

ARCHITECTURE DES MICROPROCESSEURS

Proramme

- Première partie
 - algèbre de Boole
 - circuits logiques
- Deuxième partie
 - L'Unité Centrale du microprocesseur
 - La Mémoire Centrale du microprocesseur
 - Les Périphériques et interfaces associés au microprocesseur
 - Le logiciel de base

Proramme

- Première partie
 - algèbre de Boole
 - circuits logiques
- Deuxième partie
 - L'Unité Centrale du microprocesseur
 - La Mémoire Centrale du microprocesseur
 - Les Périphériques et interfaces associés au microprocesseur
 - Le logiciel de base

Deuxième partie

Circuits logiques à mémoire

Sujets abordés

- L'Unité Centrale du microprocesseur
- La Mémoire Centrale du microprocesseur
- Les Périphériques et interfaces associés au microprocesseur
- Le logiciel de base

Deuxième partie

Circuits logiques à mémoire

Sujets abordés

- **L'Unité Centrale du microprocesseur**
- La Mémoire Centrale du microprocesseur
- Les Périphériques et interfaces associés au microprocesseur
- Le logiciel de base

Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

Deuxième partie

Unité centrale du microprocesseur introduction

- **But** : mettre en évidence les principaux constituants d'une **Unité Centrale**
- **Rappels** :
 - ❖ ALU ou UAL (Unité Arithmétique et Logique) permettant de réaliser différentes *opérations* grâce à un *décodeur*
 - ❖ MEMOIRE (circuit de mémoire) et en particulier les lignes RD/WR associées, CS (chip select), les bus d'adresses et de données : ce circuit mémoire permet de stocker (W) de l'information en vue d'un usage futur (R).

Remarques :

→ l'UAL est un composant de l'UC

→ le circuit de mémoire peut être considéré comme un périphérique particulier

Deuxième partie

Unité centrale du microprocesseur introduction

- Problème :
exécutions des instructions *évoluées* à l'aide d'instructions *machine* élémentaires

```
C := A + B ;           {1}
if C > 0 then      action_1 ;       {2}
                    action_2 ...
...
...
```

- Comment est représenté chaque identificateur (A, B, C) avant la phase d'exécution ?
→ par une *zone de mémoire d'adresse connue* que nous noterons &A, &B ou &C (arbitrairement et symboliquement)

Deuxième partie

Unité centrale du microprocesseur introduction

```
C := A + B ; {1}
if C > 0 then      action_1 ; {2}
                    action_2 ...
...
...
```

- Pour coder l'instruction {1} on peut imaginer :
 - une seule instruction
add &A, &B, &C ; C←[&A]+[&B]
 - trois instructions différentes utilisant explicitement l'ACCU (zone mémoire interne à l'UC: registre accumulateur) :
load &A ; ACCU ← [&A]
add &B ; ACCU ← [ACCU]+[&B]
store &C ; C ← [ACCU]

Deuxième partie

Unité centrale du microprocesseur introduction

$C := A + B ;$

if $C > 0$ then

...

{1}

action_1 ;

{2}

action_2 ...

- Codage de l'instruction {2} :

21 load &C ; ACCU \leftarrow [&C]

{2.1}

22 comp 0 ; [ACCU] :: 0

{2.2}

23 jg 47 ; aller_a_l'adresse 47 si [ACCU]>0 {2.3} jg : jump if greater

24 ...

{2.1} : charge dans ACCU le contenu de C

{2.2} : compare [ACCU] et 0 et positionne un indicateur

(l'indicateur correspondant est mis à VRAI et les autres à FAUX) selon que:

[ACCU] >0 : greater_flag \leftarrow VRAI

[ACCU] <0 : lower_flag \leftarrow VRAI

[ACCU] =0 : zero_flag \leftarrow VRAI

{2.3} : si greater_flag est VRAI, alors la prochaine instruction exécutée sera l'instruction 47, sinon l'instruction 24.

Deuxième partie

Unité centrale du microprocesseur introduction

- Codage de l'instruction {2} :

21	load &C ; ACCU \leftarrow [&C]	{2.1}
22	comp 0 ; [ACCU] :: 0	{2.2}
23	jg 47 ; aller_a_l'adresse 47 si [ACCU]>0	{2.3} jg : jump if greater
24	...	

Remarques :

Mise en évidence de

*une zone spéciale (registre) contient l'adresse de l'instruction à exécuter par la CPU (UC)

*elle est contenue dans le compteur ordinal ou PC.

*Une instruction telle que "jg" peut modifier implicitement la valeur de ce registre.

C := A + B ;

{1}

if C > 0 then

action_1 ;

{2}

action_2 ...

...

Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

Deuxième partie

Unité centrale du microprocesseur

étude d'un cas d'école → présentation du microprocesseur fictif

- *principales caractéristiques :*

mots de 8 bits pour instructions et données

adresses sur 6 bits ($\Rightarrow 64$ mots adressables)

un registre ACCU de 8 bits

un jeu de 4 instructions de format général :

- **Jeu d'instructions :**

8	7	6	5	4	3	2	1
Code op (2 bits)	Adresse opérande (6 bits)						

forme mnémonique **(code)** **signification**

load M **(00)** **ACCU \leftarrow [M]**

store M **(01)** **M \leftarrow [ACCU]**

add M **(10)** **ACCU \leftarrow [ACCU]+[M]**

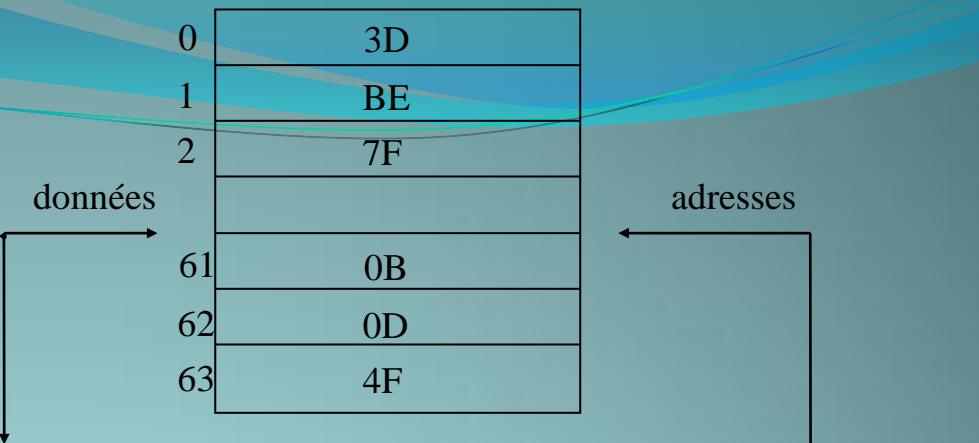
sub M **(11)** **ACCU \leftarrow [ACCU]-[M]**

Deuxième partie

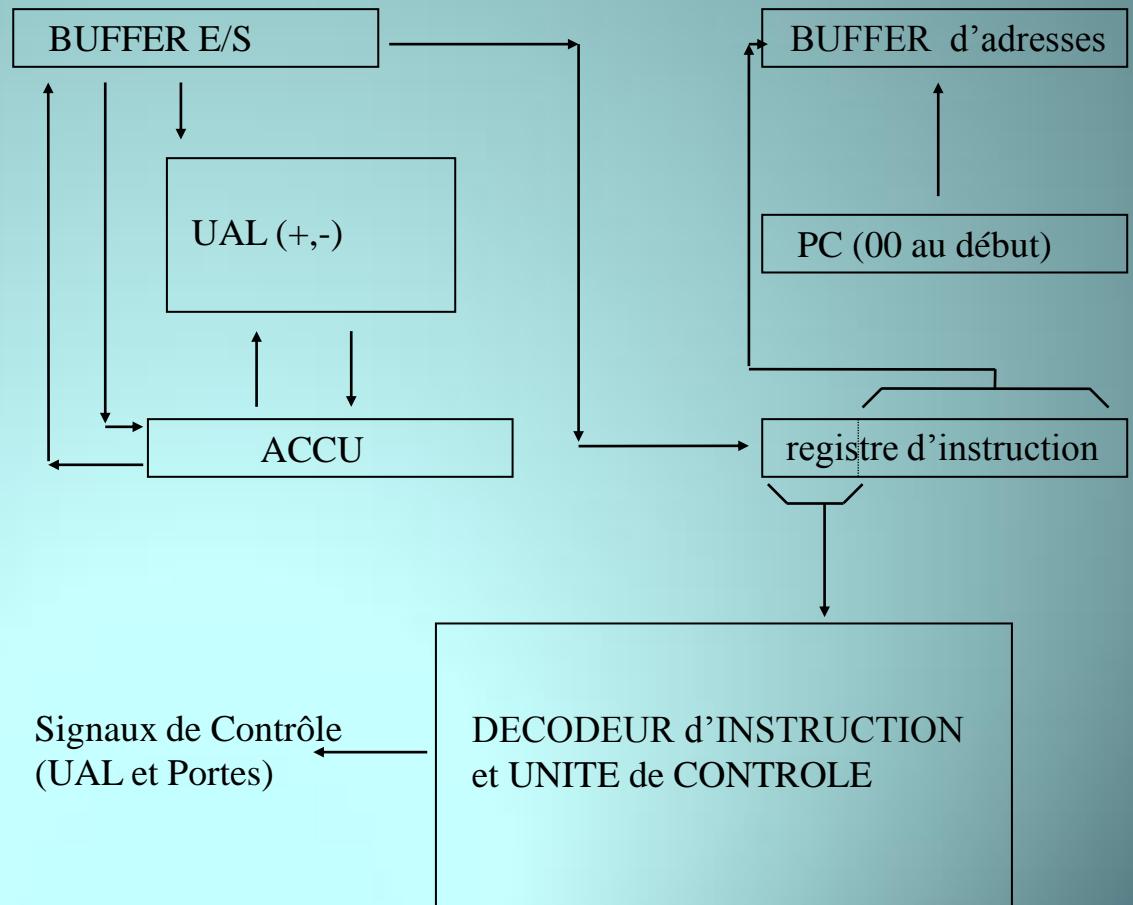
Unité centrale du microprocesseur

étude d'un cas d'école

→ exécution d'un programme



la suite des opérations ...?



Deuxième partie

Unité centrale du microprocesseur

étude d'un cas d'école → exécution des instructions

Remarques :

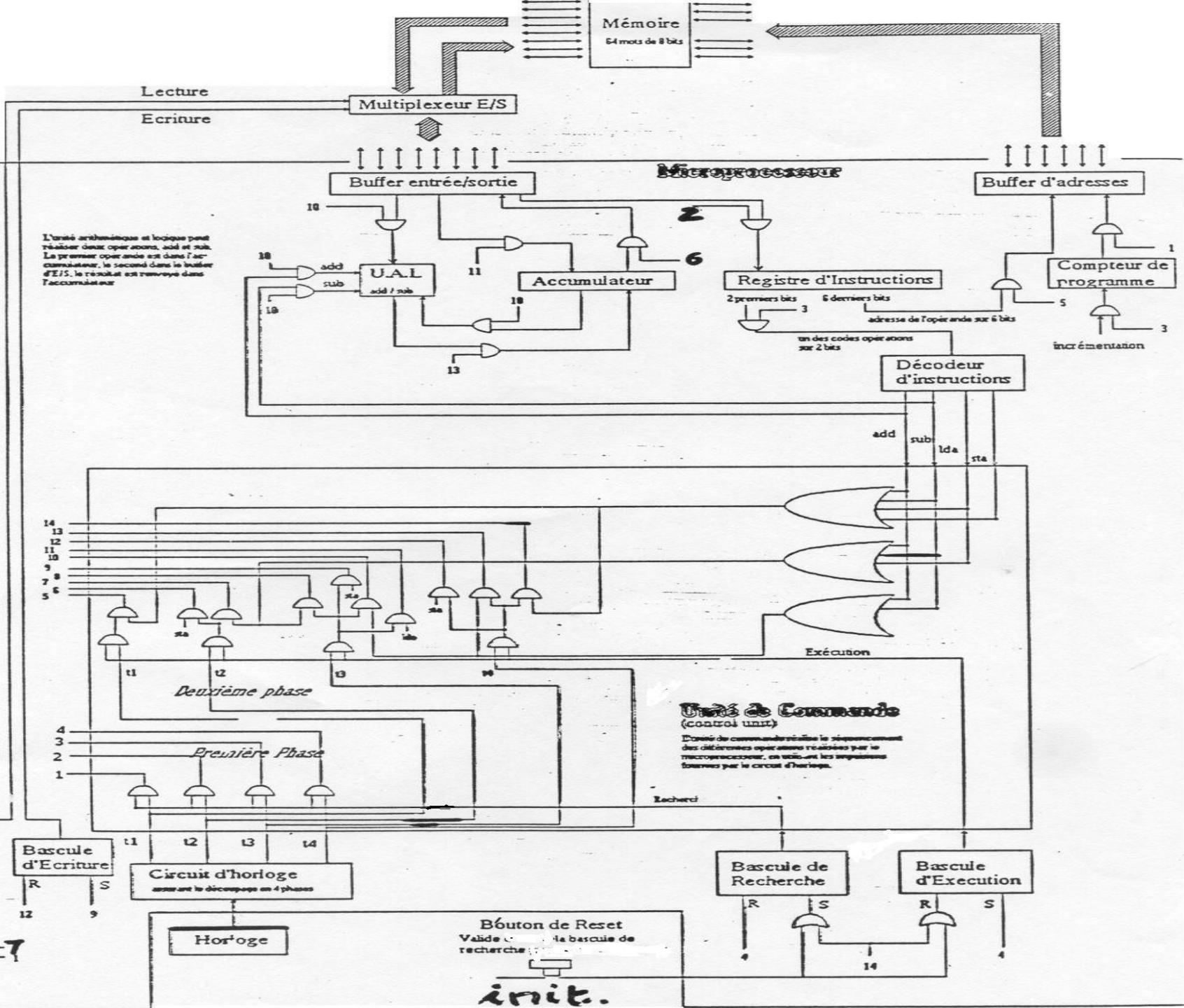
❖ le cycle d'exécution d'une instruction se divise en deux phases :

- "fetch" ou recherche d'instruction (et chargement dans le registre d'instruction)
- exécution proprement dite

❖ Modes de fonctionnement via des données

- ECRITURE : adresse dans buffer adresse
 - ✓ donnée dans buffer d'E/S → mémoire
 - ✓ valider bascule ECRITURE
- LECTURE : adresse dans buffer adresse
 - ✓ valider bascule LECTURE
 - ✓ donnée dans mémoire → buffer d'E/S

Note : un buffer est un registre lié à un bus



Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

classification générale des instructions :

- C1 : mouvements de données entre registres et mémoire centrale
`mov ax, mot ; $ax \leftarrow [MOT]$ (16 bits)`
- C2 : instructions arithmétiques et logiques : +, -, *, /, in(dé)crémentation , ET, OU, décalage, rotation, comparaison, complémentation, ...
`add bl, cl ; $bl \leftarrow [bl] + [cl]$ (8bits)`
- C3 : instructions de contrôle ou branchement conditionnel ou inconditionnel ...
`jmp LABAS ; $PC \leftarrow \text{adresse}(LABAS)$`
- C4 : instructions de gestion de l'environnement : mise à jour de la pile (*) , opérations d'E/S, positionnement / lecture des masques de bits d'interruption (*) et indicateurs (bits d'état) ...
`out PORT, al ; $PORT(\text{d'E/S}) \leftarrow [al]$`

(*) vu plus tard

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Les opérandes:

- *Exemple* : pour effectuer une addition, on a besoin de deux opérandes au moins (opérateur binaire) et il faut aussi prévoir où mettre le résultat ...
- Le *nombre des opérandes* peut être variable :
 - hlt (halt) est une instruction sans opérande
 - jmp adr_symb : 1 opérande
 - mov reg1, reg2 : 2 opérandes etc...
- Un *opérande* est localisé à une adresse précise (explicite ou implicite) et on appelle *mode d'adressage* une méthode permettant de calculer l'adresse effective d'un opérande

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Les opérandes → modes d'adressage

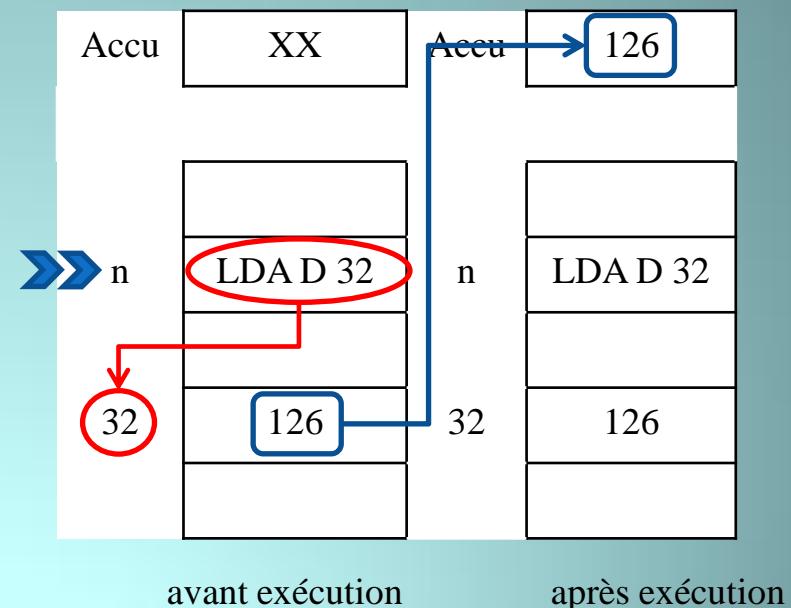
- Trois *modes d'adressage de base* :
- (*LDA : Load A : charger accumulateur*)

M1 : adressage direct (LDA D)

M2 : adressage immédiat (LDA IM)

M3 : adressage indirect (LDA IND)

M4 : adressage indexé (LDA X)



Adressage direct

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Les opérandes → modes d'adressage

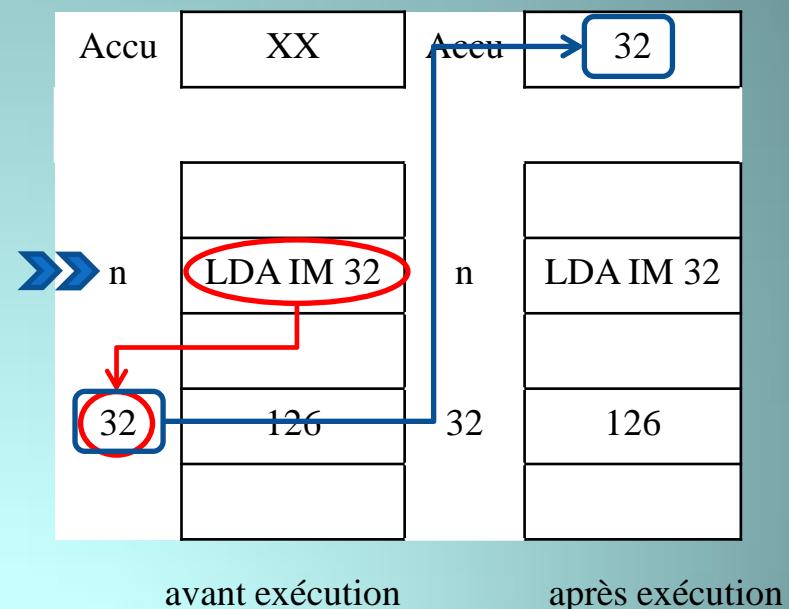
- Trois *modes d'adressage de base* :
- (*LDA : Load A : charger accumulateur*)

M1 : adressage direct (LDA D)

M2 : adressage immédiat (LDA IM)

M3 : adressage indirect (LDA IND)

M4 : adressage indexé (LDA X)



Adressage immédiat

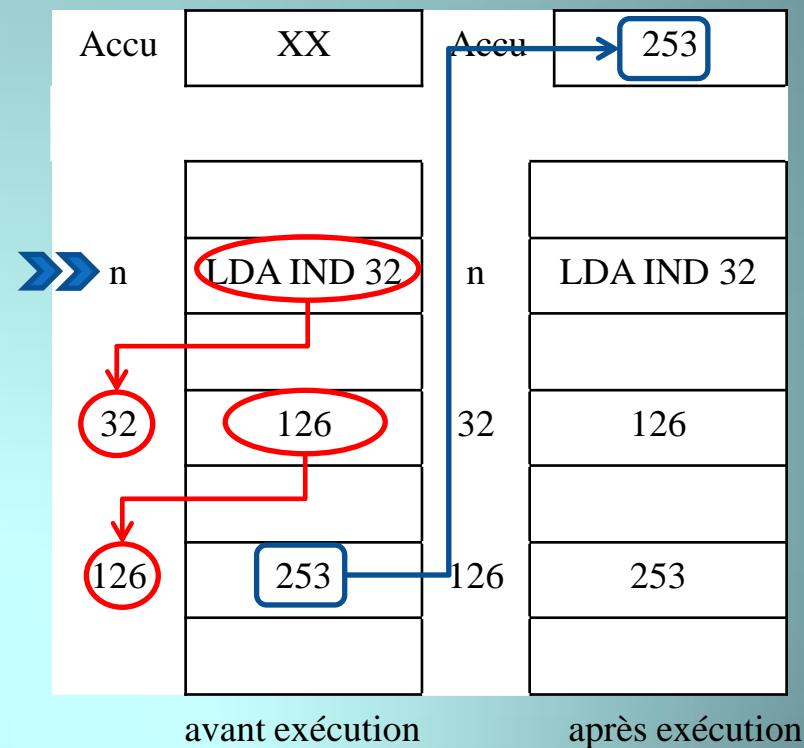
Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Les opérandes → modes d'adressage

- Trois *modes d'adressage de base* :
- (*LDA : Load A : charger accumulateur*)
M1 : adressage direct (LDA D)
M2 : adressage immédiat (LDA IM)
M3 : adressage indirect (LDA IND)
M4 : adressage indexé (LDA X)



Adressage indirect

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Les opérandes → modes d'adressage

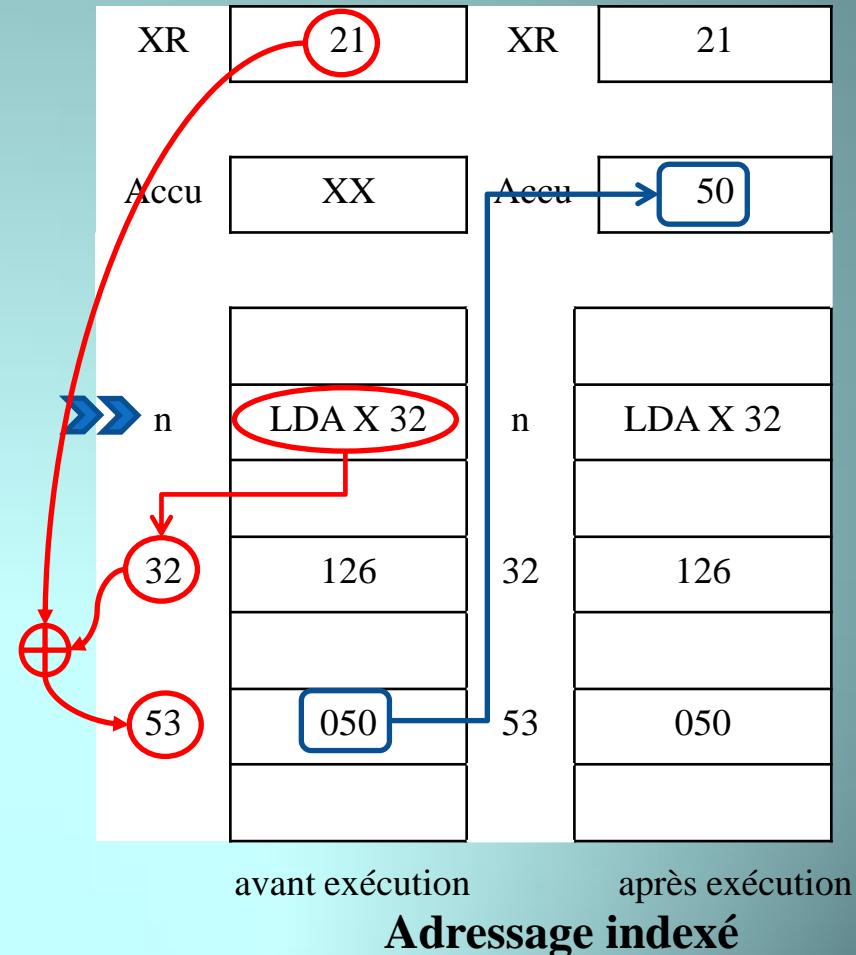
- Trois *modes d'adressage de base* :
- (*LDA : Load A : charger accumulateur*)

M1 : adressage direct (LDA D)

M2 : adressage immédiat (LDA IM)

M3 : adressage indirect (LDA IND)

M4 : adressage indexé (LDA X)



Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

- Exemples d'utilisation des modes d'adressage

Codes
mnémoniques des
instructions

SOURCE
SOURCE+1
...
SOURCE+i
...
SOURCE+n-1

DEST
DEST+1
...
DEST+i
...
DEST+n-1

transfert

LDA	Load A – charger accumulateur
STA	Store A – ranger accumulateur
ADDI	Addition avec opérante immédiat
INC	Incrémantation
SUB	Subtraction – Soustraction
BNE	Branche if not equal – branchement si non égalité avec 0
LDXI	Load Index immédiat – chargement index avec opérande immédiat
INX	Incrémantation de l'index
CPXI	Comparaison de l'index avec opérande immédiat
	à côté d'une adresse spécifie l'indirection
X	à côté d'une adresse spécifie l'indexation
*	Position courante du compteur ordinal

LDA SOURCE <charger SOURCE dans l'accumulateur>
STA DEST <ranger l'accumulateur dans la mémoire DEST>
LDA SOURCE + 1 <charger SOURCE+1 dans l'accumulateur>
STA DEST + 1 <ranger l'accumulateur dans DEST+1>
LDA SOURCE + 2 id
STA DEST + 2 id
..... → trop lourd quand n est grand

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>

Accumulateur

M1

M2

M3

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE VAL_0

SOURCE + n - 1 VAL_n

DEST

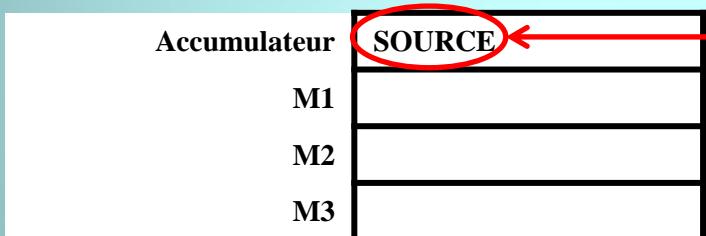
DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
 suite du programme <suite du programme si égalité>



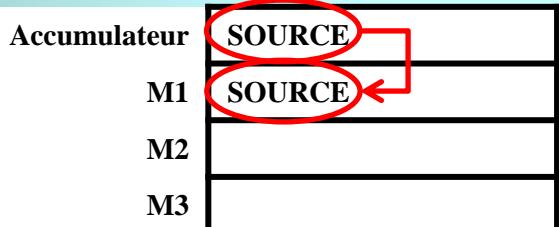
000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme
	SOURCE → VAL_0
	SOURCE + n - 1 → VAL_n
	DEST
	DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>

Accumulateur **SOURCE + n**

M1 SOURCE

M2

M3

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE VAL_0

SOURCE + n - 1 VAL_n

DEST

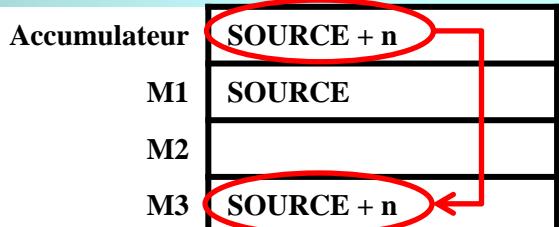
DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

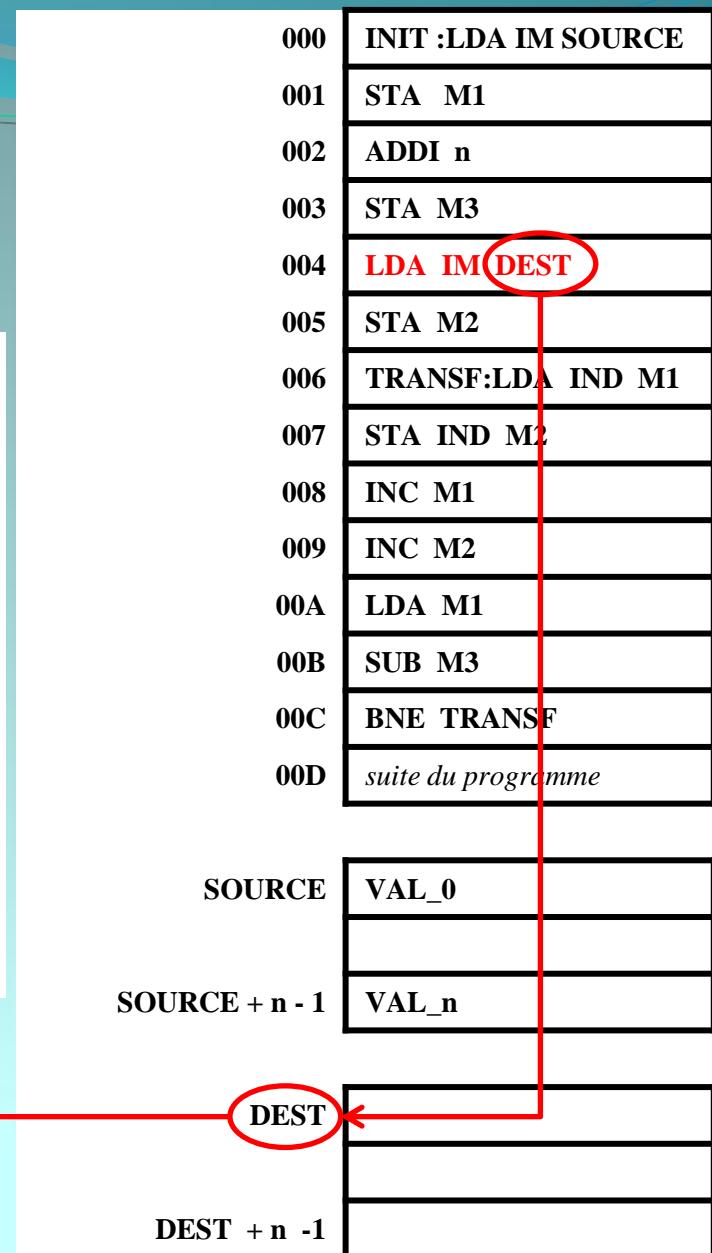
SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
 suite du programme <suite du programme si égalité>

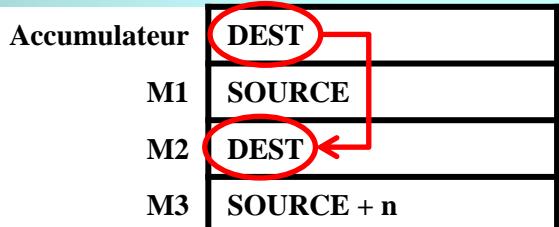


Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme

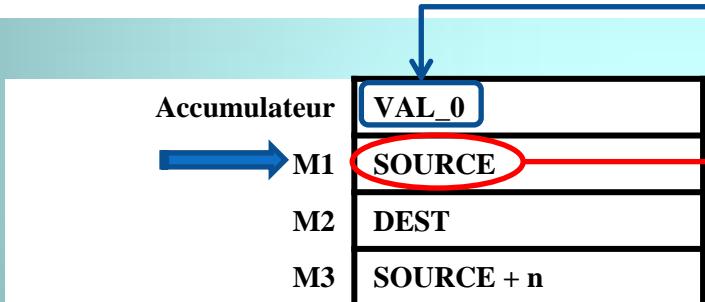
SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	
DEST + n - 1	

Deuxième partie

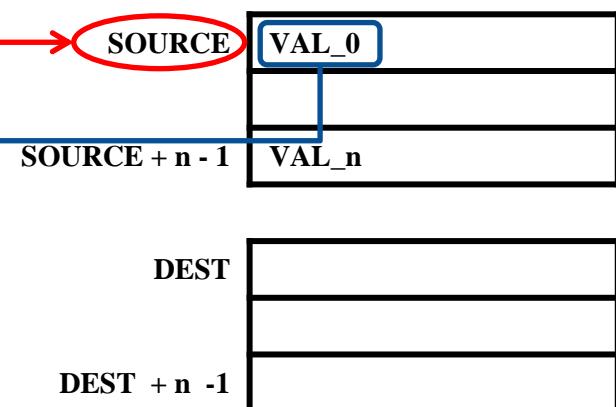
Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
 suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme

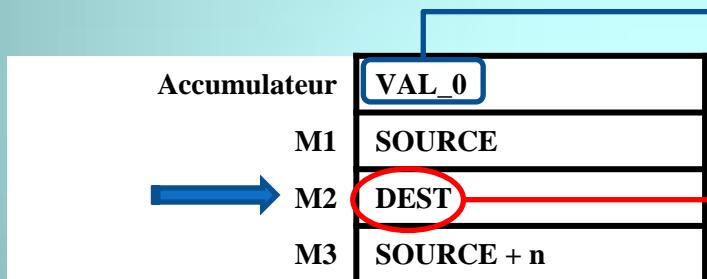


Deuxième partie

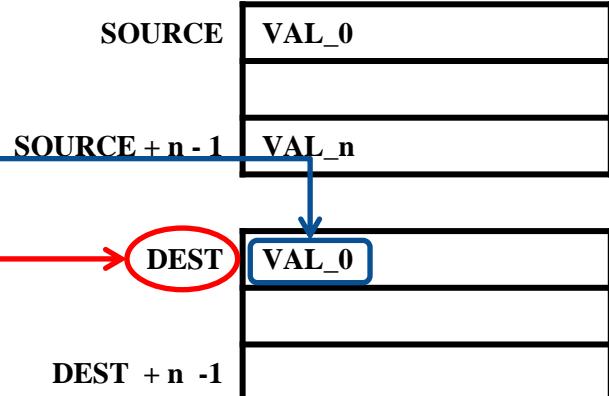
Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
 suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme

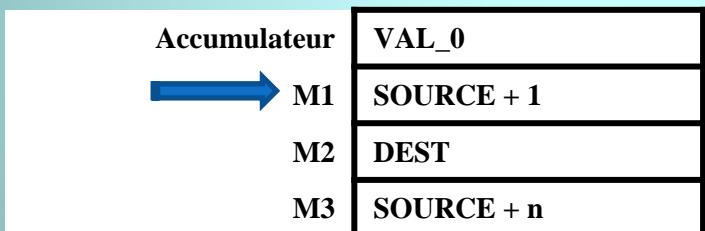


Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

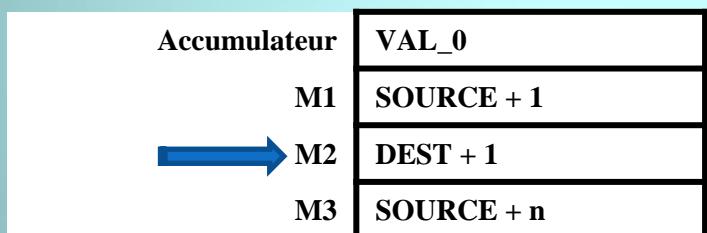
SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	VAL_0
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0 >
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

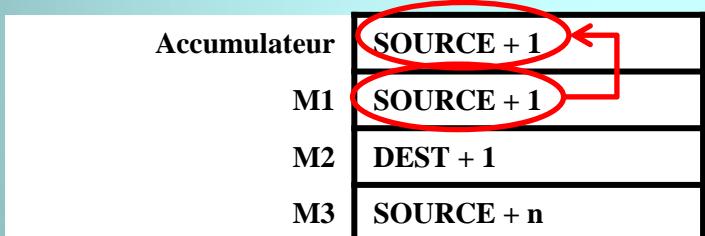
SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	VAL_0
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0>
suite du programme <suite du programme si égalité>



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	VAL_0
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0>
suite du programme <suite du programme si égalité>

Accumulateur	SOURCE+1- SOURCE-n = 1-n
M1	SOURCE + 1
M2	DEST + 1
M3	SOURCE + n

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE	VAL_0
SOURCE + n - 1	VAL_n

DEST	VAL_0
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <**M1 contient l'adresse de début SOURCE**>
 ADDI n <addition immédiate avec n>
 STA M3 <**M3 contient l'adresse de fin de SOURCE**>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <**M2 contient l'adresse de début de DEST**>
TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0>
suite du programme <suite du programme si égalité>

Accumulateur	1 - n = 0 ?
M1	SOURCE + 1
M2	DEST + 1
M3	SOURCE + n

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	VAL_0
DEST + n - 1	

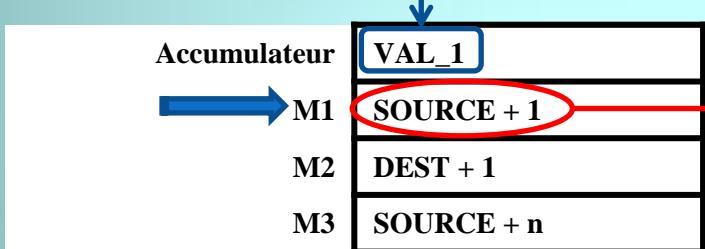
Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0>
 suite du programme <suite du programme si égalité>

Si $1-n \neq 0$



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme

SOURCE

VAL_0
VAL_1
VAL_n

DEST

VAL_0
DEST + n - 1

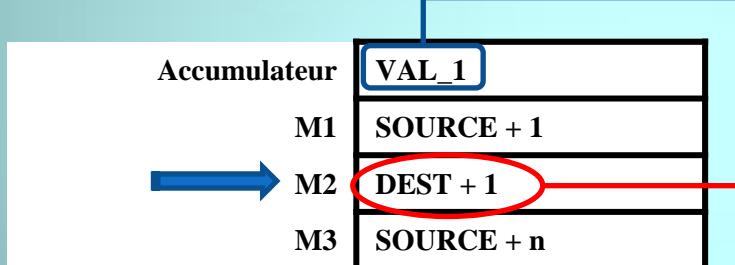
Deuxième partie

Unité centrale du microprocesseur

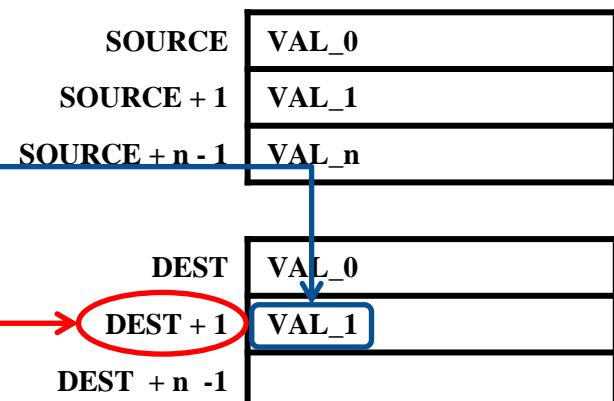
INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <M1 contient l'adresse de début SOURCE>
 ADDI n <addition immédiate avec n>
 STA M3 <M3 contient l'adresse de fin de SOURCE>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
 BNE TRANSF <branchement en TRANSF si non égalité avec 0>
 suite du programme <suite du programme si égalité>

Si $1-n \neq 0$



000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	suite du programme



Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
STA M1 <M1 contient l'adresse de début SOURCE>
ADDI n <addition immédiate avec n>
STA M3 <M3 contient l'adresse de fin de SOURCE>
LDA IM DEST <charger accu avec adresse de début destination>
STA M2 <M2 contient l'adresse de début de DEST>

TRANSF :LDA IND M1 <chargement indirect avec M1>
STA IND M2 <rangement indirect avec M2>
INC M1 <+1 dans M1>
INC M2 <+1 dans M2>
LDA M1 <chargement direct avec M1>
SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0>
suite du programme <suite du programme si égalité>

Accumulateur

M1	SOURCE + 2
M2	DEST + 2
M3	SOURCE + n

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE VAL_0

SOURCE + 1 VAL_1

SOURCE + n - 1 VAL_n

DEST VAL_0

DEST + 1 VAL_1

DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDA IM SOURCE <charger accu avec adresse de début de SOURCE>
 STA M1 <**M1 contient l'adresse de début SOURCE**>
 ADDI n <addition immédiate avec n>
 STA M3 <**M3 contient l'adresse de fin de SOURCE**>
 LDA IM DEST <charger accu avec adresse de début destination>
 STA M2 <**M2 contient l'adresse de début de DEST**>

TRANSF :LDA IND M1 <chargement indirect avec M1>
 STA IND M2 <rangement indirect avec M2>
 INC M1 <+1 dans M1>
 INC M2 <+1 dans M2>
 LDA M1 <chargement direct avec M1>
 SUB M3 <différence adresse courante – adresse fin de SOURCE>
BNE TRANSF <branchement en TRANSF si non égalité avec 0>
suite du programme <suite du programme si égalité>

Si $2-n = 0$

Accumulateur

M1	2 - n = 0 ?
M2	SOURCE + 2
M3	DEST + 2
	SOURCE + n

000	INIT :LDA IM SOURCE
001	STA M1
002	ADDI n
003	STA M3
004	LDA IM DEST
005	STA M2
006	TRANSF:LDA IND M1
007	STA IND M2
008	INC M1
009	INC M2
00A	LDA M1
00B	SUB M3
00C	BNE TRANSF
00D	<i>suite du programme</i>

SOURCE	VAL_0
SOURCE + 1	VAL_1
SOURCE + n - 1	VAL_n

DEST	VAL_0
DEST + 1	VAL_1
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE<chargement indexé avec SOURCE>

STA X,DEST<rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>

Accumulateur

XR



000	INIT : LDXI 0
001	TRANSF:LDA X,SOURCE
002	STA X,DEST
003	INX
004	CPX I n
005	BNE TRANSF
006	<i>suite du programme</i>
007	
008	
009	
00A	
00B	
00C	
00D	

SOURCE VAL_0

SOURCE + n - 1 VAL_n

DEST

DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE<chargement indexé avec SOURCE>

STA X,DEST<rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>

Accumulateur

XR 0

000	INIT : LDXI 0
001	TRANSF:LDA X,SOURCE
002	STA X,DEST
003	INX
004	CPX I n
005	BNE TRANSF
006	<i>suite du programme</i>
007	
008	
009	
00A	
00B	
00C	
00D	

SOURCE VAL_0

SOURCE + n - 1 VAL_n

DEST

DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE <chargement indexé avec SOURCE>

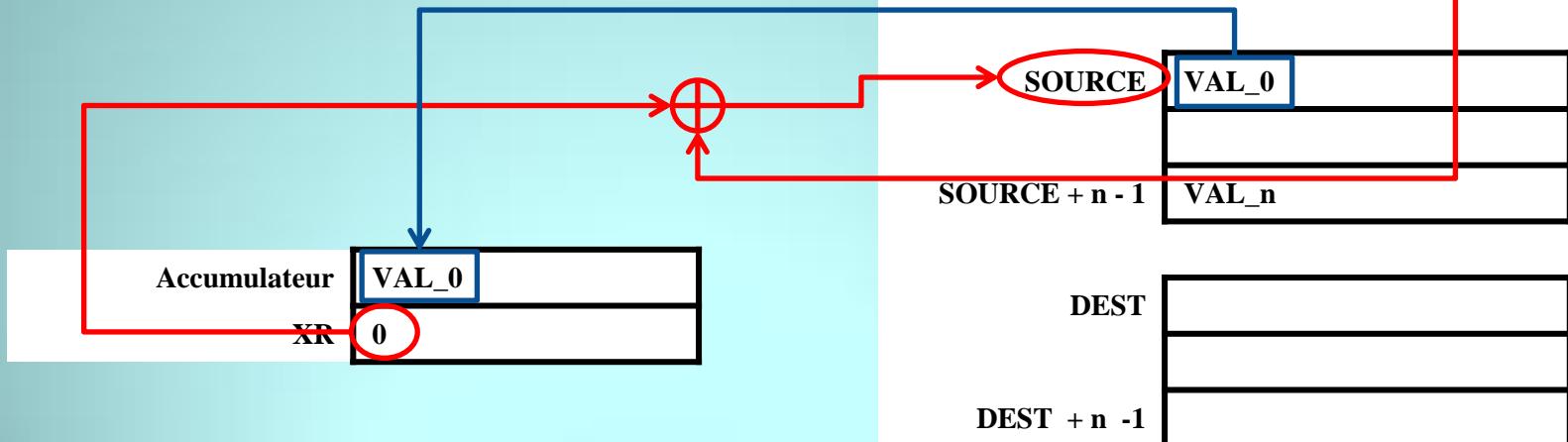
STA X,DEST <rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>



Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE <chargement indexé avec SOURCE>

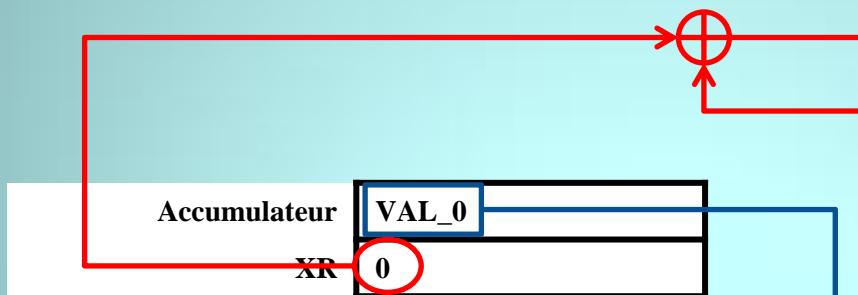
STA X,DEST <rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>



000	INIT : LDXI 0
001	TRANSF:LDA X,SOURCE
002	STA X,DEST
003	INX
004	CPX I n
005	BNE TRANSF
006	<i>suite du programme</i>
007	
008	
009	
00A	
00B	
00C	
00D	
SOURCE	VAL_0
SOURCE + n - 1	VAL_n
DEST	VAL_0
DEST + n - 1	

Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE<chargement indexé avec SOURCE>

STA X,DEST<rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>

Accumulateur

VAL_0

XR

0+1 = 1 =n ?

000	INIT : LDXI 0
001	TRANSF:LDA X,SOURCE
002	STA X,DEST
003	INX
004	CPX I n
005	BNE TRANSF
006	<i>suite du programme</i>
007	
008	
009	
00A	
00B	
00C	
00D	

SOURCE VAL_0

SOURCE + n - 1 VAL_n

DEST VAL_0

DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

INIT : LDXI 0 <chargement immédiat de l'index avec 0>

TRANSF:LDA X,SOURCE <chargement indexé avec SOURCE>

STA X,DEST <rangement indexé à DEST>

INX <+1 dans l'index>

CPX I n <comparaison immédiate de l'index à n>

BNE TRANSF <branchement en TRANSF si non égalité avec 0 >

suite du programme <suite du programme si égalité>

Accumulateur **VAL_0**

XR **1**

000	INIT : LDXI 0
001	TRANSF:LDA X,SOURCE
002	STA X,DEST
003	INX
004	CPX I n
005	BNE TRANSF
006	<i>suite du programme</i>
007	
008	
009	
00A	
00B	
00C	
00D	

SOURCE **VAL_0**

SOURCE + n - 1 **VAL_n**

DEST **VAL_0**

DEST + n - 1

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Réalisation d'une instruction

Deux possibilités pour implémenter les instructions interprétées par l'UC :

1. "câblage" : instruction → fonction booléenne → circuit (technologie adaptée)
 - avantage : rapide à l'exécution
 - inconvénient : complexe, cher et figé
2. "microprogrammation" : une instruction est considérée comme une fonction ou une procédure constituée de micro-instructions câblées ; l'instruction est enregistrée en mémoire morte
 - avantage : circuit du processeur simple et jeu d'instructions modifiables
 - inconvénient : vitesse d'exécution assez peu élevée

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Réalisation d'une instruction

Lorsque le jeu d'instructions est limité, on peut procéder de la manière suivante :

→ **émulation ou simulation logicielle** (instruction → fonction assembleur ou langage C) ; inconvénient : "trés" lent ...

→ ajout d'un *co-processeur* câblé ou microprogrammé (processeur spécialisé qui exécute des instructions spécifiques) ; l'Unité Centrale sous-traite ...

Exemple : le co-processeur arithmétique 80xx7 associé au 80xx6 réalise les opérations en Virgule Flottante et les fonctions mathématiques

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Constitution d'une Unité Centrale

- ❖ le processeur est caractérisé par le nombre très élevé de signaux à transmettre/recevoir
 - problème lors de la conception /réalisation d'un µP
- ❖ Le µP est réalisé
 - sur une plaquette de silicium (5mmx15mm) ;
 - enchâssé dans un boîtier plastique entouré de broches →le contact avec l'extérieur
- ❖ Insuffisant du nombre de broches
 - ⇒ certaines broches combinent *plusieurs fonctions* : nécessité de multiplexage
- ❖ L'horloge (interne ou externe au µP, commandée par un quartz) est indispensable pour cadencer le fonctionnement du µP
- ❖ Le jeu d'instructions de base peut être étendu grâce au *co-processeur*
 - "virgule flottante"
 - "accès direct mémoire" (DMA)
 - "graphique"

Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

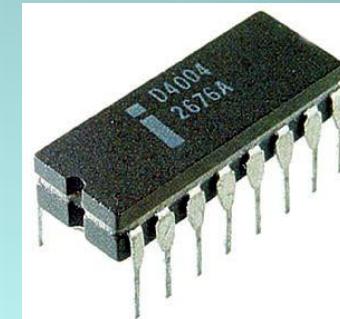
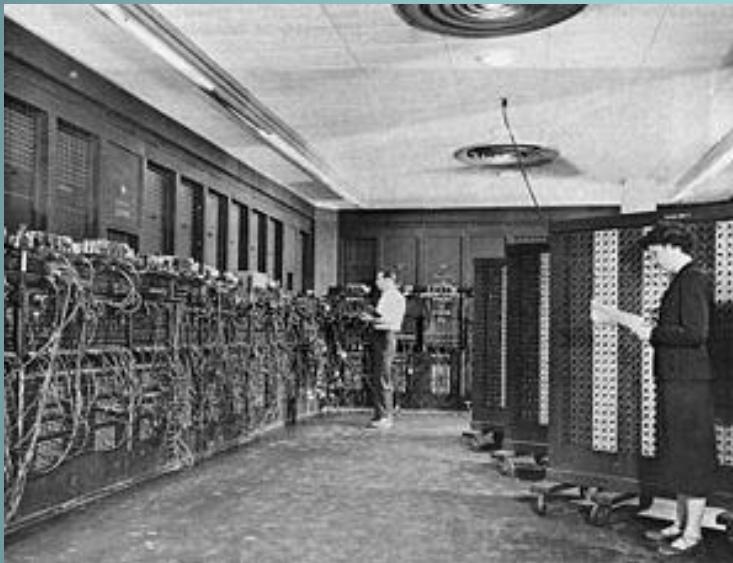
Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 4004

- Premier microprocesseur de l'histoire et de l'industrie
- Performances équivalentes aux 30 mètres cube d'un ENIAC concentré sur 10 mm carrés



Format de la puce : 3,81 mm de long sur 2,79 mm de large : 10,62 mm² à peine.

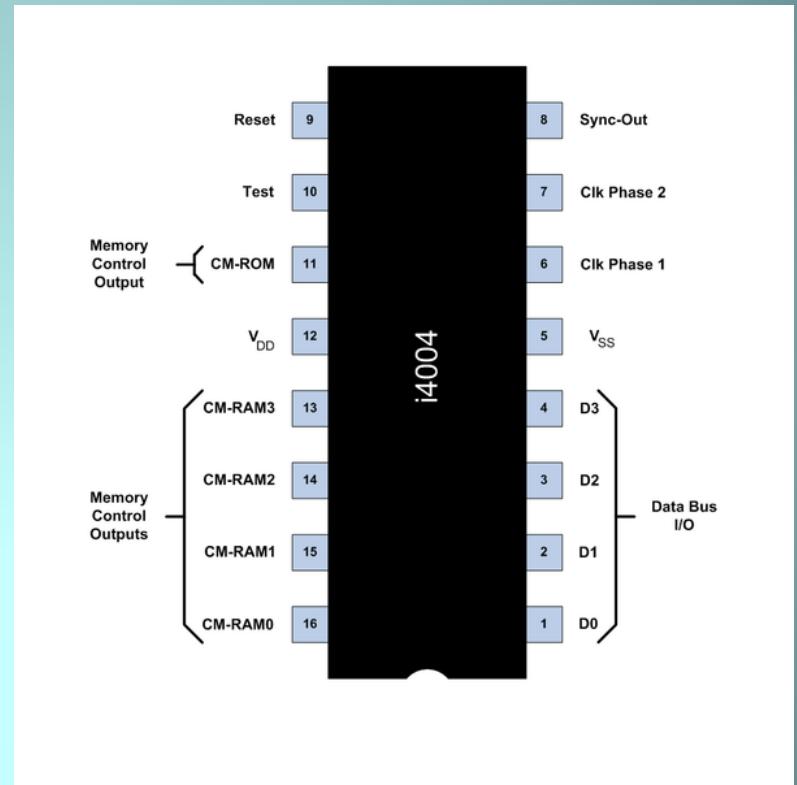
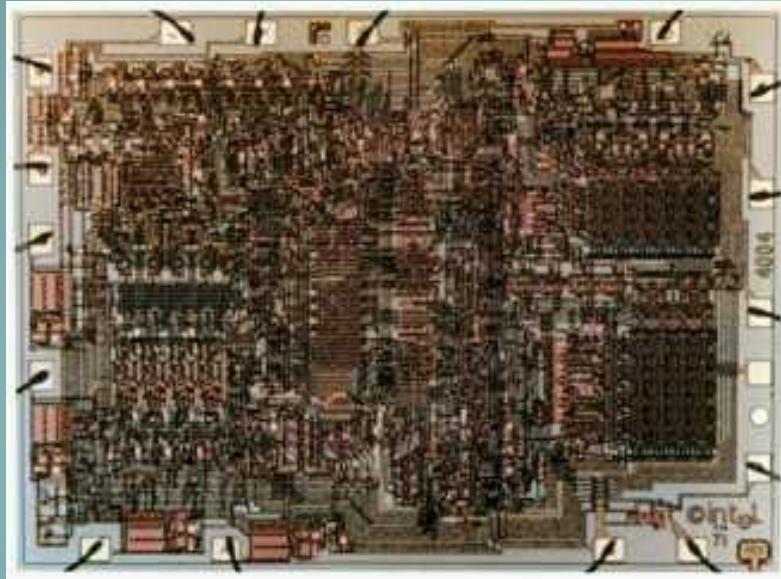
L'ENIAC, acronyme de **Electronic Numerical Integrator Analyser and Computer**, est le premier ordinateur entièrement électronique construit pour être Turing-complet. Son poids est de 30 tonnes pour des dimensions de 2,4 x 0,9 x 30,5 mètres occupant une surface de 167 mètres carrés. Sa consommation est de 150 kilowatts.

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 4004

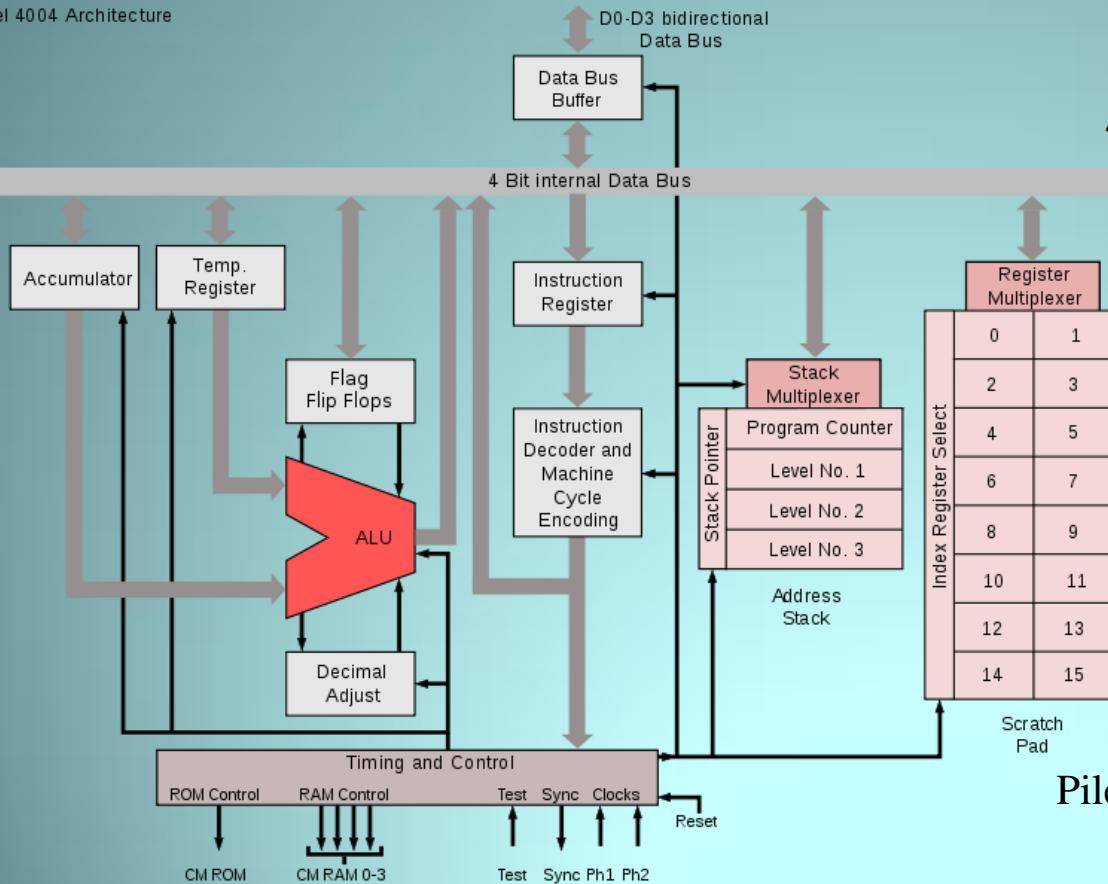


Deuxième partie

Unité centrale du microprocesseur

Microprocesseur 4004

Intel 4004 Architecture



Calculateur 4 bits,
16 registres de données 4 bits,
jeu de 46 instructions

(41 de 8 bits, et 5 de 16 bits),
1280 demi-octets de données,

4 Ko d'instructions programme en mémoire
ROM de 256 octets,

Largeur du bus de donnée 4 bits,
mémoire adressable 640 octets,

Fréquence d'horloge : 740kHz,
8 cycles (10.8us) par instruction,

16 broches DIP (soudé),
15 volt

2300 transistors MOS
60 000 instructions par seconde

Architecture Harvard

12 bits d'adresses (mémoire adressable 640
octets)

Pile interne de sous-programme profonde de 3
niveaux.

Deuxième partie

Unité centrale du microprocesseur

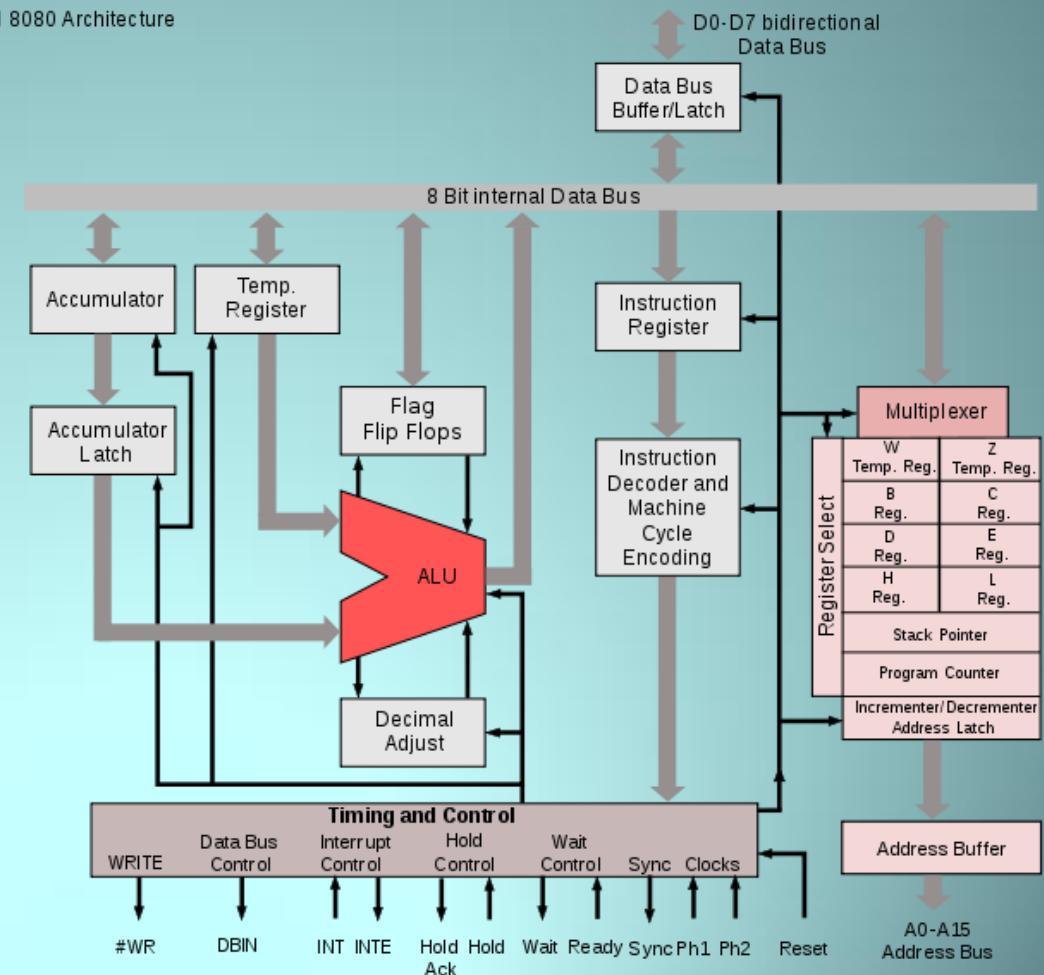
principes généraux de conception d'une unité centrale

Microprocesseur 8080 Intel

Intel® 8080 Microprocessor Die, 1974

(6mm x 6mm)

Intel 8080 Architecture



Deuxième partie

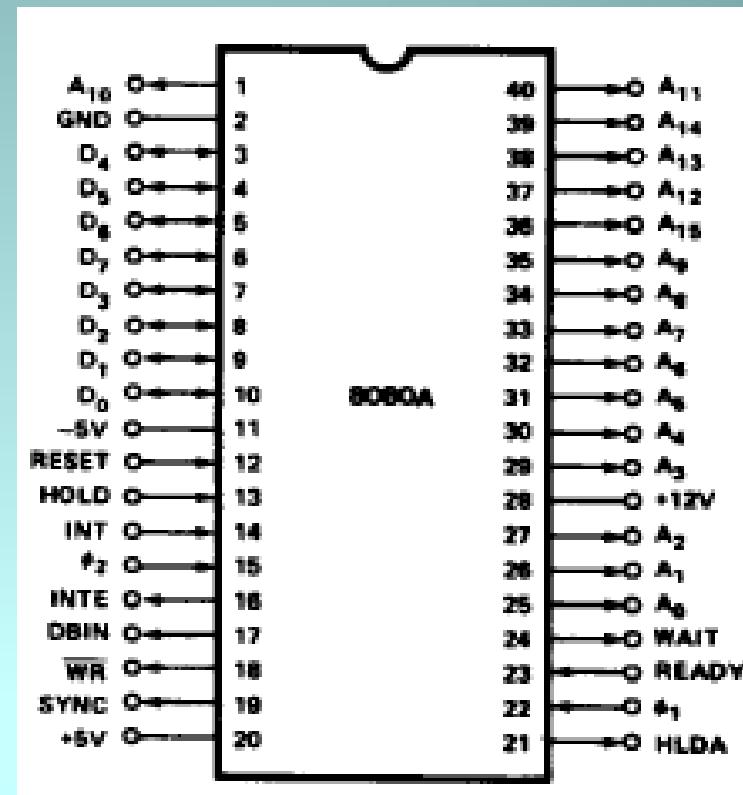
Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8080 Intel



- un pack DIP à 40 broches
- un bus d'adresses sur 16 bits
- un bus de données sur 8 bits
- 64 Ko de mémoire
- sept registres de 8 bits
- un pointeur de pile sur 16 bits
- un compteur de programme sur 16 bits
- possède 256 ports d'entrées/sorties accessibles par des instructions dédiées.



Deuxième partie

Unité centrale du microprocesseur

principes généraux de fonctionnement

Microprocesseur 8080 Intel

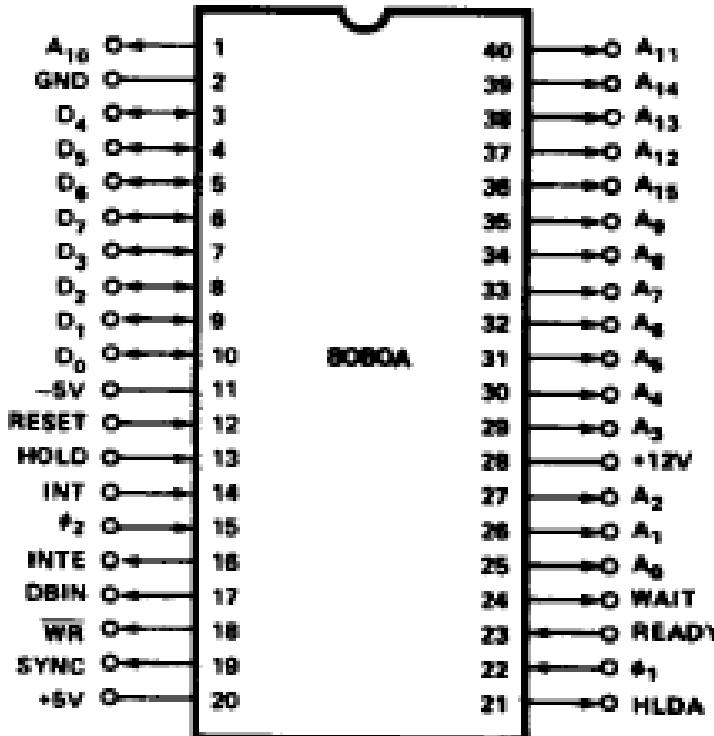


Table 1. Pin Description

Symbol	Type	Name and Function
A ₁₅ -A ₀	O	ADDRESS BUS: The address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A ₀ is the least significant address bit.
D ₇ -D ₀	I/O	DATA BUS: The data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D ₀ is the least significant bit.
SYNC	O	SYNCHRONIZING SIGNAL: The SYNC pin provides a signal to indicate the beginning of each machine cycle.
DBIN	O	DATA BUS IN: The DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.
READY	I	READY: The READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.
WAIT	O	WAIT: The WAIT signal acknowledges that the CPU is in a WAIT state.
WR	O	WRITE: The WR signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the WR signal is active low (WR = 0).
HOLD	I	HOLD: The HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these busses for the current machine cycle. It is recognized under the following conditions: <ul style="list-style-type: none"> • the CPU is in the HALT state. • the CPU is in the T2 or TW state and the READY signal is active. As a result of entering the HOLD state the CPU ADDRESS BUS (A₁₅-A₀) and DATA BUS (D₇-D₀) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.
HLDA	O	HOLD ACKNOWLEDGE: The HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at: <ul style="list-style-type: none"> • T3 for READ memory or input. • The Clock Period following T3 for WRITE memory or OUTPUT operation. In either case, the HLDA signal appears after the rising edge of φ ₂ .
INTE	O	INTERRUPT ENABLE: Indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.
INT	I	INTERRUPT REQUEST: The CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the interrupt Enable flip/flop is reset it will not honor the request.
RESET ¹	I	RESET: While the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.
V _{SS}		GROUND: Reference.
V _{DD}		POWER: + 12 ± 5% V.
V _{CC}		POWER: + 5 ± 5% V.
V _{BB}		POWER: - 5 ± 5% V.
φ ₁ , φ ₂		CLOCK PHASES: 2 externally supplied clock phases. (non TTL compatible)

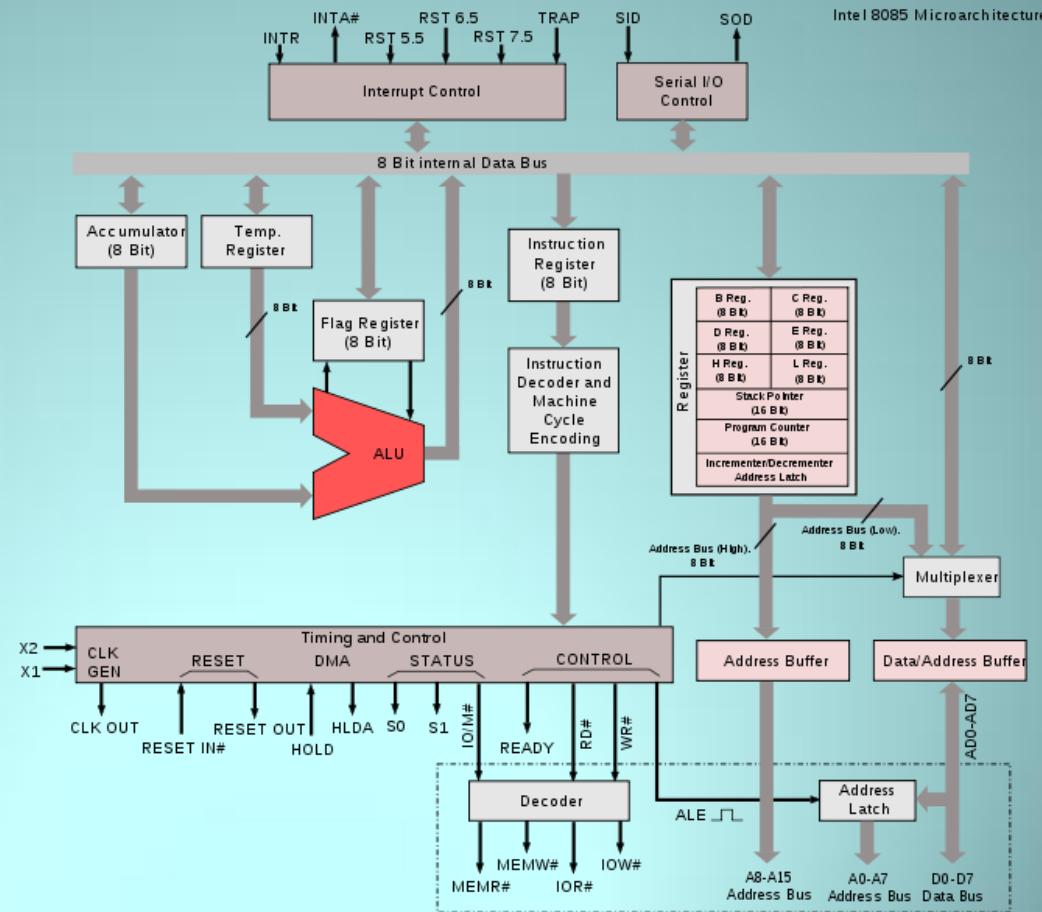
- 1) combien de lignes d'alimentation pour le 8080 ?
- 2) où sont les entrées horloge pour le 8080 ?

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel



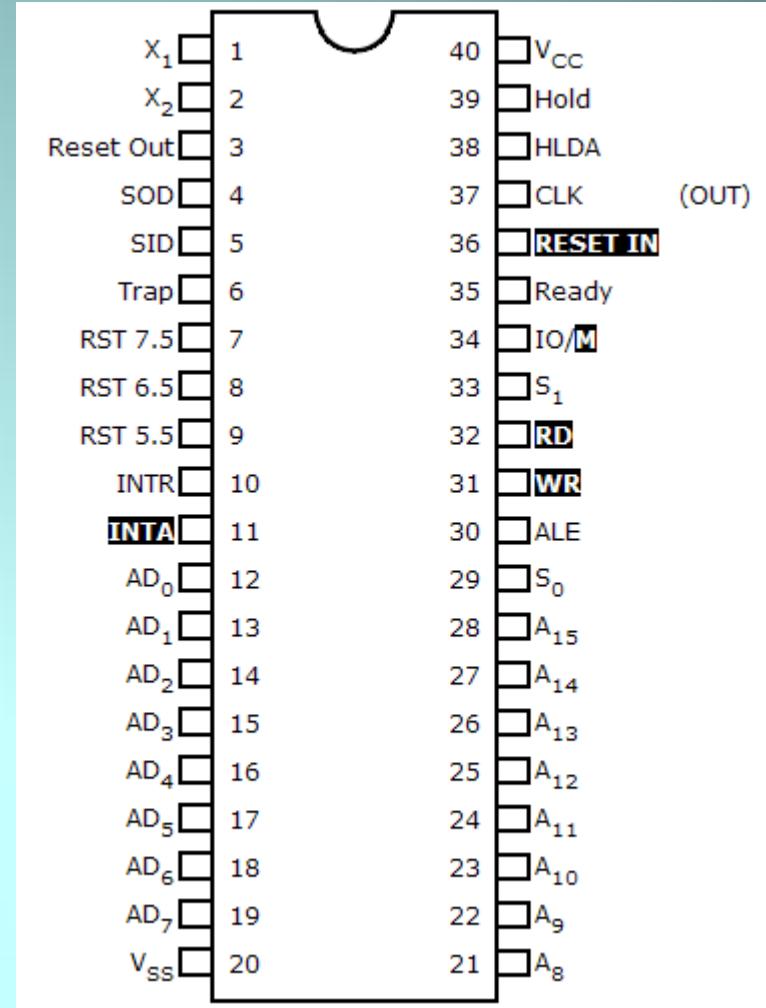
Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel

- Vcc + 5 volt power supply
- Vss Ground
- X1, X2 : Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
- The frequency is internally divided by two. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected externally.
- CLK (output)-Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.
- A8 - A15 (output; 3-state) It carries the most significant 8 bits of the memory address or the 8 bits of the I/O address;
- AD0 - AD7 (input/output; 3-state)



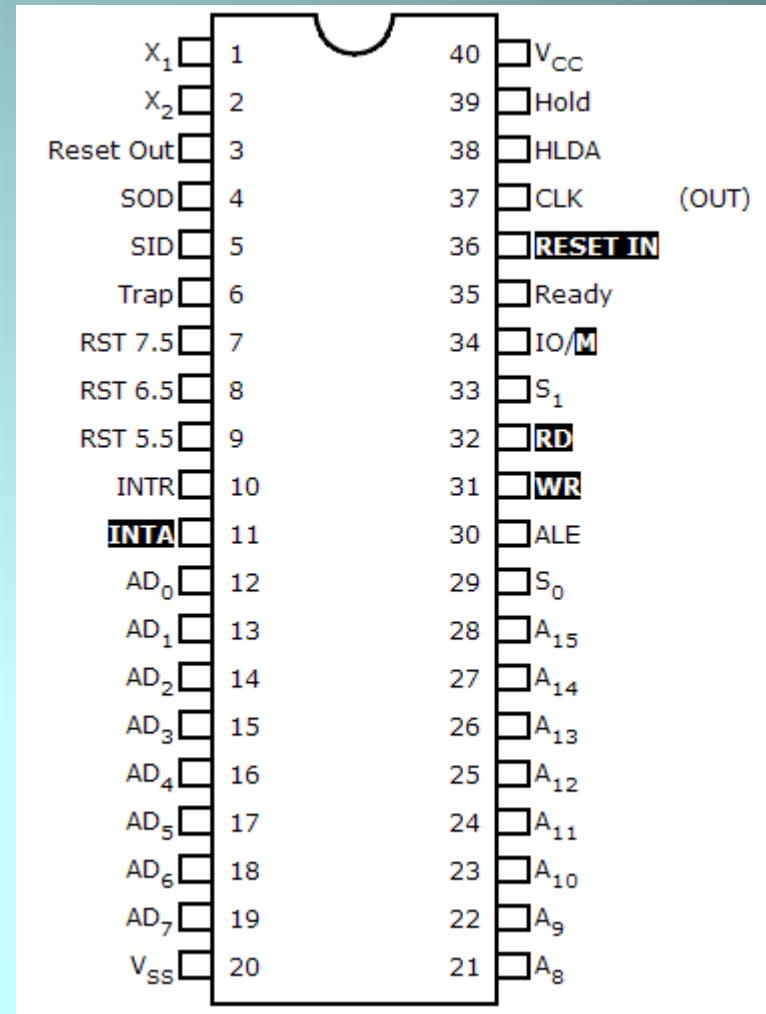
Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel

- **ALE (output) - Address Latch Enable.**
- **RD (output 3-state, active low) - Read memory or IO device.**
- **WR (output 3-state, active low) - Write memory or IO device.**
- **IO/M (output) - Select memory or an IO device.**
- This status signal indicates that the read / write operation relates to whether the memory or I/O device.
- It goes high to indicate an I/O operation.
- It goes low for memory operations.
- **RST5.5, RST6.5 and RST7.5 are a maskable interrupts.**



Deuxième partie

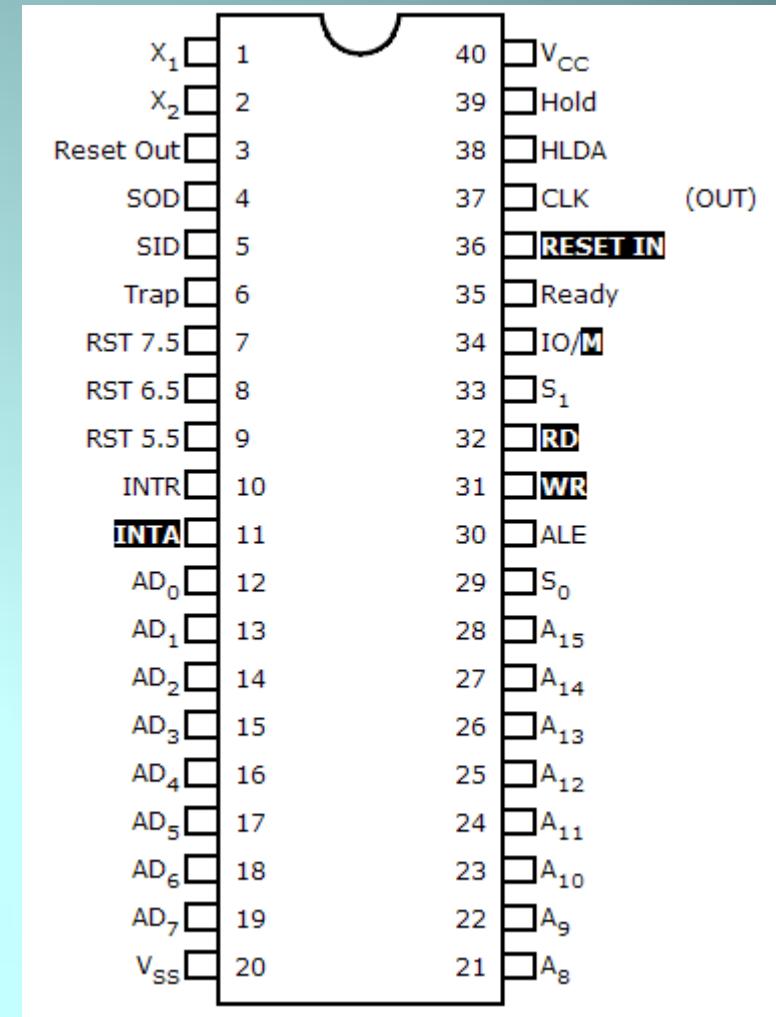
Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel

<http://www.8085projects.info>

IO/M(active Low)	S0	S1	Data Bus Status (Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge



Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Arithmetic Group:

Microprocesseur 8085 Intel

<http://www.8085projects.info>

Data Transfer Croup:

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memory
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from Memory
SHLD	Store H & L Registers Directly in Memory
LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register Pair
STAX	Store Accumulator in Address in Register Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add Immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry)
SBI	Flag
INR	Subtract Immediate from Accumulator Using Borrow
DCR	(Carry) Flag
INX	Increment Specified Byte by One
DCX	Decrement Specified Byte by One
DAD	Increment Register Pair by One
	Decrement Register Pair by One
	Double Register Add; Add Content of Register
	Pair to H & L Register Pair

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Branch Group:

Microprocesseur 8085 Intel

<http://www.8085projects.info>

Logical Group:

ANA	AND with Accumulator
ANI	AND with Accumulator Using Immediate Data
ORA	OR with Accumulator
OR	OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data
CMP	Compare
CPI	Compare Using Immediate Data
RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry
CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

JMP	Jump		
CALL	Call		
RET	Return		
NZ	Not Zero (Z = 0)		
Z	Zero (Z = 1)		
NC	No Carry (C = 0)		
C	Carry (C = 1)		
PO	Parity Odd (P = 0)		
PE	Parity Even (P = 1)		
P	Plus (S = 0)		
M	Minus (S = 1)		
<u>Jumps</u>	<u>Calls</u>	<u>Returns</u>	
C	CC	RC	(Carry)
INC	CNC	RNC	(No Carry)
JZ	CZ	RZ	(Zero)
JNZ	CNZ	RNZ	(Not Zero)
JP	CP	RP	(Plus)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)
PCHL	PCHL	Move H & L to Program Counter	
RST	RST	Special Restart Instruction Used with Interrupts	

Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel

<http://www.8085projects.info>

Stack I/O, and Machine Control Instructions:

The following instructions affect the Stack and/or Stack Pointer:

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

The I/O instructions are as follows:

IN	Initiate Input Operation
OUT	Initiate Output Operation

The Machine Control instructions are as follows:

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

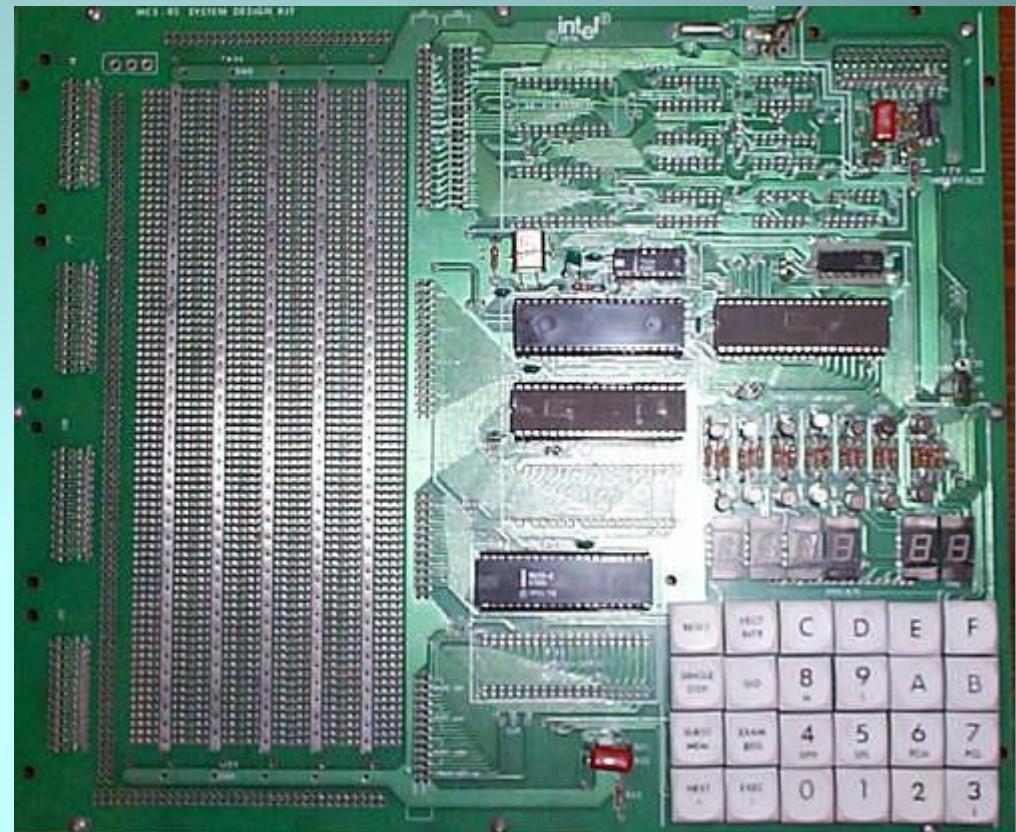
Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel → un exemple de programme sur le SDK 85

<http://www.8085projects.info>



Deuxième partie

Unité centrale du microprocesseur

principes généraux de conception d'une unité centrale

Microprocesseur 8085 Intel → un exemple de programme sur le SDK 85

(lire deux nombres d'un chiffre hexadecimal au clavier; afficher la somme. Donner le programme complet sur SDK 85)

Adresses	donnés	assembleur	commentaires
2000	31c220	LXI SP, 20c2H	; définir le SP
2003	3e08	MVI A, 08H	; pour MSE à 1
2005	30	SIM	; masquer inter.
2006	cde702	Loop: CALL RDKBD	; lire premier nb
2009	47	MOV B, A	; sauver nb dans B
200a	cde702	CALL RDKBD	; lire second nbre
200d	80	ADD B	; ajouter les 2 nb
200e	cd6e03	CALL UPDDT	; afficher somme
2011	c30620	JMP 2006	; affichage dû à
2014	...		; la boucle

Deuxième partie

Unité centrale du microprocesseur

- **introduction**
- **étude d'un cas d'école**
- **principes généraux de conception d'une Unité Centrale**
- **quelques exemples**
- **étude d'un composant du marché**
- **notions de programmation en langage d'assemblage**

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → les µP CISC d'Intel

	Prix lancement actuel	MIPS lancement maximum	Nombre trans adressage
8080 Juin 1978	\$360 variable	0,33 0,75	29000 1 Mo
80286 Février 1982	\$360 \$8	1,2 2,66	134000 16 Mo
80386 Octobre 1985	\$299 \$91	5 11,4	275000 4 Go
80486 Août 1989	\$950 \$317	20 54(*)	1,2 million 4 Go
Pentium Mars 1993	\$900 \$200	112 (**) (66 MHz)	3,1 millions 4 Go
P6 2è semestre 1995	\$1400	2èPentium (100MHz)	5,5 millions 4 Go

MIPS :
Million
d'Instructions
Par Seconde.

(*) 54 MIPS pour le 486 DX2-66 MHz(**) soit 150 fois plus puissant que le plus rapide 8086 !

Source : BYTE May 1993 (annonce Pentium) p.94

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel

➤ **Le processeur 8086/8088 d'Intel → base des processeurs Pentium actuels.**

Les processeurs successifs (de PC) se sont en effet construits petit à petit en ajoutant à chaque processeurs des instructions et des fonctionnalités supplémentaires, mais en conservant à chaque fois les spécificités du processeur précédent. C'est cette façon d'adapter les processeurs à chaque étape qui permet qu'un ancien programme écrit pour un 8086 fonctionne toujours sur un nouvel ordinateur équipé d'un Pentium IV.

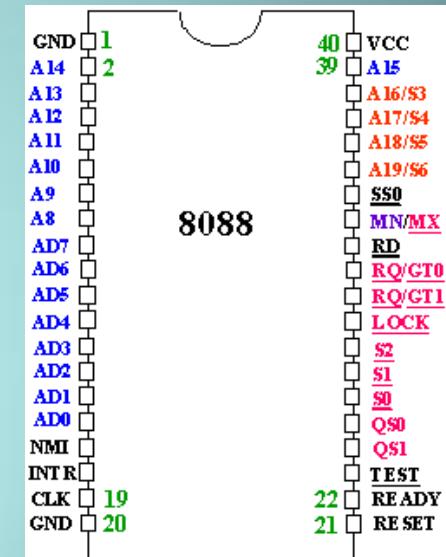
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → description générale du 8088 d'Intel

- DATE DE NAISSANCE : 1979
- Contrairement au 8086 (16 bits) le 8088 est un "faux 8 bits" : *chemin de données interne de 16 bits mais dialogue avec l'extérieur par un bus de données de 8 bits*
- équipe les premiers PC(Personal Computer) IBM
- µP nmos proposé dans un boîtier de 40 broches
- son architecture et son jeu d'instructions reprennent ceux du 8080 (8 bits) apparu 5 ans plus tôt, mais possibilité de travailler avec :
 - registres de 16 bits et
 - d'avantage de mémoire
- µP de base d'une famille de circuits :
 - 8086/88 µP
 - 8087 coprocesseur arithmétique
 - 8089 coprocesseur d'E/S
 - 8259 A contrôleur d'interruptions
 - 8284 circuit d'horloge
 - 8288 contrôleur de bus
 - 8289 circuit d'arbitrage des priorités



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → description générale du 8088 d'Intel

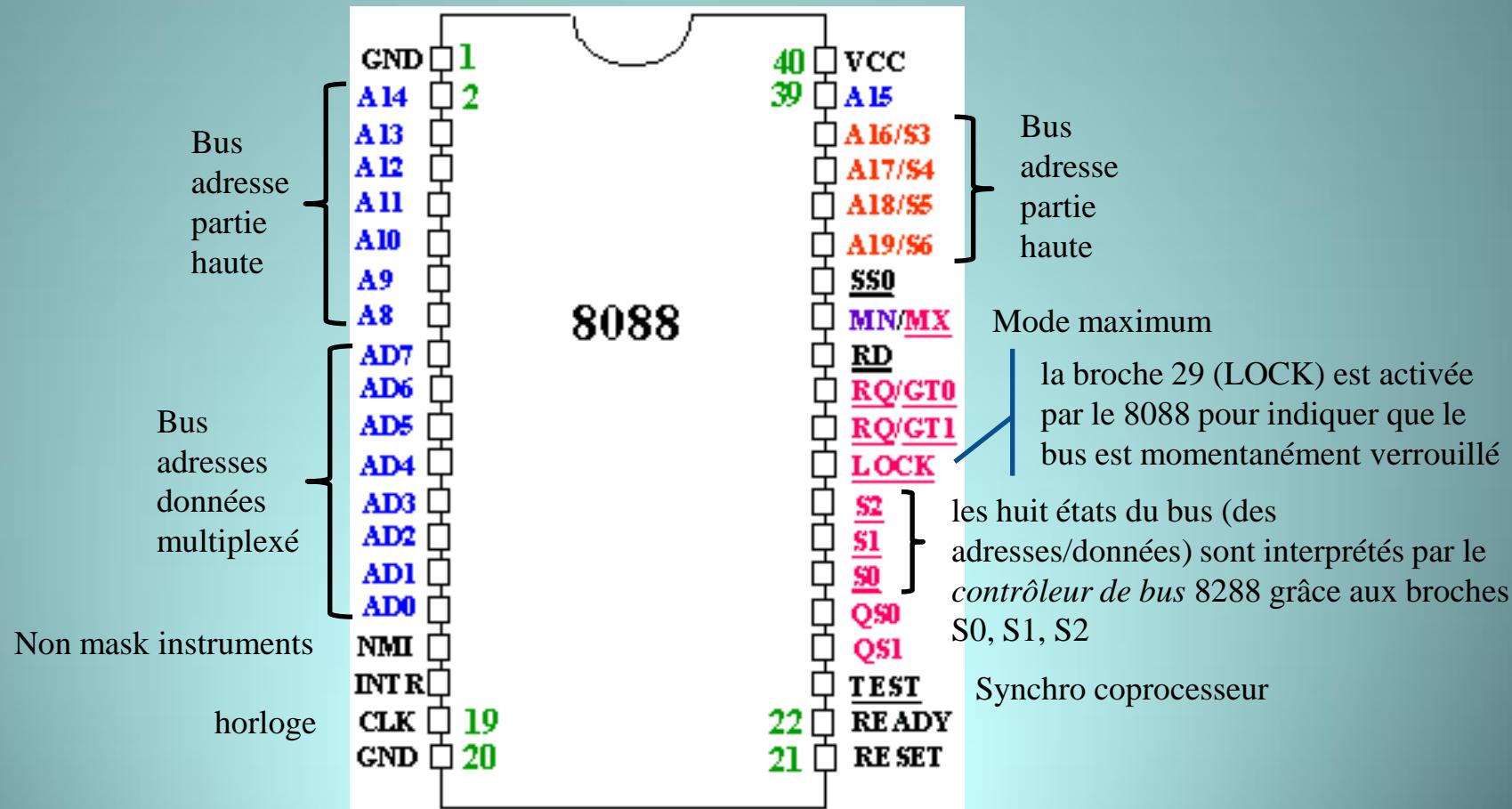
- le 8088 possède un bus adresses/données *multiplexé* (AD0-AD7) de 8 bits
- le bus d'adresses est sur 20 bits ⇒ 1 Mega octets adressables
- dans la phase "*recherche de l'instruction*" le 8088 met en oeuvre un mécanisme *d'anticipation d'acquisition* des instructions ⇒ lorsque l'instruction courante est terminée, la prochaine instruction est déjà présente dans le µP et peut être exécutée.
- le 8088 a *deux modes fonctionnels* (cf. Broche 33 MN/MX) :
 - mode minimum (cf. lave-linge)
 - mode maximum (cf. Ordinateur)
- la plupart des broches du 8088 ont :
 - des *fonctions identiques* à celles du 8085
 - des *fonctions spécifiques* au mode dans lequel opère le µP (mini ou maxi) : c'est le cas des broches 24 à 31 et notamment les broches S0, S1, S2 qui définissent les 8 états (types de cycle) du bus des adresses/données.

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → description générale du 8088 d'Intel

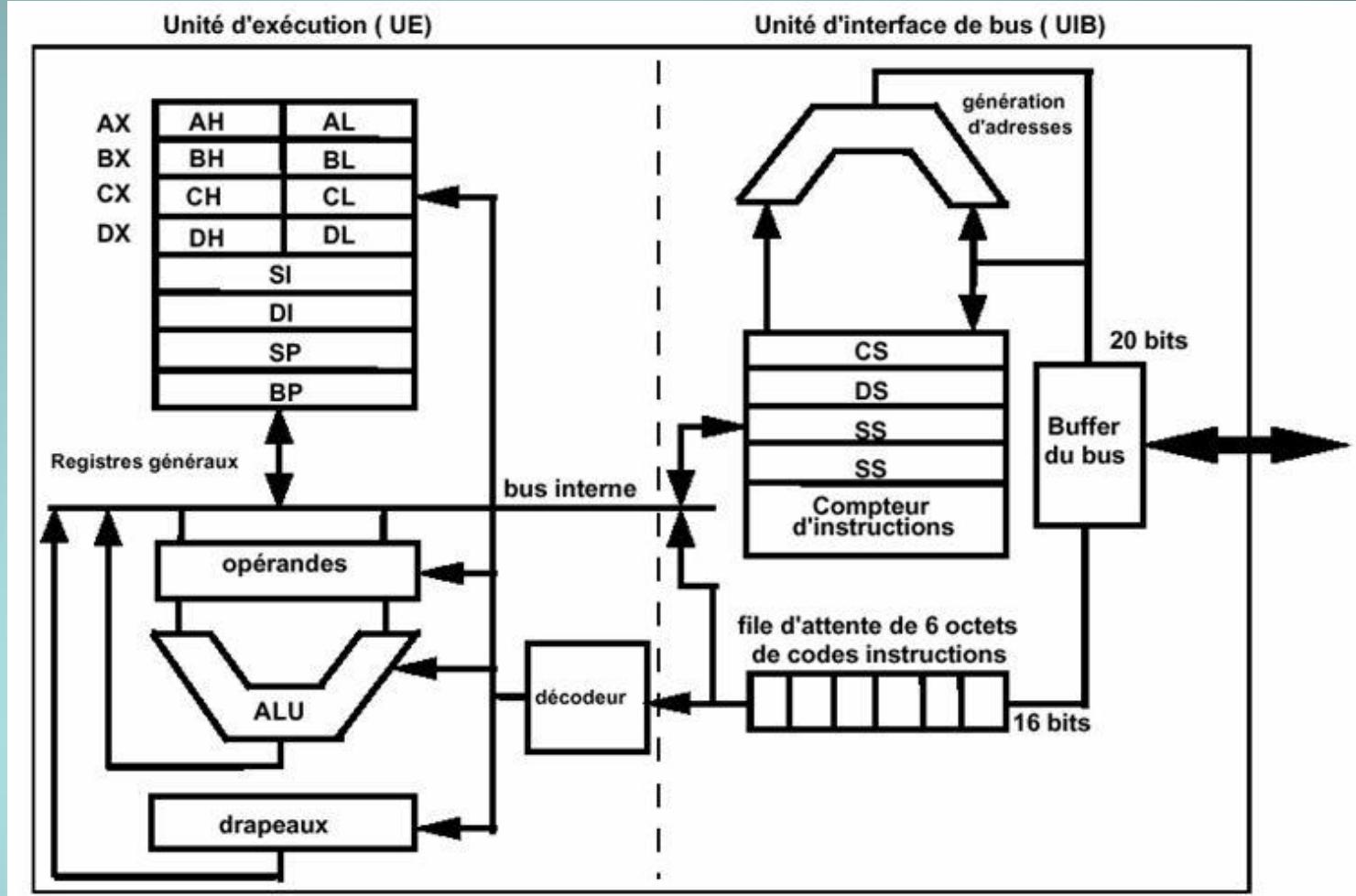


Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

- deux unités internes distinctes:
 - l'UE (Unité d'Exécution)
 - l'UIB (Unité d'Interfaçage avec le Bus)
- Lorsque l'exécution d'une instruction est terminée, l'UE reste inactif un court instant, pendant que l'UIB extrait l'instruction suivante



Deuxième partie

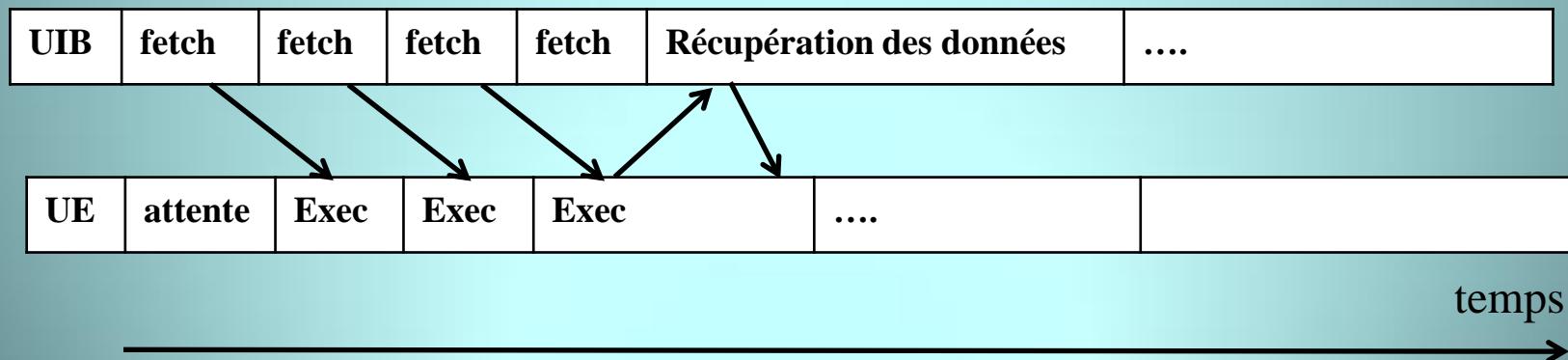
Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → description générale du 8088 d'Intel

Architecture « pipeline »

→ Pour remédier à ce temps d'attente, le *prétraitement* ou *traitement pipeline* a été introduit dans le 8086/8088. Pendant que l'UE exécute les informations qui lui sont transmises, l'instruction suivante est chargée dans l'UIB. Les instructions qui suivront sont placées dans une file d'attente. Lorsque l'UE a fini de traiter une instruction l'UIB lui transmet instantanément l'instruction suivante, et charge la troisième instruction en vue de la transmettre à l'UE. De cette façon, l'UE est continuellement en activité.



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

1. 8 REGISTRES GENERAUX

❖ **4 REGISTRES GENERAUX ou de travail (op' arithm.) sur 16 bits (DATA REGISTERS):**

Registre AX : (Accumulateur)

- les opérations de transferts de données avec les entrées-sorties
- le traitement des chaînes de caractères
- les opérations arithmétiques et logiques.
- les conversions en BCD du résultat d'une opération arithmétique (addition, soustraction, multiplication et la division)

Registre BX : (registre de base)

Il est utilisé pour l'adressage de données dans une zone mémoire différente de la zone code : en général il contient une adresse de décalage par rapport à une adresse de référence.). De plus il peut servir pour la conversion d'un code à un autre.

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

1. 8 REGISTRES GENERAUX

❖ **4 REGISTRES GENERAUX ou de travail (op' arithm.) sur 16 bits (DATA REGISTERS) :**

Registre CX : (Le compteur)

Lors de l'exécution d'une boucle on a souvent recours à un compteur de boucles pour compter le nombre d'itérations, 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 mais surtout pour contenir le numéro d'un port d'entrée/sortie pour adresser les interfaces d'E/S.

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

1. 8 REGISTRES GENERAUX

❖ 4 REGISTRES GENERAUX ou de travail (op' arithm.) sur 16 bits (DATA REGISTERS) :

	7	0	7	0	
ax	ah		al		accumulateur
bx	bh		bl		base
cx	ch		cl		compteur
dx	dh		dl		données

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

1. 8 REGISTRES GENERAUX

❖ **4 REGISTRES GENERAUX d'ADRESSAGE dans un ESPACE D'ADRESSAGE / SEGMENT sur 16 bits (POINTER & INDEX REGISTERS) :**

→ spécialement adaptés au traitement des éléments dans la mémoire. Ils sont généralement munis de propriétés d'incrémentation et de décrémentation.

L'indexe SI : (source indexe) :

Il permet de pointer la mémoire et forme en général un décalage (un offset) par rapport à une base fixe (le registre DS), il sert aussi pour les instructions de chaîne de caractères, en effet il pointe sur le caractère source.

L'indexe DI : (Destination indexe) :

Il permet aussi de pointer la mémoire il présente un décalage par rapport à une base fixe (DS ou ES), il sert aussi pour les instructions de chaîne de caractères, il pointe alors sur la destination

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

1. 8 REGISTRES GENERAUX

❖ **4 REGISTRES GENERAUX d'ADRESSAGE dans un ESPACE D'ADRESSAGE / SEGMENT sur 16 bits (POINTER & INDEX REGISTERS) :**

Les pointeurs SP et BP : (Stack pointer et base pointer)

- Ils pointent sur la zone pile (une zone mémoire qui stocke l'information avec le principe FILO, ils présentent un décalage par rapport à la base (le registre SS))
- Pour le registre BP il a un rôle proche de celui de BX, mais il est généralement utilisé avec le segment de pile.

	15	0	
sp			Stack pointer
bp			Base pointer
si			Source index
di			Dest. index

Deuxième partie

Unité centrale du microprocesseur

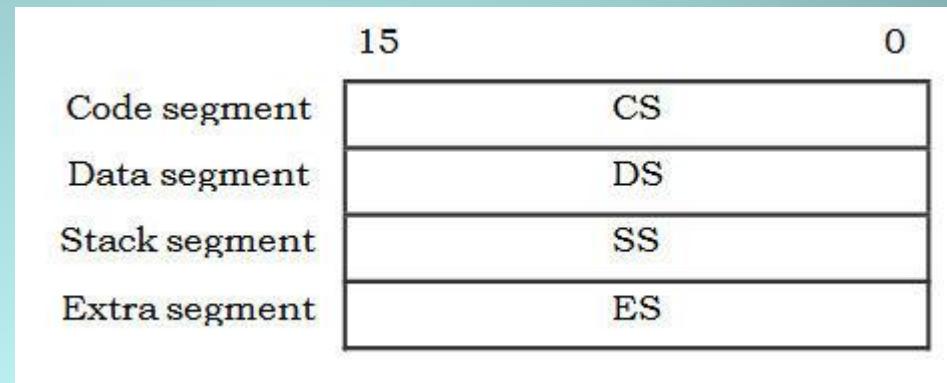
Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

2) 4 REGISTRES D'ADRESSAGE DES SEGMENTS (*SEGMENT REGISTERS*) :

➤ Quatre **registres segments** de 16 bits chacun :

- CS (code segment)
- DS (Data segment)
- ES (Extra segment)
- SS (stack segment)



➤ ces registres sont chargés de sélectionner les différents segments de la mémoire en pointant sur le début de chacun d'entre eux. Chaque segment de mémoire ne peut excéder les 65535 octets(2^{16}).

⇒ *l'espace mémoire est divisé en 4 segments de capacité maximale 64 K octets*

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → GESTION DES ADRESSES MEMOIRE DU 8088 d'Intel

Le registre CS (code segment) :

Il pointe sur le segment qui contient les codes des instructions du programme en cours.

Le registre DS (Data segment) :

Le registre segment de données pointe sur le segment des variables globales du programme

Le registre ES (Extra segment) :

Le registre de données supplémentaires ES est utilisé par le microprocesseur lorsque l'accès aux autres registres est devenu difficile ou impossible pour modifier des données, de même ce segment est utilisé pour le stockage des chaînes de caractères.

Le registre SS (Stack segment) :

Le registre SS pointe sur la pile : la pile est une zone mémoire où on peut sauvegarder les registres (ou les adresses ou les données) pour les récupérer après l'exécution d'un sous programme ou l'exécution d'un programme d'interruption. En général il est conseillé de ne pas changer le contenu de ce registre car on risque de perdre des informations très importantes (exemple les passages d'arguments entre le programme principal et le sous programme)

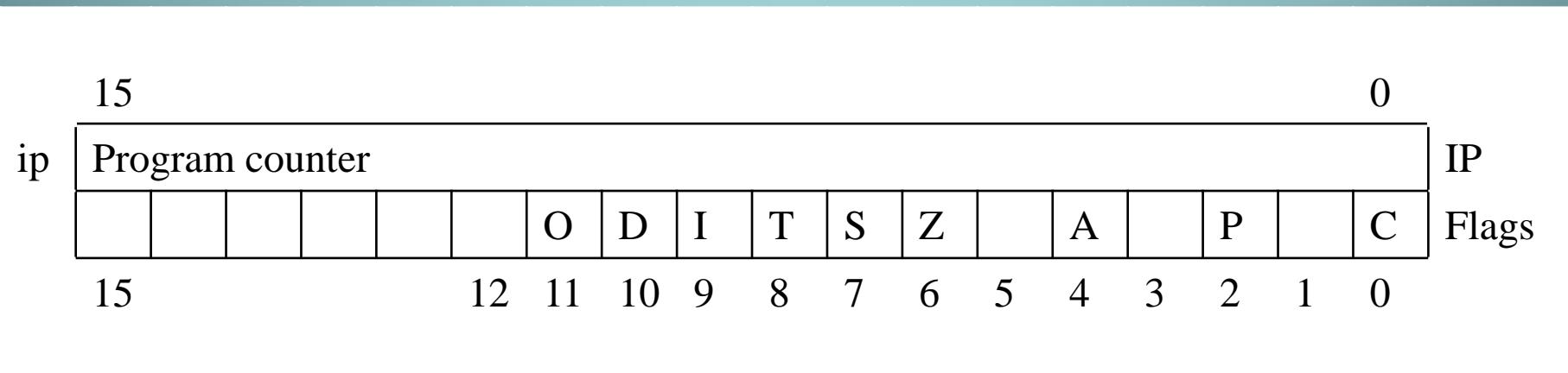
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

3) COMPTEUR ORDINAL (IP)



Instruction Pointer (IP) ou **Compteur de Programme (PC)**, contient l'adresse de l'emplacement mémoire où se situe la prochaine instruction à exécuter. Autrement dit, il doit indiquer au processeur la prochaine instruction à exécuter. Le registre IP est constamment modifié après l'exécution de chaque instruction afin qu'il pointe sur l'instruction suivante.

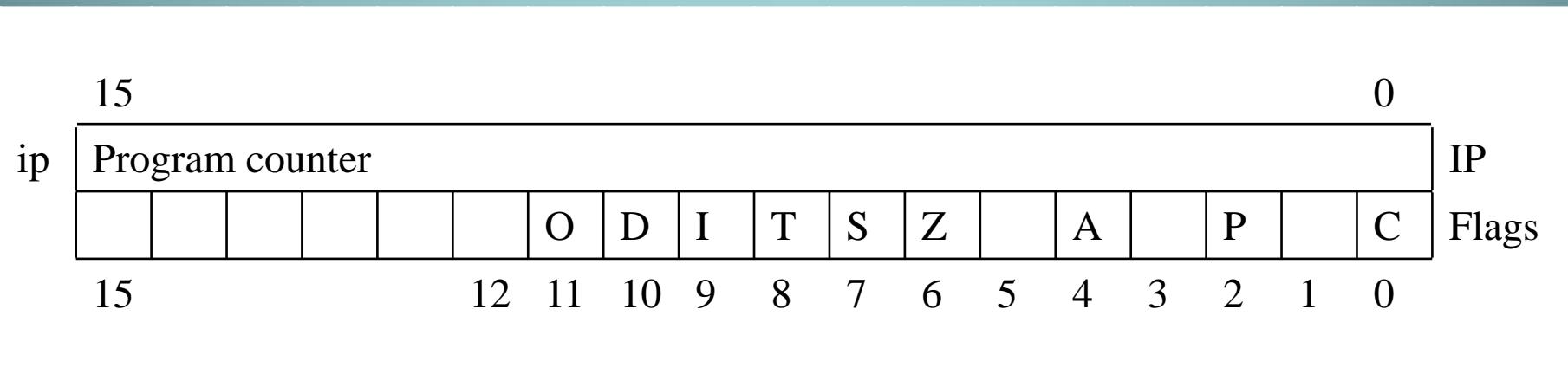
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



Le registre d'état FLAG sert à contenir l'état de certaines opérations effectuées par le processeur. Par exemple, quand le résultat d'une opération est trop grand pour être contenu dans le registre cible (celui qui doit contenir le résultat de l'opération), un bit spécifique du registre d'état (le bit OF) est mis à 1 pour indiquer le débordement.

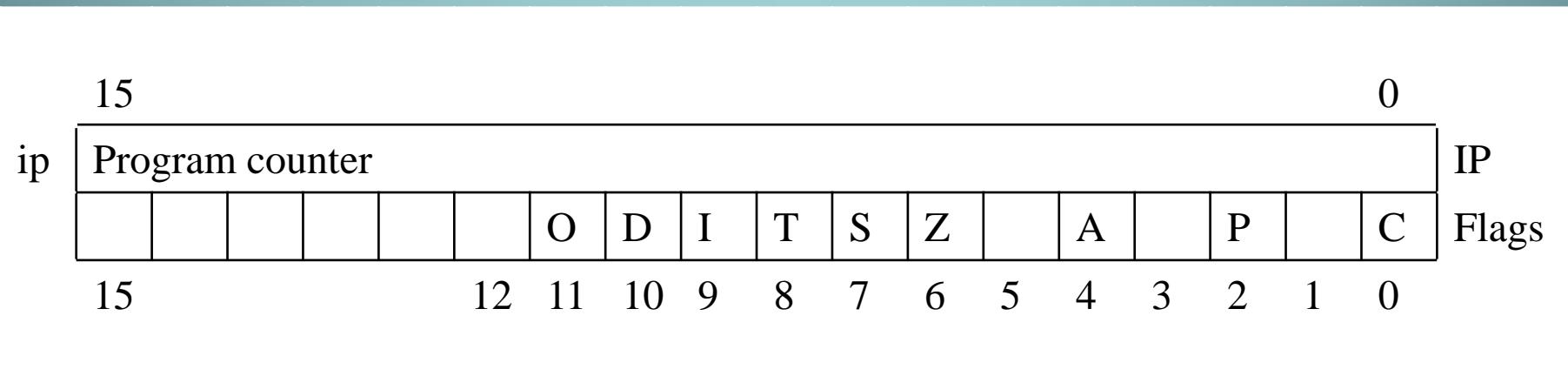
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



O : Overflow Flag

Débordement : si on a un débordement arithmétique ce bit est mis à 1. c ad le résultat d'une opération excède la capacité de l'opérande (registre ou case mémoire), sinon il est à 0.

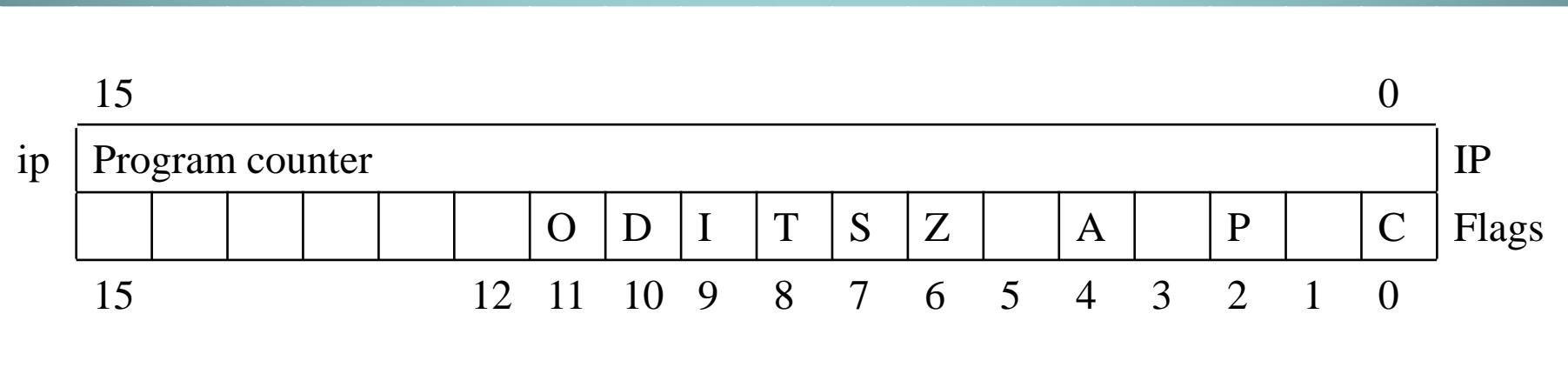
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



D : Direction Flags (Strings)

Auto Incrémentation/Décrémentation : utilisée pendant les instructions de chaîne de caractères pour auto incrémenter ou auto décrémenter.

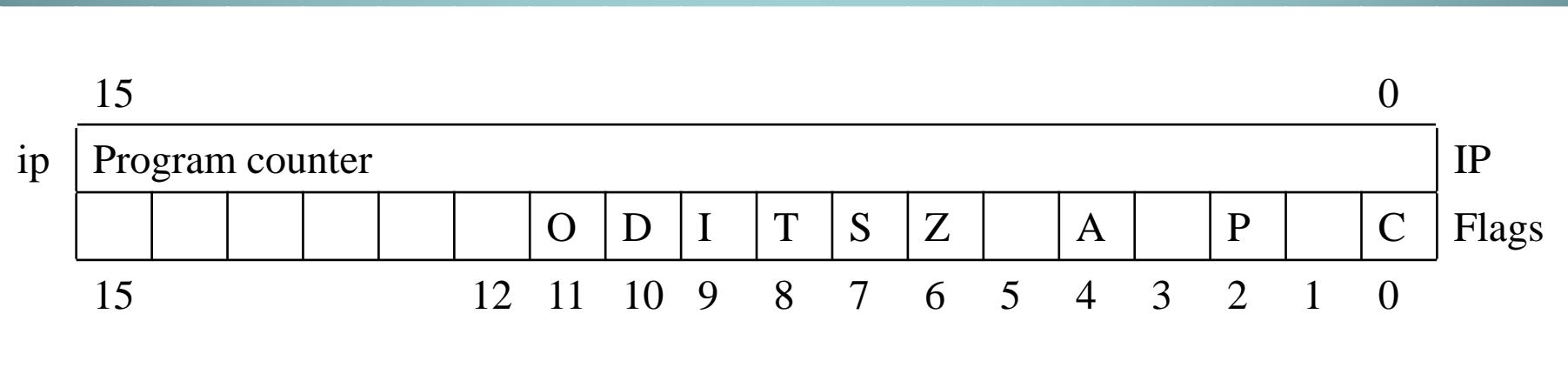
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



I : Interrupt Flag

Masque d'interruption : pour masquer les interruptions venant de l'extérieur ce bit est mis à 0, dans le cas contraire le microprocesseur reconnaît l'interruption de l'extérieur.

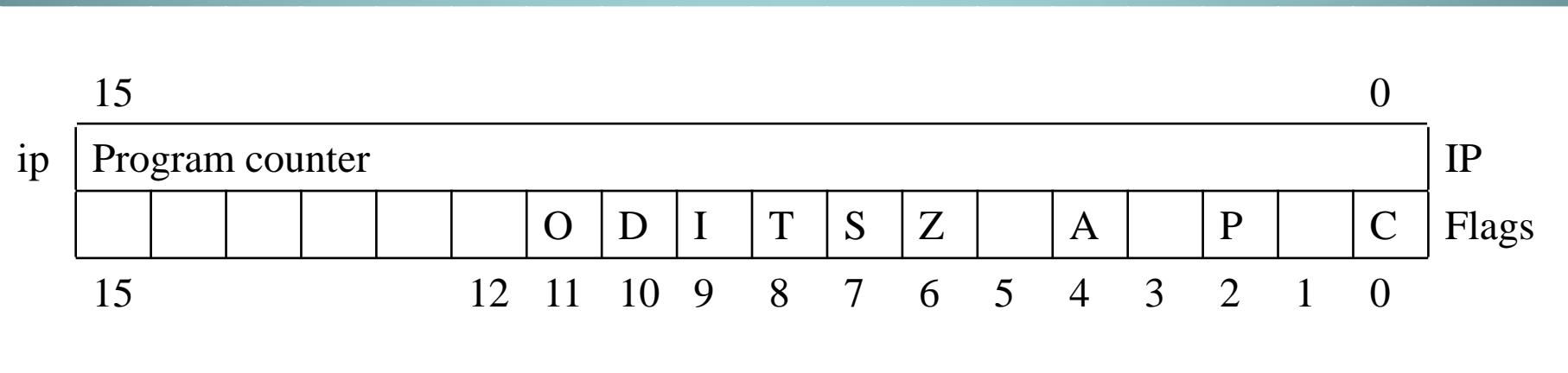
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



T : Trap (single step) Flag

Piège : pour que le microprocesseur exécute le programme pas à pas du.

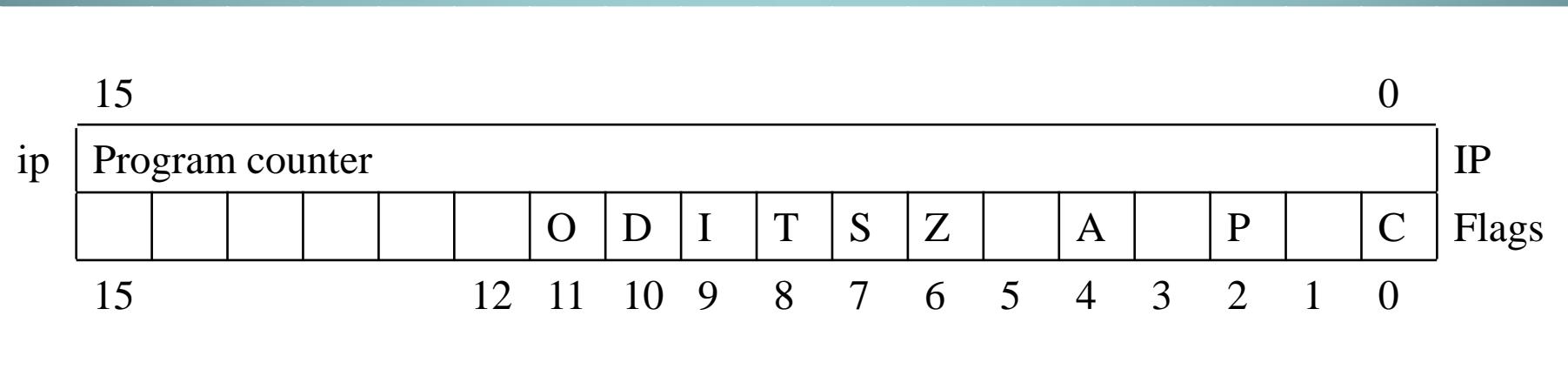
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



S : Sign Flag

SF est positionné à 1 si le bit de poids fort du résultat d'une addition ou soustraction est 1 ; sinon SF=0. SF est utile lorsque l'on manipule des entiers signés, car le bit de poids fort donne alors le signe du résultat.
Exemples (sur 8 bits) :

10010110	11011001
+ 01010100	+ 01010010
<u> </u>	<u> </u>
SF=1 11101010	SF=0 00101011

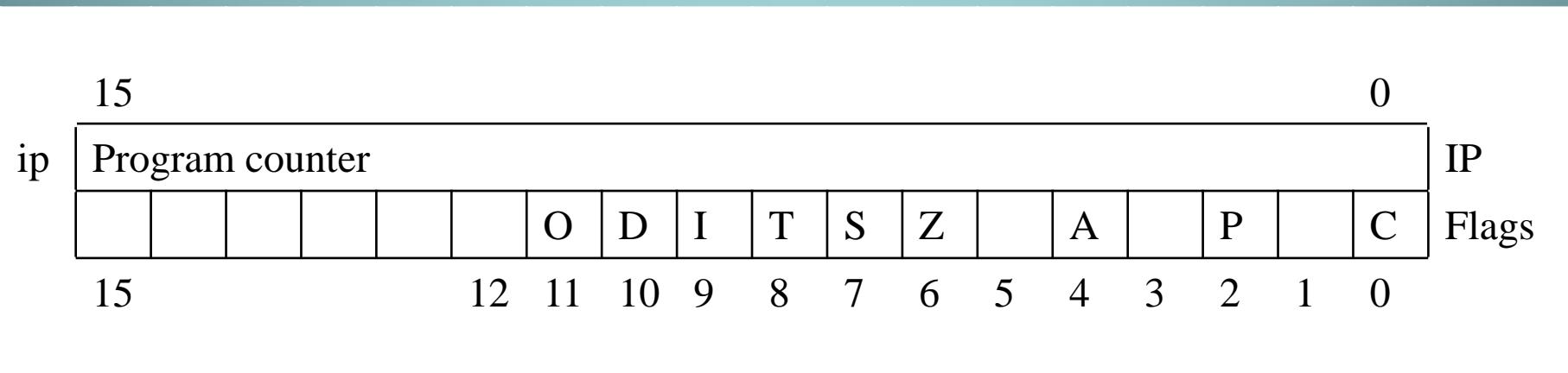
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



Z : Zero Flag

Zéro : Cet indicateur est mis à 1 quand le résultat d'une opération est égal à zéro. Lorsque l'on vient d'effectuer une soustraction (ou une comparaison), ZF=1 indique que les deux opérandes étaient égaux. Sinon, ZF est positionné à 0.

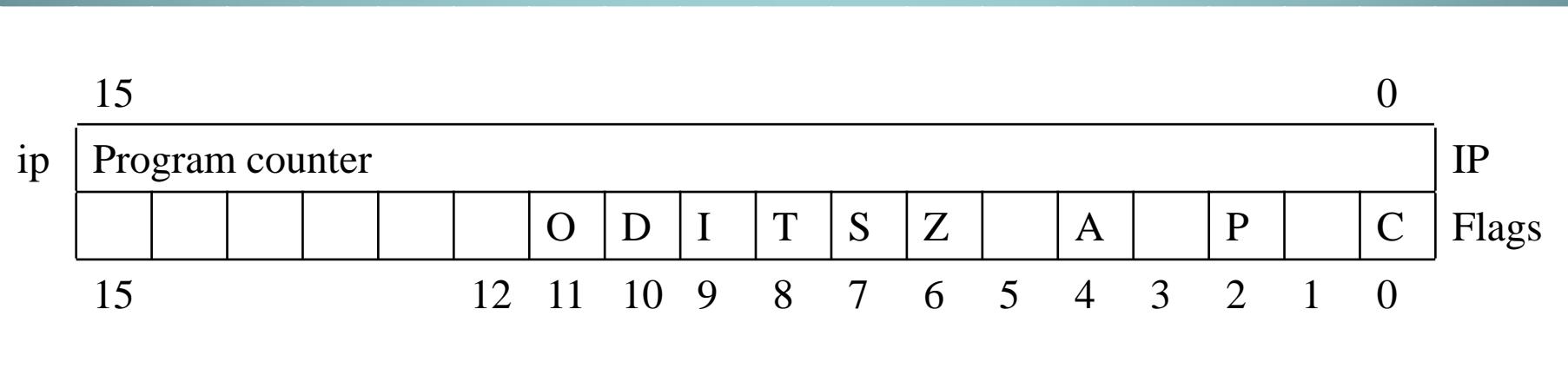
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



A : Auxiliary Carry (BCD)

Demie retenue : Ce bit est égal à 1 si on a une retenue du quartier de poids faible dans le quartier de poids plus fort.

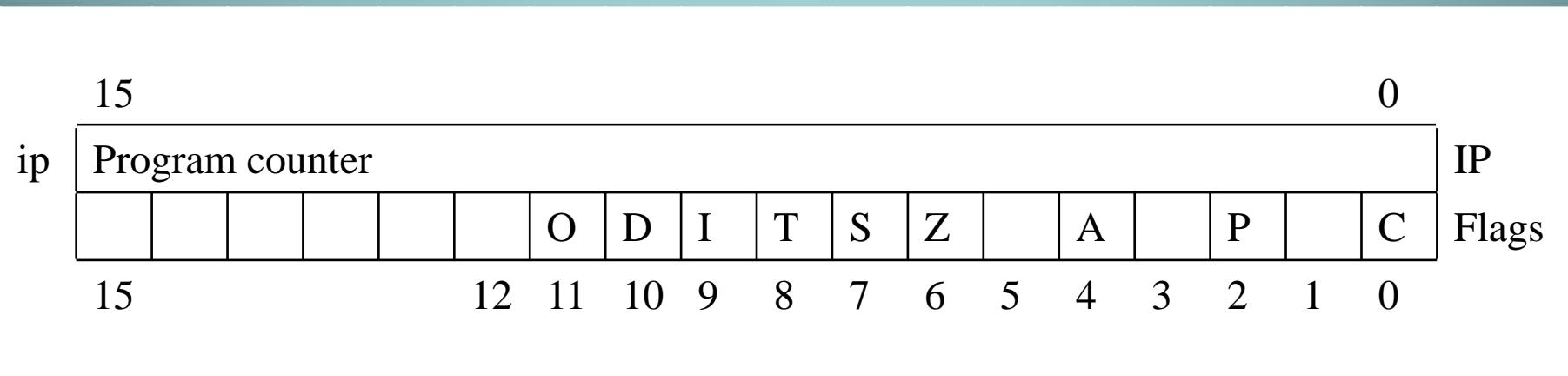
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



P : Parity Flag

Parité : si le résultat de l'opération contient un nombre pair de 1 cet indicateur est mis à 1, sinon zéro.

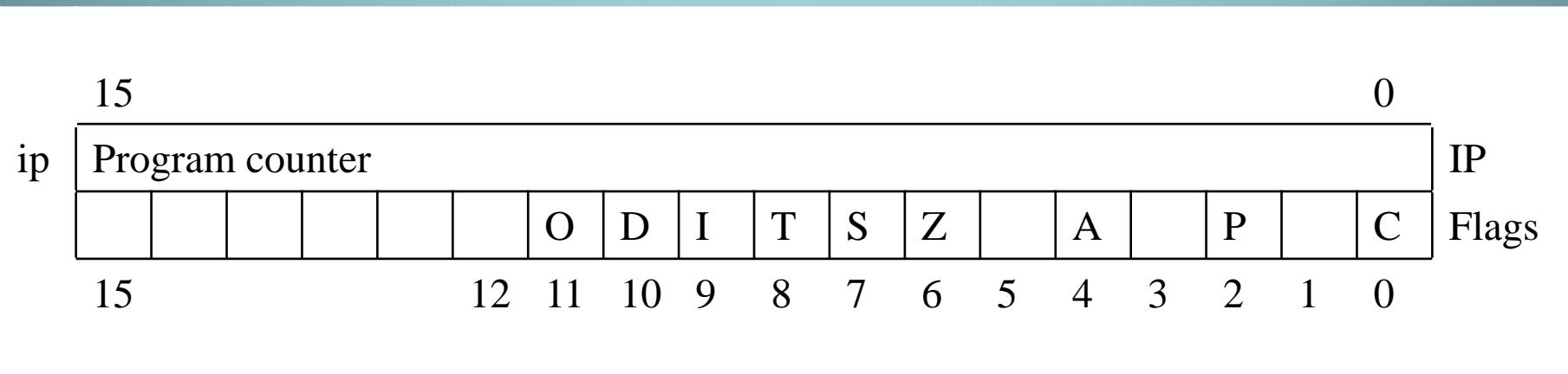
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → ARCHITECTURE INTERNE DU 8088 d'Intel

4) REGISTRES D'ETAT (FLAGS) :



C : Carry Flag

Retenue : cet indicateur est mis à 1 lorsque il y a une retenue du résultat à 8 ou 16 bits.

Il intervient dans les opérations d'additions (retenue) et de soustractions (borrow) sur des entiers naturels .

Il est positionné en particulier par les instructions ADD, SUB et CMP(comparaison entre deux valeurs).

CF = 1 s'il y a une retenue après l'addition ou la soustraction du bit de poids fort des opérandes. Exemples (sur 8 bits pour simplifier) :

$\begin{array}{r} 10010110 \\ + \underline{01010100} \\ \hline \text{CF=0} \quad 11101010 \end{array}$	$\begin{array}{r} 11011001 \\ + \underline{01010010} \\ \hline \text{CF=1} \quad 00101011 \end{array}$
--	--

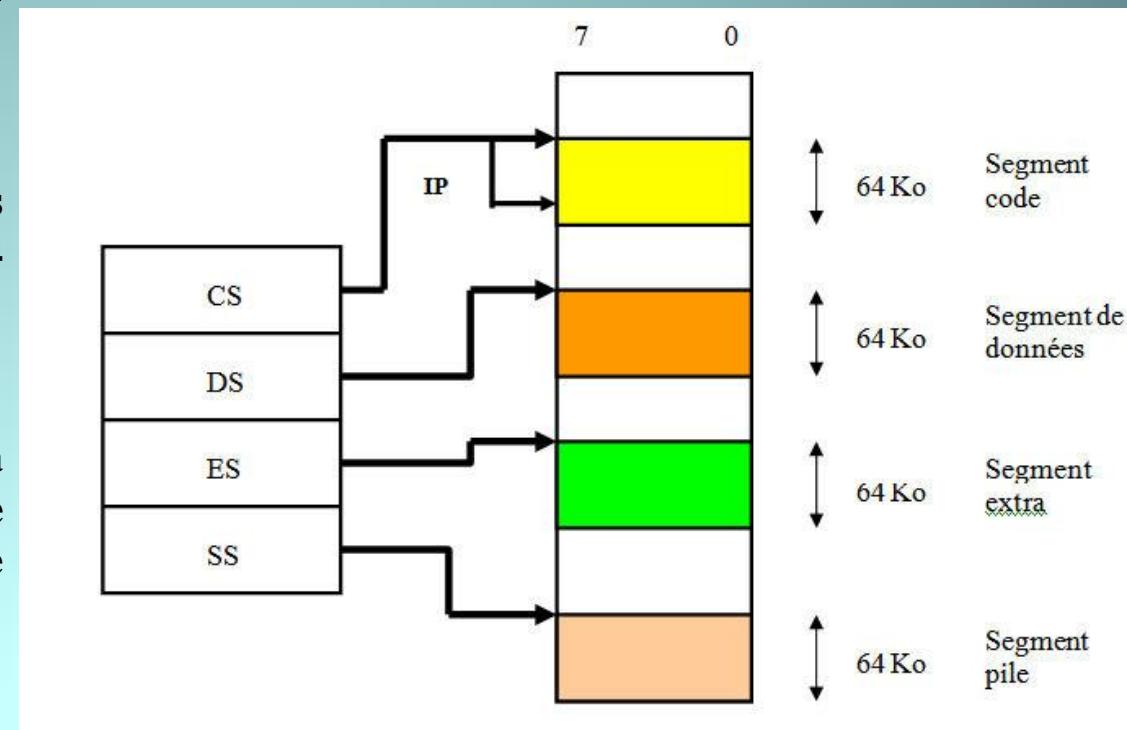
Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → GESTION DES ADRESSES MÉMOIRE du 8088

- L'accès direct et simultané à ces espaces
- le compteur de programme est de 16 bits
→ possibilité d'adressage est de $2^{16} = 64$ Ko → segments logiques de 64 Ko
- L'espace mémoire adressable :
1 méga = 2^{20} → 20 bits du bus d'adresse
- 4 segments logiques ne couvre pas la totalité de la mémoire, → utilisation de deux registres pour indiquer une adresse au processeur



- Chaque segment débute à l'endroit spécifié par le registre segment. Le déplacement (offset) à l'intérieur de chaque segment se fait par un registre de décalage qui permet de trouver une information à l'intérieur du segment. Exemple la paire de registre CS:IP : pointe sur le code d'une instruction (CS registre segment et IP Déplacement)

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

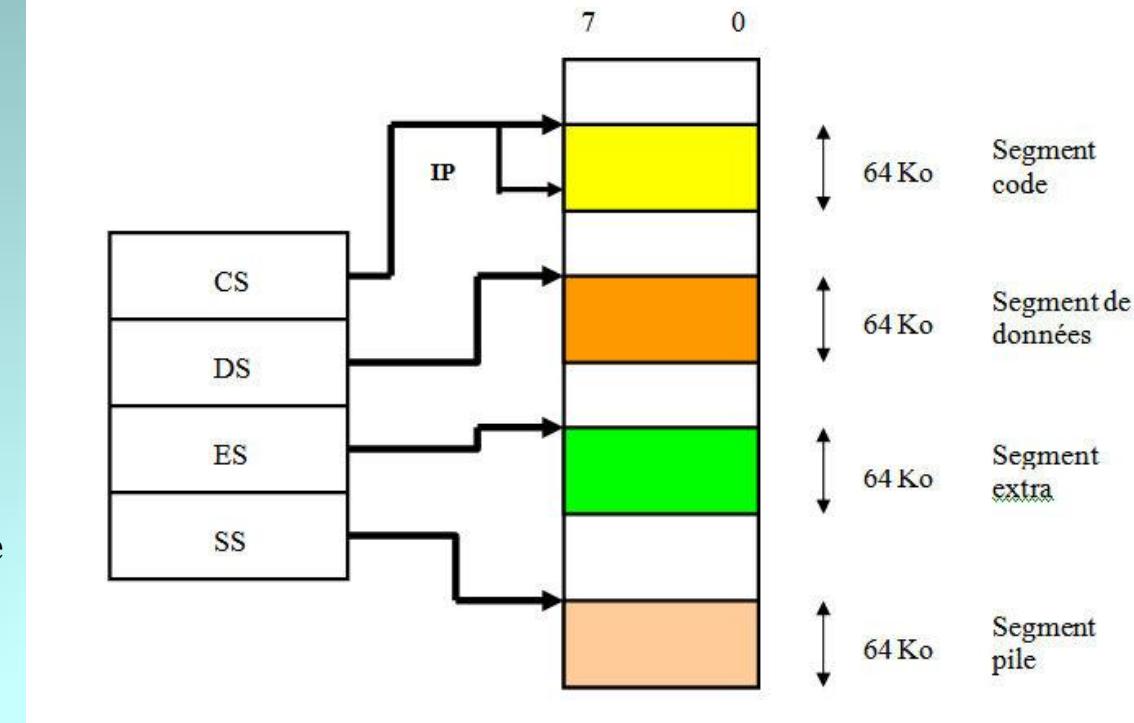
Les µprocesseur CISC d'Intel → GESTION DES ADRESSES MÉMOIRE du 8088

❖ Pas de problème si :

- taille progr. \leq 64Ko et
- données \leq 128 Ko

(sécurité : code et données séparés)

❖ Si la taille du programme est $>$ 64Ko,
le programmeur segmente (découpe) son
application en autant de segments qu'il le
souhaite, dans la limite du Mo disponible



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les microprocesseur CISC d'Intel → GESTION DES ADRESSES MÉMOIRE du 8088

Adresse physique (Segmentation de la mémoire) :

- les données → regroupées dans une zone mémoire nommée ***segment de données DS***
 - les instructions → placées dans un ***segment d'instructions CS***(de même pour le segment pile et segment de données supplémentaires).

⇒ Ce partage se fonde sur la notion plus générale de segment de mémoire, qui est à la base du mécanisme de gestion des adresses par les processeurs 80x86.

- le registre IP de 16 bits → adressage de 64 Ko
 - Le bus d'adresses du 8086 possède 20 bits.

⇒ l'adresse de 20 bits est formée par la juxtaposition d'un registre segment (16 bits de poids fort) et d'un déplacement (*offset*, 16 bits de poids faible).

Adresse physique = Base * 16 + offset

Bits	20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
	Base
+	0 0 0 0
=	Offset
	Adresse physique

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → GESTION DES ADRESSES MÉMOIRE du 8088

Adresse physique (Segmentation de la mémoire) :

Bits	20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
	Base 0 0 0 0
+	Offset
=	Adresse physique

Exemples : *Mots écrits avec le code hexadécimal

Adresse logique ou virtuel	CS→1000	10000
	IP→2006	+2006
Adresse physique →		12006

CS→12C4	12C40
IP→0022	+0022
	12C62

Deuxième partie

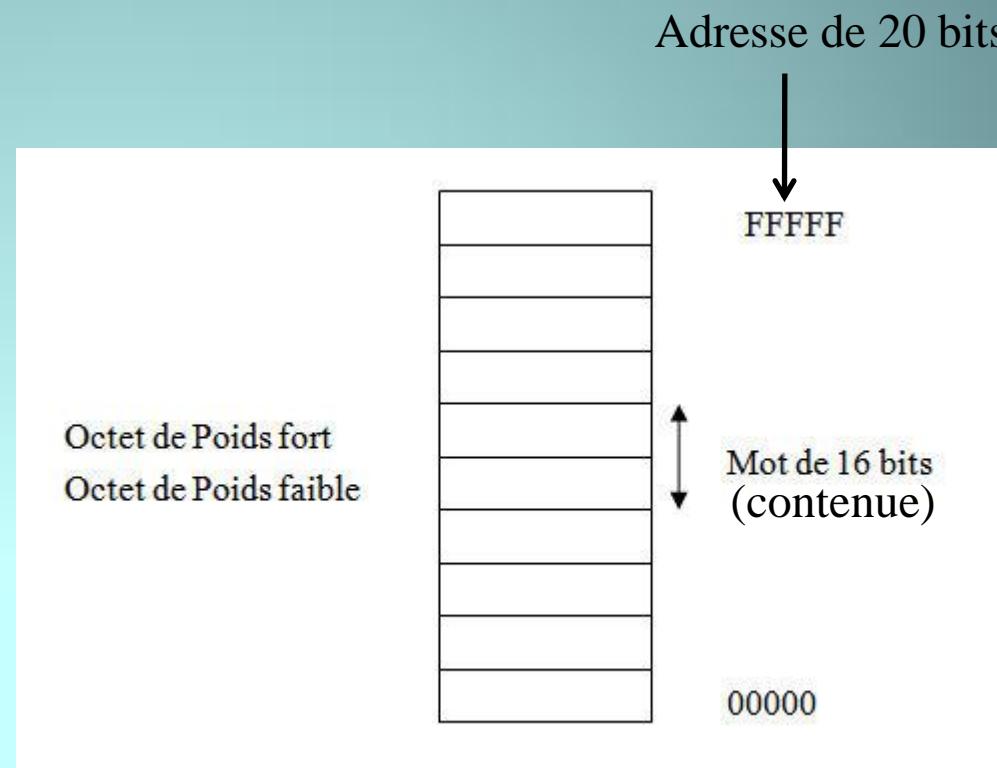
Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → Organisation de la mémoire

1. Organisation logique

Le microprocesseur 8088 est processeur 16 bits (bus de données de 16 bits) → possibilité d'accéder en même temps à deux cases mémoires de 8 bits.



Deuxième partie

Unité centrale du microprocesseur

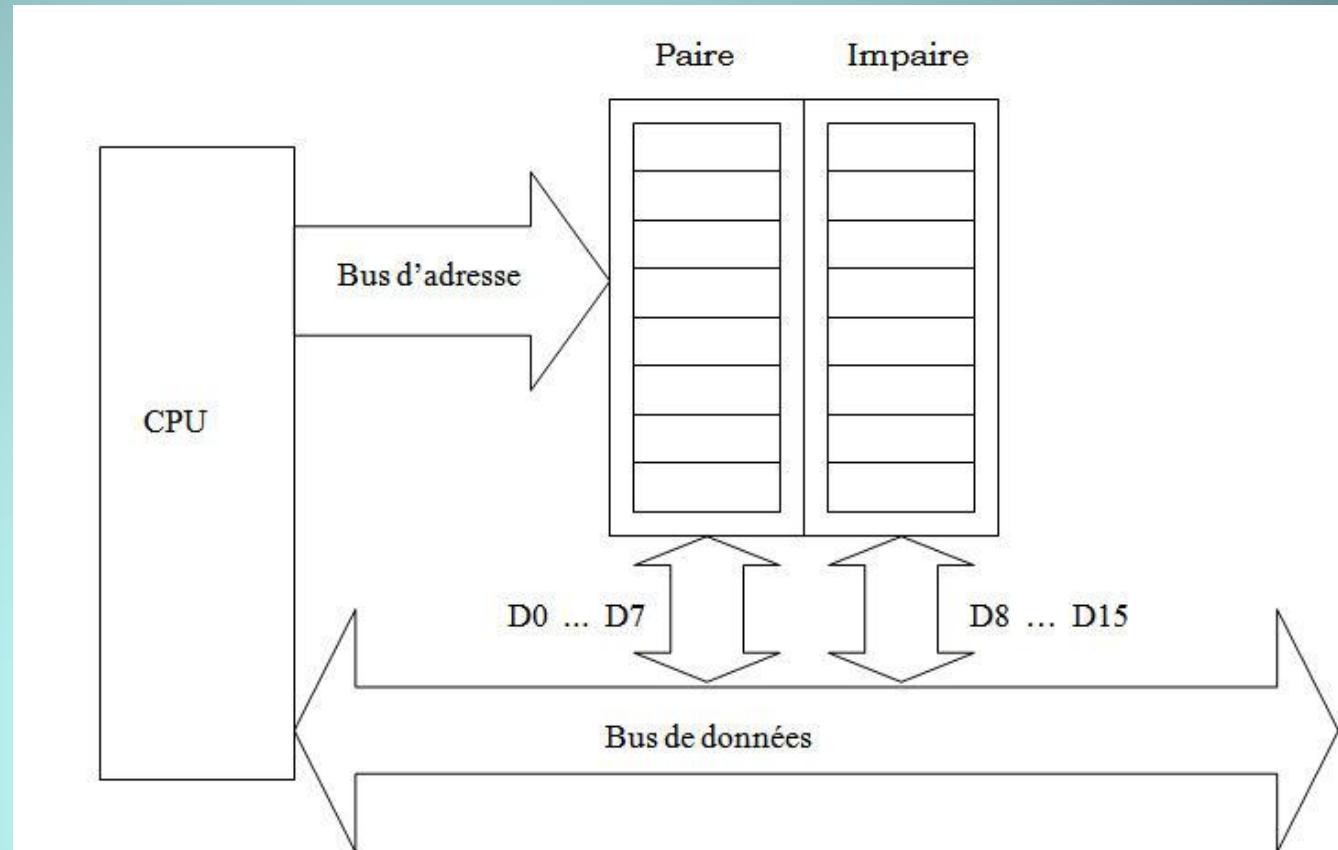
Etude du composant du marché

Les µprocesseur CISC d'Intel → Organisation de la mémoire

1. Organisation physique

⇒la mémoire est organisée en deux Banks chacun de 512 Koctet)

- un bank pair
- un bank impair



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → Organisation de la mémoire

1. Organisation physique

bits D0..D7 → partie base

bits D8..D15 → partie haute

⇒ Le microprocesseur peut charger :

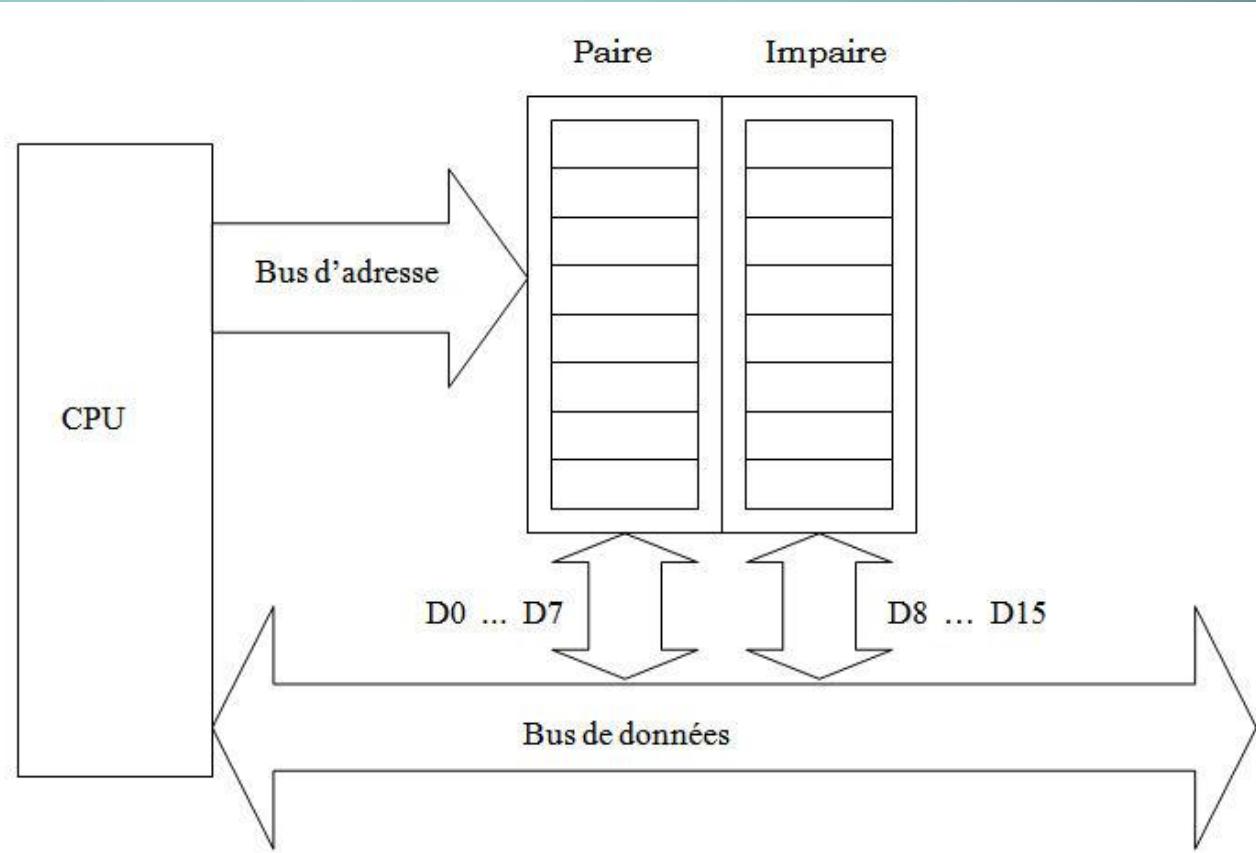
- un octet (8 bits)

→ adresse de l'octet

- un mot (16 bits)

→ cherche l'octet du poids faible à l'adresse donnée et l'octet du poids le plus fort à l'adresse qui suit

- un double mot (32 bits)



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → Organisation de la mémoire

1. Organisation physique

Rangement des cases

Paire	Impaire
octet 6	octet 7
octet 4	octet 5
octet 2	octet 3
octet 0	octet 1

Deuxième partie

Unité centrale du microprocesseur

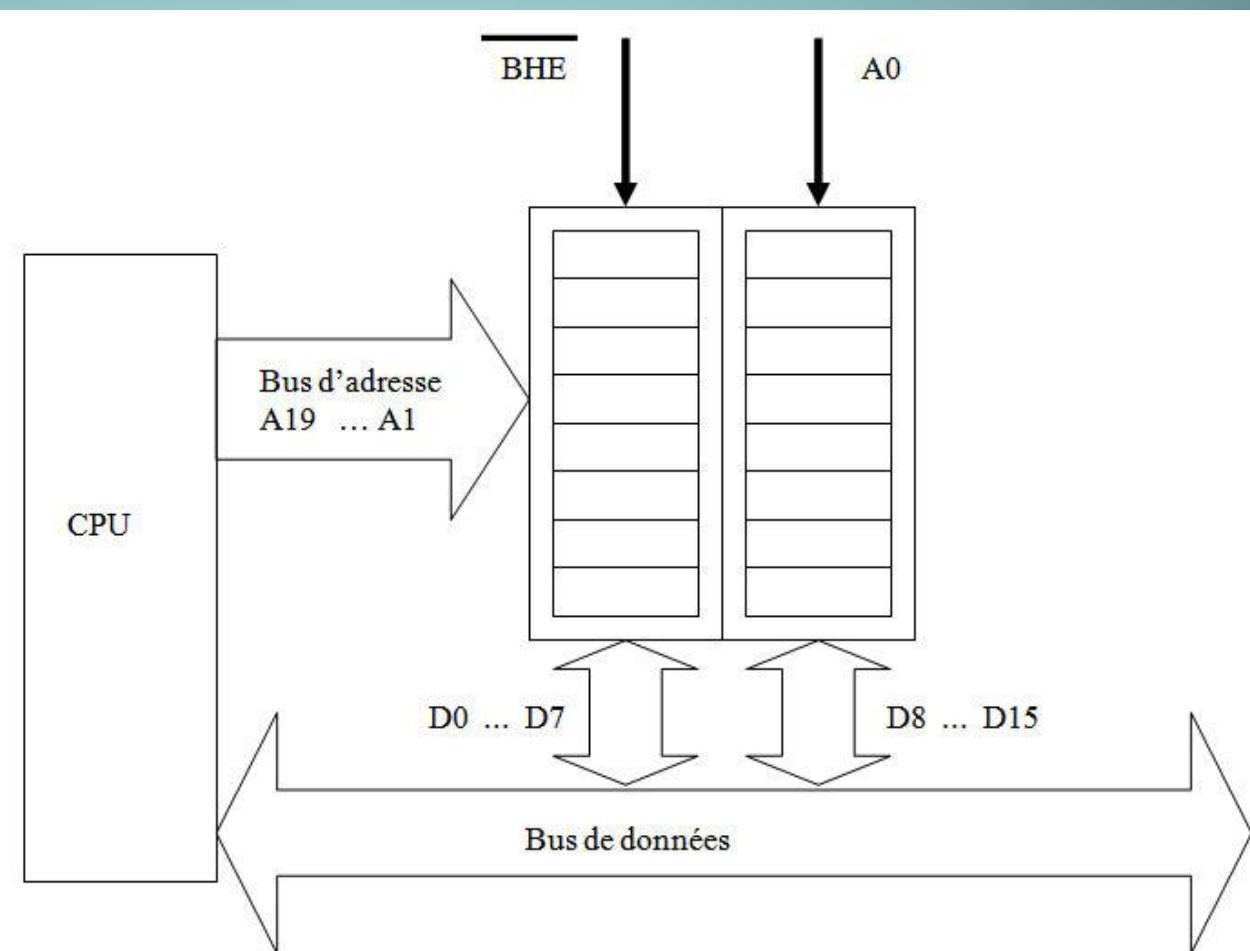
Etude du composant du marché

Les µprocesseur CISC d'Intel → Organisation de la mémoire

1. Organisation physique

- sélectionner les banks pair et impair → le microprocesseur utilise deux signaux

- BHE (BHE = 0)
- A0 : le premier bit du bus d'adresse (A0=1)



Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

Les µprocesseur CISC d'Intel → LES INTERRUPTIONS DU 8086 d'Intel

- ❖ deux types :
 - externes : générées par l'extérieur (périphérique,...)
 - internes au microprocesseur 8086 (trap, déroutement, ...)
- ❖ 3 interruptions *externes* :
 - RESET** : réinitialise le microprocesseur
 - NMI** : Non Masquable Interrupt
 - INTR** : interruption prise en compte ou non selon l'état du bit I (Interrupt Enable Flag) du registre d'état
- ❖ 2 sortes d'interruptions *internes* :
 - les automatiques (div./0, d.d.c., pas à pas)
 - les logicielles (les autres) : SVC
- ❖ Toute interruption acceptée provoque l'exécution d'un sous-programme (*handler d'interruption*) démarrant à une adresse spécifique à chaque interruption et contenue dans une table
 - 1 élément de la table = 4 octets
 - CS ← les deux premiers octets (adresse de base)
 - IP ← les deux derniers octets (déplacement)
- ❖ Les adresses 00000H à 003ffH sont réservées à la table , dite table des interruptions (256 interruptions * 4 octets = 1Ko)

Deuxième partie

Unité centrale du microprocesseur

Etude du composant du marché

jeu d'instructions du 8086/8088 d'Intel

Les instructions de transfert de données

Usage	Nom	Fonction
Général	MOV	Transfert d'octets ou de mots
	PUSH	Chargement de la pile
	POP	Déchargement de la pile
	PUSHA	Chargement de tous les registres dans la pile
	POPA	Déchargement de tous les registres
	XCHG	dans la pile Echange d'octet ou de mot
	XLAT	Translation d'octet
Entrées-sorties	IN	Entrée de mot ou d'octet
	OUT	Sortie de mot ou d'octet
Adresses	LEA	Chargement de l'adresse effective
	LDS	Chargement du pointeur avec DS
	LES	Chargement du pointeur avec ES
Indicateurs	LAHF	Transfert des indicateurs dans AH
	SAHF	Rangement de AH dans les indicateurs
	PUSHP	Chargement des indicateurs dans la pile
	POPF	Déchargement des indicateurs de la pile.

Instructions arithmétiques :

Les instructions arithmétiques peuvent manipuler quatre types de nombres :

- * Les nombres binaires non signés
- * Les nombres binaires signés.
- * Les nombres décimaux codés binaires (DCB), non signés.
- * Les nombres DCB non condensés, non signés.

Usage	Nom	Fonction
Addition	ADD ADC INC AAA DAA	Addition sur un octet ou un mot Addition sur un octet ou un mot avec retenue Incrémantation de 1 Ajustement ASCII Ajustement décimal
Soustraction	SUB SBB DEC NEG CMP AAS DAS	Soustraction sur un octet ou un mot Soustraction sur un octet (mot) avec retenue Décrémentation de 1 Mètre un octet ou un mot en négatif Comparaison d'octet ou mot Ajustement ASCII Ajustement décimal
Multiplication	MUL IMUL AAM	Multiplication d'octet ou de mot <u>non signée</u> Multiplication d'octet ou de mot <u>signée</u> Ajustement ASCII
Division	DIV IDIV AAD CBW CWD	Division d'octet ou de mot <u>non signée</u> Division d'octet ou de mot <u>signée</u> Ajustement ASCII Conversion d'un octet en un mot Conversion d'un mot en double mots

Les instructions logiques (de bits)

Usage	Nom	Fonction
Logique	NOT AND OR XOR TEST	Inversion logique sur un octet ou un mot Et logique Ou logique Ou exclusif Et logique sans résultat, affecte uniquement les indicateurs du registre des flags.
Décalages	SHL SAL SHR SAR	Décalage logique à gauche Décalage arithmétique à gauche Décalage logique à droite Décalage arithmétique à droite
Rotation	ROL ROR RCL RCR	Rotation à gauche Rotation à droite Rotation à gauche à travers le bit de retenue Rotation à droite à travers le bit de retenue

Instructions de sauts de programme

Elles permettent de faire des sauts dans l'exécution d'un programme (rupture de séquence)

Remarque :

Ces instructions n'affectent pas les Flags. Dans cette catégorie on trouve toutes les instructions de branchement, de boucle et d'interruption après un branchement

Type	Nom	Fonction
Branchements inconditionnels	CALL RET JMP	Appel à un sous programme Retour d'un sous programme Saut
Branchements conditionnels (arithmétique non signée)	JA/JNBE JAE/JNB JB/JNAE JBE/JNA	Si supérieur / Si non inférieur ou non égal Si supérieur ou égal/ Si non inférieur Si inférieur/si non supérieur ni égal Si inférieur ou égal/si non supérieur.
Branchements conditionnels (arithmétique signée)	JG/JNLE JGE/JNL JL/JNGE JLE/JNG	Si plus grand/si pas inférieur ni égal Si plus grand ou égal/Si pas inférieur Si moins que/Si pas plus grand ni égal Si moins que ou égal/Si pas plus grand
Branchement conditionnels (flags)	JC JE/JZ JNC JNE/JNZ JNO JNP/JPO JNS JO JP/JPE JS	Si retenue Si égal/Si zéro Si pas de retenue Si non égal / Non zéro Si pas de débordement Si pas de parité/ Si parité impaire Si pas de signe Si débordement Si parité / Si parité paire Si signe (négatif)
Boucles	LOOP LOOPE/LOOPZ LOOPNE/LOOPNZ JCXZ	Boucle Boucle si égal/Si zéro Boucle si différent/si diff 0 Branchement si CX=0
Interruptions	INT INTO IRET	Interruption Interruption si débordement Retour d'interruption.

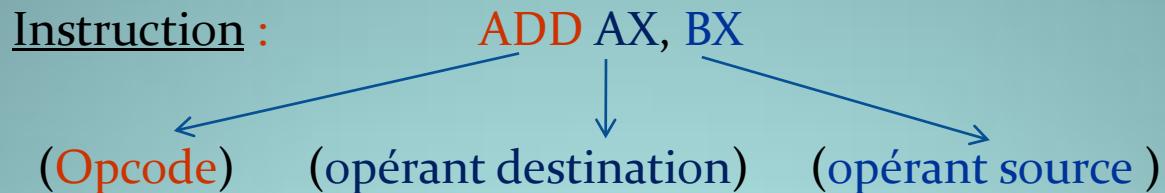
Les instructions de chaînes de caractères

Nom	Fonction
REP	Préfixe de répétition
REPE/REPZ	Répétition tant qu'égal à zéro
REPNE/REPNZ	Répétition tant que différent de zéro
MOVS	Déplacement de chaîne
MOVSB/MOVSW	Déplacement de chaîne
CMPS	Comparaison de chaînes
INS	Entrée (de port d'E/S)
OUTS	Sortie (vers un port d'E/S)
SCAS	Balayage d'une chaîne
LODS	Chargement de chaîne
STOS	Rangement de chaînes

Les instructions de commande du processeur

Type	Nom	Fonction
Indicateur (FLAGS)	STC	Met à 1 la retenue CF
	CLC	MET à 0 la retenue CF
	CMC	Complément la retenue
	STD	Met à 1 la direction DF
	CLD	Met à 0 la direction DF
	STI	Met à 1 l'autorisation d'interruption
	CLI	Met à 0 l'autorisation d'interruption
Synchronisation	HLT	Halte jusqu'à interruption ou RESET
	WAIT	Attente jusqu'à broche TEST passe à 0
	ESC	Pour un coprocesseur
	LOCK	Verrouillage des bus pendant la prochaine instructions
Sans opération	NOP	Pas d'opération

Codage des instructions du 8086/8088 d'Intel



LABEL: INSTRUCTION ; COMMENTAIRE

Identificateur d'adresse

aucun code machine généré

Ex. START: MOV AX, BX ; copy BX into AX

⇒ Programmation avec « Assembleur » → moins de mémoires nécessaires
→ plus rapide
→ applications à temps réels

Codage des instructions du 8086/8088 d'Intel

❖ Conversion des instructions d'« Assembleur » aux codes machines



➤ Un code machine est formé par 1 à 6 bytes

➤ Premier byte contient 3 informations

- **Opcode** (6 bits) spécifie l'opération

- Direction du registre (bit **D**) informe que la donnée REG dans byte 2 est l'opérant source ou l'opérant destination

 → 1 : destination → 0 : source

- Dimension de donnée (bit **W**) : 8 bits ou 16 bits

 → 0 : 8 bits → 1 : 16 bits

➤ Deuxième byte contient 3 informations :

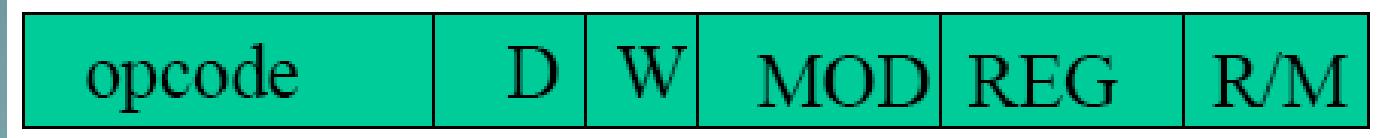
- Mode (**MOD**) 2 bits

- Registre (**REG**) 3 bits: identification du registre cité dans byte 1

- Registre ou mémoire (**Register/Memory**) 3 bits

Codage des instructions du 8086/8088 d'Intel

❖ Conversion des instructions d'« Assembleur » aux codes machines

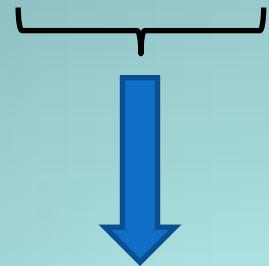


REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Codage des instructions du 8086/8088 d'Intel

❖ Conversion des instructions d'« Assembleur » aux codes machines

opcode	D	W	MOD	REG	R/M
--------	---	---	-----	-----	-----



MOD	Commentaires
00	Mode mémoire sans décalage <i>Exceptionnel quand R/M = 110 donc 16 bits avec décalage</i>
01	Mode mémoire 8 bits avec décalage
10	Mode mémoire 16 bits avec décalage
11	Mode registre sans décalage

mmm : Function	rrr : W=0	rrr : W=1	rrr : reg32	sss : Segment Register	rrr : Index Register
000 : DS:[BX+SI]	000 : AL	000 : AX	000 : EAX	000 : ES	000 : EAX
001 : DS:[BX+DI]	001 : CL	001 : CX	001 : ECX	001 : CS	001 : ECX
010 : SS:[BP+SI]	010 : DL	010 : DX	010 : EDX	010 : SS	010 : EDX
011 : SS:[BP+DI]	011 : BL	011 : BX	011 : EBX	011 : DS	011 : EBX
100 : DS:[SI]	100 : AH	100 : SP	100 : ESP	100 : FS (Only 386+)	100 : No Index
101 : DS:[DI]	101 : CH	101 : BP	101 : EBP	101 : GS (Only 386+)	101 : EBP
110 : SS:[BP]	110 : DH	110 : SI	110 : ESI		110 : ESI
111 : DS:[BX]	111 : BH	111 : DI	111 : EDI		111 : EDI

Codage des instructions du 8086/8088 d'Intel

❖ Exemples

MOV BL,AL (88C3₁₆)

Opcode pour MOV = 100010

D = 0 (AL opérant source)

W bit = 0 (8-bits)

Donc byte 1 est $10001000_2 = 88_{16}$

- MOD = 11 (mode registre)
 - REG = 000 (code pour AL)
 - R/M = 011 (destination is BL)

Donc Byte 2 est $11000011_2 = C3$

Codage des instructions du 8086/8088 d'Intel

❖ Exemples

ADD AX,[SI] (0304₁₆)

Opcode pour ADD = 000000

D = 1 (AX opérant destination)

W bit = 1 (16 bits)

Donc byte 1 est 00000011₂=03₁₆

- MOD = 00 (mode mémoire)
 - REG = 000 (code pour AX)
 - R/M = 100 (destination est SI)

Donc Byte 2 est $00000100_2 = 04_{16}$

Codage des instructions du 8086/8088 d'Intel

❖ Exemples

PUSH CS $(0E_{16})$

Opcode pour PUSH = 00sss110 pour segment

sss = 001 pour segment CS

Donc byte 1 est $00001110_2 = 0E_{16}$

INC BX (43_{16})

Opcode pour INC = 01000rrr pour mot du registre

rrr = 011 pour registre BX

Donc byte 1 est $01000011_2 = 43_{16}$

Codage des instructions du 8086/8088 d'Intel

❖ Exemples

MOV AX,[0413] (A11304₁₆)

le mot situé à l'adresse DS:0413 est chargé dans AX

Opcode pour MOV = 101000 (Accumulateur \leftarrow mémoire)

D = 0 (REG dans byte 2 est l'opérant source)

W = 1 (16 bits)

Donc byte 1 est 10100001₂=A1₁₆

Adresse basse 13, adresse haute 04

Donc byte 2 est 1304₁₆ (octets adresse inversés)

Codage des instructions du 8086/8088 d'Intel

❖ Exemples

MOV BX,[7DF9] (8B1EF97D₁₆)

le mot situé à l'adresse (directe) DS:7DF9 est chargé dans BX

Opcode pour MOV = 100010 (registre \leftarrow mémoire)

D = 1 (REG dans byte 2 est l'opérant destination)

W = 1 (16 bits)

Donc byte 1 est $10001011_2 = 8B_{16}$

MOD = 00 (mémoire)

REG = 011 (pour BX)

R/M = 110 (SS:BP \rightarrow adressage direct) \Rightarrow byte 3 : adresse de l'opérant source

Donc byte 2 est $(00011110_2) = (1E_{16})$

byte 3 (adresse basse) et byte 4 (adresse haute) sont F97D

opcode	D	W	MOD	REG	R/M
--------	---	---	-----	-----	-----

Mémoire cache

- la lecture et l'écriture d'informations
- Ce sont des RAM (Random Access Memory) mémoire dont le temps d'accès à l'information est le même quelque soit le mot sollicité ⇒ mémoires vives

Deux familles de RAM: statiques et dynamiques

- Les *RAM statiques* (étudiées en cours) semblables aux **bascules D**
⇒ garantir la mémorisation de l'information aussi longtemps que l'alimentation électrique est maintenue sur la mémoire
- Les *RAM dynamiques* sont composées d'un ensemble de petits **condensateurs**, chacun pouvant recevoir ou restituer une charge électrique emmagasinée.
Inconvénient: la charge électrique mémorisée diminue avec le temps
⇒ les rafraîchir une fois toutes les quelques millisecondes → disposer d'un dispositif interne auto rafraîchissement: *les mémoires quasi-statiques.*

Mémoire cache

- ❖ Les unités centrales deviennent beaucoup plus rapides que les mémoires principales. Ainsi, lorsque l'unité centrale sollicite la mémoire, elle passe une bonne partie de son temps à attendre que la mémoire réagisse.
- ❖ Le microprocesseur doit donc "attendre" la mémoire vive à chaque accès, on dit que l'on insère des "Wait State" dans le cycle d'horloge d'un micro.

Ex: un 386 DX cadencé à 33 MHz ne peut fonctionner sans état d'attente que si les RAM ont un temps d'accès de 40 ns.

- ❖ le problème **technologique : NON**
 économique : Oui

Il est possible de construire des mémoires aussi rapides que les unités centrales mais leur coût, pour des capacités de plusieurs mégaoctets serait prohibitif.

- ❖ Les choix : à l'exception des superordinateurs ($performance > coût$)
→ disposer d'une faible quantité de mémoire rapide associée à une quantité importante de mémoire relativement plus lente. Cette mémoire plus rapide est appelée **mémoire cache, cache ou antémémoire**.

Mémoire cache

❖ **mémoire cache → mémoire vive statique**

15 ns à 20 ns de temps d'accès

s'insère entre le processeur et la RAM dynamique.

❖ **Un contrôleur de mémoire cache est chargé de recopier les instructions et les données les plus fréquemment utilisées par le processeur dans le cache.**

❖ **Le principe du cache repose sur deux constatations:**

- en cours d'exécution d'un programme, lorsque le microprocesseur va chercher une instruction en mémoire il y a statistiquement de fortes chances pour que celle-ci se trouve à proximité de l'instruction précédente.

- de plus, les programmes contiennent un grand nombre de structures répétitives de sorte qu'ils utilisent souvent les mêmes adresses.

❖ **Gestion de la mémoire cache**

le contrôleur du cache intercepte les adresses émises par le microprocesseur et dans un premier temps recopie le contenu d'un bloc entier de mémoire dans le cache de sorte qu'il y ait une forte probabilité que la mémoire cache contienne les prochaines instructions.

Ce principe permet au microprocesseur, via le contrôleur de cache, d'avoir 90% de chance d'obtenir l'information dans la mémoire cache donc sans "Wait State".

Mémoire cache

Principe de fonctionnement

- ❖ Le dispositif est constitué de deux éléments principaux:
 - la mémoire cache, constituée de RAM
 - le contrôleur de mémoire cache comprenant
 - la gestion du cache
 - un index d'adresses stockant les adresses des blocs contenus dans le cache
 - un comparateur qui met en correspondance les adresses émises par le microprocesseur et celles contenues dans l'index quand il y a correspondance
- ❖ l'information est prise dans le cache sinon elle est transférée de la RAM et le contrôleur recopie tout un bloc dans le cache.

Mémoire cache

Différentes étapes du fonctionnement d'un cache:

1. Le microprocesseur demande une information (instruction ou donnée) I1 située à l'adresse A1 de la mémoire vive
2. Le contrôleur de mémoire cache intercepte la demande et examine sa table d'index pour vérifier si A1 y est présente, et donc si une copie de l'information se trouve dans le cache.
3. Si c'est le cas, l'information est délivrée au microprocesseur depuis la mémoire cache sans temps d'attente
4. Dans le cas contraire, le contrôleur de cache accède à l'information de A1 de la RAM, la délivre à l'unité centrale avec plusieurs temps d'attente et simultanément la recopie en mémoire cache en même temps qu'un bloc d'informations contigüës, parmi lesquelles I2, I3, I4 etc... et actualise sa table d'index.
5. Le microprocesseur réclame l'information suivante, il y a statistiquement 90% de chance pour que ce soit I2, donc présente dans le cache et délivrée dans l'UC sans temps d'attente.

Mémoire cache

Exemple: le cache du 486 (cache interne, cache externe)

Avec la génération **80486** et **68040** le contrôleur et la mémoire cache sont intégrés dans le processeur, on parle de cache premier niveau. Un second cache, de deuxième niveau, peut lui être associé en externe.

Le **486** possède un cache interne de 8 ko fonctionnant à la même vitesse que le processeur, ce qui n'est pas le cas par rapport aux caches externes.

Power PC 620: (64 ko cache interne) peut gérer 1 Mo de cache externe