Ministère de l'enseignement supérieur et de la recherche scientifique
Institut Supérieur des Etudes Technologiques de Sousse

Département informatique
Spécialité : Informatique industrielle
Niveau 4

Support de cours

Microprocesseur

8086

Ali HMIDENE

Agrégé en Génie Electrique

Technologue à l'ISET de Sousse

2009/2010

TABLE DES MATIERES

INITIATION AU MICROPROCESSEUR

Ce chapitre présente l'architecture interne d'un système microprocesseur. Il donne une aperçue de la structure du microprocesseur 8086/8088, et d'importantes notions communes à la plupart des microprocesseurs.

1. ARCHITECTURE GENERALE D'UN SYSTEME A MICROPROCESSEUR

La Figure 1 montre l'architecture d'un système micro-ordinateur type. Le microprocesseur (MPU, *Micro-Processing Unit*) apparaît à gauche. Il remplit les fonctions d'une unité centrale (CPU, *Central Processing Uint*) en un seul boîtier ; il comprend une unité arithmétique et logique (ALU, *Arithmetic en logic Unit*) avec ces registres internes, et une unité de contrôle (CU, *Control Unit*) chargée de décoder et d'envoyer les signaux nécessaires aux différentes unités.

Le MPU a trois bus : un bus d'adresses, un bus de données et un bus de commande.

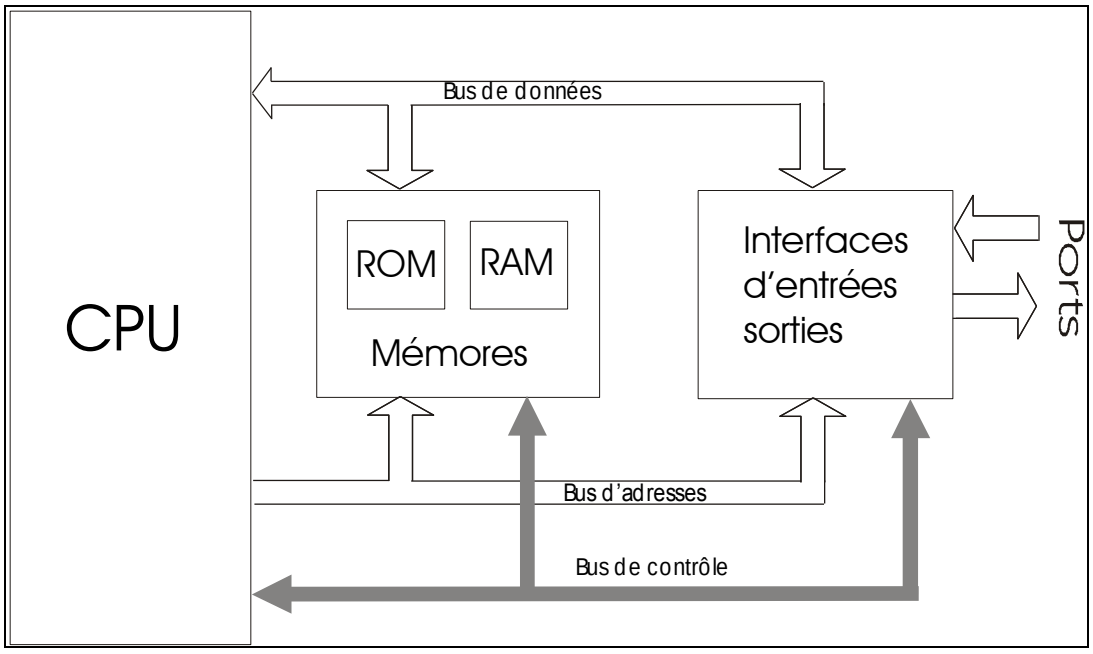


Figure 1

A côté du CPU il y a la ROM, la RAM et l'interface d'entrée-sortie (IO). Ce sont les principaux éléments d'un système à microprocesseur. La ROM (*Read Only Memory*) stocke le programme du système ; elle contient toujours un programme de chargement initial ou un moniteur pour permettre le démarrage du système. Dans un contexte de contrôle de processus, presque tous les programmes résideront en ROM ; car ils ne seront probablement

jamais changés et doivent donc être protégés contre les pannes d'alimentations.

La RAM (*Random Access Memory*) est la mémoire à lecture et écriture du système. Dans le cas des applications individuelles et lors de mise au point des programmes, la plupart de ceux-ci sont stockés en RAM pour pouvoir être modifiés facilement. Dans un système de contrôle de processus, la RAM est généralement petite, elle ne sert qu'aux données.

Enfin le système comporte un ou plusieurs boîtiers d'interface afin de pouvoir communiquer avec le monde extérieur (clavier, unité d'affichage, contrôle des processus…).

Le système présenté a une architecture à trois bus. La fonction du bus d'adresses est de fournir une communication entre ROM, RAM et interface d'entrée-sortie. Le bus de données transmet l'information entre le CPU et l'unité valide du système. Le bus de commande effectue la double tâche consistant d'une part à définir électriquement le type de communication, et d'autre part à provoquer le début et la fin de transfert.

Dans la plupart des applications, le CPU commande tous les bus. C'est à lui d'assurer leur synchronisation pour permettre une communication précise.

La largeur d'un bus détermine le nombre de lignes de signaux contenus dans le groupe comportant le bus. En ce qui concerne le 8086/8088, la largeur du bus d'adresses peut atteindre 20 bits. La largeur du bus des données est de 16 bits ; celle du bus de commande varie, mais sa valeur nominale est de 5 bits.

2. ARCHITECTURE D'UN CPU

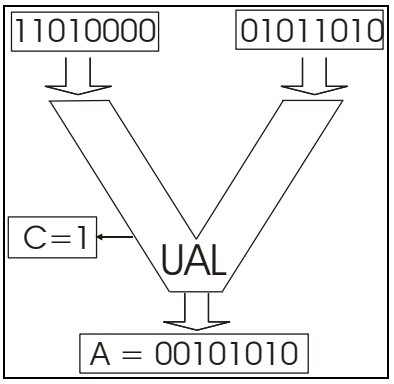
Le microprocesseur est constitué des unités fonctionnelles suivantes :

- L'unité arithmétique et logique (ALU),
- Des registres,
- Une unité de contrôle (CU).

2.1. Unité arithmétique et logique

Cet organe, interne au microprocesseur, permet la réalisation d'opérations arithmétiques et logiques. L'ALU possède un additionneur n bits et des fonctions logiques. Outre les opérations arithmétiques et logiques, l'ALU permet aussi les opérations de décalage et de rotation. Le registre

d'état est lié étroitement à L'ALU, nous donne à travers ces inducteurs (Flags) des renseignements supplémentaires sur le résultat d'une opération (Résultat nul, négatif, dépassement…). Dans l'exemple ci-dessous, une addition est effectuée entre les deux opérandes, le résultat est mis dans le registre A.



2.2. Les registres

Le microprocesseur comporte deux types de registres : les registres à usage général, et les registres pointeurs d'adresses.

2.2.1. Les registres à usage général

Ce sont des mémoires rapides, à l'intérieur du microprocesseur, qui permettent à l'UAL de manipuler des données à vitesse élevée. Ils sont connectés au bus de données interne du microprocesseur.

Les registres sont repérés par leurs nom (on donne généralement comme nom une lettre A, B, C ...)

Exemple : **MOV C,B** : transfert du contenu du registre B dans le registre C

2.2.2. Les registres d'adresses (pointeurs)

Ce sont des registres connectés sur le bus adresses.

On peut citer comme registre:

- Le compteur ordinal (pointeur d'instructions PC)
- Le pointeur de pile (stack pointer SP)
- Les registres d'index (index source SI et index destination DI)

Le compteur ordinal (pointeur de programme PC)

Il contient l'adresse de l'instruction à rechercher en mémoire. L'unité de contrôle incrémente le compteur ordinal (PC) du nombre d'octets sur lequel l'instruction, en cours d'exécution est codée. Le compteur ordinal contiendra alors l'adresse de l'instruction suivante.

Exemple (Figure 2) : (PC)=10000H ; il pointe la mémoire qui contient l'instruction MOV C,B qui est codé sur deux octets (89 D9H) ; l'unité de contrôle incrémentera de deux le contenu du PC : (PC) = 10002H (la mémoire sera supposée être organisée en octets).

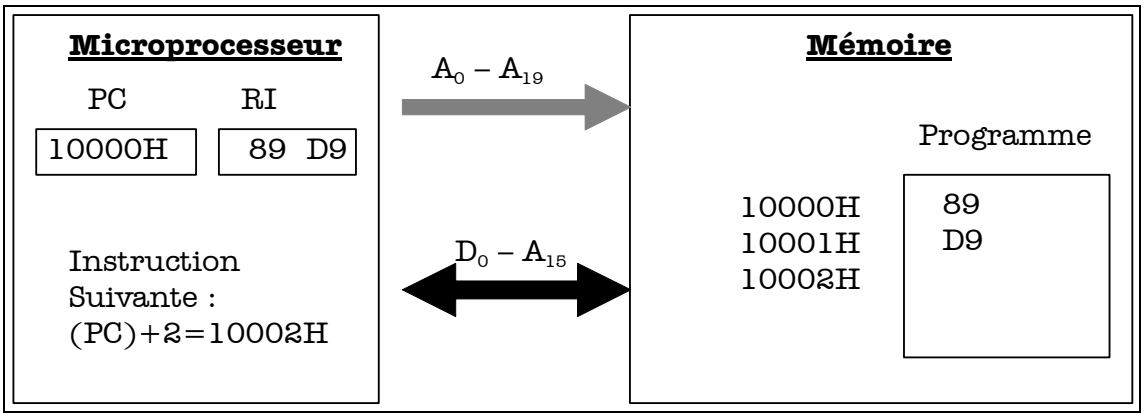


Figure 2

Le pointeur de pile (stack pointer SP)

Il contient l'adresse de la pile. Celle-ci est une partie de la mémoire, elle permet de stocker des informations (le contenu des registres) relatives au traitement des interruptions et des sous-programmes.

La pile est gérée en **LIFO** : (Last IN First Out) dernier entré premier sorti. Le fonctionnement est identique à une pile d'assiette. Le pointeur de pile SP pointe le haut de la pile (31000H dans le cas de la Figure 3), il est décrémenté avant chaque empilement, et incrémenté après chaque dépilement. Il existe deux instructions pour empiler et dépiler : ***PUSH*** et ***POP***.

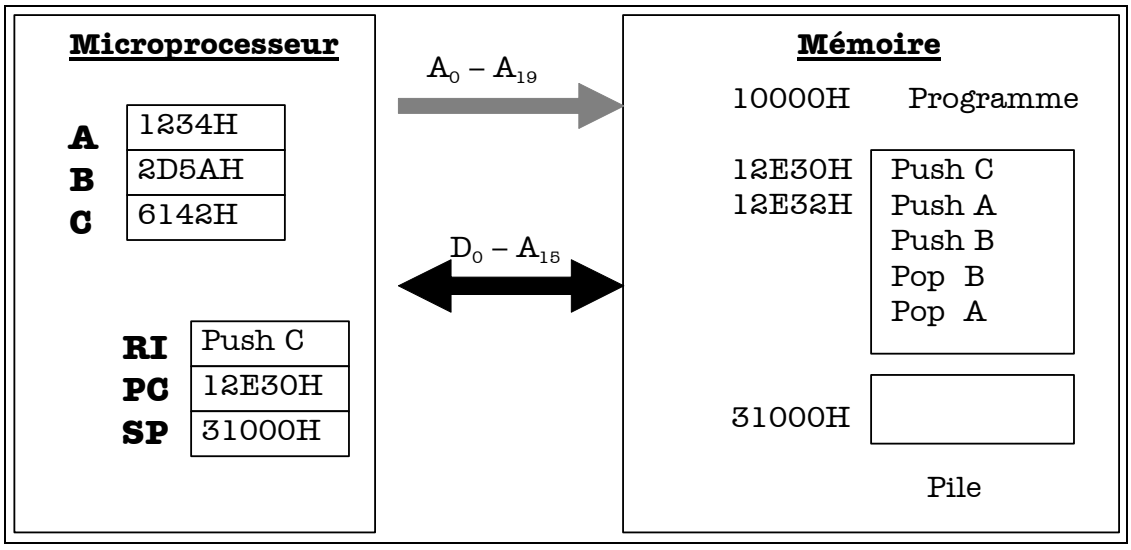


Figure 3

Exemple:

PUSH A empilera le registre A et POP A le dépilera.

Sur la Figure 4, on montre le fonctionnement de la pile lors des instructions comme PUSH et POP.

Question?

Que se passera-t-il durant l'exécution du programme commençant en 12E30H? Que vaudra SP et que contiendra la pile à cette adresse à la fin du programme?

Réponse.

Le programme commence par sauvegarder le contenu de registre C dans la pile (PUSH C). Pour cela (SP) est décrémenté de deux ((SP)=31000H -2 = 30FFEH), ensuite le contenu du registre (C) sera chargé dans la mémoire à l'adresse 30FFEH.

Pour PUSH A on obtient : (30FFCH)=1234H, et pour PUSH B : (30FFAH)=2D5AH.

Pour l'instruction POP B, le contenu de la case mémoire pointée par la registre SP (noté ((SP)) : contenu du contenu) est chargé dans le registre B. ((SP))=30FFAH ; (B)=2D5AH) puis (SP) est incrémenté de deux ((SP)= 30FFAH+2=30FFCH). Enfin, pour POP A on obtient : (A)=1234H et (SP)=30FFCH + 2 = 30FFEH.

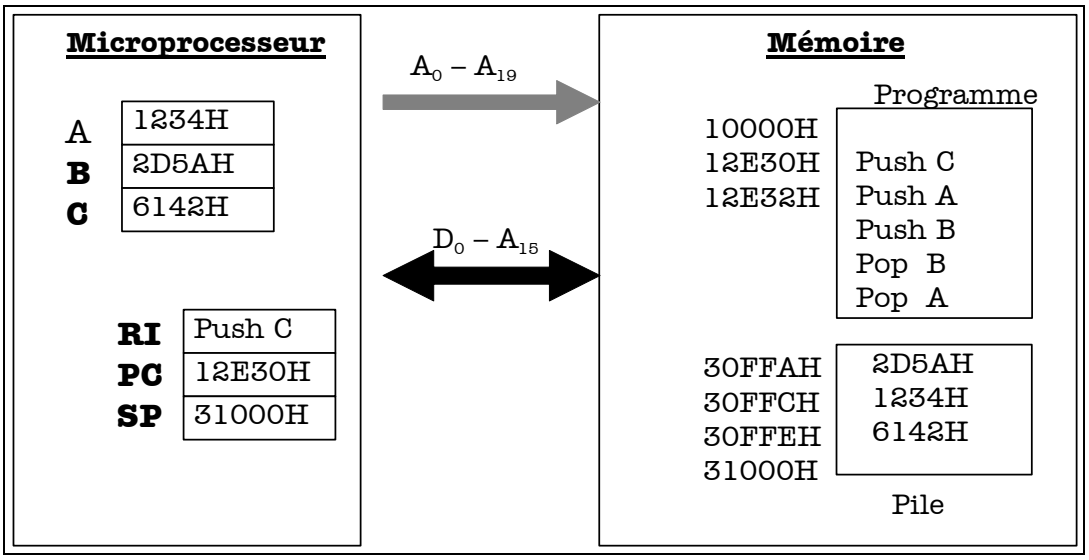


Figure 4

Les registres d'index (index source SI et index destination DI)

Les registres d'index permettent de mémoriser une adresse particulière (par exemple : début d'un tableau). Ces registres sont aussi utilisés pour adresser la mémoire de manière différente. C'est le mode d'adressage indexé.

Exemple :

MOV A,[SI+10000H] place le contenu de la mémoire d'adresse 10000H+le contenu de SI, dans le registre A.

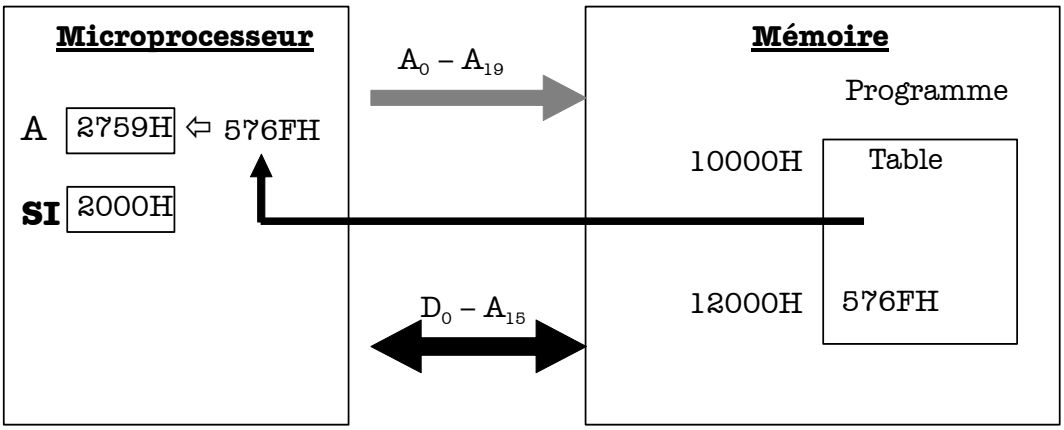


Figure 5

2.3. Architecture standard

L'ensemble des composants cités ici est regroupé dans le schéma présenté par la Figure 6. L'architecture présentée est dite "Architecture Van Nuoman ". Notamment, il existe aussi les architectures à plusieurs bus internes.

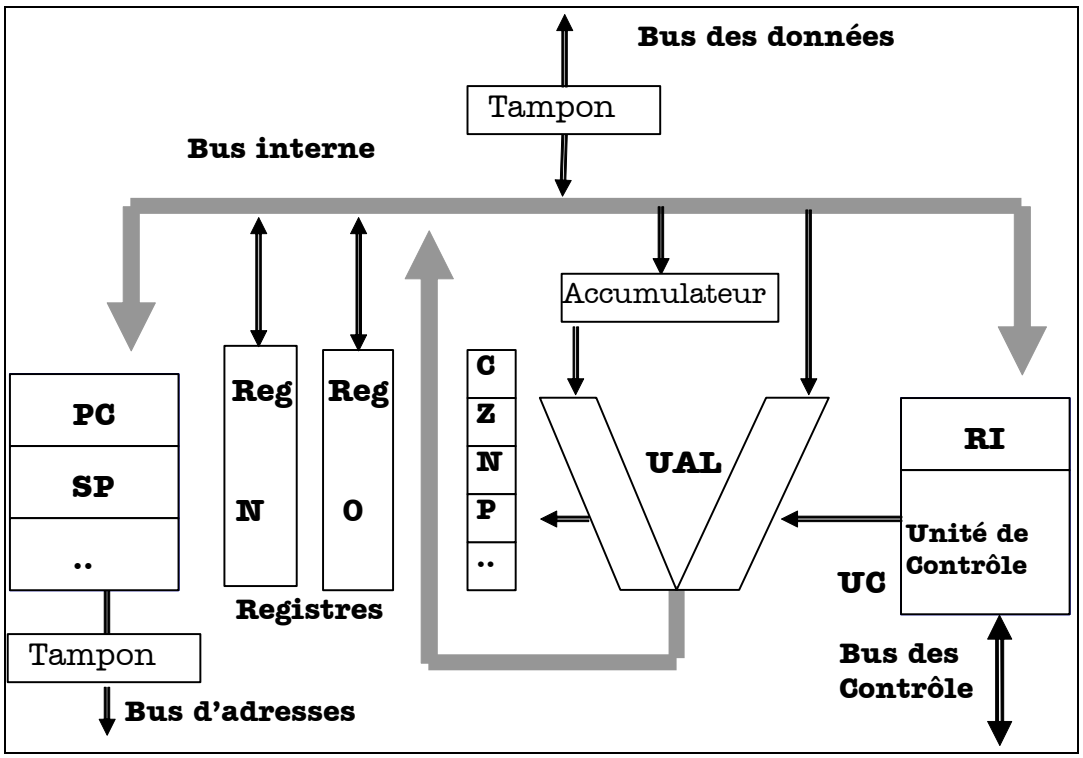


Figure 6

3. EXECUTION D'UNE INSTEUCTION

Les microprocesseurs sont capables d'effectuer un certain nombre d'opérations élémentaires.

Une instruction au niveau machine doit fournir à l'unité centrale toutes les informations nécessaires pour déclencher une telle opération élémentaire : type d'action, où trouver le ou les opérandes, où ranger le résultat …. C'est pourquoi une instruction comporte en général plusieurs champs ou groupes de bits. Le premier champ contient le code opération. Les autres champs peuvent comporter des données ou l'identificateur des opérandes. Sur certaines machines les instructions sont toutes de même longueur, sur d'autres cette longueur peut varier avec le code opération ou le mode d'adressage.

Le traitement d'une instruction peut se découper en plusieurs phases. Celles-ci sont au nombre de 3 :

- ✓ Recherche de l'instruction (Fetch)
- ✓ Décodage (decode)
- ✓ Exécution (execute)

3.1. Recherche d'une instruction (Fetch)

Le contenu de PC (compteur ordinal) est placé sur le bus des adresses (c'est l'unité de contrôle qui établit la connexion).

L'unité de contrôle (UC) émet un ordre de lecture (READ=RD=1) Au bout d'un certain temps (temps d'accès à la mémoire) le contenu de la case mémoire sélectionné est disponible sur le bus des données.

L'unité de commande charge la donnée dans le registre d'instruction pour décodage.

Dans l'exemple de la Figure 7, Le microprocesseur place le contenu de PC (10000H) sur le bus adresses et met RD à 1 (cycle de lecture). La mémoire met sur le bus données le contenu de sa mémoire n° 10000H (ici 89D9H qui est le code de MOV C,B). Le microprocesseur place dans son registre d'instruction le contenu du bus données (89D9H). L'unité de commande décode et exécute l'instruction MOV C,B .

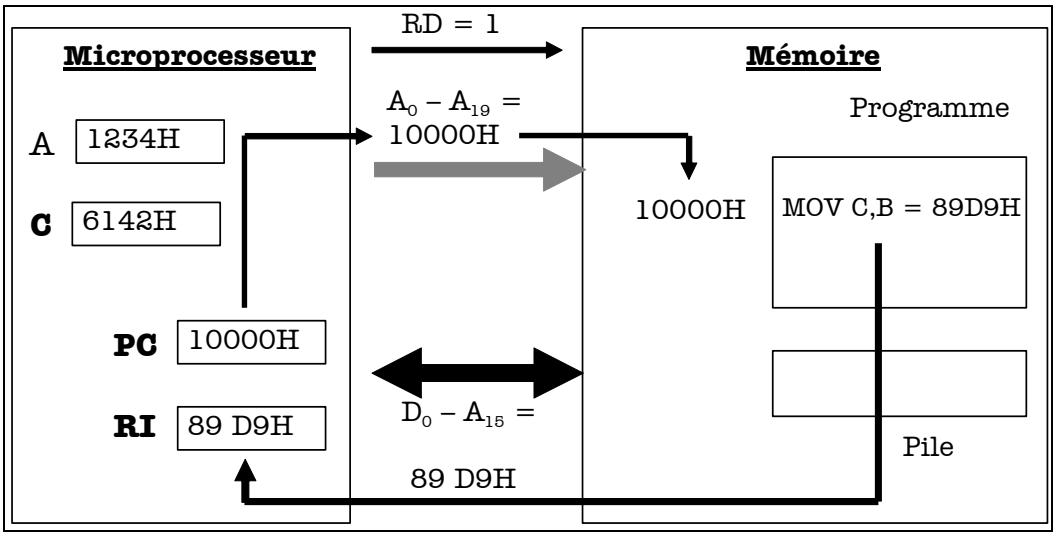


Figure 7

3.2. Décodage

Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs octets. Ce premier mot contient le **code opératoire** qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de d'octets de l'instruction. L'unité de commande décode le code opératoire et peut alors exécuter l'instruction.

3.3. Exécution

Le micro-programme réalisant l'instruction est exécuté. Les indicateurs sont positionnés (registre d'état). L'unité de commande positionne le compteur ordinal (PC) pour l'instruction suivante.

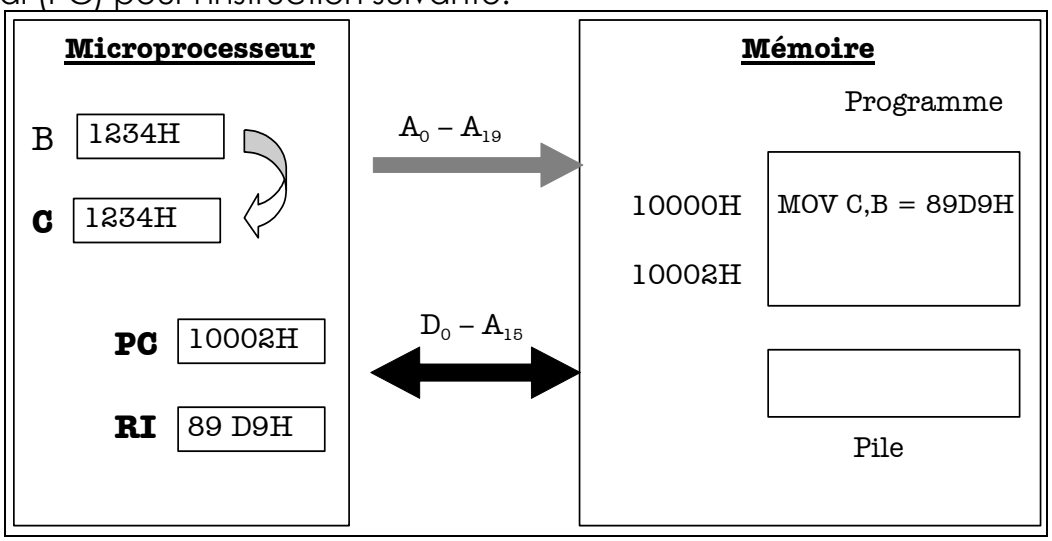


Figure 8

Après cette exécution, nous aurons C=B=1234H et PC = 10002H.

4. LES INTERRUPTIONS

4.1. Les interruptions matérielles

Le microprocesseur possède des broches spécialisées pour les interruptions. Si un composant extérieur au microprocesseur veut l'interrompre, il lui suffit de changer la valeur d'une de ces broches (0 ⇨1 ou 1⇨0 suivant les broches). Le programme qui traite l'interruption (appelé sous-programme d'interruption) a été placé en mémoire par le concepteur à une adresse connue du microprocesseur.

Lors d'une demande d'interruption, le microprocesseur exécute l'instruction en cours, charge dans la pile d'adresse de l'instruction suivante, puis il charge le (PC) (pointeur de programme) par l'adresse de la première instruction du sous-programme d'interruption. Lorsque le microprocesseur termine l'exécution de la routine d'interruption, il restaure l'adresse de retour à partir de la pile pour et continu l'exécution du programme.

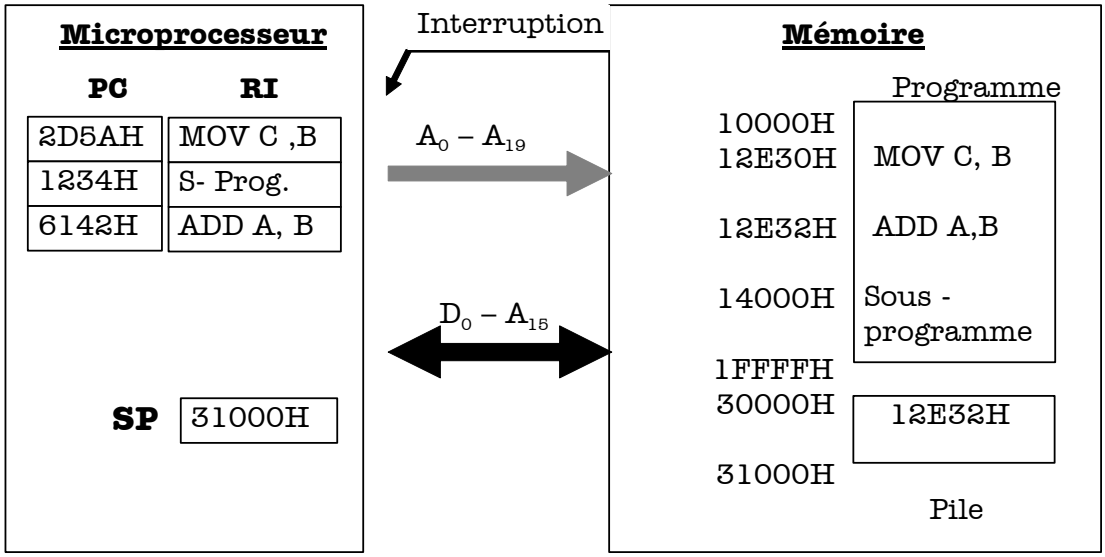


Figure 9

Dans l'exemple de la Figure 9, l'adresse de l'instruction en cours est 12E30H. Le microprocesseur exécute l'instruction MOV C,B. Il s'aperçoit de la demande d'interruption. Avant de la traiter, il termine l'instruction, sauvegarde l'adresse de retour dans la **pile**, c'est à dire l'adresse de l'instruction suivante (dans notre exemple 12E32H), puis il se branche à l'adresse du sous-programme d'interruption.

A la fin de la routine d'interruption, peut facilement revenir au programme principal en reprenant l'adresse sauvegardée dans la **pile**, grâce à une instruction de retour (interrupt return IRET).

Pour éviter le problème lié à la modification des registres internes par le sous-programme d'interruption. Le programmeur devra par l'intermédiaire de l'instruction **PUSH** sauvegarder l'état des registres internes dans la **pile.**

Cet empilement s'appelle : sauvegarde du "contexte". Le microprocesseur se charge de sauvegarder le contenu du PC (adresse de retour) et le registre d'état **F**. Les autres registres doivent êtres sauvegarder par le programmeur du micro-ordinateur**.**

4.2. Les interruptions logicielles

Les interruptions logicielles sont gérer de la même manière que les interruptions matérielles. La seule différence réside dans le faite qu'une interruption matérielle est déclenchée par un évènement externe, alors qu'une interruption logicielle est déclenchée par un appel explicite. Les interruptions logicielles permettent au programme utilisateur d'utiliser les fonctions systèmes. Dans le cas du microprocesseur 8086 l'accès aux périphériques du PC (clavier, écran, disque dur …) se fait à travers les fonctions DOS et BIOS.

LE MICROPROCESSEUR 8086/8088

1. ORGANISATION INTERNE DU 8086/8088

Les microprocesseurs 8086/8088 se présentent en boitier de 40 broches. Ils peuvent en effet, adresser 1 M octets (M = million) ce qui exige 20 lignes pour le bus d'adresses qui est, multiplexé avec le bus de données (de 16 bits pour le 8086 et de 8 bits pour le 8088.

Le schéma de la structure interne (Figure 10) fait apparaître deux unités, l'une qui exécute les instructions (EU, *Execution Unit*) avec son unité arithmétique et logique (UAL) et ses registres ; l'autre qui assure la liaison avec le monde extérieur (BIU, *Bus Interface Unit*) génère les adresses grâce à un additionneur (Σ) et lit les instructions qu'il range dans une file d'attente (Queue), de 6 octets pour le 8086 et de 4 pour le 8088.

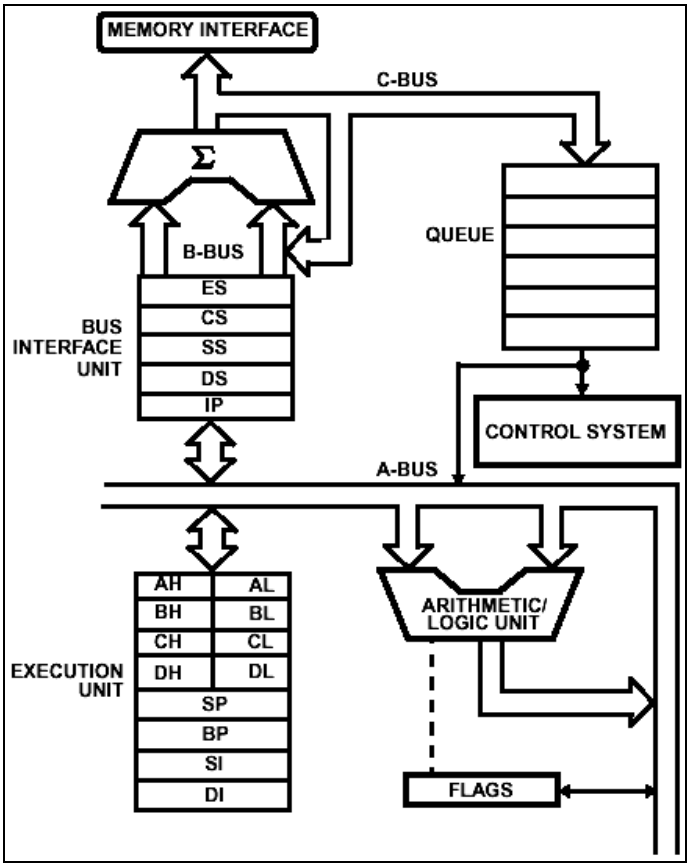


Figure 10

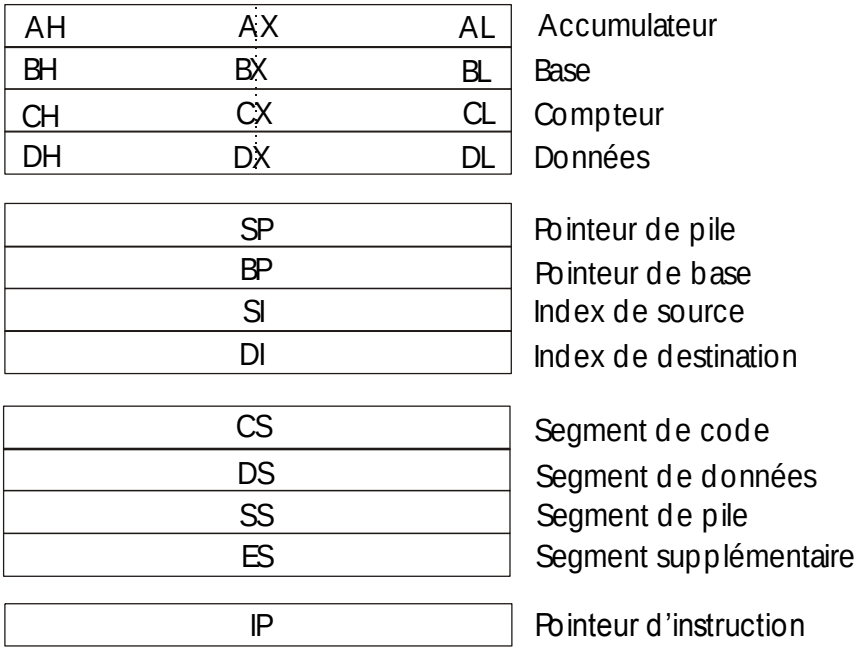
L'unité d'exécution reçoit les instructions de la file d'attente et transmet à l'unité d'interface le résultat de son travail.

L'unité d'interface assure le contrôle du bus externe, lit les instructions, sans les décoder, et les range dans la file d'attente, écrit en mémoire les résultats qui lui fournit l'unité d'exécution, de sorte que le bus de données est toujours occupé. Dés qu'un octet est libre dans la file d'attente, le BIU va rechercher l'instruction suivante.

2. LES REGISTRES

Le 8086 dispose de 3 jeux de registres de 16 bits :

- les registres généraux,
- les pointeurs et index,
- les registres de segments,



2.1. Registres généraux

Ils sont au nombre de 4 et peuvent travailler par moitié (8 bits) :

- Accumulateur : AX composé de AH et AL
- Base : BX composé de BH et BL
- Compteur : CX composé de CH et CL
- Donnée : DX composé de DH et DL

Bien qu'ils soient des registres généraux, ils ont des fonctions bien précises en dehors des opérations arithmétiques et logiques classiques, comme nous le verrons plus loin :

BX sert à l'adressage en mémoire de données

CX et CL servent de compteurs de boucles

DX permet l'adressage des ports s'entrée/sortie et sert également d'extension à AX, dans le cas des opérations arithmétiques (donnée sur 32 bits).

2.2. Les pointeurs et index

Parmi ces quatre registres de 16 bits, deux sont appelés pointeurs et deux appelés index. Les deux pointeurs SP (Stack Pointer) et BP (Base Pointer), servent à la génération des adresses des données en particulier en pile pour SP.

Les deux index permettent la gestion de suite de mots, il s'agit de l'index de destination (DI) et l'index de source (SI) qui peuvent également participer à la génération des adresses.

2.3. Registres de segments

L'espace mémoire de 1 Mo est découpé en tranches de 64 Ko maximum, appelées « segments ». Le CPU possède un accès direct aux quatre segments. Leurs adresses de base se trouvent dans les registres segments.

- Segment des codes (CS) sert à l'adressage des octets du programme (code),

- Segment des données (DS) sert à l'adressage des données,

- Segment de pile (SS) gère la pile,

- Extrat Segment (ES) complète DS.

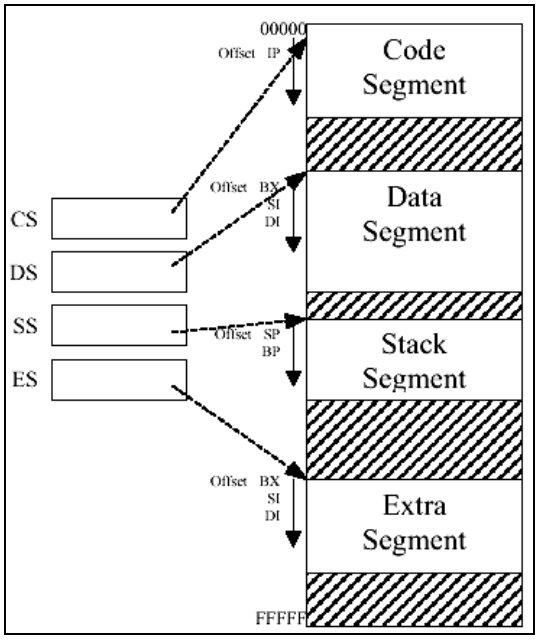


Figure 11

2.4. Compteur Ordinal et Registre d'état (Flags)

Le compteur de programme appelé ici pointeur d'instruction (IP). Ce registre est mis à jour par le BIU afin qu'il pointe vers l'adresse de l'instruction suivante.

Le microprocesseur 8086/8088 à six indicateurs d'états, qui sont mis à jour par l'unité d'exécution. L'état de ces indicateurs dépend du résultat de l'opération arithmétique et logique qui vient d'avoir lieu.

La Figure 12 donne l'emplacement des inducteurs dans le registre d'état.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Figure 12

CF (Carry Flag) indique un dépassement de capacité, c'est le neuvième ou dix-septième bit du résultat.

PF (Parity Flag). Si ce bit est mis à 1, le résultat de l'opération contient un nombre pair de 1.

ZF (Zero Flag), le zéro indique que le résultat d'une opération est nul, dans ce cas il est mis à 1.

SF (Sign flag), le signe est le bit de poids fort, huitième ou seizième bit, d'un nombre en arithmétique signé. Comme les nombres binaires négatifs sont représentés en complément à deux, **SF** représente le signe du résultat : SF =0, le nombre et positif, SF = 1, le nombre est négatif.

AF (Auxiliary Flag), la retenue intermédiaire est la retenue qui se propage du quartet (4 bits) de poids faible vers le quartet du poids fort.

OF (Overflow Flag). Quand cet indicateur est mis à 1, c'est qu'un débordement arithmétique a eu lieu. Ceci signifie que le format du résultat a dépassé les possibilités de stockage de la destination est qu'un bit significatif a été perdu.

IF (Interrupt Flag), sert à l'autorisation des interruptions externes.

TF (Trap Flag), Pas à pas, ce mode permet la mise au point d'un programme sans gestion externe.

DF (Direction Flag), selon qu'il vaut 0 ou 1, permet de balayer une suite de données par valeur croissante, décroissante des adresses. Il a son importance dans l'exécution des instructions de transfert par bloc de type MOVS, SCAS,…

3. ADRESSAGE

L'espace adressable est de 1 Mo pour le 8086/8088. Ce champ d'adresse est segmenté, chaque segment peut compter 64 Ko dont la première adresse est donnée par le contenu d'un registre segment (16 bits) multiplié par 16. Les segments peuvent être disjoints ou se recouvrir. Les octets d'un programme sont toujours pointés par le contenu de IP (Instruction pointer) par rapport au contenu de CS (Code Segment),

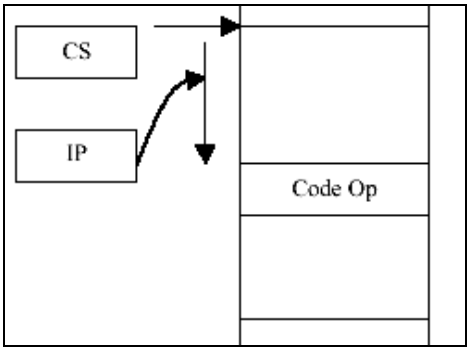


Figure 13

Les octets stockés en pile sont toujours pontés par le contenu de SP (Stack Pointer) par rapport au contenu de SS (Stack-Segment). SP pointe toujours le haut de la pile (TOS : TOP OF STACK). L'exemple suivant montre comment SP se décrémente de 2 lors d'un empilement, et s'incrémente de 2 lors du dépilement.

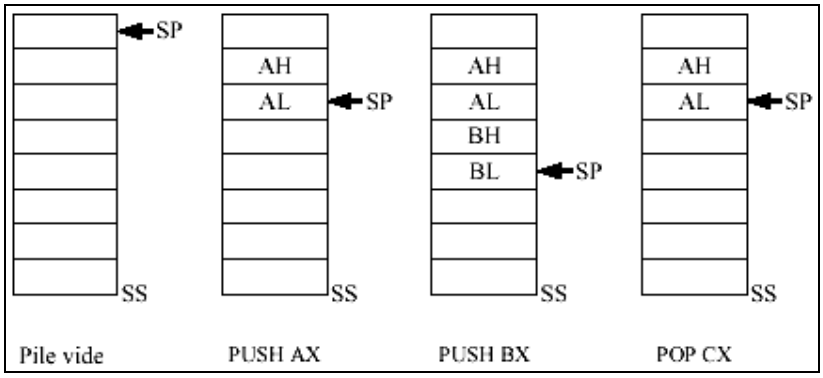


Figure 14

Les octets de données sont pointés par une adresse effective (AE) par rapport au contenu du segment DS ou ES. Cette adresse effective est le résultat de l'addition d'un mot (déplacement : DEP), d'un registre de base (BX ou BP) et d'un registre index (SI ou DI).

Tous les registres internes du 8086 sont de 16 bits ; Cependant celui-ci doit fournir une adresse physique sur 20 bits pour couvrir les 1 méga-octets. Cette adresse est calculée comme le montre la Figure 15, par la somme du contenu d'un registre segment multiplié par 16 (décalé de 4 bits) et celui d'un autre registre auquel il fait référence.

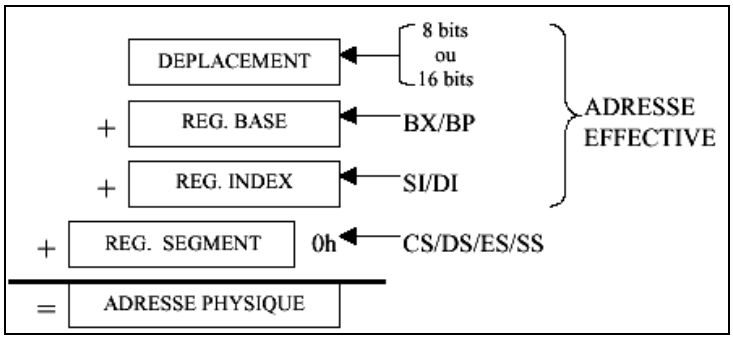


Figure 15

3.1. Mode d'adressage

Un champ adresse peut permettre de référencer un registre ou un mot en mémoire. Il peut contenir le numéro du registre ou l'adresse effective du mot mais ce ne sont pas les seules manières d'identifier un opérande. Pour faciliter la programmation il existe de nombreux modes d'adressage.

3.1.1. Adressage immédiat

La donnée est spécifiée immédiatement après l'instruction. Elle est donc située dans le segment de code.

Exemple : MOV AX, 20

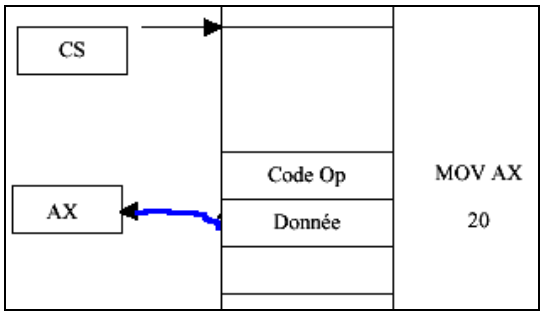


Figure 16

3.1.2. Adressage Registre

Le champ adresse contient le numéro du registre opérande.

Exemple : MOV AX, DX ; le contenu de DX est chargé dans AX

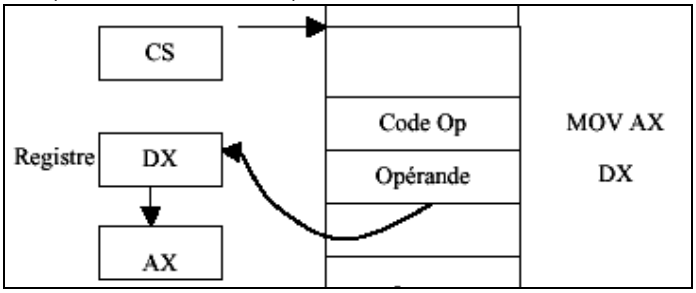


Figure 17

3.1.3. Adressage direct

Le champ d'adresse contient l'adresse effective (généralement une étiquette). L'assembleur traduit les étiquettes en offsets. Les données sont implicitement référencées par rapport au contenu de DS.

Exemple : MOV AX, DATA1 ; l'octet à l'adresse DATA1 est mis dans AL et
; celui à l'adresse DATA1 +1 est mis dans AH.

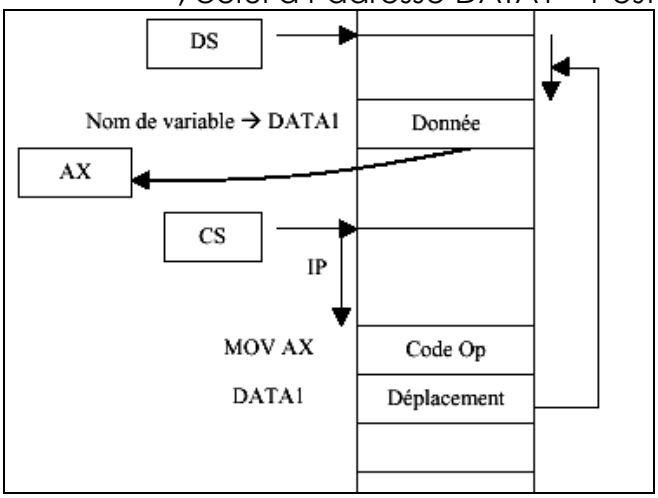


Figure 18

3.1.4. Adressage indirect

On distingue quatre modes :

Adressage indirect registre

Dans ce mode l'adresse effective est le contenu d'un registre de base (BX ou BP) ou index (SI ou DI).

Exemple : MOV AX, [BX] ; Charger AX par le contenu de la case mémoire pointée par BX.

MOV AX, [SI] ; Charger AX par le contenu de la case mémoire pointée par SI.

Adressage indirect registre avec déplacement

L'adresse effective est formée par le contenu d'un registre plus un déplacement. Les registres utilisés sont toujours les registres de base ou index.

Exemple : MOV AL, TABLE [DI]

Adressage indirect base indexée

Ce mode d'adressage utilise la somme du contenu de deux registre pour déterminer l'adresse effective.

Exemple : MOV [BP][DI], AX

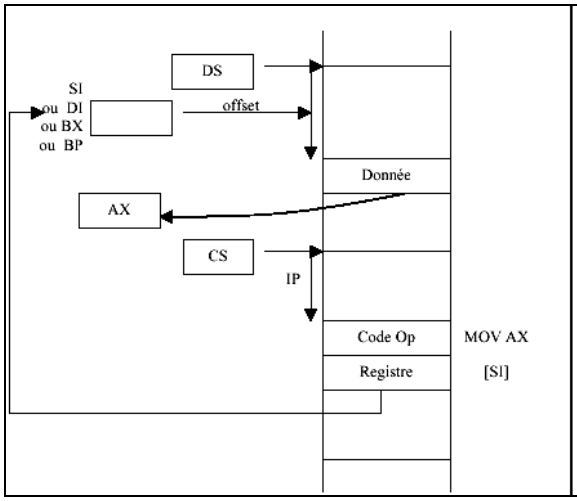
Adressage indirect base indexée + déplacement

C'est le mode d'adressage le plus complexe, il diffère du précédent par l'ajout d'une constante.

Exemple : MOV AX, COMPTE [BX][DI]

Les figures suivantes illustrent les différents modes :

Base ou Index



Base + Index

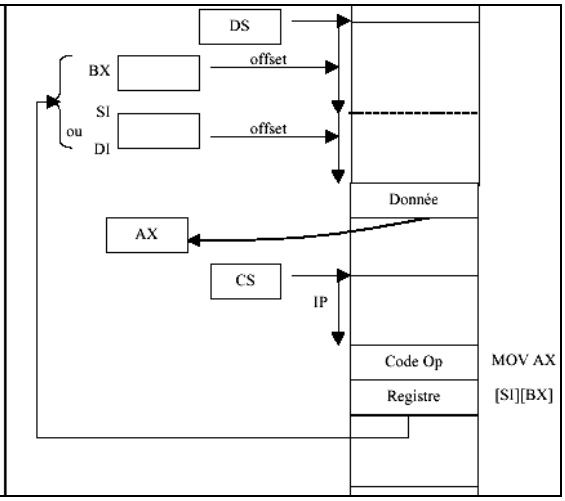
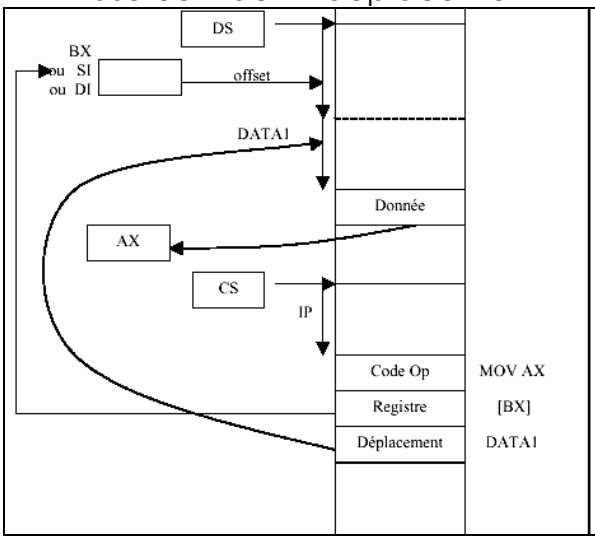


Figure 19

Base ou index + déplacement



Base + index + déplacement

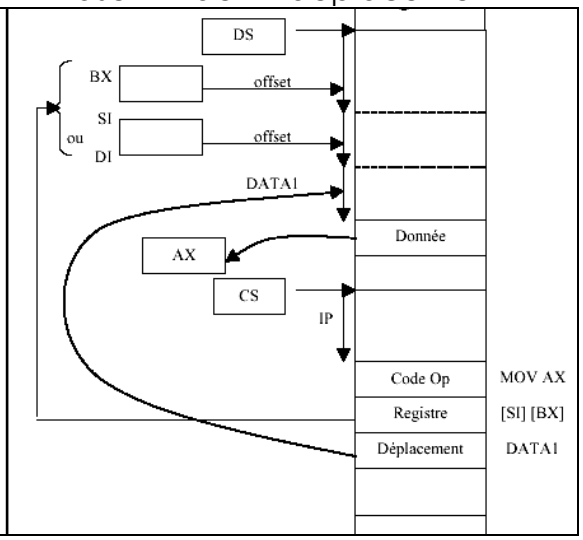


Figure 20

Exemple d'instruction	***Mode d'adressage***
MOV AX, 5	Immédiat
MOV AX, BX	Registre
MOV AX, DATA1	Direct mémoire
MOV AX, [BX]	Indirect base

MOV AX, [SI]	Indirect index
MOV AX, [BX + 5]	Indirect base + déplacement
MOV AX, [SI + 5]	Indirect index + déplacement
MOV AX, [BX][SI]	Indirect base + index
MOV AX, [BX + DI]	
MOV AX, [BX][DI] 5	Indirect base + index + déplacement
MOV AX, [BX + SI + 5]	

3.2. Organisation de la mémoire

Les µp 8086/88 possèdent 20 bits d'adresse et sont capables d'adresser une espace mémoire de 1Mo.

- **8086** : L'espace d'adresse mémoire est constitué physiquement de deux banques de 512 Ko. Un mot rangé à une adresse paire est lu en une fois. Un mot rangé à une adresse impaire est lu en deux fois selon le schéma donné par la figure suivant :
- **8088** : La mémoire est une banque unique d'un million d'octets (BHE n'existe pas et A0 est un bit d'adresse normal). Pour lire un mot de 16 bits, le 8088 qui a un bus de données de 8 bits opère en deux temps.

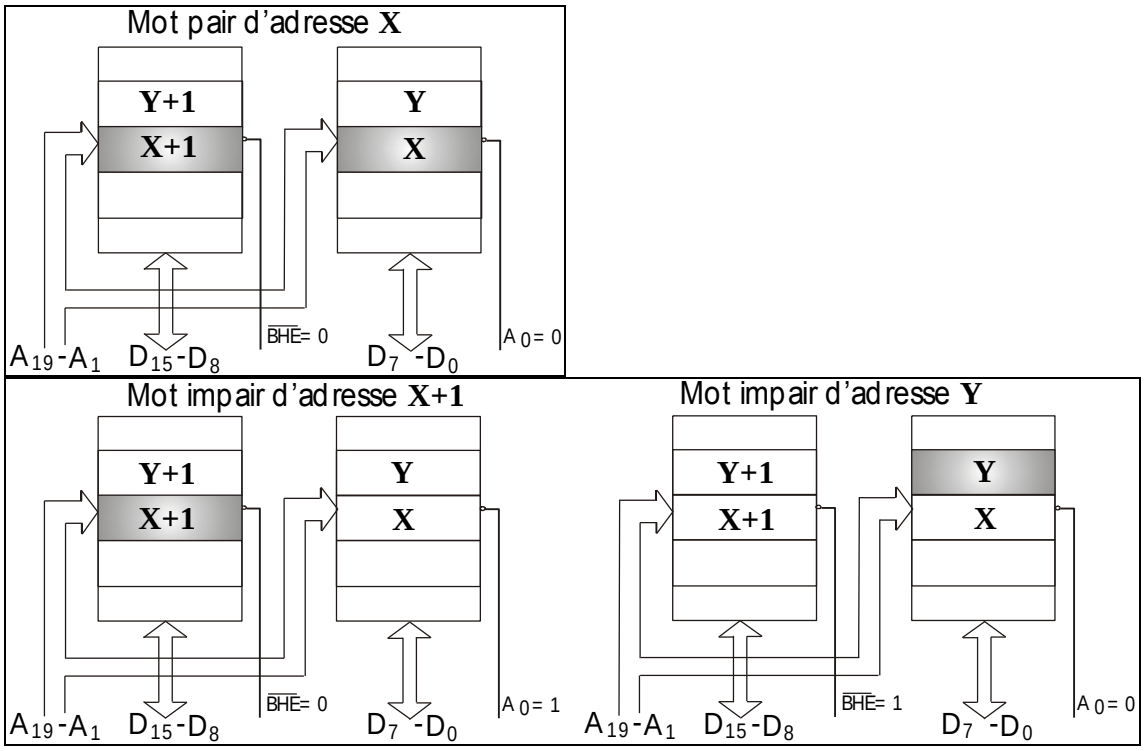


Figure 21

3.3. L'interface avec les Entrées/Sorties

Les entrées-sorties, en mode IN/OUT (I/O MAPPED) se sont vues réserver les 64 Ko du début de l'espace adressable (les quatre bits du poids fort sont toujours à zéro), les premiers 256 octets peuvent être adressés directement, pour les autres, ils sont adressés de manière indirecte à travers le registre DX.

3.4. Interruptions et zones réservées

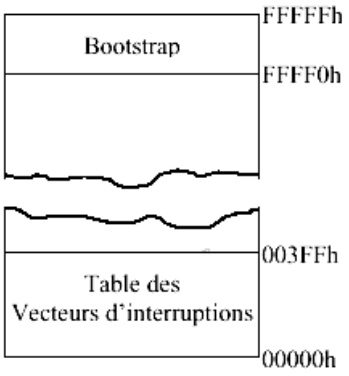
Certaines zones de la mémoire sont réservées à des usages particuliers :

a) De FFFF0h à FFFFFh, réservé au bootstrap, c'est à dire à l'initialisation du système (vecteur RESET). En effet après un RESET, le contenu des registres segments, du pointeur d'instruction et des flags sont données par le tableau suivant :

8086/88	Flags	CS	IP	DS	ES	SS
RESET	0000H	FFFFH	0000	0000H	0000H	0000H

b) De 0 à 3FFh, réservée aux vecteurs d'interruptions. Chaque vecteur nécessite 4 octets (base + offset).

Le microprocesseur est capable de reconnaître jusqu'à 256 niveaux d'interruptions. Lorsqu' une interruption est générée, le microprocesseur identifie le demandeur et vient chercher dans la table des vecteurs, l'adresse de la routine de gestion du demandeur.



4. LES INSTRUCTIONS DU 8086/8088

Le 8086/8088 possède une centaine d'instructions, qu'on va présenter dans ce paragraphe quelques unes. Une liste complète est fournie en annexe.

4.1. Les instructions de transfert

MOV Destination, Source

C'est l'instruction la plus courante, elle permet de transférer une donnée ou d'un mot entre un registre et une case mémoire ou entre deux registres, la source pouvant être une valeur immédiate. Par exemple :

```
MOV AX, 77H          ; adressage immédiat
MOV AX, [1064H]      ; adressage direct
MOV BX, OFFSET TAB1 ; charger dans BX l'adresse de TAB1

MOV AX, SEG_DATA   ; Il ne faut jamais charger un registre segment par
MOV DS, AX         ; une valeur; CS n'est jamais un registre cible.
```

Xchange XCHG

Cette instruction permet d'échanger le contenu de deux registres ou un registre et une case mémoire. Exemple :

```
XCHG AX, CX          ; pas pour les registres segments
XCHG BX, CASE_MEM
```

PUSH & POP

PUSH : Sauvegarde une donnée dans la pile,

POP : Restauration d'un mot depuis la pile,

Le mot peut être un Registre 16 bits, Registre SEGMENT ou Case Mémoire. Si le mot est de 8 bits il sera étendu à 16 bits signés.

Exp :
```
PUSH AX      ; Sauvegarde AX
POP AX       ; Restaure AX
PUSHF        ; Sauvegarde les Flags
POPF         ; Restaure F
PUSH Var1    ; Sauvegarde une variable
POP Var1     ;  Restaure Var1
```

Instructions de transfert d'adresse

LEA : Cette instruction charge l'adresse d'un opérande (offset) dans un registre.

Syntaxe : **LEA** Reg16, Mem16

```
LEA BX, Table          ; charge dans BX l'offset de Table
LEA BX, [Table + SI]   ; charge dans BX L'adresse effective Table + SI
```

LDS : chargement simultané d'un registre et du segment DS

LDS BX, Table ; Charge dans BX offset Table et dans DS le segment

Instructions d'Entrées/Sorties

IN & OUT : Transfert des données pour les périphériques,

IN Accumulateur, Port ; port compris entre 0 et 255

OUT Port, Accumulateur

Il est plus commode de charger l'adresse du port dans DX

IN AL, DX ; charger un octet dans le registre AL

OUT DX, AX ; le contenu de AX est sorti par le port d'adresse DX

4.2. Les instructions Arithmétiques

Instructions d'addition

ADD destination , Source : Addition sans retenu de destination+source

ADC destination , source : Addition avec retenu de destination+source

Exp : ADD AX,BX ; AX ←AX+BX

AAA Ajustement ASCII pour Addition en ASCII

DAA Ajustement Décimal pour Addition en BCD

Exp : ADD AL, 37H

AAA

Instructions de soustraction

SBB Soustraction avec retenue

SUB destination , source : destination ←destination - source

Exp : SBB AX,BX ; AX ←AX-BX –CF

SUB AX,BX ; AX ←AX-BX

AAS Ajustement ASCII pour division

Instructions de multiplication

MUL : multiplication non signe du contenu de l'accumulateur par le continu d'un registre ou d'une case mémoire.

Exp : MUL BX ; DX&AX ←AX * BX

IMUL : multiplication signée

AAM : correction ASCII de multiplication

Instructions de division

DIV : Réalise la division non signée d'un nombre de 16 bits, contenu dans AX (ou 32bits contenu dans DX et AX) par le nombre de 8 ou 16 bits. Le quotient est stocké dans AL (ou AX) et le reste dans AH (ou DX).

DIV BX ; AL←AX/BX et AH←reste (AX/BX)

IDIV : division signe

AAD : Ajustement ASCII pour division

NEG Destination

Cette instruction va tout simplement inverser le contenu d'un registre ou d'une adresse mémoire. Pour cela, elle inverse d'abord tous les bits, et ajoute 1.

Flags modifiés : AF - OF - PF - SF – ZF

Exemple : NEG AX

INC Destination :

Cette instruction incrémente de 1 un registre ou un emplacement mémoire.

Exemple : INC AX

INC TAB[DI]

DEC Destination

Décrémente d'une unité un registre ou une case mémoire.

Exemple : DEC AL

DEC TABLE

CMP Opérande1, Opérande2

Soustrait les deux opérandes. Le résultat de la soustraction n'est pas sauvé, seuls les indicateurs sont modifiés.

Flags modifiés : AF - CF - OF - PF - SF - ZF

Exemple : CMP AX, 7

4.3. Les instructions logiques

AND Opérande1, Opérande2

Cette instruction est l'équivalant d'une porte ET logique, elle réalise un ET entre 2 opérandes. Le résultat est placé dans le premier opérande.

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : AND AH, BH

OR Opérande1, Opérande2

Encore une autre porte logique, cette porte effectue un OU entre deux opérandes.

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : OR AH, BH

XOR Opérande1, Opérande2

Cette instruction effectue un OU exclusif.

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : XOR AH, BH

NOT Destination

C'est l'équivalent de la porte logique NON, le principe de cette porte est simple, inverser tous les bits.

Flags modifiés : Aucun

Exemple : NOT AX

TEST Opérande1, Opérande2

Cette instruction va effectuer un ET logique, mais le résultat ne sera pas gardé; seul les indicateurs seront modifiés

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : TEST AL, 01010110 B

4.4. Les instructions de décalage et de rotation

En mode immédiat, seuls les décalages et rotations de 1 bit peuvent se réaliser. Pour réaliser des décalages ou rotations de plusieurs bits, il faut explicitement utiliser le registre CL.

SHR (Shift logical Right)

Décale à droite de x positions, x contenu dans le deuxième opérande, les bits entrant sont mis à 0, les bits sortant sont placés successivement dans CF, sans conservation, celui- ci prend donc la valeur du dernier bit sorti.

Flags modifiés: CF - OF - PF - SF - ZF

Exemple: SHR AH, CL

SHL / SAL (Shift logical Left / Shift Arithmetic Left)

Effectue un décalage des bits du premier opérande, les bits sortant sont insérés dans CF, les bits entrant sont mis à 0.

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : SHL AH, 1

SAR (Shift Arithmetic Right)

Cette instruction effectue un décalage vers la droite, le nombre de positions à décaler est inscrit dans le second opérande. Le bit sortant est mis dans CF, et le bit rentrant varie en fonction du signe du nombre au départ; si au départ le nombre était positif, tous les bit entrant seront de 0, sinon,

l'inverse. Comme le carry (CF) prend successivement les valeurs des bits sortant, après cette instruction CF est égal au dernier bit sorti.

Flags modifiés : CF - OF - PF - SF - ZF

Exemple : SAR AH, CL

ROR (ROtate Right)

A peu près le même style que les instructions précédentes, les bits sont décalés vers la droite, mais, cette fois, le bit sortant est injecté à dans CF et dans le bit de poids fort.

Flags modifiés : CF - OF

Exemple : ROR AH, 1

ROL (ROtate Left)

Identique à ROR, mais décale sur la gauche.

Flags modifiés : CF - OF

Exemple : ROL AL, CL

RCR (Rotate trought Cary Right)

Cette instruction agit sur les bits de l'opérande, elle les décale de x vers la droite. Le bit contenu dans CF prend la position du bit le plus fort, puis CF prend la valeur du bit sortant. X est la valeur contenue dans le deuxième opérande.

Flags modifiés : CF - OF

Exemple : RCR AH, CL

RCL (Rotate trought Cary Left)

Identique à RCR, mais déplace les bits vers la gauche.

Flags modifiés : CF - OF

Exemple : RCL AH, 1

4.5. Instructions de branchement

4.5.1. Les branchements inconditionnels

JMP : (Jump)

Cette instruction effectue un saut sans condition. Le saut peut être sur un label, une adresse mémoire, ou un registre.

Flags modifiés : Aucun

```
Exemple :    JMP AX          ; IP ← AX
             JMP Boucle
```

INT (Interrupt)

INT active la procédure d'interruption spécifiée par l'opérande de type de l'interruption. L'adresse de pointeur d'interruption est calculée en multipliant le type d'interruption par 4. On verra plus loin l'interruption INT 21H du DOS.

Cette instruction sauvegarde dans la pile les contenus des registres (Flags, CS et IP) et remet à zéro les flags TF et IF.

La routine d'interruption se termine toujours par l'instruction IRET (Interrupt Return)

CALL

Cette instruction permet l'exécution d'une procédure (sous-programme) située en dehors du programme principal.

- un CALL intra-segment sauvegarde dans la pile le contenu de IP.
- un CALL inter-segment sauvegarde dans la pile le contenu de CS et IP.

Une procédure se termine toujours par l'instruction RET (Return)

4.5.2. Les branchements conditionnels

Ces instructions permettent de sauter "une partie d'un programme" si une condition est vérifiée. Les tableaux suivants donnent la liste des instructions de saut sur test des nombres signés et non signés

Test des nombres non signés

Instruction	Condition	Indicateurs	Définition
JA	A > B	CF = 0 et ZF = 0	Jump if Above
JNBE			Jump if Not Below or Equal
JAE	A >= B	CF = 0	Jump if Above or Equal
JNB			Jump if Not Below
JNC			Jump if Not Carry
JBE	A <= B	(CF = 1 et ZF = 1) ou (CF <> ZF)	Jump if Below or Equal
JNA			Jump if Not Above
JB	A < B	CF = 1	Jump if Below
JC			Jump if Carry
JNAE			Jump if Not Above or Equal
JE	A = B	ZF = 1	Jump if Equal
JZ			Jump if Zero
JNE	A <> B	ZF = 0	Jump if Not Equal
JNZ			Jump if Not Zero

Test des nombres signés

Instruction	Condition	Indicateurs	Définition
JG	A > B	ZF = 0 et OF = SF	Jump if Greater
JNLE			Jump if Not Lower or Equal
JGE	A >= B	SF = OF	Jump if Greater or Equal
JNL			Jump if Not Lower
JNG	A <= B	(ZF = 1) ou (SF <> OF)	Jump if Not Greater
JLE			Jump if Lower or Equal
JNGE	A < B	SF <> OF	Jump if Not Greater or Equal
JL			Jump if Lower
JE	A = B	ZF = 1	Jump if Equal
JZ			Jump if Zero
JNE	A <> B	ZF = 0	Jump if Not Equal
JNZ			Jump if Not Zero

4.5.3. Les instructions de boucle

LOOP

Cette instruction effectue une répétition tant que CX n'est pas égal à zéro; celle ci décrémente CX de 1 à chaque saut.

LOOPE / LOOPZ

Cette instruction décrémente le contenu de CX, et provoque un saut si le flag ZF vaut 1 ET le contenu de CX n'est pas nul (on sort de la boucle si ZF = 0 ou CX =0).

LOOPNE / LOOPNZ

Cette instruction décrémente le contenu de CX, et provoque un saut si le flag ZF est nul ET le contenu de CX n'est pas nul (on sort de la boucle si ZF = 1 ou CX =0).

L'ASSEMBLEUR 8086/88

1. GENERALITES

1.1. Les identificateurs

Les identificateurs permettent de repérer une donnée, une ligne dans un programme (Label ou étiquette), une case mémoire, un sous programme… Ils peuvent être composés de 31 caractères ou plus, les caractères supplémentaires n'étant pas décodés.

Les identificateur peuvent être des lettres majuscules ou minuscules, des chiffres, et quelques caractères spéciaux (@, ?, _, ***les identificateurs ne doivent pas commencer par un chiffre***, ils se terminent par : deux point (:) dans le cas de label , un espace, une tabulation, ou un RC (retour chariot).

Exp : AFF_1
Tab1_2

1.2. Constantes numériques

On peut représenter les constantes en :

Binaire : 01001010B, très utilisés pour les masques

Décimal : 65 ou 65D

Héxadécimal : 72H, 0F7H ; n'oublier pas de mettre un 0 devant une lettre.

Octal (base 8) : 45O ou 45Q.

Une constante peut être définie par un nom en utilisant la directive **EQU** :

```
Exp :   Duree    EQU  12
        PORT1    EQU  0F2H
```

1.3. Les caractères

Les chaînes de caractères sont définies entre apostrophes, chaque caractère sera traduit en code ASCII (un octet par caractère).

```
Exp     'B'
        ' ASM 86'
```

1.4. Les variables

Les variables peuvent être de différents **TYPE** selon les directives suivantes :

DB *Define byte*, la variable est de un octet soit de TYPE 1,

DW *Define word*, la variable est de un mot (2 octets) soit de TYPE 2,
DD *Define double*, la variable est constituée de 2 mots soit de TYPE 4.
Les variables ont une longueur **LENGTH** et une dimension **SIZE**.
LENGTH : nombre de termes de la variable (octets, mots, double mots),
SIZE : nombre d'octets (SIZE = LENGTH * TYPE)
Exp :
CHIFFRES **DB** 0,1,2,3,4,5,6,7,8,9 : a une longueur 10 et une dimension 10
ANNEES **DW** 1980, 1981, 1992, 2001 : a une longueur 4 et une dimension 8
MESSAGES **DB** 'Assembleur ASM 86' : a une longueur 17 et une dimension 17

2. SEGMENTATION

La définition minimale du segment comprend :
- son nom, un identificateur suivi de SEGMENT,
- sa borne supérieure donnée par ENDS précédée du nom du segment.

Exp :
```
DATA  SEGMENT
        :
DATA ENDS
CODE  SEGMENT
        :
CODE ENDS
```

2.1. Segmentation et variables

A l'intérieur du segment, les variables sont caractérisées par :
- le segment dont elles dépendent,
- les adresses à l'intérieur du segment ou OFFSET
- les attribues (TYPE, LENGTH, SIZE)

Par exemples :
```
TABLES          SEGMENT
        TAB1  DB    '123456789ABCDEF'
        TAB2  DB    'AXBXCXDX'
TABLES  ENDS
```
Nous pouvant définir, l'adresse de TAB1 à l'intérieur du segment grâce à la directive ORG.

2.2. Directive ASSUME

Pour la génération du code, il faut bien préciser quels sont les registres de segment. Cette opération est dévolue à la directive **ASSUME** que l'on écrit généralement en première ligne dans le segment du code.

Exp :

```
TABLES          SEGMENT
        TAB1  DB    '123456789ABCDEF'
        TAB2  DB    'AXBXCXDX'
TABLES  ENDS
CODE    SEGMENT
        ASSUME CS : CODE, DS : TABLES
        MOV BL, TAB1
        MOV BX, TAB2                ; Erreur
        MOV BX WORD PTR Tab2        ; correct
        MOV BX, OFFSET TAB1
        MOV AX,
CODE    ENDS
END ; fin du programme
```

3. L'OPERATEUR PTR,

Permet de préciser la taille du mot concerné lors de traitement des cases mémoires avec un adressage indirect.

```
Exp :   MOV BYTE PTR [DI], 7AH
        MOV WORD PTR [DI], 7Ah     ; correct
        JMP WORD PTR [BX]       ; s'il s'agit d'un saut intra segment
        JMP DWORD PTR [BX]      ; s'il s'agit d'un saut inter segment
        PUSH WORD PTR [DI]
        JMP NEAR PTR branche  ; Saut intra segment
        JMP FAR PTR branche     ; Saut inter segment
```

4. PROCEDURE

La directive PROC permet de définir un label et un ensemble d'instructions qui est interprété comme un sous-programme. Si le sous programme est dans le même segment que le programme appelant, il est dit NEAR sinon il est FAR.

Exemple :

```
NOM_PROC    PROC    NEAR
            .
            .       Corps de la procédure
            .
            RET
NOM_PROC    ENDP
```

L'appel de la procédure se fait par l'instruction CALL suivi du nom de la procédure. **Exemple** : CALL NOM_PROC

5. FONCTIONS DOS

Pour réaliser les opérations standards (affichage, saisie, …) le système d'exploitation (ici DOS) fournit des fonctions pré-écrites :

– Affichage d'un caractère
```
MOV DL,'A'      ; caractère
MOV AH, 2       ; fonction n°2
INT 21H         ; appel système
```

– Saisie d'un caractère (avec écho)
```
MOV AH,1
INT 21H         ; (résultat dans AL)
```

– Saisie d 'un caractère (sans écho)
```
MOV AH,7
INT 21H         ; (résultat dans AL)
```

– arrêt du programme
```
MOV AH, 4CH     ; A mettre a la fin de
INT 21H         ; chaque programme
```

6. EXEMPLES DE PROGRAMMES

Affichage de l'alphabet en majuscule (alphabet.asm)

```
CODE        SEGMENT
        ASSUME CS: CODE
MAIN:   MOV DL, "A"
        MOV CX, 26
        MOV AH,2
ENCORE: INT21H
        INC DL
```

```
            LOOP ENCORE
            MOV AX, 4C00H
            INT 21H
CODE        ENDS
            END  MAIN
```

Saisie d'un caractère du clavier, et affichage de son code ASCII en binaire

```
CODE        SEGMENT
      ASSUME    CS:CODE
MAIN:       MOV AH,1          ; saisie
            INT 21H           ; le caractere lu arrive dans AL
            MOV BL, AL
            MOV CX,8          ; compteur de boucle
MASQUE:     TEST BL, 10000000B; Tester le bit le plus fort
            JNZ BIT1
BIT0:       MOV DL,"0"        ; s'il est Zéro  afficher « 0 »
            JMP AFFICHE
BIT1:       MOV DL,"1"        ; s'il est un  afficher  « 1 »
AFFICHE:    MOV AH,2          ; affichage d'un bit
            INT 21H
            SHL BL,1          ; décaler à gauche BL de 1 bits
            LOOP MASQUE       ; fin de la boucle d'affichage des 8 bits
            MOV AX,4C00H
            INT 21H
CODE        ENDS
  END       MAIN
```

Transfert d'une chaîne de caractère

```
Donnees     SEGMENT
  M1        DB    'Assalamou Alaycom'
  M2        DB    16 DUP¨('* ')
Donnees     ENDS
Code        SEGMENT
      Assume CS : Code, DS : Donnees
Main :      MOV AX, Donnees   ; Initialisation du segment de données
            MOV DS, AX
            MOV CX, 0
```

```
                MOV CL, Length M1       ; longueur de la chaine dans CX
                LEA SI, M1              ; adresse de M1 dans SI
                MOV DI, OFFSET M2       ; Adresse de M2 dans DI
Trans :         MOV AL, BYTE PTR[SI]    ; transférer un caractère de la chaîne M1
                MOV BYTE PTR[DI], AL    ; dans le chaîne M2
                INC SI                  ; passer au caractère suivant
                INC DI
                LOOP Trans              ; boucler jusqu'à la fin de chaîne
                MOV AX, 4C00h
                INT 21h
Code            ENDS
        END     Main
```

Cherche un caractère dans un tableau, et l'affiche tant de fois qu'elle exicte.

```
DATA            SEGMENT
        C       DB      "H"
        TAB     DB       "B","C","D","G","H","J","L","$"
DATA            ENDS
CODE            SEGMENT
        ASSUME CS:CODE, DS:DATA
MAIN :          MOV AX, DATA
                MOV DS,AX
                MOV SI, 0
BOUCLE:         MOV DL, TAB[SI]
                CMP AH,"$"
                JE FIN
AFF:            CMP DL,C
                JNE SUITE
                MOV AH,2
                INT 21H
SUITE:          INC SI
                JMP BOUCLE
FIN:            MOV AX, 4C00H
                NT 21H
CODE                    ENDS
END MAIN
```

CARTES A BASE DE 8086

Dans ce chapitre on va présenter le principe de fonctionnement des cartes à base du microprocesseur 8086.

1. BROCHAGE ET DESCRIPTION DES BROCHES

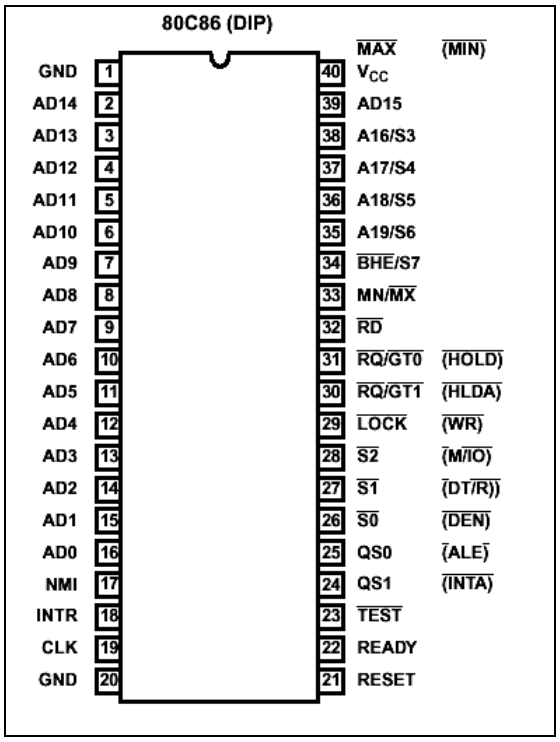


Figure 22

Le microprocesseur 8086 peut travailler selon deux modes, le mode « minimum » et le mode « maximum », les broches du mode minimum sont mises entre parenthèses.

- AD0 – AD15 : les 16 lignes d'adresses et données multiplexées
- A16 – A19 : partie haute de l'adresse. ces lignes sont aussi multiplexées avec les lignes S3 - S6.
- CLK : entrée d'horloge, cette ligne est liée à la sotie du générateur d'horloge 8284.
- $\overline{RD}$ et $\overline{WR}$: signaux de lecture écriture.
- ALE (Adress Latch Enable), cette sotie indique que le microprocesseur est entrain de sortir une adresse sur le bus. Elle est utilisée pour le démultiplexage est données et les adresses.
- RESET : cette ligne sert à initialiser le microprocesseur.
- MN/$\overline{MX}$: Elle sert à choisir le mode minimum (MN) ou maximum ($\overline{MX}$).

- $DT/\overline{R}$: Cette sorite indique le sens de transfert des données.
- $\overline{DEN}$: cette sortie est mise à 0 lorsqu'il y a une donnée sur le bus. Si non elle est en état haute impédance.
- INTR : borche de demande d'interruption (Interruption masquable).
- NMI : broche de demande d'interruption non masquable.
- $\overline{INTA}$: C'est le signal de lecture d'acquittement d'interruption.
- HOLD/HOLDA : utilisés dans un système multi-maitres.
- READY : l'entrée READY est utilisée pour insérer des cycles d'attentes. Utilisée lorsqu'on connecte au microprocesseur des circuits lents.

2. INTERFAÇAGE

Le microprocesseur peut travailler selon deux modes :

- En mode « minimum », le microprocesseur gère seul son espace mémoire. Ce mode peut être « bufférisé » ou non selon le nombre de composants connectés sur le bus.
- Le mode maximum est utilisé dans des systèmes complexes (multi-cartes et multi-processeur).

La figure suivante un exemple des cartes à bases du microprocesseur 8086 en mode minimum.

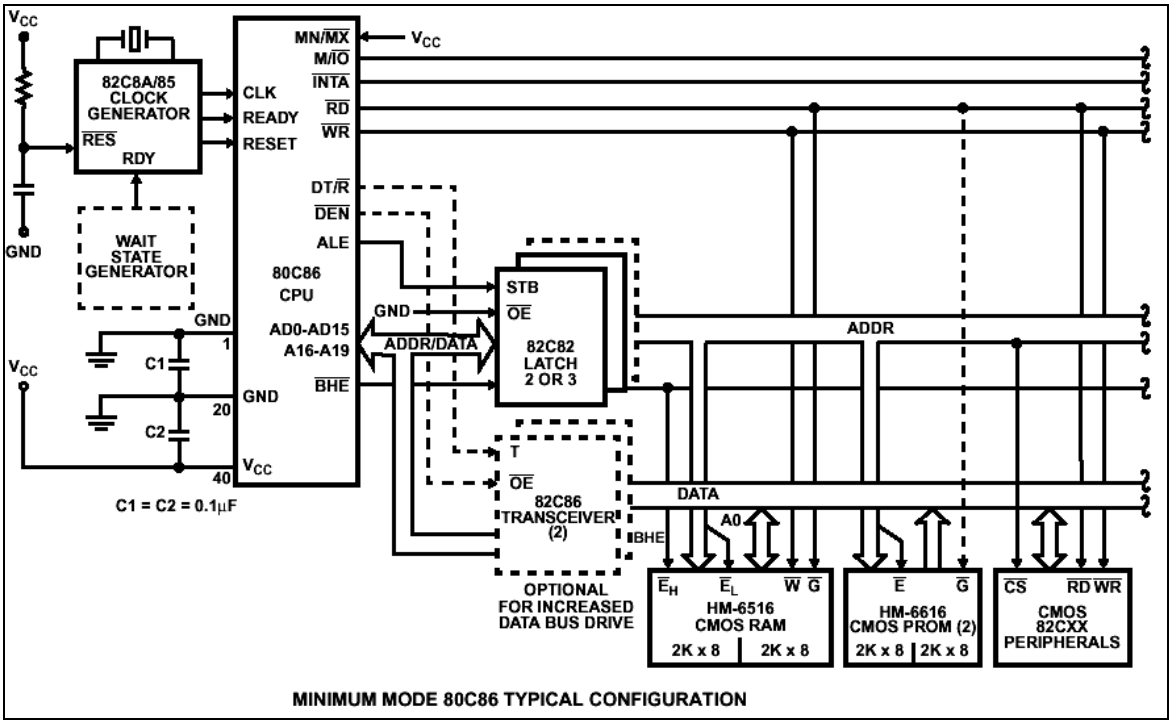


Figure 23

Le générateur d'horloge fourni au microprocesseur le signal d'horloge, de RESET et du signal d'insertion des cycles d'attente.

Les « LATCHs », 8282 sont utilisés pour le démultiplexages des données et des adresses. Le signal ALE permet le verrouillage des circuits 8282.

En remarque la présence des « buffers » bidirectionnels 8286, optionnels, permettant d'augmenter la charge du bus de données.

3. APPLICATION (EXTRET D'EXAMEN JANVIER 2002)

On considère la carte à base du microprocesseur 8086 de la figure 1.

- U2, U3 et U4 : Latch 74LS373
- U6 et U7 : EPROM 27C256
- U9 : comparateur d'égalité 8 bits
- U10 : décodeur 3 vers 8
- U11 : timer 8254
- U12 : PPI 8255
- U13 : Convertisseur Analogique numérique 8 bits
- U15, U16 : inverseur - amplificateur collecteur ouvert.

Cette carte est utilisée pour la commande d'une alimentation à découpage (figure 2). Le timer 8254 permet de délivrer la signal de commande du transistor de puissance de l'alimentation, ainsi que le signal d'horloge de convertisseur A/N. L'ADC0804 et le PPI 8255A sont utilisés dans le but de faire l'acquisition et l'affichage de la tension de sortie sur des afficheurs 7 segments.

On devra respecter dans tout le problème les notations suivantes :

- **PORTA**, **PORTB**, **PORTC** et **CW_8255**, les adresses respectives des ports A, B, C et du registre de contrôle du 8255A.
- **COMP0**, **COMP1**, **COMP2** et **CW_8254**, les adresses respectives des compteurs 0, 1, 2 et du registre de contrôle du 8254.
- **ADR_ADC** : l'adresse du convertisseur analogique – numérique.
-

1°/ Donner les adresses la plus basse et la plus haute des circuits U6 et U7.

2°/ Déterminer la capacité totale du champ mémoire adressable.

3°/ Déterminer l'adresse de chacun des ports des circuits U11 et U12.

4°/ Donner les adresses possibles du circuit U13.

5°/ Quel est le rôle des circuits U2, U3 et U4.

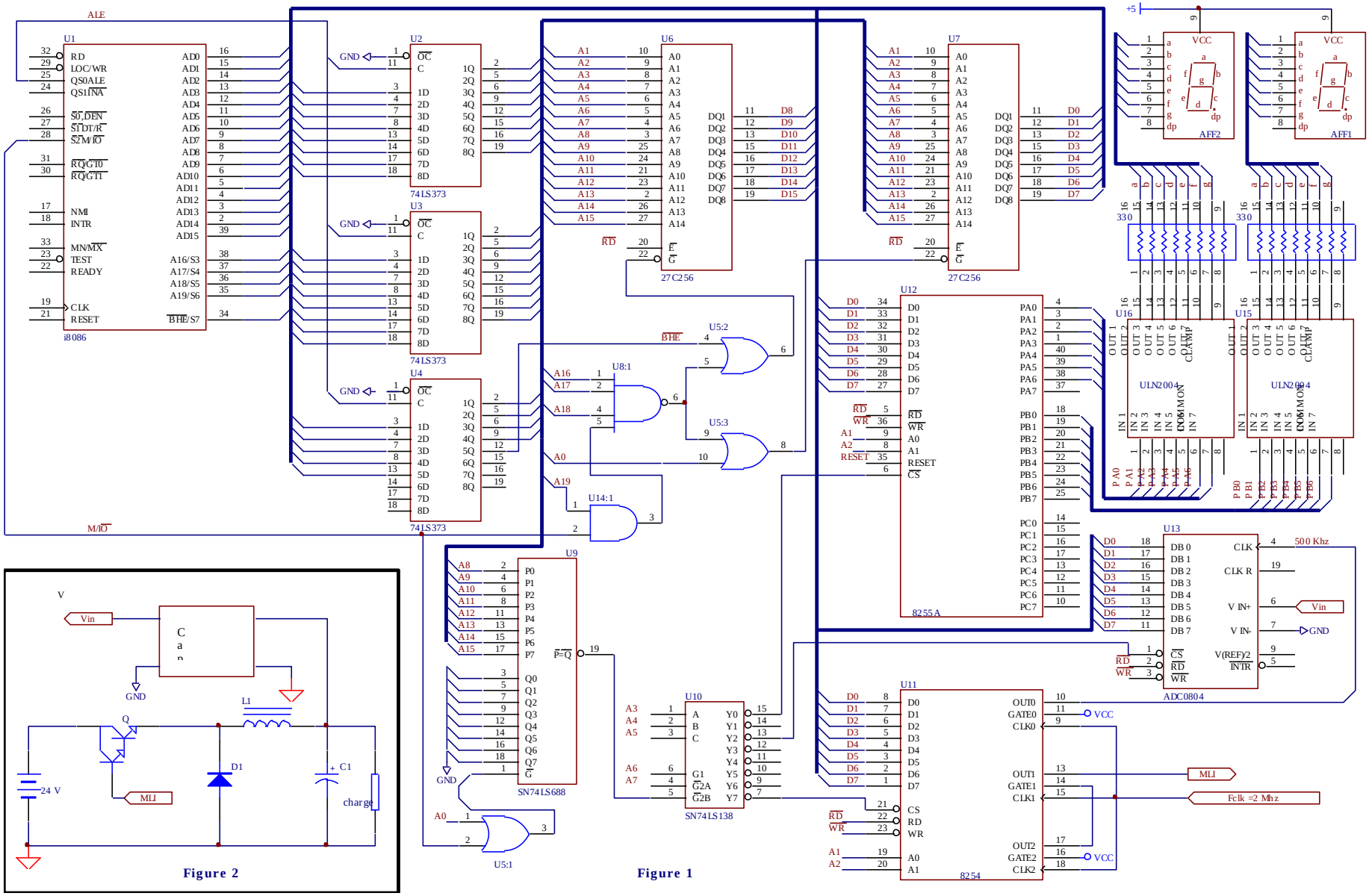


Figure 1

Figure 2

L'INTERFACE PARALLELE

1. L'INTERFACE PARALLELE 8255

Le PPI (Programmable Peripheral Interface) 8255A est un interface Programmable travaillant en mode parallèle ; il possède :

- un bus de données de 8 bits pour la communication avec le µp.
- 24 lignes programmables en entrées ou en sorties, réparties en 2 port de 8 bits (A et B) et deux demi-ports (C) de 4 lignes.
- Tension d'alimentation 5V ±5%

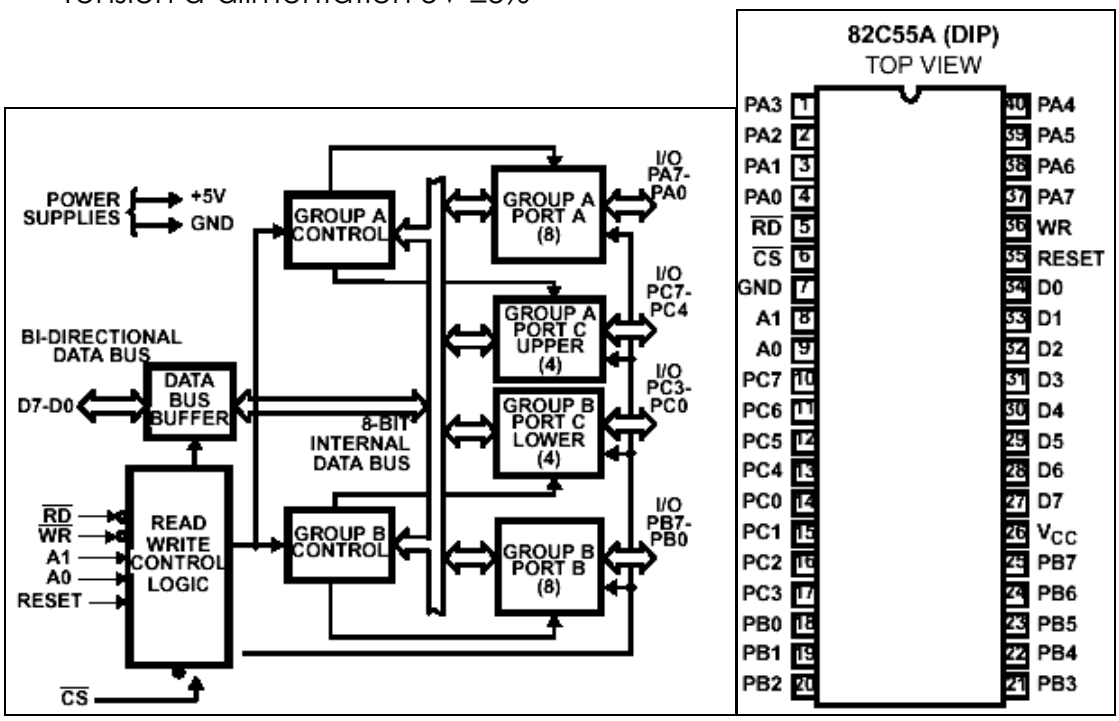


Figure 24

Le buffer de données transfère les données entre le bus de données externe et le registre de contrôle ou les ports. Quand il n'est pas sélectionné, les entrées du buffer de données sont en 3ème état. Les bits A0 et A1 sont utilisés pour sélectionner les registres internes.

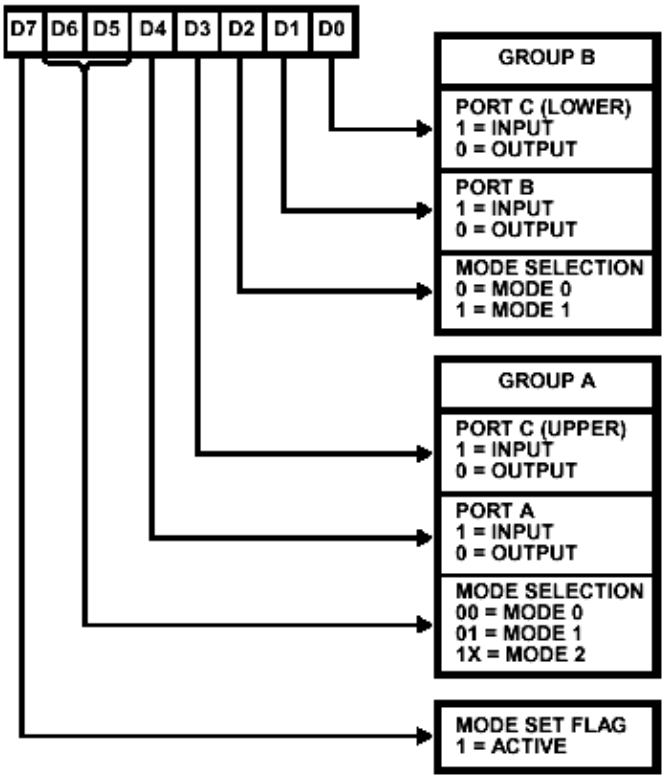
A1	A0	Port
0	0	PA
0	1	PB
1	0	PC
1	1	Reg. de contrôle

Lors de l'alimentation, le signal Reset initialise le trois ports en entrée avec un niveau logique 1 sur les broches de trois ports. Le 8255 reste dans cet état jusqu'à l'écriture du mot de commande dans le registre de contrôle, qui définit le mode de fonctionnement :

MODE 0 : Entrée/Sortie de base

MODE 1 : Entrée/Sortie échantillonné

MODE 2 : Bus bidirectionnel

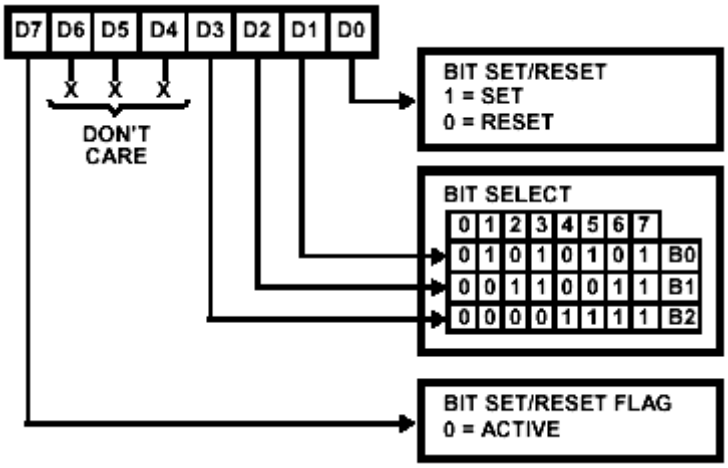


MODE 0 du 8255

Dans ce mode chaque port peut être programmé en entrée ou en sortie. Par exemple le mot de contrôle 8AH, met le port A en sortie, le quartet haute du port C en entrée, le quartet base du port C en sortie et le port B en entrée.

```
MOV DX, ADR_8255 + 3        ; adresse du registre de contrôle dans DX
MOV AL, 8AH                 ; mot de contrôle dans AL
OUT DX, AL                  ; programmation du 8255
```

Les bits du port C du 8255 pourraient être mis à 1 ou à 0 individuellement, lorsque le bit 7 du registre de contrôle est mis à 0, dans ce cas il s'agit d'une commande SET/RESET



LE TIMER 8254

1. PRESENTATION

Le PIT 8254 (Programmable Interval Timer) est constitué de trois circuits de comptage identiques de 16 bits, possédant chacun deux entrées CLK et GATE et une sortie OUT. Ce circuit est compatible avec le microprocesseur 8086/8088. La fréquence maximale d'entrée peut attendre 8 Mhz pour le 8254 et 10 Mhz pour le 8254-2.

Le timer 8254 est sélectionné par un signal bas sur l'entrée CS. Quand il n'est pas sélectionné, les entrées du buffer de données sont en 3ième état. Les lignes A0 et A1 sont utilisés pour la sélection des registres internes du circuit.

Chaque circuit de comptage possède un registre de contrôle, un registre de d'état (status), un registre compteur **CR** de 16 bits (CR_M et CR_L)pour recevoir le compte initial, un élément compteur **CE** effectuant le décomptage mais non accessible par le microprocesseur et enfin un latch de sortie **OL** (OL_M et OL_L) pour verrouiller le contenu du compteur et le rendre disponible pour la lecture. En plus le 8254 possède six modes de fonctionnement.

Les figures suivantes présentent l'architecture du circuit, le schéma interne d'un compteur et la description des broches du 8254.

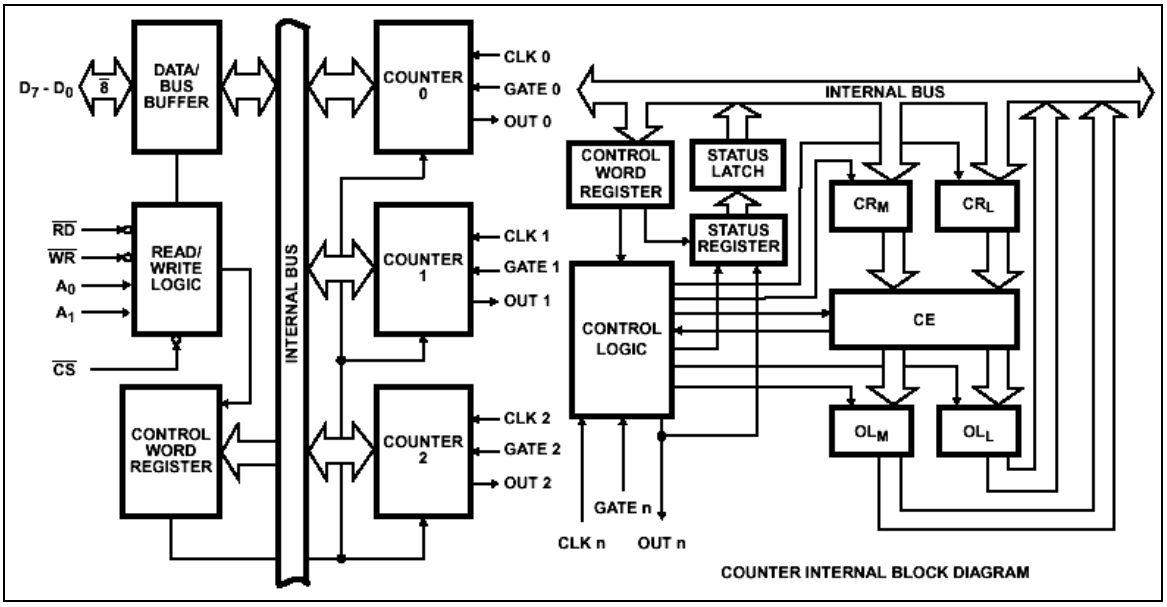


Figure 25

SYMBOL	DIP PIN NUMBER	TYPE	DEFINITION
D7 - D0	1 - 8	I/O	DATA: Bi-directional three-state data bus lines, connected to system data bus.
CLK 0	9	I	CLOCK 0: Clock input of Counter 0.
OUT 0	10	O	OUT 0: Output of Counter 0.
GATE 0	11	I	GATE 0: Gate input of Counter 0.
GND	12		GROUND: Power supply connection.
OUT 1	13	O	OUT 1: Output of Counter 1.
GATE 1	14	I	GATE 1: Gate input of Counter 1.
CLK 1	15	I	CLOCK 1: Clock input of Counter 1.
GATE 2	16	I	GATE 2: Gate input of Counter 2.
OUT 2	17	O	OUT 2: Output of Counter 2.
CLK 2	18	I	CLOCK 2: Clock input of Counter 2.
A0, A1	19 - 20	I	ADDRESS: Select inputs for one of the three counters or Control Word Register for read/write operations. Normally connected to the system address bus.
$\overline{CS}$	21	I	CHIP SELECT: A low on this input enables the 82C54 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
$\overline{RD}$	22	I	READ: This input is low during CPU read operations.
$\overline{WR}$	23	I	WRITE: This input is low during CPU write operations.
VCC	24		VCC: The +5V power supply Pin A 0.1µF capacitor between pins VCC and GND is recommended for decoupling.

A1	A0	SELECTS
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register

2. PROGRAMMATION DU COMPTEUR

2.1. Procédure d'écriture

Après la mise sous tension, le Timer 8254 se trouve dans un état indéfini, le mode de fonctionnement, le compte initial et l'état de sorties des compteurs ne sont pas définis. La procédure de programmation de chaque compteur consiste à écrire dans l'ordre le mot de commande dans le registre de contrôle ensuite le compte d'initialisation dans le registre CR.

Le format du registre de contrôle est la suivant :

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Les bits D7 et D6 sont utilisés pour la sélection du compteur, donc pour chaque compteur il faut programmer le mot de commande. La commande Read-Back permet de lire le compte ou bien le Status.

SC - Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command

Les bits D5 et D4 sont utilisés pour

sélectionner lequel des registres va recevoir le compte CR_L, CR_M ou les deux en cas d'écriture du compte initial.

RW - Read/Write

RW1	RW0	
0	0	Counter Latch Command
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

Les bits D3, D2 et D1 utilisés pour le choix du mode de fonctionnement du compteur.

M - Mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

En fin le bit D0 permet de choisir le format de l'écriture du compte initial (binaire ou BCD).

BCD - Binary Coded Decimal

0	Binary Counter 16-bit
1	Binary Coded Decimal (BCD) Counter (4 Decades)

2.2. Procédure de lecture

La procédure de lecture permet de savoir à tout moment le contenu de l'élément compteur **CE** ; cette procédure est souvent utilisée pour calculer le temps entre deux évènements. Deux types de commande sont alors possible.

2.2.1. La commande COUNT LATCH

Cette procédure permet de verrouiller le contenu de **CE** dans le registre **OL** correspondant, pour ne pas perturber le fonctionnement de du compteur **CE**.

Dans ce cas il faut lire le contenu du compteur juste après l'écriture du mot de commande. Le format du mot de commande est le suivant :

A1, A0 = 11; $\overline{CS}$ = 0; $\overline{RD}$ = 1; $\overline{WR}$ = 0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	COUNTER
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

2.2.2. Commande Read-Back

La commande Read-Back est plus puissante que la précédente. Elle permet en plus de la lecture du compte, la lecture de l'état de chaque compteur.

Les bits D5 et D4 permettent de verrouiller le compte ou l'état du compteur (status) ; les bits D3, D2 et D1 servent pour la sélection du compteur ; cette commande a l'avantage de verrouiller le contenu de plus d'un compteur à la fois.

Le format du mot de commande est le suivant :

D7	D6	D5	D4	D3	D2	D1	D0
1	1	$\overline{COUNT}$	$\overline{STATUS}$	CNT 2	CNT 1	CNT 0	0

D5: 0 = Latch count of selected Counter (s)
D4: 0 = Latch status of selected Counter(s)
D3: 1 = Select Counter 2
D2: 1 = Select Counter 1
D1: 1 = Select Counter 0
D0: Reserved for future expansion; Must be 0

Le mot de d'état (status) est obtenu en recopiant les 6 bits de faible poids du mot de contrôle ; les bits D7 et D6 permettent de savoir respectivement l'état de la proches OUT du compteur et qu'il est entrain de compter ou non.

D7	D6	D5	D4	D3	D2	D1	D0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D7: 1 = Out pin is 1
0 = Out pin is 0
D6: 1 = Null count
0 = Count available for reading
D5 - D0 = Counter programmed mode

3. LES MODES DE FONCTIONNEMENT DU 8254.

Partant d'un compte initial N, les six modes de fonctionnement sont :

3.1. Mode 0 : Interruption lors de l'épuisement du compte

GATE = 1 : valide le comptage, GATE =0 l'inhibe mais n'a aucun effet sur la sortie OUT. Un fois le microprocesseur a chargé le mot de contrôle, la sortie OUT passe à 0 et y reste jusqu'à l'épuisement de compte. Le mode 0 sert surtout au comptage d'évènements.

3.2. Mode 1 : Rédeclenchement par impulsion extérieure

La sortie OUT est initialement à l'état Haut ; Après écriture de mot de commande et du compte initiale (N), le registre CR est chargé. Une transition positive sur GATE charge le contenu de CR dans CE et met la sortie OUT à l'état bas à la prochaine impulsion d'horloge. OUT reste à cet état jusqu'à l'épuisement de compte. On obtient à la sortie une impulsion négative de largeur N périodes d'horloge. Cette impulsion est rédeclenchable par GATE. Si une transition positive sur GATE survient lorsque la sortie est à l'état bas, elle prolonge la largeur de l'impulsion d'un autre cycle ; tandis qu'un nouveau changement de CR, n'affecte pas l'impulsion en cours.

3.3. Mode 2 : Timer d'intervalle périodique

L'application typique de ce mode est la génération d'une interruption temps réel. La sortie est initialement au niveau haut. Après chargement de compte d'initialisation dans CR, la prochaine impulsion d'horloge transfère le contenu de CR dans CE, et le décomptage commence ; quand le compte arrive à 1, la sortie passe à 0 pendant une période d'horloge, puis repasse à 1 ; CE est rechargé à nouveau par le contenu de CR et le cycle recommence. La mode 2 est périodique.

GATE =1 valide le comptage, GATE = 0 l'inhibe. Si GATE passe à l'état bas durant l'impulsion de sortie, OUT passe immédiatement à 1 ; le retour de GATE au niveau haut réinitialise le comptage à la prochaine impulsion d'horloge. L'entrée GATE peut être utilisée pour la synchronisation du compteur.

L'écriture d'un nouveau compte pendant le décomptage n'affecte pas le cycle en cours.

3.4. Mode 3 : Générateur d'ondes carrés

Ce mode est périodique, il est similaire au mode 2, sauf que OUT passe à 0 à la moitié du compte, et reste dans cet état jusqu'à la fin du cycle.

GATE =1 : valide le comptage, GATE = 0 l'inhibe. Si GATE passe à l'état bas lorsque la sortie est à l'état bas, OUT passe immédiatement au niveau haut ; le retour du GATE au niveau haut réinitialise le comptage à la prochaine impulsion d'horloge. L'entrée GATE peut être utilisée pour la synchronisation du compteur.

L'écriture d'un nouveau compte pendant le décomptage n'affecte pas le cycle en cours. Si une impulsion sur GATE est reçue après l'écriture d'un nouveau compte, mais avant la fin du demi-cycle de l'onde carré, cette nouvelle valeur est chargée à la prochaine impulsion d'horloge, et le décomptage continu à partir de cette nouvelle valeur.

3.5. Mode 4 : Strobe déclenché par logiciel

OUT est initialement à l'état haut. Après écriture du mot de contrôle et du compte initial, le compteur sera chargé à la prochaine impulsion d'horloge. A l'épuisement de compte, la sortie passe à 0 pendant une impulsion d'horloge, puis revient à 1. Une autre séquence se déclenche par l'écriture d'un nouveau compte.

Si un nouveau compte initial est écrit pendant le décomptage, le compteur continu le décomptage à partir de cette valeur à la prochaine impulsion d'horloge.

GATE = 1 : valide le comptage, GATE =0, l'inhibe ; GATE n'affecte pas la sortie OUT.

3.6. Mode 5 : Strobe déclenché (rédeclenchable) par hardware

La sortie OUT est initialement à 1. Après chargement du compte, un front montant sur GATE, initialise le compteur à la prochaine impulsion d'horloge. Comme dans le cas du mode 4, une impulsion négative est générée à la sortie à la $(N+1)^{ième}$ impulsion d'horloge après le déclenchement du décomptage. GATE peut réinitialiser le décomptage à tout moment.

L'écriture d'un nouveau compte, n'affecte pas la séquence de comptage en cours.

3.7. Opérations communes pour tous les modes

3.7.1. Programmation

Lorsque le mot de commande d'un compteur est écrit ; sa logique de contrôle est immédiatement remis à zéro et la sortie OUT passe à l'état haut.

3.7.2. L'entrée GATE

L'entrée GATE est échantillonnée toujours au front montant du signal d'horloge. En modes 0, 2, 3 et 4 l'entrée GATE est active par niveau ; elle est alors échantillonnée au front montant du signal d'horloge.

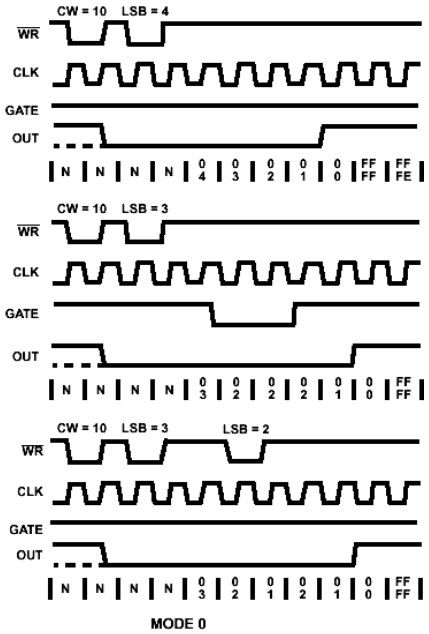
En mode 1, 2, 3 et 5 l'entrée GATE est active au front montant ; dans ce cas la transition positive sur GATE est mémorisée dans une bascule, qui sera alors échantillonnée au prochain front montant d'horloge. La bascule est remise à zéro après échantillonnage.

3.7.3. Compteur

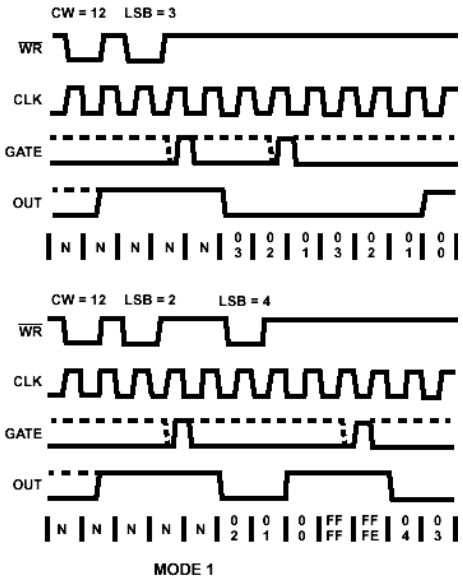
Le chargement du compte (contenu de CR dans CE) et le décomptage se font au front descendant d'horloge.

La valeur maximale du compte initial vaut FFFFH lorsqu'il compte en binaire et 9999 lorsqu'il compte en BCD.

Le compteur ne s'arrête pas lorsqu'il arrive à zéro. Les modes 2 et 3 sont périodique, le compteur se recharge automatiquement par le compte initial.



MODE 0



MODE 1

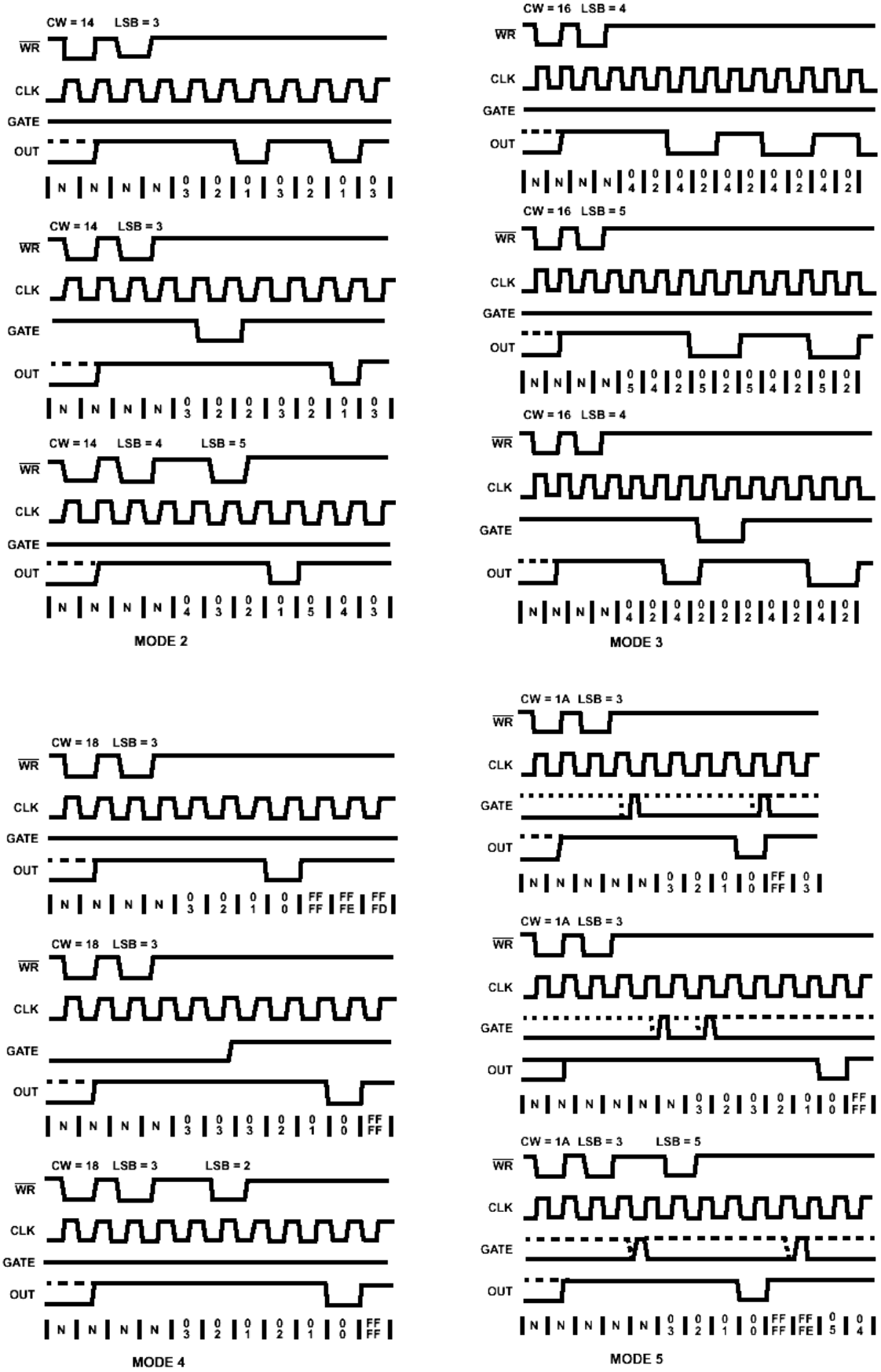

CW = 14 LSB = 3
WR
CLK
GATE
OUT
N N N N 03 02 01 03 02 01 03
CW = 14 LSB = 3
WR
CLK
GATE
OUT
N N N N 03 02 02 03 02 01 03
CW = 14 LSB = 4 LSB = 5
WR
CLK
GATE
OUT
N N N N 04 03 02 01 05 04 03
MODE 2
CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 04 02 04 02 04 02 04 02 04 02
CW = 16 LSB = 5
WR
CLK
GATE
OUT
N N N N 05 04 02 05 02 05 04 02 05 02
CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 04 02 04 02 02 02 04 02 04 02
MODE 3
CW = 18 LSB = 3
WR
CLK
GATE
OUT
N N N N 03 02 01 00 FF FF FF FE FF FD
CW = 18 LSB = 3
WR
CLK
GATE
OUT
N N N N 03 03 03 02 01 00 FF FF
CW = 18 LSB = 3 LSB = 2
WR
CLK
GATE
OUT
N N N N 03 02 01 02 01 00 FF FF
MODE 4
CW = 1A LSB = 3
WR
CLK
GATE
OUT
N N N N N 03 02 01 00 FF FF 03
CW = 1A LSB = 3
WR
CLK
GATE
OUT
N N N N N N 03 02 03 02 01 00 FF FF
CW = 1A LSB = 3 LSB = 5
WR
CLK
GATE
OUT
N N N N N 03 02 01 00 FF FF FE 05 04
MODE 5

Figure 26

L'INTERFACE SERIE « UART 8250 »

1. DESCRIPTION GENERALE

L'UART 8250 (*Universal Asynchronous Receiver Transmitter*) offre une interface bi-directionnelle série asynchrone avec un diviseur de l'oscillateur baudrate à bord et programmable (divisez par 16 * N, où N peut varier de 1 à 65535). En plus de l'entrée de données série, sortie série de données ; ce circuit a la possibilité de s'interfacer avec plusieurs architectures de microprocesseurs. Le circuit dispose de quatre broches de sortie programmable et quatre broches d'entrée qui peut être assigné au contrôle du modem et du contrôle du flux matériel (handshaking) des signaux du standard de RS232-C.

- RTS (sortie) : Demande d'émission (*Request to send*)
- CTS (entrée) : Accord de réception (*Clear to Send CTS)*
- DTR (sortie) : Terminal prêt (*Data Terminal Ready*)
- DSR (entrée) : Emission prête (*Data Set Ready*)
- DCD (entrée) : Détection de porteuse (*Data Carrier Detect*)
- RI (entrée) : Indicateur sonnerie (*Ring Indicator*)

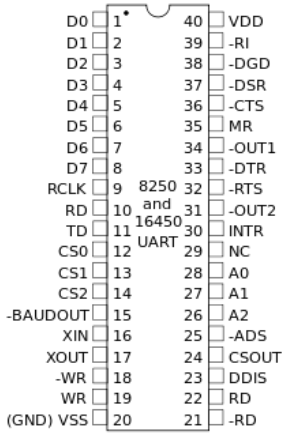


Figure 27

2. INTERFAC AGE DU CIRCUIT

Trois bits d'adresse (A2 - A0) et trois broches Chip Select (CS0 CS1 et CS2 /) permettent de déterminer l'adressage du circuit. Les broches CS doivent tous être actives pour sélectionner la circuit et les bits d'adresses servent à sélectionner les registres internes. Huit lignes bidirectionnelles (D7 - D0) permettent de véhiculer les données en le microprocesseur et le circuit. Les quatre lignes de contrôle en lecture/écriture (DISTR/DISTR et DOSTR/DOSTR) permettent au concepteur de contrôler le circuit en logique positive ou négative.

La ligne ADS (Address Strobe), peut être utilisé avec les interfaces où l'adresse et les données sont multiplexés sur les mêmes broches (exemple le

8085). La broche d'interruption (INT) passe à l'état haut chaque fois qu'une interruption est générée.

3. LES REGISTRES DE L'UART 8250

L'UART 8250, dispose de 11 registres qui sont mappés dans 8 emplacements mémoires. Cela veut dire qu'il y quelques registres qui partagent la même adresse ; le bit DLAB du registre LCR est utilisé en conjonction avec les trois lignes d'adresses A0 .. A2 pour sélectionner convenablement les différents registres.

DLAB	A2	A1	A0	registre
0	0	0	0	**RBR** : Registre buffer de réception
0	0	0	0	**THR** : Registre buffer d'émission
1	0	0	0	**DLL** : Registre diviseur LSB
1	0	0	1	**DLM** : Registre diviseur MSB
0	0	0	1	**IER** : Registre de validation des interruptions
X	0	1	0	**IIR** : Registre d'identification des interruptions
X	0	1	1	**LCR** : Registre de contrôle de ligne
X	1	0	0	**MCR** : Registre de contrôle de modem
X	1	0	1	**LSR** : Registre d'état de ligne
X	1	1	0	**MSR** : Registre d'état du modem
X	1	1	1	**SCR** : « Sratch Register », à usage général

On remarque que les registres RBR et THR sont mappés à la même adresse, ceci est logique car le registre RBR est accessible seulement en réception et le registre THR est accessible seulement en émission.

3.1. Registre contrôle de ligne (LCR)

Bits	Description			
7	***DLAB*** : Divisor Latch Access Bit			
6	0 : Force la ligne TXD à zéro (break)			
3, 4 , 5	**Bit 5**	**Bit 4**	**Bit 3**	Parité
	0	0	0	Pas de parité
	0	0	1	Parité impaire
	0	1	1	Même la parité
	1	0	1	Forcé à 1
	1	1	1	Forcé à 0

2	0	Un bit STOP	
	1	1,5 ou 2 bits de STOP	
0 , 1	**Bit 1**	**bit 0**	Longueur des mots
	0	0	5 bits
	0	1	6 Bits
	1	0	7 Bits
	1	1	8 Bits

3.2. Registre contrôle de modem (MCR)

Bits	**Description**	
7, 6, 5	Non utilisés	
4	Fonct. en boucle TXD reliée à RXD	
3	OUT2\	Permettent d'activer les broches correspondantes
2	OUT1\	
1	RTS\	
0	DTR\	

3.3. Registre d'état de ligne (LSR)

Bits	**Description**
7	Non utilisé, lu comme 0
6	Registre de décalage TSR vide
5	Registre de transmission THR vide
4	Détection de l'état de break
3	Erreur de cadrage (bit de STOP non détecté)
2	Erreur de parité
1	Erreur d'écrasement d'un caractère
0	Buffer de réception plein

3.4. Registre d'état de modem (MSR)

Bits	**Description**	
7	DCD\	Reflète l'état des broches correspondantes
6	RI\	
5	DSR\	
4	CTS\	
3	Delta DCD	

2	Delta RI	Indiquent le changement d'état des entrées correspondantes à partir de la dernière lecture par le CPU
1	Delta DSR	
0	Delta CTS	

3.5. Registre Diviseur d'horloge (DLM, DLL)

Les registres DLM et DLL forment un registre de 16 bits, ils permettent de fixer le débit de la liaison série selon la formule suivante :

$$débit(bps) = \frac{\text{fréquence d'horloge}}{16 \times (\text{DLM, DLL})}$$

Exemple : calculez la valeur à charger dans le registre diviseur d'horloge pour avoir un débit de 19200 bps, si la frequence d'horloge est égale à 1,8432Mhz.

$divieur = \frac{1{,}8432 \cdot 10^6}{16 \times 19200} = 6$ d'où DLM = 0 et DLL = 6

3.6. Registre de validation des interruptions (IER)

Bits	Description
7, 6, 5, 4	Non utilisés, lus comme 0
3	Activer l'interruption état modem
2	Activer l'interruption d'erreur
1	Activer l'interruption THR vide
0	Activer l'interruption RBR plein (caractère reçu)

3.7. Registre d'indentification des interruptions

Bits	Description				
7, 6, 5, 4, 3	Non utilisés, lus comme 0				
2, 1, 0	**2**	**1**	**0**	**Signification**	**Action à faire**
	0	0	1	Pas d'interruption en cours	-
	0	0	0	Changement d'état modem	Lire le registre MSR
	0	1	0	Registre émission vide	Ecrire la donnée dans THR
	1	0	0	Arrivé d'une donnée	Lire le buffer RBR
	1	1	0	Détection d'erreur	Lire le registre LSR

BIBLIOGRAPHIE

[1] R. TOURKI. *Cours de micro-informatique MP4* - 1992/93

[2] B. GEOFFRION. *iAPX 186 – 188 – 286 – 80386 Programmation en langage assembleur.* Edition RADIO, 1984

[3] JAMES W. COFFRON. *Programmation du 8086 – 8088* Edition SYBEX, 1983

[4] Cyril CAUCHOIS. Support de cours : Assembleur d'INTEL Institut Universitaire de Technologie d'Amiens, 1999/2000

[5] DATA SHEETS 80C86.**HARRIS SEMICONDUCTOR**, March 1997

[6] R. LITWAK Cours d'initiation aux microprocesseurs. 1999