I) Introduction :

Le processeur 8086 d'Intel Le microprocesseur Intel 8086 est un microprocesseur 16 bits, développé par Intel en juin 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.

Les processeurs successifs du 8086 se sont en effet construits petit à petit en ajoutant à chaque processeur 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 par exemple d'un Pentium IV, l'inverse n'est évidemment pas toujours vrai.

II) Architecture externe du 8086

Le 8086 est équipé d'un bus multiplexé de 20lignes Le bus de données est constitué de 16 lignes $D_{15}\ldots D_0$ .le bus d'adresses de 20 lignes. $A_{19}\ldots A_0$ et fonctionne à des fréquences diverses 5, 8 ou 10 MHz.

Le brochage et le schéma fonctionnel du 8086 sont décrits respectivement par les figures 1 et 2 ; on peut y voir que le 8086 comprend 40 broches.

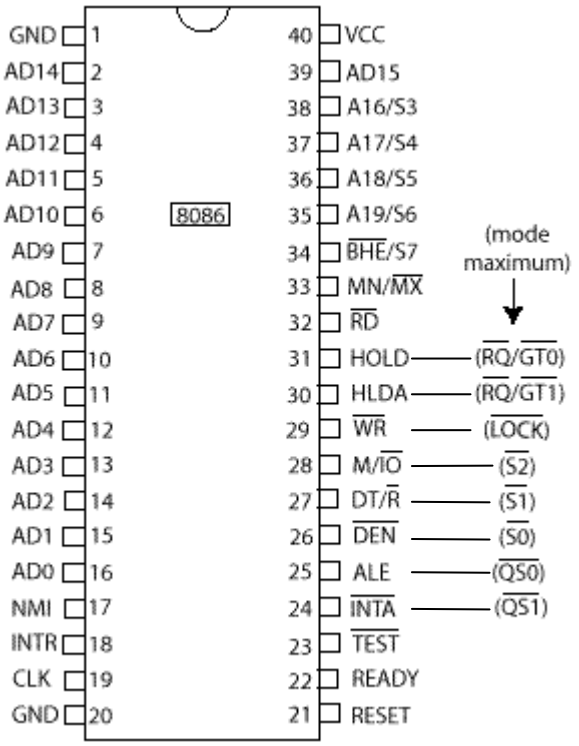


Figure 1: brochage du 8086

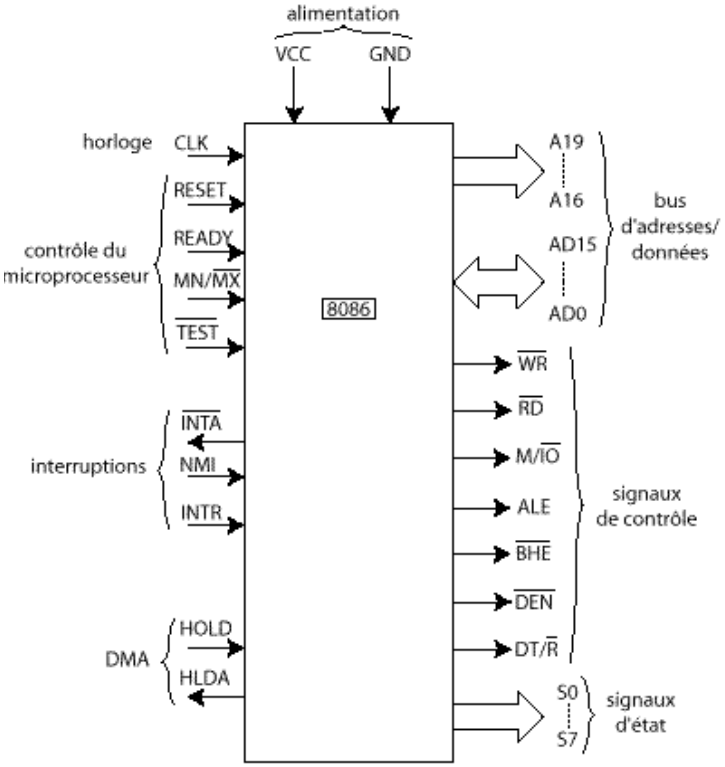


Figure 2: Schéma fonctionnel du 8086 .

II.1 Description des broches

Vcc et GND assurent l'alimentation électrique du microprocesseur.

CLK : entrée destinée à recevoir le signal de l'horloge système, qui cadence le fonctionnement du microprocesseur. Ce signal provient d'un générateur d'horloge : le 8284.
READY : permet la synchronisation des mémoires et périphériques lents avec le CPU 8086.

RESET: un signal, de remise à l'état initial, est généré à partir d'un signal externe RES qui se synchronise avec l'horloge CLK. Ce signal doit rester à l'état haut pendant au moins quatre cycles d'horloge, pour permettre au CPU 8086 de s'initialiser correctement, en commençant l'exécution à partir de l'adresse FFFF:0000.

MN/$\overline{\text{MX}}$: Sélectionne entre l'un des deux modes de fonctionnement du 8086.

- Mode minimum: le CPU 8086 fonctionne d'une manière autonome et en monoprocesseur et il génère par lui même les signaux de bus de commande.
- Mode maximum : Ces signaux sont produit par un contrôleur de bus 8288, ce qui lui permet d'opérer dans un environnement multiprocesseur.

$\overline{\text{TEST}}$: entrée de synchronisation entre le CPU 8086 et le coprocesseur

NMI, INTR: entrées de demande d'interruption. INTR: interruption normale, NMI (Non Masquable Interrupt) : interruption prioritaire.

$\overline{\text{INTA}}$ (Inerrupt Acknowledge) : indique que le microprocesseur a pris en compte l'interruption.

HOLD: entrée de demande d'accès au bus.
HLDA: indique que le microprocesseur a pris en compte la demande d'accès au bus.

S0 à S7: Signaux d'état indiquant le type d'opération 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, d'où la nécessité d'un multiplexage pour obtenir séparément les bits d'adresse et de données.
$\overline{\text{RD}}$ (Read): signal de lecture d'une donnée.

$\overline{\text{WR}}$ (Write): Signal d'écriture d'une donnée.

M/$\overline{\text{IO}}$ (Memory /Input-Output): indique que le CPU 8086 adresse la mémoire ou les unités d'entrées/sorite.

$\overline{\text{DEN}}$(Data ENable): indique que la donnée est disponible sur le bus de données.

ALE (Adress Latch Enable) : indique que l'adresse est disponible sur le bus d'adresses.

DT/$\overline{\text{R}}$ (Data Transmit/ Receive): indique le sens de transfert des données.

$\overline{\text{\quad\quad}}$

BHE (Bus High Enable): Signal de validation de l'octet du poids fort du bus de données

III) Architecture interne du 8086 :

Il existe deux unités internes distinctes: l'UE (Unité d'Exécution) et l'UIB (Unité d'Interfaçage avec le Bus). Le rôle de l'UIB est de récupérer et stocker les informations à traiter, et d'établir les transmissions avec les bus du système. L'UE exécute les instructions qui lui sont transmises par l'UIB. La figure3 résume les notions présentées ici.

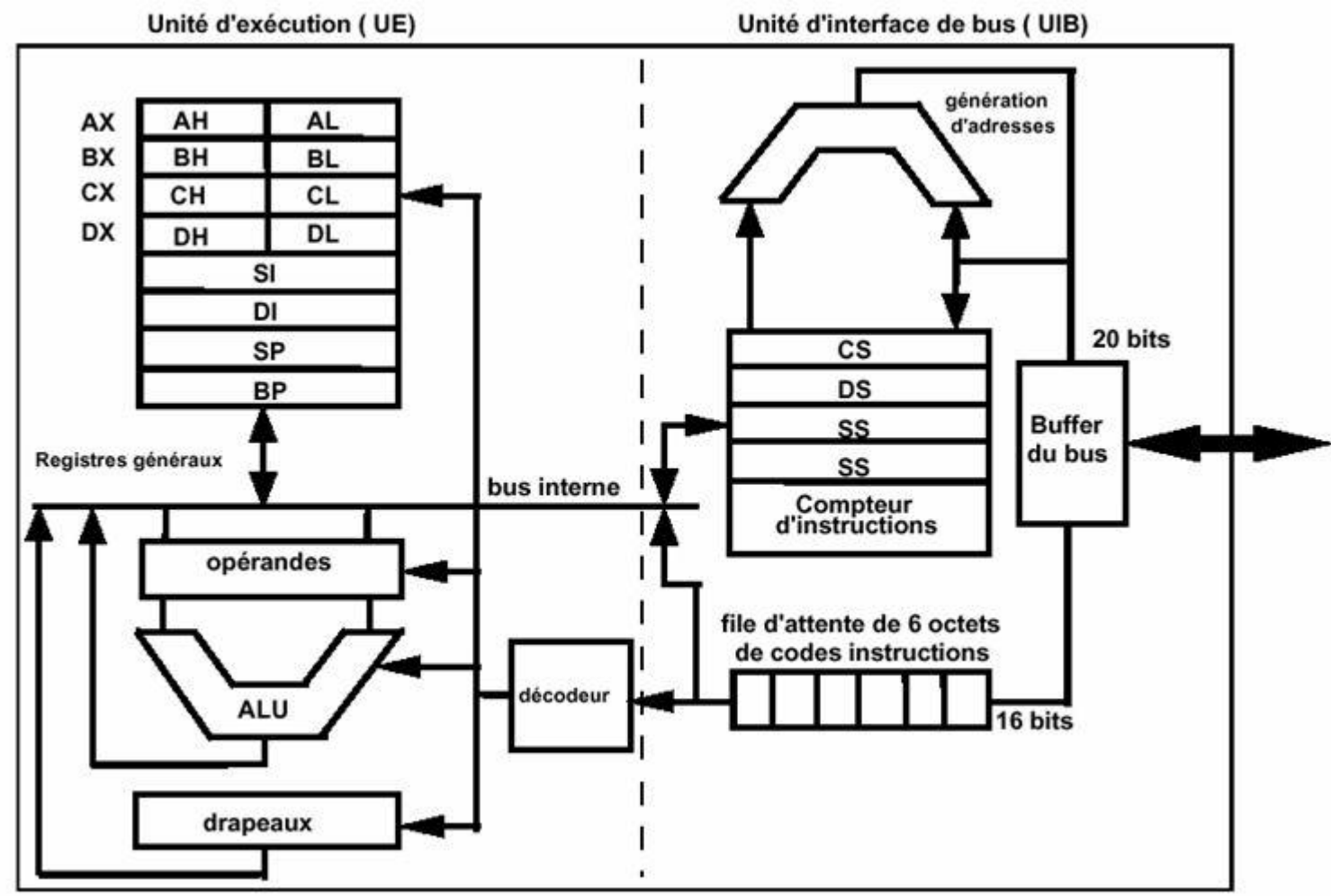


Fig3 : Architecture interne du 8086

La file d'attente d'instructions contient des instructions qui attendent d'être traitées par l'UE. Le microprocesseur 8086 est capable de mémoriser jusqu'à six octets. Les microprocesseurs actuels sont bien entendu équipés d'une file d'attente plus rapide et plus large, c'est à dire capable d'emmagasiner plus d'informations.

les deux unités travaillent en parallèle ; c'est la naissance de l'architecture pipeline.

IV) Les registres du 8086/8088 :

IV-1) Introduction :

Le jeu de registres contient l'ensemble des registres du microprocesseur. Un registre est une petite partie de mémoire intégrée au microprocesseur, dans le but de recevoir des informations spécifiques, notamment des adresses et des données stockées durant l'exécution d'un programme. Il existe plusieurs types de registres. Certains d'entre eux sont affectés à des opérations d'ordre général et sont accessibles au programmeur à tout moment. Nous disons alors qu'il s'agit de registres généraux. D'autres registres ont des rôles bien plus spécifiques et ne peuvent pas servir à un usage non spécialisé.

IV-2) Les registres généraux :

Les registres généraux peuvent être utilisés dans toutes les opérations arithmétiques et logiques que le programmeur insère dans le code assembleur. Un registre complet est constitué de 16 bits. Le programmeur dispose de 8 registres généraux de 16 bits qu'on peut diviser en deux catégories de registres :

- *registres de données* : formé par 4 registres de 16 bits (AX, BX ,CX, et DX) chaque registre peut être divisé en deux registres de 8 bits :(AH,AL,BH,BL,CH,CL,DH et DL)

	15 8	7 0
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

:

Registre AX : (Accumulateur)

Toutes les opérations de transferts de données avec les entrées-sorties ainsi que le traitement des chaînes de caractères se font dans ce registre, de même les opérations arithmétiques et logiques.

Registre BX : (registre de base)

Il est utilisé pour l'adressage de données : en général il contient une adresse de base. (Par exemple, l'adresse de début d'un tableau). De plus il peut être utilisé dans toutes les opérations arithmétiques et logiques.

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.

Remarque :

Le registre CL sert en tant que compteur pour les opérations de décalage et de rotation, dans ce cas il va compter le nombre de décalages (rotation) de bits à droite ou à gauche.

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.

- *Registres pointeur et index :*

Ces registres sont plus spécialement adaptés au traitement des éléments dans la mémoire. Il s'agit des 4 registres suivants (SI, DI, SP, BP)

- :

	15	0
Stack pointer	SP	
Base pointer	BP	
Source index	SI	
Destination index	DI	

L'index SI : (source indexe) :

Il permet de pointer la mémoire il 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'index 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

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 fifo : voir plus loin), 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 utilisé avec le segment de pile.

IV -3) Les registres segment:

	15	0
Code segment	CS	
Data segment	DS	
Stack segment	SS	
Extra segment	ES	

Le 8086 a quatre registres segments de 16 bits chacun : CS (code segment),

DS (Data segment), ES (Extra segment) et 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 64Koctets (65535 octets.)

Le registre CS (code segment) :

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

Remarque :

Si la taille du programme dépasse les 64Ko alors on peut diviser le code sur plusieurs segments et pour basculer d'une partie à une autre du programme il suffit de changer la valeur du registre CS ; cela se fait via des instructions ; et de cette manière on résout le problème des programmes qui ont une taille supérieure à 64Ko.

Le registre DS (Data segment) :

Le registre segment de données pointe sur le segment des variables du programme, bien évidemment la taille ne peut excéder 64Ko (si on a des données qui dépassent cette limite, on utilise la même astuce citée dans la remarque précédente)..

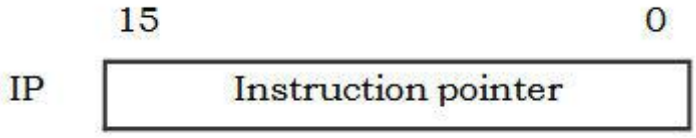
Le registre ES (Extra segment) :

Le registre de données supplémentaires ES pointe sur l'extra segment. ce segment est utilisé pour le stockage des chaînes de caractères.

Le segment SS (Stack segment) :

Le registre SS pointe sur la pile : la pile est une zone mémoire ou on peut sauvegarder des Informations telles les adresses de retour , le PSW etc ; pour pouvoir les récupérer après l'exécution d'un sous programme ou d'un programme d'interruption,.

IV-4) Le registre IP : (Le compteur de programme) :



Instruction Pointer ou Compteur ordinal, contient l'adresse de l'emplacement mémoire où se situe 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.

V-5) Le registre d'état (PSW) :

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 nul, un bit spécifique du PSW
le bit ZF est mis à 1.
Chaque bit du PSW est considéré comme un drapeau.

Les drapeaux sont des indicateurs qui annoncent une condition particulière suite à une opération arithmétique ou logique.

Le registre d'état du 8086 est formé par les bits suivants :

Remarque :

X : bit non utilise.

CF (*Carry Flag) :*
Retenue : cet indicateur et 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:

AF (*Auxiliary Carry) :*
Demi-retenue : Ce bit est égal à 1 si on a une retenue du quartet de poids faible vers le le quartet suivant.

PF (*Parity Flag) :*
Parité : si le résultat de l'opération contient un nombre pair de 1 cet indicateur est mis à 1,

ZF (*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.

SF (*Sign Flag) :*
SF est positionné à 1 si le bit de poids fort du résultat d'une opération 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
SF=1	11101010	SF=0	00101011

OF (*Overflow Flag) :*

OF prend la valeur 1 pour indiquer la perte de signe après une opération arithmétique, significatif en arithmétique signée.

	10010110	01100110
+	11000001	01000001
OF=1	01110111	10100111

DF (*Direction Flag) :*

Auto Incrémentation/Décrémentation : utilisée par les instructions de chaîne de caractères pour auto incrémenter ou auto décrémenter le SI et le DI.

IF(*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.

V) Gestion de la mémoire :

V-1) Introduction :

L'espace mémoire adressable (1 mégaoctet = 2^{20}O ,20 bits du bus d'adresses) du 8086 est divisé en segments logiques allant jusqu'à 64 K Octets chacun. Chaque segment est pointé par le registre segment qui lui correspond,

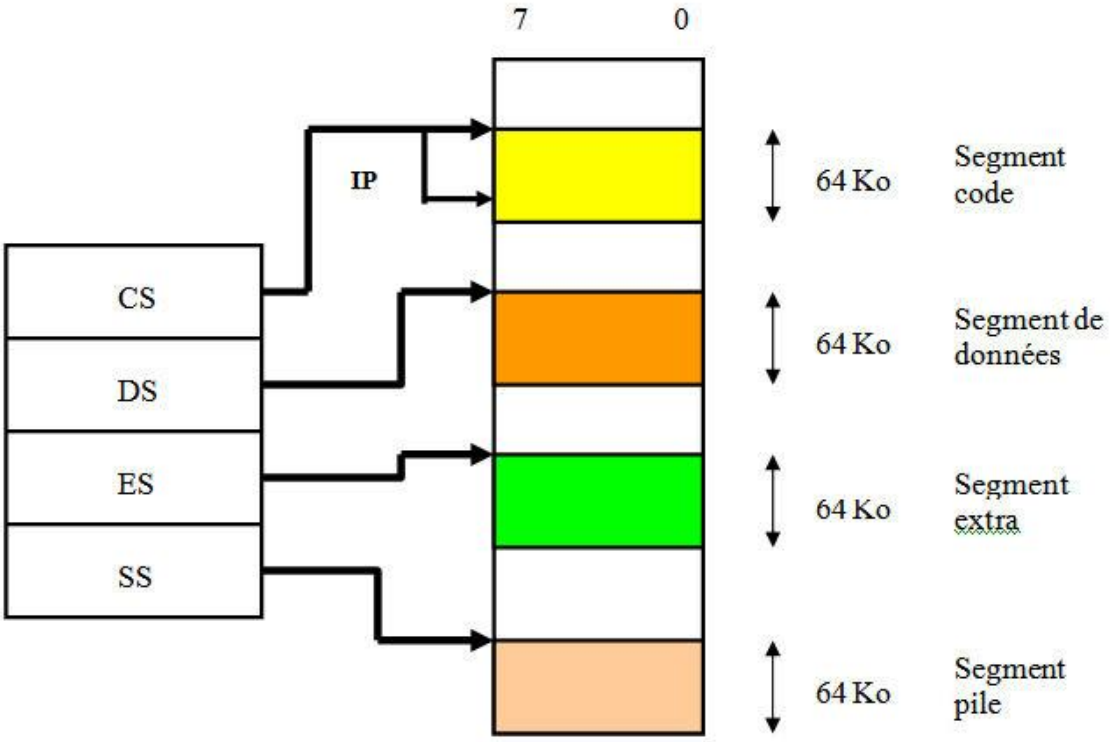


Fig 4 : Les segments du 8086

Mais comme une adresse est codée sur 20 bits, le registre segment ne pourra jamais contenir toute l'adresse.

Pour résoudre le problème il a été décidé qu'un segment ne peut être chargé qu'a une adresse multiple de 16 (elle se terminera par 0H 0000B),ainsi le registre segment contient les 16 bits de poids fort de l'adresse du segment, et cette dernière sera égale a

16 *registre segment, cette adresse est dite adresse physique

L'adresse logique (déplacement ou offset) d'une information (variable, instruction) est obtenue en supposant que l'adresse de début du segment est égale à 0.

Dans le programme exécutable, une variable est remplacée par son adresse logique

Le registre IP contient l'adresse logique de la prochaine instruction à exécuter.

L'adresse physique d'une information représente la véritable adresse de l'information en mémoire centrale

Une information est donc caractérisée par 2 adresses : adresse logique et l'adresse physique.

L'adresse physique formée de 20 bits est obtenue en appliquant la formule suivante :

Adresse physique= Base * 16+ adresse logique, la base est la valeur du registre segment dans lequel se trouve l'information.

EXEMPLE

```
DATA SEGMENT
X DW 1ABCH
TAB DB 20 DUP 0AH
CHAINE DB 'archit12010'
Fin DB 'fin'
DATA  ENDS.     On suppose que DS=FF00H
```

VARIABLE	@ LOGIQUE	@PHYSIQUE
X	0000H	FF000H
TAB	0002H	FF002H
CHAINE	0016H	FF016H
FIN	0021H	FF021H

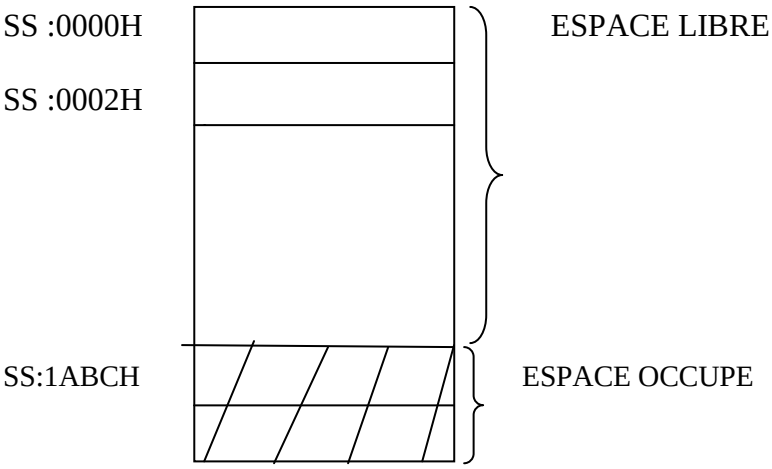
Pour les microprocesseurs 80X86, lorsqu'un mot est écrit dans la mémoire centrale (data segment), l'octet de poids faible sera chargé avant l'octet de poids fort.

Dans l'octet d'adresse FF000H on trouvera donc la valeur BCH tandis que l'octet suivant contiendra la valeur 1AH,

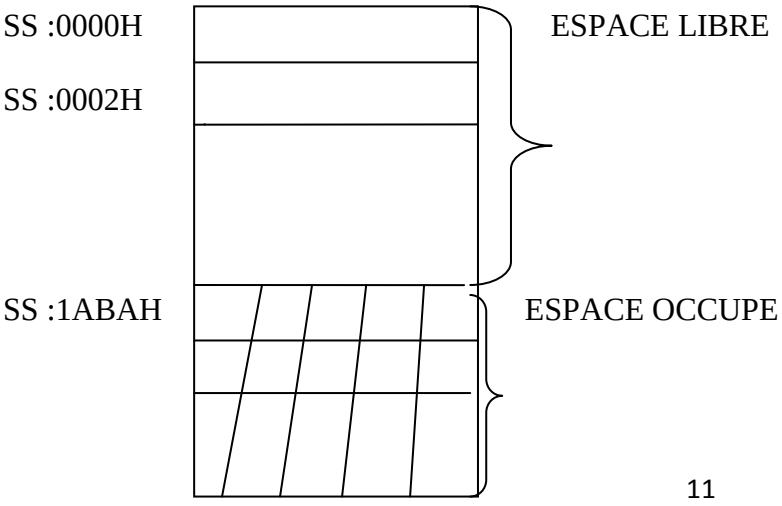
V-2) Implémentation de la pile :

Le pointeur de pile (en combinaison avec le segment de pile SS) pointe vers le dessus de la pile (TOS : top of stack,sommet de pile) en mémoire. Une pile est un ensemble de données placées en mémoire de manière à ce que seulement la donnée du "dessus" soit disponible à un instant donné. Celle-ci est pointée par le registre SP

On suppose que SS=1111H , SP=1ABCH ,



L'état de la pile, après empilement d-un mot est le suivant



V.3 : Organisation de la mémoire

La mémoire est organisée en deux Bancs (un banc pair et un banc impair chacun de 512 Octets) comme le montre la figure suivante :

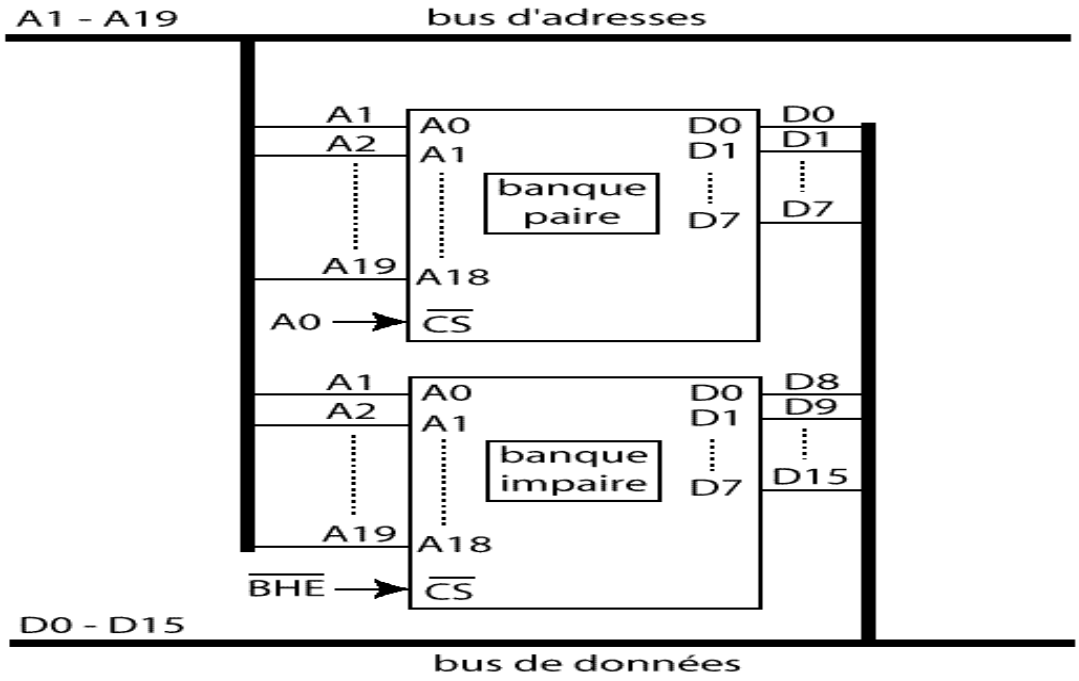


Figure 5: Organisation de la mémoire

la lecture (écriture) d'un mot d'adresse paire nécessite un seul accès, par contre celle d-un mot d'adresse impaire nécessitera deux accès, les deux octets du mot ne sont pas alignés.

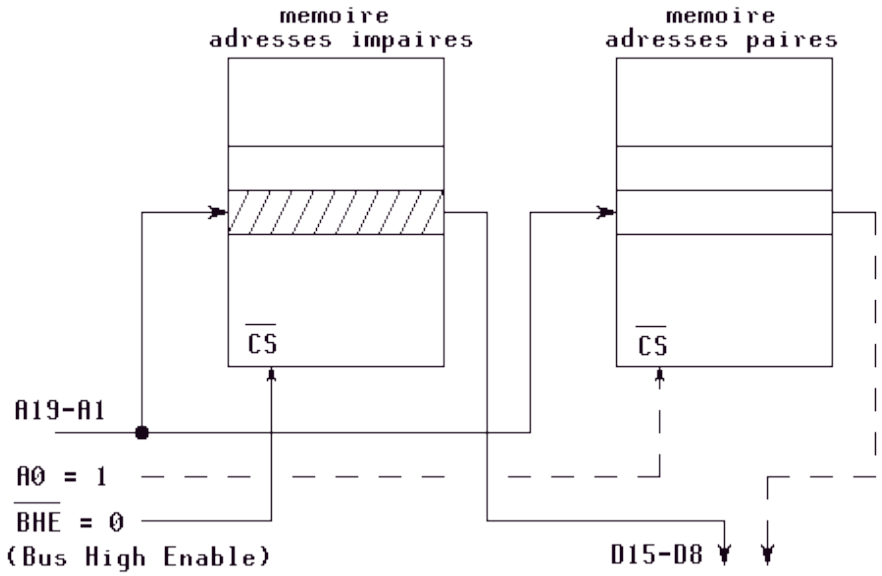
Le signal BHE/ permet de sélectionner le banc impair, pour cela il prendra la valeur 0
Le tableau suivant montre l'utilité du signal BHE/

BHE/	A0	ACCES EFFECTUE
0	0	L/E d'un mot d'@paire
0	1	L/E d'un octet d'@impaire
1	0	L/E d'un octet d'@paire
1	1	impossible

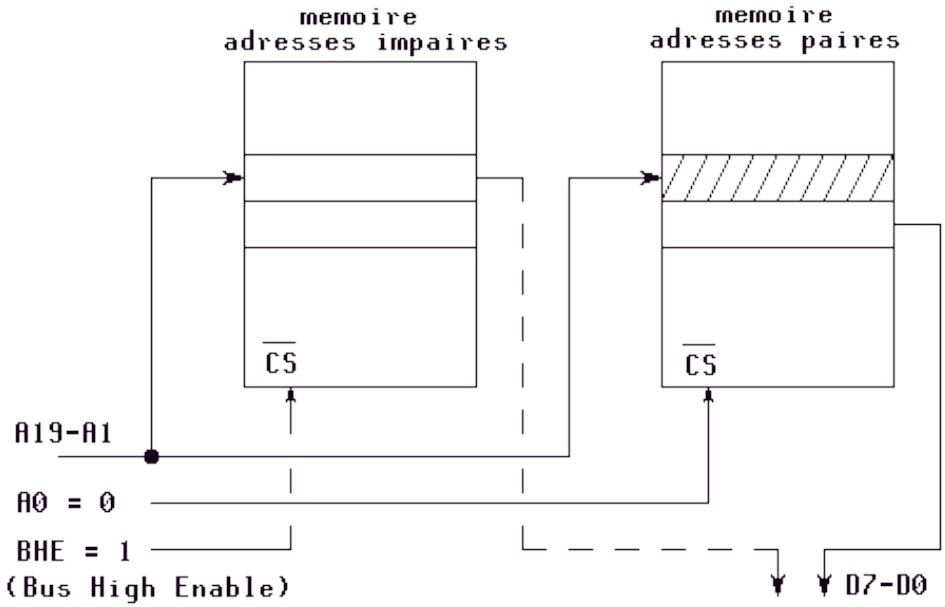
Pour lire (écrire) un mot de 16 bits qui se trouve à une adresse impaire, deux accès mémoire sont nécessaires. Le premier cycle mémoire correspond à la lecture du poids faible de la donnée, tandis que le second cycle permet la lecture du poids fort de la donnée, Ceci est illustré par le schéma suivant.

Lecture d'un mot situé à une adresse impaire,

1er cycle : Lecture du premier octet situé à une adresse impaire,:



2eme cycle : Lecture de l'octet suivant situé à une adresse paire



Architecture pipeline

Une architecture pipeline est caractérisée par la présence de plusieurs unités ou étages qui travaillent en parallèle en échangeant des informations.

L'exemple suivant nous montre l'efficacité d'une architecture pipeline

Soit un processeur où 5 cycles sont nécessaires pour accomplir une instruction

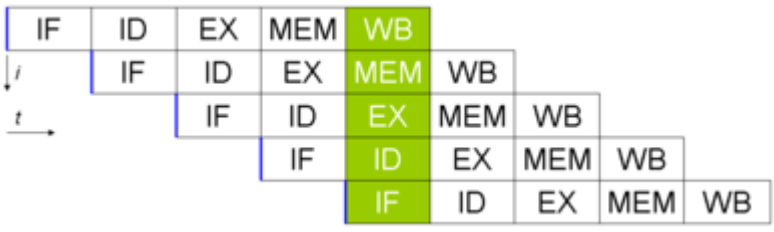
1. IF (Instruction Fetch) charge l'instruction à exécuter.
2. ID (Instruction Décode) décode l'instruction.
3. EX (Exécute) , calcul de l'adresse physique.
4. MEM (Memory), dénote un transfert depuis un registre vers la mémoire dans le cas d'une instruction du type MOV (accès en écriture) ou de la mémoire vers un registre dans le cas d'un MOV (accès en lecture).
5. WB (Write Back) stocke le résultat dans un registre. La source peut être la mémoire ou bien un registre.

En supposant que chaque étape met 1 cycle d'horloge pour s'exécuter, il faut au maximum 5 cycles pour exécuter une instruction, 15 pour 3 instructions :

Séquençage des instructions dans un processeur sans pipeline. Il faut 15 cycles pour exécuter 3 instructions.

Si l'on utilise un processeur composé de 5 étages ou unités du processeur, celui-ci peut alors contenir plusieurs instructions, chacune à une étape différente.

Les 5 instructions s'exécuteront en 9 cycles, et le processeur sera capable de terminer une instruction par cycle à partir de la cinquième, bien que chacune d'entre elles nécessite 5 cycles pour s'exécuter complètement. Ceci est illustré par le schéma suivant.



Séquençage des instructions dans un processeur doté d'un pipeline à 5 étages.

Il faut 9 cycles pour exécuter 5 instructions. À *t* = 5, tous les étages du pipeline sont sollicités, et les 5 opérations ont lieu en même temps.

Au 5^{e} cycle, tous les étages sont en cours d'exécution.