ENICARTHAGE

المدرسة الوطنية للمهندسين بقرطاج

Ecole Nationale d'Ingénieurs de Carthage

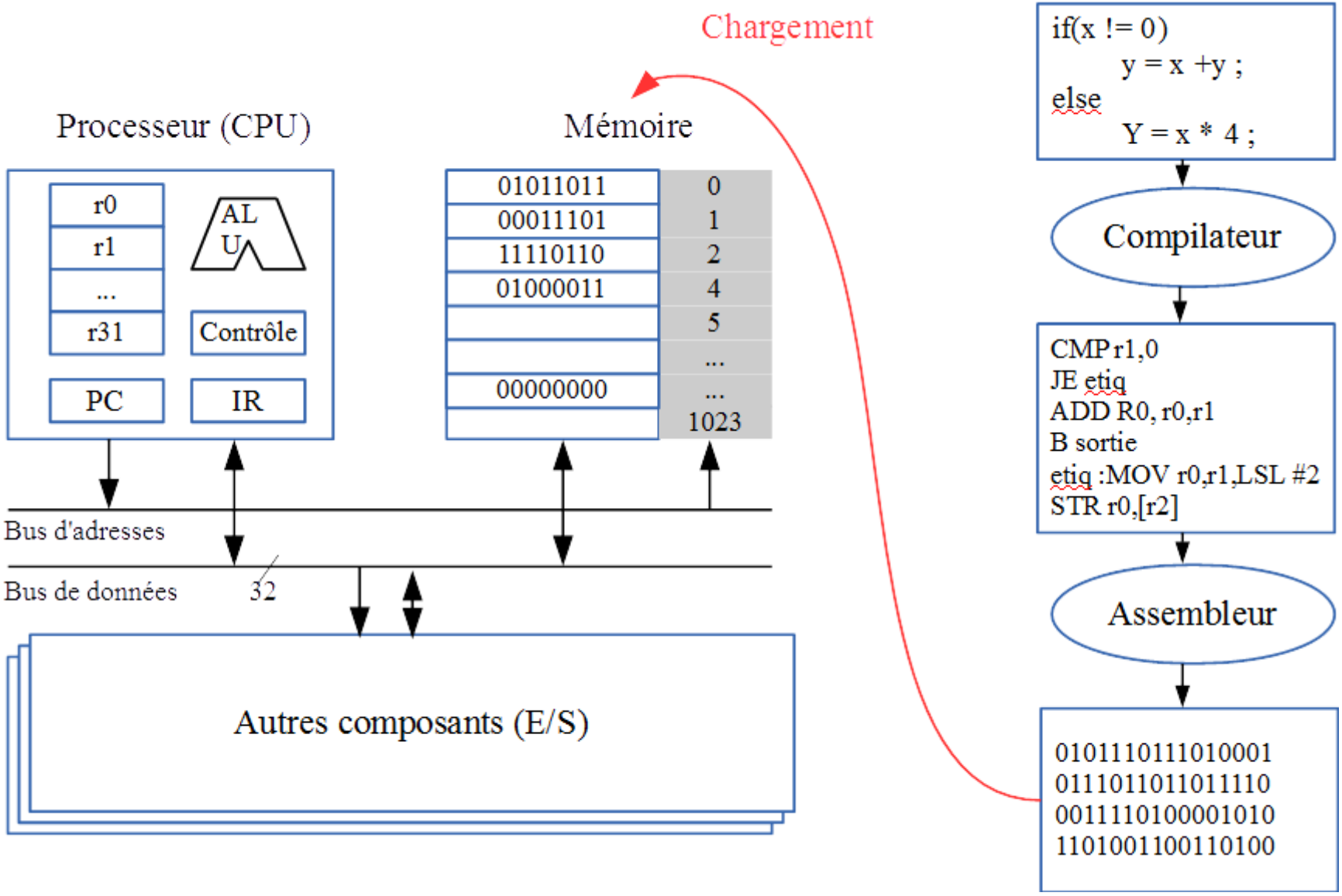
MODULE: Architecture des systèmes à microprocesseurs

CHAPITRE III:

LES MICROPROCESSEURS

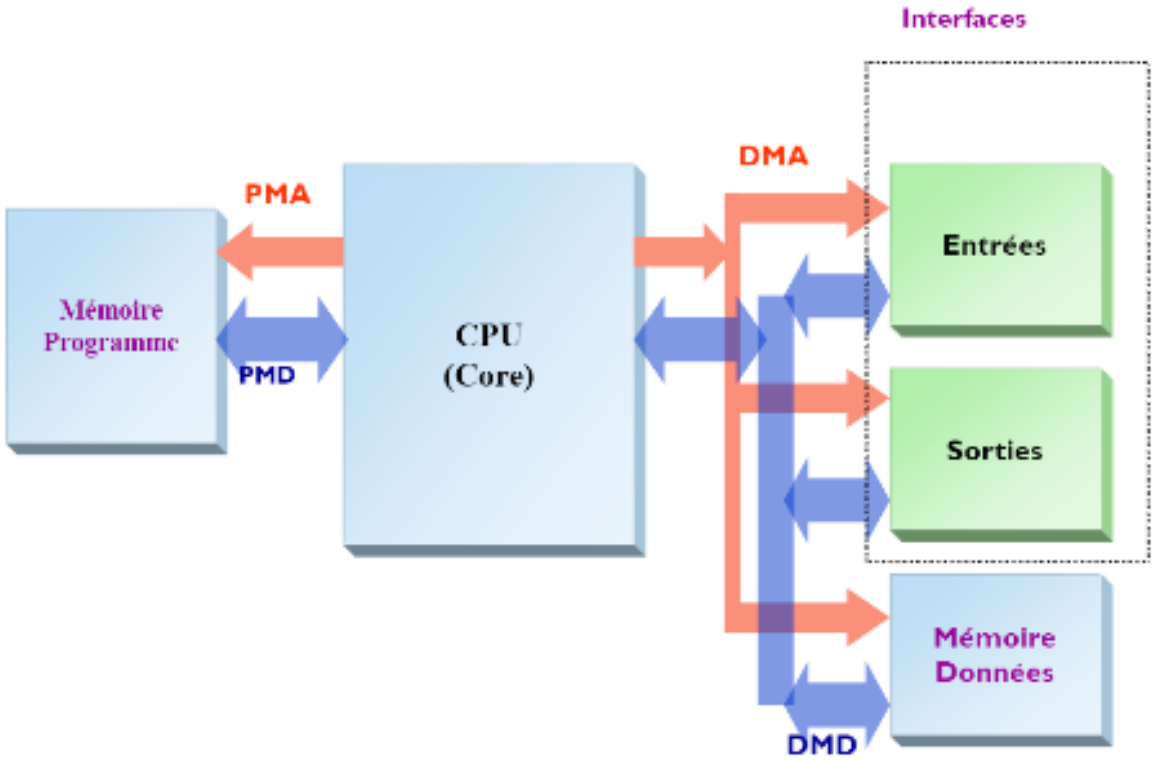
2018-2019

Machine de Von Neumann



Von Neumann vs Harvard

- **Architecture de Von-Neumann :** Une seule mémoire pour les instructions et les données
- **Architecture de Harvard:** mémoire instruction et mémoire de données séparées.

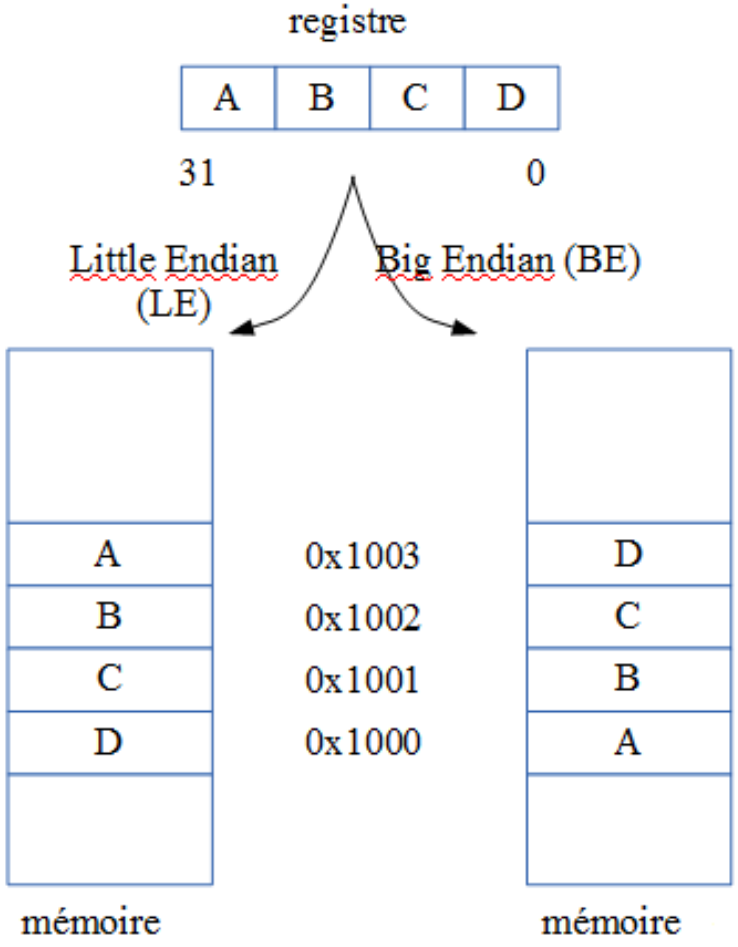


L'ENDIANISME (BOUTISME)

Comment les octets d'un entier formé de plusieurs octets sont rangés en mémoire:

- **Big Endian (BE) :** les octets de poids le plus fort occupent les adresses les plus basses

- **Little Endian (LE):** les octets de poids le plus fort occupent les adresses les plus hautes



Fonctionnement d'un Microprocesseur

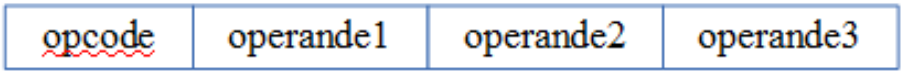
Un microprocesseur exécute un programme. Le programme est une suite d'instructions stockées dans la mémoire. Une instruction peut être codée sur un ou plusieurs octets.

Instruction: OpCode + { opérandes }

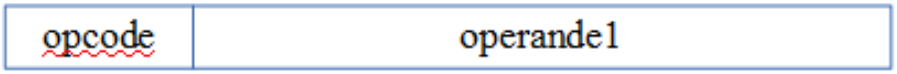
Opérande = Valeur immédiate | Registre | mémoire ...

OpCode : code opération

Instruction Type 1
Ex : ADD r0,r1,r2

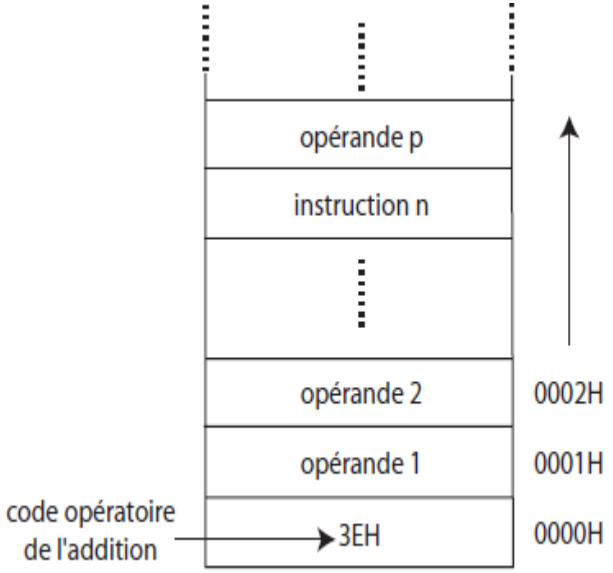


Instruction Type 2
Ex : B etiquette

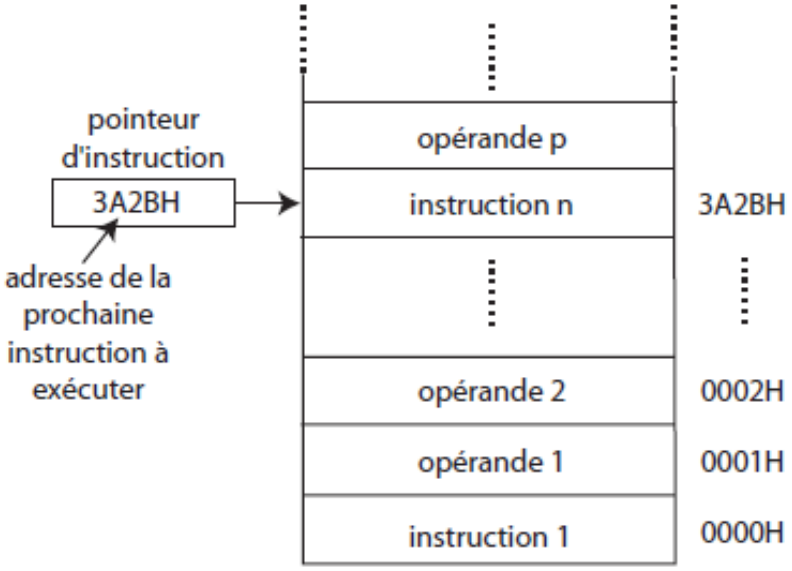


Rangement en mémoire

Pour exécuter les instructions dans l'ordre établi par le programme, le microprocesseur doit savoir à chaque instant l'adresse de la prochaine instruction à exécuter. Le microprocesseur utilise un registre contenant cette information. Ce registre est appelé *pointeur d'instruction (IP : Instruction Pointer) ou compteur d'instructions ou compteur ordinal.*



Exemple



Remarque : la valeur initiale du pointeur d'instruction est fixée par le constructeur du microprocesseur. Elle vaut une valeur bien définie à chaque mise sous tension du microprocesseur ou bien lors d'une remise à zéro (reset).

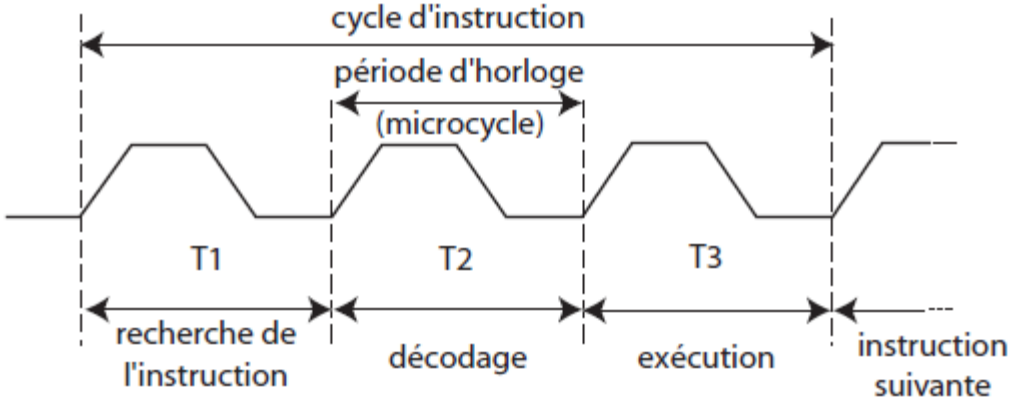
Pour savoir quel type d'opération doit être exécuté (addition, soustraction, ...), le microprocesseur lit le premier octet de l'instruction pointée par le pointeur d'instruction (code opératoire) et le range dans un registre appelé registre d'instruction. Le code opératoire est décodé par des circuits de décodage contenus dans le microprocesseur. Des signaux de commande pour l'UAL sont produits en fonction de l'opération demandée qui est alors exécutée.

Le processeur exécute en boucle infinie la séquence suivante:

1) Chercher (Fetch) l'instruction en cours: l'adresse est contenue dans le registre pointeur d'instruction (registre compteur de programme PC). On aura directement la mise à jour PC pour pointer vers l'instruction suivante.
2) Décoder l'instruction (registre IR) pour connaître sa nature
3) Exécuter l'instruction:

✓L'exécution peut nécessiter la lecture d'une ou plusieurs données de la mémoire

✓L'exécution peut entraîner la mise à jour d'un ou plusieurs registres internes ou une valeur en mémoire



Chaque instruction est caractérisée par le nombre de périodes d'horloge qu'elle utilise .

Exemple : horloge à 5 MHz, période T = 1/f = 0,2 µs. Si l'instruction s'exécute en 3 microcycles, la durée d'exécution de l'instruction est : 3 × 0,2 = 0,6 µs.

L'horloge est constituée par un oscillateur à quartz dont les circuits peuvent être internes ou externes au microprocesseur.

Jeu d'instructions ISA

➢ **ISA** (Instruction set architecture) :
✓ Information nécessaire pour pouvoir programmer un processeur.
✓ Manuel de référence (*datasheet) utilisé par : l*e compilateur, le système d'exploitation et aussi par le programmeur "bas niveau" (en assembleur)

➢ L'ISA décrit essentiellement :
✓ Le nombre et la nature des registres internes du processeur
✓ L'ensemble des instructions qui sont supportées par le processeur

➢ L'ISA ne dit rien (en principe) sur la réalisation (implémentation) du processeur
✓ Deux processeurs différents peuvent implémenter le même ISA : exemple core i5 et core i7 sont deux implémentations différentes du même ISA (x86)
✓ Une ISA peut être vue comme une famille de processeurs

CISC vs RISC

- **CISC** : Complex Instruction Set Computer

✓ Jeu d'instructions riche (le nombre d'instructions reconnues généralement varie entre 75 et 150)

✓La même instruction peut avoir plusieurs formes

- RISC : Reduced Instruction Set Computer

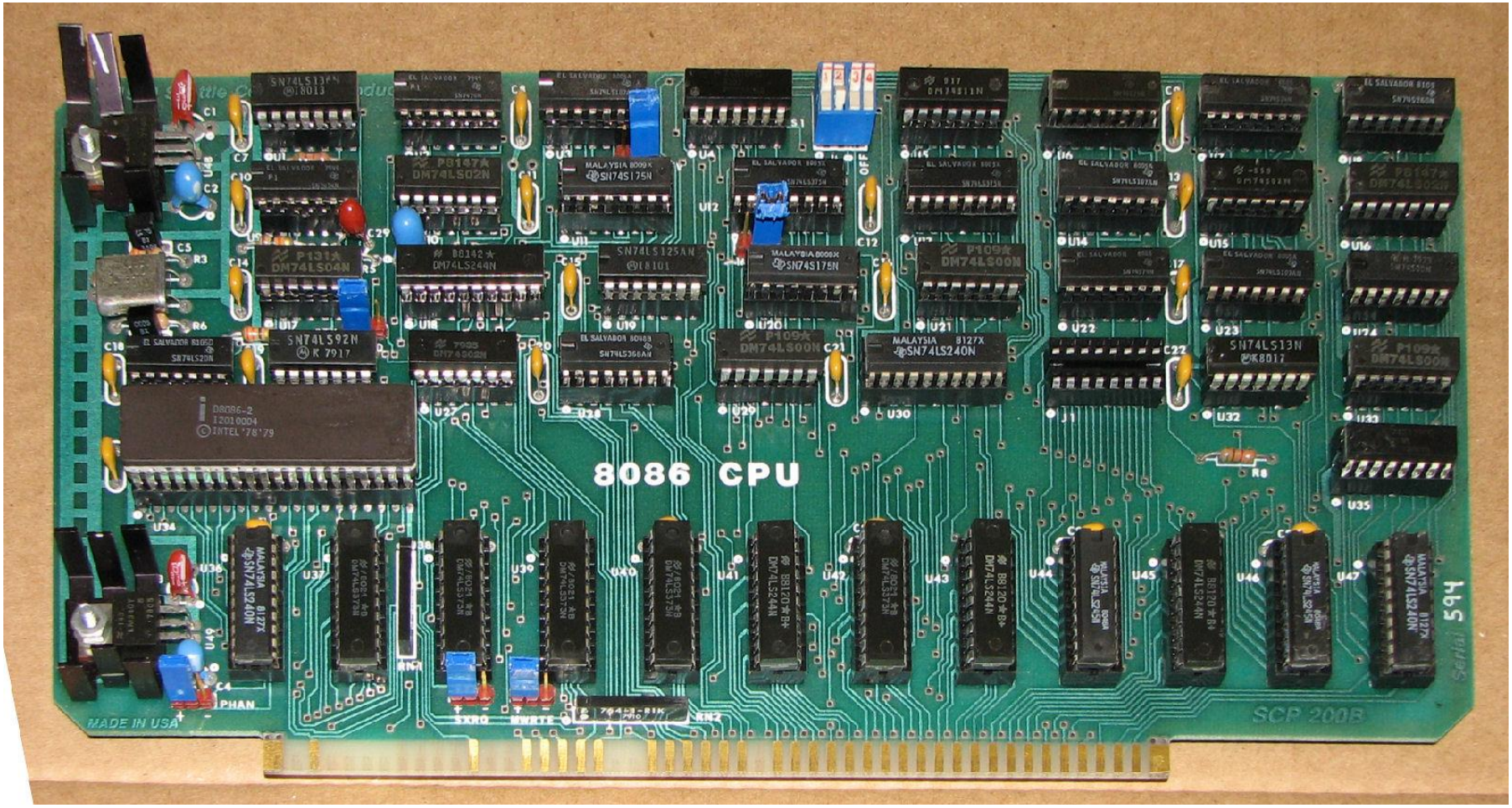
✓ Jeu d'instruction « réduit » (généralement entre 10 et 30 instructions)

✓ Machine load/store: l'accès à la mémoire se fait exclusivement via les instructions LOAD et STORE, tandis que les autres opérations n'accèdent pas à la mémoire

✓Architecture plus simple avec des instructions de taille fixe (plus simple à décoder), théoriquement plus rapide et un nombre de registres plus élevé (plus d'espace disponible)

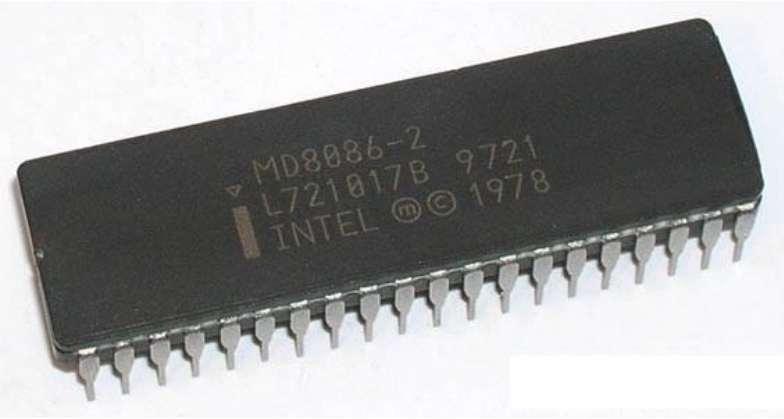
Architecture CISC	Architecture RISC
Un jeu étendu d'instructions complexes nécessitant plusieurs cycles d'horloge. Chacune de ces instructions peut effectuer plusieurs opérations élémentaires.	Un jeu d'instructions réduit où chaque instruction effectue une seule opération élémentaire (nécessitant un seul cycle).
Décodeur complexe (microcode)	Décodeur simple (câblé)
Peu de registres	Beaucoup de registres
Toutes les instructions sont susceptibles d'accéder à la mémoire	Seules les instructions LOAD et STORE ont accès à la mémoire
Beaucoup de mode d'adressage	Peu de mode d'adressage
Compilateur simple	Nécessite un compilateur très évolué dans le cas de programmation en langage de haut niveau.

Exemple de CISC: Intel 8086



Description Physique du 8086

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



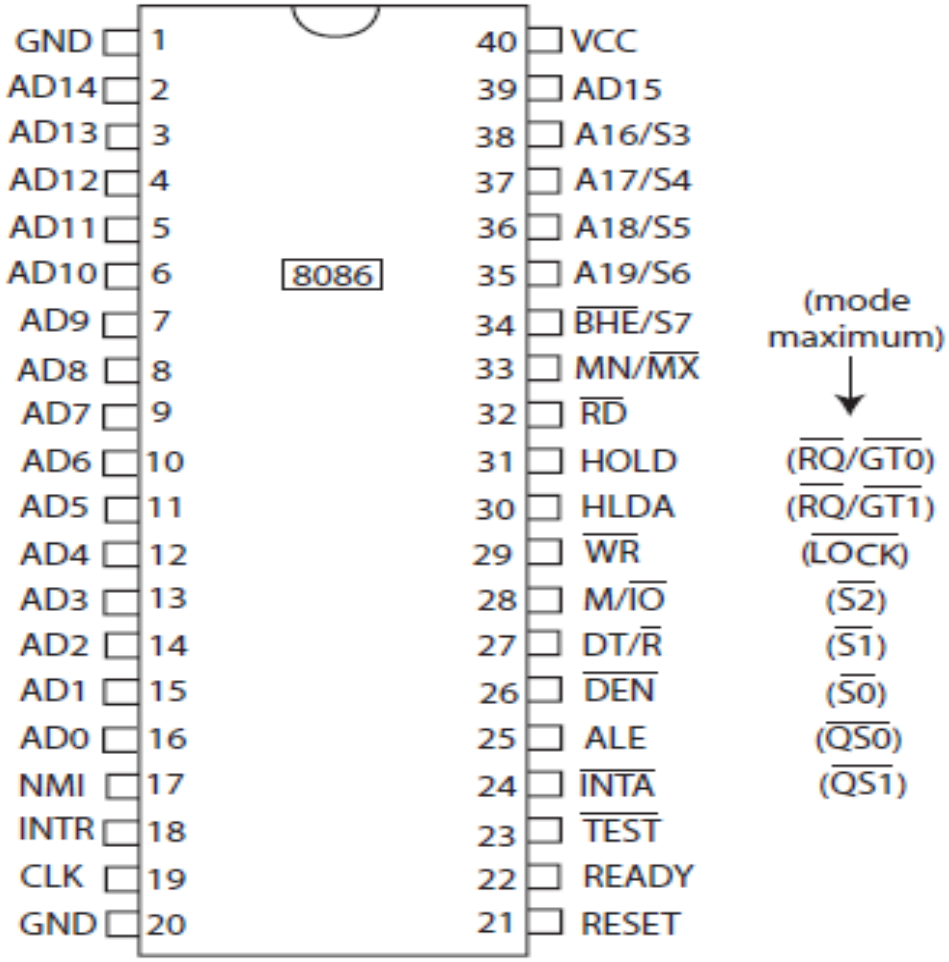
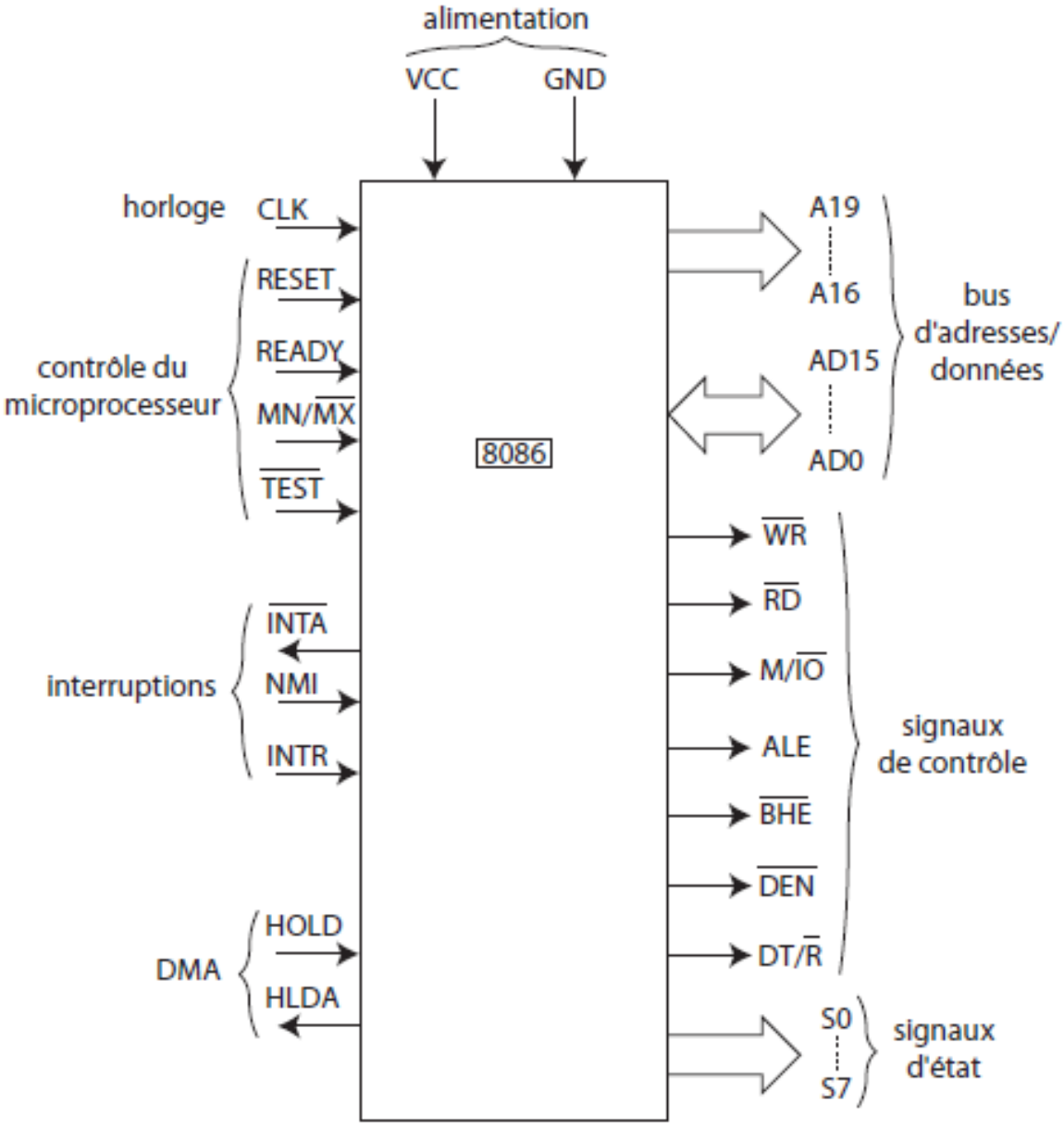
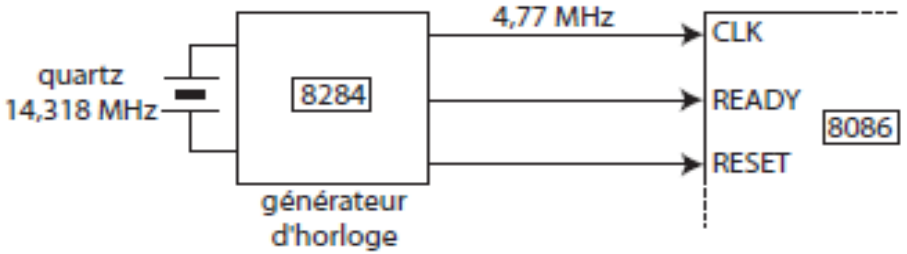


Schéma Fonctionnel du 8086



Description et Utilisation des Signaux du 8086

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



➢ **RESET :**entrée de remise à zéro du microprocesseur. Lorsque cette entrée est mise àl'état haut pendant au moins 4 périodes d'horloge, le microprocesseur est réinitialisé: il va exécuter l'instruction se trouvant à l'adresse $FFFF0_H$ (adresse de bootstrap). Le signal de RESET est fourni par le générateur d'horloge.

➢ **READY :**entrée de synchronisation avec la mémoire. Ce signal provient également du générateur d'horloge.

➢ **MN/$\overline{MX}$:** entrée de choix du mode de fonctionnement du microprocesseur :

- Mode minimum (MN/$\overline{MX}$= 1) : le 8086 fonctionne de manière autonome, il génère lui-même le bus de commande (RD, WR, ...) ;
- Mode maximum (MN/$\overline{MX}$ = 0) : ces signaux de commande sont produits par un contrôleur de bus, le **8288**. Ce mode permet de réaliser des systèmes multiprocesseurs.

➢**$\overline{TEST}$:** entrée de mise en attente du microprocesseur d'un événement extérieur.

➢ **$\overline{\text{INTA}}$:** indique que le microprocesseur accepte l'interruption.

➢ **NMI et INTR :**entrées de demande d'interruption.

- INTR : interruption normale.
- NMI (Non MaskableInterrupt) : interruption prioritaire.

➢ **HOLD et HLDA :** signaux de demande d'accord d'accès direct à la mémoire (Direct Memory Access - DMA).

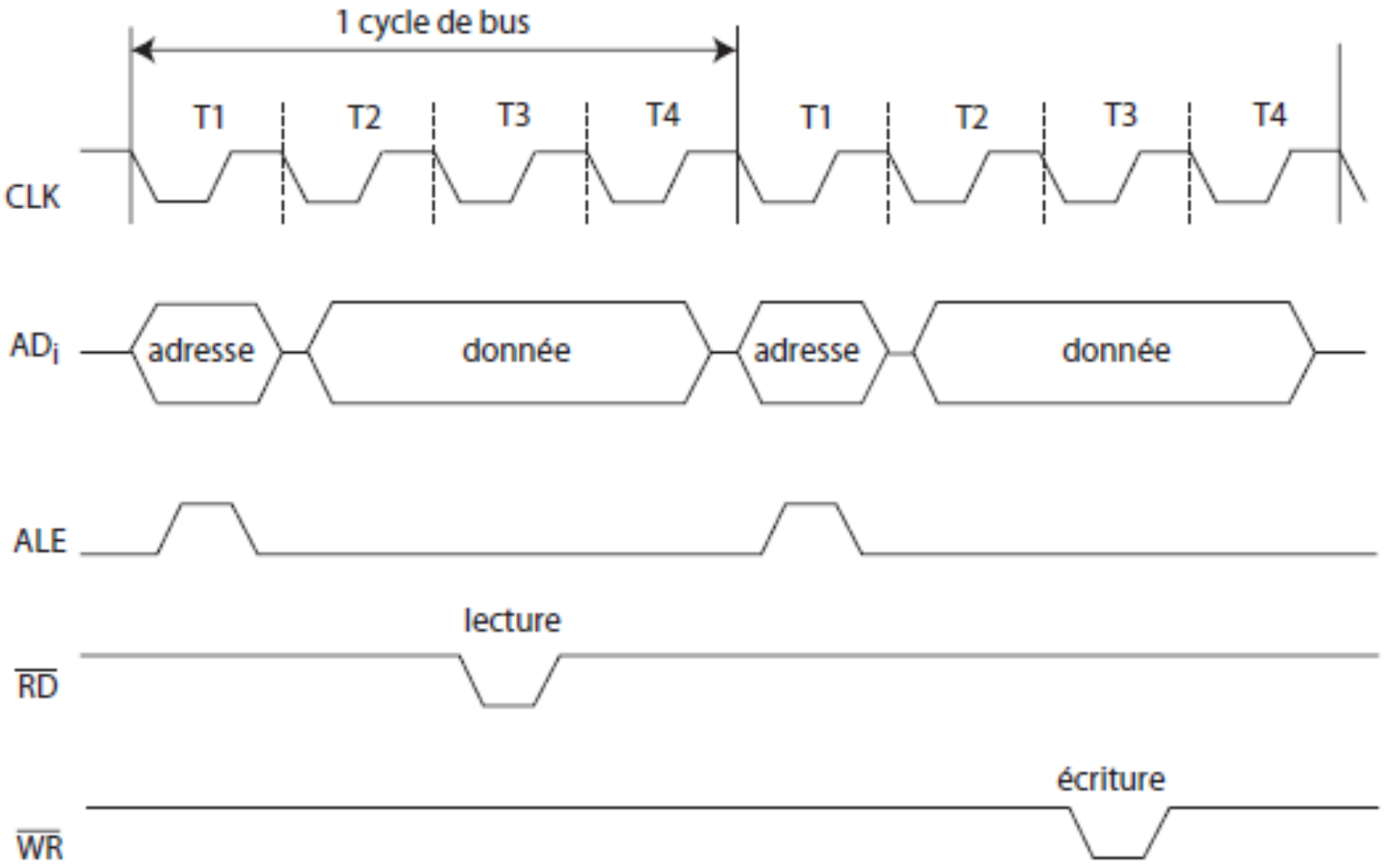
➢ **S0 à S7 :**signaux d'état indiquant le type d'opération en cours sur le bus.

➢ **A16/S3 à A19/S6 :**4 bits de poids fort du bus d'adresses, multiplexés avec 4 bits d'état.

➢ **AD0 à AD15 :**16 bits de poids faible du bus d'adresses, multiplexés avec 16 bits de données. Le bus A/D est multiplexé(multiplexage temporel) d'où la nécessité d'un démultiplexage pour obtenir séparément les bus d'adresses et de données :

- 16 bits de données (microprocesseur 16 bits) ;
- 20 bits d'adresses, d'où 2^{20} = 1 Mo d'espace mémoire adressable par le 8086.

❖ Chronogramme du bus A/D

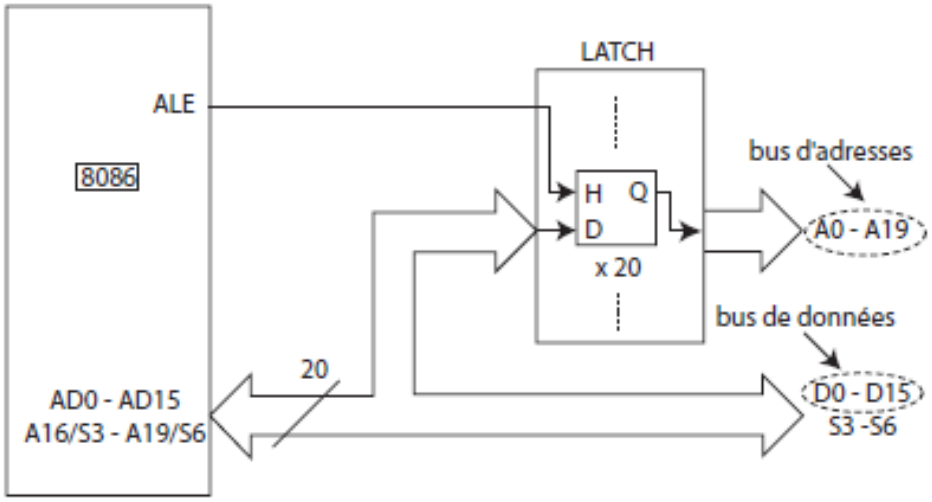


Le démultiplexage des signaux AD0 à AD15 (ou A16/S3 à A19/S6) se fait en mémorisant l'adresse lorsque celle-ci est présente sur le bus A/D, à l'aide d'un *verrou (latch)*, composé par un ensemble de bascules D. La commande de mémorisation de l'adresse est générée par le microprocesseur : *signal ALE (Address Latch Enable)*.

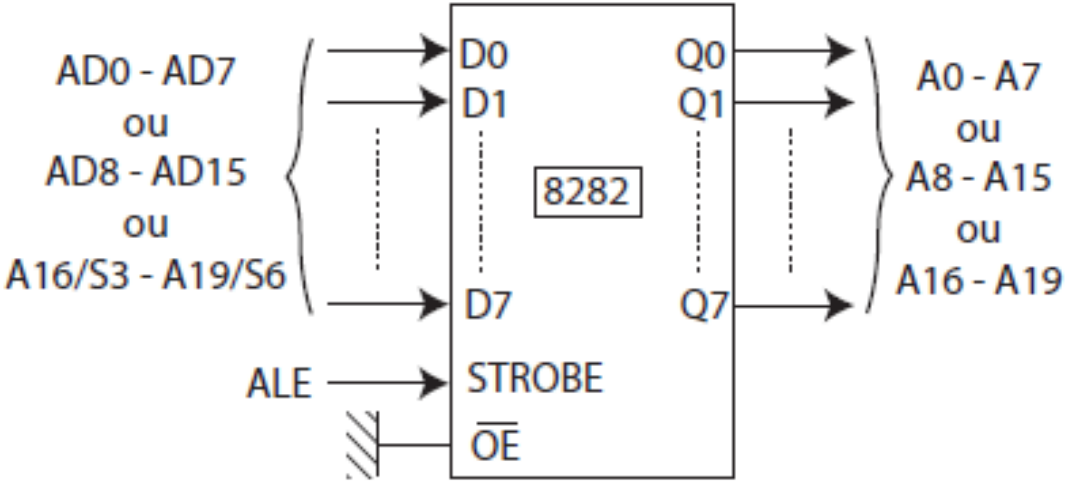
❖ Circuit de démultiplexage A/D

Fonctionnement :

- Si ALE = 1, le verrou est transparent (Q = D) ;
- Si ALE = 0, mémorisation de la dernière valeur de D sur les sorties Q.
- Les signaux de lecture ($\overline{RD}$) ou d'écriture ($\overline{WR}$) ne sont générés par le microprocesseur que lorsque les données sont présentes sur le bus A/D.



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



➢**$\overline{RD}$:** signal de lecture d'une donnée.

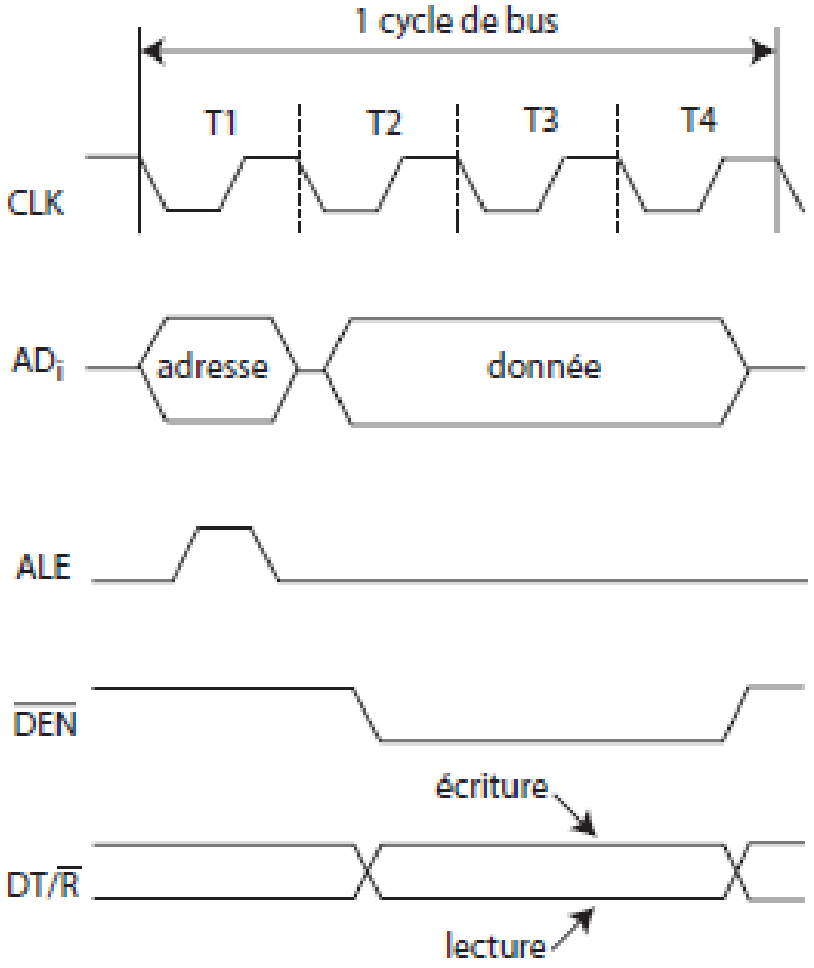
➢**$\overline{WR}$:** signal d'écriture d'une donnée.

➢**M /$\overline{IO}$:** (Memory/Input-Output) indique si le 8086 adresse la mémoire (M/$\overline{IO}$= 1) ou les entrées/sorties (M/$\overline{IO}$ = 0).

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

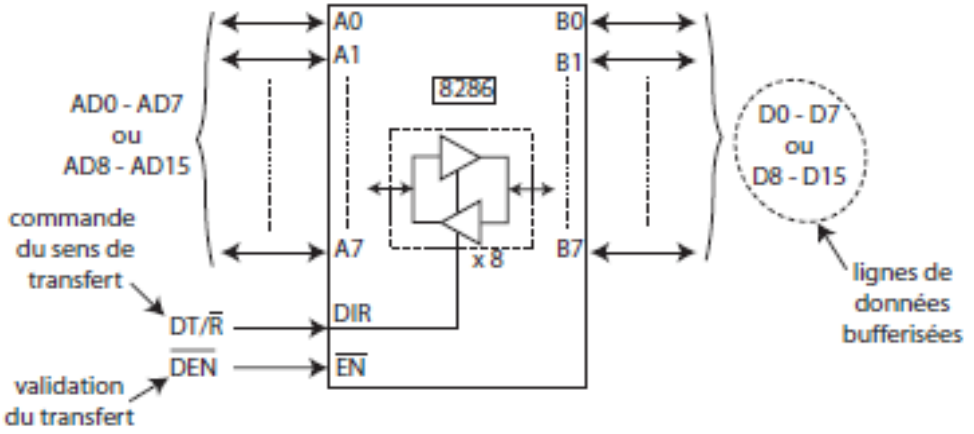
➢ **DT/ $\overline{R}$:** (Data Transmit/Receive) indique le sens de transfert des donnes :

- DT/$\overline{R}$ = 1 : données émises par le microprocesseur (écriture) ;
- DT/$\overline{R}$ = 0 : données reçues par le microprocesseur (lecture).



Les signaux $\overline{DEN}$ et DT/$\overline{R}$sont utilisés pour la commande de tampons de bus (buffers) permettant d'amplifier le courant fourni par le microprocesseur sur le bus de données.

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

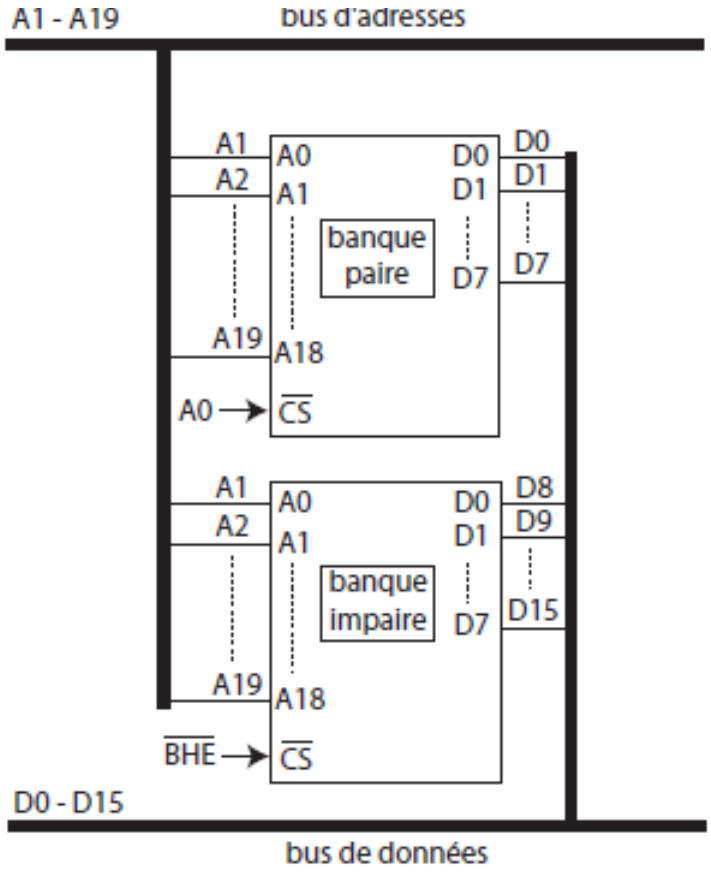


➢**$\overline{BHE}$:** (Bus High Enable) signal de lecture de l'octet de poids fort du bus de données. Le 8086 possède un bus d'adresses sur 20 bits, d'oùla capacitéd'adressage de 1 Mo ou 512 Kmots de 16 bits (bus de données sur 16 bits).

Le méga-octet adressable est diviséen deux banques de 512 Ko chacune : la banque inférieure (ou paire) et la banque supérieure (ou impaire).

Ces deux banques sont sélectionnées par :

- A0 pour la banque paire qui contient les octets de poids faible ;
- $\overline{BHE}$pour la banque impaire qui contient les octets de poids fort.

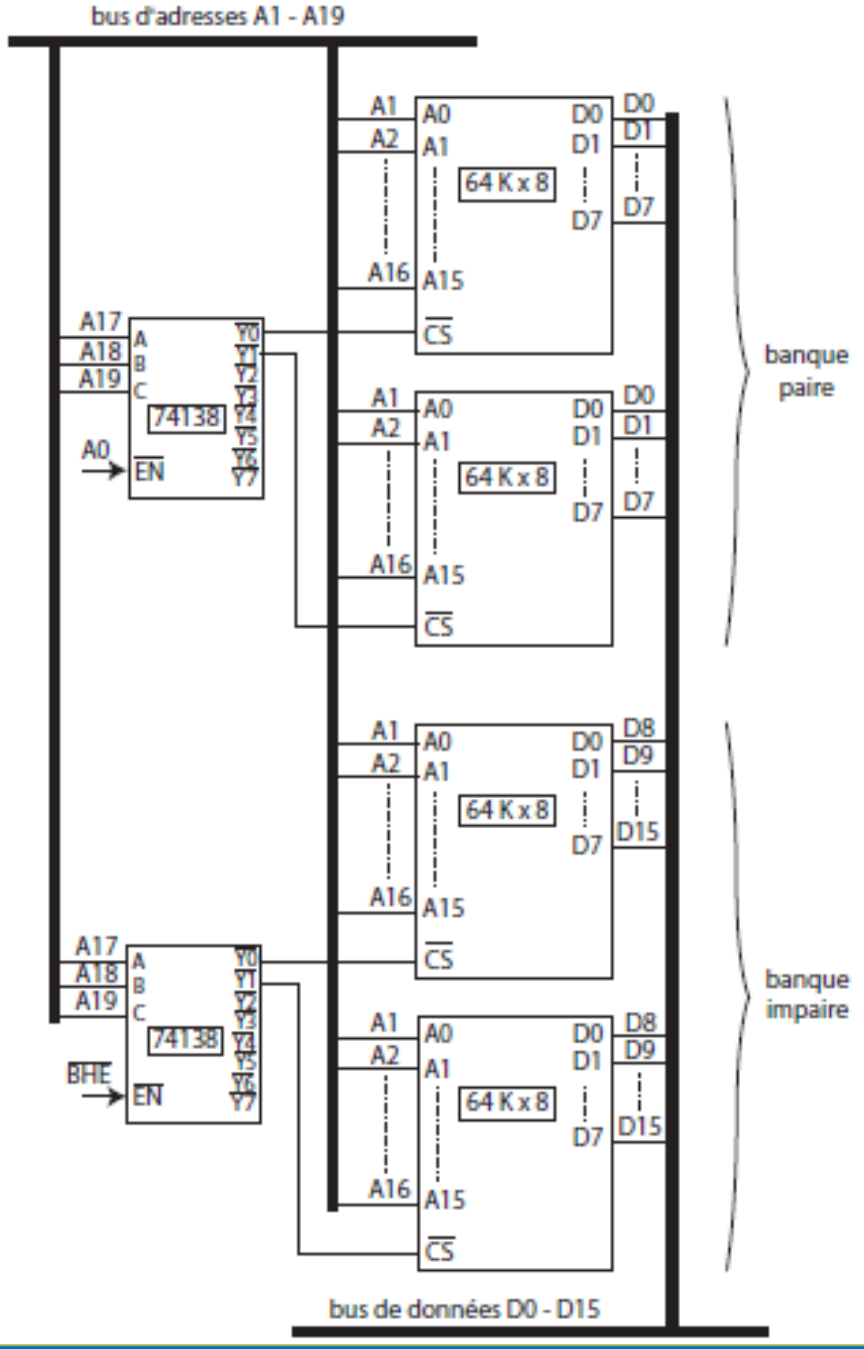


Seuls les bits A1 à A19 servent à désigner une case mémoire dans chaque banque de 512 Ko. Le microprocesseur peut ainsi lire et écrire des données sur 8 bits ou sur 16 bits :

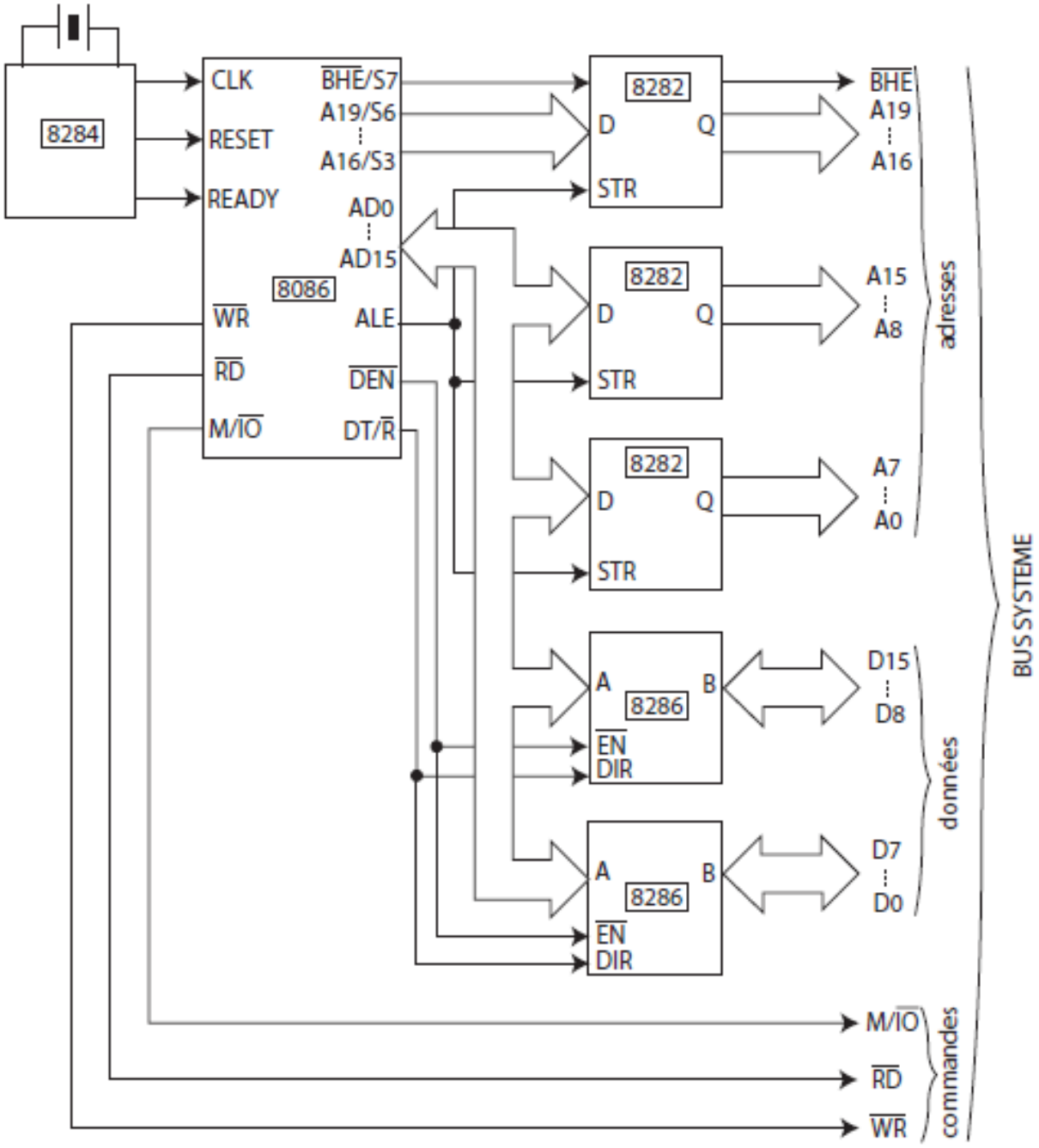
BHE	A0	Octets transférés
0	0	Les deux octets (mot complet)
0	1	Octet fort (adresse impaire)
1	0	Octet faible (adresse paire)
1	1	Aucun octet

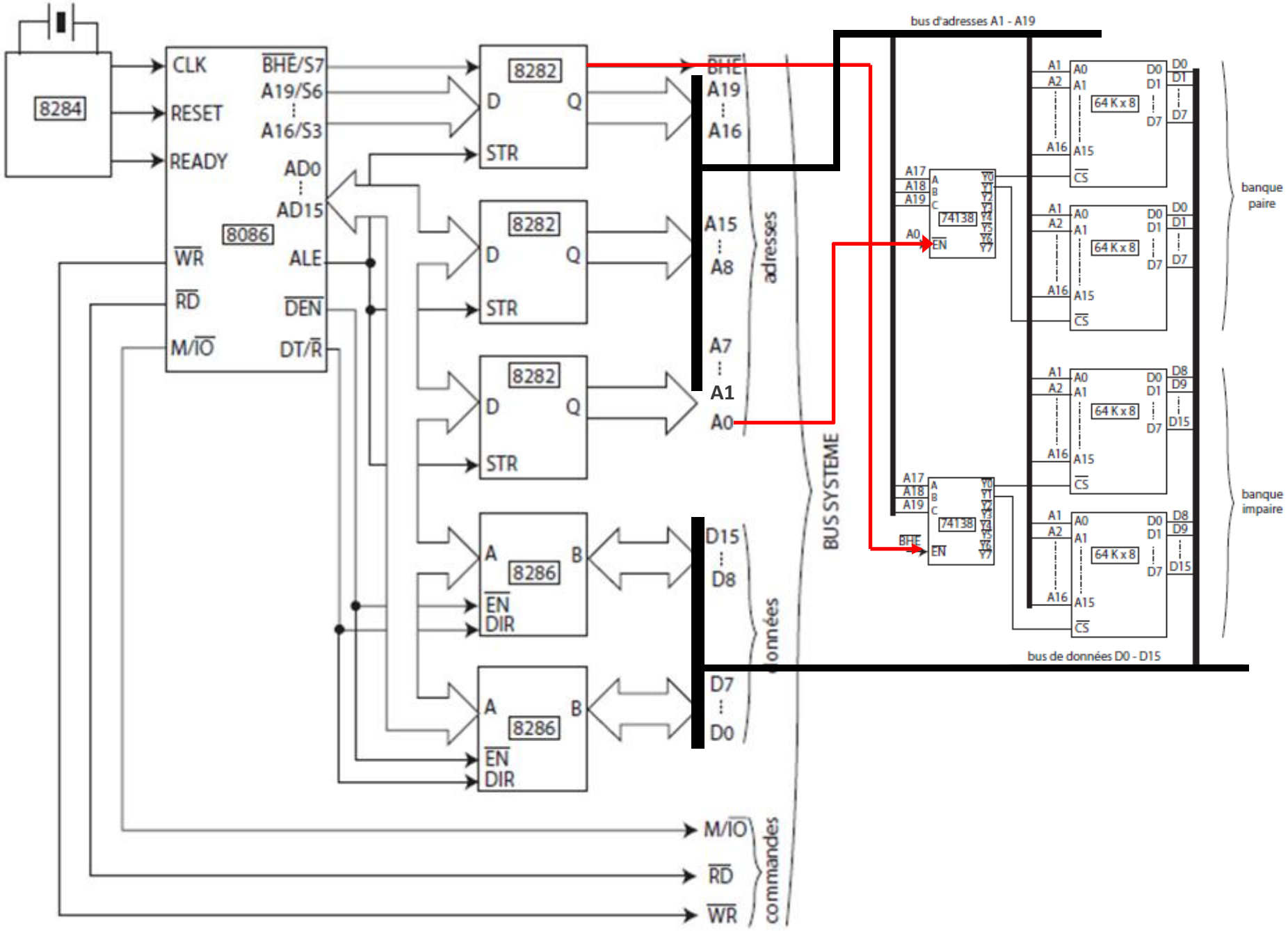
Remarque : le 8086 peut lire une donnée sur 16 bits en une seule fois, uniquement si l'octet de poids fort de cette donnée est rangé à une adresse impaire et l'octet de poids faible à une adresse paire (alignement sur les adresses paires), sinon la lecture de cette donnée doit se faire en deux opérations successives, d'où on a une augmentation du temps d'exécution du transfert dû à un mauvais alignement des données.

Réalisation des deux banques avec plusieurs boîtiers mémoire



Création du bus système du 8086




8284
CLK
RESET
READY
8086
WR
RD
M/IO
BHE/S7
A19/S6
A16/S3
AD0
AD15
ALE
DEN
DT/R
8282
D
Q
STR
8286
A
B
EN
DIR
BHE
A19
A16
A15
A8
A7
A1
A0
D15
D8
D7
D0
adresses
données
commandes
BUS SYSTEME
bus d'adresses A1 - A19
bus de données D0 - D15
74138
A17
A18
A19
64 K x 8
banque paire
banque impaire

Organisation Interne du 8086

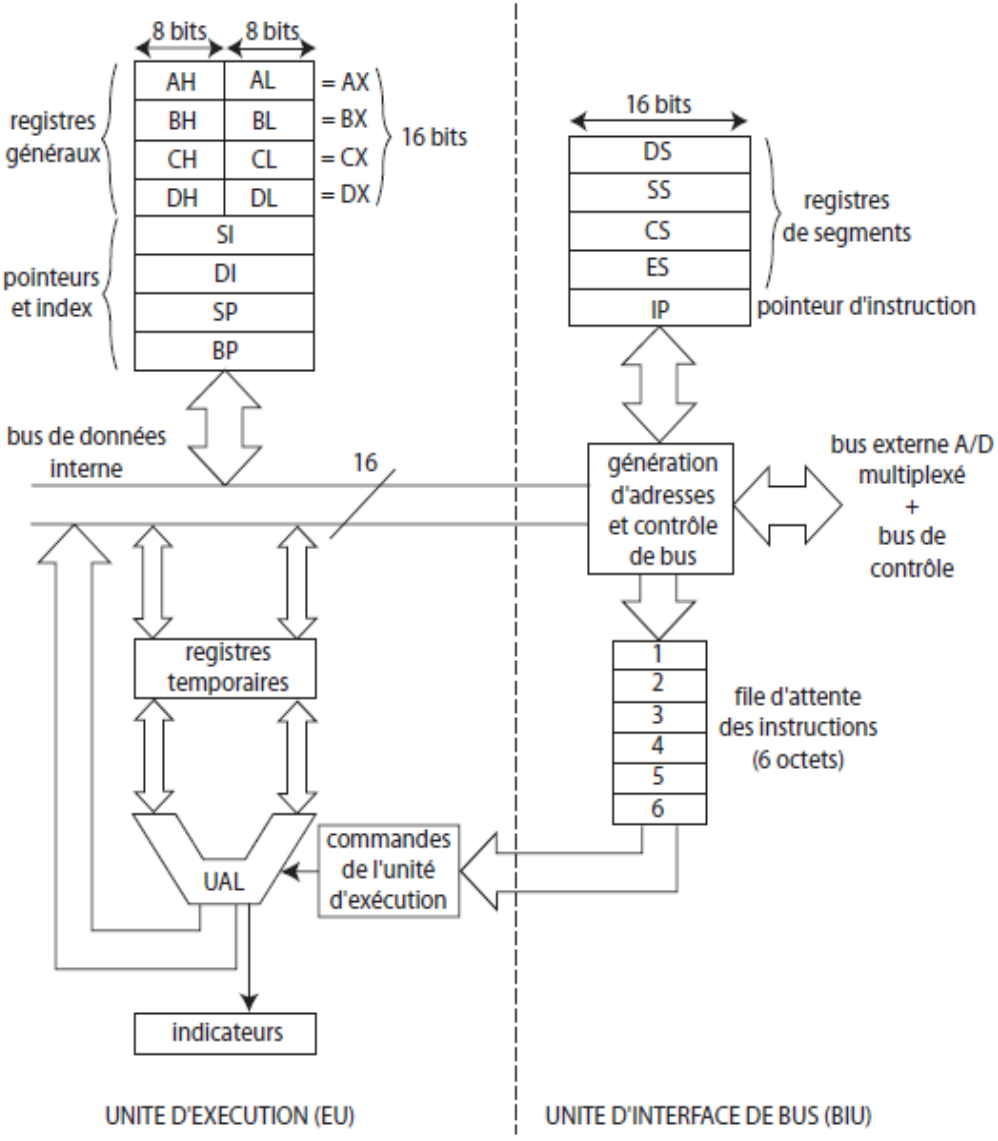
❖ Le 8086 est constitué de deux unités fonctionnant en parallèle :

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

❖ Rôle des deux unités :

- L'unité d'interface de bus (BIU) recherche les instructions en mémoire et les range dans *une file d'attente* ;
- L'unité d'exécution (EU) exécute les instructions contenues dans la file d'attente.

Les deux unités fonctionnent simultanément, d'où une accélération du processus d'exécution d'un programme.



Le microprocesseur 8086 contient 14 registres répartis en 4 groupes :

❖ **Registres généraux :** 4 registres sur 16 bits.

16 bits	8 bits (High)	8 bits (Low)
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

Ils peuvent être également considérés comme 8 registres sur 8 bits. Ils servent à contenir temporairement des données. Ce sont des registres généraux mais ils peuvent être utilisés pour des opérations particulières.
Exemple : AX = accumulateur, CX = compteur.

❖ **Registres de pointeurs et d'index :** 4 registres sur 16 bits.

Pointeurs :

- ✓ ***SP (Stack Pointer) :*** pointeur de pile (la pile est une zone de sauvegarde de données en cours d'exécution d'un programme) ;
- ✓ ***BP (Base Pointer) :*** pointeur de base, utilisé pour adresser des données sur la pile.

Index :

SI : Source Index ;

DI : Destination Index.

Ils sont utilisés pour les transferts de chaînes d'octets entre deux zones mémoire.

Les pointeurs et les index contiennent des adresses de cases mémoire.

❖ **Pointeur d'instruction et indicateurs (flags) :** 2 registres sur 16 bits.

✓ ***Pointeur d'instruction (IP)*** contient l'adresse de la prochaine instruction à exécuter.

✓ ***Flags :***

--	--	--	OF	DF	IF	TF	SF	ZF	--	AF	--	PF	--	CF

CF : indicateur de retenue (carry) ;

PF : indicateur de parité;

AF : indicateur de retenue auxiliaire ;

ZF : indicateur de zéro ;

SF : indicateur de signe ;

TF : indicateur d'exécution pas à pas (trap) ;

IF : indicateur d'autorisation d'interruption ;

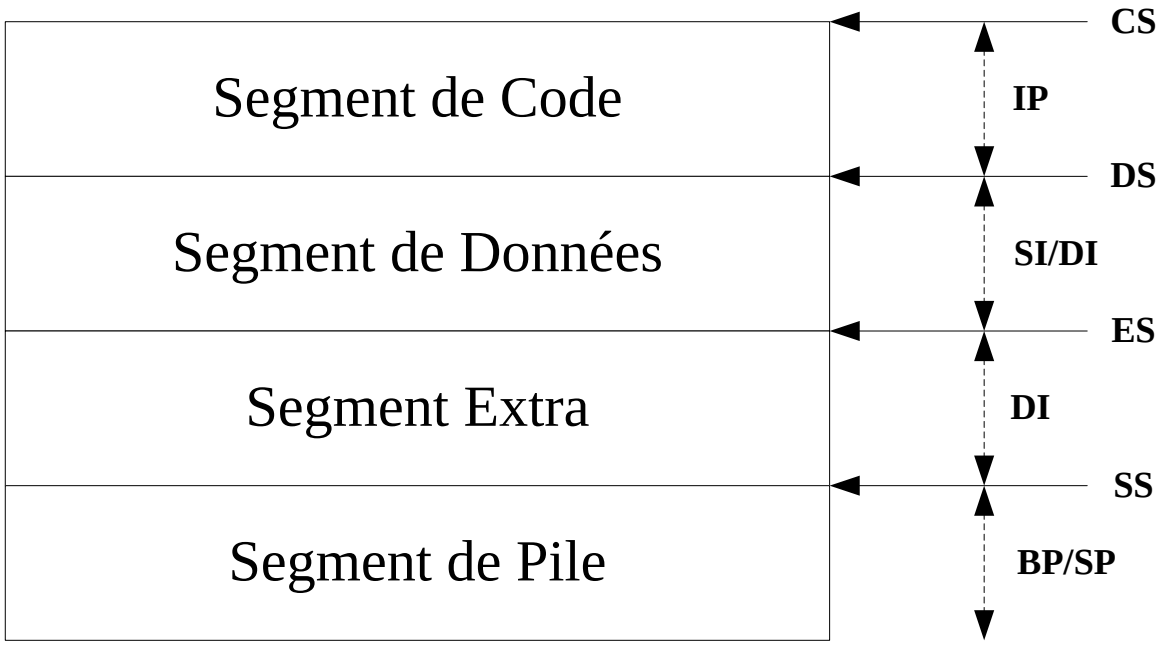
DF : indicateur de décrémentation ;

OF : indicateur de dépassement (overflow).

❖ **Registres de segments :** 4 registres sur 16 bits.

CS (Code Segment) : registre de segment de code ;
DS (Data Segment) : registre de segment de données ;
SS (Stack Segment) : registre de segment de pile ;
ES (Extra Segment) : registre de segment supplémentaire pour les données ;

Les registres de segments, associés aux pointeurs et aux index, permettent au microprocesseur 8086 d'adresser l'ensemble de la mémoire.

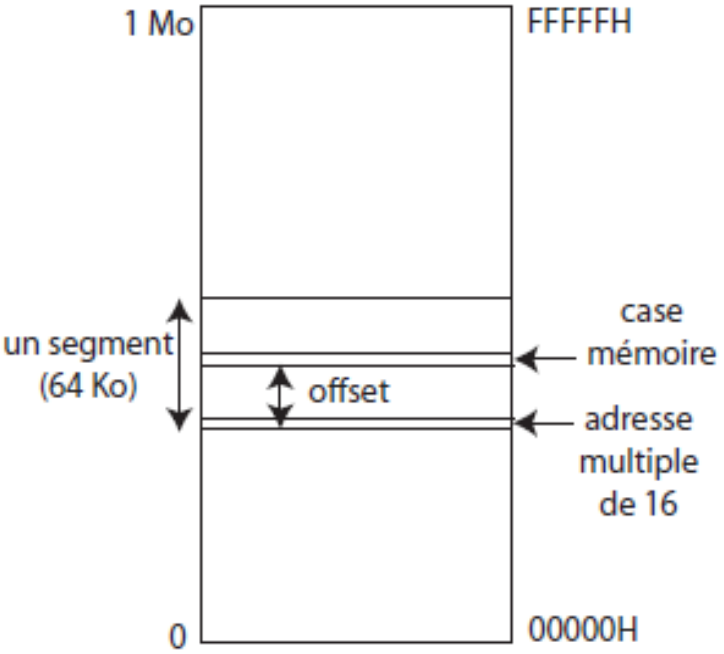


Gestion de la Mémoire par le 8086

L'espace mémoire adressable par le 8086 est de 2^{20} = 1 048 576 octets = 1 Mo (20 bits d'adresses). Cet espace est divisé en *segments*. Un segment est une zone mémoire de 64 Ko (65 536 octets) définie par son adresse de départ qui doit être un multiple de 16.
Dans une telle adresse, les 4 bits de poids faible sont à zéro. On peut donc représenter l'adresse d'un segment avec seulement ses 16 bits de poids fort, les 4 bits de poids faible étant implicitement à 0.
Pour désigner une case mémoire parmi les 2^{16} = 65 536 contenues dans un segment, il suffit d'une valeur sur 16 bits.

Ainsi, une case mémoire est repérée par le 8086 au moyen de deux quantités sur 16 bits :

- *L'adresse d'un segment ;*
- *Un déplacement ou offset (appelé aussi adresse effective) dans ce segment.*

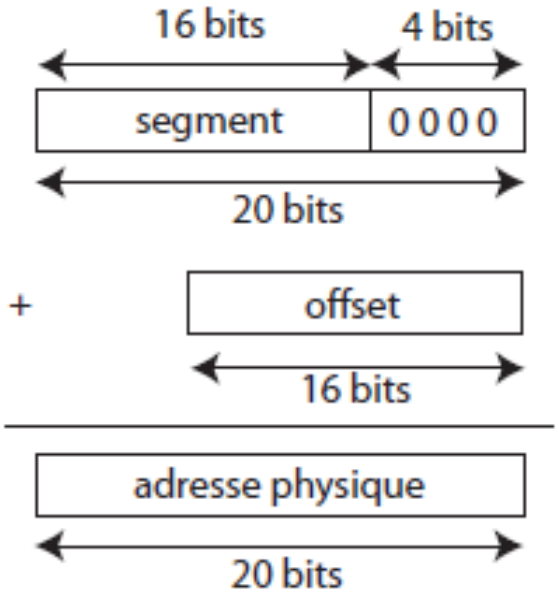


Cette méthode de gestion de la mémoire est appelée *segmentation de la mémoire.*

La donnée d'un couple (segment, offset) définit *une adresse logique*, notée sous la forme ***segment : offset.***

L'adresse d'une case mémoire donnée sous la forme d'une quantité sur 20 bits est appelée *adresse physique* car elle correspond à la valeur envoyée réellement sur le bus d'adresses A0 - A19.

Correspondance entre adresse logique et adresse physique:



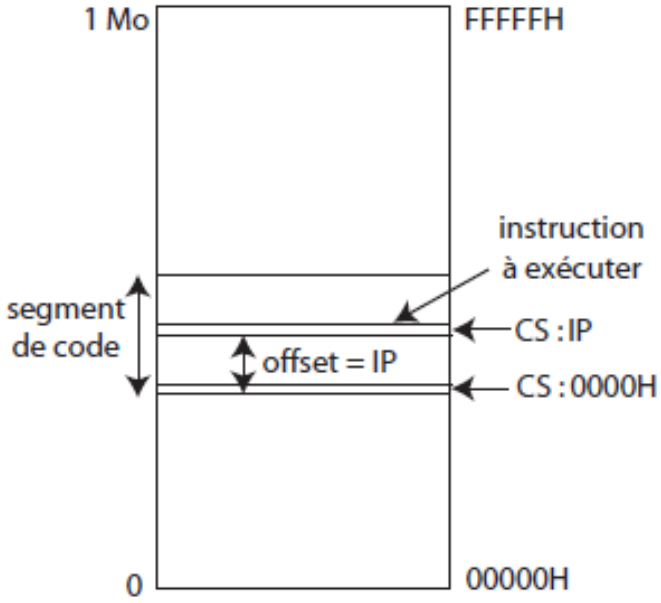
Ainsi, l'adresse physique se calcule par l'expression :

adresse physique = 16 × *segment* + *offset*

car le fait d'injecter 4 zéros en poids faible du segment revient à effectuer un décalage de 4 positions vers la gauche, c'est à dire une multiplication par 2^4.

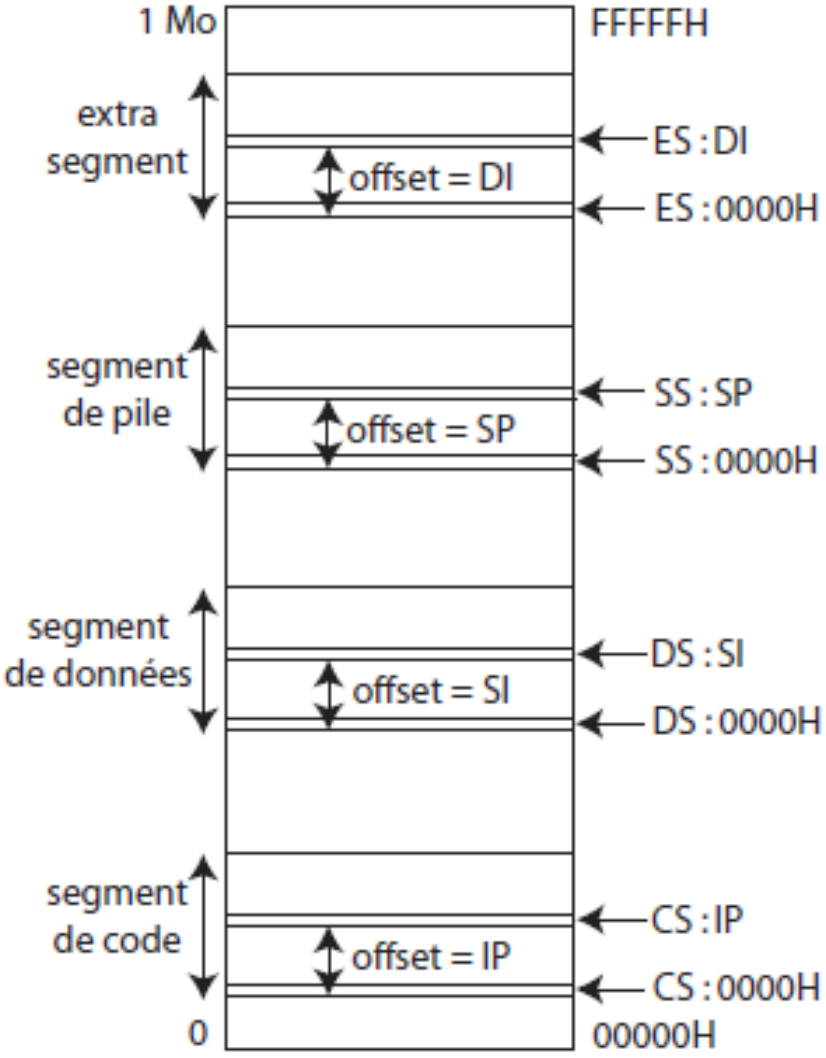
A un instant donné, le 8086 a accès à 4 segments dont les adresses se trouvent dans les registres de segment CS, DS, SS et ES. Le segment de code contient les instructions du programme, le segment de données contient les données manipulées par le programme, le segment de pile contient la pile de sauvegarde et le segment supplémentaire peut aussi contenir des données.

Le registre CS est associé au pointeur d'instruction IP, ainsi la prochaine instruction à exécuter se trouve à l'adresse logique CS : IP.

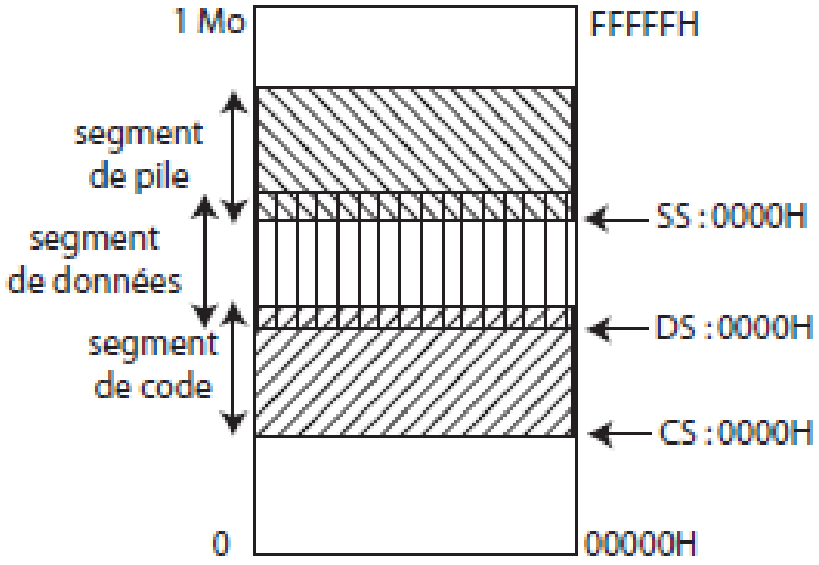


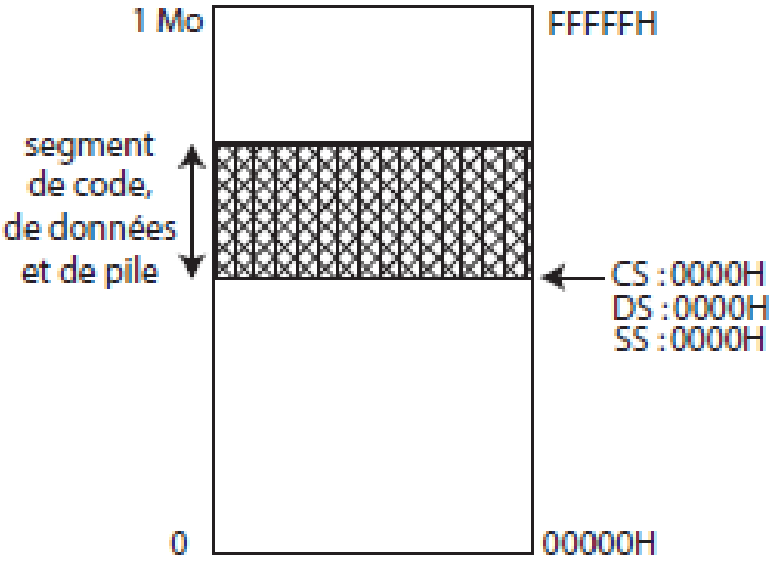
De même, les registres de segments DS et ES peuvent être associés à un registre d'index. *Exemple :* DS : SI, ES : DI. Le registre de segment de pile peut être associé aux registres de pointeurs : SS : SP ou SS : BP.

Mémoire accessible par le 8086 à un instant donné :



Remarque: les segments ne sont pas nécessairement distincts les uns des autres, ils peuvent se chevaucher ou se recouvrir complètement.





Le nombre de segments utilisé définit *le modèle mémoire du programme.*

Contenu des registres après un RESET du microprocesseur :

$$IP = 0000_H$$
$$CS = FFFF_H$$
$$DS = 0000_H$$
$$ES = 0000_H$$
$$SS = 0000_H$$

Puisque CS contient la valeur $FFFF_H$ et IP la valeur 0000_H, la 1ère instruction exécutée par le 8086 se trouve donc à l'adresse logique $FFFF_H : 0000_H$, correspondant à l'adresse physique $FFFF0_H$ (bootstrap). Cette instruction est généralement un saut vers le programme principal qui initialise ensuite les autres registres de segment.