الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière d'Electronique

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم : الهندسة الكهربائية
شعبة : الكترونيك

Microprocesseurs et ses interfaces

Support de cours : 1ere année Master
INSTRUMENTATION et Micro-informatique

Préparer par : Dr. Okba Benelmir

Année Universitaire : 2015-2016

Sommaire

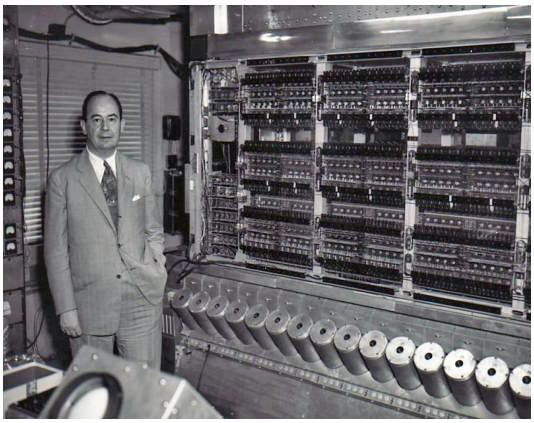
Chapitre 1 : Introduction aux microprocesseurs

I.1 Introduction

Dans les années 70 les technologies des ordinateurs numériques et celles des circuits intégrés sont fondues pour donner naissance au microprocesseur**.**

Le microprocesseur possède une architecture semblable à l'ordinateur en plus tous les deux effectuent des calculs sous le contrôle d'un programme.

La plupart des systèmes informatiques d'aujourd'hui sont basés sur un principe de conception proposée par le Dr John Von Neumann (1946).



I.2 Architecture d'un système à microprocesseur

Dans un système à microprocesseur (Architecture de Von Neumann), on retrouve au minimum : (1) Un microprocesseur, (2) Une mémoire morte (ROM) + Une mémoire vive (RAM), (3) Une interface entrées/sorties

Tous ces organes sont reliés entre eux avec 3 bus : Bus de données bidirectionnel, Bus d'adresse, Bus de commande

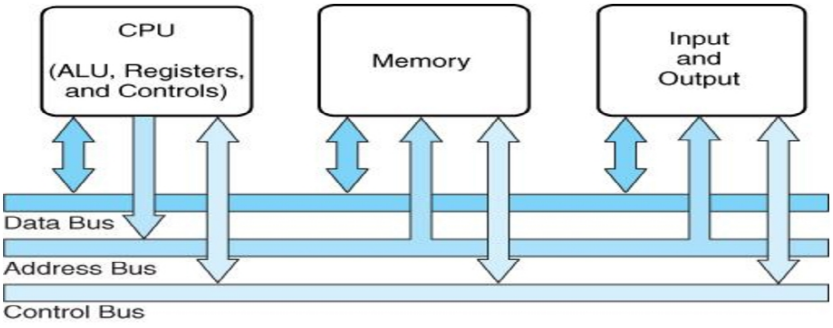


Figure I.1 : Système à microprocesseur

I.2.1 Le microprocesseur

C'est le cœur du système. C'est lui qui a la charge des fonctions suivantes :

- Fournit les signaux de synchronisation et de commande à tous les éléments du système
- Prend en charge les instructions et les données dans la mémoire
- Transfère les données entre la mémoire et les dispositifs d'I/O et vice versa
- Décode les instructions
- Effectue les opérations arithmétiques et logiques commandées par les instructions
- Réagit aux signaux de commande produits par les entrées/sorties comme le signal RESET et les INTERRUPTIONS

I.2.2 La mémoire

La mémoire a pour rôle de conserver des groupes de chiffres binaires (mots) qui sont soit des instructions formant un programme, soit des données dont le programme a besoin.

Elle se décompose souvent en :

a/ une mémoire morte (ROM = Read Only Memory) chargée de stocker le programme. C'est une mémoire à lecture seule.

• **ROM** : Read Only Memory. Mémoire à lecture seule, sans écriture. Son contenu est programmé une fois pour toutes par le constructeur. Avantage : faible coût.

Inconvénient : nécessite une production en très grande quantité.

• **PROM**: Programmable Read Only Memory. ROM programmable une seule fois par l'utilisateur.

• **EPROM** : Erasable PROM, appelée aussi UVPROM. ROM programmable électriquement avec un programmateur et effaçable par exposition à un rayonnement ultraviolet.

• **EEPROM** : Electrically Erasable PROM. ROM programmable et effaçable électriquement.

Les EEPROM contiennent des données qui peuvent être modifiées.

b/ une mémoire vive (**RAM** = Random Access Memory) chargée de stocker les données intermédiaires ou les résultats de calculs. On peut lire ou écrire des données dedans, ces données sont perdues à la mise hors tension.

• **SRAM:** Static Random Access Memory. Mémoire statique à accès aléatoire, à base de bascules à semi-conducteurs à deux états (bascules RS).

• **DRAM** : Dynamic RAM. Basée sur la charge de condensateurs : condensateur chargé = 1, condensateur déchargé = 0.

Remarque :

Les disques durs, disquettes, CDROM, etc… sont des périphériques de stockage et sont considérés comme des mémoires secondaires.

On peut donc schématiser un circuit mémoire par la figure suivante où l'on peut distinguer :

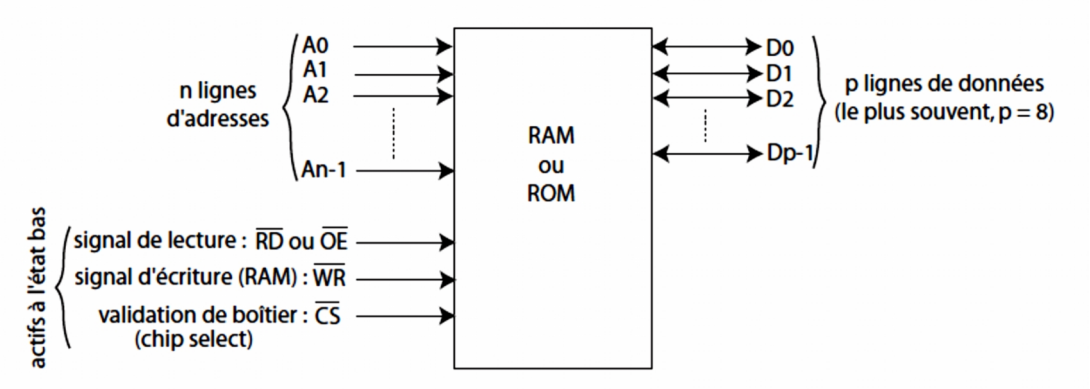


Figure I.2 : Schéma fonctionnel d'une mémoire

Les entrées d'adresses, les entrées / sorties de données, les entrées de commandes :
Entrée de sélection de lecture ou d'écriture. ($R/\overline{W}$), une entrée de sélection du circuit. ($\overline{CS}$).

Le nombre de lignes d'adresses dépend de la capacité de la mémoire : *n* lignes d'adresses permettent d'adresser 2^n cases mémoire: 8 bits d'adresses permettent d'adresser 256 octets, 16 bits d'adresses permettent d'adresser 65536 octets (= 64 Ko), ...
Exemple : mémoire RAM 6264, capacité = 8K×8 bits : 13 broches d'adresses A0 à A12,

$2^{13} = 8192 = 8$ Ko.

I.2.3 Interface I/O

C'est un circuit intégré permettant au microprocesseur de communiquer avec l'environnement extérieur (périphériques) : clavier, écran, imprimante, bouton poussoir, processus industriel, actionneurs…

Les transferts E/S correspondent à des instructions de lecture et d'écriture. Le plus souvent le circuit d'interface comporte un registre appelé « port ».

Ecriture : Le microprocesseur envoie l'ordre d'écriture et les données sont mémorisées dans le registre de l'interface.

Lecture : le circuit se contente de présenter les informations du périphérique sur le bus de données.

I.3 Architecture de base d'un microprocesseur

Le microprocesseur est construit autour de deux éléments principaux :

Une unité de traitement et Une unité de commande

I.3.1 Unité de traitement

Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions.

Les blocs de l'unité de traitement :

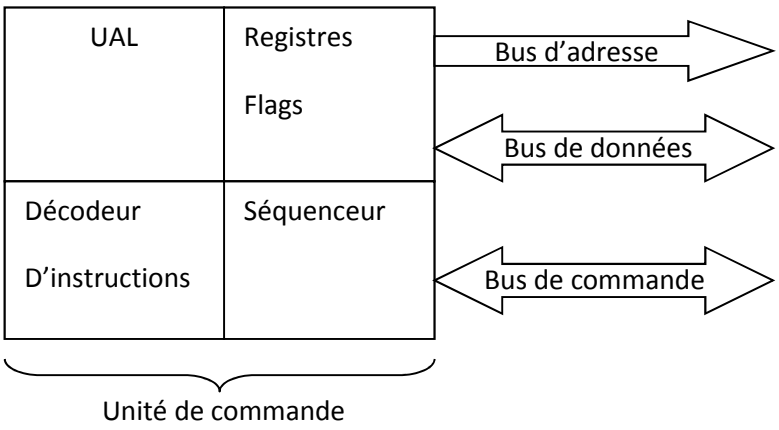


Figure I.3: Principales sections d'un microprocesseur

- **ALU (Arithmetic and Logic Unit)**

C'est l'unité qui permet d'effectuer des calculs arithmétiques simples (additions, soustractions, décalages, etc.) et des opérations logiques (ET, OU, etc.).

- **Les registres généraux**

Les registres généraux participent aux opérations arithmétiques et logiques ainsi qu'à l'adressage. Le plus important est appelé **accumulateur**.

- **Registre d'état**

C'est l'ensemble des drapeaux (flags) qui forme le registre d'état.

Les flags réagissent après certaines instructions et permettent de savoir si le résultat de l'opération est zéro (flag Z), si il y a eu un dépassement (flag C), si le nombre de bit à 1 est pair (flag P) et le signe (flag S).

I.3.2 Unité de commande

C'est l'ensemble séquenceur/décodeur d'instruction. Elle est composée par :

- **Séquenceur**

  C'est le maître d'orchestre, il cadence le microprocesseur. C'est lui qui pilote le bus de commande et les blocs internes

- **Décodeur d'instruction**

  Il reconnaît l'instruction et la transforme en signaux internes grâce à un microcode contenu dans une ROM interne.

- **Le compteur de programme**

  Constitué par un registre dont le contenu est initialisé avec l'adresse de la première instruction du programme. Il contient toujours l'adresse de l'instruction à exécuter.

- **Stack Pointer**

  Il a pour rôle d'indiquer au microprocesseur la prochaine case disponible dans la pile.

I.4 La famille de 80x86

I.4.1 Le microprocesseur 8086 (1978)

Adresse bus 20-bit.

16 bits du bus de données internes.

16 bits du bus de données externes.

L'unité d'interface de bus (BIU) et l'unité d'exécution (EU).

Registres de 16 bits (certains registres sont découpés en deux et on peut accéder séparément à la partie haute et à la partie basse.).

Il peut être utilisé avec un coprocesseur mathématique externe.

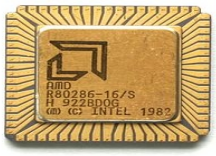
I.4.2 Le microprocesseur 8088 (1979)

Il est entièrement compatible avec le 8086, le jeu d'instruction est identique. La seule différence réside dans la taille du bus de données, celui du 8088 fait seulement 8 bits.

I.4.3 Les microprocesseurs 80186 & 80188 (1982)

Le 80186/80188 intègre sur une seule puce un microprocesseur 8086/8088 plus un générateur d'horloge, un temporisateur programmable, un contrôleur d'interruption programmable, un contrôleur d'accès direct à la mémoire et un circuit pour sélectionner les dispositifs d'E / S.

I.4.4 Le microprocesseur 80286 (1982)



Bus d'adresse de 24 bits.

16 bits du bus de données internes.

16 bits du bus de données externes.

Conçu pour être compatible avec 8086 & 80186 microprocesseurs.

Fournit deux modes de programmation:

Mode réel : le processeur fonctionne exactement comme le processeur 8086.

Mode protégé : Dans ce mode, le processeur utilise l'espace mémoire complète qui est 16MB.

Ce mode est appelé mode protégé parce que plusieurs programmes peuvent être chargés en mémoire à la fois (chacun dans son propre segment), mais ils sont protégés l'un contre l'autre.

I.4.5 Le microprocesseur 80386 (1984)

Bus d'adresse de 32 bits.

32 bits du bus de données internes.

32 bits du bus de données externes.

Des registres de 32 bits.

Offre trois modes:

Mode réel (identique à celui de 80 286)

Le mode protégé (gère 4 Go de mémoire d'une manière similaire à celle du 80 286).

Mode virtuel (similaire au mode réel, sauf que plusieurs processeurs 8086 peuvent fonctionner simultanément).

I.4.6 Le microprocesseur 80486 (1989)

Bus d'adresse de 32 bits.

Bus de données 32 bits interne et externe.

Des registres de 32 bits.

On-chip cache (stocke les instructions et les données les plus récemment utilisées)

Unité à virgule flottante intégrée (FPU)

Modes réels et protégés comme dans 80386

I.4.7 Le microprocesseur Pentium (1993)

Bus d'adresse de 32 bits.

32-bit interne

64 bits du bus de données externes.

Des registres de 32 bits.

Deux instructions pipelines

On-chip cache

FPU intégré

Chapitre 2 : Le Microprocesseur Intel 8086

II.1 Description physique du 8086

Le microprocesseur Intel 8086 est un microprocesseur 16 bits, apparu en 1978. C'est le premier microprocesseur de la famille Intel 80x86 (8086, 80186, 80286, 80386, 80486, Pentium, ...). Il se présente sous la forme d'un boitier DIP (Dual In-line Package) à 40 broches :

II.2 Schéma fonctionnel du 8086

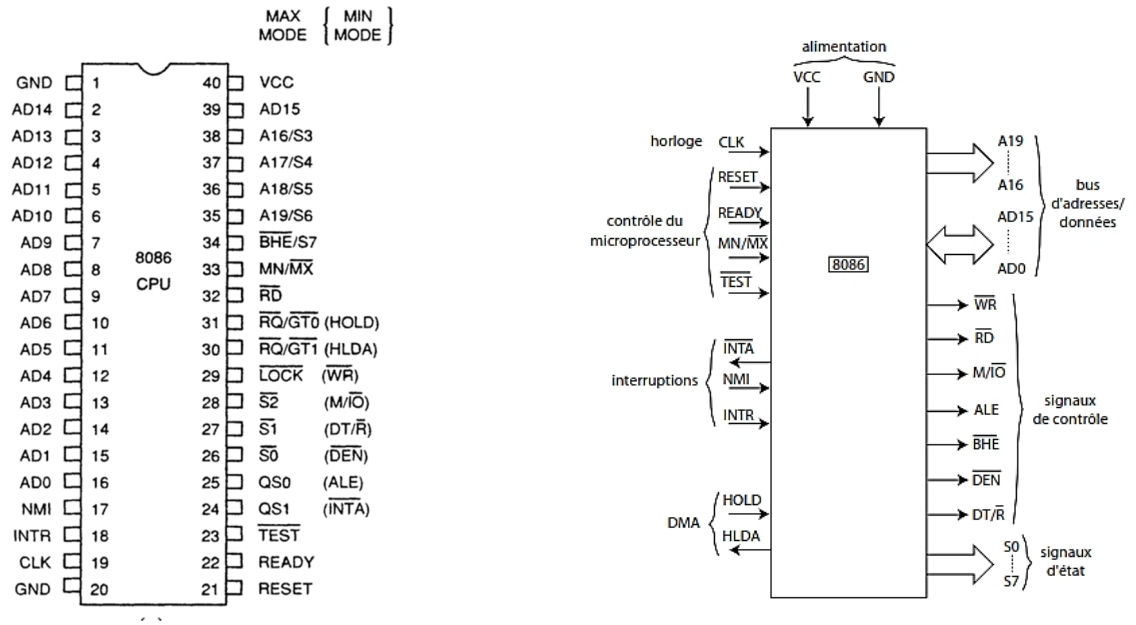


Figure II.1: Brochage et Schéma fonctionnel du 8086

Brochage et Fonctions des Pins du 8086 :

CLK : entrée du signal d'horloge qui cadence le fonctionnement du microprocesseur. Ce signal provient d'un générateur d'horloge : le 8284.

AD0 à AD15 : Ces lignes représentent 16 bits de bus d'adresse multiplexés avec 16 bits de lignes de données.

Pendant T1, elles représentent des lignes d'adresse A15-A0.

Pendant T2, T3, T4, elles représentent des lignes de données D0-D15.

A16/S3 à A19/S6 : Ces lignes adresse sont multiplexées avec les lignes d´état.

Pendant T1, elles représentent des lignes d'adresse A19-A16.

Pendant T2, T3, T4, elles représentent des signaux d'état S6-S3.

S0 à S7 : signaux d'état indiquant le type d'opération en cours sur le bus.

ALE (Address Latch Enable): Ce signal est une impulsion active pendant T1, elle indique que l'information qui circule dans bus A/D est une adresse.

Elle est fournie par le Cpu pour verrouiller les lignes d'adresse au cours des cycles T2, T3, T4.

$\overline{RD}$: Read, signal de lecture d'une donnée.

$\overline{WR}$: Write, signal d´écriture d'une donnée.

M/$\overline{IO}$: Memory/Input-Output, indique si le CPU adresse la mémoire (M/$\overline{IO}$ = 1) ou les entrées/sorties (M/IO = 0).

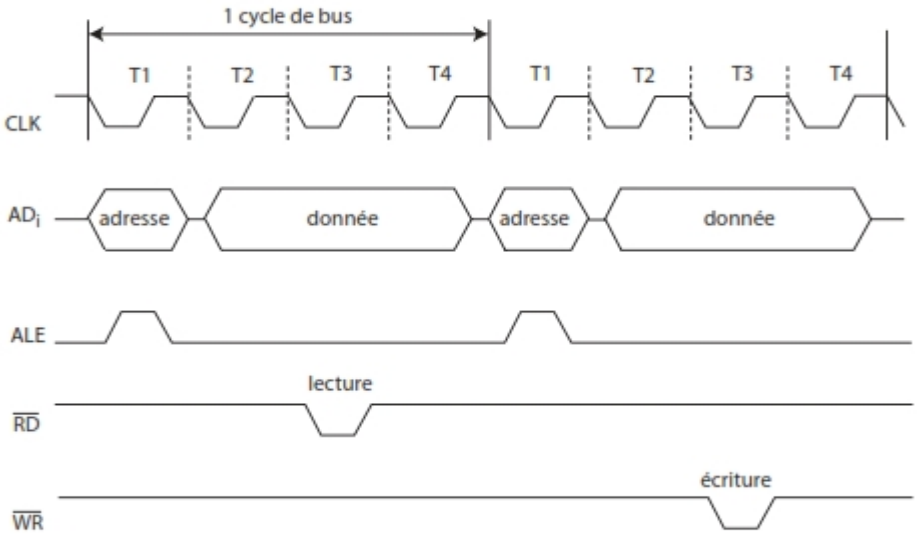
$\overline{DEN}$: Data Enable, indique que des données sont en train de circuler sur le bus A/D (équivalente de ALE pour les données).

DT/$\overline{R}$: Data Transmit/Receive, indique le sens de transfert des données :

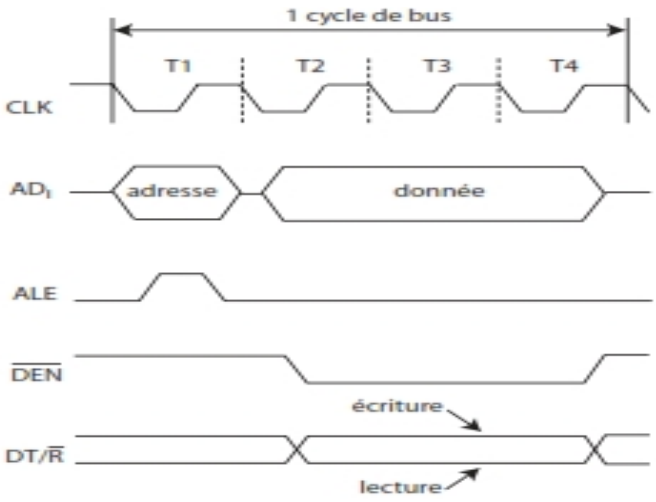
• DT/$\overline{R}$ = 1 : données émises par le microprocesseur (écriture) ;

• DT/$\overline{R}$ = 0 : données reçues par le microprocesseur (lecture).

Chronogramme de séparation de bus A/D



Chronogramme de sens de transfert de données sur le bus de données



READY : Entrée de synchronisation avec la mémoire.

TEST : Entrée de mise en attente du microprocesseur d'un évènement extérieur.

RESET : Entrée de remise à zéro du microprocesseur.

MN/MX : indique dans quel mode le processeur doit fonctionner

HIGH → minimum mode.

LOW → maximum mode.

NMI et INTR : Entrées de demande d'interruption.

INTR : interruption normale,

NMI (Non Maskable Interrupt) : interruption prioritaire.

INTA : Interrupt Acknowledge, indique que le microprocesseur accepte l'interruption.

HOLD et HLDA : Signaux de demande d'accord d'accès direct à la mémoire (DMA).

BHE : Bus High Enable, signal de lecture de l'octet de poids fort du bus de données.

Il est utilisé avec A0 pour sélectionner le mot entier, octet fort, octet faible ou aucun.

BHE	A0	Data Selected
0	0	les deux octets (mot complet) (D15-D0)
0	1	octet fort (adresse impaire) (D15-D8)
1	0	octet faible (adresse paire) (D7-D0)
1	1	aucun octet

II.2.1 Connexions de base

GND: se connecter à 0V.

VCC: se connecter à 5V.

MN / MX : se connecter à 5V (mode minimum).

NMI et INTR: se connecter à 0V (pas de support pour les interruptions).

CLK: se connecter à la sortie CLK du générateur d'horloge.

HOLD: se connecter à 0V (pas d'accès direct à la mémoire).

TEST : se connecter à 0V (pas d'attente pour les co-processeur).

READY: se connecter à 5V (pas de cycles d'attente pour les périphériques lents).

RESET: se connecter à zéro de la sortie du générateur d'horloge.

Remarque :

Un système à microprocesseur très simple est composé des parties suivantes:

(1) 8284A Clock Generator (15 MHz Crystal)

(2) 8086 Microprocessor (Minimum Mode)

(3) Bus System (Demultiplexed and Buffered)

(4) Memory System (ROM & RAM Modules)

(5) I/O System (Switches and LEDs)

II.3 Organisation interne du 8086

Le 8086 est constitué de deux unités internes distinctes:

- l'unité d'exécution (EU : Execution Unit) ;
- l'unité d'interface de bus (BIU : Bus Interface Unit).

Le rôle de la BIU est de récupérer et stocker les informations à traiter, et d'établir les transmissions avec les bus du système. L'EU exécute les instructions qui lui sont transmises par la BIU.

Pendant que l'EU du 8086/8088 exécute les informations qui lui sont transmises, l'instruction suivante est chargée dans la BIU. Les instructions qui suivront sont placées dans une file d'attente. Lorsque l'EU a fini de traiter une instruction la BIU lui transmet instantanément l'instruction suivante, et charge la troisième instruction en vue de la transmettre à l'EU. De cette façon, l'EU est continuellement en activité.

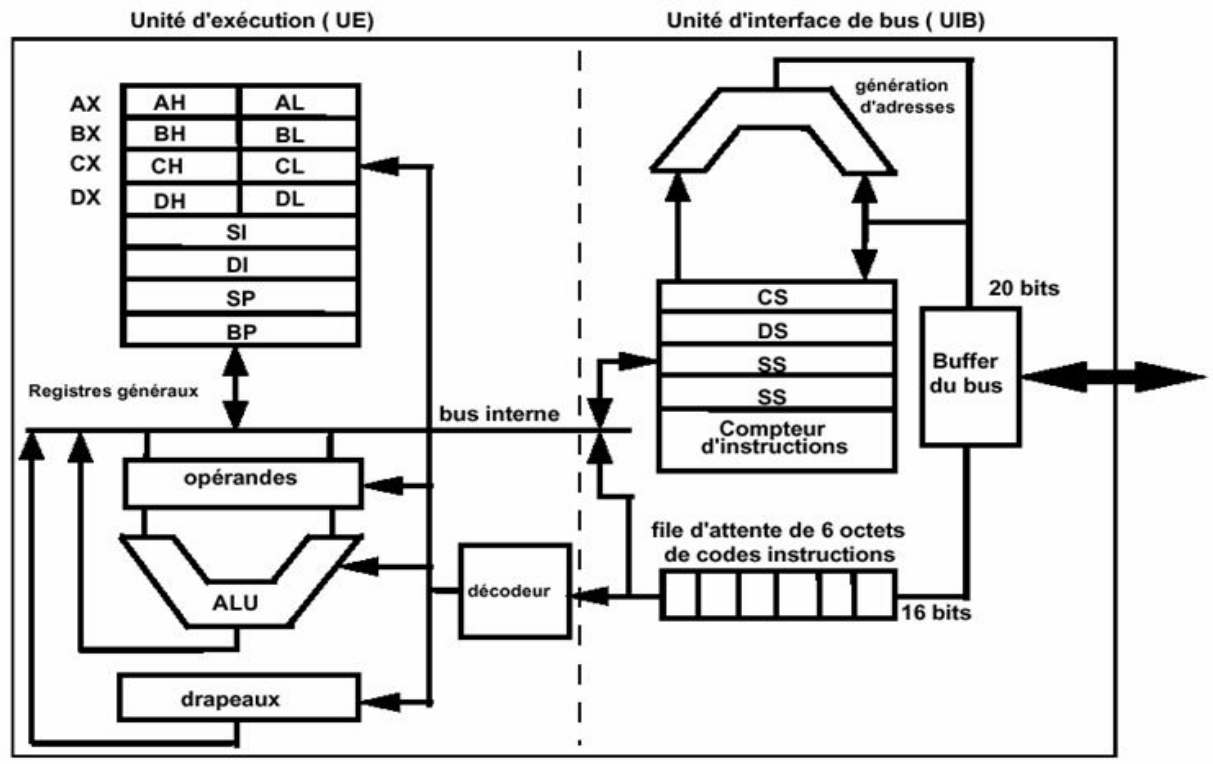


Figure II.2 : architecture du microprocesseur 8086

Le microprocesseur 8086 contient 14 registres répartis en 4 groupes :

1) Registres de travails : 4 registres sur 16 bits AX, BX, CX et DX.

Chacun de ces registres est de 16-bits de large, mais Ils peuvent être également considérés comme 8 registres sur 8 bits.

Registre AX : (Accumulateur) : registre de travail principal.

Registre BX : (registre de base) : Il est utilisé pour l'adressage.

Registre CX : (Le compteur) : le registre CX a été fait pour servir comme compteur lors des instructions de boucle.

Registre DX : On utilise le registre DX pour les opérations de multiplication et de division et pour contenir le numéro d'un port d'entrée/sortie pour adresser les interfaces d'E/S.

2) Registres de segments : 4 registres sur 16 bits.

Ils sont employés par la CPU pour déterminer les adresses de segment de mémoire.

Le registre CS (code segment) : Il pointe sur le segment qui contient les codes des instructions du programme exécutable.

Le registre DS (Data segment) : Le registre segment de données pointe sur la zone mémoire de données.

Le registre ES (Extra segment) : segment auxiliaire pour données.

Le segment SS (Stack segment) : Le registre SS pointe sur la pile.

Remarque : la pile est une zone mémoire ou on peut sauvegarder le contenu des registres, les adresses ou les données pour les récupérer après l'exécution d'un sous-programme ou l'exécution d'un programme d'interruption.

3) Registres de pointeurs et d'index : 4 registres sur 16 bits.

Ce sont des registres de 16 bits ils sont utilisés comme pointeurs de mémoire.

L'index SI : (source indexe) : Il est associé au registre DS, il sert aussi pour les instructions de chaîne de caractères, en effet il pointe sur le caractère source.

L'index DI : (Destination indexe) : Il est associé aux registres (DS ou ES), il sert aussi pour les instructions de chaîne de caractères, il pointe alors sur la destination.

Les pointeurs SP et BP : (Stack pointer et base pointer) :
Ils sont associés au registre SS, Ils pointent sur la zone pile. Pour le registre BP a un rôle proche de celui de BX, le registre SP Pointe sur la tête de la pile.

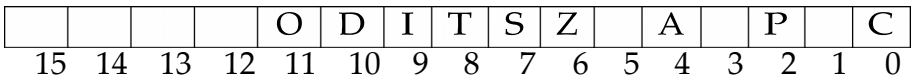
Le pointeur IP : Associé au registre CS, il a seulement une fonction qui est d'indiquer la prochaine instruction à exécuter par le CPU.

4) le registre FLAG : La plupart des instructions arithmétiques et logiques affectent le registre d'état du processeur (ou registre de FLAG).

C'est un registre de 16 bits. Six des drapeaux sont des indicateurs d'état reflétant les propriétés du résultat de la dernière instruction arithmétique ou logique.

Trois des indicateurs peuvent être activés ou remis à zéro par le programmeur: (TF, IF, DF).

Flags :



CF : indicateur de retenue (carry) ;
PF : indicateur de parité;
AF : indicateur de retenue auxiliaire ;
ZF : indicateur de zéro ;
SF : indicateur de signe ;
TF : indicateur d'exécution pas à pas (trap) ;
IF : indicateur d'autorisation d'interruption ;
DF : indicateur de décrémentation ;
OF : indicateur de dépassement (overflow).

II.4 La segmentation de la mémoire

Le processeur 8086 a un bus d'adresses du bit 20, Ceci permet au processeur d'adresser 2^{20} ou 1,048,576 emplacements mémoire différents (1 Mo).

Puisque les registres du 8086 ne font que 16 bits et avec 16 bits en peut adresser que 64 ko alors INTEL a proposé de fractionner la mémoire totale adressable de 1 Mo en bloc de 64 ko appelés segments. On utilise alors deux registres pour adresser une case mémoire (registre de segment combiné au registre d'offset), on appelle donc une **adresse logique :**

le couple : **segment : offset.**

On appelle **adresse physique :** L'adresse d'une case mémoire donnée sous la forme d'une quantité sur 20 bits.

Adresse physique = segment x 16+ offset

Exemple : CS : IP = 3F51 : 0021 donc Adresse physique = 3F510 + 0021 = 3F530h.

Chapitre 3 : **La programmation du microprocesseur 8086**

III.1 Généralités

Le programme est une suite d'instructions. Une instruction est un mot binaire décrivant l'action que doit exécuter le microprocesseur,

Une instruction est définie par son code opératoire, qu'on écrit généralement en hexa, mais pour une plus grande facilité on les désignera par leurs mnémoniques. Le mnémonique est une abréviation en caractère latin.

La programmation en assembleur utilise un ensemble d'instructions appelé jeu d'instructions. Chaque microprocesseur à son propre jeu d'instructions.

Pour les microprocesseurs classiques, le nombre d'instructions reconnues varie entre 75 et 150 (microprocesseurs CISC : Complex Instruction Set Computer). Il existe aussi des microprocesseurs dont le nombre d'instructions est très réduit (microprocesseurs RISC : Reduced Instruction Set Computer) : entre 10 et 30 instructions, permettant d'améliorer le temps d'exécution des programmes.

Les instructions peuvent être classées en groupes :

- instructions de transfert de données ;
- instructions arithmétiques ;
- instructions logiques ;
- instructions de branchement ...

III.2 Les modes d'adressage

La structure la plus générale d'une instruction est la suivante :

L'opération est réalisée entre les 2 opérandes et le résultat est toujours récupéré dans l'opérande de gauche.

Il y a aussi des instructions qui agissent sur un seul opérande.

Les opérandes peuvent être des registres, des constantes ou le contenu de cases mémoire, on appelle ça le mode d'adressage

Les modes d'adressage définissent comment identifier l'opérande de chaque instruction.

Un mode d'adressage spécifie comment calculer l'adresse de la mémoire d'un opérande en utilisant les informations contenues dans les registres et / ou Les constantes contenues dans l'instruction.

Il y a plusieurs modes d'adressage dans le 8086:

III.2.1 Adressage registre

Ce mode utilise les registres internes de CPU, dans ce mode le CPU transfert d'une copie d'un octet (8 bits) ou un mot (16 bits) d'un registre (source) vers un autre registre (destination).

Exemples : MOV AX, BX : Copier le contenu de BX dans AX

Notez que le transfert d'un registre 8 bits avec un registre 16 bits est une erreur.

III.2.2 Adressage immédiat

L'adressage immédiat transfère la source, un octet immédiat ou le mot, dans le registre de destination ou l'emplacement mémoire.

Exemples : ADD AX, 177h : Additionner le registre AX avec le nombre hexadécimal 177.

MOV AX, 'ab': Charger AH par le code ASCII du caractère 'a' et AL par le code ASCII du caractère 'b'.

III.2.3 Adressage direct

L'adressage direct déménage un octet ou un mot entre un emplacement mémoire et un registre. L'adresse de la case mémoire ou plus précisément son [Rseg : Offset] est précisé directement dans l'instruction.

Exemples : MOV AL, [1234H] : Copier le contenu de la mémoire d'adresse DS:1234 dans AL.

III.2.4 Adressage indirect par registre

L'adressage indirect de registre transfère un octet ou un mot entre un registre et un emplacement mémoire adressés par un intermédiaire (un des 4 registres d'offset BX, BP, SI ou DI).

Exemple : MOV AX, [BX] : transfert de contenu de la case mémoire 16 bits pointé par DS:BX.

MOV AX, [ES:BP] : Charger AX par le contenu de la mémoire d'adresse ES:BP.

Remarque :

L'adressage indirect est divisé en 3 catégories (l'adressage Basé, l'adressage indexé et l'adressage basé indexé) selon le registre d'offset utilisé.

III.2.5 Adressage basé

Il utilise les registres de base (BX, BP) comme pointeur. On peut ajouter un déplacement de registre offset pour déterminer l'offset,

Exemple : MOV AX, [BX] : Charger AX par le contenu de la mémoire d'adresse DS:BX

MOV AX, [BX+1000H] : Charger AX par le contenu de la mémoire d'adresse DS:BX+1000

III.2.6 Adressage indexé

Le même mode que le basé, mais il utilise les registres d'indexes (SI, DI) comme pointeur. On peut ajouter un déplacement de registre offset pour déterminer l'offset

Exemple : MOV AX, [SI] : Charger AX par le contenu de la mémoire d'adresse DS:SI

MOV AX, [SI+100] : Charger AX par la mémoire d'adresse DS:SI+100.

III.2.7 Adressage basé indexé

C'est un mode combine entre de modes d'adressage comme l'indique son nom.

Exemple : MOV AX, [BP+SI-8] : AX est chargé par la mémoire d'adresse SS:BP+SI-8

MOV AX, [BX+DI+4] : AX est chargé par la mémoire d'adresse DS:BX+DI+4

III.3 Les instructions de transfert

Elles permettent de déplacer des données d'une source vers une destination :

- registre vers mémoire ;
- registre vers registre ;
- mémoire vers registre.

Remarque : le microprocesseur 8086 n'autorise pas les transferts de mémoire vers mémoire (pour ce faire, il faut passer par un registre intermédiaire).

III.3.1 Instruction MOV

Le format général d'une instruction MOV est : **MOV Destination, Source**
(MOV Op1, Op2).
L'instruction MOV copie le contenu de l'opérande source dans l'opérande de destination.
La source peut être : Registre, mémoire, valeur immédiate (latérale).
La destination peut être : Registre, mémoire.
Toutes les combinaisons autorisées des deux opérandes sont énumérées ci-dessous:

MOV AX, BX	copier un registre dans un autre.
MOV M, AX	copier le contenu d'une case mémoire dans un registre.
MOV AX, M	copier un registre dans une case mémoire.
MOV AX, 4	copier une constante dans un registre.
MOV taille M, 4	copier une constante dans une case mémoire. (taille = BYTE ou WORD)

Les actions suivantes ne sont pas autorisées avec l'instruction MOV:

1- Interdit d'utiliser le registre (CS) comme destination.
2- Interdit de lire ou écrire dans le registre (IP).
3- Interdit de copier la valeur d'un registre de segment vers un autre registre de segment, il faut passer par une case mémoire ou par un registre interne de 16 bits.
4- Interdit de copier une valeur immédiate vers un registre de segment, il faut passer par un registre interne ou une case mémoire de 16 bits.
5- Interdit de transférer le contenu d'une case mémoire vers une autre, il faut passer par un registre interne de même taille.

III.3.2 Instruction de pile

Les instructions de pile sont des instructions importantes qui stockent et récupérent des données à partir de la pile mémoire selon le protocole LIFO (dernier entré, premier sorti). Le processeur 8086 prend en charge 4 instructions de pile:

PUSH
POP
PUSF (Push Drapeau enregistrer)
POPF (Pop Drapeau enregistrer)

a/ PUSH

Le format général d'une instruction de PUSH est : **PUSH opérande**

- L'instruction PUSH stocke le contenu de l'opérande Op (16 bits) dans le sommet de Pile
- Décrémente SP de 2

Les opérandes admis de l'instruction PUSH sont:

PUSH AX registre
PUSH word [@] une case mémoire de 16 bits.

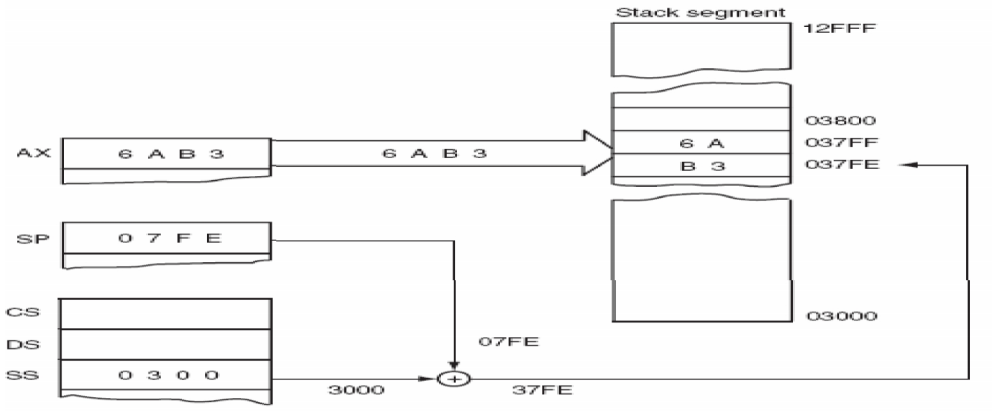


Figure III.1 : Fonctionnement de l'instruction PUSH

La figure III.1 montre le fonctionnement de l'instruction PUSH AX aprés exécution.
PUSHF : Empilement de registre FLAG vers le sommet de la PILE.

b/ POP

Le format général d'une instruction POP est : **POP opérande**.

- L'instruction POP Dépile une valeur de 16 bits depuis le sommet de la Pile (STACK SEGMENT) et la stocke dans les opérandes (16 bits).
- Incrémente SP de 2

- Les opérandes admis de l'instruction POP sont:

POP AX registre
POP word [@] une case mémoire de 16 bits.

Figure III.2 montre le fonctionnement de l'instruction POP BX aprés exécution.

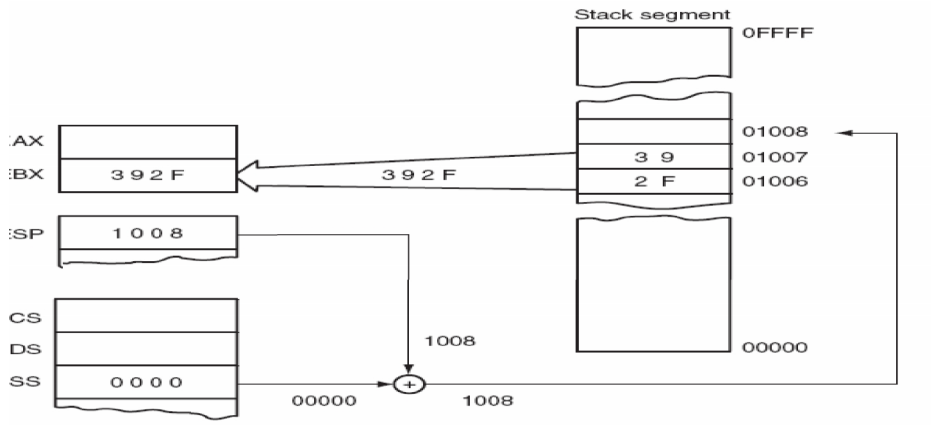


Figure III.2 : Fonctionnement de l'instruction POP

POPF : Dépilement de registre FLAG depuis le sommet de la Pile (STACK SEGMENT).

III.3.3 Instruction XCHG

Le format général de l'instruction XCHG est : **XCHG Op1, Op2.**
Echange l'opérande Source Op2 avec l'opérande Destination Op1. Impossible sur segment.
XCHG AL, AH
XCHG [@], AH
XCHG AH, [@]

III.4 Les instructions arithmétiques

Avec le 8086 on a les instructions arithmétiques de base : l'addition, la soustraction, la multiplication et la division. Qui incluent diverses variantes. Plusieurs modes d'adressage sont possibles. Les opérations peuvent s'effectuer sur des nombres de 8 bits ou de 16 bits signés ou non signés.

III.4.1 Instruction ADD

La forme générale de l'instruction ADD est : **ADD destination, source.**
L'instruction ADD Additionne l'opérande source et l'opérande destination avec résultat dans l'opérande destination, Destination ⟸ destination + source.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, valeur immédiate.
Toutes les combinaisons autorisées des deux opérandes sont énumérées ci-dessous:
Les FLAGs affectés : CF ; OF; AF ; SF ; ZF ; PF.
Exemple : ADD AX, 4 ADD AX, BX ADD [@], BX ADD AX, [SI]

III.4.2 Instruction ADC

L'instruction ADC Additionne l'opérande source, l'opérande destination et le curry avec résultat dans l'opérande destination :
ADC Destination, Source ; Destination ⟸ destination + source + CF.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, valeur immédiate.
Les FLAGs affectés : CF ; OF; AF ; SF ; ZF ; PF.

III.4.3 Instruction INC

Incrémentation d'un opérande.
INC Destination ; Destination ⟸ destination+1.
La destination peut être : registre, mémoire.
Les FLAGs affectés : OF ; AF ; SF ; ZF ; PF
CF reste inchangé
Pour incrémenter une case mémoire, il faut préciser la taille :
INC byte [ ], INC word [ ]

III.4.4 Instruction SUB

La forme générale de l'instruction SUB (Soustraction) est : **SUB destination, source.**
L'instruction SUB Soustrait l'opérande source et l'opérande destination et place le résultat dans l'opérande de destination. Destination ⟸ destination – source.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, valeur immédiate.
Les FLAGs affectés : CF ; OF; AF ; SF ; ZF ; PF.

III.4.5 Instruction SBB

L'instruction SBB fait la soustraction entre deux opérandes avec carry.
SBB Destination, Source ; Destination ⟸ destination – source – CF.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, valeur immédiate.
Les FLAGs affectés : CF ; OF; AF ; SF ; ZF ; PF.

III.4.6 Instruction DEC

Décrémentation d'un opérande : **DEC Destination** ; Destination ⟸ destination – 1.
La destination peut être : registre, mémoire.
Les FLAGs affectés : OF ; AF ; SF ; ZF ; PF.
CF reste inchangé.

III.4.7 Instruction NEG

L'instruction NEG donne le complément à 2 de l'opérande Op : remplace Op par son négatif : NEG Destination ; Destination ⟸ Cà2 (destination).
La destination peut être : registre, mémoire.
Les FLAGs affectés : CF ; OF; AF; SF ; ZF; PF.

III.4.8 Instruction CMP

L'instruction CMP fait la comparaison entre deux opérandes et positionne les drapeaux en fonction du résultat. L'opérande Op1 n'est pas modifié.
CMP Op1, Op2 ; Résultat ⟸ Op1 – Op2.
Op1 peut être : registre, mémoire.
Op2 peut être : registre, mémoire, valeur immédiate.
Les FLAGs affectés : CF ; OF ; AF ; SF ; ZF ; PF.

III.4.9 Instruction MUL

Cette instruction exécute la multiplication entre l'accumulateur et un opérande (non signé) : **MUL Opérande**.
Si l'opérande est une valeur sur 8 bits : AX ⟸ AL * Opérande.
Si l'opérande est une valeur sur 16 bits : DX : AX ⟸ AX * Opérande.
L'opérande ne peut pas être une donnée, c'est soit un registre soit une position mémoire, dans ce dernier cas, il faut préciser la taille (byte ou word)
Les FLAGs affectés : CF et OF si la partie haute du résultat est non nulle. La partie haute est AH pour la multiplication 8 bits et DX pour la multiplication 16 bits
Exemple :

```
MUL  BL   ;     AL x  BL    →    AX
MUL  CX   ;     AX x CX     →    DX:AX
```

III.4.10 Instruction DIV

Cette instruction exécute la Division entre l'accumulateur et un opérande (non signé):
DIV Opérande.
Si l'opérande sur 8 bits : AL ⟸ AX / Opérande ; AH ⟸ Reste.
Si l'opérande sur 16 bits : AX ⟸ (DX : AX) / Opérande ; DX ⟸ Reste.
L'opérande doit être soit un registre soit une mémoire. Dans le dernier cas il faut préciser la taille de l'opérande, exemple : DIV byte [adresse] ou DIV word [adresse].
Les FLAGs : CF ; OF ; ZF ; SF ; PF ; et AF sont inconnus (X).

III.4.11 Instructions IMUL et IDIV

La multiplication et la division de l'accumulateur avec un opérande (signé).
Les mêmes syntaxes que MUL et DIV.

III.5 Les instructions logiques

Ce sont des instructions qui permettent de manipuler des données au niveau des bits. Les opérations logiques de base sont : ET ; OU ; OU exclusif ; décalages et rotations.

III.5.1 Instruction AND

ET logique entre deux opérandes bit à bit : **AND Destination, Source ;**
Destination ⟸ destination (ET) source.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, immédiate.
Les FLAGs affectés : ZF ; SF ; PF

III.5.2 Instruction OR

OU logique entre deux opérandes bit à bit : **OR Destination, Source ;**
Destination ⟸ destination (OU) source.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, immédiate.
Les FLAGs affectés : ZF ; SF ; PF

III.5.3 Instruction XOR

OU Exclusif entre deux opérandes bit à bit : **XOR Destination, Source ;**
Destination ⟸ destination (ou exclusif) source.
La destination peut être : registre, mémoire.
La source peut être : registre, mémoire, immédiate.
Les FLAGs affectés : ZF ; SF ; PF
Remarque : Astuces sur les instructions logiques (OR, XOR, AND) :
1. Forçage à 1 : c'est le OR avec un « 1 ».
2. Masquage à 0 : c'est le AND avec un « 0 ».
3. Inverser un bit : c'est le XOR avec un « 1 ».

III.5.4 Instruction TEST

L'instruction TEST effectue l'opération ET.
La différence est que l'instruction AND change l'opérande de destination, alors que l'instruction TEST positionne uniquement les FLAG.
TEST Op1, Op2 ; Résultat ⟸ Op1 (ET) Op2.
Op1 peut être : registre, mémoire.
Op2 peut être : registre, mémoire, immédiate.
Les FLAGs affectés : ZF ; SF ; PF

III.5.5 Instruction NOT

Complément à 1 de l'opérande : NOT **Destination.**
La destination peut être : registre, mémoire.

III.5.6 Instructions de Décalages

Ces instructions déplacent d'un certain nombre de positions les bits d'un mot vers la gauche ou vers la droite.
Dans les décalages, les bits qui sont déplacés sont remplacés par des zéros. Il y a les décalages logiques (opérations non signées) et les décalages arithmétiques (opérations signées).
Le 8086 prend en charge quatre instructions de décalage différentes :

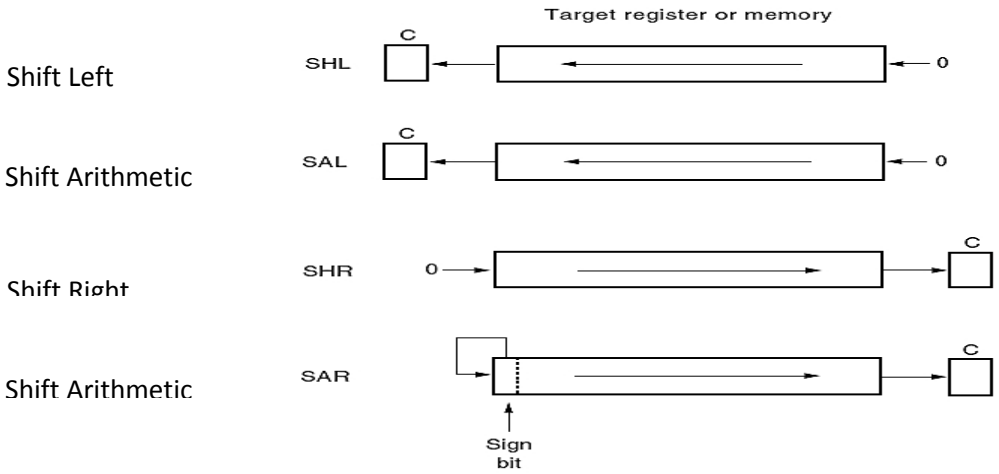


Figure III.3 Les instructions de décalage

Remarque : Les décalages arithmétiques permettent de conserver le signe. Ils sont utilisés pour effectuer des opérations arithmétiques comme des multiplications et des divisions par 2.

La forme générale des instructions de décalage est : **XXX destination, N**
Où: XXX est le mnémonique de décalage (i.e., SHL, SHR, SAL, SAR),
La destination peut être : registre, mémoire et l'opérande *N* peut être soit une constante (immédiat) soit le registre CL :

Exemples: SHL AX, 1
SHL AX, CL

III.5.7 Instructions de Rotations

Dans les rotations, les bits déplacés dans un sens sont réinjectés de l'autre côté du mot. Les quatre opérations de rotation disponibles apparaissent dans la figure suivante :

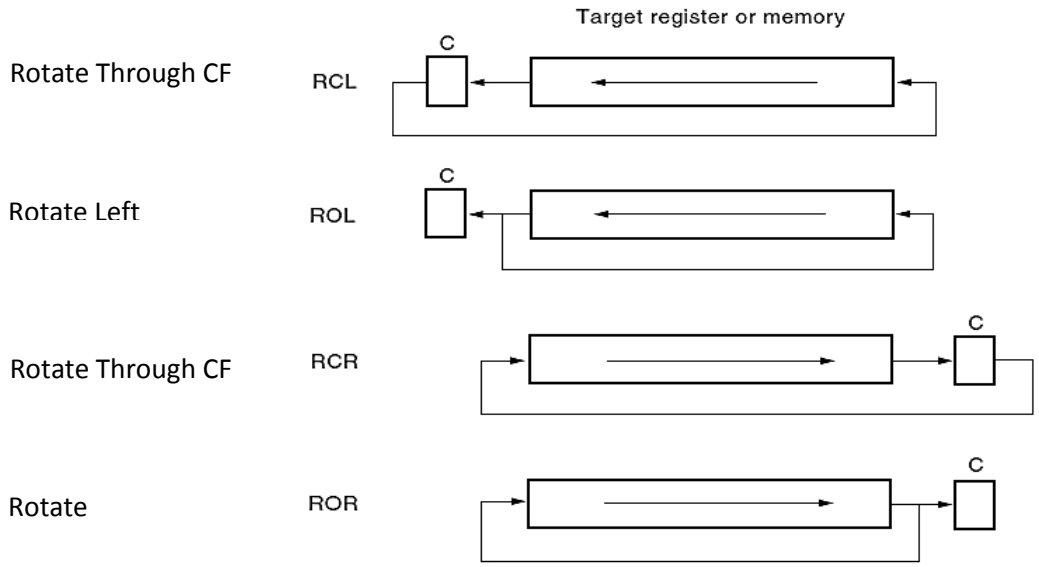


Figure III.4: instructions de rotation du registre ou de la mémoire indiquant la direction et le fonctionnement de chaque rotation.

La forme générale des instructions de rotation est : **XXX destination, N**
Où: XXX est la mnémonique de rotation (i.e., RCL, RCR, ROL, ROR),

Exemples: ROL AX, 1
ROL AX, CL

III.6 Les instructions de branchement

Les instructions de branchement (ou saut) permettent de faire un saut dans le programme, et exécuter une instruction beaucoup plus éloignée, ou au contraire revenir en arrière. Il existe 3 types de saut :

- saut inconditionnel ;
- sauts conditionnels ;
- appel de sous-programmes.

III.6.1 Instruction de saut inconditionnel

JMP label ; Provoque un saut sans condition à la ligne portant l'étiquette label.
Où label est un identificateur d'adresse de programme.

III.6.2 Instructions de sauts conditionnels

Instructions de sauts conditionnels sont conditionnées par l'état des indicateurs (drapeaux) qui sont eux même positionnés par les instructions précédentes.
Jxxx label où: xxx décrit la condition du saut.

Tableau 1: Liste des instructions de sauts conditionnels du 8086.

	Instruction	Description	Tested Condition
1	JA	Jump if above	CF = 0 and ZF = 0
2	JNBE	Jump if neither bellow nor equal	CF = 0 and ZF = 0
3	JBE	Jump if bellow or equal	CF = 1 or ZF = 1
4	JNA	Jump if not above	CF = 1 or ZF = 1
5	JAE	Jump if above or equal	CF = 0
6	JNB	Jump if not bellow	CF = 0
7	JB	Jump if bellow	CF = 1
8	JNAE	Jump if neither above nor equal	CF = 1
9	JG	Jump if greater than	ZF = 0 and SF = OF
10	JNLE	Jump if not less nor equal	ZF = 0 and SF = OF
11	JLE	Jump if less or equal	ZF = 1 or SF ≠ OF
12	JNG	Jump if not greater	ZF = 1 or SF ≠ OF
13	JGE	Jump if greater than or equal	SF = OF
14	JNL	Jump if not less than	SF = OF
15	JL	Jump if less than	SF ≠ OF
16	JNGE	Jump if neither greater nor equal	SF ≠ OF
17	JO	Jump if overflow	OF = 1
18	JNO	Jump if no overflow	OF = 0
19	JZ	Jump if zero	ZF = 1
20	JE	Jump if equal	ZF = 1
21	JNZ	Jump if not zero	ZF = 0
22	JNE	Jump if not equal	ZF = 0
23	JC	Jump on carry	CF = 1
24	JNC	Jump on not carry	CF = 0
25	JS	Jump if sign (negative)	SF = 1
26	JNS	Jump if no sign (not negative)	SF = 0
27	JP	Jump if parity	PF = 1
28	JPE	Jump if even parity	PF = 1
29	JNP	Jump if no parity	PF = 0
30	JPO	Jump if parity odd	PF = 0
31	JCXZ	Jump if CX is zero	CX = 0

Remarque :

- Les termes supérieur et inférieur (above and below) pour les nombres non signés.
- Les termes plus grand et plus petit (greater than and less than) pour les nombres signés.

III.6.3 Instruction CALL

Appel d'une procédure (sous-programme) : CALL **nom_ sous-programme**
Le CPU fait plusieurs choses quand une instruction CALL est exécutée: (1) Il ajuste le contenu du registre IP comme s'il va exécuter l'instruction suivante.

(2) Il stocke le contenu du registre IP dans la pile de programme.
(3) On ajuste le contenu du registre d'IP de nouveau pour pointer sur la première instruction de la procédure appelée.
L'unité centrale continue à exécuter les instructions jusqu'à ce qu'elle rencontre une instruction RET.

III.6.4 Instruction RET

Un RET (Return) instruction de retour de sous-programme : RET
Quand le CPU rencontre une instruction RET, il effectue les actions suivantes :
(1) Il récupère le contenu du registre IP stocké dans la pile de programme par l'instruction CALL.
(2) Il met la valeur restaurée dans le registre IP et continue l'exécution.
Le CPU revient à l'état où il était lorsque l'instruction CALL a été rencontrée.

III.6.5 Instructions des interruptions

INT n: Appel à l'interruption logicielle n° n
IRET : Cette instruction termine un sous-programme de traitement d'une interruption.

III.6.6 Instruction de boucles LOOP

L'instruction LOOP décrémente le contenu de CX et provoque un saut relatif court si ce dernier n'est pas nul.
La forme générale d'une instruction de boucle est : **LOOP label**
Où label est un identificateur d'adresse de programme.
L'instruction LOOP est équivalente au deux instructions :
DEC CX
JNZ label
La diférence entre les deux instructions ci-dessus et de l'instruction LOOP est que cette dernière, n'affecte pas les états des flags.
Le 8086 à 4 variations de l'instruction LOOP:
LOOPZ (Loop While Zero) ou **LOOPE** (Loop While Equal) : si le flag ZF=1 et si CX ≠ 0.
LOOPNZ (Loop While Not Zero) ou **LOOPNE** (Loop While Not Equal) : si le flag ZF=0 et si CX ≠ 0.
Ces variations permettent le déroulement du programme pour deux conditions (le registre CX et le drapeau zéro (ZF) en même temps).

III.7 Les instructions de manipulation de chaines

Dans le Cpu 8086 on a 5 instructions de manipulation de chaînes :
MOVS, LODS, STOS sont des instructions de transfert, elles peuvent être répétées à l'aide du préfixe **REP**
CMPS et SCAS sont des instructions de comparaisons, elles peuvent être répétées à l'aide du préfixe **REPZ**

Remarque : Chacune des instructions existe en deux versions, entre deux octets, ou entre 2 words, le préfixe REP pour les instructions de transfert MOVS LODS STOS, Ce préfixe utilise le registre CX comme un compteur de répétition.

Les préfixes REPE et REPZ : Répétition des instructions de comparaisons SCAS et CMPS tant que ZF=1 et CX≠0.

Les préfixes REPNE et REPNZ : Répétition des instructions de comparaisons SCAS et CMPS tant que ZF=0 et CX≠0.

Ces instructions sont utilisées sans opérandes. les registres utilisés sont : pour la source DS:SI, et pour la destination ES:DI. A chaque exécution d'une instruction de traitement de chaîne, les registres d'indexes sont incrémentés ou décrémentés selon le flag DF de registre d'état.

* DF = 0 : SI et DI sont incrémentés, (l'instruction CLD mit (DF = 0))
* DF = 1 : SI et DI sont décrémentés. (l' instruction STD mit (DF = 1))

SI et DI sont incrémentés de 1 ou de 2 selon que l'opération s'effectue sur un octet (byte) ou sur un mot de 16 bits (word).

MOVSB : Copie Le contenu d'un octet de mémoire DS: [SI] dans un octet de mémoire ES: [DI], puis auto inc/decrémente les registres SI et DI.

MOVSW : Copie Le contenu du mot de mémoire DS: [SI] dans le mot de mémoire ES: [DI], puis auto inc/decrémente de 2 les registres SI et DI

LODSB : Copie l'octet source DS: [SI] dans AL puis inc/décr le registre SI.

LODSW : Copie le mot source DS: [SI] dans AX puis inc/décr de 2 le registre SI.

STOSB : Copie AL dans l'octet destination ES: [DI] et inc/décr le registre DI.

STOSW : Copie AX dans le mot destination ES: [DI] et inc/décr de 2 le registre DI.

CMPSB : Compare l'octet source ES : [DI] avec l'octet destination DS: [SI], positionne les indicateurs puis inc/décrémente les registres SI et DI.

CMPSW : Compare le mot source ES : [DI] avec mot destination DS: [SI], positionne les indicateurs puis inc/décrémente de 2 les registres SI et DI

SCASB : Compare AL avec l'octet destination ES : [DI], positionne les indicateurs puis inc/décr le registre DI.

SCASW : Compare AX avec le mot destination ES : [DI], positionne les indicateurs puis inc/décr le registre DI de 2.

Chapitre 4 : Interfaçage du microprocesseur 8086

Un système à microprocesseur très simple est composé des parties suivantes:

(1) Microprocesseur 8086 (Minimum Mode)
(2) Générateur d'horloge 8284A (15 MHz Crystal)
(3) Bus système (Demultiplexed and Buffered)
(4) Mémoire système (ROM & RAM)
(5) I/O système (Switches and LEDs)

IV.1 Microprocesseur 8086 (Minimum Mode)

Le 8086 a Deux modes de fonctionnement :

Mode minimum (MN/MX = 1): Ce mode permet de diminuer le nombre de circuits extérieurs pour les systèmes à processeur unique. Le processeur prend en charge la gestion des bus.

Mode maximum (MN/MX = 0) : ces signaux de commande sont produits par un contrôleur de bus, le 8288. Ce mode permet de réaliser des systèmes multiprocesseurs.

IV.2 Générateur d'horloge

Le 8284A est un circuit intégré conçu spécialement pour les microprocesseurs 8086/8088. Ce circuit fournit les fonctions ou les signaux de bases suivantes: génération d'horloge, RESET, READY.

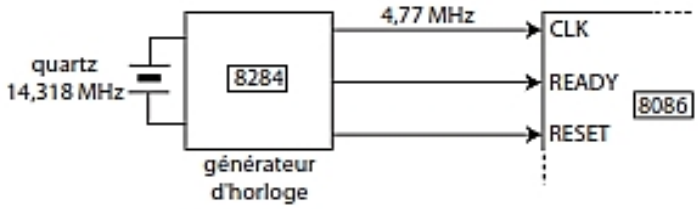


Figure IV.1 : Interfaçage du 8284 avec le 8086

IV.3 Bus système (Demultiplexed and Buffered)

Avant d'utiliser le microprocesseur 8086 avec les mémoires ou les interfaces E /S, les pines multiplexé (adresse /données et d'adresses/lignes d'état) doivent être démultiplexes. Il faut aussi amplifier le courant fourni par ces lignes pour pouvoir commander d'autres composants dans le système.

Le démultiplexage des signaux AD0 à AD15 (ouA16/S3 à A19/S6) sefait à l'aide d'un verrou (latch), ensemble de bascules D. cet événement est indiqué par le signal ALE.

Circuit de demultiplexage A/D:

Exemples de bascules D: circuits 8282, 74373, 74573.

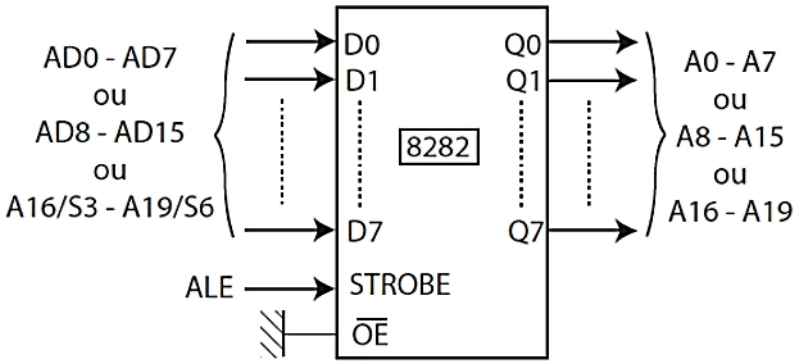


Figure IV.2 : bascules D circuits 8282

Si plus de 10 composants sont attachés aux broches de bus, l'ensemble du système 8086 doit être amplifié. Les broches démultiplexés (adresse bus A19-A0 et BHE) sont déjà amplifiées par les verrous (latch). Donc, il faut amplifier le bus de données D15 - D0. Le bus de données est bidirectionnel, nous devons utiliser des tampons (buffers) bidirectionnels :

Exemples de tampons de bus : circuits transmetteurs bidirectionnels 8286 ou 74245 :

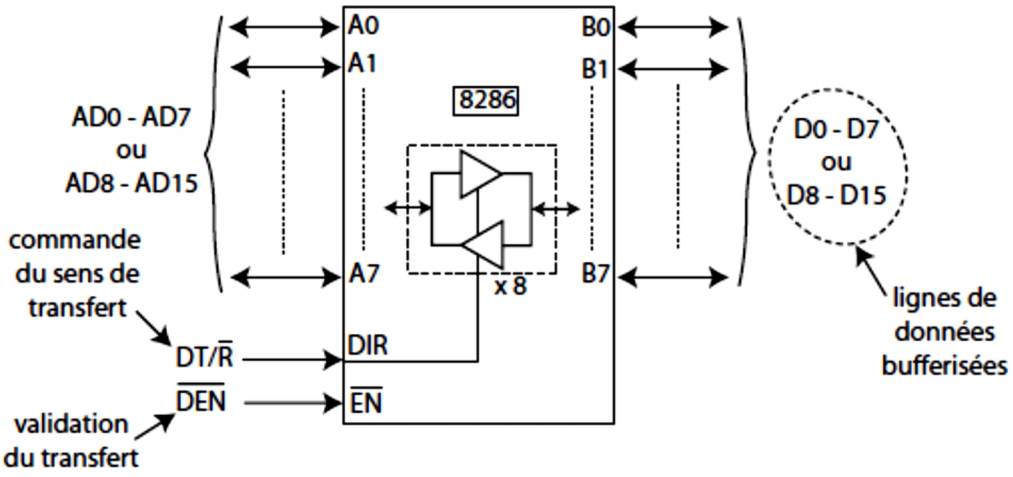


Figure IV.3 : circuits transmetteurs bidirectionnels 8286

Remarque : Les signaux de bus de contrôle (M/IO, RD et WR) les signaux de bus de contrôle sont des signaux unidirectionnels. Par conséquent, nous avons besoin d'utiliser des tampons unidirectionnels tels que le buffer 74LS244.

La figure suivante montre la conception de bus système de µp 8086

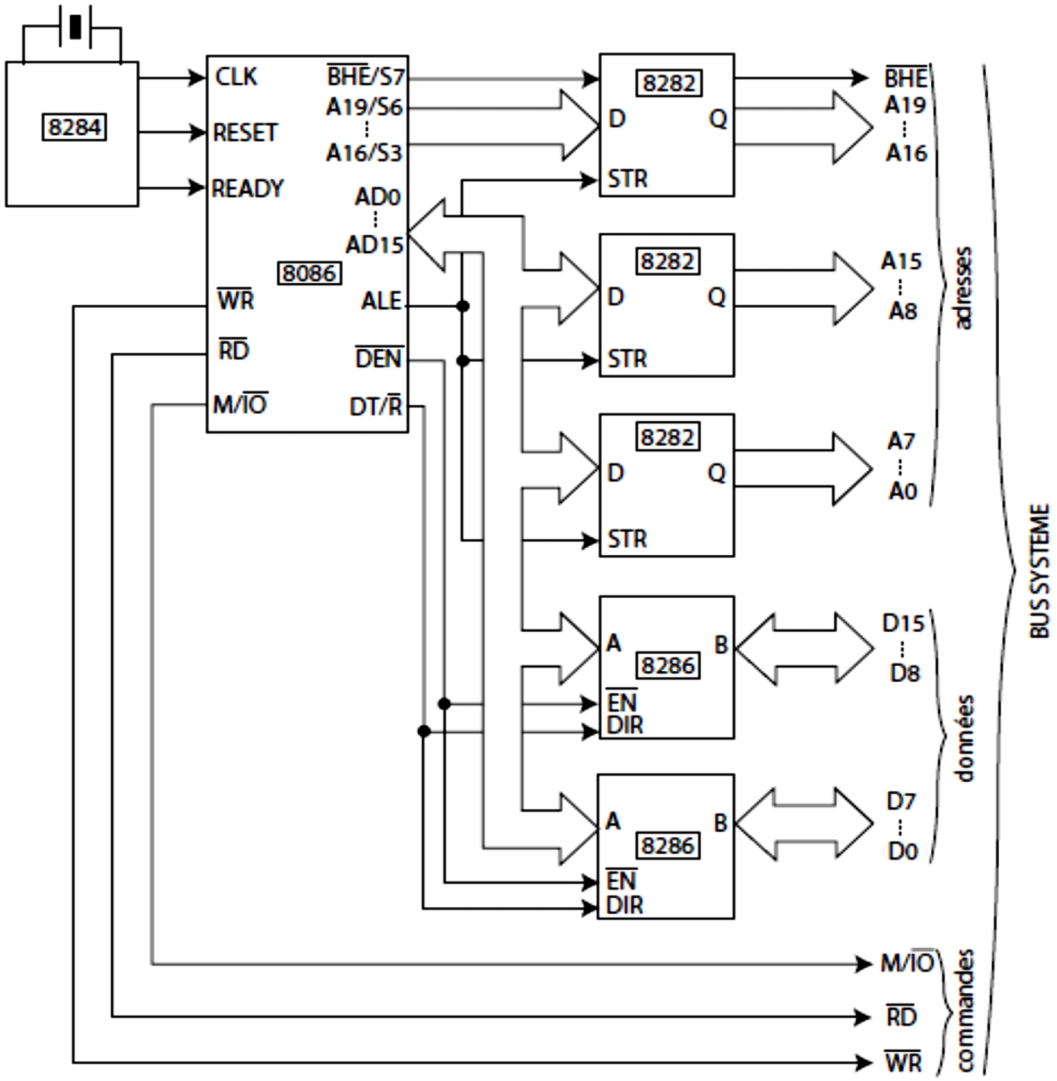


Figure IV.4 : conception de bus système de µp 8086

IV.4 Mémoire système (ROM & RAM)

Le microprocesseur 8086 à 20 bits d'adresses (A19 - A0) donc un espace mémoire D'une capacité de 1 Mo.

Par conséquent, le micro-processeur 8086 nécessite que cet espace mémoire soit organisé en deux banques (512 Ko chacune) appelées les banques paires et impaires.

La partie paire est connectée au bus de données D0/D7, et la partie impaire est connectée au bus D8/D15. Ces deux banques sont sélectionnées par A0 est BHE.

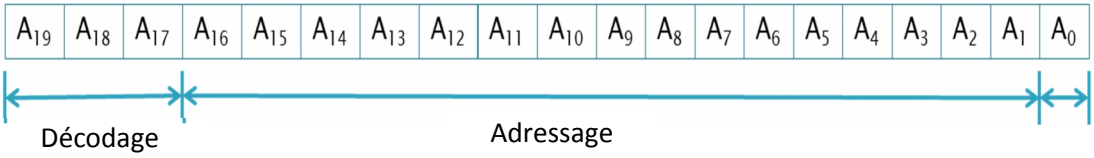
Puisque le 8086 a un espace mémoire D'une capacité de 1 Mo. D'où la possibilité d'avoir jusqu'à 8 blocs mémoire de 128Ko chacun.

Si on va utiliser deux boitiers mémoire RAM de 64 ko chacun. Donc on a 16 lignes d'adresse ($64K = 2^6 \times 2^{10} = 2^{16}$).

Le 8086 sera interfacé à la RAM comme suit:

Les lignes d'adresse les moins significatives A16-A1 sont utilisées pour adresser des emplacements de mémoire dans la RAM.

Les lignes d'adresse les plus significatives A19-A17 sont utilisées pour sélectionner la plage d'adresses nécessaires dans l'espace de mémoire de 8086 (sélectionner les deux boitiers RAM), comme sur figure.



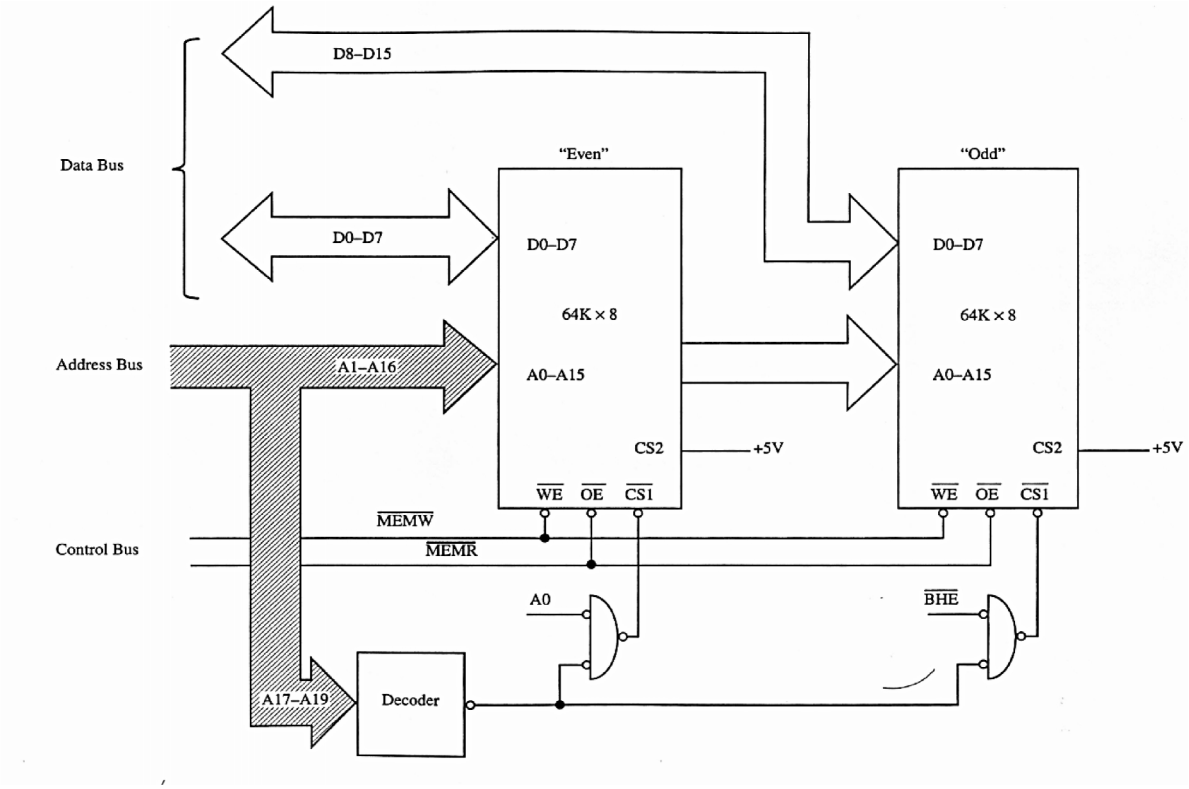


Figure IV.5 : interfaçage de la mémoire.

Remarque : Le microprocesseur 8086 fournit trois signaux de commande RD, WR et M / IO qui peuvent être décodés comme représentés sur la figure suivante : les signaux (MEMR, MEMW, IOR et IOW).

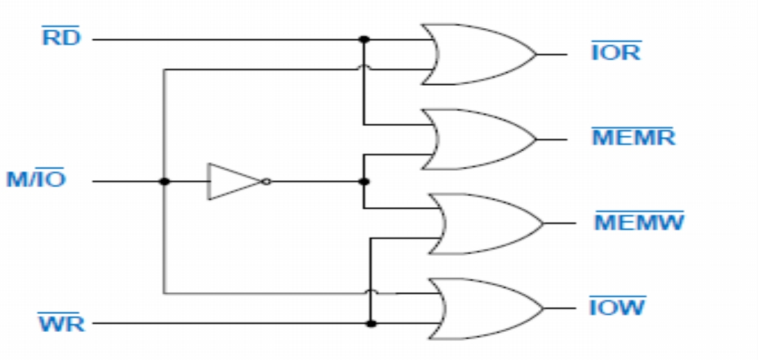


Figure IV.6 : décodage signaux lecture et écriture mémoire et I/O.

IV.5 E/S système (Switches and LEDs)

Les ports d'E/S sont utilisés pour l'échange de données entre le µp et un périphérique. Un port d'E/S est similaire à un emplacement mémoire sauf que chaque port doit avoir sa propre adresse.

Avec les processeurs 80x86, on a 16 lignes d'adresse pour le port d'E/S avec une taille 64Ko.

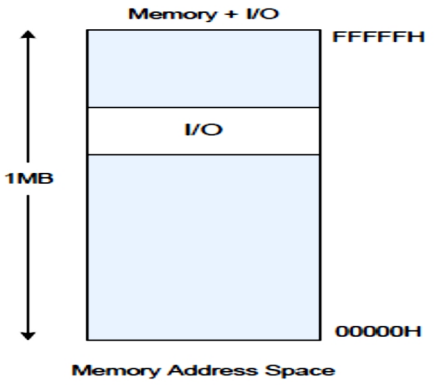
Il existe deux méthodes différentes d'interface d'E/S à microprocesseur 8086:

E / S isolé (adresse cartographique) ;

E / S Mappé en mémoire (adresse cartographique).

IV.5.1 Adresse cartographique

Les adresses des ports d'E/S appartiennent au même espace mémoire que les circuits mémoires (Les adresses des ports d'E/S appartiennent au même espace mémoire que les circuits mémoires (on dit que les E/S sont mappées en mémoire)



IV.5.2 Adressage indépendant

Le microprocesseur considère deux espaces distincts :

- L'espace d'adressage des mémoires
- L'espace d'adressage des ports d'E/S

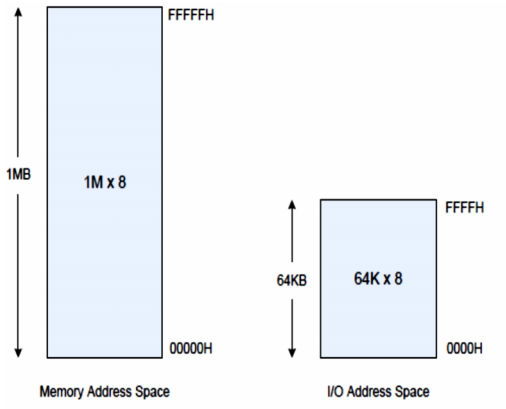


Tableau IV.1: Comparaison entre Adressage cartographique et indépendant

Adresse cartographique	Adressage indépendant
• l'espace d'adressage des mémoires diminue • l'adressage des ports d'E/S se fait avec une adresse de même longueur que pour les cases mémoires • toutes les instructions employées avec des cases mémoires peuvent être appliquées aux ports d'E/S. • Les mêmes instructions permettent de lire et écrire dans la mémoire et les ports d'E/S, tous les modes d'adressages étant valables pour les E/S	• contrairement à l'adressage cartographique, l'espace mémoire total adressable n'est pas diminué • l'adressage des ports d'E/S peut se faire avec une adresse plus courte que les circuits mémoires • les instructions utilisées pour l'accès à la mémoire ne sont plus utilisables pour l'accès aux ports d'E/S, ceux-ci disposent d'instructions spécifiques

Remarque : la mémoire et les ports I/O partage le même bus de données et d'adresse, c'est Le signal M/IO qui permet de différencier l'adressage de la mémoire de l'adressage des ports E/S :

• pour un accès à la mémoire, $M/\overline{IO} = 1$;

• pour un accès aux ports d'E/S, $M/\overline{IO} = 0$.

Les instructions utilisées pour R/W d'un port sont « IN / OUT ». Il existe deux types d'accès, accès direct et un autre indirect. Les registres utilisés sont l'accumulateur « AL / AX » et le registre de données « DX ». Les instructions directes d'E / S fournissent l'adresse de port (qui doit être un nombre compris entre 0 et FFh).

Si l'adresse du port d'E/S est sur un octet:

Lecture d'un port d'E/S	Ecriture d'un port d'E/S
IN AL, @: lecture d'un port sur 8 bits IN AX, @: lecture d'un port sur 16 bits	OUT @, AL: écriture d'un port sur 8 bits OUT @, AX: écriture d'un port sur 16 bits

Pour accéder à la plage complète des ports d'E/S de 0000H à FFFFH, les instructions indirectes d'E / S doivent être utilisées.

Pour ces instructions, le registre DX doit être préchargé avec l'adresse du port.

• si l'adresse du port d'E/S est sur deux octets:

Lecture d'un port d'E/S	Ecriture d'un port d'E/S
IN AL, DX : lecture d'un port sur 8 bits IN AX, DX : lecture d'un port sur 16 bits	OUT DX, AL : écriture d'un port sur 8 bits OUT DX, AX: écriture d'un port sur16 bits

Où le registre DX contient l'adresse du port d'E/S à : lire / écrire.

Chapitre 5 : Les interfaces d'entrées/sorties

V.1 Définition

Une interface d'entrées/sorties est un circuit permettant au Cpu de communiquer avec l'environnement extérieur (périphérique) : clavier, écran, imprimante, processus industriel etc...

Les interfaces d'E/S sont connectées au microprocesseur à travers les bus d'adresses de données et de commandes.

Un circuit d'E/S possède des registres pour gérer les échanges avec les périphériques :

- Registre de configuration
- Registre de données

A chaque registre est assignée une adresse : le microprocesseur accède à un port d'E/S en spécifiant l'adresse de l'un de ses registres.

Intel a développé plusieurs circuits intégré de contrôle de périphérique conçus pour soutenir la famille de processeurs 80x86 tels que:

Le 8255A Programmable Peripheral Interface (PPI),

Le 8259 Programmable Interrupt Controller (PIC),

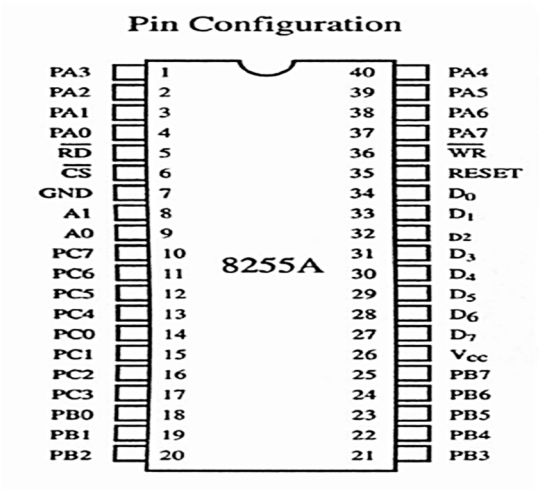
Le 8253/54 Programmable Interval Timer (PIT),

Le 8237 Programmable DMA Controller.

V.2 L'interface parallèle 8255

Le rôle d'une interface parallèle est de transférer des données du microprocesseur vers un périphérique et l'inverse en parallèle. Le 8255 est une interface parallèle programmable, elle peut être configurée en entrée et/ou en sortie par programme.

Le 8255 fournit 24 lignes d'E/S qui peuvent être organisées en trois ports d'E/S (A, B, et C) de 8 bits chacun. De plus le8255 peut fonctionner selon 3 modes: mode 0, mode1 ou mode2.



Pin Names

D_7–D_0	Data Bus (Bidirectional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0, A1	Port Address
PA7–PA0	Port A (bit)
PB7–PB0	Port B (bit)
PC7–PC0	Port C (bit)
V_{CC}	+5V
GND	0V

Figure V.1: configuration des broches du 8255

Tableau V.1: Sélection des ports de 8255

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
					Input operation (READ)
0	0	0	1	0	Port A → data bus
0	1	0	1	0	Port B → data bus
1	0	0	1	0	Port C → data bus
					Output operation (WRITE)
0	0	1	0	0	Data bus → port A
0	1	1	0	0	Data bus → port B
1	0	1	0	0	Data bus → port C
1	1	1	0	0	Data bus → control
					Disable function
X	X	X	X	1	Data bus tristate
1	1	0	1	0	Illegal condition
X	X	1	1	0	Data bus tristate

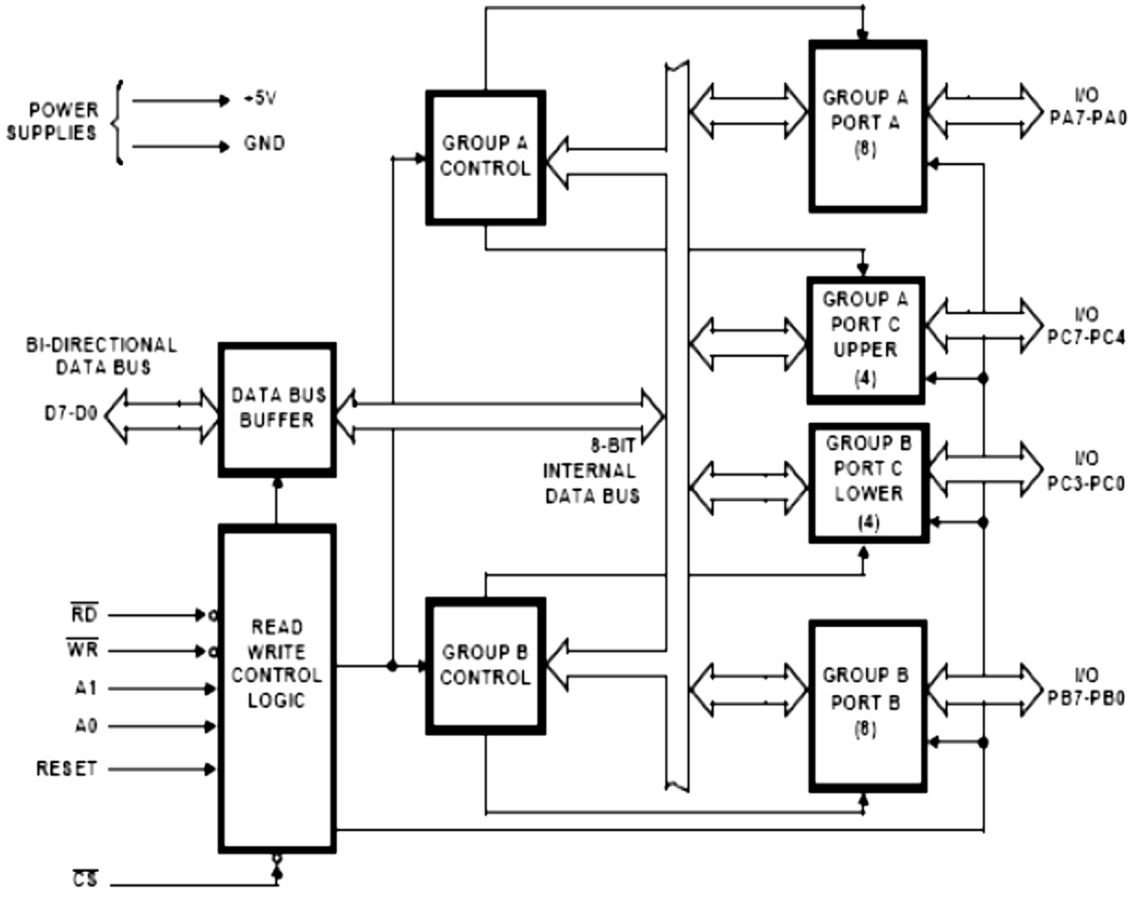


Figure V.2: Schéma fonctionnel du 8255

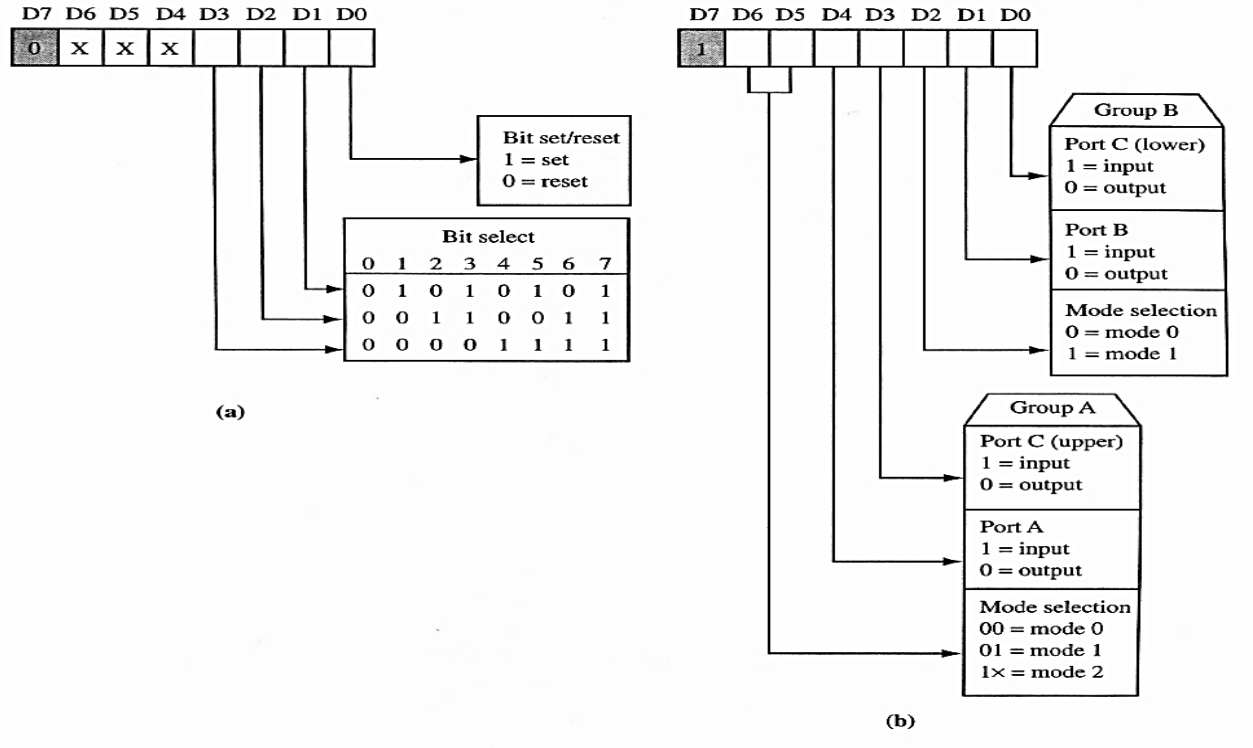


Figure V.3: Structure du registre de commande

La figure précédente montre le fonctionnement du PPI 8255 :
(a) Lorsque le bit 7 = 0, mise à « 1 » ou « 0 » d'une ligne du PORTC individuellement.
(b) Lorsque le bit 7 = 1, l'un des modes 0, 1, ou 2 peuvent être programmés.

Mode 0 (E/ S de base): Ports A et B fonctionnent comme entrées ou sorties. Le port C est divisé en deux groupes de 4 bits qui peuvent être configurés comme entrées ou sorties.

Mode 1 : Il est utilisé pour le dialogue avec des périphériques nécessitant un asservissement (contrôle).Ports A et B fonctionnent comme des entrées ou sorties comme en mode 0. Port C est utilisé pour le contrôle.

Mode 2: le port A est bidirectionnel (entrée et sortie).
Port C est utilisé comme signaux du contrôle. Port B n'est pas utilisé.

Remarque : Ces modes peuvent aussi être mélangés. Par exemple, le port A peut être programmé pour fonctionner en mode 2, tandis que le port B fonctionne en mode 0.

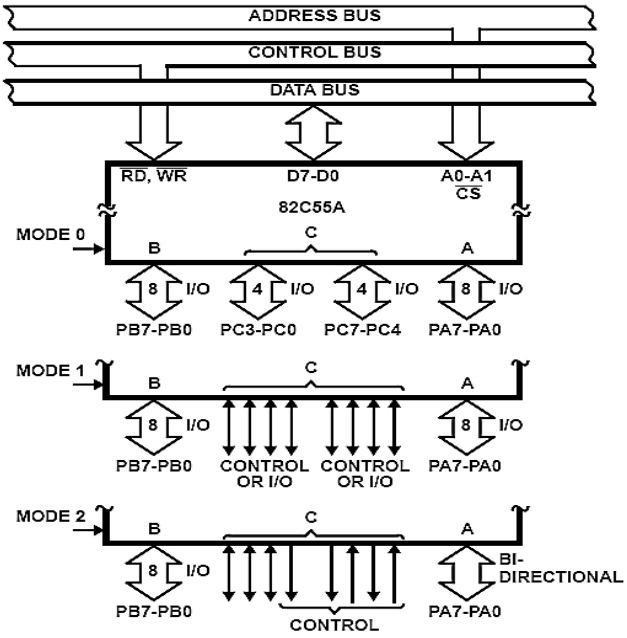


Figure V.4 : Les trois modes de base du 8255

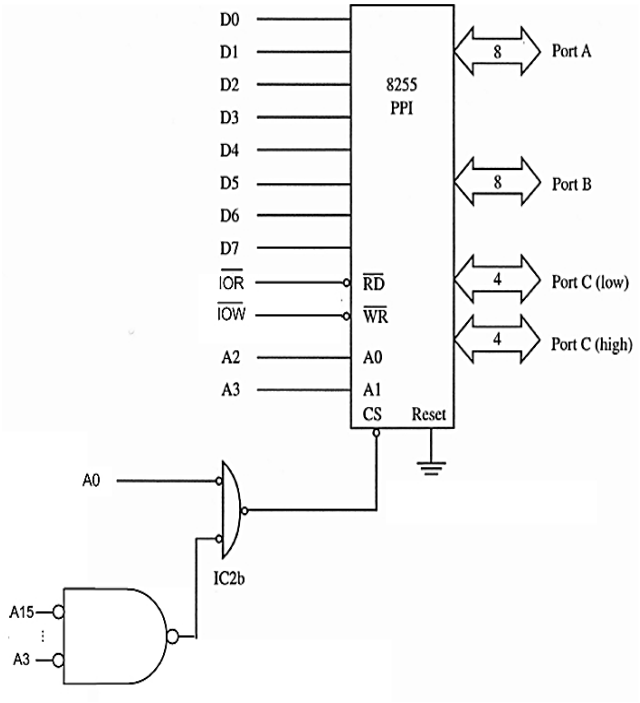
V.2.1 Interfaçage de 8086 avec le 8255

On peut connecter le bus de données du 8255 sur les lignes de données de poids faible du 8086 (D0 - D7) ou sur celles de poids fort (D8 - D15).

Donc l'un des deux signaux A0 ou BHE doit être utilise pour sélectionner le 8255 alors les adresses des registres du 8255 se trouvent à des adresses paires (validation par A0) ou impaires (validation par BHE).

Exemple :

A15-A12	A11-A8	A7-A4	A2	A1	A0
0000	0000	0000	0	0	0
0000	0000	0000	0	1	0
0000	0000	0000	1	0	0
0000	0000	0000	1	1	0
Chip Select (CS)			Sélection de port		adresses paires



V.3 Le 8279

Intel 8279 est un circuit intégré d'interface programmable clavier / l'écran :
Scans et encode jusqu'à un clavier de 64-clé.
Contrôle jusqu'à un affichage numérique de 16 chiffres.
La Section Clavier possède une mémoire tampon de 8 caractères FIFO intégrée.
L'affichage est commandé à partir d'une RAM interne 16x8 qui mémorise les informations d'affichage codées.

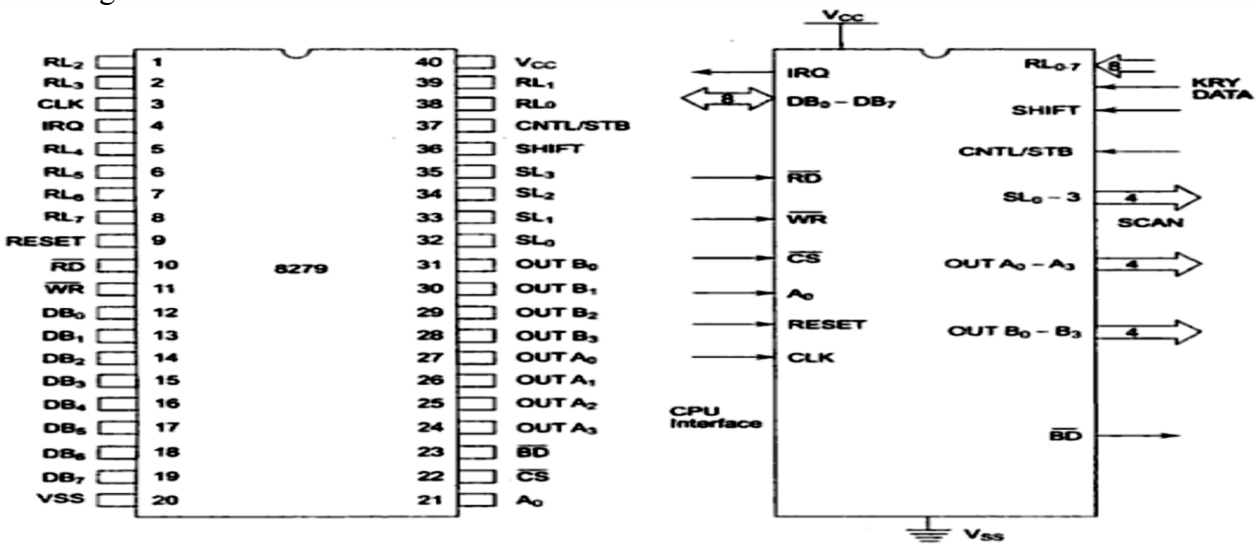


Figure V.5 : (a) Schéma fonctionnel et (b) brochage du 8279.

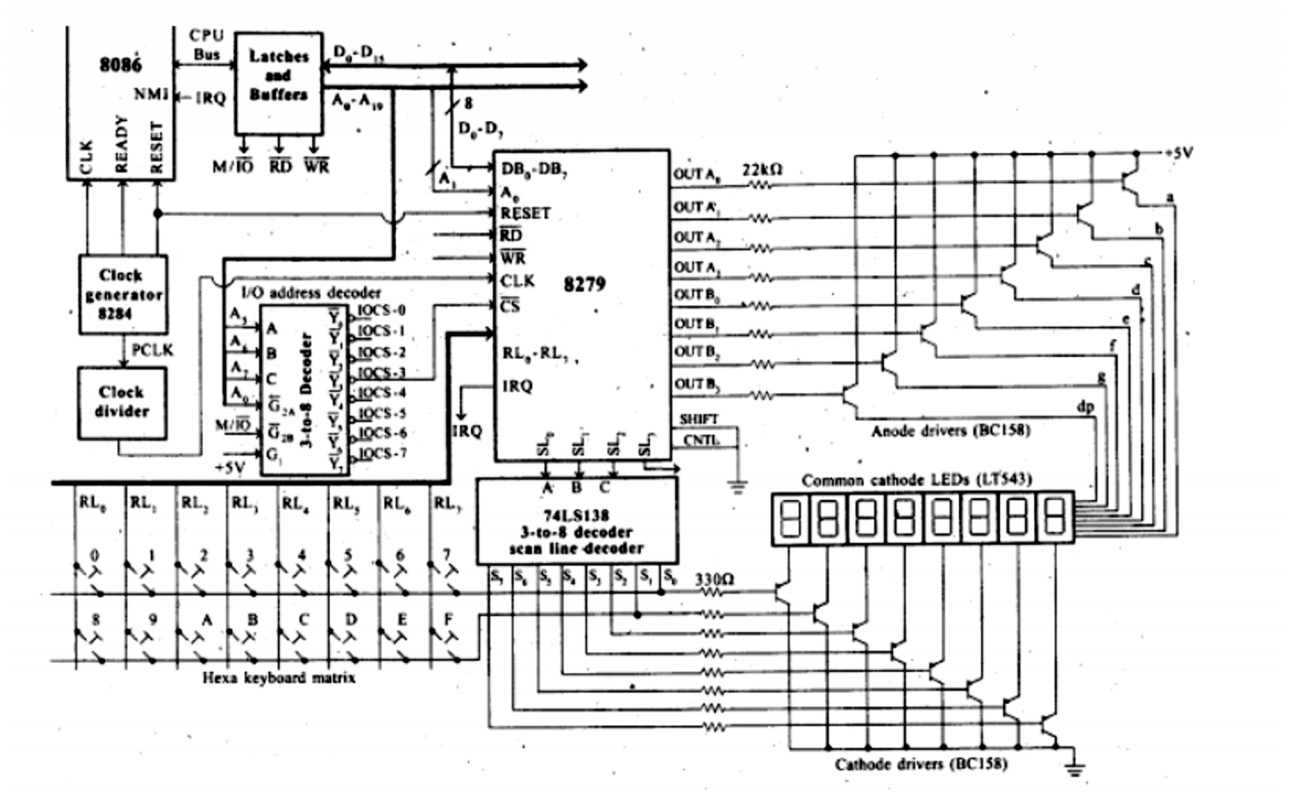


Figure V.6: Interface du 8279 (Un clavier Hexa et afficheur 7 segments) avec le 8086.

V.4 L'interface série 8250

L'interface entrées/sorties permet d´échanger des données entre le microprocesseur et un périphérique soit d'une manière parallèle (un mot de 8 bits) ou d'une maniéré série (bit par bit).

Mais sur des distances supérieures à quelques mètres, il est difficile de mettre en œuvre une transmission en parallèle. On utilise alors une liaison série.

Le circuit intégré 8250 est le composant chargé de la gestion des transmissions séries asynchrones il est appelé UART (Universal Asynchronous Receiver Transmitter).

Il existe deux types de transmissions séries:

• synchrone: les octets successifs sont transmis par blocs séparés par des octets de synchronisation.

• asynchrone : chaque octet peut être émis ou reçu sans durée déterminée entre un octet et le suivant ;

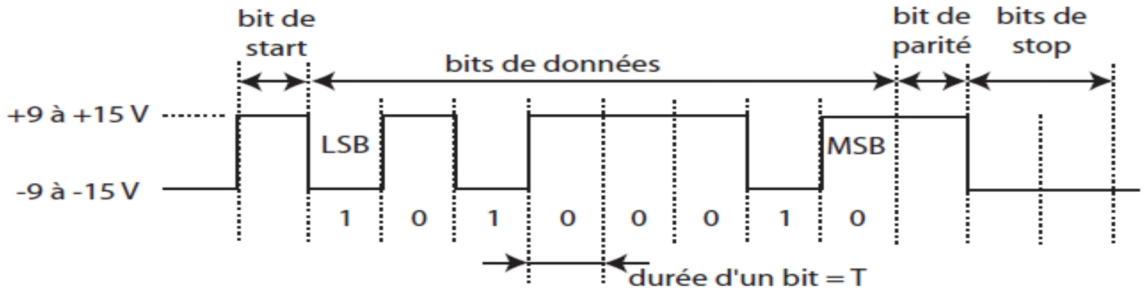


Figure V.7: Protocole de transmission de données asynchrone

• dans le protocole de transmission de données on a que deux états significatifs : 0 (ou low),

et 1 (ou high). Quand on ne transmet rien, la ligne est à l'état high (tension négative). Comme le montre la figure V.7,

- Le bit de Start permettra au récepteur de savoir que des données vont être transmises;
- Les bits de données sont transmis l'un après l'autre en commençant par le bit de poids faible. Ils peuvent être au nombre de 5, 6, 7 ou 8.
- Le bit de parité est un bit supplémentaire dont la valeur dépend du nombre de bits de données égaux à 1. Il est utilisé pour la détection d'erreurs de transmission ;
- Les bits de stop (1, 1.5 ou 2) marquent la fin de la transmission du caractère.

V.4.1 Structure de l'UART

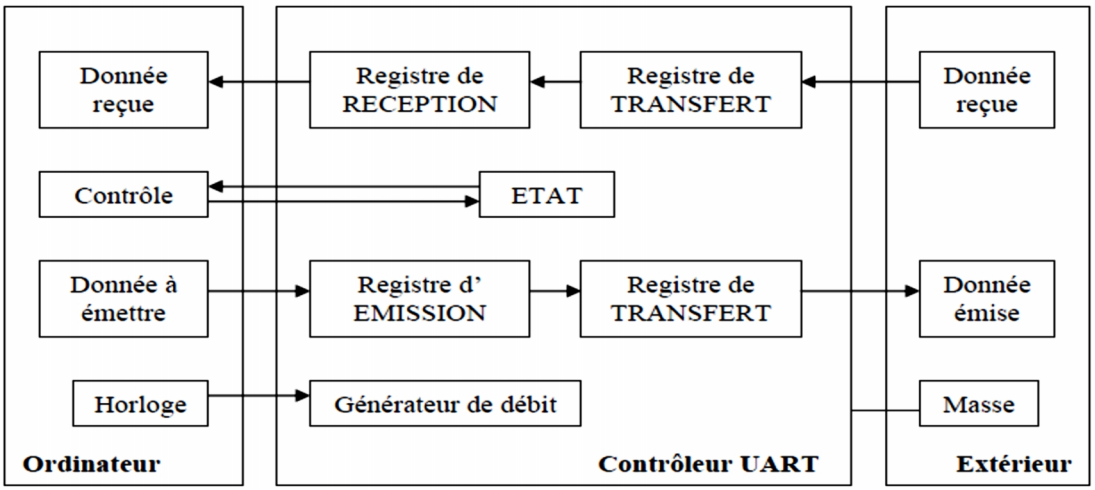


Figure V.8: Structure de l'UART

L'UART est conçue pour gérer les signaux en provenance d'un modem. Aussi est-il nécessaire de donner quelles broches sont utilisées par l'ETCD (Equipement Terminal de Communication de Données).

signal	Signification
TxD	Transmit Data
RxD	Receive Data
RTS	Request To Send
CTS	Clear To Send
DTR	Data Terminal Ready
DSR	Data Set Ready
DCD	Data Carrier Detect
RI	Ring Indicator

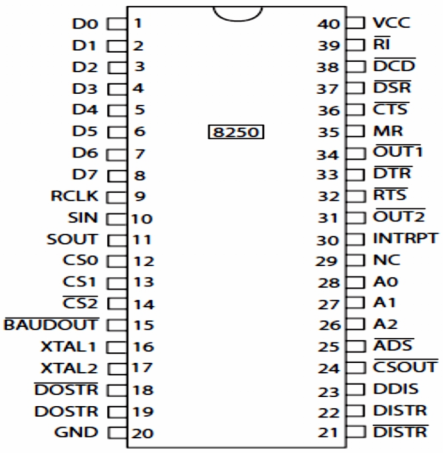


Figure V.9 : Brochage du 8250

Les 2 signaux TxD et RxD servent à transmettre les données. Les autres signaux sont des signaux de contrôle de l'échange de données.

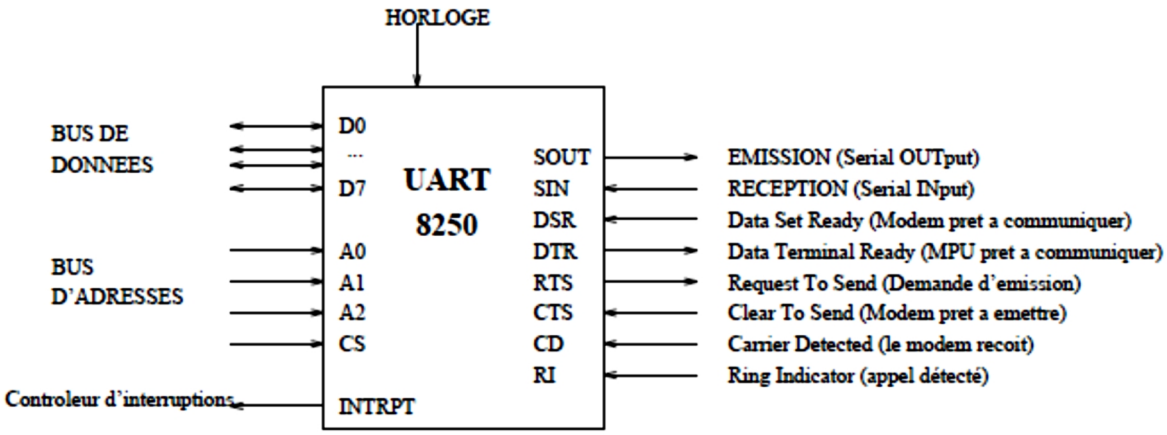


Figure V.10 : Bornes du circuit UART 8250.

V.4.2 Les registres du 8250

L'interface 8250 possède plus de 8 registres de 8 bits permettant de gérer la communication. Ces registres sont lus et modifiés par le processeur par le biais des instructions IN et OUT vues plus haut.

Comme on ne dispose que de 3 bits d'adresse :

(A0, A1 et A2), plusieurs registres doivent se partager la même adresse La solution est d'utiliser un bit d'un registre spécial, **DLAB.**

En fonction de l'état de **DLAB** (Divisor Latch Access Bit), on a accès soit au registre d'émission / réception, soit au diviseur d'horloge, soit au masque d'interruptions.

Adresse				REGISTRES
DLAB	A2	A1	A0	
0	0	0	0	RBR : Receiver Buffer (registre de réception)
0	0	0	0	THR : Transmitter Holding Register (registre d'émission)
1	0	0	0	DLL : Divisor Latch LSB (poids faible diviseur horloge)
1	0	0	1	DLM : Divisor Latch MSB (poids fort diviseur horloge)
0	0	0	1	IER : Interrupt Enable Register
x	0	1	0	IIR : Interrupt Identification Register
x	0	1	1	LCR : Line Control Register
x	1	0	0	MCR : Modem Control Register
x	1	0	1	LSR : Line Status Register
x	1	1	0	MSR : Modem Status Register
x	1	1	1	non utilisé

Dans ce cours on ne va pas étudier le fonctionnement du 8250 ni sa programmation car d'habitude elle n'est pas utilisée d'une façon individuelle mais elle fait partie du système de communication.

Chapitre 6 : Les interruptions

VI.1 Définition d'une interruption

Dans un système à microprocesseur pour dialoguer avec ces périphériques le microprocesseur a deux façons de communiquer avec ces derniers :

- **polling (scrutation périodique) :** En questionnant de façon continue le périphérique pour vérifier que des données peuvent être lues ou écrites. Cela permettra de limiter les tâches qui pourraient être accomplies par le micro-ordinateur.

- **interruption :** En l'interrompant lorsqu' un périphérique est prêt à lire ou écrire des données, avec une demande d'interruption (**IRQ :** Interrupt Request).

En utilisant cette technique, le processeur peut passer la plupart de son temps à d'autres tâches, et n'exécute une lecture des ports d'E/S que lorsqu' une donnée est disponible.

Le processeur 8086 reçoit les interruptions de trois sources différentes:

(1) **interruptions processeur :** c'est le processeur lui-même qui les génère, en raison d'un défaut interne (par exemple, une tentative de diviser par zéro)

(2) **L'interruption logicielle :** à l'aide de l'instruction **INT n** (couramment utilisée dans le PC pour accéder au BIOS et fonctions DOS),

(3) **interruptions matérielles :** produites par l'activation des lignes **INTR** et **NMI** du microprocesseur.

A la suite d'une demande d'interruption par un périphérique :

• le microprocesseur termine l'exécution de l'instruction en cours ;

• il range le contenu des principaux registres sur la pile de sauvegarde : CS : IP, flags, ...

• il émet un accusé de réception de demande d'interruption (Interrupt Acknowledge) indiquant au circuit d'E/S que la demande d'interruption est acceptée.

• il abandonne l'exécution du programme en cours et va exécuter un sous-programme de service de l'interruption (**ISR :** Interrupt Service Routine) ;

• il termine l'exécution de l'ISR avec l'instruction **IRET** (retour d'interruption).

• les registres sont récupérés à partir de la pile et le microprocesseur reprend l'exécution du programme qu'il avait abandonné.

Le processeur 8086 a seulement 2 broches interruption matérielle: INM ; INTR.

- **NMI (interruption non masquable) :** Elle ne peut pas être bloquée; le processeur doit répondre. Pour cette raison l'entrée NMI est habituellement réservée pour les fonctions critiques du système.
- **INTR (Interruption) :** Elle est masquable via le drapeau **IF**.

Pour (8086), l'adresse de la routine de service d'interruption est stockée dans 4 emplacements de mémoire consécutifs (un double-mot) dans une table de vecteur d'interruption commençant à l'adresse 00000h.

Remarque : **adresse vecteur d'interruption** = 4 × type de l'interruption

Exemple : interruption 10H, adresse du vecteur = 4 × 10H = 40H.

VI.2 Le contrôleur programmable d'interruptions 8259

Le microprocesseur 8086 ne dispose que de deux lignes de demandes d'interruptions matérielles (NMI et INTR). Pour pouvoir connecter plusieurs périphériques utilisant des interruptions, on peut utiliser le contrôleur programmable d'interruptions.

Un contrôleur d'interruption programmable (PIC) fonctionne comme un directeur général dans un environnement de système commandé par interruption.

Il accepte les demandes provenant de l'équipement périphérique, détermine laquelle des demandes entrantes est de la plus haute importance (priorité), vérifie si la demande entrante a une valeur de priorité plus élevée que le niveau en cours d'exécution et émet une interruption vers le processeur sur la base de cette détermination.

Exemple : le rôle de PIC 8259 :

- recevoir des demandes d'interruptions des périphériques ;
- résoudre les priorités des interruptions ;
- générer le signal INTR pour le 8086 ;
- émettre le numéro de l'interruption sur le bus de données.

Le 8259A peut gérer jusqu'à 8 demandes d'interruptions matérielles. Il est programmé par le logiciel du système comme un périphérique d'E / S.

Une sélection de modes de priorité est disponible pour le programmeur de sorte que la manière dont les demandes sont traitées par le 8259A peut être configurée pour correspondre à ses besoins du système.

Les modes de priorité peuvent être modifiés ou reconfigurés dynamiquement à tout moment pendant le programme principal. Cela signifie que la structure d'interruption complète peut être définie selon les besoins.

VI.3 Brochage du 8259 :

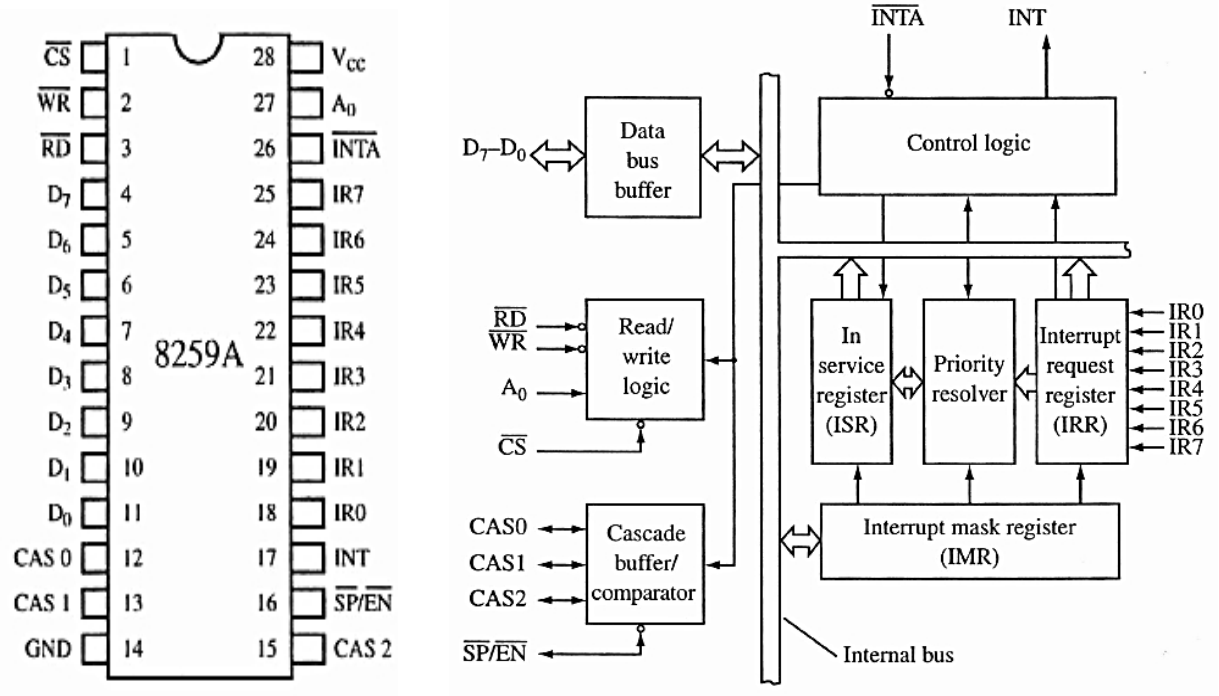


Figure VI.1 : Brochage et Schéema fonctionnel du 8259 .

Exemple : interfaçage de 8259 avec le 8086 :

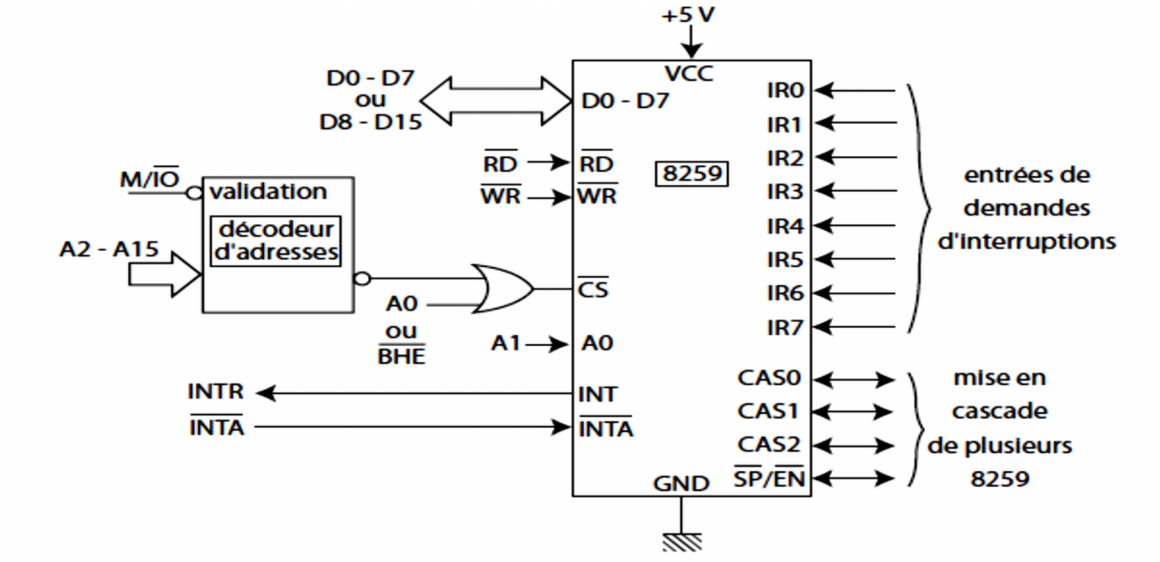


Figure VI.2 : interfaçage de 8259 avec le 8086 .

Remarque : si le nombre de demandes d'interruptions est supérieur à 8, on peut placer plusieurs 8259 en cascade.

VI.3 Le temporisateur programmable 8253/8254

On peut facilement exécuter une boucle de temporisation mais lorsque le microprocesseur exécute la temporisation il ne peut plus rien faire d'autre. Pour cela on fait appel à un circuit spécialisé c'est le PROGRAMMABLE TIMER.

Le 8253 programmable interval timer/counter est spécialement conçu pour les circuits d'Intel. Le programmeur configure le 8253 pour correspondre à ses besoins, initialise un des compteurs du 8253 avec la quantité désirée, puis avec le mot de commande le 8253 prendra en charge la temporisation et signalera la fin d'exécution avec une interruption du CPU.

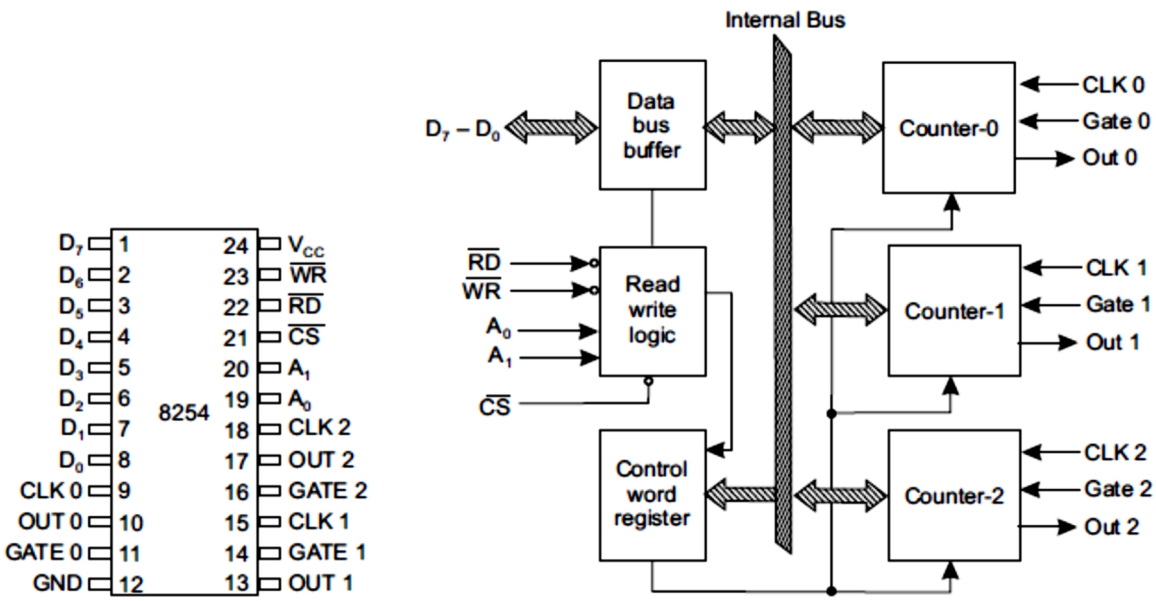


Figure VI.3 : Schéma fonctionnel et les descriptions des broches du 8254.

Il y a trois registres de comptage de 16 bits, dont chacun peut être programmé comme un timer ou un compteur d'événements. Plus un registre de commande qui peut ne peut-être qu'écrit.

L'accès aux registres du 8254 s'effectue de la façon suivante :

Tableau VI.1: Description des opérations de base

CS	RD	WR	A1	A0	fonction
0	1	0	0	0	Chargement compteur 0
0	1	0	0	1	Chargement compteur 1
0	1	0	1	0	Chargement compteur 2
0	1	0	1	1	Ecriture mot de mode
0	0	1	0	0	Lecture compteur 0
0	0	1	0	1	Lecture compteur 1
0	0	1	1	0	Lecture compteur 2
0	0	1	1	1	Etat de haute impédance
1	X	X	X	X	
0	1	1	X	X	

Table VI.2: Pin description of the 8254.

Pin Symbol	Name	Input/Output	Function
D_7 - D_0	Bidirectional Data Bus	Input/Output	Three-state 8-bit bidirectional data bus used when writing control words and count values, and reading count values upon reception of $\overline{WR}$ and $\overline{RD}$ signals from CPU.
$\overline{CS}$	Chip Select Input	Input	Data transfer with the CPU is enabled when this pin is at low level. When at high level, the data bus (D_0 thru D_7) is switched to high impedance state where neither writing nor reading can be executed. Internal registers, however, remain unchanged.
$\overline{RD}$	Read Input	Input	Data can be transferred from MSM82C53-2 to CPU when this pin is at low level.
$\overline{WR}$	Write Input	Input	Data can be transferred from CPU to MSM82C53-2 when this pin is at low level.
A_0 - A_1	Address Input	Input	One of the three internal counters or the control word register is selected by A_0/A_1 combination. These two pins are normally connected to the two lower order bits of the address bus.
CLK_{0-2}	Clock Input	Input	Supply of three clock signals to the three counters incorporated in MSM82C53-2.
$GATE_{0-2}$	Gate Input	Input	Control of starting, interruption, and restarting of counting in the three respective counters in accordance with the set control word contents.
OUT_{0-2}	Counter Output	Output	Output of counter output waveform in accordance with the set mode and count value.

La programmation du 8254 se fait par l'envoi d'un mot de commande dans le registre de commande CR. Les bits 6 et 7 de ce mot de commande permettent de l'affecter à l'un des trois compteurs. Le format du mot de commande est le suivant :

7	6	5	4	3	2	1	0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC - Selection du compteur

SC1	SC0	Description
0	0	Compteur 0
0	1	Compteur 1
1	0	Compteur 2

RW - Lecture/Ecriture

RW1	RW0	Description
0	0	Compteur Latch
0	1	D'abord LSB
1	0	D'abord MSB
1	1	D'abord LSB puis MSB

M - Mode

M2	M1	M0	Description
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

BCD - Format du compteur

0	Compteur binaire 16 bits
1	Compteur BCD 4 décimales

On remarque que chaque compteur peut être utilisé selon l'un des 6 modes possibles :

Mode 0 :

Il Permet au 8254 d'être utilisé comme un compteur ou temporisateur. Dans ce mode, la sortie est au niveau logique 0 Lorsque le décomptage atteint 0, la sortie passe au niveau 1. On notera que l'entrée **GATE** doit être un 1 logique pour permettre au compteur de compter. Si G devient 0 au milieu du comptage, le compteur s'arrête jusqu'à ce que G redevient 1.

Mode 1 :

Le 8254 à la fonction comme un monostable redéclenchable. Un front montant sur la gâchette GATE déclenche le décompte au top d'horloge qui suit. Alors la sortie passe à 0 et s'y maintient jusqu'à ce que le décompte arrive à 0. Chaque front montant sur GATE relance le processus à partir du compte initial (si celui- ci n'a pas été modifié).

Mode 2 :

C'est le mode diviseur par n il permet au 8254 de générer une série d'impulsions continues. La séparation entre les impulsions est déterminée par le comptage. Par exemple, pour un compte de 10, la sortie est un état logique 1 pendant neuf périodes d'horloge et 0 pour une période d'horloge. Ce cycle est répété jusqu'à ce que le compteur est programmé avec un nouveau comptage ou jusqu'à ce que l'entrée G est placée à un niveau logique 0. L'entrée G doit être un 1 logique pour ce mode pour générer une série continue d'impulsions.

Mode 3:

Ce mode est similaire au mode 2 sauf que OUT passe au niveau bas lorsque la moitié du compte initial est atteinte, soit N/2, et reste dans cet état jusqu'à ce que le compte arrive à 0 et le cycle recommence. Comme pour le mode 2, un niveau 1 sur GATE valide le décompte et un niveau 0 l'inhibe alors qu'un front montant le réinitialise. De ce fait, une valeur impaire amène (N+l)/2 avec sorties au niveau haut et (N-1)/2 au niveau bas.

Mode 4:

Ce mode est similaire au mode 0 sauf que OUT est au niveau haut pendant le décomptage et produit une seule impulsion négative lorsque le compte devient nul.

Mode 5:

Ce mode est similaire au mode 4. Cependant il est démarré par une impulsion de déclenchement sur la broche G au lieu par le programme. Ce mode est également similaire au mode 1, car il est redéclenchable.

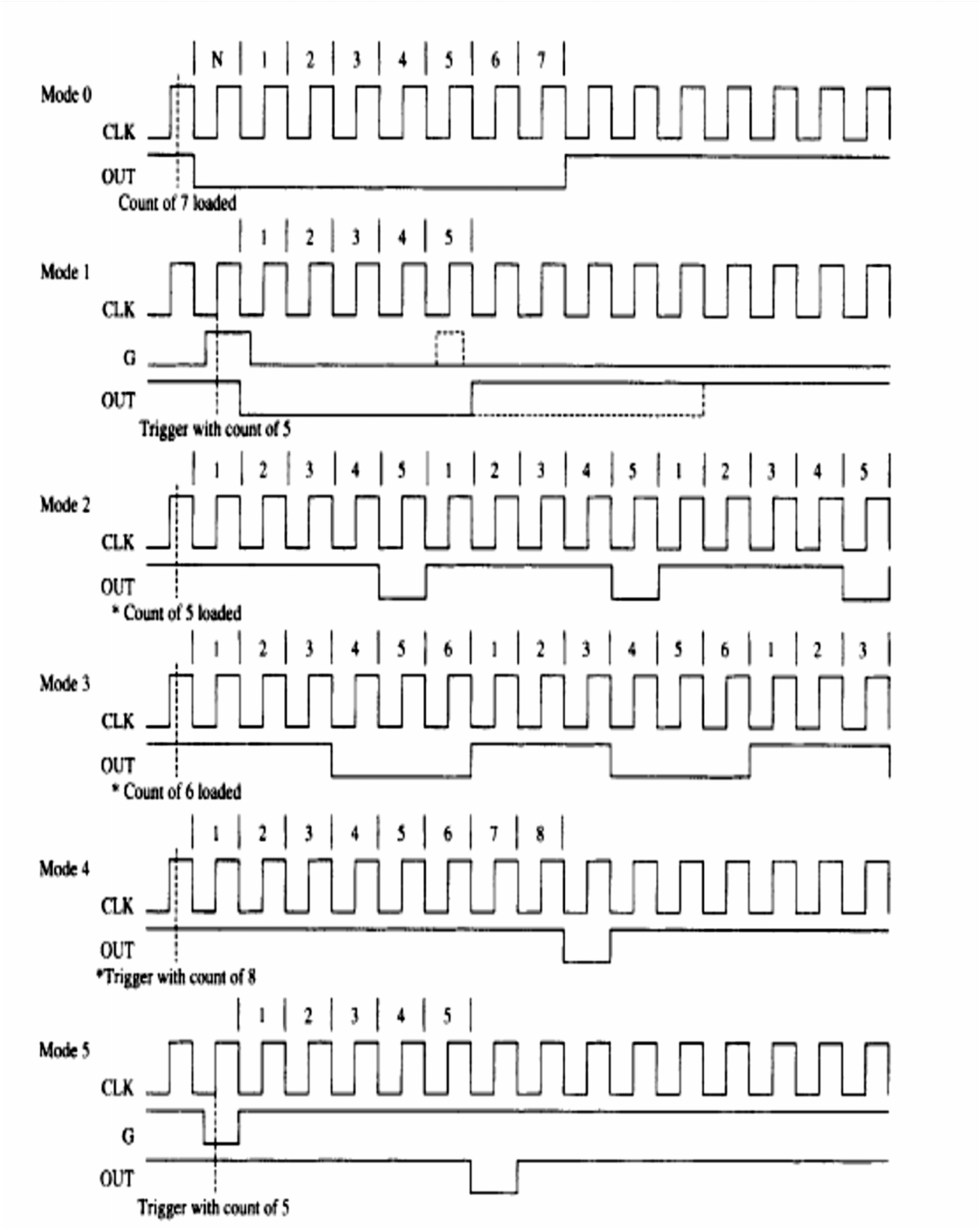


Figure VI.4 : Les six modes de fonctionnement pour le 8254. L'entrée de G = 0 arrête le compteur en mode 2, 3, et 4.

VI.4 DMA (Accès Direct à la Mémoire)

Précédemment, nous avons discuté d'E / S de base et de traitement d'interruption. Maintenant, nous nous tournons vers la forme finale d'E / S appelé accès direct à la mémoire (DMA). La technique DMA offre l'accès directe à la mémoire pendant que le microprocesseur est temporairement désactivé. Ceci permet aux données d'être transférées entre la mémoire et le dispositif d'E / S à une vitesse qui est limitée uniquement par la vitesse des composants de mémoire dans le système ou le dispositif de commande DMA. Les transferts DMA sont utilisés à de nombreuses fins, mais plus fréquents comme pour, transfert de données entre la mémoire et le disque dur, transfert de données vers la carte graphique ou la carte de son, etc.

On utilise pour cela le contrôleur de DMA un circuit intégré qui gère le transfert par DMA, lors de l'initialisation d'un transfert par DMA, le contrôleur de DMA négocie l'accès au bus de données avec le microprocesseur via les deux broches **HRQ** et **HLDA** du microprocesseur :

– Le cycle commence avec la requête du périphérique via une entrée **DREQ** (pour Dma REQuest) du DMAC.

– Le DMAC positionne alors l'entrée **HOLD** du 8086 à niveau haut, requérant ainsi que le microprocesseur entre dans un état HOLD qui laisse la main au DMAC.

– Le microprocesseur répond en terminant le cycle de bus en cours (s'il y en a un) et met ses adresses, données et la plupart des contrôles en position ouverte (haute impédance). La broche **HLDA** (pour **HOLD** Acknowledge) est positionnée au niveau haut par le microprocesseur pour accuser réception de la requête.

Dans un système avec des tampons de bus d'adresse, de données et de contrôle, **HDLA** est utilisé pour désactiver ces tampons de façon à ce que le microprocesseur soit complétement déconnecté de la mémoire et des entrées-sorties.

– Lorsqu'il reçoit **HDLA**, le DMAC applique **DACK** (pour Dma ACknowledge) au périphérique requérant le service. Le DMAC contrôle alors le système, fournissant les signaux de bus d'adresse et de contrôle comme si il était le microprocesseur (ce qu'il est réellement).

– Tant que le DMAC utilise les bus pour des transferts, le microprocesseur est inactif (et réciproquement, lorsque le microprocesseur est actif, le DMAC est inactif). Lorsque le DMAC a terminé son travail, il met **HOLD** à un niveau bas de façon à ce que le microprocesseur reprenne la main.

VI.5 Le 8237

Intel a conçu le DMAC 8237 pour être associé au microprocesseur 8080. Il est également utilisé pour les microprocesseurs 8085 et 8086/88. Le 8237 est en fait un microprocesseur à usage spécial dont le travail est le transfert de données à grande vitesse entre la mémoire et les E / S. la Figure suivante montre le brochage et le diagramme du contrôleur de DMA programmable 8237.

Le 8237 a quatre canaux pour transférer les données, c'est-à-dire qu'il peut être relié à quatre périphériques. Bien entendu, à un instant donné, un seul périphérique peut utiliser le DMAC pour transférer des données.

A chaque canal est associé deux signaux : **DREQ** et **DACK**. Il y a un seul signal HOLD et un seul signal HLDA, ce qui signifie que les quatre canaux utilisent les mêmes bus système, mais le DMAC décide quel périphérique doit prendre le contrôle à partir d'un registre des priorités qui peut être programmé.

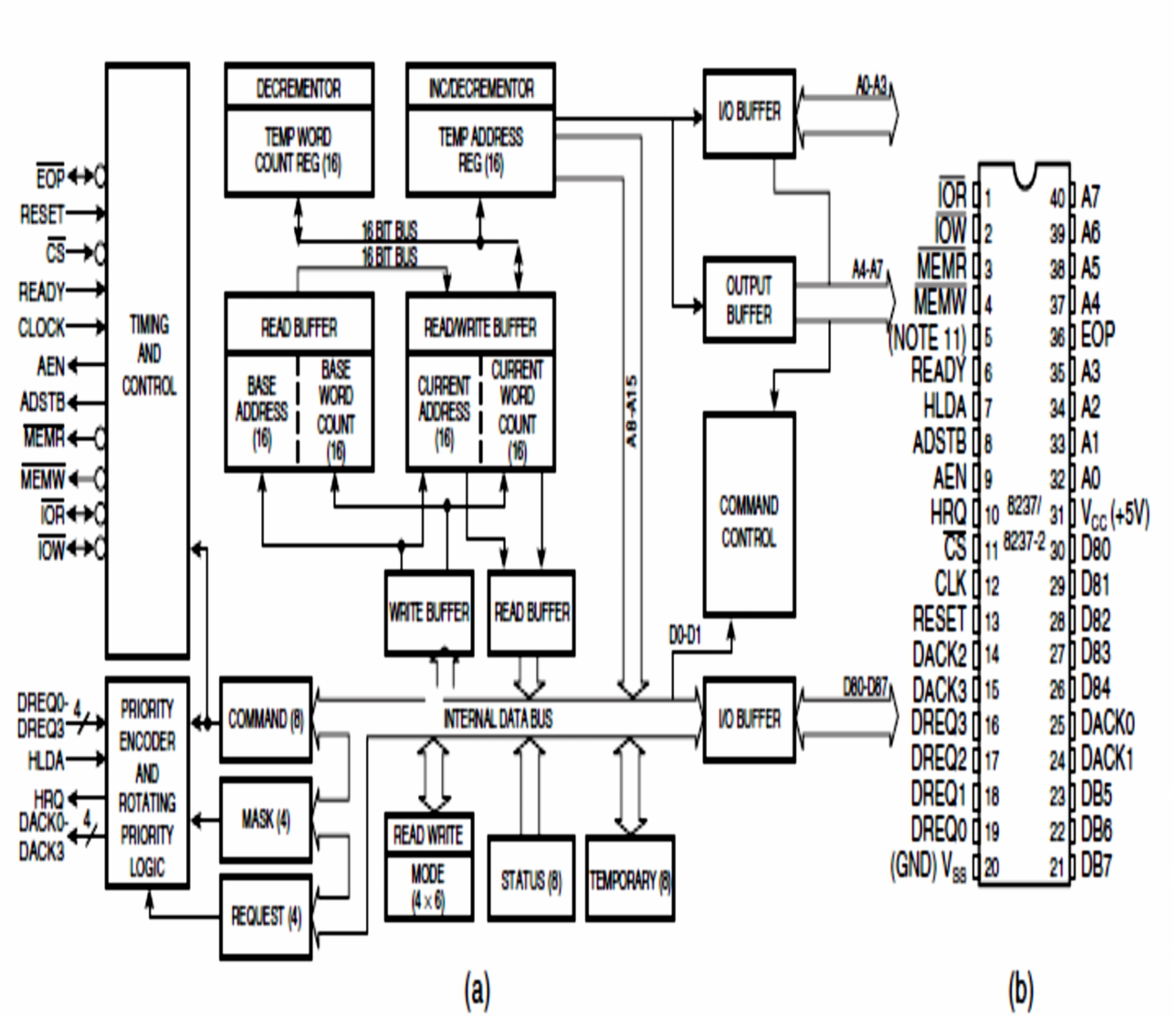


Figure VI.5 : (a) Schéma fonctionnel et (b) brochage du contrôleur DMA programmable 8237.

On ne va pas aller plus loin dans l'étude du contrôleur DMA programmable 8237 car il est rarement utilisé d'une façon individuelle.

Bibliographie

[1] Brey, Barry B. **The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing** ;8th ed; Pearson Prentice Hall ; 2009.

[2] D.a.godse A.p.godse. **Microprocessors and Interfacing**; first edition ; technical publications pune ; 2009 .

[3] M.aumiaux . **Les systèmes à microprocesseurs** ; 2eme edition ; masson ; 1982.

[4] **Intel The 8086 Family**; Users Manual ; 1979.

[5] Dr J.Y. Haggége. **MICROPROCESSEUR** ; (cours Institut Supérieur des études Technologiques de Radés) ; 2003

[6] A. Oumnad. **MICROPROCESSEURS DE LA FAMILLE 8086**.

[7] www.chipdocs.com

[8] www.datasheetcatalog.com