Certaines transmissions sont effectuées en mode hexadécimal. Ce mode de transmission permet de transmettre des fichiers binaires en mode ASCII, c'est à dire en n'utilisant que les 128 premiers caractères de la Table ASCII du Tableau 1.

Tableau 1 Codes ASCII 7 et 8 bits

DECIMAL	HEXADECIMAL	ASCII	CTRL	DECIMAL	HEXADECIMAL	ASCII
0	00	NUL	@	40	28	(
1	01	SOH	A	41	29	)
2	02	STX	B	42	2A	*
3	03	ETX ♥	C	43	2B	+
4	04	EOT ♦	D	44	2C	,
5	05	ENQ ♣	E	45	2D	-
6	06	ACK ♠	F	46	2E	.
7	07	BEL	G	47	2F	/
8	08	BS	H	48	30	0
9	09	HT	I	49	31	1
10	0A	LF	J	50	32	2
11	0B	VT	K	51	33	3
12	0C	FF	L	52	34	4
13	0D	CR	M	53	35	5
14	0E	SO	N	54	36	6
15	0F	SI	O	55	37	7
16	10	DLE	P	56	38	8
17	11	DC1	Q	57	39	9
18	12	DC2	R	58	3A	:
19	13	DC3	S	59	3B	;
20	14	DC4	T	60	3C	<
21	15	NAK §	U	61	3D	=
22	16	SYN	V	62	3E	>
23	17	ETB	W	63	3F	?
24	18	CAN	X	64	40	@
25	19	EM	Y	65	41	A
26	1A	SUB	Z	66	42	B
27	1B	ESC	[	67	43	C
28	1C	FS	\	68	44	D
29	1D	GS	]	69	45	E
30	1E	RS	^	70	46	F
31	1F	US	_	71	47	G
32	20	Space		72	48	H
33	21	!		73	49	I
34	22	"		74	4A	J
35	23	#		75	4B	K
36	24	$		76	4C	L
37	25	%		77	4D	M
38	26	&		78	4E	N
39	27	'		79	4F	O

DECIMAL	HEXADECIMAL	ASCII		DECIMAL	HEXADECIMAL	ASCII
80	50	P		122	7A	z
81	51	Q		123	7B	{
82	52	R		124	7C	|
83	53	S		125	7D	}
84	54	T		126	7E	~
85	55	U		127	7F	DEL
86	56	V		128	80	Ç
87	57	W		129	81	ü
88	58	X		130	82	é
89	59	Y		131	83	â
90	5A	Z		132	84	ä
91	5B	[		133	85	à
92	5C	\		134	86	
93	5D	]		135	87	
94	5E	^		136	88	
95	5F	_		137	89	
96	60	`		138	8A	
97	61	a		139	8B	
98	62	b		140	8C	
99	63	c		141	8D	
100	64	d		142	8E	
101	65	e		143	8F	
102	66	f		144	90	
103	67	g		145	91	
104	68	h		146	92	
105	69	i		147	93	
106	6A	j		148	94	
107	6B	k		149	95	
108	6C	l		150	96	
109	6D	m		151	97	
110	6E	n		152	98	
111	6F	o		153	99	
112	70	p		154	9A	
113	71	q		155	9B	
114	72	r		156	9C	
115	73	s		157	9D	
116	74	t		158	9E	
117	75	u		159	9F	
118	76	v		160	A0	
119	77	w		161	A1	
120	78	x		162	A2	
121	79	y		163	A3	

DECIMAL	HEXADECIMAL	ASCII		DECIMAL	HEXADECIMAL	ASCII
164	A4			206	CE	
165	A5			207	CF	
166	A6			208	D0	
167	A7			209	D1	
168	A8			210	D2	
169	A9			211	D3	
170	AA			212	D4	
171	AB			213	D5	
172	AC			214	D6	
173	AD			215	D7	
174	AE			216	D8	
175	AF			217	D9	
176	B0			218	DA	
177	B1			219	DB	
178	B2			220	DC	
179	B3			221	DD	
180	B4			222	DE	
181	B5			223	DF	
182	B6			224	E0	
183	B7			225	E1	
184	B8			226	E2	
185	B9			227	E3	
186	BA			228	E4	
187	BB			229	E5	
188	BC			230	E6	
189	BD			231	E7	
190	BE			232	E8	
191	BF			233	E9	
192	C0			234	EA	
193	C1			235	EB	
194	C2			236	EC	
195	C3			237	ED	
196	C4			238	EE	
197	C5			239	EF	
198	C6			240	F0	
199	C7			241	F1	
200	C8			242	F2	
201	C9			243	F3	
202	CA			244	F4	
203	CB			245	F5	
204	CC			246	F6	
205	CD			247	F7	

DECIMAL	HEXADECIMAL	ASCII		DECIMAL	HEXADECIMAL	ASCII
248	F8			252	FC	
249	F9			253	FD	
250	FA			254	FE	
251	FB			255	FF	

ANNEXE N° 6

REPRESENTATION DES NOMBRES DANS LE 8087

A.I Nombres normaux

Le coprocesseur mathématique 8087 peut lire et écrire en mémoire sept (7) formes différentes de représentations numériques groupées en quatre (4) types. Les échanges entre le 8086 et le 8087 se font à travers la mémoire.

A.II Nombres anormaux

Le 8087 compte plusieurs nombres anormaux. Dans les nombres NaN = Not a Number, tous les bits de l'exposant sont à 1 et la mantisse a une valeur quelconque. Ces nombres sont générés dans le cas d'un Stack overflow et underflow, d'une indétermination (0/0), d'une racine carrée d'un nombre négatif $((-X)^{1/2})$, le logarithme d'un nombre négatif (log (-X)) et de l'utilisation d'un opérande NaN. Le Stack overflow a lieu lors d'une tentative de l'écriture d'un nombre dans une pile pleine. Le Stack underflow a lieu lors d'une tentative de lecture d'un nombre dans une pile vide. Un autre type de nombres anormaux sont les nombres $\pm\infty$ qui ont tous les bits de leur exposant à 1 et leur mantisse nulle. Enfin, les nombres ± 0 sont d'exposant et de mantisse nuls.

Remarque: *Une mantisse M normalisée doit satisfaire $0.5 \leq M < 1$.

Tableau 1 Formats des nombres dans le 8087

TYPE	GRANDEUR	FORMAT	ETENDU DE VARIATION	REMARQUES IMPORTANTES
WORD	16 bits dont 1 de signe	Entier	-2^{15} à $+2^{15}-1$	Entier uniquement
DWORD	32 bits dont 1 bit de signe	Entier court	-2^{31} à $+2^{31}-1$	
		Réel court	Signe sur 1 bit Exposant sur 8 bits *Mantisse sur 23 bits	Norme IEEE : Format Simple Précision en Fortran. Format d'E/S uniquement. Dans cette représentation, un nombre s'écrit en excédent 2^{7}: $(-1)^{s}\,2^{e-128}$ où $0 \le e \le 2^{8}-1$
QWORD	64 bits dont 1 bit de signe	Entier long	-2^{63} à $+2^{63}-1$	
		Réel long	Signe sur 1 bit Exposant sur 11 bits *Mantisse sur 52 bits	Norme IEEE : Format Double Précision en Fortran. Format d'E/S uniquement. Dans cette représentation, un nombre s'écrit en excédent 2^{10}: $(-1)^{s}\,2^{e-1023}$ où $0 \le e \le 2^{11}-1$
TBYTE	80 bits dont 1 bit de signe	BCD	Les bits 0 à 17 représentent un quartet chacun dont la valeur est entre 0 et 9. Soit 4x18=72 bits. Les 7 autres bits (bits 72 à 78) ne sont pas significatifs.	Format utilisé pour les E/S mais occupe beaucoup de place mémoire.
		Réel temporaire	Signe sur 1 bit Exposant sur 15 bits *Mantisse sur 64 bits	Utilisé en mémoire pour garder les résultats intermédiaires lorsque la pile interne est pleine. Toutes les opérations internes sont effectuées sous cette forme. Aucune erreur d'arrondi à craindre. Dans cette représentation, un nombre s'écrit en excédent 2^{14}: $(-1)^{s}\,2^{e-16384}$ où $0 \le e \le 2^{15}-1$

ANNEXE N°7

INTERRUPTIONS BIOS ET DOS USUELLES

Cette annexe explique comment appeler les fonctions résidantes en mémoire morte (BIOS) et celles du système d'exploitation (DOS) à partir de programmes en langage assembleur. Ce n'est qu'à partir de Windows 95 que l'accès au matériel à commencer à se faire en mode protégé. Cela exige une connaissance des mécanismes de mis en œuvre par l'architecture du 386 ainsi que celle de librairies ou ressources matérielles.

La plupart de ces fonctions sont accessibles à partir de l'interruption 21H. Certaines d'entre elles ne sont cependant pas documentées, car elles sont susceptibles de modification lors d'un changement de version du système d'exploitation.

Nous n'avons nullement la prétention de reprendre ici toutes les interruptions BIOS et DOS qui sont apparues avec les premiers PC, ni de leurs différentes versions qui se sont succédées. Lesquelles versions ont été le support de nouveaux périphériques et de fonctionnalités étendues pour le réseau, la mémoire paginée ou étendue, etc. Nous nous contenterons de donner quelques exemples, en particulier, ceux qui sont utiles au cours, travaux dirigés et travaux pratiques décrits dans ce manuel.

INT N°	Effet	FCT N°	Effet de la fonction	Entrées	Sorties
INT 10 H	Vidéo	00H	Sélection du mode vidéo actif et effacement de la page courante. Sélectionne aussi le contrôleur vidéo actif si plusieurs sont présents.	AH=00H AL= Mode Vidéo Exemple: AL=03H; Mode texte 80x25et 16 couleurs	Rien
		05H	Sélection de la page affichée.	AH=05H AL=N° de la page (0-7) Exemple : AL=01 pour les modes EGA et VGA	CGA, EGA, MCGA et VGA.
		OEH	Ecriture d'un caractère en mode télétype. Affiche un caractère ASCII à la position courante du curseur puis incrémente sa position (du curseur).	AH=0EH AL=Caractère ASCII BH=N° de la page. Par défaut : Page courante BL=Couleur d'écriture	Rien
		13H	Ecriture d'une chaine en mode télétype. Transfère une chaine vers le buffer vidéo de la carte vidéo active, en commençant à la position spécifiée.	AH=13H AL=Mode d'écriture : 0, 1, 2 ou 3 BH=N° de la page BL=Attribut (Si AL=00H ou AL=01H) CX=Longueur de la chaine DH=y DL=x ES : BX=Segment : Offset de la chaine AH=0FH	Rien
		0FH	Demande du mode vidéo courant du contrôleur vidéo actif. Retourne le n° de la page active dans BH.		AH=Nombre de colonnes de caractères à l'écran AL=Mode d'affichage (Voir INT 10H, fonction 00H) BH=N° de la page actuellement affichée

INT N°	Effet	FCT N°	Effet de la fonction	Entrées	Sorties
INT 13H	Disque dur ou lecteur de disquettes	02H	Lecture d'un ou de plusieurs secteurs du disque en mémoire. Le bootstrap est situé à la piste 0, secteur 1, face out tête. En fonction du DOS utilisé, nous pouvons compter : 512 bytes/secteur 8-15 secteurs 8 et plus secteurs/piste Le nombre de cluster=nombre de têtes pour le disque dur.	AH=02H AL=Nombre de secteurs à lire CH=N° du Cylindre ou N° de la piste CL=N° du secteur DH=Tête ou N° de la face 0 ou 1 DL=Unité (0=A et 1=B) 00H-7FH pour le lecteur de disquettes 80H-FFH pour le disque dur ES :BX= Segment :Offset Buffer	Si la lecture est correcte, alors CF effacé AH=00H AL=Nombre de secteurs transférés Si la lecture à échoué CF positionné AH=Etat (Voir INT 13H fonction 01H) 01H Commande invalide 02H Marque d'adresse non trouvée 03H disque protégé 04H Secteur non trouvé...
INT 16H	Clavier	00H	Lecture d'un caractère du clavier et renvoie aussi le code de la touche.	AH=00H	AH=Code de la touche AL=Caractère ASCII
		01H	Demande de l'état du clavier.	AH=01H	Si une touche est prête alors Z=0. AH=Code de la touche AL=Caractère ASCII Si aucune touche n'est disponible alors Z=1

INT N°	Effet	FCT N°	Effet de la fonction	Entrées	Sorties
INT 17H	Imprimante	00H	Envoi d'un caractère vers le port d'interface imprimante spécifié et renvoi de l'état courant de celui-ci. **Très important:** Avant d'envoyer un caractère vers l'imprimante, il est conseillé 1- d'appeler INT 17H à travers la fonction 02H qui permet d'acquérir l'état de celle-ci. 2- d'appeler INT 17H à travers la fonction 00H qui permet d'envoyer un caractère vers celle-ci. Dans le cas d'une première utilisation, il faut d'abord initialiser l'imprimante à travers INT 17H, fonction 01H.	AH=00H AL=Caractère DX=N° de l'imprimante (0=LPT1 et 1=LPT2, ...	AH=Etat de l'imprimante Bit: Signification (si positionné) 7: Imprimante prête 6: Confirmation de l'imprimante 5: Plus de papier 4: Imprimante sélectionnée 3: Erreur d'E/S 2: Inutilisé 1: Inutilisé 0: Time out
INT 20 H	Terminaison d'un programme avec retour au DOS				

INT N°	Effet	FCT N°	Effet de la fonction	Entrées	Sorties
INT 21 H	Terminaison d'un programme avec retour au DOS	00H		AH=00H	
		4CH		AH=4CH	
		4D H	Récupération du code retour d'interruption	AH=4DH	AH=00H Fin d'exécution normale par INT 20H, INT 21H fonction 00H ou INT 21H fonction 4CH AH=01H Fin d'exécution par CTRL-C AH=02H Fin d'exécution par la routine d'erreur critique AH=03H Fin d'exécution via INT 21 fonction 31H ou INT 27H)
		02H	Affichage d'un caractère	AH=02H DL=Caractère à envoyer codé sur 8 bits.	Rien
		09H	Affiche une chaine de caractères à l'écran. La chaine de caractères doit se terminer par un '$' i.e., fin de message.	AH=09H DS:DX=Adresse de la chaine (Segment:Offset)	Rien
		25H	Affectation d'un vecteur d'interruption. Permet d'affecter une adresse de routine de service à un vecteur d'interruption donné.	AH=25H AL= Numéro du vecteur à affecter. DS:DX=Adresse de la routine de service (Segment:Offset).	Rien
		35H	Obtention de l'adresse d'un vecteur d'interruption.	AH=35H AL=Numéro du vecteur d'interruption.	ES:BX=Adresse du vecteur d'interruption (Segment:Offset).

INT N°	Effet	FCT N°	Effet de la fonction	Entrées	Sorties
33H	Gestion de la souri	00H	Initialisation de la souri et disparition de son pointer.	AX=0000H	Si AX=FFFFH si avec succès BX= Nombre de boutons Sinon AX=0000
		01H	Apparition du pointer de la souri.	AX=0001H	Rien
		02H	Disparition du pointer de la souri.	AX=0002H	Rien
		03H	Lecture de la position de la souri et de l'état de ses boutons.	AX=0003H	BX=0001H Si le bouton de gauche est enfoncé BX=0002H SI le bouton de droite est enfoncé BX=0003H SI les deux boutons sont enfoncés CX=x DX=y

REFERENCES BIBLIOGRAPHIQUES

[1] Machines Algorithmiques, R.Davio, J.P. Deschamps, Presses polytechniques normandes, édition 1988.

[2] Intel microprocessor and peripheral data book, 1989.

[3] Technical reference for the IBM Personal computer AT, Intenational Business Machine Corporation, 1984.

[4] Le grand livre du turbo assembleur et du turbo déboggeur, M.Tisher, Micro Applications, 1989.

[5] Microprocesseur 8086-8088 Architecture et programmation, J.M. Trio, Eyrolles, 1990.

[6] SIM Manuel d'utilisation, Hoffman.

[7] Programmation en langage assembleur, B. Geoffrion, éditions Radio, 1986.

[8] MS-DOS, 130 astuces et utilitaires, Daniel Rougé, édition SYBEX, 1989.

[9] Programmer avec les fonctions BIOS-IBM, Ray Duncan, édition Microsoft Press, 1981.

[10] PC XT et AT (Maintenance et Réparation), Benoît Michet et Michel Benoît, édition B.C.M.s.c., 1988.

[11] Assembleur facile, Ph. Mercier, édition Marabout, 1989.

[12] 80386, A programming and Design Handbook, Pen Brumm and Don Brumm, édition TAB BOOKS Inc., 1987.

[13] Architecture de l'ordinateur, Adrew Tanenbaum, InterEditions, 1987.

[14] VIA P4M800-P4 M/B Motherboard Quick Guide, version 1.0, April 2005.

[15] Comment identifier votre Chipset Intel.
www.intel.com/support/fr/chipsets/sb/cs-009245.htm

[16] iAPX86 Datasheet, Circuit, PDF, & Application Note Results.
www.datasheetarchive.com/iAPX86-datasheet.html

[17] Comparatif cartes mères Intel P35
clubic.com/article-79354-1-comparatif-meres-intel-p35-dos...

[18] Nouveau chipset pour Core 2 Duo et DDR3.

[19] clubic.com/article-73884-1-chipset-intel-p35-core-2-duo-ddr3-intel...

[20] Identification d'un Chipset Intel.
www.intel.com/support/fr/chipsets/sb/cs-009245.html

[21] Intel Core i7 en pratique.
hardware.fr/articles/.../dossier-intel-core-i7-pratique.html

[22] Intel Atom. Processor Macroarchitecture.
intel.com/technology/.../microarchitecture.htm

[23] Difference between Intel Core 2 Extreme/Quad/Duo vs Core
chotocheeta.com/2009/11/28/difference-between-intel-core-2-extreme-quad-duo-vs-core...

[24] Serial ATA (SATA ou S-ATA)
www.commentcamarche.net/contents/pc/serial-ata.php3
[25] PCI Express
www.commentcamarche.net/contents/pc/pci-express.php3
[26] PCI Express 3.0 déjà abordé.
www.clubic.com/actualite-143790-pci-express-aborde.html
[27] Bus PCI
www.commentcamarche.net/contents/pc/pci.php3
[28] Bus USB (Universal Serial Bus)
www.commentcamarche.net/contents/pc/usb.php3
[29] AGP
www.commentcamarche.net/contents/pc/agp.php3
[30] De la vulgarisation de l'ordinateur, des réseaux, des bases de données et des langages informatiques.
www.commentcamarche.net
[31] Descriptif sommaire de la famille des processeurs Intel iAPx86.
www.fil.univ-lille1.fr/~sedoglav/Archi/TP002.html
[32] Bus AGP
www.commentcamarche.net/contents/pc/agp.php3
[33] GA's - CGA VGA/SVGA XGA SXGA UXGA WXGA WSXGA QXGA QSXGA QUXGA.
www.infocellar.com/hardware/ga.html
[34] Aide-mémoire PC et MSDOS, Gérad Blanchet,
http://perso.telecom-paristech.fr/~blanchet/DOCUMENTS/MSDOS_PCs.pdf
[35] Intel Core i7Procesors and Intel DX58SO X58...
http://www.gadgetadvisor.com/computer-hardware/intel-core-i7-processors-and-intel-dx58so-motherboard-review
[36] HP Compac Business PC
Documentation and Diagnosis DVD, dx2400, dx2450 and dx2390 models