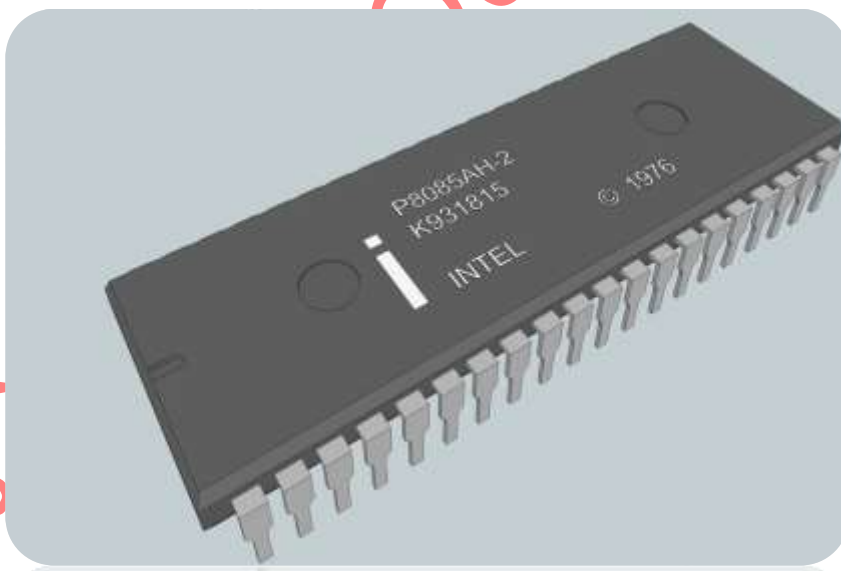
CHAPITRE 3

Etude d'un microprocesseur 8 bits



Systèmes ... OUKKOU ... IV_Jijel

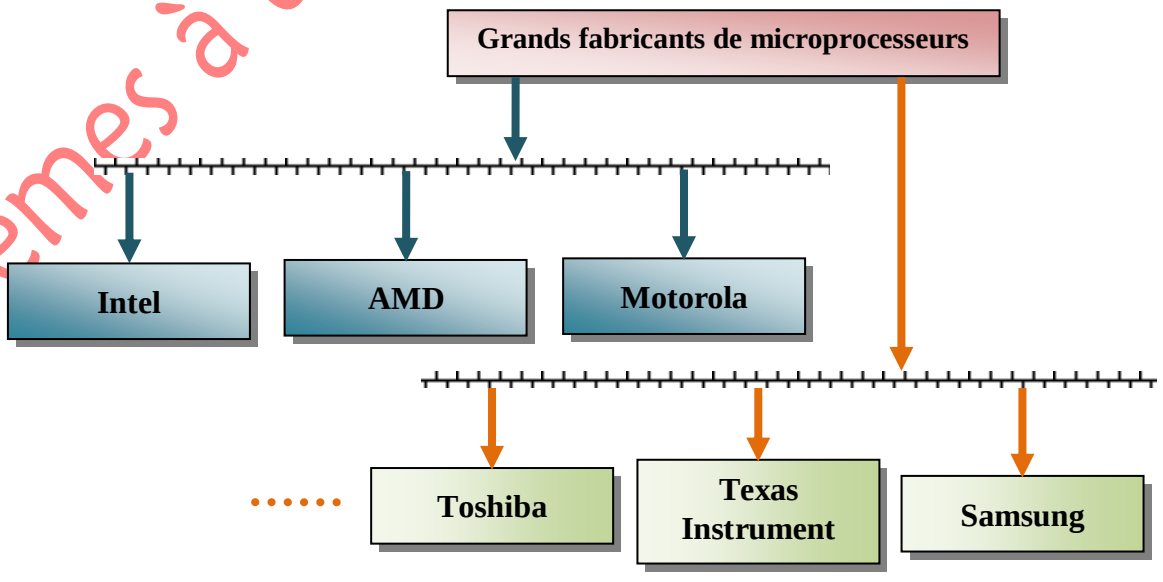
CHAPITRE 3

Etude d'un Microprocesseur 8 bits

3.1. Généralités

Le microprocesseur, (ou **CPU**) est le composant essentiel d'un ordinateur qui interprète les instructions et traite les données d'un programme.

- ✓ C'est un des composants nécessaires au fonctionnement de tous les types d'ordinateurs.
- ✓ C'est l'unité **intelligente** de traitement des informations.
- ✓ Son travail consiste à lire des programmes (des suites d'instructions), à les décoder et à les exécuter.
- ✓ Les années 80 voyaient l'émergence de ces circuits avec les Zylog Z80, 6800 de Motorola, le 8085 d'Intel qui est souvent utilisé en tant que microcontrôleur.
- ✓ Avec l'arrivée des PC-XT d'IBM et l'utilisation du 8088, Intel devenait maître du marché fin des années 80.
- ✓ Il existe des processeurs basés sur l'architecture CISC et d'autres basés sur l'architecture RISC.
- ✓ Certains processeurs sont difficilement classifiables comme le CPU i486 également appelé 80486.



Systèmes à µP Dr SOUKKOU_Univ_Jijel

L'histoire des microprocesseurs est intimement liée à celle de la technologie des semi-conducteurs. Le tableau suivant décrit les principales caractéristiques des microprocesseurs (unités centrales) fabriqués par Intel et montre la fulgurante évolution des microprocesseurs autant en augmentation du nombre de transistors, en miniaturisation des circuits et en augmentation de puissance.

Table 3.1 : *Évolution des microprocesseurs.*

Date	Nom	Nombre de Transistors	Finesse de gravure (µm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300			4 bits/4 bits bus	
1974	**8080 / 8085**	**6 000**	**6**	**2 MHz**	**8 bits/8 bits bus**	**0,64**
1979	8088/8086	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 MHz	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	25 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8	60 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35	233 MHz	32 bits/64 bits bus	300
1999	Pentium III « !!! »	9 500 000	0,25	450 MHz	32 bits/64 bits bus	510
2000	Pentium 4C	42 000 000	0,18	1,5 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09	3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,16 GHz (E8500)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200

- **Largeur des données** : Le premier nombre indique **la taille de bus de données**. Le second nombre indique **la taille de bus d'adresse**.
- **MIPS** : Le nombre de millions d'instructions complétées par le microprocesseur en une seconde.

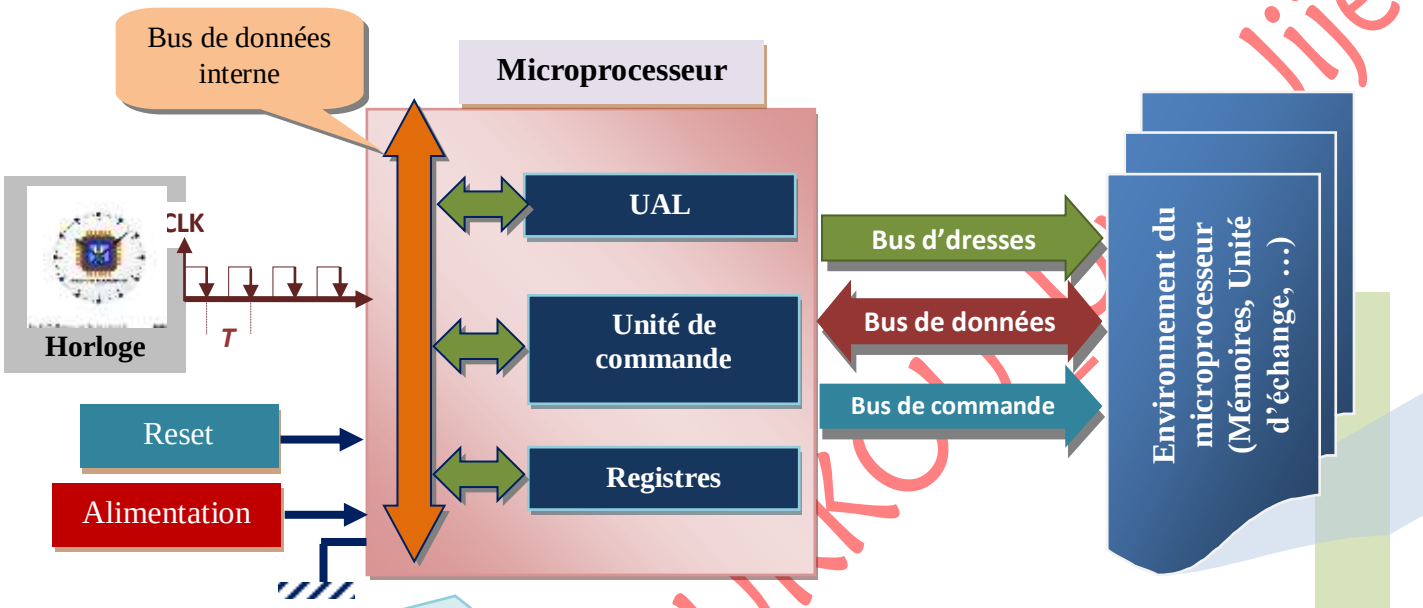
Un microprocesseur est constitué de:

- ✓ Une unité de **commande** qui lit les instructions et les décode;
- ✓ une unité de **traitement** (**UAL** - unité arithmétique et logique) qui exécute les instructions;
- ✓ D'un ensemble de mémoire appelés **registres**;
- ✓ D'un **bus de données** externe;
- ✓ D'un **bus d'adresse** externe;

Systèmes à microprocesseur

- ✓ D'un **bus de commande** externe;
- ✓ D'un bus de données interne reliant l'unité de commande, l'UAL et les registres.

Lorsque tous ces éléments sont regroupés sur une même puce, on parle alors de **Microprocesseur**. La Figure 3.1 donne une idée sur l'architecture interne d'un microprocesseur. Sur cette figure nous pouvons voir les 3 bus qui permettent au microprocesseur de communiquer avec l'extérieur.



Les principales caractéristiques d'un microprocesseur sont :

- Le fabriquant du circuit (Intel, AMD, …).
- Le type de boitier DIP, PLCC, …).
- Technologies de fabrication : NMOS, PMOS, CMOS.
- La **complexité de son architecture → Echelle d'intégration**. Cette complexité se mesure par le nombre de **transistors** contenus dans le microprocesseur.
- Le nombre de **bits** que le processeur peut traiter en une instruction (**taille des registres internes du microprocesseur ou la taille de bus de données**) (**4, 8, 16, 32, ...**).
- La taille de bus d'adresse, qui permet de définir l'espace mémoire accessible par le microprocesseur.
- **Le jeu d'instructions** qu'il peut exécuter. Un processeur peut exécuter plusieurs douzaines d'instructions différentes.
- **La vitesse maximale de l'horloge** qu'il peut supporter → **Vitesse d'exécution des instructions**.
- **Consommation d'énergie.**
- **…**
- **…**
-

Figure 3.1 : *Architecture schématique d'un microprocesseur.*

3.2. Les différentes familles des microprocesseurs 8 bits

Les fabricants des microprocesseurs 8 bits les plus connus sont :

- ✓ Intel,
- ✓ Zilog,
- ✓ Motorola et

✓ National Semiconductor,

comme schématise la figure ci-dessous.

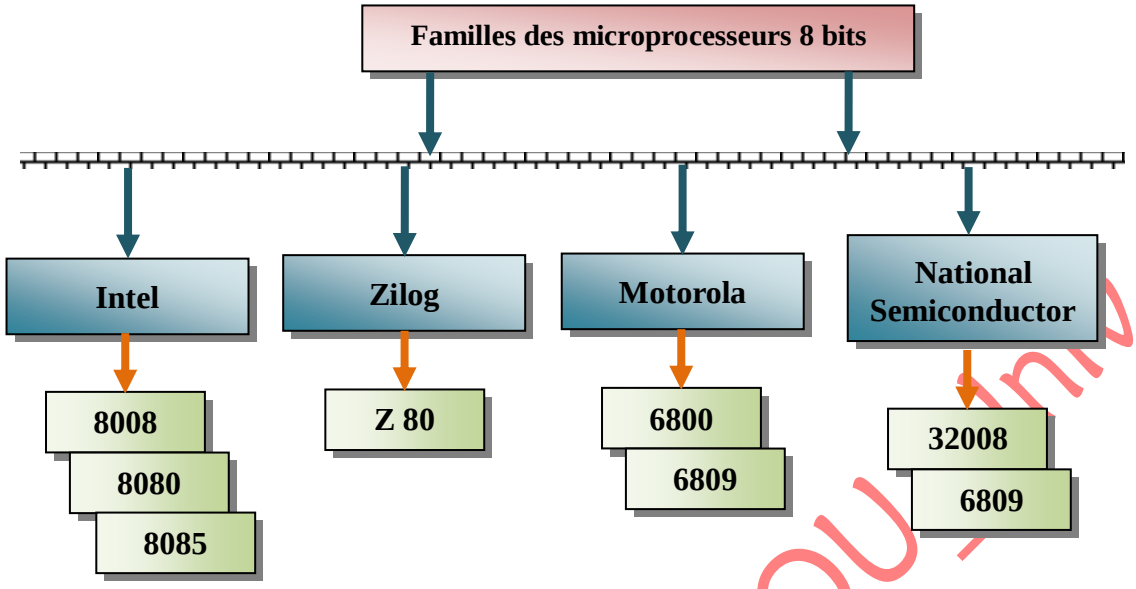


Figure 3.2 : *Les différentes familles des microprocesseurs 8 bits.*

3.3. Etude d'un microprocesseur 8 bits : 8088/8085

L'**Intel 8085** est un microprocesseur **8 bits** fabriqué par Intel au milieu des années 1970.

- ✓ Il était compatible au niveau du code binaire avec le plus célèbre Intel 8080, mais demandait moins de matériel environnant, ce qui permit la création de micro-ordinateurs plus simples et moins chers à construire.
- ✓ Disponible en version à 40 broches
- ✓ Le « **5** » dans le numéro du modèle provient du fait que les 8085 exigeaient seulement une alimentation de +5V plutôt que les +5V, -5V et +12V exigés par les 8080.
- ✓ Il existe en plusieurs versions 8085A, 8085AH, 8085AH-1 et 8085AH-2.
- ✓ Cependant, il était **plus lent** que le 8080.
- ✓ Fonctionnement à 3 MHz, 5 MHz et 6 MHz ;
- ✓ 1,3 µs par cycle d'instruction pour le 8085AH, 0,8 µs pour 8085AH-2 et 0,67 µs pour le 8085AH-1 ;
- ✓ Générateur d'horloge interne (avec quartz externe ou réseau LC ou RC) ;
- ✓ Possède 4 vecteurs d'interruption dont 1 non masquable
- ✓ Le **8088/8085** ont parfois été utilisés dans des ordinateurs basés sur le système d'exploitation CP/M.
- ✓ Ils furent par la suite supplantés par le **Zilog Z80**, compatible et plus efficace, qui remporta la majeure partie du marché des ordinateurs CP/M et des ordinateurs personnels du milieu et de la fin des années 1980.
- ✓ Le **8085** fut utilisé ultérieurement comme microcontrôleur (surtout grâce au coût réduit des composants).
- ✓ Ainsi, il équipait le dispositif à bandes DECtape et le terminal vidéo VT100.
- ✓ Il continua donc à être produit pendant toute la durée de vie de ces produits.
- ✓ De même, il fut embarqué sur le robot de la mission **Mars Pathfinder**.
- ✓ Il est actuellement encore utilisé dans l'enseignement.

Systèmes à µP Dr. SOUKKOU Univ_Jijel

3.3.1. Brochage du up8085

Le microprocesseur 8085 (8085A : version légèrement révisée du 8085) est fabriqué sous forme de boitier DIP à 40 broches comme illustrée par la Figure 3.3.

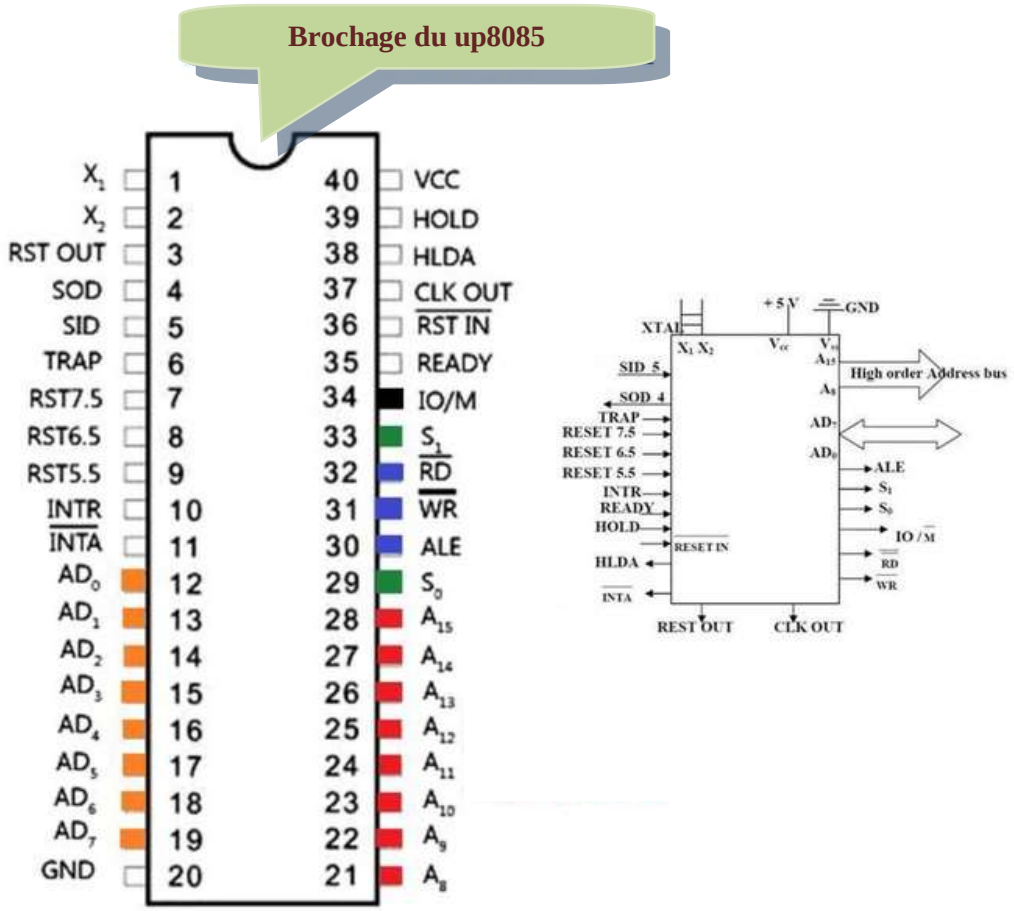


Figure 3.3 : *Brochage du microprocesseur 8085.*

- 16 broches pour le bus d'adresse → Espace mémoire adressable $\mathbf{2^{16} = 64\ Ko}$.
- 08 broches pour le bus de donnée.
- La fonction de chaque broche du up8085 est donnée par le tableau 4.2.

Table 3.2 : *Fonctions des broches du up8085.*

Broche	Type (E/S)	Description
AD0-AD7	Bidirectionnel, 3-états	✓ Bus d'adresse / données.
A8-A15	Sortie, 3-états	✓ Bus d'adresse.
ALE	Sortie à 3 états	✓ Validation de verrou d'adresse.
$\overline{RD}$	Sortie, 3-états	✓ Commande de lecture.
$\overline{WR}$	Sortie, 3-états	✓ Commande d'écriture.
IO/$\overline{M}$	Sortie, 3-états	✓ Indicateur E/S ou mémoire.
S0, S1	Sortie	✓ Indicateurs d'état de bus.
READY	Entrée	✓ Requête de mode attente.
SID	Entrée	✓ Entrée de données sérielles.

Système à up Dr. A. Soukkou

SOD	Sortie	✓ Sortie de données sérielles.
HOLD	Entrée	✓ Requête d'attente.
HOLDA	Sortie	✓ Accusé de réception d'attente.
INTR	Entrée	✓ Requête d'interruption.
TRAP	Entrée	✓ Requête d'interruption non masquable.
RST 5.5 **RST 6.5** **RST 7.5**	Entrée	✓ Requêtes d'interruption matérielles vectorisées.
$\overline{\textbf{INTA}}$	Sortie	✓ Accusé de réception d'interruption.
$\overline{\textbf{TESET IN}}$	Entrée	✓ Réinitialisation système.
RESET OUT	Sortie	✓ Réinitialisation périphériques.
X1, X2	Entrée	✓ Connexions cristal ou RC.
CLK	Entrée	✓ Connexions cristal ou RC.
VCC, GND	/	✓ Alimentation, Masse.

Les sorties **IO/$\overline{\textbf{M}}$**, **S0** et **S1** sont des signaux de commande qui informent les périphériques du type de cycle machine que le up8085 est en train d'exécuter. Le tableau ci-dessous illustre les combinaisons correspondantes de signaux de sorties des broches **IO/$\overline{\textbf{M}}$**, **S0** et **S1**.

Table 3.3 : *Combinaisons correspondantes de signaux de sorties des broches* **IO/$\overline{\textbf{M}}$**, ***S0*** *et* ***S1*** *du up8085.*

Broches			Etat du cycle machine
IO/$\overline{\textbf{M}}$	**S1**	**S0**	
0	**0**	**0**	✓ Ecriture mémoire.
0	**1**	**0**	✓ Lecture mémoire.
1	**0**	**1**	✓ Ecriture E/S.
1	**1**	**0**	✓ Lecture E/S.
0	**1**	**1**	✓ Extraction code opération (COP).
1	**1**	**1**	✓ Accusé réception d'interruption.
Hi-z	**0**	**0**	✓ Halte.
Hi-z	**x**	**x**	✓ Attente.
Hi-z	**x**	**x**	✓ Réinitialisation.

Hi-z : Etat de haute impédance.
X : Etat non spécifié.

3.3.2. Architecture interne du up8085

La figure ci-dessous illustre l'architecture interne du up8085.

Systèmes à ... SOUKKOU ... Jijel

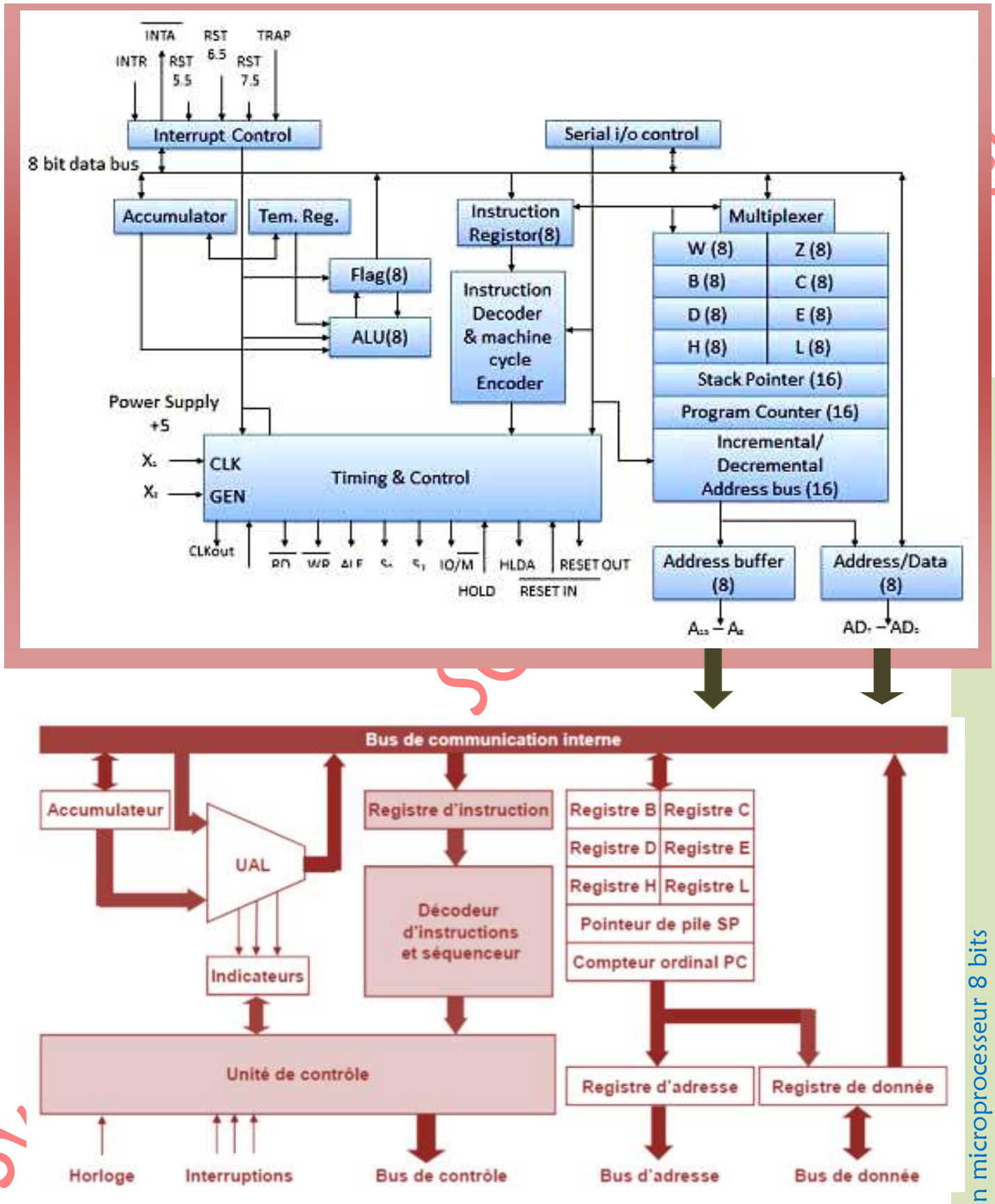
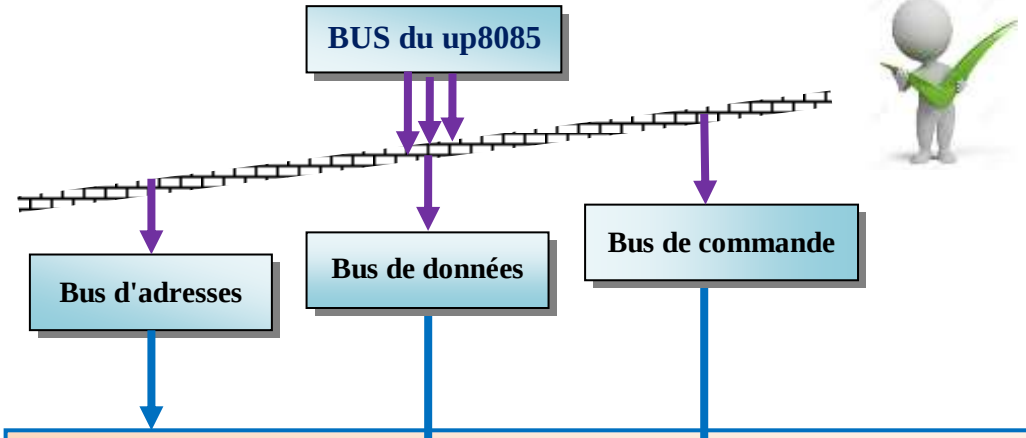


Figure 3.4 : *Architecture interne du microprocesseur 8085.*



- ✓ Se compose de **16 lignes d'adresse**: **A0… A15.**
- ✓ Fonctionne en mode continu (unidirectionnel): Les bits d'adresse sont toujours envoyés du up8085 aux périphériques, et pas l'inverse.
- ✓ 16 lignes d'adresse sont capables d'adresser un espace mémoire total de 216 = 65.536 (64 Ko) emplacements dans la mémoire
- ✓ Adresses mémoire: De 0000_{HEX} à $FFFF_{HEX}$.

- ✓ Se compose de **8 lignes de données**: **D0… D7.**
- ✓ Fonctionne en mode bidirectionnel: Les bits d'informations sont envoyés du up8085 aux périphériques, aussi bien que des périphériques au MPU.
- ✓ Plage des données: 00_{HEX} – FF_{HEX}.

- ✓ Le bus de commande est utilise aux transferts de signaux de commande.
- ✓ Il se compose de diverses lignes portant les signaux de commande tels que l'activation de droit de lecture/écriture, les bits de drapeau.
- ✓ Les signaux de commande typiques sont:
  - ➢ Lecture de la mémoire.
  - ➢ L'écriture en mémoire.
  - ➢ La lecture E/S.
  - ➢ L'écriture E/S.

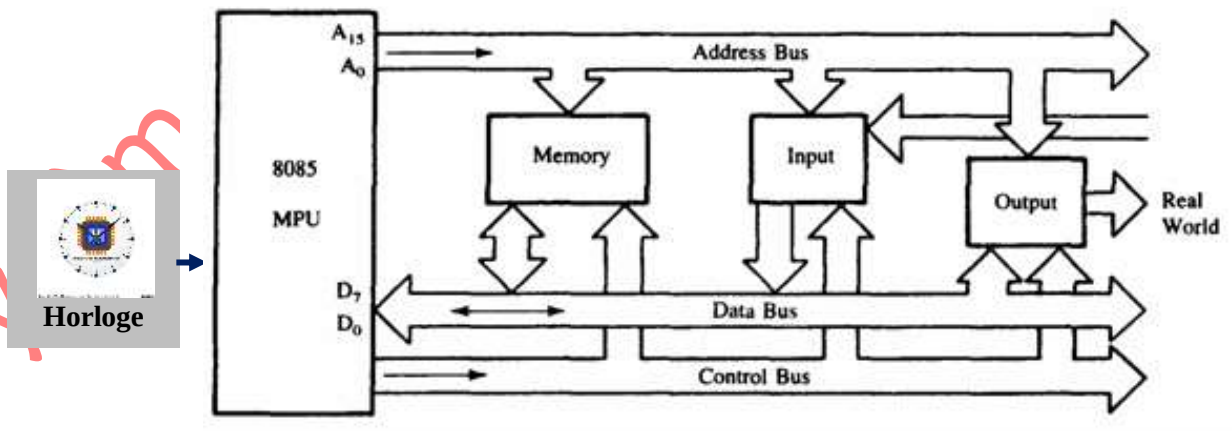


Figure 3.5 : *Bus du microprocesseur 8085.*

L'unité centrale (CPU - Central Processing Unit) regroupe 5 blocs fonctionnels :

- Unité de contrôle.
- Unité arithmétique et logique.
- Registres généraux.
- Registres spécialisés.
- Interfaçage avec le monde extérieur.

Unité de contrôle

- Elle contrôle la totalité du fonctionnement de l'unité centrale :
  - Lecture, décodage, et exécution des instructions.
  - Lecture et écriture des données en mémoire centrale.
  - Lecture et écriture des registres.
  - Contrôle de l'unité arithmétique et logique.
  - Contrôle de l'interface avec l'extérieur :
    - Bus d'adresse et de données.
    - Fonctions d'accès à la mémoire centrale.
    - Interruptions, ...

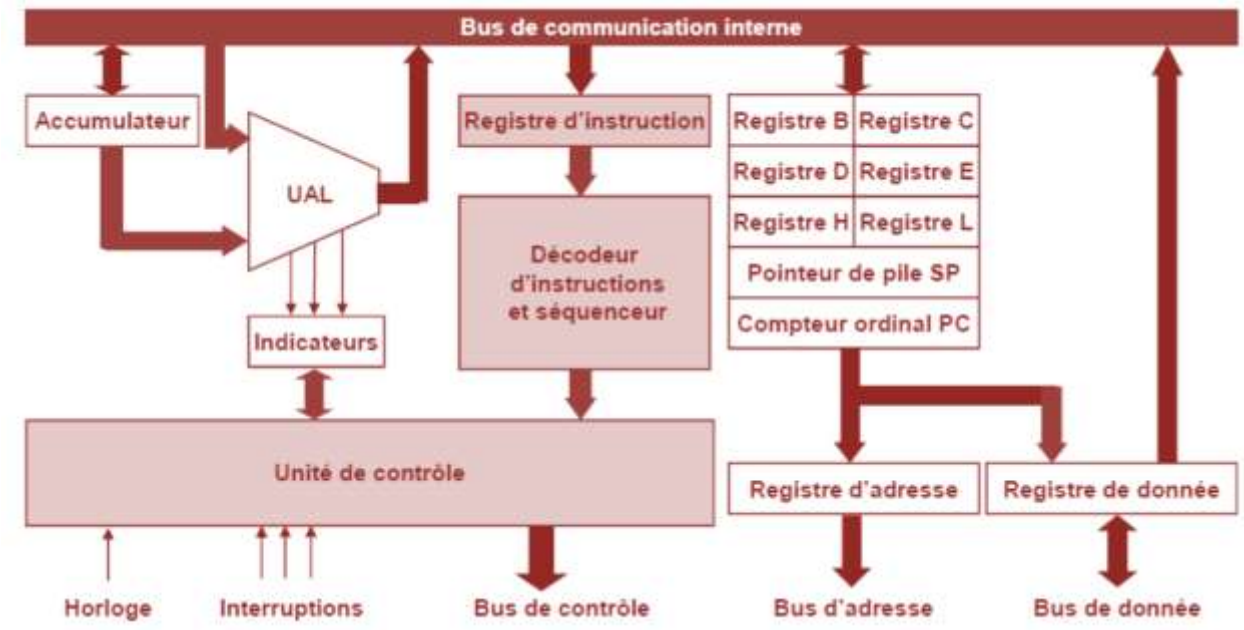


Figure 3.6 : *Unité de contrôle du microprocesseur 8085.*

Unité arithmétique et logique (UAL)

- **Rôle** : Calcul d'opérations élémentaires :
  - Opérations arithmétiques :
    - Addition, soustraction, multiplication, division.
    - Changement de signe.
  - Opérations logiques :
    - ET, OU, OU exclusif, Négation.
    - Décalage, rotation, …
- Traite des mots de taille fixe (1, 2, 4 octets)
- Génère les indicateurs

Systèmes à ... Univ_Jijel

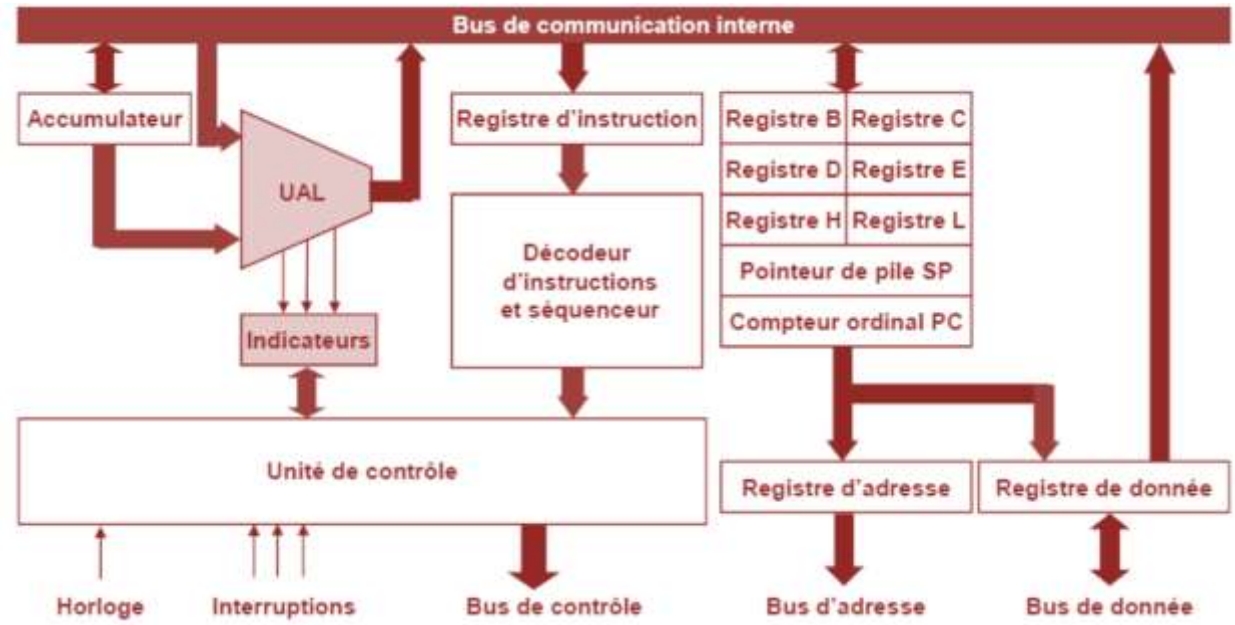


Figure 3.7 : *UAL du microprocesseur 8085.*

Registres :

- **Registres généraux**
  - Opérandes pour l'unité arithmétique et logique.
  - Résultats des calculs (accumulateur).
  - En nombre variable (2 à plusieurs dizaines).
  - Taille = taille des mots traités par l'UAL.

- **Registres spécialisés**
  - Compteur ordinal (Program Counter - PC).
  - Registre d'état (Status Register - SR) → PSW.
  - Pointeur de pile (Stack Pointer - SP).

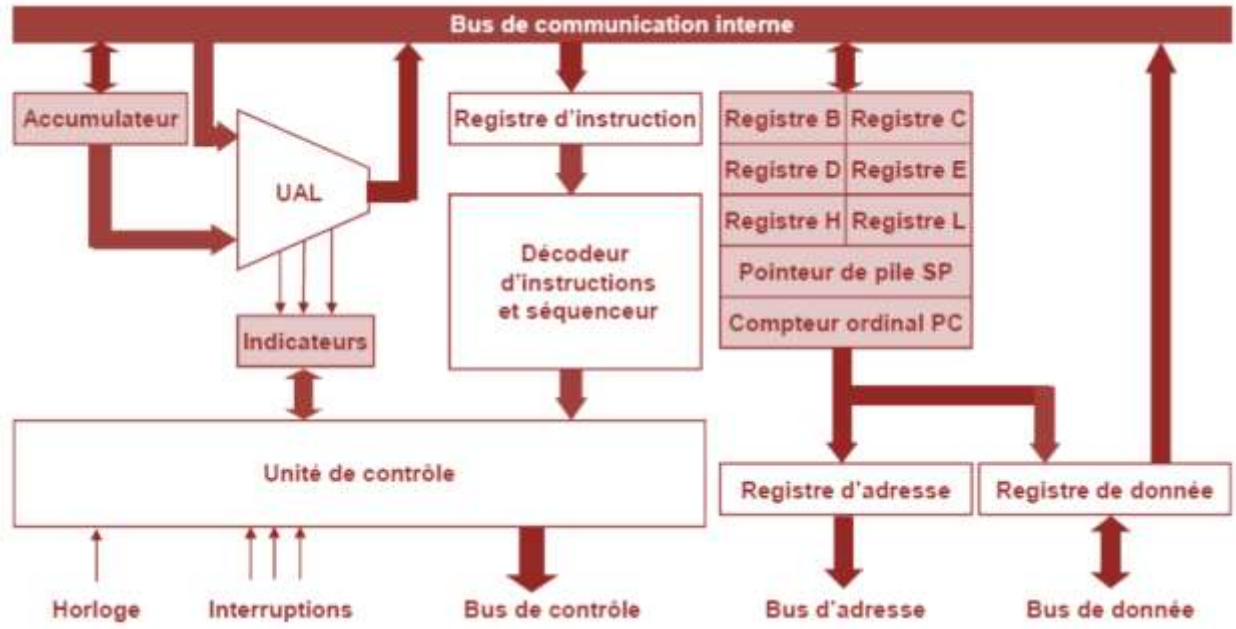
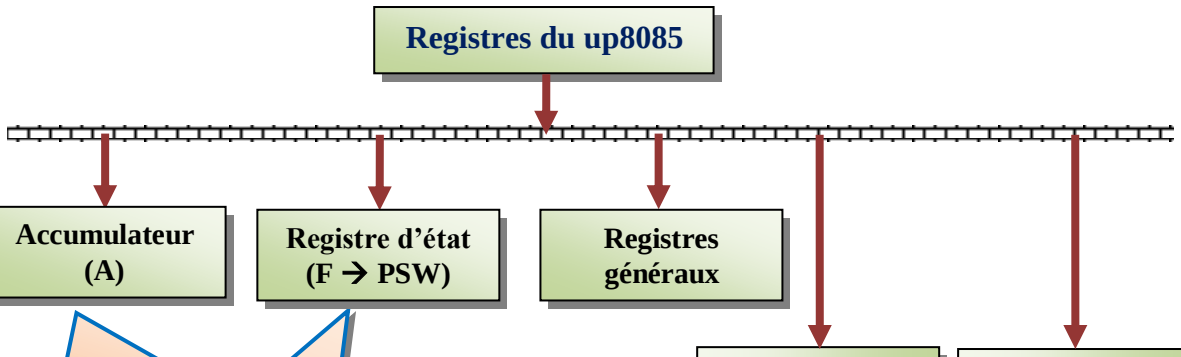


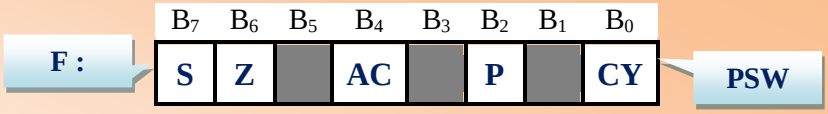
Figure 3.8 : *Registres du microprocesseur 8085.*

Dr. SOUKKOU – Cours



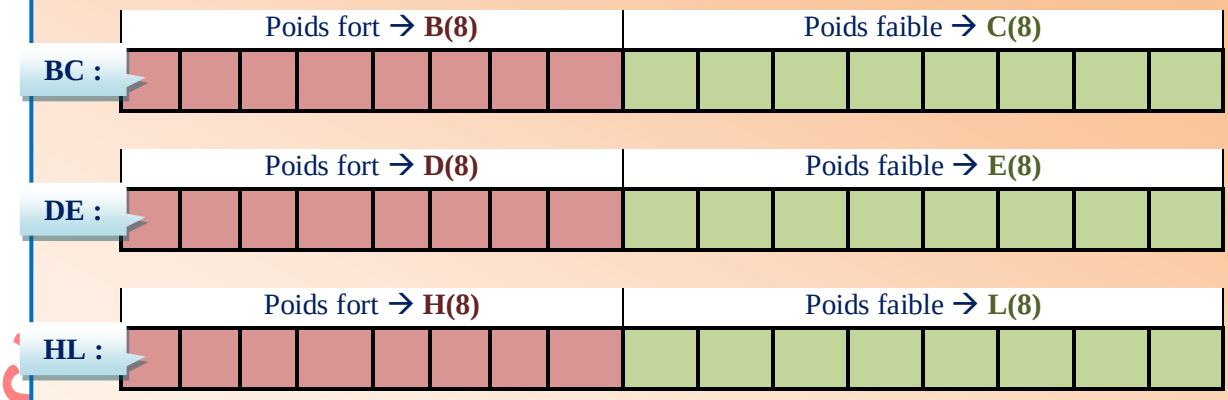
✓ A (registre 8 bits) est le nœud de toutes les opérations du up 8085

✓ Registre drapeau (**F**): comporte 5 drapeaux (sollicite seulement pour les opérations mathématiques) qui sont:



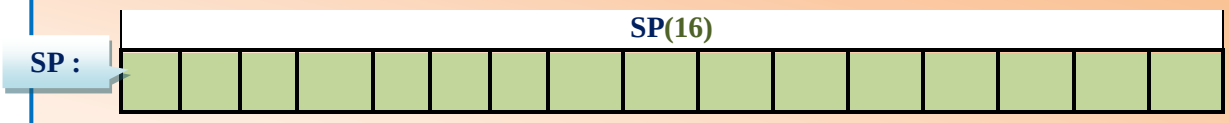
- **S**- Sign bit ou signe bit (le tout premier bit): il est soit 0 ou 1.
- **Z**- zero bit (prend la valeur 1 quand des 8 bits, aucun n'est 0).
- **AC**- Auxillary Carry bit (prend la valeur 1 si s'ajoute 1 au 5e bit du milieu allant de la droite vers la gauche dans une addition mathématique).
- **P**- parity bit ou bit de parité (prend la valeur 1 le nombre total des bits en 1 est paire).
- **CY**- carry bit (prend la valeur 1 si l'addition ne nous amène pas a 9 bits au lieu de 8).

✓ **Les registres généraux :**



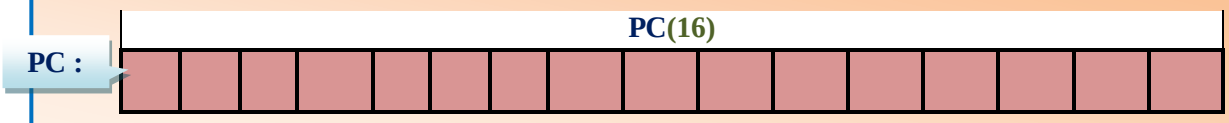
- Quelques instructions utilisent les paires BC et DE comme pointeurs d'adresse.
- La paire HL (dite pointeurs de données par Intel) peut être utilisée pour pointer sur les adresses.

✓ **Pointeur de PILE ou stack pointer (SP):**

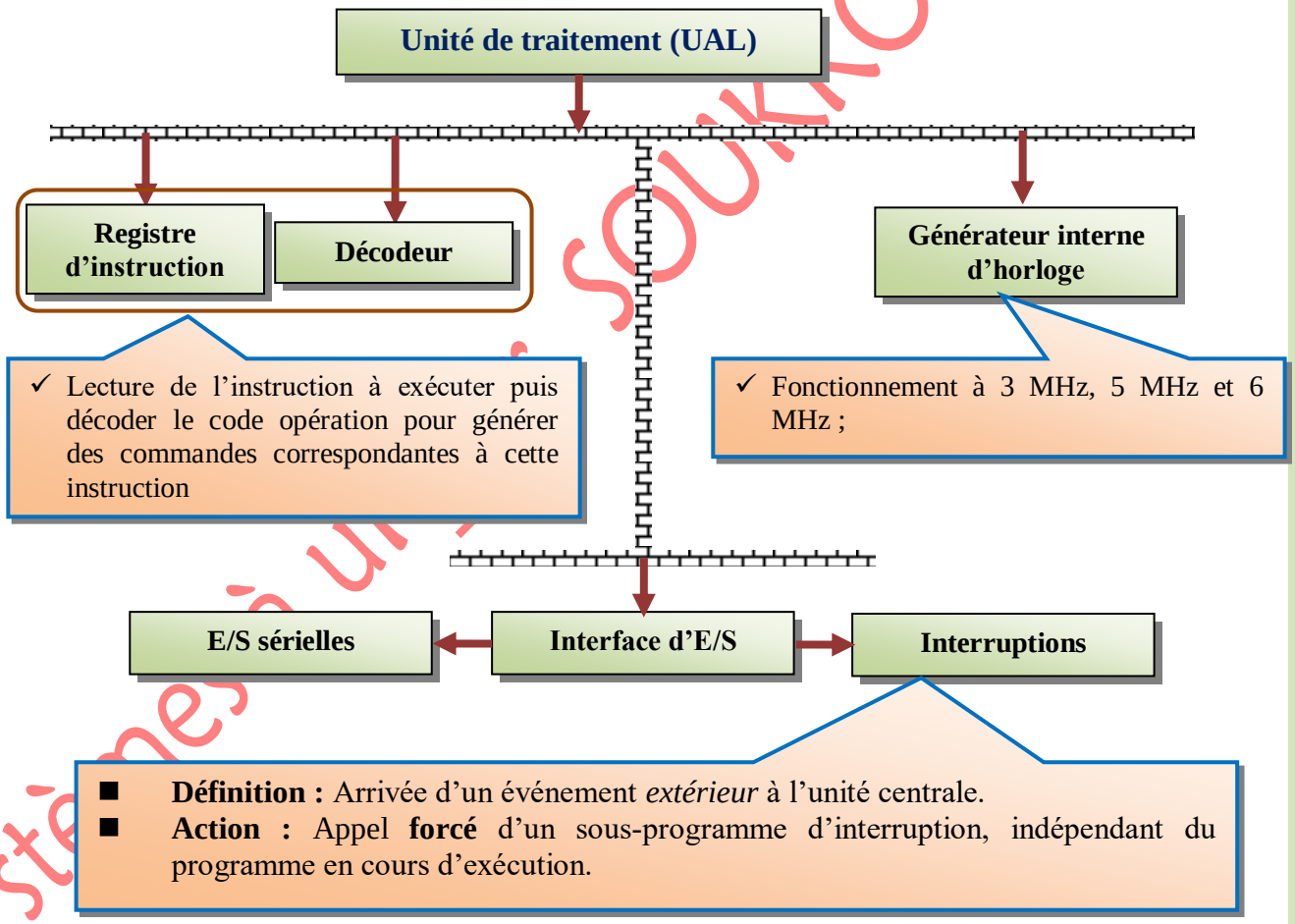


- SP ou pointeur d'adresse (ou de données) qui pointe toujours sur le sommet de la pile dans la RAM.

✓ **Compteur d'instruction (Program Counter) (PC):**



- PC pointe toujours sur la case mémoire de la prochaine l'instruction à exécuter.



La mémoire centrale (RAM) peut être représentée par le schéma ci-dessous. La taille adressable par le up8085 est de 2^{16} = 64 Ko.

Systèmes à µ... SOUKKOU ...yel

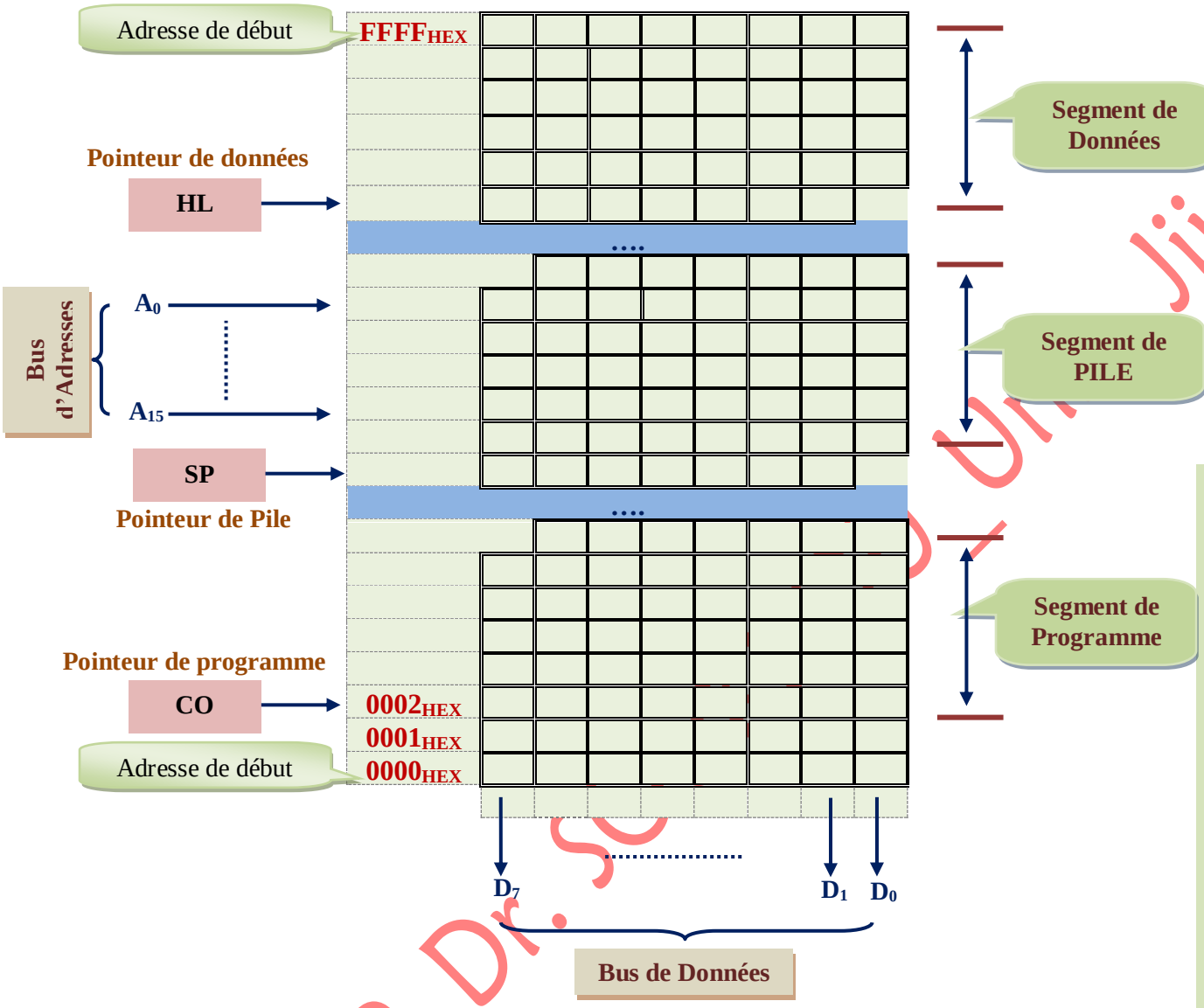


Figure 3.9 : *Segmentation de la mémoire du up8085.*

- **AD0- AD7: 8** bits de poids faible du bus d'adresses, **multiplexés** avec 8 bits de données.
- Le bus AD est multiplexé (multiplexage temporel) d'où la nécessité d'un **démultiplexage** pour obtenir séparément les bus d'adresses et de données:
  - ✓ 8 bits de données (microprocesseur 8 bits).
  - ✓ 16 bits d'adresse d'où 2^{16} = 64 Ko d'espace mémoire adressable par le up8085.
- Le démultiplexage des signaux **AD0- AD7** se fait en mémorisant l'adresse lorsque celle-ci est présente sur le bus A/D, à l'aide d'un **VERROU (LATCH)** → Ensemble des bascules **D**.
- La commande de mémorisation de l'adresse est générée par le up8085 est le signal **ALE** (Adresse Latch Enable).
  - ✓ ALE = 1, le circuit latch est transparent (Q = D).
  - ✓ ALE = 0, mémorisation de la dernière valeur D sur les sorties Q.

Les signaux de lecture (RD) ou d'écriture (WR) ne sont générés par le up8085 que lorsque les données sont présentes sur le bus A/D.

Systèmes à µP Dr. SOUKKOU Ahmed Djamel Eddine Jijel

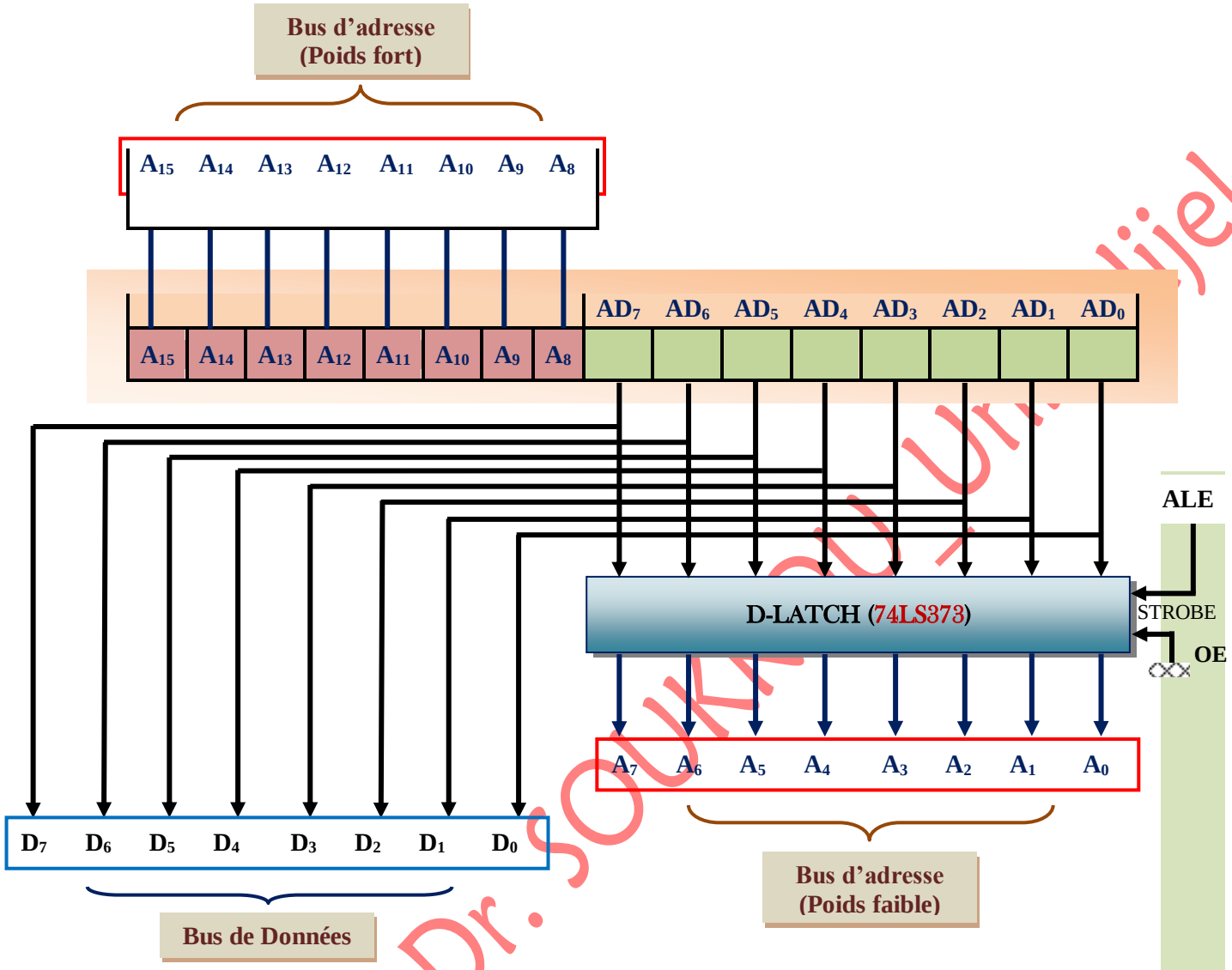


Figure 3.10 : *Démultiplexage des lignes d'adresses/données du up8085.*

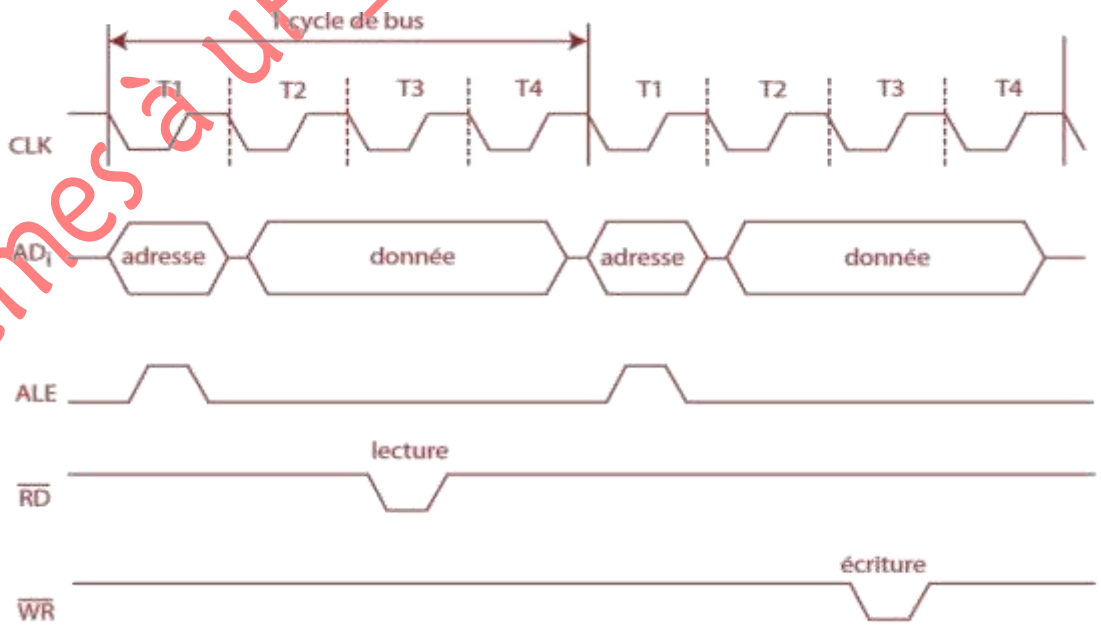


Figure 3.11 : *Chronogramme de bus d'adresse/données.*

Systèmes à µP_Dr. SOUKKOU Université

Exemple :

- Conception d'un système à base du up8085 avec :
  - 32 Ko de mémoire RAM, en utilisant des mémoires RAM 2 x 16Ko.
  - 32 Ko de mémoire ROM, en utilisant des mémoires ROM 2 x 16Ko.

➔ 04 circuits ayant 14 lignes d'adresses pour chacun.

Le tableau ci-dessous illustre l'espace mémoire réservé pour chaque circuit mémoire. Les signaux de commande des boitiers (CS) sont générés par le décodeur d'adresse 2x4 (ou 3x8), comme suit :

$$\begin{cases} \textbf{ROM 1} & \rightarrow \overline{CS} = \overline{A_{14}} \cdot \overline{A_{15}} \\ \textbf{ROM 2} & \rightarrow \overline{CS} = A_{14} \cdot \overline{A_{15}} \\ \textbf{RAM 1} & \rightarrow \overline{CS} = \overline{A_{14}} \cdot A_{15} \\ \textbf{RAM 2} & \rightarrow \overline{CS} = A_{14} \cdot A_{15} \end{cases}$$

Décodage		Adresses																
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Espace mémoire**	**Adresse**	**Circuit**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0000H**	**Début**	**ROM 1**
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**3FFFH**	**Fin**	
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**4000H**	**Début**	**ROM 2**
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**7FFFH**	**Fin**	
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**8000H**	**Début**	**RAM 3**
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
1	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**BFFFH**	**Fin**	
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**C000H**	**Début**	**RAM 4**
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-		
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**FFFFH**	**Fin**	

Syst... OU_Univ_Jijel

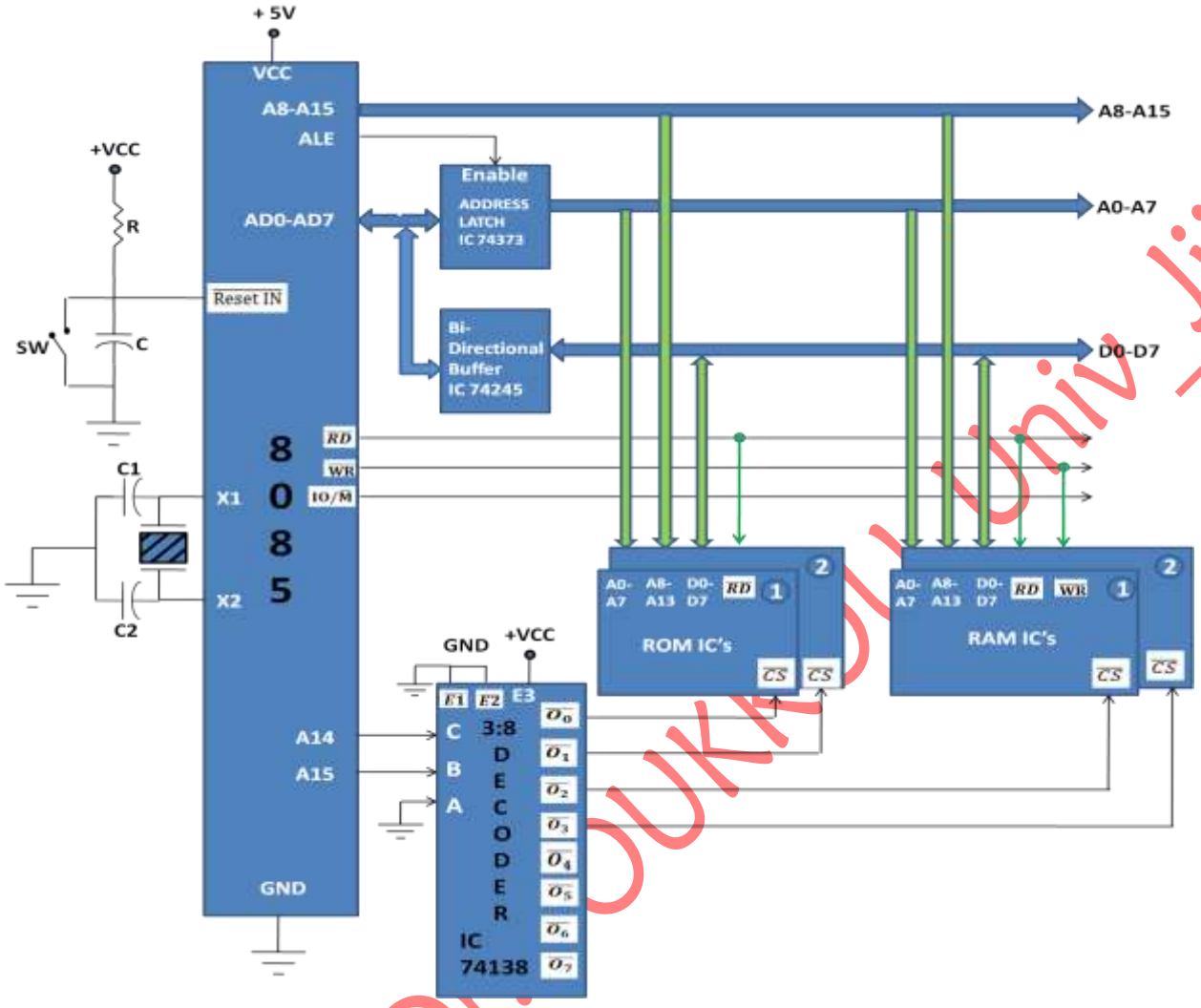


Figure 3.12 : *Interfaçage up8085-mémoires.*

Le up8085 possède cinq broches d'interruptions :

- ✓ L'interruption **INTR** correspond à l'interruption du 8080 avec laquelle elle est compatible.
- ✓ **RST5.5** est une interruption masquable.
- ✓ **RST6.5** est une interruption masquable.
- ✓ **RST7.5** est une interruption masquable.
- ✓ **Trap** est une interruption non masquable.

Les instructions masquables peuvent être activées ou désactivées toutes ensembles en utilisant les instructions **EI** et **DI**.

Les interruptions **RST5.5**, **RST6.5** et **RST7.5** peuvent être activées ou désactivées individuellement en utilisant l'instruction **SIM**.

a) Modes d'adressage

- Permettent de **localiser précisément** les opérandes d'une instruction → Cheminement des informations (données).
- Certaines architectures offrent les instructions disponibles avec différents modes d'adressage, d'autres possèdent des modes d'adressage spécifique à chaque instruction.
- Le up8085 utilise 5 modes d'adressage, à savoir :

Systèmes à µP_Dr. SOUKKOU_Univ_Jijel

1. Adressage implicite.
2. Adressage immédiat.
3. Adressage par registre.
4. Adressage Direct.
5. Adressage indirect.

Par la suite, on utilise la nomenclature de la table ci-dessous.

Table 3.4 : *Nomenclature utilisée.*

Nomenclature	Description
OS	✓ Opérande source.
OD	✓ Opérande destination.
INST	✓ Instruction (COP).
R	✓ Registre.
M	✓ Case mémoire.
adr	✓ Adresse mémoire.
[adr]	✓ Contenu d'une adresse mémoire.
val	✓ Valeur numérique.
OP	✓ Opérande.

OD ← (OD) INST (OS) ;

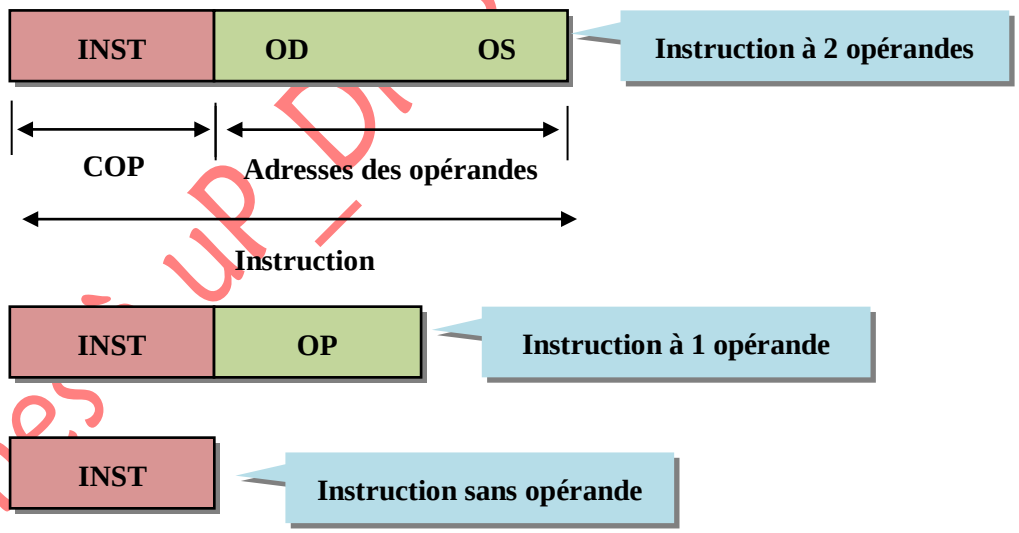


Figure 3.13 : *Différents types des instructions du up8085.*

Table 3.5 : *Modes d'adressage du up8085.*

Mode	Instruction	Exemple	Description
Adressage implicite	**INST R ;**	**INC A ;**	✓ La valeur ou la localisation de(s) opérandes est spécifiée directement dans l'instruction.

Systèmes_up_Dr_SOUKKOU_Univ_Jijel

			✓ Il s'agit d'instructions qui travaillent sur des registres spécifiques.
Adressage immédiat	**INST R/M, val**	**ADD A, #3C ;**	✓ Les données à traiter sont disponibles dans l'instruction elle-même.
Adressage par registre	**INST R, R ;**	**ADD A, BC ;**	✓ Dans ce type d'adressage, les données d'un registre sont passées a un autre registre
Adressage direct	**INST R, [adr] ;** **INST [adr] ,R ;**	**ADD A, 6EH ;**	✓ L'adresse des données est spécifiée dans l'instruction elle-même.
Adressage indirect	**INST R, [R] ;** **INST [R], R;**	**ADD A, @R0 ;**	✓ L'adresse de la donnée est disponible dans le registre spécifié.
Adressage indexé	/	**JMP @A +DPTR**	✓ Ce mode est similaire au mode indirect sauf que cette fois, l'adresse finale est obtenue par l'addition d'un index à une adresse de base.

3.4. Jeu d'instruction du up8085

Intel groupe les instructions du 8080/8085 sous les rubriques mentionnées ci-dessous.

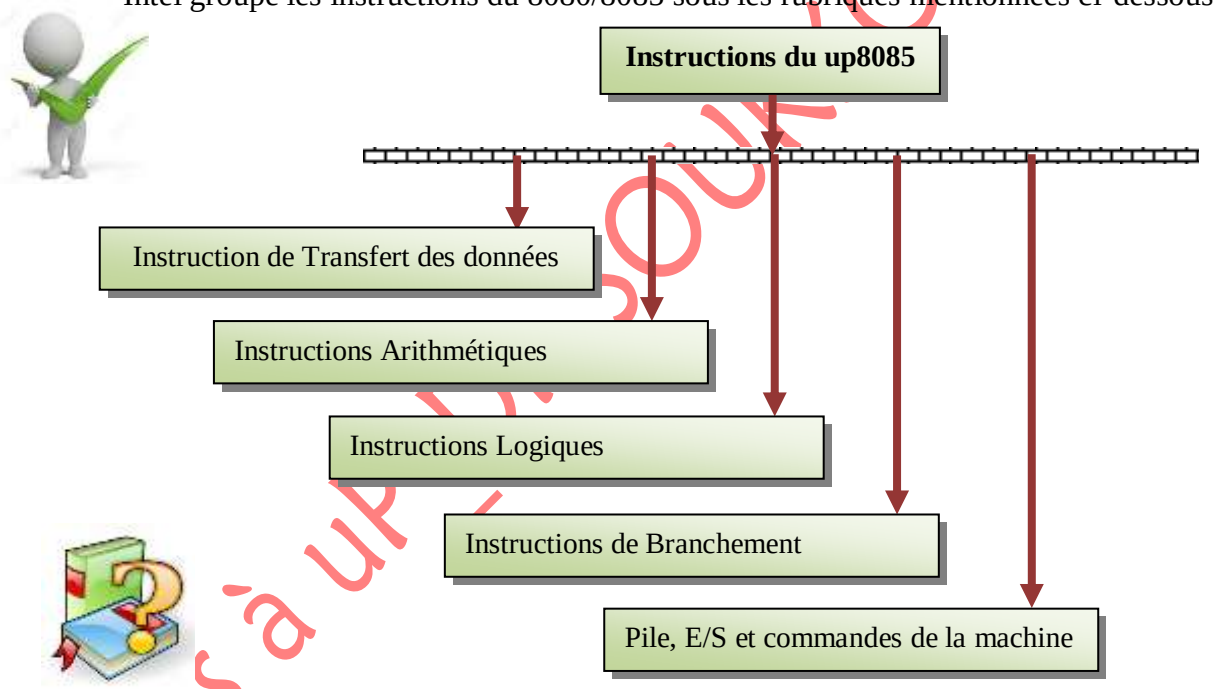


Figure 3.14 : *Groupes d'instructions du up8085.*

Table 3.6 : *Jeu d'instruction du up8085.*

	COP	Code mnémonique	Description
Instructions de Transfert des données	**MOV**	MOV R1, R2 ;	✓ Déplacer le contenu d'un registre vers un autre. **[R1] ← [R2].**
		MOV R, M ;	✓ Déplacer le contenu d'une case mémoire vers un registre. **[R] ← [M].**
		MOV M, R ;	✓ Déplacer le contenu d'un registre vers une case mémoire. **[M] ← [R].**
	MVI	MVI R, data ;	✓ Déplacer une donnée **immédiate** vers un registre. **[R] ← data.**

Systèmes à up - SOUKKOU - Cours

		MVI M, data ;	✓ Déplacer une donnée **immédiate** vers une case mémoire. **M ← data.**
	LXI	LXI R, data 16 ;	✓ Load register pair immediate. **[R] ← data 16 bits, [R] ← 8 LSBs of data.**
	LDA	LDA addr ;	✓ Load Accumulator direct. **[A] ← [addr].**
	STA	STA addr ;	✓ Store accumulator direct. **[addr] ← [A].**
	LHLD	LHLD addr ;	✓ Load H-L pair direct. **[L] ← [addr], [H] ← [addr+1].**
	SHLD	SHLD addr ;	✓ Store H-L pair direct. **[addr] ← [L], [addr+1] ← [H].**
	LDAX	LDAX R ;	✓ LOAD accumulator indirect. **[A] ← [[R]].**
	STAX	STAX R ;	✓ Store accumulator indirect. **[[R]] ← [A].**
	XCHG	XCHG ;	✓ Exchange the contents of H-L with D-E pair. **[H-L] ←→ [D-E].**
Instructions arithmétiques	**ADD**	ADD R ;	✓ Add register to accumulator. **[A] ← [A] + [R].**
		ADD M ;	✓ Add memory to accumulator. **[A] ← [A] + [M].**
	ADC	ADC R ;	✓ Add register with carry to accumulator. **[A] ← [A] + [R] + [CS].**
		ADC M ;	✓ Add memory with carry to accumulator. **[A] ← [A] + [M] + [CS].**
	ADI	ADI data ;	✓ Add immediate data to accumulator. **[A] ← [A] + data.**
	ACI	ACI data ;	✓ Add with carry immediate data to accumulator. **[A] ← [A] + data + [CS].**
	DAD	DAD R ;	✓ Add register pair to H-L pair. **[H-L] ← [H-L] + [R].**
	SUB	SUB R ;	✓ Subtract register from accumulator. **[A] ← [A] – [R].**
		SUB M ;	✓ Subtract memory from accumulator. **[A] ← [A] – [M].**
	SBB	SBB R ;	✓ Subtract register from accumulator with borrow. **[A] ← [A] – [R] – [CS].**
		SBB M ;	✓ Subtract memory from accumulator with borrow. **[A] ← [A] – [M] – [CS].**
	SUI	SUI data	✓ Subtract immediate data from accumulator. **[A] ← [A] – data.**
	SBI	SBI data ;	✓ Subtract immediate data from accumulator with borrow. **[A] ← [A] – data – [CS].**
	INR	INR R ;	✓ Increment register content. **[R] ← [R] + 1.**
		INR M ;	✓ Increment memory content. **[M] ← [M] + 1.**
	DCR	DCR R ;	✓ Decrement register content. **[R] ← [R] - 1.**
		DCR M ;	✓ Decrement memory content.

			[M] ← [M] - 1.
	INX	INX R ;	✓ Increment register pair. **[R] ← [R] + 1.**
	DCX	DCX R ;	✓ Decrement register pair. **[R] ← [R] - 1.**
	DAA	DAA ;	✓ Decimal adjust accumulator.
Instructions logiques	**ANA**	ANA R;	✓ AND register with accumulator. **[A] ← [A] AND [R].**
		ANA M ;	✓ AND memory to accumulator. **[A] ← [A] AND [M].**
	ANI	ANI data ;	✓ AND immediate data with accumulator. **[A] ← [A] AND data.**
	ORA	ORA R ;	✓ OR register with accumulator. **[A] ← [A] AND [R].**
		ORA M ;	✓ OR memory with accumulator. **[A] ← [A] OR [M].**
	ORI	ORI data ;	✓ OR immediate data with accumulator. **[A] ← [A] OR data.**
	XRA	XRA R ;	✓ EXCLUSIVE –OR register with accumulator **[A] ← [A] XOR [R].**
		XRA M ;	✓ EXCLUSIVE –OR memory with accumulator **[A] ← [A] XOR [M].**
	XRI	XRI data ;	✓ EXCLUSIVE-OR immediate data with accumulator **[A] ← [A] XOR data.**
	CMA	CMA ;	✓ Complement the accumulator **[A] ← [A].**
	CMC	CMC ;	✓ Complement the carry status. **[CS] ← [CS].**
	CMP	CMP R ;	✓ Compare register with accumulator. **[A] - [R].**
		CMP M ;	✓ Compare memory with accumulator. **[A] - [M].**
	CPI	CPI data ;	✓ Compare immediate data with accumulator. **[A] –data.**
	RLC	RLC ;	✓ Rotate accumulator left. **[An+1] ← [An], [A0] ← [A7], [CS] ← [A7].**
	RRC	RRC ;	✓ Rotate accumulator right. **[A7] ← [A0], [CS] ← [A0], [An] ← [An+1].**
	RAL	RAL ;	✓ Rotate accumulator left through carry. **[An+1] ← [An], [CS] ← [A7], [A0] ← [CS].**
	RAR	RAR ;	✓ Rotate accumulator right through carry. **[An] ← [An+1], [CS] ← [A0], [A7] ← [CS]**
Instructions de branchement	**JMP**	JMP addr(label)	✓ Branchement (Saut) **non conditionné** vers une instruction spécifiée par une adresse : addr(label).
	J*	Branchement conditionné	
		JZ addr(label) ;	✓ Jump if the result is zero.
		JNZ addr(label) ;	✓ Jump if the result is not zero.
		JC addr(label) ;	✓ Jump if there is a carry.
		JNC addr(label)	✓ Jump if there is no carry.
		JP addr(label) ;	✓ Jump if the result is plus.

Système

		JM addr(label) ;	✓ Jump if the result is minus.
		JPE addr(label) ;	✓ Jump if even parity.
		JPO addr(label) ;	✓ Jump if odd parity.
	CALL	CALL addr(label) ;	✓ **Unconditional CALL**: call the subroutine identified by the operand.
	RET	RET;	✓ Return from subroutine.
	RST	RST n (Restart);	✓ Restart is a one-word CALL instruction. ✓ The content of the program counter is saved in the stack. ✓ The program jumps to the instruction starting at restart location.
Pile, E/S et contrôle de la machine	**IN**	IN port-address ;	✓ Input to accumulator from I/O port. **[A] ← [Port].**
	OUT	OUT port-address ;	✓ Output from accumulator to I/O port. **[Port] ← [A].**
	PUSH	PUSH R;	✓ Push the content of register pair to stack.
		PUSH PSW;	✓ PUSH PSW (PUSH Processor StatusWord .
	POP	POP R ;	✓ Pop the content of registerpair, whichwassaved, fromthe stack.
		POP PSW ;	✓ Pop Processor StatusWord.
	HLT	HLT ;	✓ Halt.
	XTHL	XTHL	✓ Exchange stack-top with H-L.
	EI	EI ;	✓ Enable Interrupts.
	DI	DI ;	✓ Disable Interrupts.
	SPHL	SPHL ;	✓ Move the contents of H-L pair to stack pointer.
	SIM	SIM ;	✓ Set Interrupt Masks.
	RIM	RIM ;	✓ Read Interrupt Masks.
	NOP	NOP ;	✓ No Operation.

a) Piles (Stacks)

✓ La pile est un secteur de mémoire identifié par le programmeur pour la mémoire temporaire d'information.
✓ La pile est une structure de LIFO.
✓ La pile se développe normalement dans l'arrière-plan de la mémoire.
✓ En d'autres termes, le programmeur définit le fond de la pile dans l'idée de réduire la plage d'adresses.

Dans le up8085, la pile est définie en plaçant le registre de SP (Stack Pointer ou Indicateur de Pile).

LXI SP, FFFFH ;

Ceci place l'indicateur de pile a la cellule $FFFF_{HEX}$ (fonds de mémoire pour les 8085).

✓ Deux opérations possibles sur les files d'attente :

1. **PUSH** (**EMPILER**): Mettre une information au sommet de la pile, en poussant vers le bas les informations déjà présentes dans la pile.
2. **POP** (**DEPILER**): Prend l'information qui se trouve au sommet de la pile, en poussant tout le contenu de la pile vers le sommet.

PUSH R ; /* **R = (RH-RL)** : Registre sur 16 bits*/

- **((SP)-1) ← (RH)** → Le contenu du registre supérieur de la pair R est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.

Systèmes à µP Dr.

- **((SP)-2) ← (RL)** → Le contenu du registre inferieur de la paire R est transféré à la mémoire dont l'adresse est inférieure de 2 au contenu du registre SP.
- **(SP) ← (SP)-2** → Le contenu du registre SP est décrémenté de 2.

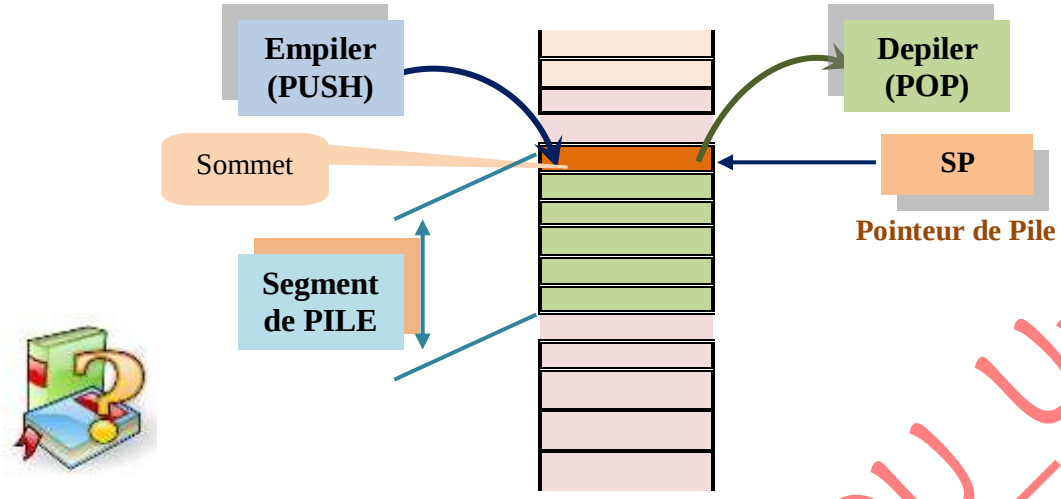


Figure 2.15 : *Représentation graphique de la Pile.*

PUSH PSW ; /* Push Processsor Status Word : Empiler le registre d'état du up8085.

- **((SP)-1) ← (A)** → Le contenu de l'accumulateur est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.
- **$((SP)-2)_0$ ← (CY)**
- **$((SP)-2)_1$ ← X**
- **$((SP)-2)_2$ ← (P)**
- **$((SP)-2)_3$ ← X**
- **$((SP)-2)_4$ ← (AC)**
- **$((SP)-2)_5$ ← X**
- **$((SP)-2)_6$ ← (Z)**
- **$((SP)-2)_7$ ← (S)**

Le contenu des indicateurs est assemblé en un PSW, et ce mot est transféré à la mémoire dont l'adresse est inférieure de 2 au contenu du registre SP.

- **(SP) ← (SP)-2** → Le contenu du registre SP est décrémenté de 2.

POP R ; /* **R = (RH-RL)** : Registre sur 16 bits*/

- **(RL) ← ((SP))** → Le contenu de la mémoire dont l'adresse est spécifié par le registre SP est transféré au registre inférieur de la pair R.
- **RH) ← (((SP)+1)** → Le contenu de la mémoire dont l'adresse est supérieur de 1 au contenu du registre supérieur de la pair R.
- **(SP) ← (SP)+2** → Le contenu du registre SP est incrémenté de 2.

POP PSW ; /* Push Processsor Status Word : Dépiler le registre d'état du up8085.

- **((SP)-1) ← (A)** → Le contenu de l'accumulateur est transféré à la mémoire dont l'adresse est inferieur de 1 au contenu du registre SP.
- **(CY) ← $((SP))_0$**
- **(P) ← $((SP))_2$**
- **(AC) ← $((SP))_4$**
- **(Z) ← $((SP))_6$**
- **(S) ← $((SP))_7$**
- **(A) ← ((SP)+1)**

Le contenu de la mémoire dont l'adresse est spécifié par le contenu du registre SP est utilisé à la récupération des indicateurs de condition.

Le contenu de la mémoire dont l'adresse est supérieure de 1 au contenu de SP est transféré au registre A

- **(SP) ← (SP)+2** → Le contenu du registre SP est incrémenté de 2.

Systèmes à µP_Dr. SOUKKOU_Univ_Jijel

3.5. Programmation du microprocesseur 8080/8085

Les programmes sources seront écrits en langage assembleur 8085. Le format utilisé par Intel divise chaque ligne en langage assembleur selon les champs suivant

Etiquette	Code Opération (COP)	Opérande (s)	Commentaires

Exemple 1 :

	LXI	H, 2020H ;	Initialiser le registre HL à 2020H
	MVI	B, 01H ;	Initialiser le registre B à 01H
	MOV	A, H ;	Transférer le contenu de registre H vers l'accumulateur
	CMA		Complémenter l'accumulateur
	ADD	B ;	A ← (A) + (B)

Exemple :

	LDA	2000H ;	// Load multiplicant to accumulator
	MOV	B, A ;	// Move multiplicant from A(acc) to B register
	LDA	2001H ;	// Load multiplier to accumulator
	MOV	C, A ;	// Move multiplier from A to C
ETIQ :	**ADD**	B ;	// Add B(multiplier) with A
	DCR	C ;	// Decrement C, it act as a counter
	JNZ	**ETIQ** ;	// Jump to L if C reaches 0
	STA	2010H ;	// Store result in to memory
	HLT		// End

3.5.1. Les soubroutines

Une soubroutine est un groupe d'instructions qui seront employées a plusieurs reprises dans différents emplacements du programme.

- ✓ Plutôt que répéter les mêmes instructions plusieurs fois, elles peuvent être groupes dans une soubroutine qu'on appellerait a partir de différents emplacements.
- ✓ Dans le langage d'assemblage, une soubroutine peut exister n' importe où dans le code.
- ✓ Cependant, il est usuel de placer des soubroutines séparément du programme principal.
- ✓ Le modèle 8085 a deux instructions pour traiter avec les soubroutines.
- ✓ L'instruction **CALL** est utilisée comme moyen de réorienter l'exécution du programme a la soubroutine.
- ✓ L'instruction **RET** est employée pour renvoyer l'exécution au programme d'appel.

Systèmes à microprocesseur

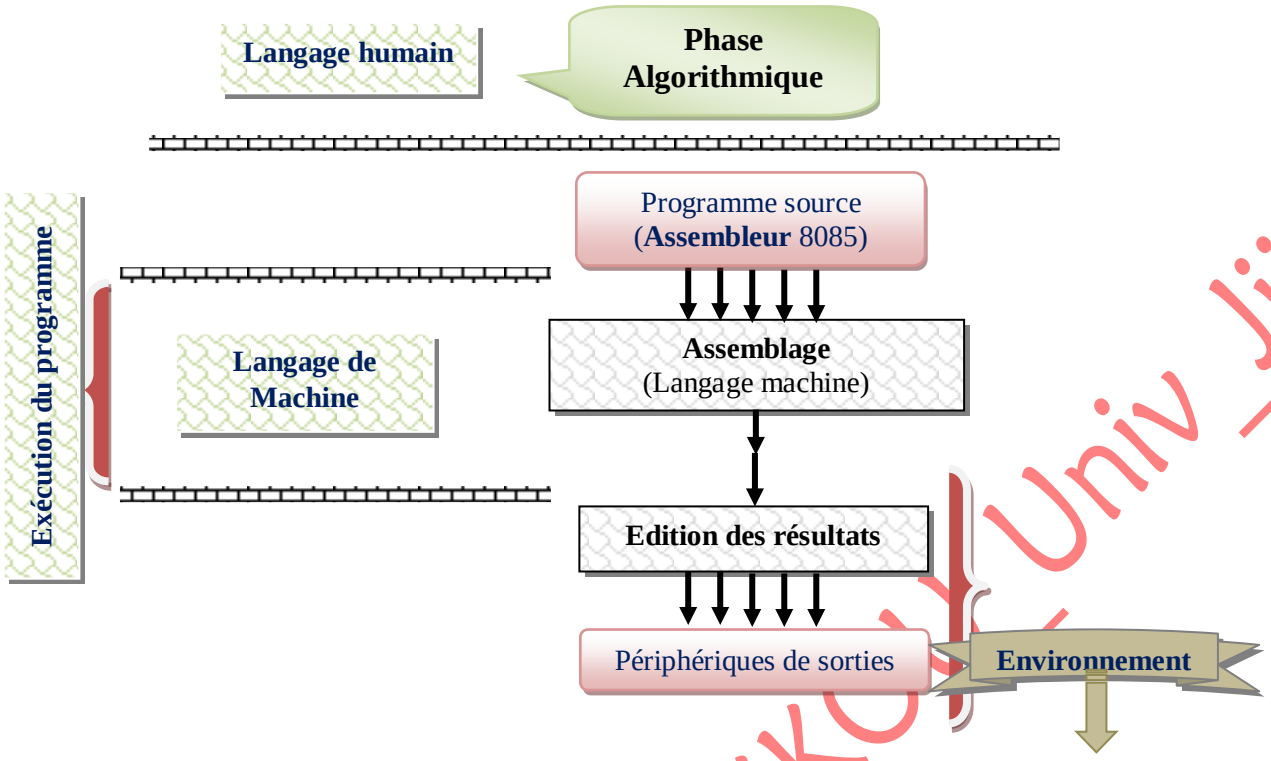


Figure 3.16 : *Programmation assembleur 8085.*

Systèmes à microprocesseurs - Dr. SOUKKOU - Univ_Jijel