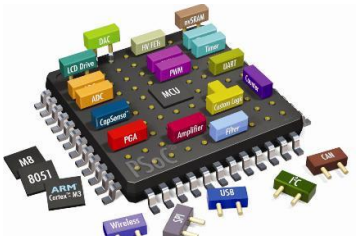
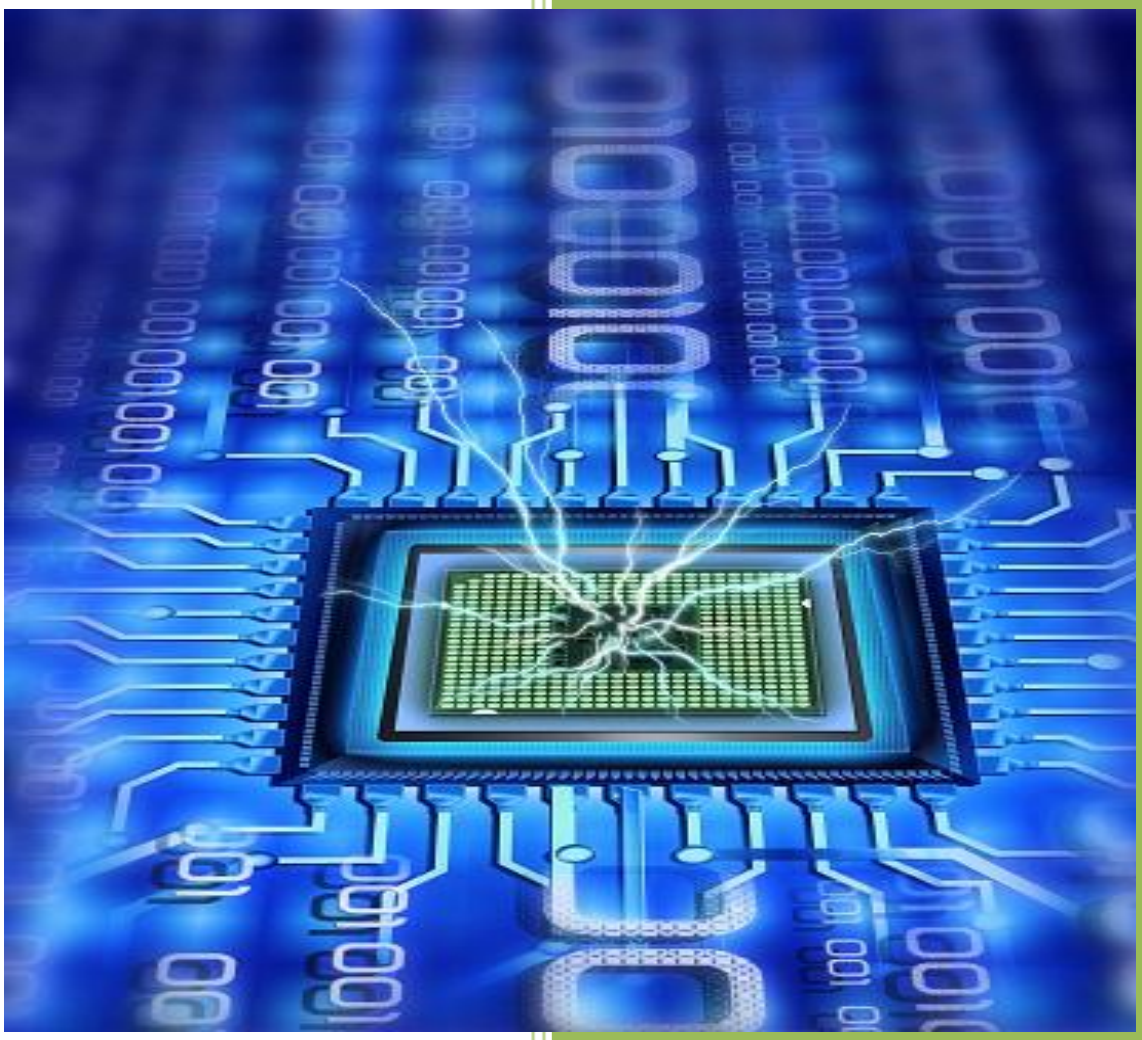
Ministères de l'enseignement supérieur
Institut Supérieur des Etudes Technologiques de Gabès

Année Universitaire 2013-2014



SUPPORT DE COURS MICROPROCESSEURS



Elaboré par :

TAYARI LASSAAD

TECHNOLOGUE A ISET GABES

E-mail :lassaad.tayari@isetn.rnu.tn

Chapitre 1

INTRODUCTION GENERALE AUX MICROPROCESSEURS

Objectifs:

① Rappeler à l'étudiant l'architecture de base d'un système informatique.

② Introduire le principe de fonctionnement d'un microprocesseur.

Pré requis:

① Informatique générale

② Architecture des ordinateurs

③ Les systèmes logiques & logique programmé

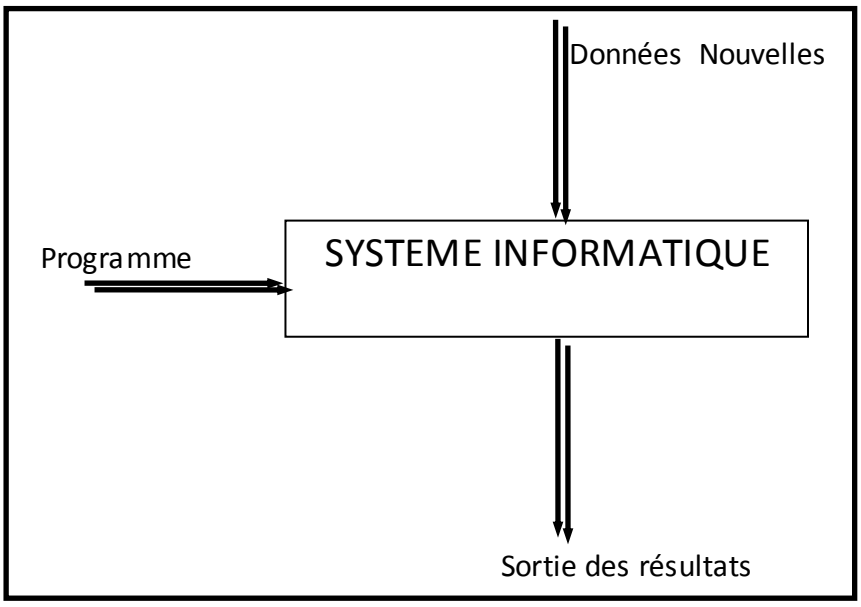
Plan

Chapitre 1

INTRODUCTION GENERALE AUX MICROPROCESSEURS

I/ INTRODUCTION AUX SYSTEMES INFORMATIQUE

Un système informatique sert à exécuter des programmes. Il procure des résultats ou prend des décisions en fonction des données introduites.

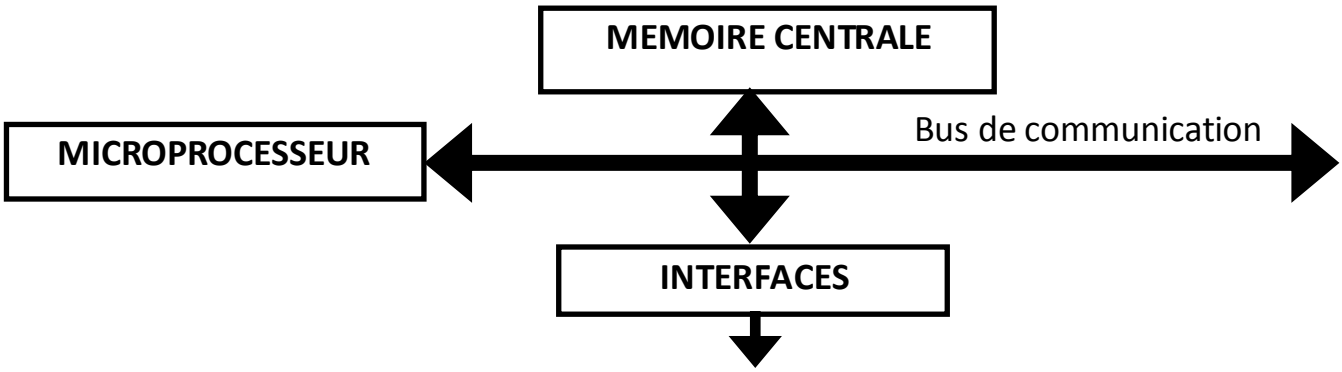


En fait, un système informatique se compose au moins d'une unité centrale et de plusieurs périphériques.

L'unité centrale, constituée dans sa configuration minimale, d'un microprocesseur, doit lire son programme, recevoir des données et fournir des résultats.

D'ou la nécessité de communiquer avec des périphériques via des circuits d'Entrée / Sorties (interfaces).

On a donc la synoptique suivante:



II/LES BUS

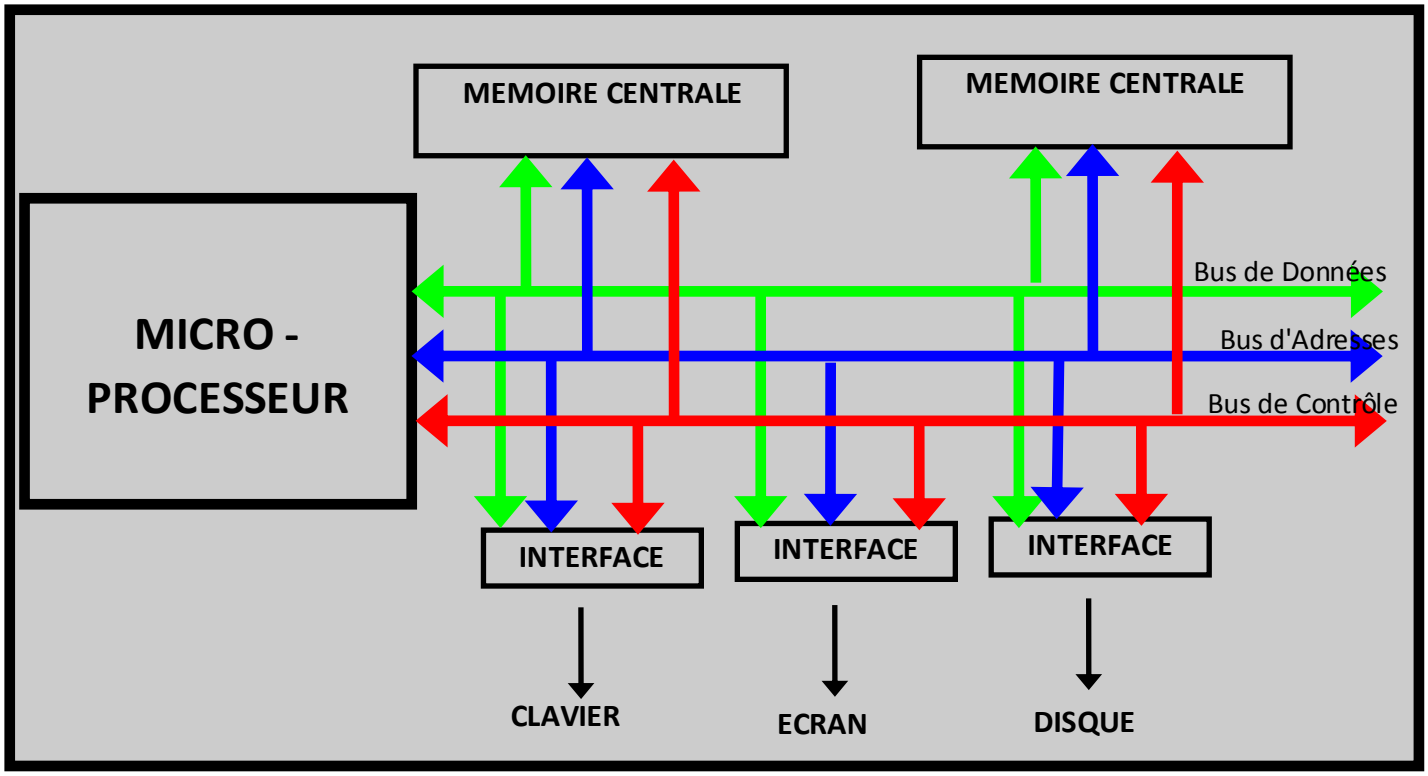
Le bus de communication est constitué éventuellement de trois bus, qui sont:

1. Le bus de données: permet le transport des données ou les instructions du programme
2. Le bus d'adresses: permet d'acheminer les adresses des cases mémoires ou des ports d'E/S.
3. Le bus de commande: permet de transmettre les ordres dans tout les sens.

Exemple: - Lecture de la case mémoire adressé par le bus d'adresse.

- Ecriture dans la case mémoire adressé par le bus d'adresse.

D'ou le synoptique suivant:

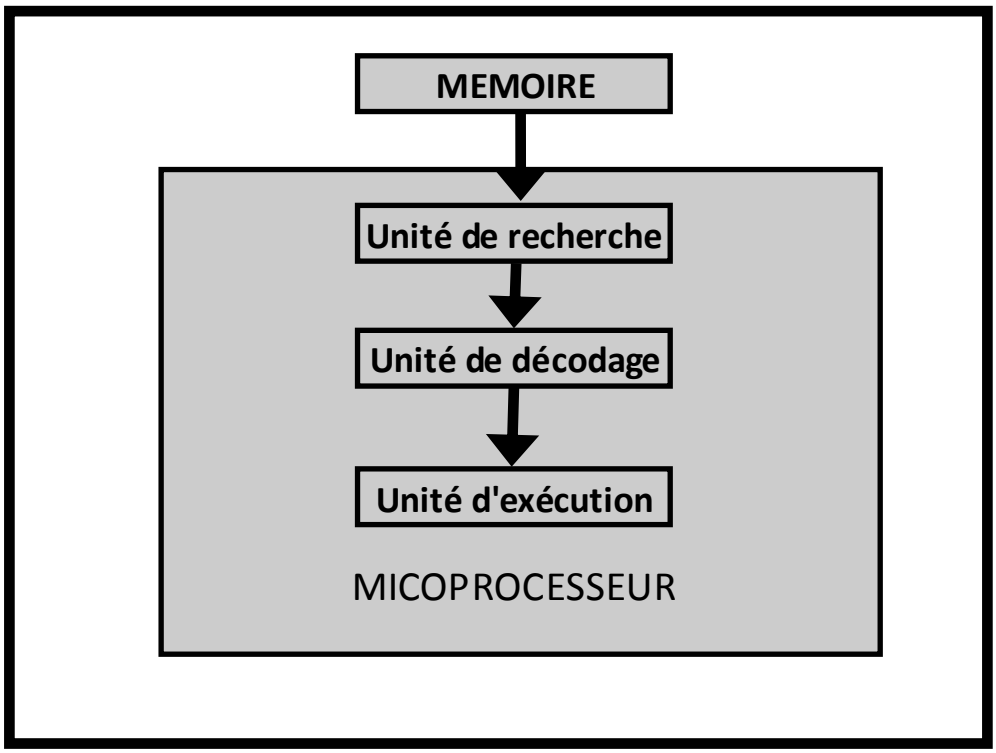


III/ PRINCIPE DE FONCTIONNEMENT DU MICROPROCESSEUR

Un programme est composé d'instructions rangés aux adresses croissantes de la mémoire.

Le microprocesseur doit accomplir les tâches suivantes:

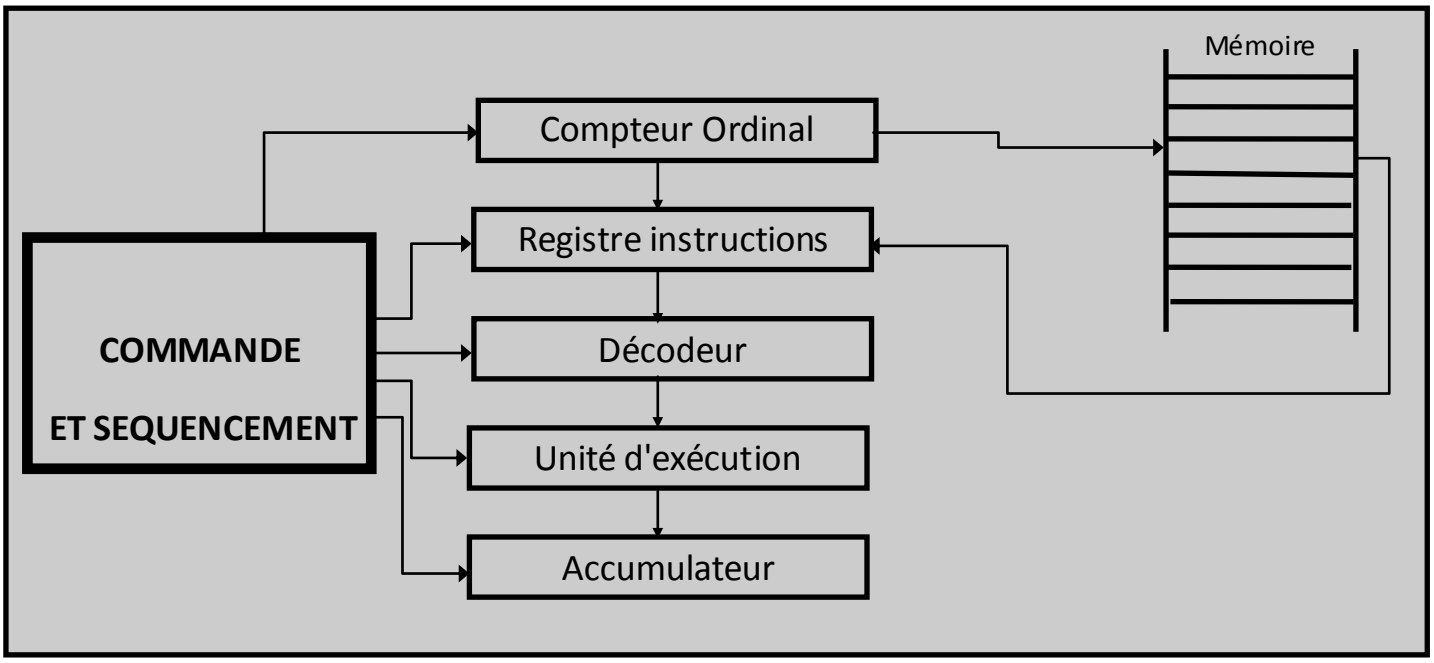
1. Il doit appeler une instruction qu'il lit en mémoire.
2. Il la décode, c'est à dire, qu'il traduit en commandes internes ce qu'elle lui dit de faire.
3. Il l'exécute.



Le fonctionnement du microprocesseur dans le cas le plus simple est le suivant:

1. Le microprocesseur recherche une instruction graçe a l'unité de recherche. Les instructions sont rangées dans les cellules mémoires d'adresses croissantes. Il doit donc savoir en permanence, quelle est l'adresse de la cellule à laquelle il doit accéder. Pour cela, il dispose d'un registre qui conserve le nombre correspondant à cette adresse. C'est le **compteur ordinal**.
2. Le code de l'instruction qui vient d'être lue dans la mémoire est ramené dans le microprocesseur. Il est donc décodé et le processeur sait maintenant ce qu'il doit faire.
3. Le microprocesseur exécute l'instruction.

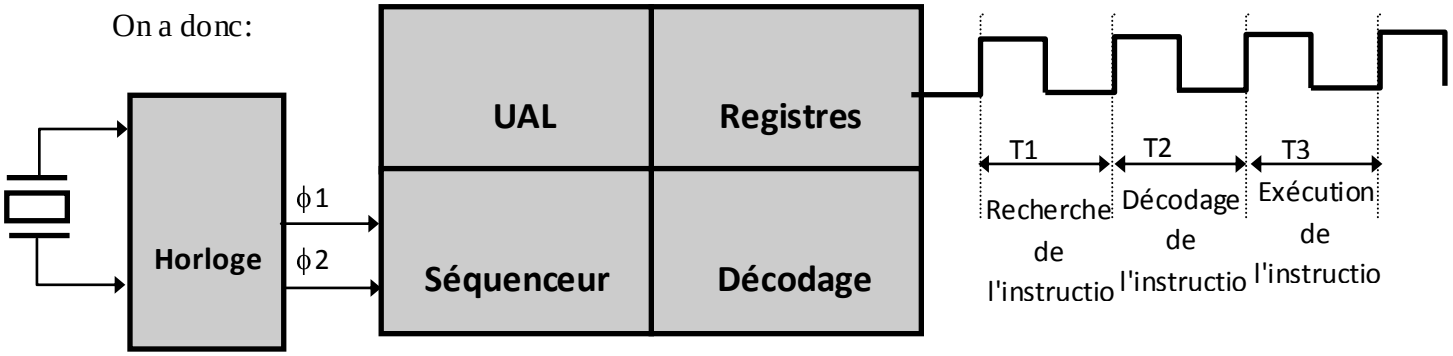
Le déroulement de l'exécution d'un programme est piloté par des circuits de commande de séquencement qui met les différentes unités en service à tour de rôle.



Unité arithmétique et logique **UAL**	**Registres** **- Accumulateur** **- Compteur ordinal** **- Registre instructions**
Circuits de commande et de séquencement	**Décodeur d'instructions**

Les circuits de commande de séquencement mettent chacune des sections du microprocesseur en service à tour de rôle. Ils sont eux mêmes pilotés par une horloge à quartz.

On a donc:



Les battements de l'horloge orchestrent l'exécution des instructions

1 période horloge = 1 micro cycle

1. Les registres

a) L'accumulateur: Utilisé généralement pour ranger l'un des opérandes devant être manipulé par l'ALU.

b) Le compteur de programme: C'est un registre qui contient l'adresse de l'instruction suivante du programme.

c) Registre d'instruction et décodeur: c'est un registre qui contient l'instruction suivante qui sera décodé pour qu'elle soit exécutée par l'unité d'exécution.

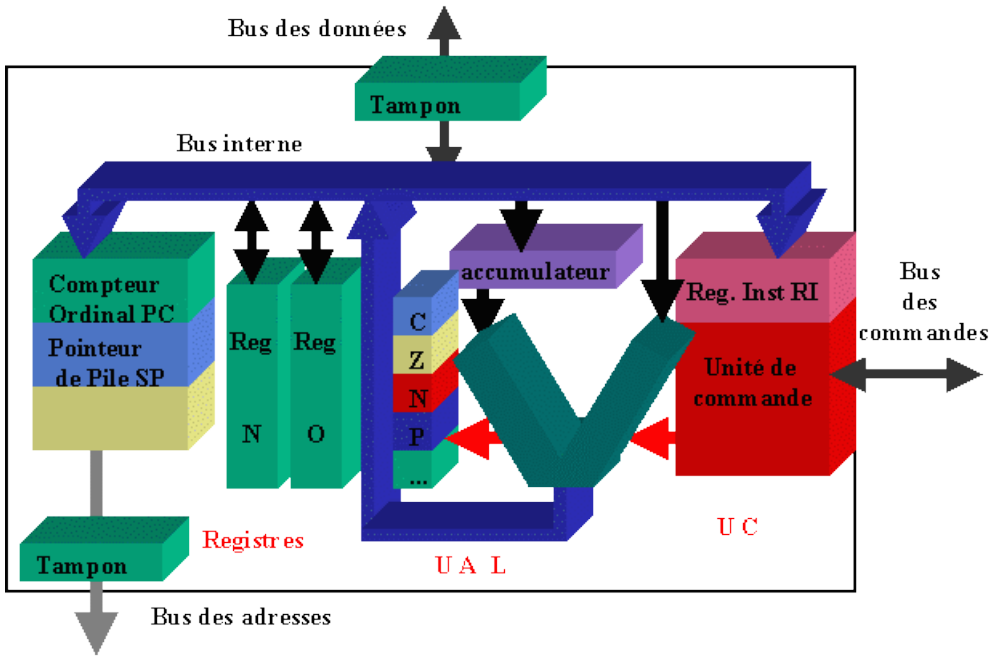
d) Registre d'adresse: Un CPU peut utiliser un registre ou une paire de registres pour ranger l'adresse d'un emplacement mémoire.

2. L'unité arithmétique et logique

C'est la partie du CPU qui assure les opérations de calcul arithmétique et logique (*,+,-,/,ET,OU,NON) et contient un registre flags qui indique le résultat de certaines conditions sous forme de drapeaux (FLAGS). Exemple: Carry,Zéro, Sign et Parity.

3. La circuiterie de contrôle

Cette partie du microprocesseur commande la séquence adéquate d'évènements requis pour toute tache de traitement. Donc c'est cette partie qui contrôle l'exécution de tout traitement (programme).



Chapitre 2

LES MICROPROCESSEURS 16 BITS

(le 8086/88 d'Intel)

Objectifs:

① Etudier l'architecture et le fonctionnement interne d'un processeur 16 bits (le 8086).

② Etudier le fonctionnement du 8086 avec l'extérieur.

Pré requis:

① Informatiques générale

② Architecture des ordinateurs

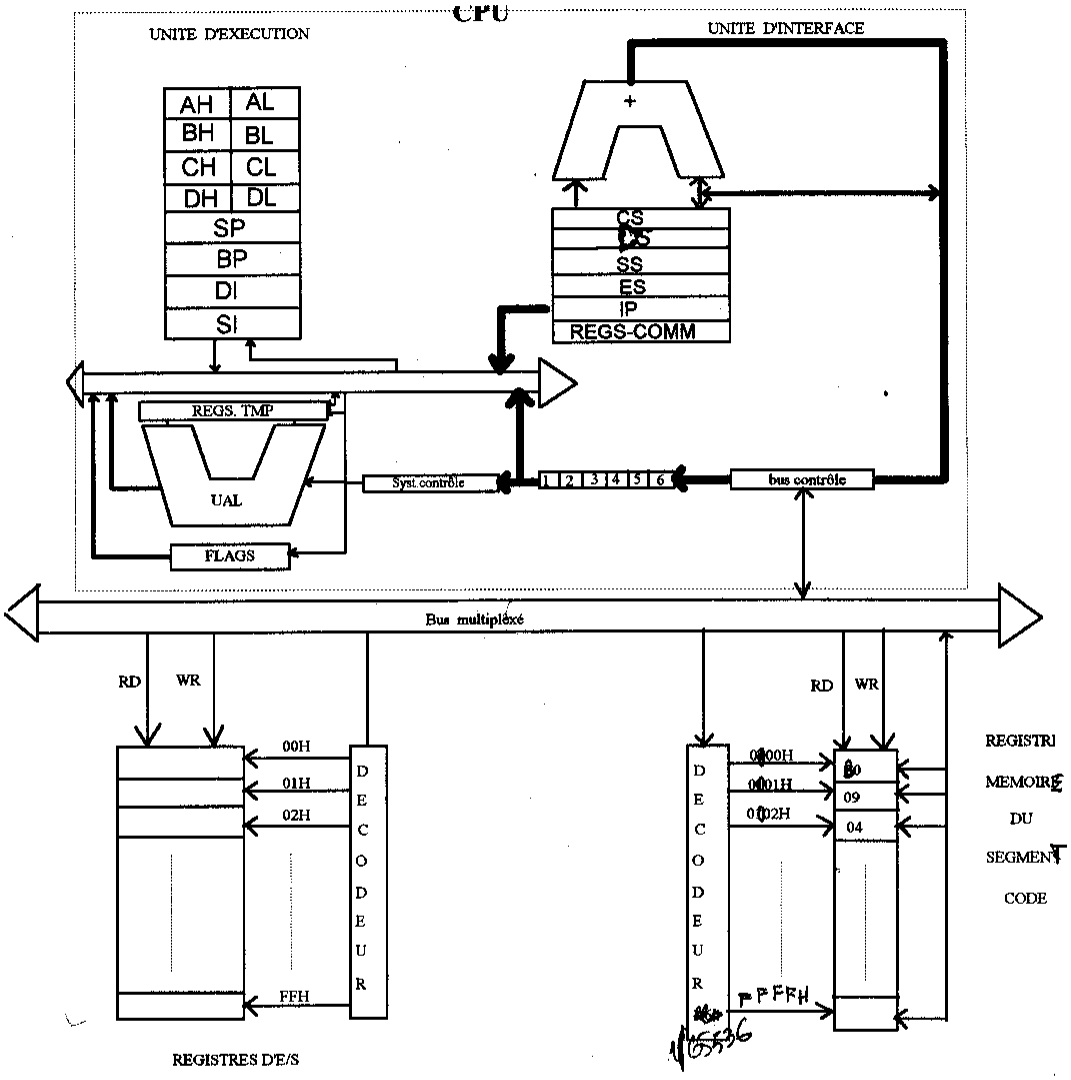
③ Les systèmes logiques

Plan

I/ ARCHITECTURE DU 8086

LES MICRO PROCESSEURS 16 BITS

I/ ARCHITECTURE ET FONCTIONNEMENT D'UN ORDINATEUR:



Exemple de programme:

MOV AL,9	B0	09		
ADD AL, [0009]	04	06	09	00
OUT 1, AL	E6	01		
HLT	F4			

Format = Action - Destination - Source

II/ ARCHITECTURE ET FONCTIONNEMENT

-1/ Architecture du 8086

Le microprocesseur exécute un programme en répétant les cycles suivants:

1. FETCH: Recherche de la prochaine instruction.
2. LECTURE: Lecture, si requise par l'instruction, d'un opérande.
3. EXECUTION: Exécution de l'instruction
4. ECRITURE: Ecriture, si requise par l'instruction, du résultat.

Il y a 2 unités essentielles au sein du CPU pour l'exécution séquentielle d'un programme:

1. *L'unité d'exécution (EU):*

   Exécute les instructions.

2. *L'unité d'interface avec le bus(BIU):*

- recherche les instructions,
- lit les opérandes,
- écrit les résultats.

BIU=unité d'interface avec le bus

Les 2 unités fonctionnent de façon indépendantes. Le recouvrement dans le temps entre l'exécution et la recherche fait disparaître le temps alloué a ce dernier.

Exemple:	T1	T2	T3	T4
Instruction 1	E1			
Instruction 2	D2	E2	E2	
Instruction 3	F3	Q3	E3	
Instruction 4		F4	F4	D4
Instruction 5				F5

F=Fetch, Q=Mise en Queue : effectuer par le BIU

D=Décodage, E=Exécution : Effectuer par l'EU

-2/ L'unité d'exécution(EU):

L'unité d' exécution comporte essentiellement:

- l' UAL(16 bits),
- Les registres généraux,
- Le registre FLAG.

L'UAL manipule les registres généraux et les opérandes d'instruction.

Selon le résultat le l'opération, le STATUS du CPU est maintenu par le positionnement des FLAGS.

-3/ L'unité d'interface avec le bus (BUS) :

Le BIU effectue toutes les opérations de bus sur l'EU. Les transferts de données entre le CPU d'une par et la mémoire ou les E /S d' autre par se font sur demande de L'EU.

-4/ Les registres généraux:

Pour le programmeur, le 8086/88 est perçu comme étant un ensemble de registres. On peu classer les registres généraux en 2 groupes

1. Groupe de données.
2. Groupe de pointeurs et index.

- **Groupe de données:**

AX	AH	AL	Accumulator
BX	BH	BL	Base
CX	CH	CL	COUNT
DX	DH	DL	Data

- **Groupe de pointeurs & Index:**

SP	Stack pointer
BP	Base pointer
SI	Source index
DI	destination index

Les registres données peuvent êtres considérés, soit comme 4 registres de 16 bits ou 8 registres de 8 bits.

AX: L'accumulateur : utilisé dans la multiplication et la division de mots. Ainsi qu'aux opérations d'E/S

BX: Le registre base: employé pour : l'adressage de données dans une zone mémoire séparée du code (offset).

CX:registre de compte: utilisé comme un compteur de répétition pour les boucles et pour les opérations de chaînes.

DX: Registre de donnée: utilisé dans les opérations de multiplication et de division de mots. Il peut également contenir le numéro de port d'E/S.

SP,BP,Si et DI: Représentent des adresses de référence.

-5 Les registres segments:

L'espace mémoire de 8086/88 est divisé en segments logiques allant jusqu'à 64 KO chacun

Le CPU à un accès direct et simultané aux 4 segments. Leurs adresses de base se trouvent dans les registres segments

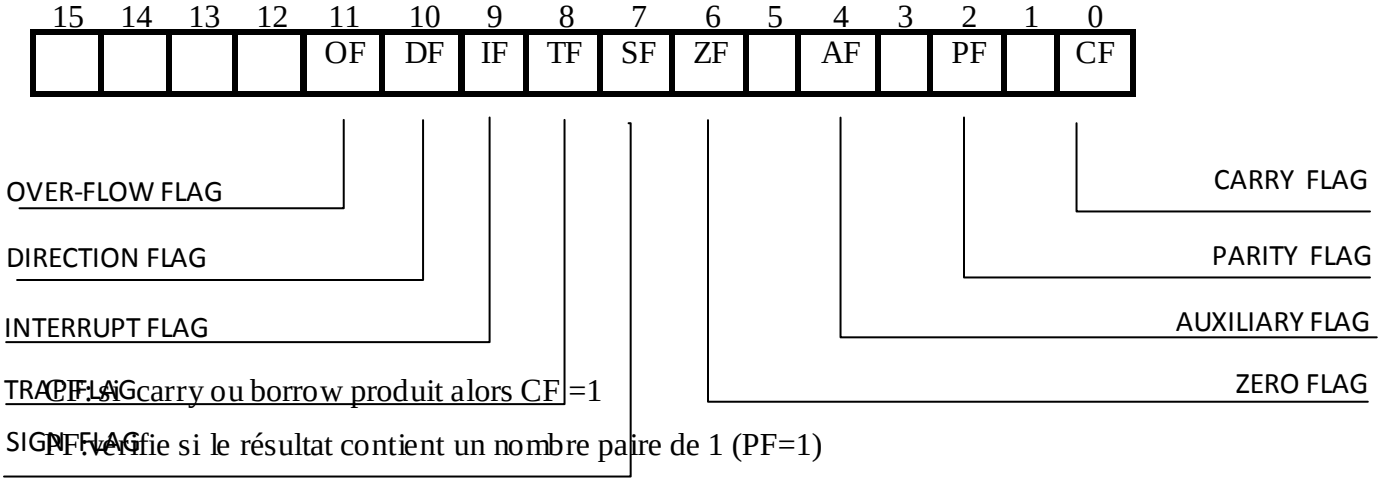
CS	Code segment
DS	Data segment
SS	Stack segment
ES	Extra segment

CS: Code segment : pointe au segment contenant le programme en cours exécution..

DS: Data segment : pointe au segment donnée contenant les variables en cours.

ES: Extra segment : pointe a un 2ème segment de donnée, utilisé pour les opérations de chaînes.

II-6 / Le registre FLAG: (16 bits):



CF: si carry ou borrow produit alors CF =1

PF: vérifie si le résultat contient un nombre paire de 1 (PF=1)

AF: ≅ CF mais par quarté

ZF: si le résultat d'une opération est 0, ZF=1

SF: si le résultat est négatif, SF=1 sinon 0

TF: exécution pas à pas

IF: permet la reconnaissance des demandes d'interruption

DF: Décrémentation ou incrémentation automatique des adresses de base (Traitement des chaînes de caractères).

OF: dépassement

III/ LE FONCTIONNEMENT INTERNE DU 8086/88

1- Le fonctionnement en mode d'interruption

Deux types d'interruptions:

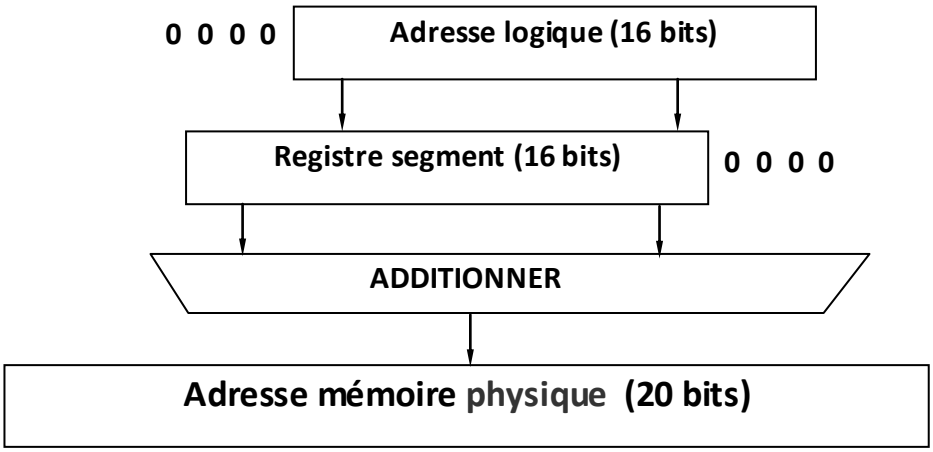
✓ INTERNE: Liée à l'exécution de l'instruction. Ce type d'interruption est non masquable, il est appelé "DEROUTEMENT" NMI.

✓EXTERNE:Provoqué par une cause externe INTR

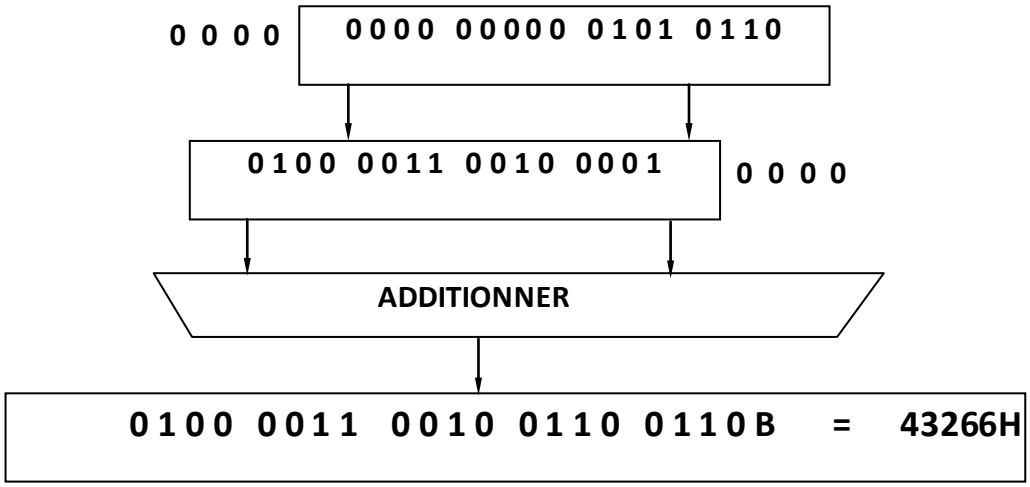
2-Le fonctionnement avec la mémoire:

a- Segmentation et adressage physique:

Pour le 8086/88 le bits couvrant 1MO(de 00000H à FFFFFH)



Exemple si offset =0056h et base segment=4321h on aura



Pour le 8086 on a toujours:

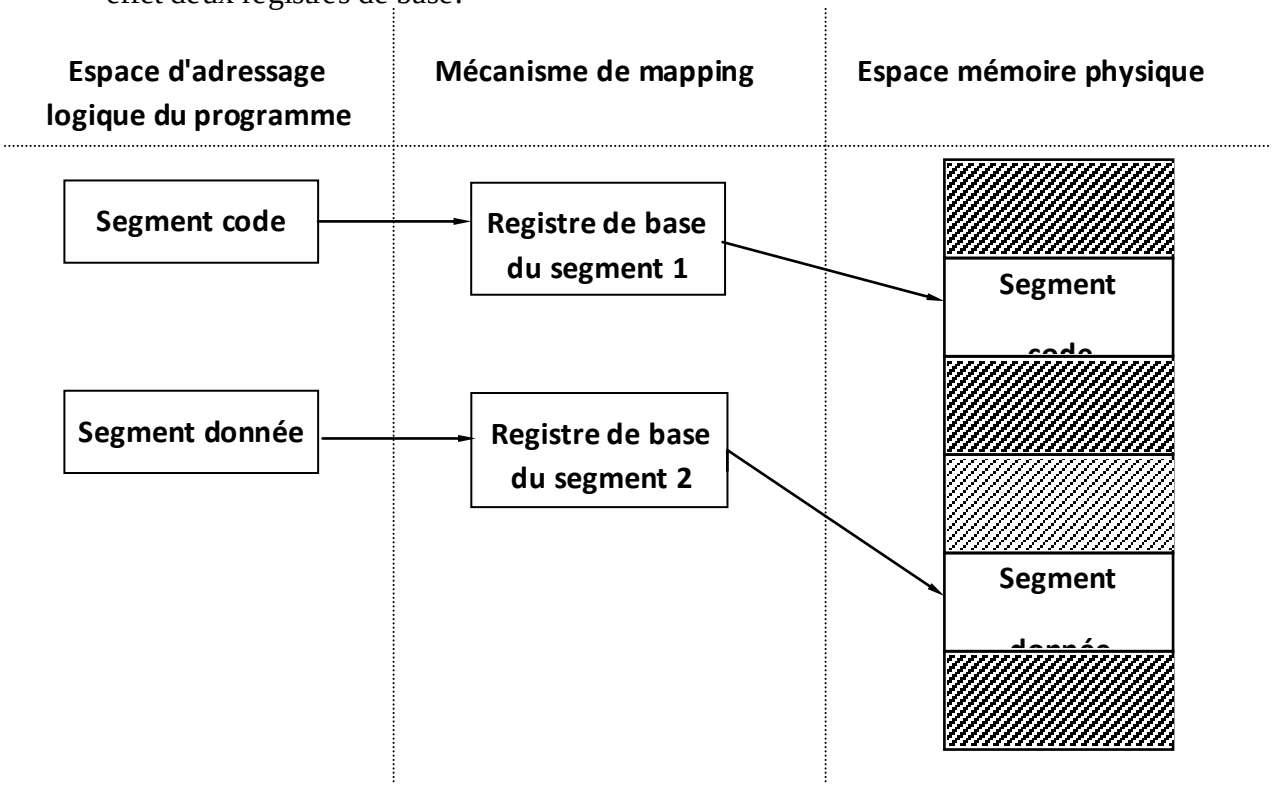
IP+CS et DI+ES } pour les instructions string

SP+SS BP+SS }

les autres +DS explicites

b- Le mapping

Lorsqu'il n'y a pas assez de blocs contiguës pour le programme entier, on pourra subdiviser celui-ci. Par exemple en segment code et en segment données. On utilise pour cet effet deux registres de base:



3-Le Fonctionnement avec les E/S

a- L'Espace d'E/S

64K ports de 8 bits ou 32 K port de 16 bits.

L'accès au ports se fait a l'aides des instructions IN et OUT.

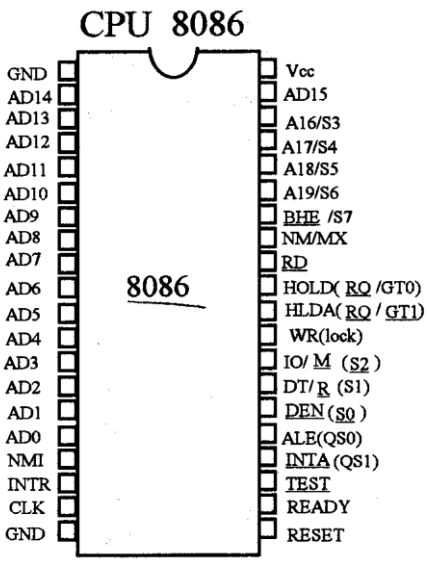
Format :

IN PORT, Données

OUT Données, PORT

IV - FONCTIONNEMNT DE 8086 /88 AVEC L'EXTERIEUR :

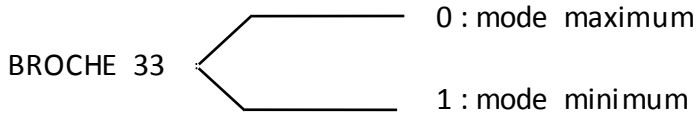
-1 Les pins du 8086 /88



Le 8086 est un processeur 16 bits. Son bus d'adresse est sur 20 bits.

Le µ processeur est présenté en un boîtier de 40 broches, alimenté par une sourse unique de 5V.

Deux modes de fonctionnement sont possibles, sélectionnés par la broche 33.



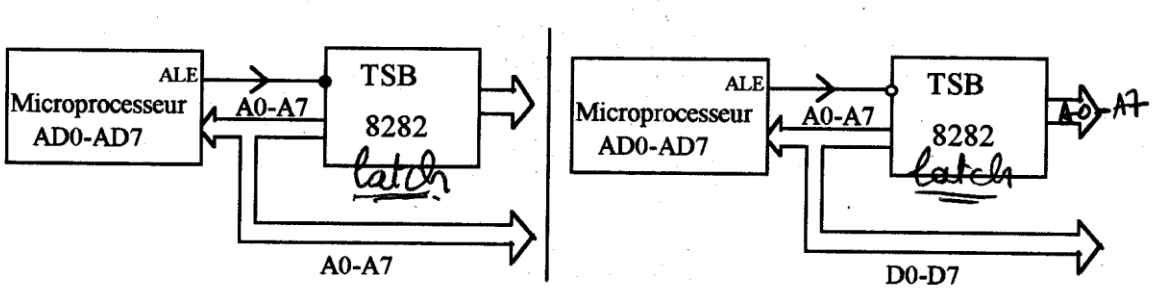
- **Mode** minimum : Ce mode permet de diminuer le nombre de circuits extérieurs pour les systèmes à processeur unique. Le processeur prend en charge la gestion des bus.
- **Mode** maximum : Ce mode étends l'architecture au configuration a multiprocesseurs et autorise ainsi l'extension du jeu d'instruction du coprocesseur de calcul. Un contrôleur de bus est donc nécessaire.
- Les pins du 8086 se divisent par leur fonctions en plusieurs catégories: Adresse, données, CONTOLE/STATUS et alimentation/ masse.

a- adresse et donnée:

AD0-AD15 : multiplexés dans le temps pour servir à véhiculer l'adresse puis la donnée.

AD0-AD15 :Ces lignes sont soit en entrée soit en sortie soit en 3éme état (haute impédance).

* A8-A15: octet d'adresse suivante non multiplexée pour le 8088.



AD8-AD15: Les bus d'adresse de poids fort activés en sortie d'adresse .

A16 -A19: (S3-S6):Ces 4 broches sont multiplexés. Durant la sortie de l'adresse, les 4bits de poids forts sont fournis pour les opérations d'écriture ou de lecture de type "mémoire". Lorsque le bus de donnée est actif , les sorties S3 et S4 donnent le nom du registre segment qui a servit à générer l'adresse physique .

S4	S3	Registre
0	0	ES
0	1	SS
1	0	CS (ou aucun)
1	1	DS

la ligne S5 donne l' état du FLAG d'autorisation d'interruption

le S6 est toujours zéro tant que le µ processeur est sur le but.

b- Horloge: c'est le générateur 8284qui délivre a l'entrée CLK un signal d'horloge

c- : ligne de contrôle est de status :

- RD indique que le µ processeur est en train de lire a partir de la mémoire ou les E/S.
- WR indique que le µ processeur est en train d'écrire dans la mémoire ou les E/S.
- ALE ↓: (front descendant) verrouillage du latche 8282 (jamais en 3ème états).
- IO/M :(mémoire ou E/S).
- RESET : (Redémarre le système).
- MN/MX: (mode MIN ou MAX).
- DT/R :(Data transmit/Resive). Pour commander le 8286 et le 8287 (Transeiver).
- DEN :(Data enable) valide la sortie 3 états dit 8286/8287 (contrôleur de bus).
- INTR: (Demande d'interruption) masquable.
- NMI : (interruption non masquable) interne.
- INTA : Acquittement d'interruption.
- READY : pour intercaler des états d'attentes (t-wait).

-2 Les circuits d'interface d'un microprocesseur

Les circuits de périphériques qui collaborent avec un microprocesseur sont:

A/ Générateur d'horloge 8284

B/ Le circuit d'E/S parallèle 8255

C/ Le circuit d'E/S série USART 8251A et UART 8250

D/ Le gestionnaire d'interruption 8259A

E/ Le contrôleur DMA 8257

F/ L'intervalle de temps programmable 8253 (timer)

Chapitre 3

PROGRAMMATION ASSEMBLEUR DU 8086/88

PLAN

Chapitre 3

PROGRAMMATION ASSEMBLEUR DU 8086/88

I/ LE PROGRAMME SOURCE

Pour aboutir à un programme correct exécutable par le processeur 8086, il est nécessaire de suivre les étapes suivantes :

1. Traduire le cahier des charges en un organigramme qui décrit le fonctionnement du programme.
2. Ecrire les instructions du programme sur papier.
3. A l'aide d'un éditeur de texte, écrire le programme dans un fichier ayant l'extension « **.asm** ».
4. Compiler et assembler le programme source a l'aide d'un compilateur (masm.exe ou tasm.exe). Le compilateur va détecter s'il y a des erreurs dans votre programme. Si le programme est syntaxiquement correct, le compilateur génèrera un fichier ayant le même nom que le fichier source mais avec l'extension « **.obj** ».
5. Faire l'édition de lien du programme à l'aide d'un éditeur de lien (link.exe ou tlink.exe), on obtient ainsi un programme exécutable qui a l'extension « **.exe** ».
6. Exécuter le programme.
7. Si les résultats sont erronés, alors faire le debuggage du programme a l'aide d'un débogueur comme le « **debug.exe** » pour localiser l'erreur, puis faire la correction.

II/ CONSTITUTION D'UN PROGRAMME SOURCE

Un programme assembleur est un ensemble de déclarations constitué de directives et d'instructions.

a) Les Instructions

Format : [Label :] Mnémonique [opérandes] [;commentaire]

Le champs Mnémonique est toujours obligatoire, les autres sont optionnels.

b) Les Directives

Format : [Nom] Directive [opérande] [; commentaire]

Seul le champs Directive est toujours obligatoire.

- **Les directives de données : (EQU; DB; DW ; DD)**

- EQU :

```
Nom  EQU expression ; assigne le nom de l'expression au nom
```

```
Exemples :
        I   EQU  2.54 ; valeur constante
        G EQU  INCH ; un autre nom symbolique  (identificateur)
        AG EQU   I*2-1 ;une expression
```

- DB :

```
[nom] DB expression [ ;commentaire]
```

DB réserve des mots a un octet.

```
Exemples :
Nb_max DB 255 ; déclaration d'une variable  initialisée a 255
Nb_min1 DB  -128
Nb_min2 DB  127
Tab_oct  DB 3 DUP(0), -5, -100, 2 DUP(30), ? ;  cette table réserve  des  cases d'octets
initialisées  avec 0,0,0,-5,-100,30,30,30, et une  case non initialisée.
Message DB 'Bonjours a tous$'
```

- DW :

```
[nom] DW expression [ ;commentaire]
```

DW réserve des mots a deux octets.

- DD :

```
[nom] DD expression [ ;commentaire]
```

DD réserve des mots a quatre octets.

- **Les directives de segment et de procédure (Assume ; segment ; proc/endp)**

- **segment :** cette directive délimite un segment.

```
nom_seg  SEGMENT [align][combine][‘class’]
……
……
……
nom_seg ENDS
```

ENDS marque la fin du segment.

```
Exemple :
DONNEE   SEGMENT
  Message  DB ‘ bonjours a tous$ ‘
DONNEE ENDS
```

- **assume :** Cette directive indique à l’assembleur que tel segment appartient à tel registre.

```
Assume reg_seg :nom_seg [ ;commentaire ]
```

Exemple:

```
CODE segment
Assume CS:CODE, DS :DATA
    …….
    …….
    …….
CODE ENDS
```

- **proc/Endp** : Cette directive marque le début et la fin d’une procédure.

```
nom_procédure  Proc
   …….
   …….
Endp
```

- **Macro/Endm** : Cette directive marque le début et la fin d’un macro.

```
nom_macro Macro     [param1,param2,……,paramn]
   …….
   …….
   …….
   …….
Endm
```

- **Les directives de controle (ORG ;END)**

- **ORG :**

Syntaxe : ORG expression [; commentaire]

Cette directive met le compteur de location à l'adresse produite par l'expression. L'assembleur range le code objet du programme à partir de cette adresse.

```
Exemple :
ORG 100h ; fait ranger le programme à 100h octets à partir du début de
         ;segment de code.
```

- **END :**

Syntaxe : END expression [; commentaire]

Cette directive marque la fin du programme source. « expression » est généralement un label à partir duquel débutera l'exécution du programme.

```
Exemple :
END main ; main est l 'adresse de debut
```

c) Les Opérateurs

- **Les opérateurs arithmétiques (+ , - , * , / , MOD,SHL , SHR)**

- L'opérateur + :

Syntaxe	Valeur1 + Valeur2
ExempleR_etat	DW Base + 2

- L'opérateur - :

Syntaxe	Valeur1 - Valeur2
Exemple R_donneés	DW R_etat - 2

L'opérateur * :

Syntaxe	Valeur1 * Valeur2
Exemple	Distance EQU Vitesse * temps

- L'opérateur / :

Syntaxe	Valeur1 / Valeur2
Exemple	Heure EQU 150/60

- L'opérateur MOD :

Syntaxe	Valeur1 MOD Valeur2
Exemple	Minutes EQU 150 MOD 60

- L'opérateur SHL :

Syntaxe	Valeur SHL expression
Exemple	Mask EQU 10111001B

Mask EQU Mask SHL 3; décalage gauche du Mask de 3 bits

- L'opérateur SHR :

Syntaxe Valeur SHL expression

Exemple Mask EQU 10111001B

Mask EQU Mask SHR 3; décalage droite du Mask de 3 bits

- **Les opérateurs logiques(AND,OR,XOR,NOT)**

- AND :Syntaxe Valeur1 AND Valeur2
- OR :Syntaxe Valeur1 OR Valeur2
- XOR :Syntaxe Valeur1 XOR Valeur2
- NOT :Syntaxe NOT Valeur1

- **Les opérateurs relationnels (EQU,NE,LT,GT,LE,GE)**

- EQU :Syntaxe Operande1 EQU Operande2

Si on a égalité des deux opérandes alors le résultat est FFFFH sinon le résultat est 0.

- NE :Syntaxe Operande1 NE Operande2

Si on a égalité des deux opérandes alors le résultat est 0 sinon le résultat est FFFFH.

- LT :Syntaxe Operande1 LT Operande2

Si on a Operande1 < Operande2 alors le résultat est FFFFH sinon le résultat est 0.

- GT : Syntaxe Operande1 GT Operande2

Si on a Operande1 > Operande2 alors le résultat est FFFFH sinon le résultat est 0.

- LE :Syntaxe Operande1 LE Operande2

Si on a Operande1 <= Operande2 alors le résultat est FFFFH sinon le résultat est 0.

- GE : Syntaxe Operande1 GE Operande2

Si on a Operande1 >= Operande2 alors le résultat est FFFFH sinon le résultat est 0.

- **Les opérateurs retournant une valeur (SEG, OFFSET, TYPE, SIZE, LENGHT)**

- SEG : Syntaxe SEG variable ou SEG label

SEG retourne la valeur segment d'une variable ou d'un label.

Exemple : mov AX,SEG table

- OFFSET : Syntaxe OFFSET variable ou OFFSET label

OFFSET retourne la valeur offset d'une variable ou d'un label.

Exemple : mov AX,OFFSET table.

- TYPE : Syntaxe TYPE variable ou TYPE label

Si l'opérande est une variable alors TYPE retourne 1 pour BYTE, 2 pour MOT ou 4 pour WORD. Si l'opérateur est un label alors TYPE retourne –1 si NEAR ou –2 si FAR.

Exemple : mov AX,TYPE table.

- SIZE : Syntaxe SIZE variable

  SIZE retourne le nombre d'octets alloués par une variable.

Exemple :soit

Table DW 100 DUP(?)

On a alors mov AX,SIZE table met dans AX 200

- LENGHT : Syntaxe LENGHT variable

  LENGHT retourne le nombre d'unités alloués par une variable.

Exemple :soit

Table DW 100 DUP(?)

On a alors mov AX,SIZE table met dans AX 100.

III/LES MODES D'ADRESSAGE DU 8086/88

Le microprocesseur 8086/88 possède sept modes d'adressage :

1. Mode d'adressage immédiat.
2. Mode d'adressage registre
3. Mode d'adressage direct
4. Mode d'adressage registre indirect
5. Mode d'adressage relatif a une base
6. Mode d'adressage direct indexé
7. Mode d'adressage indexé

1- Mode d'adressage immédiat.

Dans le mode d'adressage immédiat, l'opérande source est une donnée sur 8 ou 16 bits. L'exécution d'une telle instruction est très rapide.

Exemples :

MOV CX,500H ; dans CX=0000 0101 0000 0000 =0500H

MOV CX, -40H ; dans CX=1111 1111 1100 0000 =FFC0H

2- Mode d'adressage registre.

Dans le mode d'adressage registre, l'opérande à utiliser est contenu dans un des registres généraux du CPU (8 bits ou 16 bits).

La longueur de l'opérande source et l'opérande destination doivent être identiques.

Exemples :MOV AX,BX

MOV AL,BL

3- Mode d'adressage direct.

Le mode d'adressage direct spécifie complètement dans l'instruction l'emplacement mémoire qui contient l'opérande. Dans ce type d'adressage, l'adresse est directement ajoutée au contenu du registre segment DS (multiplié par 16) four former l'adresse physique.

Exemple : MOV AX,COMPTE ;COMPTE est une variable déjà déclarée

Dans cette instruction, le contenu des cases mémoire dont l'adresse est pointée par DS :COMPTE et DS :COMPTE+1 sera transféré dans AX.

4- Mode d'adressage registre indirect.

Dans le mode d 'adressage registre indirect, l'adresse n'et pas donnée dans l'instruction mais elle se trouve dans un registre intermédiaire qu'il faudrait évidemment charger au préalable par la bonne adresse.

Exemple : MOV BX, offset COMPTE

MOV AX,[BX]

La première instruction signifie qu'on a mis dans le registre BX, l'offset de la variable COMPTE.

La deuxième instruction signifie qu'on a charger le registre AX par le contenu de la case mémoire dont l'adresse se trouve pointée par le registre BX, (qui est l'offset de COMPTE).

5- Mode d'adressage relatif à une base.

Dans le mode d'adressage relatif à une base, l'adresse effective est la somme d'une valeur de déplacement plus le contenu du registre BX ou BP.

Avec le registre BX on peut accéder à divers structure de données qui se trouvent en différents endroits de la mémoire. Pour ce la il suffit de mettre l'adresse de base dans le registre de base (BX ou BP) et on pointe les éléments de la structure par leurs déplacement par rapport à la base. Pour se déplacer vers une autre structure, il suffit de changer le registre de base.

Exemples :

AX←DS :BX+3	**AX←SS :BP+3**
MOV AX,[BX]+3	MOV AX,[BP]+3
MOV AX,[BX+3]	MOV AX,[BP+3]
MOV AX,3[BX]	MOV AX,3[BP]

6- Mode d'adressage direct indexé.

Dans le mode d'adressage direct indexé, l'adresse effective est la somme d'un label ou d'un déplacement et d'un registre indexe SI ou DI.

Exemples :

- TAB est une table d'octets (TAB DB 100 DUP(?))

MOV DI,2

MOV AL,TAB[DI]

Cette instruction charge le 3 ème octet du TAB dans le registre AL.

- TAB est une table de mots (TAB DW 100 DUP(?))

MOV DI,4

MOV AX,TAB[DI]

Cette instruction charge le 3 ème mot du TAB dans le registre AX.

7- Mode d'adressage base indexé.

Dans le mode d'adressage base indexé, l'adresse effective est la somme de trois composantes : le registre de base, le registre index et un déplacement ou un label.

Ce modes d'adressage est intéressant dans le cas ou on veut accéder a une structure bidimensionnelle comme le matrices.

Dans ce cas le registre de base contient l'adresse de départ du tableau, le registre index contient le numéro de la ligne et le déplacement le numéro de la colonne.

Exemple : MOV AX,[BX][SI]

8- RESUME

Mode D'adressage	Format Opérande	Registre Segment	Exemple
Immédiat	Donnée	Aucun	MOV AX,5
Registre	Registre	Aucun	MOV AX,BX
Directe	Déplacement ou Label	DS	MOV AX,COMPTE

Registre Indirect	[BX]
	[BP]
	[SI]
	[DI]
		DS
		SS
		DS
		DS
			MOV AX,[BX]
			MOV AX,[BP]
			MOV AX,[SI]
			MOV AX,[DI]

Relatif à une Base	[BX]+Déplacement [BP]+Déplacement	DS SS	MOV AX,[BX]+3 MOV AX,[BP]+3
Direct Indexé	Label + [SI] ou Label[SI] Label + [DI] ou Label[DI]	DS DS	MOV AL,MSG[SI] MOV BL, MSG+[DI]

Basé Indexé	Label+[BX][SI]
	Label+[BX][DI]
	Label+[BP][SI]
	Label+[BP][DI]
		DS
		DS
		SS
		SS
			
			MOV AX, MATR[BX][DI]
			MOV AX, MATR[BP][SI]

Chapitre 4

LE JEUX D'INSTRUCTION DU 8086

PLAN

Chapitre 4

LE JEUX D'INSTRUCTION DU 8086

I-Les types d'instructions :

Le 8086 possède 92 types d'instructions de base qu'on peut diviser en sept groupes fonctionnels :

Les instructions de transfert de données

Les instructions arithmétiques

Les instructions Logiques

Les instructions de branchement

Les instructions de chaînes de caractères (string)

Les instructions d'interruption

les instructions de contrôle de processeur

II- Les instructions de transfert de données

a- Les instructions a usage général

MOV destination, source

PUSH source

POP destination

PUSHF

POPF

XCHG destination, source

XLAT table_source

b- Entrée / Sortie

IN accumulateur, port

OUT port, accumulateur

c- Transfert d'adresses

LEA reg16,mem16

LDS reg16, mem32

LES reg16, mem32

III- Les instructions arithmétiques :

a) Addition

ADD destination, source

ADC destination, source

AAA

DAA

INC destination

b) Soustraction

SUB destination, source

SBB destination, source

AAS

DAS

DEC destination

NEG destination

CMP destination

c) Multiplication

MUL source

IMUL source

AAM

Division

DIV source

IDIV source

AAD

d) Extension de signe

CBW

CWD

IV- Les instructions Logiques

a) Logiques

AND destination, source

OR destination, source

XOR destination, source

NOT destination, source

TEST destination, source

b) Décalage

SAL/SHL destination, compte

SAR destination, compte

SHR destination, compte

c) Rotation

ROL destination, compte

ROR destination, compte

RCL destination, compte

RCR destination, compte

V- Les instructions de branchement :

a) Branchement inconditionnel

CALL cible

RET [valeur pop]

JMP cible

b) Branchement conditionnel

JX label_court

c) Contrôle d'itération

LOOP label_court

LOOPE/LOOPZ label_court

LOOPNE/LOOPNZ label_court

VI- Les instructions de chaînes de caractères (string)

a) Préfixes de répétition

REP

REPE/REPZ

REPNE/REPNZ

b) Move

MOVS chaîne_dest, chaîne_source

MOVSB

MOVSW

c) Compare

CMPS chaîne_dest, chaîne_source

CMPSB

CMPSW

d) Scan

SCAS chaîne_source

SCASB

SCASW

e) Load et Store

LODS chaîne_source

LODSB

LODSW

STOS chaîne_destination

STOSB

STOSW

VII- Les instructions d'interruption :

INT type d'interruption

INTO

IRET

IIX-Les instructions de contrôle de processeur :

a) Opérations flags

STC

CLC

CMC

STD

CLD

STI

CLI

b) Synchronisation externe

HLT

WAIT

LOCK

ESC opcode_externe, source

NOP

Chapitre 5

LA GESTION DES ENTREES / SORTIES

PLAN

Chapitre 5

LA GESTION DES ENTREES/SORTIES

I- INTRODUCTION

Une interruption est fondamentalement un événement externe qui survient à un moment imprévu, cela alors que le microprocesseur vaque normalement à sa tâche. Quelqu'un ou quelque chose réclame son attention immédiate et le prie de bien vouloir interrompre un instant son travail.

C'est, par exemple, l'utilisateur qui frappe une nouvelle commande sur son clavier. Ou un incendie qui vient de se déclarer et dont il faut s'occuper. Le microprocesseur s'occupe alors de la cause de cette interruption, fait ce qu'il y a à faire. Puis, le traitement de l'interruption terminé, il revient à sa tâche primitive qu'il s'était vu contraint d'abandonner pour la poursuivre comme si de rien n'était.

Les interruptions constituent l'un des trois modes d'échanges appliqués par le processeur, les deux autres étant les échanges programmés et l'accès direct en mémoire, ou DMA.

II- Le dialogue entre l'UC et les périphériques

Pour dialoguer avec ses périphériques, l'unité centrale fait appel à trois principes:

1. ***Les échanges programmés :*** C'est le programme qui ordonne un accès à un élément périphérique quelconque.

2. ***Les interruptions :*** L'échange peut désormais avoir lieu de manière aléatoire, à n'importe quel instant, uniquement lorsque le périphérique le demande.

3. ***Le DMA :*** ou accès direct à la mémoire (*«Direct Memory Access»*). Ce mode d'échange privilégie la vitesse.

Chacune de ces méthodes offre des avantages et des inconvénients. C'est pourquoi elles sont complémentaires et non concurrentes.

III- Echanges programmés

La façon la plus immédiate de gérer des échanges consiste à les prévoir dans le programme que l'ordinateur est en train d'exécuter. Au bon moment, une instruction demandera au microprocesseur d'émettre un message vers un périphérique, ou d'observer si l'utilisateur a tapé quelque chose sur son clavier, ou, s'il a déplacé la souris, si le modem a reçu un message, etc.

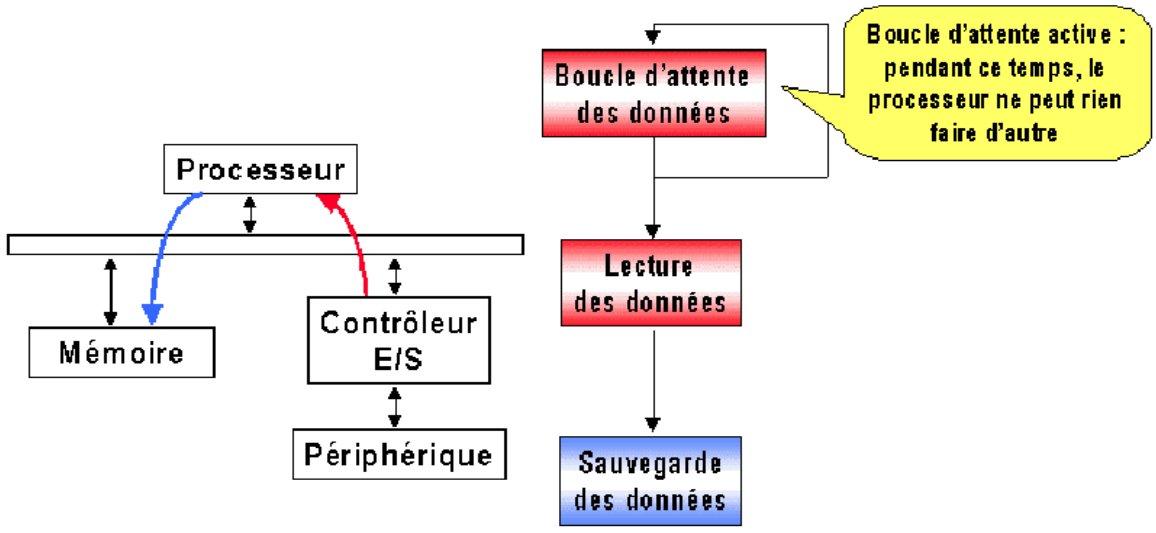
le mode d'échange programmé.

Que se passe-t-il lorsque le microprocesseur reçoit une instruction d'échange programmé ? La procédure est la suivante :

1. L'adresse du périphérique visé est mise sur le bus d'adresse.
2. Le processeur attend un court instant que tous les bits d'adresse soient stables.
3. Sur le bus de commande, il émet un ordre d'échantillonnage. Les périphériques sont informés que l'adresse est valide mais un seul la reconnaîtra comme sienne.
4. Le processeur place les données sur le bus de données s'il s'agit d'une écriture vers le périphérique. Si c'est l'inverse, une entrée, ce serait le périphérique qui placerait les données sur le bus.
5. Le processeur active un ordre d'échantillonnage sur le bus de commande pour indiquer que les données sont stables et bonnes pour utilisation.
6. L'échange a lieu et est terminé.

En résumé, cette formule d'échanges programmés par scrutation apporte :

- **Avantages :** la programmation est facile car tout est prévu dans le programme. Les échanges s'effectuent à des moments qu'on a soi-même choisis, donc sans surprise. La mise au point du programme est plus facile.
- **Inconvénients:** le traitements des échanges aléatoires, asynchrones, est plus difficile. Le balayage permanent des périphériques fait perdre beaucoup de temps et réduit la vitesse d'exécution du programme. On introduit des instructions le plus souvent inutiles pour les périphériques.



IV- Echanges par interruptions

Le principe le plus familièrement possible est le suivant :

Le microprocesseur est un personnage très occupé. On ne veut pas constamment le déranger surtout lorsqu'il est en train d'exécuter un programme.

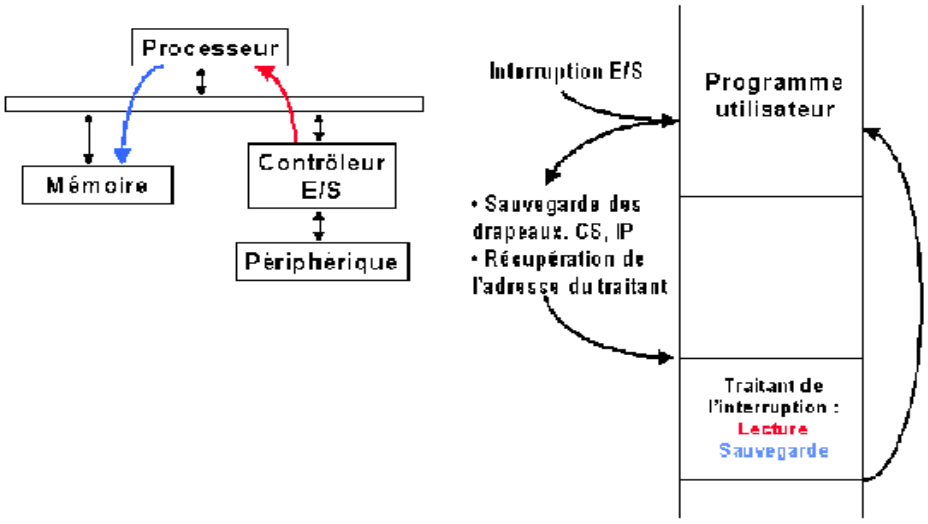
Pourtant, à un moment ou à un autre, vous désirez passer une nouvelle commande en la tapant au clavier. Dans ce cas, c'est le clavier qui va se charger d'informer le microprocesseur que l'utilisateur manifeste sa volonté, laquelle est indiscutablement prioritaire.

Pour cela, le clavier envoie un signal électrique au microprocesseur. Ce dernier le reçoit et comprend qu'on réclame son attention. Le microprocesseur termine l'instruction qu'il était en train d'effectuer, puis il range dans la mémoire tout ce qui concerne le programme en cours afin de pouvoir le retrouver en bon état plus tard. Après quoi, il examine d'où provient la demande d'interruption, puis il appelle le programme spécial appelé le programme «gestionnaire du clavier» dans le cas qui vient d'être évoqué. Enfin, il prévient le clavier : «*OK, tu peux y aller* !».

Le clavier transmet ses informations au microprocesseur. Ce dernier les «traite» immédiatement, c'est-à-dire qu'il exécute vos ordres.

Lorsque c'est terminé, le microprocesseur abandonne cette séquence. Il rappelle le programme précédent dont il avait rangé les éléments en mémoire. I1 le reprend exactement au point où il l'avait abandonné et en poursuit l'exécution.

C'est l'ensemble de cette procédure qu'on appelle une **interruption**. Parce que ce type d'interruption est déclenché par le matériel, on dit encore qu'il s'agit d'une **interruption matérielle**.



V- Echanges par DMA (8237)

Dans ce mode de transfert on utilise un circuit contrôleur appeler DMA (Direct Memory Access) qui permet les échanges d'information entre la mémoire et les périphériques.

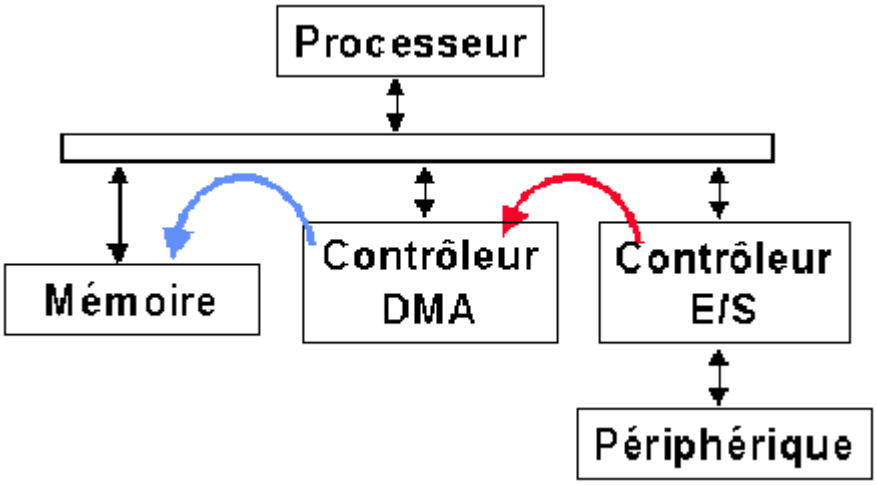
Le processeur envoie au contrôleur DMA:

- l'adresse de début
- la longueur des données
- Le sens du transfert

Puis déclenche le transfert

Le contrôleur DMA prend en charge:

- les commandes pour le contrôleur de périphériques
- les commandes et adresses pour la mémoire.



Ø**Avantages :**

- DMA externe / processeur
- transfert de données sans passer par le processeur
- DMA prioritaire sur le bus

Chapitre 6

LES BUS

PLAN

Chapitre 6

LES BUS

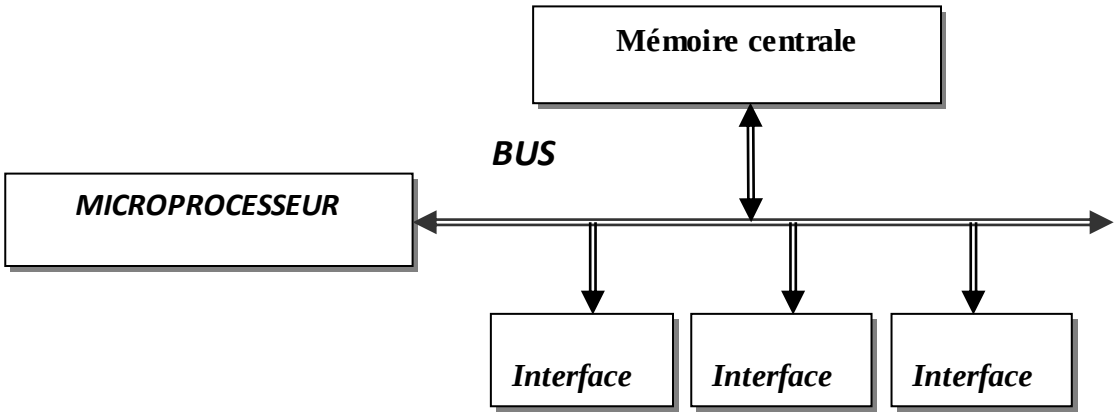
I. GENERALITES

- ✓ Le bus est physiquement constitué par un ensemble de fils conducteurs montés en parallèle.
- ✓ C'est l'équivalent d'un câble d'un point de vue logique. Le bus est un chemin partagé qui permet à l'ensemble des unités du micro ordinateur de communiquer entre elles.
- ✓ C'est est un moyen de transmission des informations ou des signaux groupés par fonction.
- ✓ Un bus est utilisé alors comme :

- Bus de données : c'est un bus bidirectionnel. Il transmet les données échangeaient par les différents composants de l'ordinateur.

↳ Généralement ce type de bus comprend autant de fils qu'il y a de bits dans un mot mémoire afin de lire ou d'écrire un mot mémoire en un seul accès

- Bus d'adresse : c'est un bus unidirectionnel, il sert à sélectionner la source pour la destination. Il indique à la mémoire et aux autres périphériques l'emplacement avec lequel il veut communiquer. Le nombre de fils de ce bus dépend de la taille de l'espace adressable.
- Bus de commande : il règle les transferts de données et assure la synchronisation des périphériques avec le micro processeur. Il transporte des informations d'état et des commandes dirigés vers la CPU ou en provenance de celle ci. C'est un bus bidirectionnel.



- ✓ Cependant, les informations doivent non seulement circuler dans le microprocesseur, mais également à l'extérieur de celui, de manière à communiquer avec la mémoire, les périphériques…

↳ On parle alors, dans le cas des micro-ordinateurs, du ***bus d 'extension***.

II. CARACTERISTIQUES D'UN BUS :

Un bus est caractérisé par :

- Le nombre de fils employés :8,16,32 bits…
- La nature des info. véhiculées : données, @ , commandes
- Le mode de fonctionnement : synchrone ou asynchrone
- Le fait qu'il soit « intelligent » (bus master) ou non,
- La manière dont sont transmises les informations (en parallèle ou en série)
- La fréquence du bus (la vitesse du bus)

III. MODE DE FONCTIONNEMENT :

Les bus sont divisés en deux classes distinctes selon le cadencement des échanges : ***les bus synchrones*** et ***les bus asynchrones***

1. LES BUS SYNCHRONES

- ✓ Les bus synchrones disposent d'une ligne spécifique d'horloge : toute opération sur le bus synchrone est effectuée en un nombre entier de périodes d'horloge.
- ✓ La période de l'horloge du bus correspond généralement au cycle du bus.
- ✓ Ils présentent toutes fois quelques limitations notamment, le fait que toute opération corresponde a un nombre entier de cycle de bus n'est pas nécessairement optimum vis à vis des performances

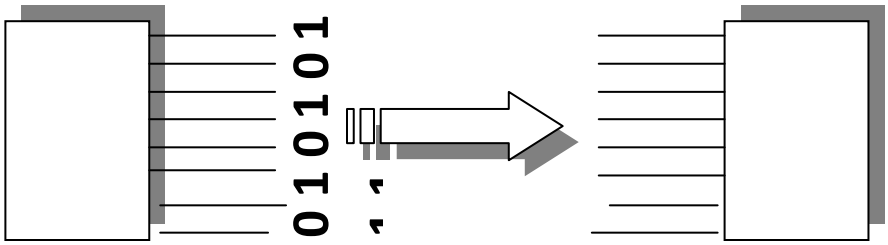
2. LES BUS ASYNCHRONES

- ✓ Les bus asynchrones ne disposent pas d'horloge pilote de bus
- ✓ Il est plus facile de construire un bus synchrone au lieu d'un bus asynchrone

IV. MODES DE TRANSMISSIONS DE L'INFORMATION :

1. BUS PARALLELE :

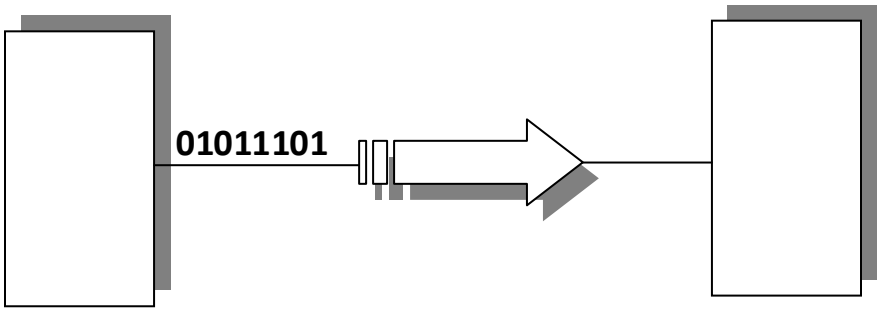
- ✓ La solution la plus simple que l'on puisse envisager, pour faire circuler un certain nombre de bits à la fois (8, 16, 32 ou 64 bits), consiste à :
- ↳ Utiliser autant de fils qu'il y a de bits
- ✓ Un tel mode de transmission est dit parallèle, mais n'est utilisable que pour des transmissions à courte distance, car coûteux et peu fiable sur des distances importantes.



- ✓ C'est le mode de transmission utilisé au sein de l'unité centrale entre processeur, la mémoire, les contrôleurs…

2. BUS SERIE :

- ✓ Pour des transmissions à plus grande distance, on utilisera alors une seule voie où les bits qui constituent les caractères sont transmis les uns après les autres : c'est la transmission série.
- ✓ Chaque bit est envoyé à tour de rôle.
- ✓ Principe de base :
- Appliquer une tension +V pendant une intervalle pour représenter un bit 1 et une tension nulle pour représenter un bit 0
- Côté récepteur, on doit alors observer les valeurs de la tension aux instants convenables.
- Une synchronisation est nécessaire, entre émetteur et récepteur pour que ce dernier fasse ses observations aux instants corrects.



- ✓ Dans tous les premiers systèmes, les informations circulaient sur un ***ensemble de fils uniques*** mettant le ***processeur*** en relation avec la ***mémoire*** ou les ***entrées-sorties***
- ✓ Ainsi, un programme faisant de nombreux appels à la mémoire monopolisait ce bus unique au détriment des autres demandeurs.
- ✓ Un premier palliatif a consisté à prévoir pour le processeur des accès directs à la mémoire au travers d'un canal ***DMA*** (*Direct Memory Access*).
- ↳ Cette technique est toujours en vigueur.

- ✓ On a ensuite réfléchi à une ***séparation*** des bus selon leur fonction de manière à obtenir des bus ***spécialisés.***
- ✓ On peut ainsi distinguer entre ***bus processeur***, ***bus local***, ***bus global*** et ***bus d'entrées-sorties*** :

- Le ***bus processeur*** ou ***bus privé*** est le bus spécifique au microprocesseur.
- Le ***bus local***, dit aussi ***bus d'extension*** prolonge le bus privé, permettant de relier directement certains composants du système tels que la mémoire… au microprocesseur.
- Le ***bus global*** qui correspond également à un bus d'extension, a pour rôle de relier entre elles les différentes cartes processeur dans une machine multiprocesseur. Il est maintenant souvent confondu avec le bus local.

- Le ***bus d'entrées-sorties*** sert aux communications avec des périphériques lents. Ils correspondent aux sorties séries ou parallèles.

✓ Le bus, comme tous les autres composants a également suivi une évolution historique, passant ainsi du bus ISA PC AT aux bus EISA, VESA, PCI, USB…

V. EXEMPLES DE BUS :

1. LE BUS ISA OU PC-AT :

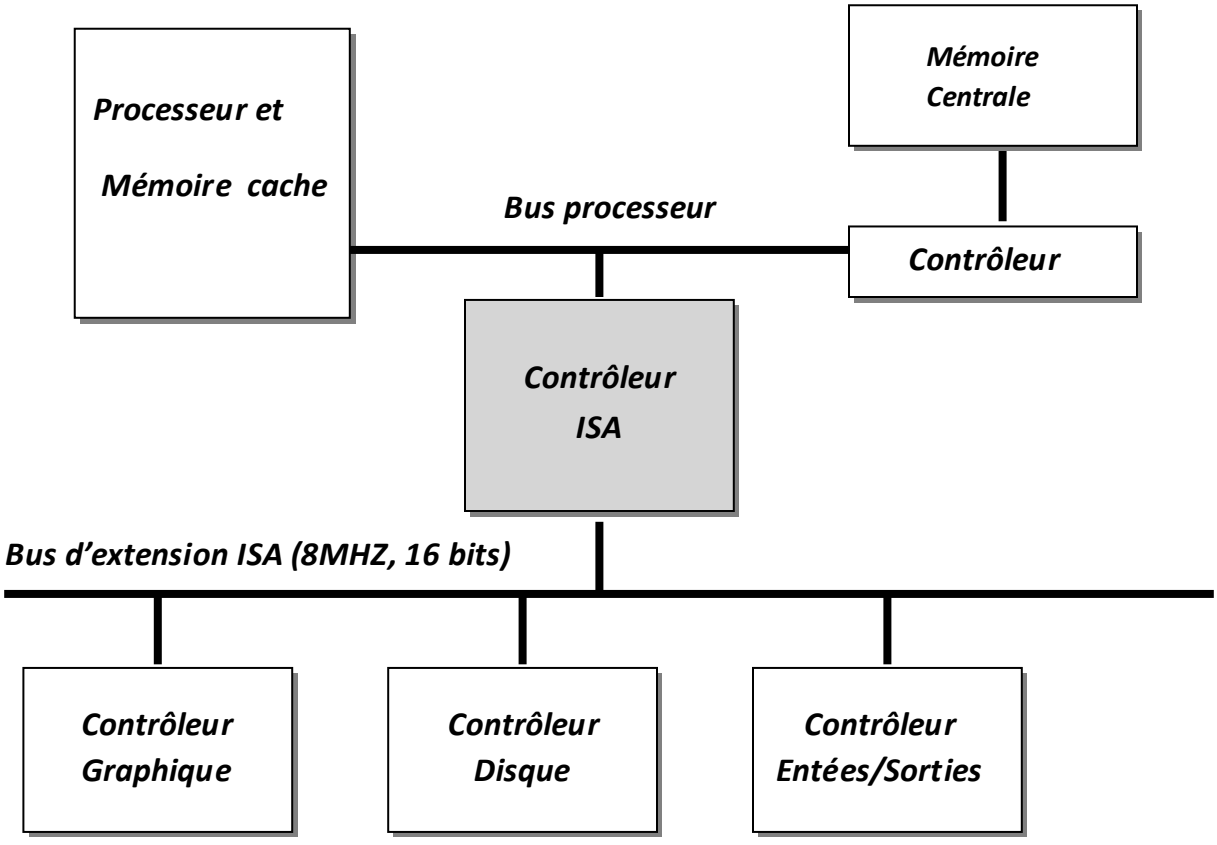
✓ Le bus ***ISA*** (***I****ndustry* ***S****tandard* ***A****rchitecture*) reste l'un des standards les plus répandus en matière de bus.

✓ Il est apparu en 1984 avec le micro-ordinateur IBM PC-AT d'où son surnom de ***bus AT*** ou ***AT-bus***.

↳ Le processeur est alors un Intel 80286 fonctionnant à 8 MHz, et le bus est synchronisé avec le processeur → les informations circulent à la même vitesse sur le bus extérieur au processeur et dans le processeur lui-même.

✓ Avec le ***bus ISA***, les cartes d'extension doivent être configurées matériellement ce qui se fait généralement en positionnant des ***micro-switchs*** ou en plaçant des cavaliers.

✓ Ce bus, d'une largeur de ***16 bits***, autorise des taux de transfert pouvant atteindre jusqu'à 8 Mo/s



- ✓ Les processeurs qui ont succédé au processeur 80286 sont des processeurs de ***32 bits***
- ↳ Ce qui pose évidemment problème (causé par la rapidité) et oblige les constructeurs à concevoir un ***bus d'extension*** pouvant fonctionner à des vitesses différentes de celle du processeur.
- ↳ Développement concurrente du bus MCA et du bus EISA
- ✓ Il se visualise sous forme de connecteurs noirs sur la carte mère.

2. LE BUS MCA :

- ✓ Le ***bus MCA*** (***M**icro **C**hannel **A**rchitecture*) développé en 1987 par IBM pour ses nouveaux PS/2 a marqué une réelle évolution.
- ✓ Bus 32 bits, ***asynchrone***, fonctionnant à 10MHz : dit bus « ***intelligent*** » ou ***bus master***
- ✓ ***MCA*** est en effet capable d'exploiter des cartes munies de leur propre processeur et gérant leurs ***entrées-sorties*** sans que n'intervienne le processeur de la carte mère, ainsi libéré pour d'autres tâches.
- ✓ L'architecture ***MCA*** étant indépendante du processeur, il peut être utilisé avec des processeurs Intel sur les PS/2 ou des processeurs ***RISC*** sur les RS-6000.
- ✓ ***MCA*** offre un taux de transfert de 20 à 50 Mo/s et supporte quinze contrôleurs « ***masterisés*** »
- ✓ Chaque carte d'extension est configurable par logiciel
- ✓ Toutefois ***MCA*** ne reconnaît pas les cartes du format ***ISA*** et son architecture complexe le rend coûteux à fabriquer.
- ✓ Il reste donc essentiellement utilisé par ***IBM***

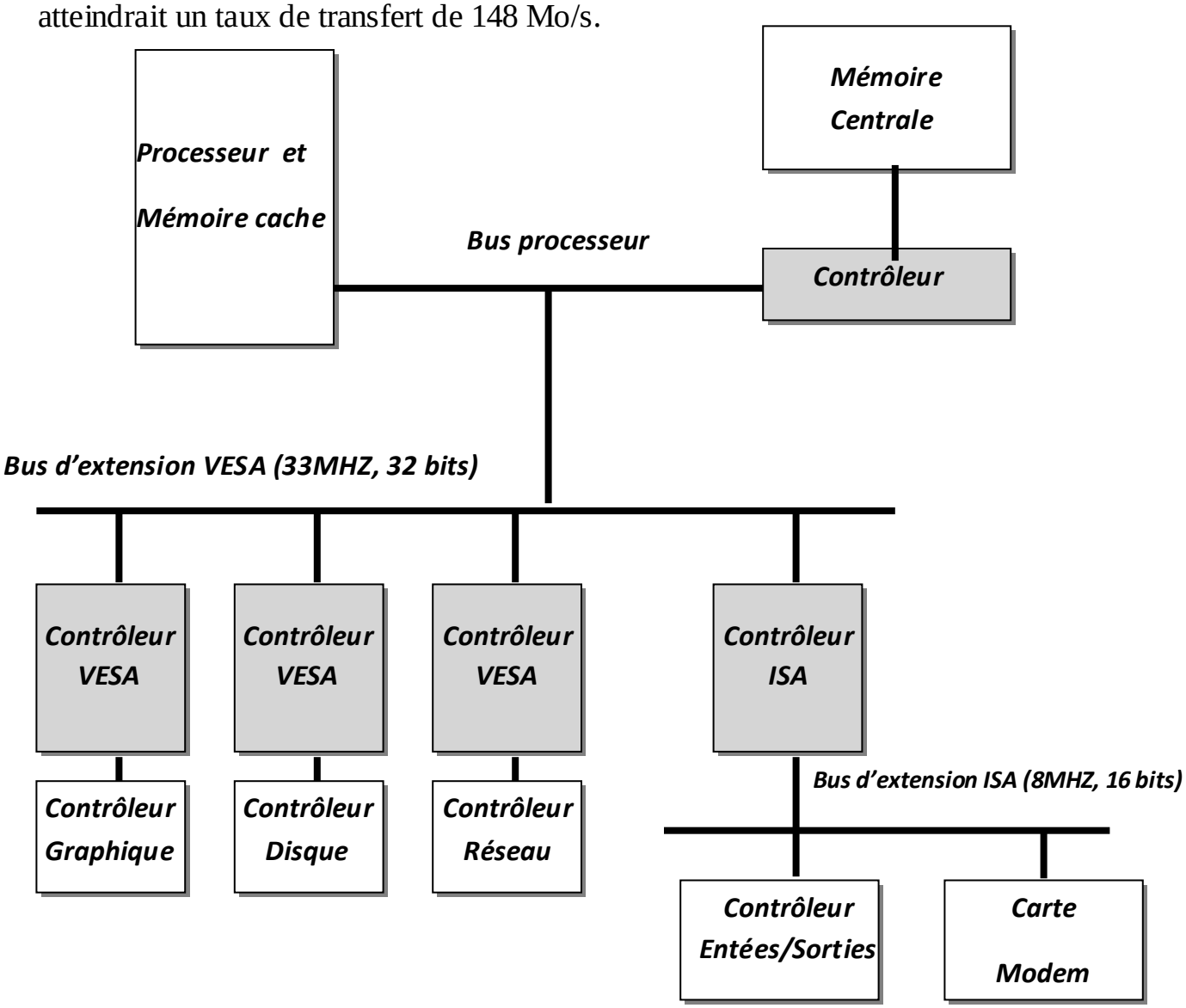
3. LE BUS EISA :

- ✓ Le ***bus EISA*** (***E**xtended **I**ndustry **S**tandard **A**rchitecture*) est apparu en 1988 pour concurrencer ***MCA*** développé par IBM.
- ✓ Un bus de ***32 bits*** prévu pour maintenir une certaine compatibilité avec le ***bus ISA***, ce qui l'oblige à continuer à fonctionner à 8 MHz comme ***le bus ISA***
- ✓ Son taux de transfert est de 33 Mo/s
- ✓ Il reprend certaines caractéristiques du ***bus MCA*** telles que ***la configuration logicielle des cartes d'extension*** et la notion des ***bus mastering***.
- ✓ Les ***contrôleurs*** sont différents de ceux utilisés avec un ***bus ISA*** et plus coûteux

4. LE BUS VESA :

- ✓ La ***norme VESA*** (***V**ideo **E**lectronics **S**tandard **A**ssociation*) ***Local Bus (VLB)***,apparue en 1993, définit dans sa version 1.0 les caractéristiques d'un ***bus local*** 32 bits, théoriquement extensible à 64 bits et fonctionnant en ***synchrone*** avec le microprocesseur
- ✓ ***VLB*** autorise l'utilisation du ***DMA*** (***D**irect **M**emory **A**ccess*) ainsi que le bus mastering.

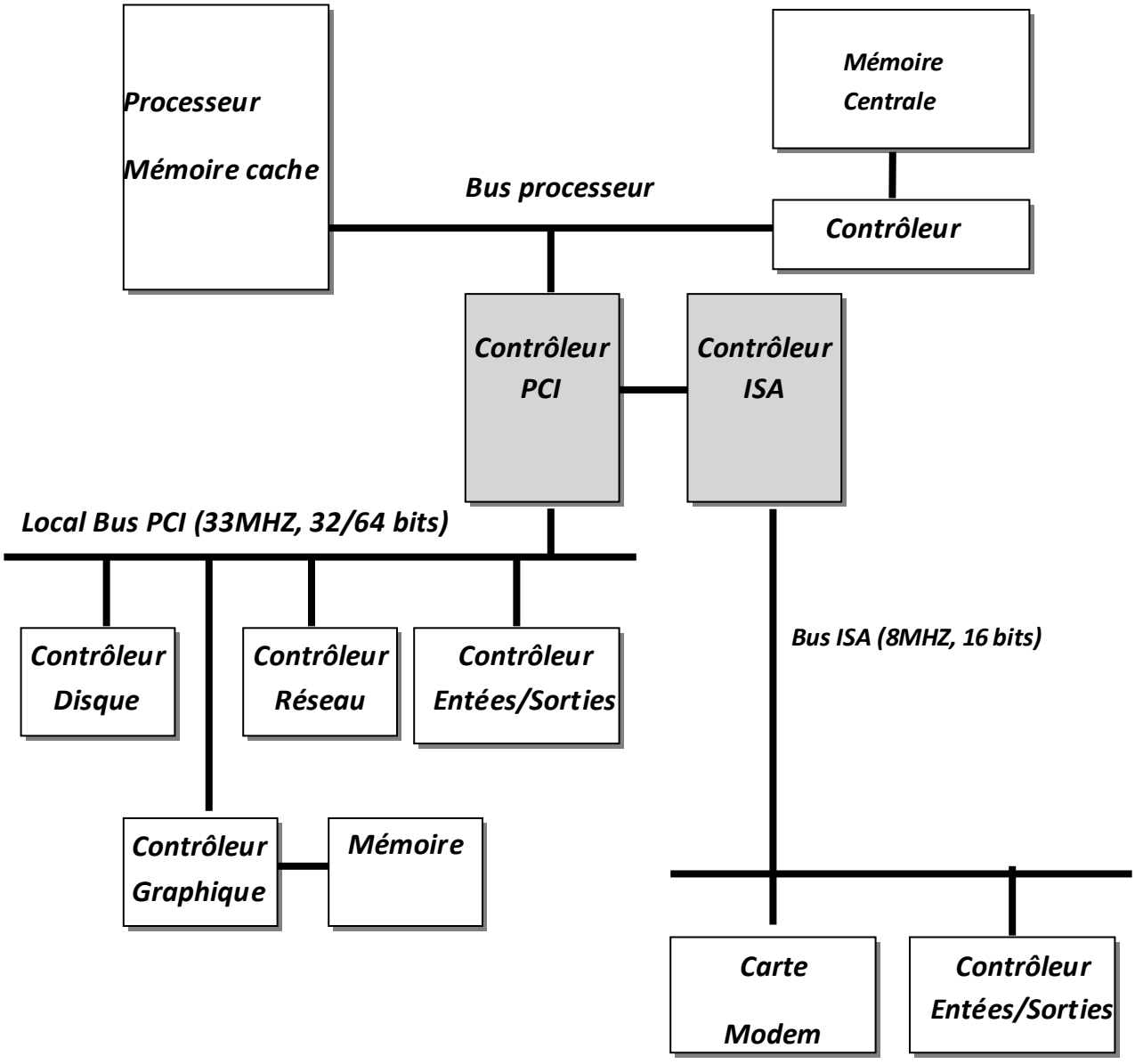
- ✓ Son architecture est très simple car il s'agit ni plus ni moins d'une ***extension du bus du processeur***.
- ✓ Le ***bus VESA***, de conception ***peu coûteuse***, fonctionne donc en ***synchrone*** avec le processeur- ainsi un processeur à 33MHz fonctionnera avec un ***bus VESA*** 33MHz.
- ✓ Ce bus permet de piloter ***trois connecteurs***
- ✓ Le taux de transfert potentiel est alors de 130 Mo/s. Avec un bus à 40 MHz on atteindrait un taux de transfert de 148 Mo/s.



- ✓ Malheureusement ***VLB*** a été conçu au départ pour des ***processeurs 486*** et reste donc lié aux caractéristiques de ce processeur
- ✓ Bien que la version ***VLB 2.0*** offre un bus 64 bits avec un débit atteignant en principe 264 Mo/s, ***VESA*** semble être actuellement en « ***perte de vitesse*** » au détriment du ***bus PCI***, qui est plus performant et d'une grande adaptabilité aux différents processeurs.

5. LE BUS PCI :

- ✓ Le ***bus PCI*** (***P**eripheral **C**omponent **I**nterconnect*) a été développé par Intel en 1993, concurremment à la norme ***VESA***.
- ✓ Il offre, dans sa version 1.0 un bus de 32 bits fonctionnant à 33 MHz ce qui permet d'atteindre un taux de transfert de 132 Mo/s comme avec un ***bus VESA***
- ✓ ***PCI*** présente toutefois le gros avantage d'être totalement indépendant du ***processeur***, ce qui n'est pas le cas du ***VESA Local Bus***.
- ✓ En effet, ***PCI*** dispose de sa propre ***mémoire tampon*** (buffer)
- ↳ C'est pourquoi on emploie également le terme de « ***pont*** » ***PCI-mémoire*** chargée de faire de lien entre ***le bus du processeur*** et les ***connecteurs d'extension***



- ✓ ***PCI*** est une architecture de bus qui peut être combinée avec une autre architecture de bus de type ***ISA*** ou ***EISA***

- ✓ Il offre l'avantage d'être autoconfigurable, les cartes connectées étant automatiquement détectées et exploitées au mieux :
- ↳ C'est le ***Plug*** and ***Play*** qui évite donc d'avoir à déplacer des switches sur la carte.
- ✓ Dans sa spécification 2.0, ***PCI*** présente de nouveaux connecteurs courts et surtout autorise l'accès 64 bits nécessaire à l'exploitation ***Pentium***.

Caractéristiques des bus

Type	Largeur	Fréquence	Nombre max. de connecteurs	Taux de transfert
ISA	16 bits	8-12.5 MHz	8	8 Mo/s
EISA	32 bits	8-12.5 MHz	18	50 Mo/s
MCA	32 bits	8-12.5 MHz	8	50 Mo/s
VESA	32 bits	33 MHz	3	132 Mo/s
PCI	32 bits	33 MHz	10	132 Mo/s

6. LE BUS SCSI :

- ✓ L'interface ***SCSI*** (***S****mall* ***C****omputer* ***S****ystem* ***I****nterface*) est un bus supportant divers périphériques, de plus en plus adoptée par les constructeurs.
- ✓ Le bus ***SCSI*** est un bus ***multimaître*** : signifie qu'il dispose de son propre microprocessur servant à le gérer

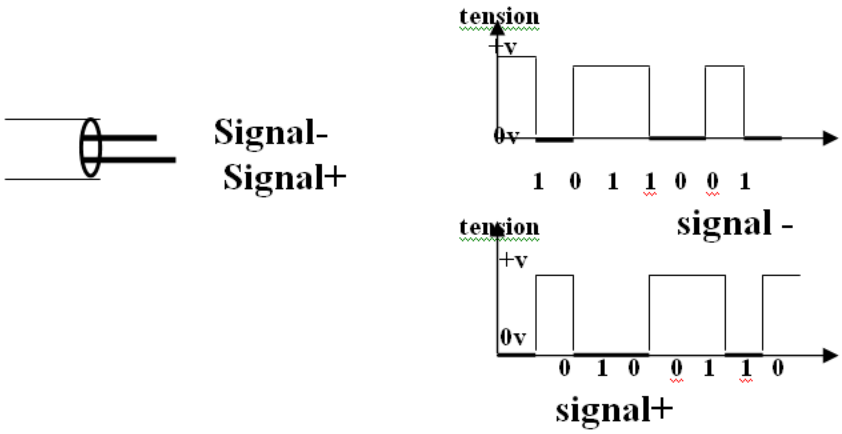
- ✓ Sa vitesse de transfert varie de l'ordre de 4 Mo/s à 80 Mo/s selon la largeur du bus et le standard ***SCSI*** employé (SCSI-1 à 4 Mo/s, SCSI-2, SCSI-3, Wide(Fast) SCSI à 20 Mo/s, Ultra 2 Wide SCSI à 80 Mo/s…).
- ✓ Ce débit dépend directement de la fréquence employée sur le bus :
- 10 MHz pour le Fast SCSI
- 20 MHz pour l'ultra SCSI
- 40 MHz pour l'ultra 2 SCSI

- ✓ Malheureusement, un inconvénient majeur est directement lié à cette augmentation de fréquence :
- ↳ La longueur du bus est inversement proportionnelle à la fréquence.
- ↳ Plus la fréquence augmente et plus la longueur du bus diminue.

- ✓ Or ***SCSI*** est prévu pour relier des composants supplémentaires aux traditionnels ***disque durs*** ou lectures ***CD-ROM***, tels que ***des scanners***, ***imprimantes***… qui ne sont pas forcément à portée immédiate de la machine.

- ✓ ***Ultra 2 SCSI-LVD***, dernière norme ***SCSI*** actuellement en vigueur, contourne ce problème en diminuant la tension employée sur le bus qui passe à 3.3v – ***LVD*** (***L****ow* ***V****oltage* ***D****ifferential*).
- ✓ La longueur est ainsi portée à 12 mètres contre 3 pour ***l'Ultra SCSI***.

- ✓ Le ***bus LVD*** est également dit « ***bus parallèle différentiel*** » car il fonctionne en mode différentiel en utilisant un fil pour le signal négatif et un autre fil pour le signal positif ce qui diminue l'effet des parasites électromagnétiques.



- ✓ L'interface ***Ultra 2 SCSI*** peut gérer jusqu'à ***31 unités physiques*** différentes ce qui permet de piloter ***disques durs ou optiques, streamer, scanner, imprimantes***…

		Taux de transfert			
	Fréquence Bus	**8 bits**	**16 bits (mode wide)**	**Distance max**	**Périph supportés**
SCSI	**5 MHZ**	**5Mo/s**	**Non**	**6m**	**7**
Fast SCSI, SCSI 2	**10 MHZ**	**10 Mo/s**	**20 Mo/s**	**3m**	**7**
Fast 20, Ultra SCSI	**20 MHZ**	**20 Mo/s**	**40 Mo/s**	**1.5 m avec 7 périph. 3m avec 3périph**	**7**
Fast 40,Ultra 2 SCSI	**40 MHZ**	**40 Mo/s**	**80 Mo/s**	**12m**	**31**

La gamme SCSI

7. LE BUS USB :

- ✓ Le ***bus USB*** (***U****niversal* ***S****erial* ***B****us*),apparu en 1995 ,est un ***bus série*** récent et évolué qui se veut le successeur des traditionnels bus séries et parallèles.
- ✓ Il permet d'exploiter ***127 périphériques*** – souris, clavier, imprimante, scanner… chaînés sur un canal.

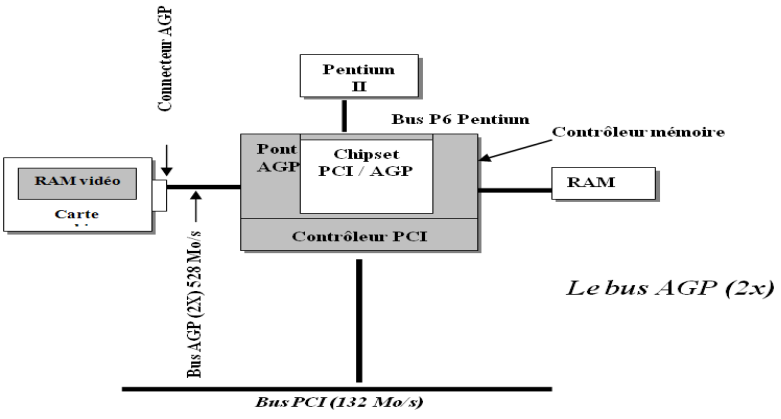
- ✓ De plus, de technologie ***Plug*** and ***Play***, il permet de reconnaître automatiquement le périphérique branché sur le canal et de déterminer automatiquement le pilote nécessaire au fonctionnement de ce dernier.
- ✓ Les caractéristiques de ce bus sont :
- Les informations sont codées en ***NRZI***
- Les informations circulent à un débit adapté au périphérique et variant de 1.5 à 12 MBps sur des câbles en ***paire torsadée*** n'excédant pas 5 mètres entre chaque périphérique
- Le port ***USB*** se présente généralement sous la forme d'un petit connecteur rectangulaire qui comporte ***4 broches***.
- ✓ ***USB*** utilise des principes de fonctionnement similaires à ceux employés dans les ***réseaux locaux*** (ensemble d'ordinateur qui sont connectés entre eux), autorisant à plusieurs périphériques un dialogue simultané sur le même bus. Par exemple impression d'un document tout en téléchargeant un fichier à l'aide du modem et en utilisant le clavier.

8. LE BUS FIREWIRE –LINK

- ✓ Le bus ***FireWire ou iLink*** dit également ***SCSI série***, est dédié aux périphériques rapides tels que des périphériques d'imagerie, PAO (Publication Assistée par Ordinateur)…
- ✓ En général, on ne le rencontre pas sur les cartes mères mais il se présente sous la forme d'une carte d'extension.
- ✓ Ce bus présente de nombreuses similitudes avec ***USB*** telles que le ***Plug*** and ***Play***
- ✓ Les débits de ce bus est de 10 à 20 fois supérieurs à ceux de ***l'USB***
- ✓ Fonctionne en mode ***bidirectionnel simultané***

9. LE BUS GRAPHIQUE AGP

- ✓ Le bus ***AGP*** (***A****ccelerated* ***g****raphics* ***p****ort*) est un bus récent – 1997 – spécialisé dans l'affichage.
- ✓ Il relie directement – au travers du ***chipset*** – le processeur de la carte
- ✓ graphique avec le processeur de l'UC et avec la mémoire vive



Le bus AGP (2x)

- ✓ Il offre un bus de 32 bits, un ***fonctionnement en mode Pipeline*** ce qui autorise des ***lectures et écritures simultanées en mémoire***
- ✓ Des débits atteignant 528 Mo/s (32bits à 66 MHZ) dans la version ***AGP 2X*** et la possibilité d'accéder également à la mémoire centrale en sus de la mémoire de la carte graphique
- ✓ On peut ainsi manipuler des images « lourdes » - affichage tridimensionnel 3D, sans saturer la mémoire de la carte graphique, puisqu'on peut placer une partie de l'image en mémoire centrale.
- ✓ La version de base ***AGP*** offre un débit de 264 Mo/s, deux fois supérieur à celui du bus ***PCI*** (132 Mo/s)
- ✓ Toutefois seules les cartes graphiques équipées de processeurs réellement compatibles ***AGP*** peuvent actuellement tirer partie de ce bus.

Chapitre 7

LA GESTION DES INTERRUPTIONS

PLAN

Chapitre 7

LA GESTION DES INTERRUPTIONS

I- INTRODUCTION

Une interruption est fondamentalement un événement externe qui survient à un moment imprévu, cela alors que le microprocesseur vaque normalement à sa tâche. Quelqu'un ou quelque chose réclame son attention immédiate et le prie de bien vouloir interrompre un instant son travail.

C'est, par exemple, l'utilisateur qui frappe une nouvelle commande sur son clavier. Ou un incendie qui vient de se déclarer et dont il faut s'occuper. Le microprocesseur s'occupe alors de la cause de cette interruption, fait ce qu'il y a à faire. Puis, le traitement de l'interruption terminé, il revient à sa tâche primitive qu'il s'était vu contraint d'abandonner pour la poursuivre comme si de rien n'était.

Les interruptions constituent l'un des trois modes d'échanges appliqués par le processeur, les deux autres étant les échanges programmés et l'accès direct en mémoire, ou DMA.

II- Echanges par interruptions

Le principe le plus familièrement possible est le suivant :

Le microprocesseur est un personnage très occupé. On ne veut pas constamment le déranger surtout lorsqu'il est en train d'exécuter un programme.

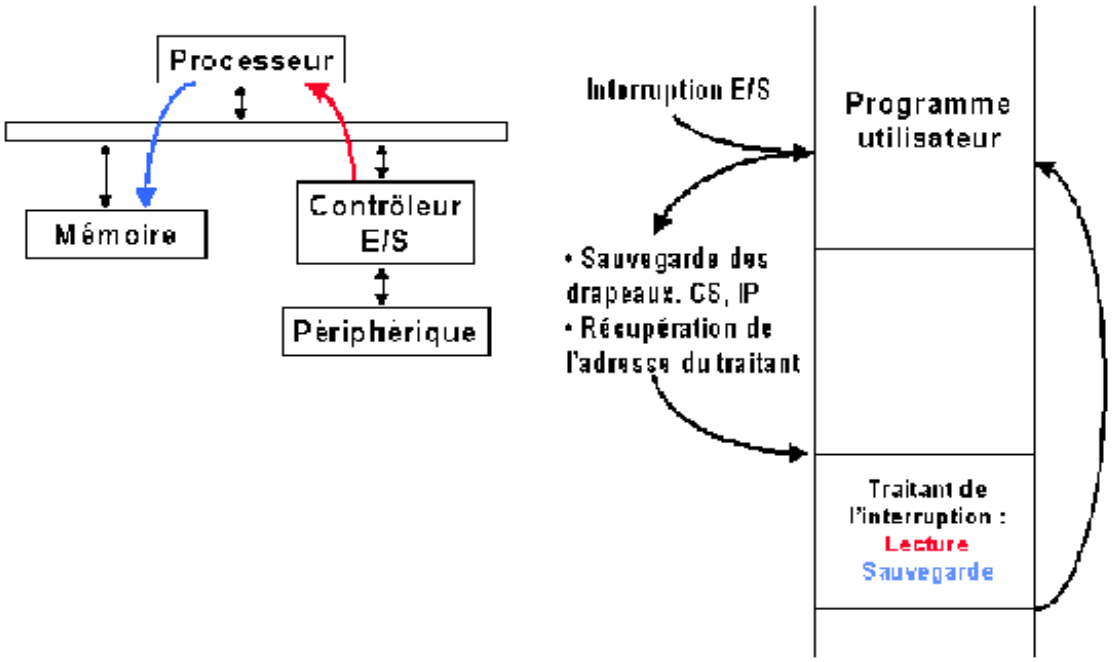
Pourtant, à un moment ou à un autre, vous désirez passer une nouvelle commande en la tapant au clavier. Dans ce cas, c'est le clavier qui va se charger d'informer le microprocesseur que l'utilisateur manifeste sa volonté, laquelle est indiscutablement prioritaire.

Pour cela, le clavier envoie un signal électrique au microprocesseur. Ce dernier le reçoit et comprend qu'on réclame son attention. Le microprocesseur termine l'instruction qu'il était en train d'effectuer, puis il range dans la mémoire tout ce qui concerne le programme en cours afin de pouvoir le retrouver en bon état plus tard. Après quoi, il examine d'où provient la demande d'interruption, puis il appelle le programme spécial appelé le programme «gestionnaire du clavier» dans le cas qui vient d'être évoqué. Enfin, il prévient le clavier : *«OK, tu peux y aller* !».

Le clavier transmet ses informations au microprocesseur. Ce dernier les «traite» immédiatement, c'est-à-dire qu'il exécute vos ordres.

Lorsque c'est terminé, le microprocesseur abandonne cette séquence. Il rappelle le programme précédent dont il avait rangé les éléments en mémoire. I1 le reprend exactement au point où il l'avait abandonné et en poursuit l'exécution.

C'est l'ensemble de cette procédure qu'on appelle une **interruption**. Parce que ce type d'interruption est déclenché par le matériel, on dit encore qu'il s'agit d'une **interruption matérielle**.

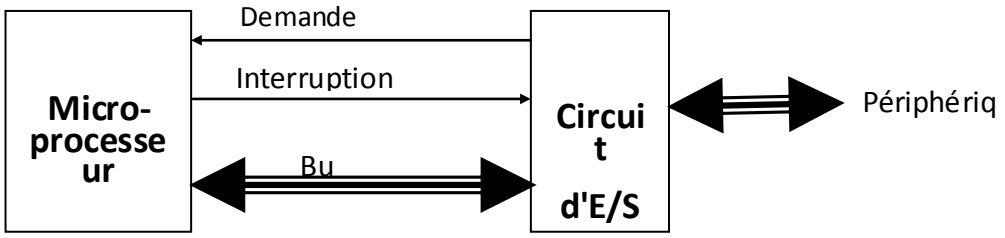


III- Principe de fonctionnement

Une interruption permet ainsi à un événement aléatoire, survenant à un moment quelconque non prévisible, d'être pris en compte par le microprocesseur. Elle offre encore d'autres avantages, en voici un autre exemple avec une imprimante.

Supposons que le microprocesseur envoie un jeu de caractères à l'imprimante; celle-ci demande un certain temps pour les imprimer; pendant ce temps, le microprocesseur ne peut en faire d'autre qu'attendre et se trouve paralysé. Mais si l'on demande aux interruptions d'intervenir, les choses s'arrangent: le jeu de caractères est envoyé à l'imprimante et le microprocesseur poursuit une tâche qu'il doit exécuter. L'imprimante imprime les caractères puis envoie une demande d'interruption au processeur pour lui demander la suite. Le gain en temps d'exécution peut être énorme pour le microprocesseur.

La procédure d'échange par interruption fait appel à des lignes spéciales reliant le périphérique au processeur.

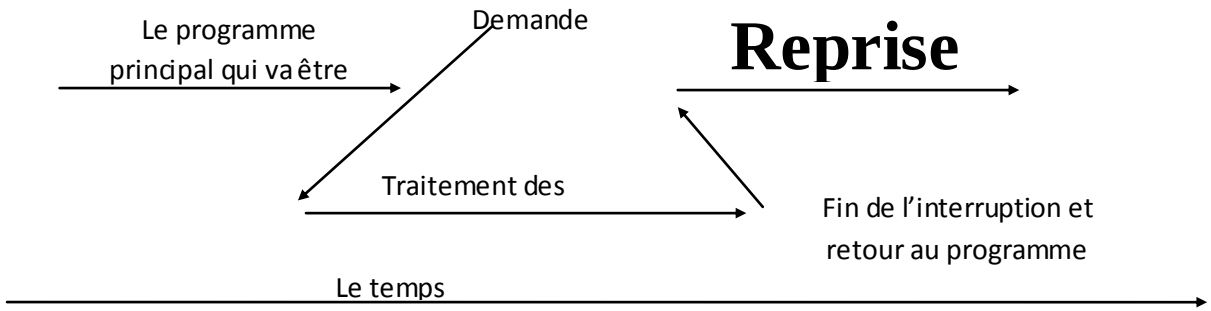


Deux lignes supplémentaires servent à gérer les interruptions: une ligne de demande d'interruption et, en réponse, un ligne d'accord d'interruption.

La procédure est la suivante :

1. Un périphérique alerte son circuit d'interface (d'entrées-sorties, noté E/S) qu'il veut transmettre une information au processeur. (8259A)
2. Le circuit d'E/S envoie une demande d'interruption au processeur, grâce à une ligne spéciale qu'il porte à l'état actif.
3. Le processeur termine l'instruction qu'il est en train d'exécuter. Il range en mémoire « pile » le contenu de tous ses registres relatifs au programme en cours d'exécution (compteur ordinal, accumulateur, index, indicateurs d'états, autres registres, etc.).
4. Le microprocesseur retourne un accusé de réception au circuit d'interface pour l'informer qu'il est prêt. Pour cela, il porte la ligne spéciale Interruption autorisée, ou Interrupt Acknowledge, à l'état actif.
5. Il passe à la séquence de traitement de l'interruption en appelant le programme spécifique nécessaire. Ce programme dépend de la source de l'interruption; c'est le programme de gestion du clavier pour une frappe au clavier, de communication si c'est le modem qui a demandé l'interruption, etc.
6. Il exécute tout ce qu'implique l'interruption.
7. Cela terminé, il peut enfin reprendre l'exécution du programme primitif interrompu. Les lignes spéciales d'interruption ont retrouvé leur état inactif.
8. Il rappelle donc tout ce qu'il avait sauvegardé dans la « pile de sauvegarde ».Celle-ci recharge tous les registres et le processeur retrouve l'environnement qu'il avait abandonné exactement au point où il l'avait laissé.
9. Le traitement de l'interruption est terminé.

   De cette séquence, on constate ainsi que traiter une interruption, c'est tout simplement passer pendant un petit moment à un autre programme, ce qui peut s'illustrer par le dessin suivant :



Par convention, on appelle :

- **Programme principal**, le programme se trouvant en cours d'exécution lors de la demande d'interruption surgit.
- **Sous-programme d'interruption (ou routine d'interruption),** le programme qui doit être mis en œuvre à sa place pour traiter l'interruption.
- **EOI**, pour «End Of Interruption», fin d'interruption, une instruction aussi appelée Retour d'interruption, qui doit obligatoirement se trouver à la fin du programme traitement de l'interruption. Elle signifie au processeur que c'est fini et qu'il peut poursuivre l'exécution du programme principal qui avait été interrompu.

IV- les sous-programmes d'interruption

On peut imaginer que, lorsqu'un périphérique demande une interruption au microprocesseur, il lui envoie simultanément l'adresse à laquelle son programme spécifique est logé.

En pratique et avec les PC, les programmes «gestionnaires de périphériques» sont chargés grâce aux fichiers système, les CONFIG.SYS et AUTOEXEC.BAT. C'est donc le système qui sait où il les a logés, à partir de quelle adresse chacun d'eux se trouve. Reste à dresser une sorte de carte de ces adresses, pour le cas où une interruption surgirait.

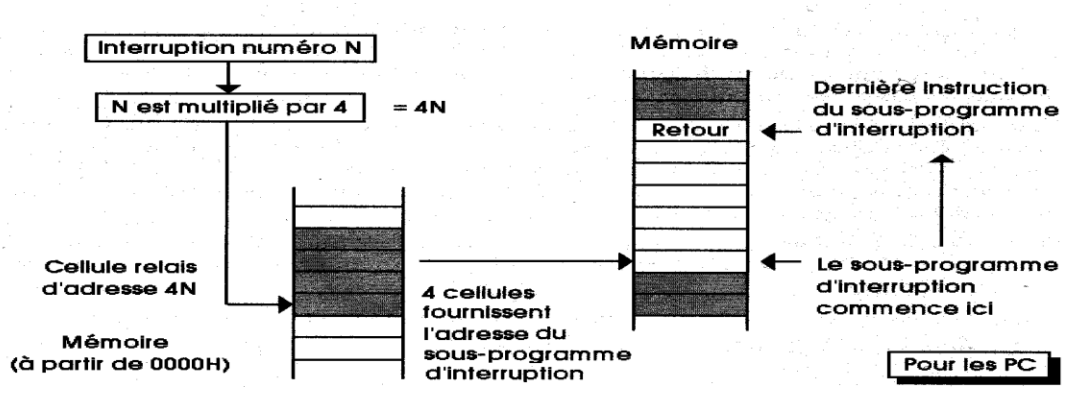
Cette table s'appelle la table des vecteurs d'interruptions. Dans les PC, elle est systématiquement et obligatoirement logée à partir de l'adresse 00000 de la mémoire centrale. En pratique, 1 Ko (soit 1024 octets) sont réservés pour cette table, laquelle s'étend ainsi de 00000 à 003FF en hexadécimal. Sachant que 4 octets sont réservés pour une adresse d'interruption, cette table peut en contenir 256 adresses.

Or, un PC classique ne dispose que de 15 lignes d'interruptions déclenchées par des périphériques et numérotées de 0 à 15, ce qui fait que les 15 premiers groupes de 4 octets d'adresses leur sont réservés.

Voici alors ce qui se passe:

1. Un périphérique demande une interruption.
2. Le circuit d'interface la transmet à un circuit spécialisé appelé gestionnaire des interruptions.
3. Ce circuit «programmable» d'interruptions transmet cette demande au microprocesseur en l'accompagnant du numéro de l'interruption.
4. Le processeur termine l'instruction en cours de traitement et incrémente le compteur ordinal, de façon à pointer l'instruction suivante (qui ne sera pas exécutée dans l'immédiat).
5. Il sauvegarde tous ses registres dans la pile.
6. Il envoie un accusé de réception vers le périphérique.
7. Il **multiplie par 4** le numéro de l'interruption pour calculer une adresse en mémoire (ceci ne s'applique qu'aux PC « processeur INTEL ».
8. Il lit en mémoire, à partir de cette adresse, les 4 octets qui vont le brancher sur le bon programme de traitement de l'interruption.
9. Il introduit cette nouvelle adresse dans son compteur ordinal.
10. Il commence l'exécution du programme de traitement de l'interruption à partir de cette adresse.
11. L'interruption traitée, il rencontre une instruction de Retour dans le sous-programme d'interruption qui se termine là dessus.
12. Le microprocesseur récupère dans la pile de sauvegarde le contenu de ses registres sauvegardés et les recharge.
13. Il reprend et poursuit l'exécution du programme principal qui avait été interrompu.

    Ce qui est important dans ce processus, c'est le mode d'adressage du sous-programme d'interruption. C'est un mode indirect par mémoire dont le fonctionnement est rappelé dans la figure ci-dessous, mais dans le seul cas précis des PC.



Mode d'adressage indirect par mémoire des sous-programmes d'interruption.

Ce mode de calcul est typique des PC.

Retenez qu'on appelle :

- **Table des vecteurs d'interruptions :** la table d'adressage indirect fournissant l'adresse des sous-programmes d'interruption.
- **Vecteurs d'interruptions** :ces adresses.
- **Interruptions vectorisées** :ce mode d'adressage des interruptions.

V- La notion de la priorité

IL est accordé aux interruptions un degré de priorité qui fait que l'interruption la plus prioritaire passe avant les autres, mises en attente.

Supposons qu'une première interruption soit prise en compte par le microprocesseur. Alors qu'elle est en cours de traitement, une seconde demande d'interruption se manifeste. Deux cas sont possibles :

1. La seconde interruption est moins prioritaire. Elle attendra que la première se termine pour être prise en compte.

2. La seconde interruption est plus prioritaire. Dans ce cas, elle interrompt à son tour le traitement de la première interruption, tout comme celle-ci avait interrompu le programme principal. Cet «emboîtage» suppose que:

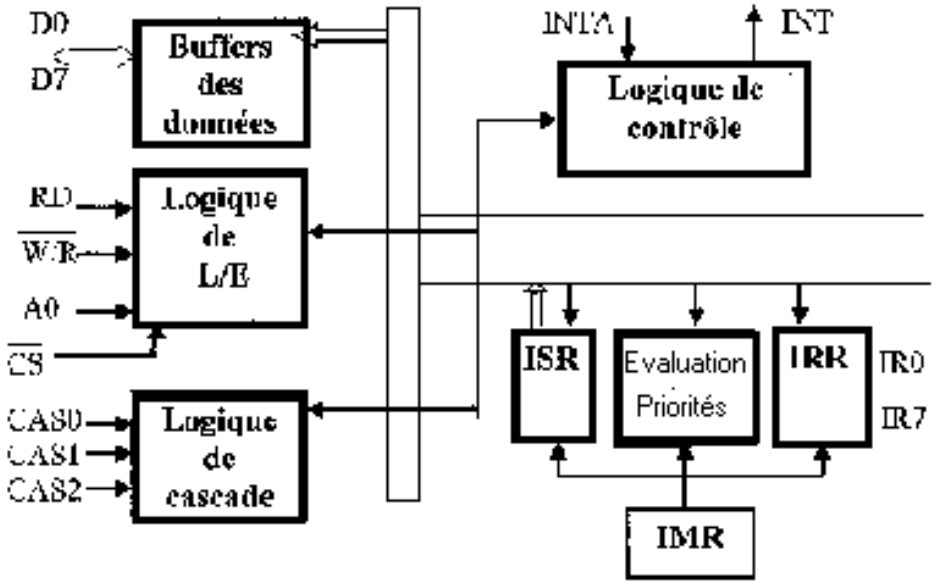
- Le microprocesseur sauvegarde ses registres dont le contenu, cette fois, dépend du traitement de la première interruption.
- Cette sauvegarde s'ajoute, dans la pile, à la précédente, celle du programme principal.
- Le microprocesseur traite la seconde interruption plus prioritaire.
- Lorsque ce traitement est terminé, il récupère le contenu des registres de la première interruption et revient au traitement de cette première interruption.
- Quand le traitement de la première interruption est, à son tour, terminé, il récupère le contenu des registres du programme principal et revient au programme principal.

VI- Contrôleur programmable d'interruptions PIC

Le contrôleur programmable d'interruptions. est appelé PIC, ce qui provient de «Programmable Interrupt Controler». Sa référence de circuit est **8259**. Il est apparu avec le premier PC et a suivi la lignée, devenant le **8259A** avec le premier AT puis le **82C89A** avec le microprocesseur 486.

C'est un circuit à nombre de broches limité, puisqu'il n'en possède que 28. Parmi elles, seules 8 broches sont réservées aux entrées d'interruptions.

- **INT** est le demande d'interruption émise par le circuit vers le microprocesseur.
- **INTA** est la réponse du microprocesseur et signifie «Interrupt Acknowledge», soit acquisition d'interruption.



Registres du PIC et câblage du circuit

Les registres caractéristiques du 8259 sont :

1. Registre de demande d'interruptions **IRR** (*«Interrupt Request Register»*). Il reçoit les huit lignes de demande d'interruptions et les mémorise.
2. Résolveur de priorité: il sélectionne l'interruption de plus haute priorité dans IRR. ,
3. Registre d'interruption en service **ISR** *«<In Service Register»)* : il spécifie l'interruption, active lorsque plusieurs 8259 sont emboîtés.
4. Registre de masquage des interruptions **IMR** (*«Interrupt Mask Register»*) : il enregistre les interruptions dont la prise en compte est interdite à un instant donné..

Le 8259 est Considéré comme un périphérique par le processeur et adressé comme tel via des commandes d'entrée-sortie (I/O) IN et OUT.

VII- Masquage des interruptions

Lorsque certaines séquences critiques d'un programme sont en cours d'exécution, on peut vouloir interdire toute prise en compte d'une interruption quelconque qui risquerait de le perturber gravement. Cette interdiction s'appelle le masquage des interruptions.

La méthode courante consiste à positionner un bit d'état, ou plusieurs bits d'état dans le mot d'état du microprocesseur. Il s'agit des bits de masquage. On les bascule du mode inactif au mode actif. Dès lors, le microprocesseur refuse toute prise en compte d'interruption. Le

positionnement de ces bits d'état se fait par logiciel, via une commande insérée au bon endroit dans un programme. La séquence de principe du programme est ainsi la suivante :

1. Positionnement des indicateurs de masquage des interruptions pour les interdire.
2. Exécution d'une séquence critique du programme.
3. Démasquage des interruptions qui peuvent de nouveau librement intervenir.

Une autre méthode de masquage consiste à verrouiller le codeur de priorités ou le circuit de gestion des interruptions s'il a été prévu pour cette fonction.

IIX- Types d'interruptions

On distingue plusieurs types d'interruptions, et tout d'abord :

1.Les interruptions matérielles: déclenchées par un périphérique.

2.Les interruptions logicielles: l'interruption est introduite sous forme d'une commande dans un programme. Vous pouvez définir vos propres interruptions.

La table des vecteurs d'interruptions offre de la place pour 256 interruptions différentes dans les PC. Or, il existe 16 interruptions matérielles numérotées de 0 à 15, l'interruption 2 étant de toute façon indisponible en raison du cascadage des circuits PIC. Par conséquent, les autres interruptions, de 16 à 255, sont réservées aux interruptions logicielles.

Le BIOS des micro-ordinateurs PC et leur système d'exploitation font généralement appel à des interruptions logicielles pour gérer une quantité de fonctions variées. Avec les PC, on parle ainsi fréquemment de l'interruption programmée INT 21H qui donne accès à de nombreuses fonctions.

Les interruptions matérielles se subdivisent encore en plusieurs catégories :

1. Une interruption de plus haute priorité, non masquable pour cette raison. Elle s' appelle interruption **NMI**, pour «Non Maskable Interrupt». Le microprocesseur dispose d'une entrée spéciale pour ce type d'interruption, qui s'ajoute par conséquent aux autres.
2. Les 15 interruptions **masquables** courantes.
3. Une interruption encore supérieure en priorité appelée **RESET** et servant à la réinitialisation du processeur.

On appelle encore ces interruptions matérielles des interruptions câblées, ou des interruptions masquables.

LES INTERRUPTIONS MATERIELLES DANS LES PC	
Numéro de l'interruption	**Origine**
0	Les clics du timer (le chrono système)
1	Clavier
2	Cascade du PIC
3	Port de communication série COM2 et COM4
4	Port de communication série COM1 et COM3
5	Port parallèle LPT2
6	Disquette
7	Port parallèle LPT1 (imprimante)
8	Horloge-calendrier en temps réel
9	Redirection IRQ2-Ecran VGA
10	Réservé ou libre
11	Réservé ou libre
12	Réservé ou libre
13	Coprocesseur mathématique
14	Disque dur
15	Réservé ou libre

IX- Programmation des interruptions

Procédure à suivre pour l'écriture d'un programme qui utilise les mécanismes d'interruption

L'écriture d'un programme qui manipule les interruptions doit contenir cinq parties principales:

1- écriture d'une routine d'interruption qui doit se terminer par l'instruction IRET.

2- sauvegarde de l'ancien vecteur d'interruption.

3- chargement du nouveau vecteur d'interruption

4- Initialisation et écriture du programme principal

5- rechargement de l'ancien vecteur d'interruption pour remettre le système dans sans état initial et retour au DOS

l'architecture globale sera donc:

```
;-------------------------------------------------
data segment
        ;déclaration des variables
data ends
;-------------------------------------------------
code segment
        Assume cs:code, ds:data
org 100h
 debut:
; sauvegarde de l'ancien vecteur d'interruption
;  Chargement du nouveau vecteur d'interruption
;  initialisation et écriture du programme principal
;rechargement de l'ancien vecteur d'interruption pour remettre le système dans sont état initial
; retour au DOS
; routine d'interruption
 routine :cli
              .
            sti
            iret
code ends
end debut
```

✓ pour la lecture d'un vecteur d'interruption vous pouvez utiliser la fonction 35h

✓ pour l'écriture d'un nouveau vecteur d'interruption (installation), utilisez la fonction 25h

Exemple : Programmation de l'interruption Horloge :

```
data segment
  vectint equ  1ch
data ends

code segment
  assume cs:code,ds:data
 org 100h
debut:mov ah,35h
     mov al,vectint
     int 21h
     push es
          push bx          ;  sauvegarde de l'ancien vecteur d'interruption
     mov ax,code
     mov ds,ax
     lea dx,routine
```

```
    mov ah,25h
    mov al,vectint
    int 21h          ;Chargement du nouveau vecteur d'interruption
    mov cl,30h
    mov ch,0
    sti
boucle: cmp cl,39h   ;initialisation et écriture du programme principal
    jle boucle
    ;rechargement de l'ancien vecteur d'interruption pour remettre le système dans sont état initial

          pop dx
    pop ds
    mov ah,25h
    mov al,vectint
    int 21h

    mov ah,4ch    ; retour au DOS
    int 21h

; routine d'interruption

routine proc far
        cli
        push ds
        push bx
        push ax
        mov ah,20h
        mov dx,0020h
        out dx,al
        mov ah,02h
        mov dl,cl
        int 21h
        inc cl
sortir:  pop ax
        pop bx
        pop ds
        sti
       iret
 routine endp
code ends
end debut
```