REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE AMAR TELIDJI-LAGHOUAT

FACULTE DE TECHNOLOGIE

DEPARTEMENT D'ELECTROTECHNIQUE

Polycopié de cours

MICROPROCESSEURS ET MICROCONTRÔLEURS

Master 1ére année :

Energies renouvelables pour l'électrotechnique

Dr. Naceur SELMANE

Maitre de conférences ''A ''

Année 2021 /2022

SOMMAIRE

I.1 Généralités

Les ordinateurs, nés vus les cennes 50, étaient des machines très volumineuses, réservés aux grandes entreprises.

Aux crus des années 70, la taille des ordinateurs est considérablement modifiée, grâce à un élément nouveau: le micro-processeur-définitions

I.1.1 Définitions :

- Un micro-processeur (µp) est un circuit intégré programmable, capable de traiter automatiquement une pièce de taches qui lui est demandé.
  Il réalise, en un seul boitier, les fonctions habituelles d'une unité centrale d'un ordinateur (unité arithmétique et logique et unité de commande).
- Un circuit intégré (CI) est un composant électronique minuscule contenant des milieu de transistors interconnectés entre eux, et réalisé sur une pastille de Silicium (SI) de quelques min, montée sur un boitier muni de Bruches.

Ces transistors forment les mémoires, les registres, les compteurs, les additionneurs, les codeurs, décodeurs, les multiplexeurs …

- Micro : très petit, suivant la densité d'intégration des TR :

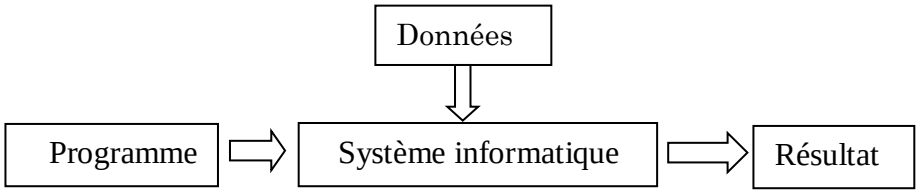
  - Circuits MSI (Medium scale intégration) ex pentium

  - Circuits LSI (Large scale intégration) 3,1 milliares de TR

  - Circuits VLSI (very large scale intégration)

- Processeur : traitaient, séquençaient d'une suite d'instructures.
- Programmable : commandé par un programme (un câble).
- Micro-ordinateur : regroupe un µp, des mémoires et des circuits d'interfaces (Entrées/ Sorties, Input/ Output) chargés d'établir les communications avec le monde extérieur (clavier, imprimante…)
- Unité centrale de traitement (UCT) ou CPU (central processing unit) : de l'ordinateur est chargée, sous la direction d'un programme, d'exécuter ses instructrices : c'est le µp.
- périphériques : permettent d'introduire (entrée) ou d'extraire (sortie) des informations vers ou de l'ordinateur : clavier, écran, imprimante, souris, disque, modem, fax…)
- Les interfaces : E/S ou I/O :permettent au µp decommuniquer avec les périphériques.

- Un système informatique : regroupe l'ordinateur, les périphériques et les logiciels.

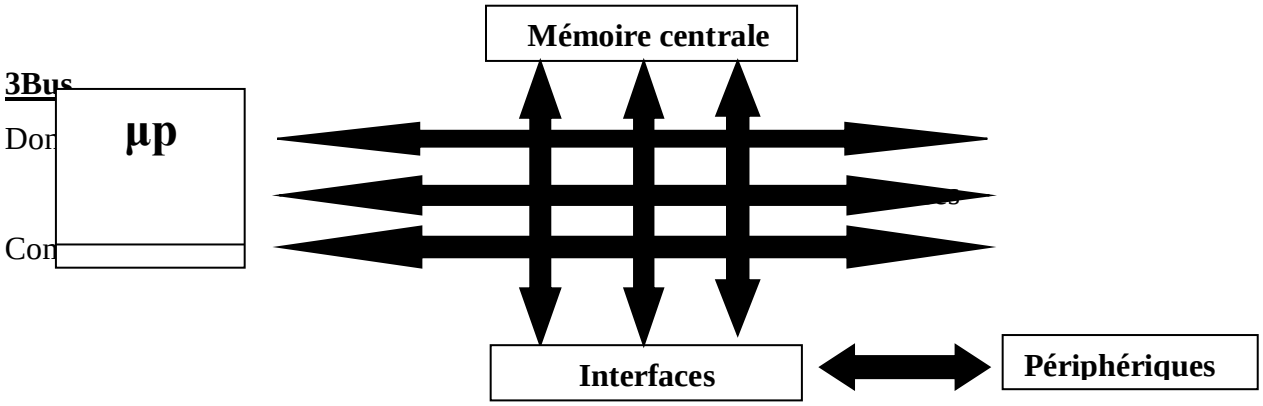


- Sous la direction d'un programme, il traite des données et fournit des résultats.
- Dans un contexte industriel, il prend des décésures.

- Logiciels (Software) : désigne les programmes, c'est le matière grise de l'ordinateur.
- Matériel (Hardware) : désigne tout ce qui est physique, circuits de l'ordinateur, les périphériques, les câbles, les documents…
- Un programme : est une suite d'instructions stockée dans la mémoire.
- Une instructeur :tache, travail à exécuter, opération à faire (addition, déplacement, entrée, sortie…)
- Les programmes manipulent des informations appelés données.

I.1.2Le µp et son environnement :

Un µp tout seul ne peut rien faire. Pour qu'il fonctionne, il doit être assurée à des circuits qui lui sort indispensable : les mémoires, les Interfaces (E/S) et les périphériques (clavier, écran…)



Ces circuits sont reliés entre eux par des connexions électriques appelées Bus.

- Bus de données qui transmettent les données et les instructions.
- Bus d'adresses : fournissent l'adresse des cellules ménines ou les E/S.
- Bus de commande : transmettent les ordres dans tous les sens demande pas les instructeurs (lecture, écriture, etc..)

* pour que le µp puisse exécuter un programme, celui-ci doit se trouver dans la mémoire centrale

* la mémoire ressemble à un meuble de rangement (commande) à tiroirs superposés et numérotés par ordre croissant (adresse) chaque tiroir s'appelle cellule mémoire.

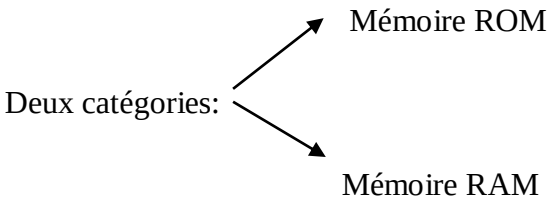
- Chaque cellule mémoire contient une information binaire compose de 8 hits (octet) codée eu hexadécimal et a adresse unique.

8A	0007
65	0008
F3	0009
C7	000A
	000B
	……

1	0	0	0	1	0	1	0

Une cellule mémoire est registre à entrée parallèle et sortie parallèle

- La mémoire centrale : est réalisée sur des circuits intégrés et se divisent en



- Les programmes de system sont rangés d'une façon définiteur dans les mémoires mortes "ROM".
- Les données sont rangés dans les mémoires que les programmes utilisateurs d'une façon transitoire : RAM.
- Les ROM sont des circuits intégrées programmés d'une manière permanant par les constructeurs

I.1.3. Fonctionnement de principe d'un µp :

* le µp est le centre de toutes les activités : c'est lui qui effectué les échanges d'informations entre les différents circuits, c'est lui qui effectué les calculs et c'est lui qui synchrone le fonctionnement de l'ensemble dans lequel il est intégré.

* Sous l'action d'un programme et de commande externe, le µp dont accompli les taches suivantes :

- Appeler une instructrice qu'il lit en mémoire

- La décoder et traduire en commandes internes ce qu'elle lui dit de faire.
- L'exécuter.
- Une fois le programme terminé, les résultats sont transférées soit vers le mémoire, soit vers la sortie (écran, imprimante), soit commander une machine ou processeur industriel.

I.1.4. Caractéristiques d'un µp :

- Un µp ne peut traiter que des informations binaires (0.1), présenté par une combinaison de 4 bits, 8 bits (octets et bite), 16 bits 2 octets, 32 bits ou 64 bits.
- Une des caractéristiques essentielles d'un µp est le nombre d'éléments binaires qu'il peut traiter en même temps (enparallèle).
- Il existe différents µp :
  - µp 4 bits : TMS 1000 (Texas instrument)
  - µp 8 bits : MC 6800 de Motorola, 780 Zilog, 8085 de Intel.
  - µp 16 bits : MC 6800 de Motorola, 8085 et 8086 de Intel.
  - µp 32 bits : 486 de pentium de Intel.
- Un µp 8 bits, transfert, traite et range toutes les données par groupe de 8 bits (1 octet) qui sont véhiculé par 8 conducteurs en parallèle (Bus données) : c'est le but de notre programme
- Un µp à 8 bits peut utiliser un maximum de 2^8: 256 instruction différentes.
- Les concepteurs décident quel mot de 8 bits, produit quel action des le µp (c'est l'instruction)

  Ex : l'instruction $\frac{\text{INPUT}}{\text{IN}}$ =DB =10011011

  Langage assembleur langage machine
- Chaque µp a ses propres instructions données par l'instructeur.

I.1.5. performance du µp :

- La fréquence de travail (si f ↗, t ↘)
- La vitesse d'exécutrice des instructeurs.
- Le nombre d'instructeurs exécutable en un seconde se mesure par milliaires :

  Mips (milliaire d'instructions par seconde)

  8085:0.75 Mips.

  802886: 1.5 Mips.

  80486: 270 Mips.

  Pentium > 100 Mips.

I.1.6. Les avantages des µp :

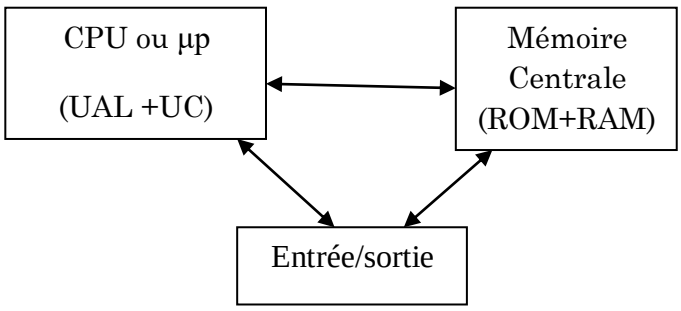
- réduction du nombre d'élément (très petit taille).
- Prix de revient réduit.
- programmabilité.

I.2 Architecture d'un microprocesseur

I.2.1- Introduction :

La structure générale d'un système à µp minimum comprend :

- Le µp, proprement dit (CPU)
- une mémoire ROM qui contient les programmes de l'application
- Une mémoire RAM qui contient les données et programmes de l'utilisateur.
- Les dispositifs d'entrée/sortie (Interfaces)
- Trois liaisons de Bus assurent la communication entre les différents sous ensembles (Bus données, adresse et commandes)



Le µp fait traiter sous le contrôle de l'UC par l'UAL des informations issue de la mémoire et cordonne les échange internes et avec le milieu extérieur (clavier, imprimante).

I.2.2 Structure interne d'un µp :

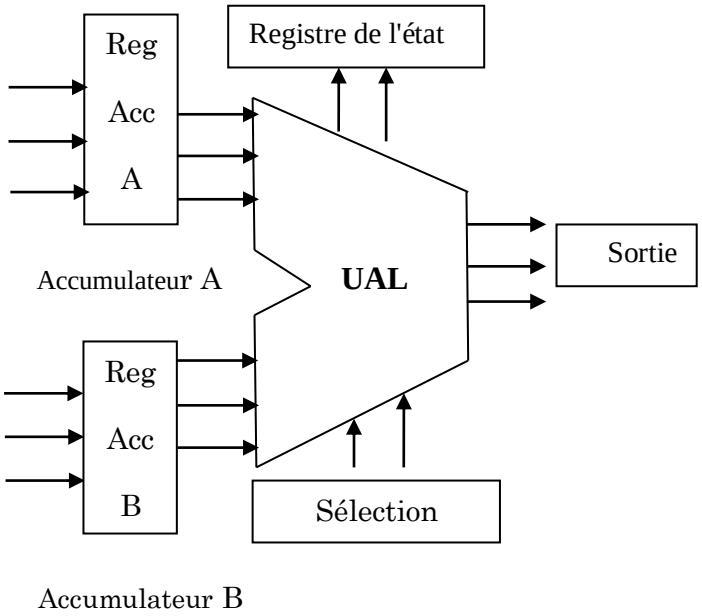
Tout µp comprend les éléments suivants :

- Une unité de commande (UC) qui organise le séquencèrent interne du µp et gère les bus de commande pour l'exécution des instructeurs.
- Une unité de calcul ou unité arithmétique et logique (UAL).
- Un registre d'instruction avec un décodeur d'instruction chargé d'interprétée et de déterminer ainsi les commandes à appliquer à l'UAL.
- D'autres registres : compteur de programme, pointeur de pile…
- Une horloge qui piloté le déroulement séquentiel des instructeurs.
- Des Bus interne de données, adresse et commande.

I.2.3 Structure fonctionnelle du µp

I.2.3.a L'unité arithmétique et logique (UAL) :

L'UAL est l'élément essentiel du µp. Elle regroupe un ensemble de circuit logique combinatoire capable de réaliser des opérations arithmétique et logique (addition, soustraction, ×, ÷, et, ou…).



L'UAL réalise des opérations direct les ordres sont données par les instructions des programmes.

Elle comprend des circuits additionneurs, comparateurs, complimenteur.

Exemple :

Le séquencèrent de l'UAL pour addition deux nombres est :

1. Charger le registre accumulateur A avec 1er opérande issue de la mémoire ou des E/S.
2. Charger le registre B avec le 2eme opérande.
3. Additionner N_1 et N_2.
4. Transférer le résultat de l'opération de l'accumulation A.
5. Transférer le contenu de l'accumulation a en ménine ou en sortie.

I.2.3.a.1 le registre accumulateur A :

L'accumulateur A est le registre de travail (spécial) de l'UAL c'est un registre (8bits) destiné à recevoir le 1eropérande de donnée avant l'opération et li stocke le résultat après l'exécuteur de l'opération.

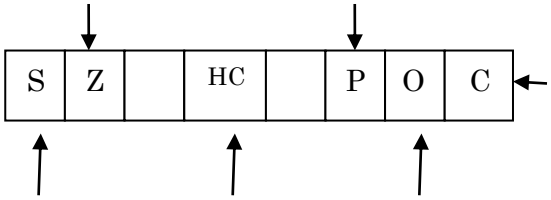
I.2.3.a.2 Le registre d'état (Flag registre) : RE

Il est nécessaire de disposer de certaines informations ou le résultat de l'opération. Pour cela ou dispose un registre d'état (8 bits) contenant un ensemble d'indications sur le résultat de l'opération réalisé par l'UAL. Ces indicateurs (flag) sont :

1. Indicateur de zéro : bit Z détecte que le résultat de l'opération est nul (Z= 1)
2. Indicateur de signe : bits détecte le si que de l'opération, S=1 si négatif.
3. Indicateur de retenue : bit C (carry) si le résultat est trop grand pour l'accumulation. La retenue est stockée dans C.
4. Indicateur de dépassement : bit 0 (over flow) : détecte le dépasser de capacité des opérations par complément à 2 (valu interdite).
5. Indicateur de demi-retenue : bit H (Half-carry) : détecte le transfert d'un retenue inter médiaire quand en travaille en BCD.
6. Indicateur de parité : bit P (party) : compte le nombre de 1 de U mot binaire et sert à détectée les erreurs de transmission.

Zéro Parité

Registre d'état du 8085



Signe Demi-retenue dépassement

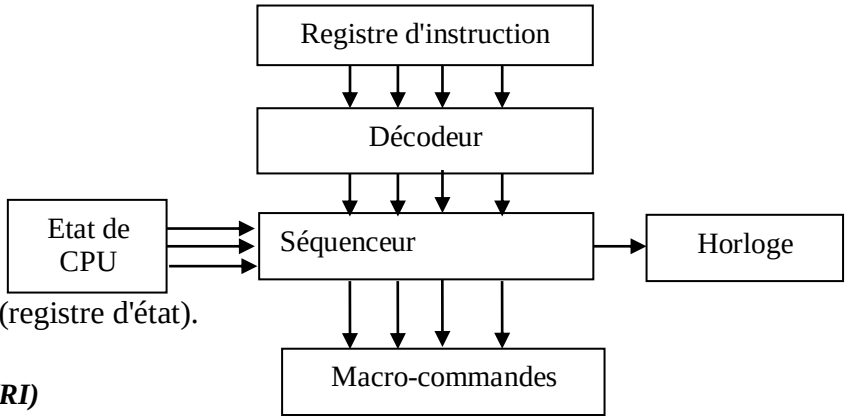
I.2.3.b L'unité de commande : UC

L'unité de commande organisée le séquencèrent interne du µp. Elle produit des macro-commandes dans l'ordre au rythme d'une horloge.

Aux circuits internes du µp pour recherche de données et d'instructeurs dans le ménure, de décodage et de l'exécution de ces instructions.

L'unité de commande se compose du registre d'instruction, du décodeur d'instruction et d'un séquenceur synchrone.-

Le séquenceur génère des macro-instructions nécessaire de décodeur des informations à partir de l'instruction et de l'état du CPU (registre d'état).

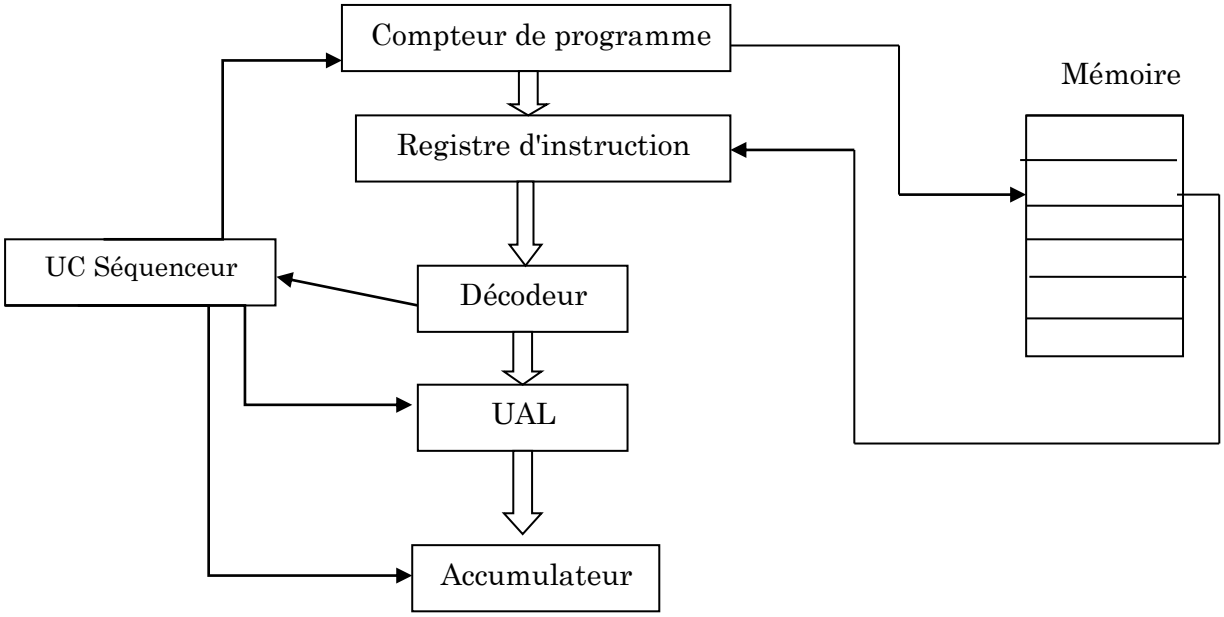


I.2.3.b.1. le registre d'instruction (RI)

C'est un registre (3×8bits) qui reçoit l'instruction à exécuter, extraite de lamémoiregrâce à l'adresse fournie par compteur de programme.

I.2.3.b.2 Le décodeur d'instruction (DI) :

Le décodeur reçoit du registre d'instruction le code de l'opération à exécutée, l'analyse, la décode (l'interprète) et transmit à l'UC le type d'opération (ordre) à exécuter.



I.2.3.c Les registres internes du µp :

I.2.3.c.1 le registre compteur de programme (CP)

Appelé aussi compteur ordinal (CO)

C'est un registre 16 bits qui contient toujours l'adresse de données ou de l'instruction suivante qui être lue dans le programme.

Il est incrémenté automatiquement de 1 chaque fois qu'une instruction est ramenée de la mémoire. Le contenu de ce registre sera déposé sur le bus d'adresse pour extraire de la mémoire, l'instruction suivante à exécuter.

Le compteur de programme contrôle l'ordre dans lequel le programme est lu et exécuté.

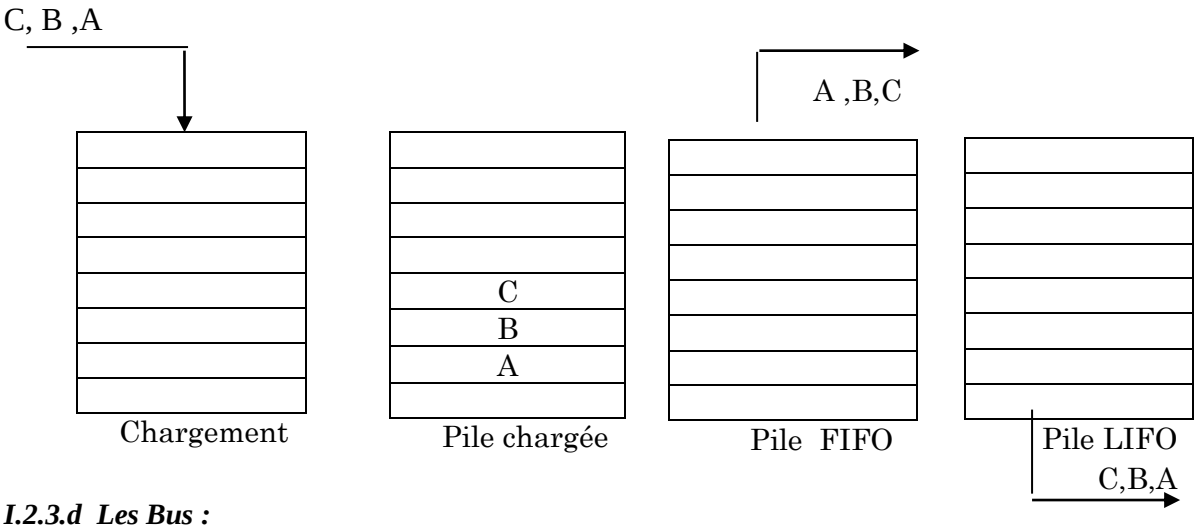
I.2.3.c.2 Le registre pointeur de pile (stock pointer) : P.P (SP) :

C'est un registre (16 bits) qui contient toujours l'adresse du sommet et de la pile. La pile est un espace mémoire, réservé dans la mémoire centrale (RAM) pour la sauvegarde du contexte du µp (sauvegarde du contenu de tous les registres) hors d'une interruption ou d'un branchement à une sous-broutine.

Hors de la suspension ou interruption de l'exécution d'une instruction ou d'un programme encours (ex : demande d'impression…), le contenu des registres internes du µp sont stockés dans des emplacements mémoires appelés pile. Après l'interruption, il y aura restitution

ducontenu des registres c'est à dire.transfert des données de la mémoire pile vers les registres internes.

Une pile est une mémoiretemporaire appelé FIFO (First in, First out) ou LIFO (Last in, Last out).



I.2.3.d Les Bus :

Trois liaisons de Bus assurent la communication entre les différents circuits:

I.2.3.d.1 Les Bus d'adresse

Les Bus d'adresse (16 bits), unidirectionnels, transmet les ordres des instructions (vers la ROM), les adresses des opérandes (vers la ROM) ou des dispositifs d'entrée/sortie., il détermine la capacité maximale d'adressage du système, c'est à dire le nombre maximum de mots de la mémoire associée (ex : 16 bits "adressent" 64 Kmots).

I.2.3.d.2 Les Bus de données

Les Bus de données (8 bits), bidirectionnels, permettent la lecture des instructions dans la ROM ou le transfert de données (lecture ou écriture) vers ou de la RAM ou les E/S.

I.2.3.d.3 Les Bus de commande

Les Bus de commande permettent la transmission des signaux des macro-commandes(lecture, écriture…) Ce bus sert à coordonner tous les échanges d'informations décrits précédemment. Il véhicule des données qui valident la mémoire et les ports d'entrées / sorties. Il introduit des délais d'attente lorsque des informations sont envoyées à un périphérique qui présente une vitesse de traitement réduite. Le bus de commandes évite les conflits de bus lorsque deux éléments cherchent à communiquer en même temps.

Remarque :

Dans certains cas, le bus de données et le bus d'adresses sont multiplexés sur un seul bus. Une logique externe doit alors effectuer le démultiplexage.

I.2.3.e l'horloge :

L'horloge est la base de temps nécessaire au séquencèrent des macro-commandes, si

$T = \mu s \Longrightarrow f = 1MHZ$

Le µp fait traiter, automatiquement, au rythme d'une horloge, une suite d'instructions (programme) rangés en mémoire.

II.1 Classification et format des instructions

II.1.1. structure d'une instruction :

- Une instruction représente ce que doit faire un µp sur une donnée appelée **<u>opérande</u>**.

  Ex : diviser par 2, ajouter 5

- L'instruction d'une µp est divisée en deux parties appelée : champs
  - Le 1er champ précis le **<u>code opération</u>**, représentant l'opération à réaliser (diviser, ajoutée, complémentée, décalée…)
  - Le 2éme champ donne **<u>le code opérande</u>**, représentant une donnée ou une adresse, ce sur qui doit porter l'opération.

Code opération : CO	Code opérande : OP

: $\frac{\text{ajouter}}{\text{C.O}}\ \frac{5}{\text{O.P}}$

- Les instructions ont des formats variables, elles peuvent être codées sur un, deux ou trois octets (bytes) (8bits).
- Les instructions à 1 octets unique, elles sont composées du code opération uniquement, l'opérande n'existe pas, elles sont exécutées directement après la lecture et décodage.

CO

Ex : 2F, INC : 3C (complémenter, Incrémenter)

- les instructions à 2 octets renferment l'octet du CO et un octet pour l'opérande qui peut être une donnée, l'adresse d'1 porte.

Ex

CO	OP

ADD 05: $\frac{CE}{CO}\ \frac{05}{OP}$

(Lire à l'entrée d'adresse 01) (Ajouter 5)

- les instructions à 3 octets renferment l'octet du CO, les 2 octets pour l'opérande qui est une adresse (16 bits= 2×8 bits)

E

CO	OP1	OP2

cker à l'adresse 0020)

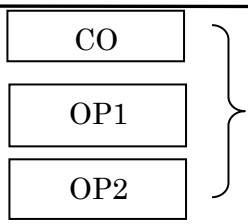
CO

} 1ere instr

CO
OP

} 2eme instr

-Le µp reconnait lui-même la longueur de l'instruction lors du décodage de celle-ci.



- Les opérandes éventuelsdoivent toujours se trouver derrière l'octet du CO dans la mémoire.

II.1.2 Langage de programmation:

Dans la mémoire, le programme est écrit en binaire. Pour l'homme il est impossible d'écurie un programme en binaire. Ainsi, les instructeurs sont codés dans un code de programmation un peu accessible à l'homme qui est le code hexadécimal ou "langage machine" chaque octet est représenté par 2 chiffres hexadécimaux.

Ex: IN: $(1101\ 1011)_2 = (DB)_{16}$

ADD: $(1100\ 0110)_2 = (C6)_{16}$

Remarque: chaque µp possède ses propres instructions codé donnés par le fabricant le langage hexadécimal est utilisé dans le kit à µp.

Dans les systèmes plus sophistiqué, on utilise un langage plus accessible à l'homme qui est le langage **ASSEMBLEUR** c'est un ensemble de lettres appelées **MNEMONIQUE** appelant l'opération à réaliser : Ex: LDA : changer l'accumulateur.

STA: stockée dans l'A

JMP: sauter.

ADD: ajouter.

Il existe aussi des langages très enlier tels que: BASIC, FORTRAN, PASCAL, C, CTT…

II.1.3. Les principales instructions du µp

Il est impossibles de se rappeler tous les codes machines des instructions ainsi pour lire ou écrire un programme, un utilisé les code assembleur qui est facile à retenir.

On distingue plusieurs catégories d'instruction:

- les instructions d'entrée (sortie)
- les instructions de transfert de données.
- Les instructions de manipulation d'information (opération arithmétique et logique)
- Les instructions de contrôle de programme.

a) **Les instructions d'entrée/sortie:**

MNEMONIQUE	HEXADECIMAL	BINAIRE	NOMBRE D'OCTETS
IN	DB	11011011	2 (DB 45)
OUT	D3	11010011	2 (D313)

DB 45 (IN45): les données présentées à la porte d'entrée d'adresse 45

Sont chargées dans l'accumulateur

D3 13 (out 13) : les contenu de l'accumulateur est transféré à la porte de sortie d'adresse 13.

b) **Les instructions de contrôle de programme**

MNEMONIQUE	HEXADECIMAL	BINAIRE	NOMBRE D'OCTETS	
JMP	C3	11000011	3	Ex: JMP 0075
JZ	CA	11001010	3	Ex: JZ 1F20
JNZ	C2	11000010	3	Ex: JNZ 00E1
HLT	76	01110110	1	

JMP 0075 : sauter à l'adresse 0075 (Saut inconditionnel)

JZ : sauter à l'adresse 1F20 si le résultat de l'opération est nul

JNZ :sauter à l'adresse 00E 1 Si le résultat de l'opération est non nul

HLT :arrêt du µp

c) **Les instructions de transfert de données:**

MVI	3E	2OCTETS	Ex: MVIA 10 ou (3E10)
STA	32	3OCTETS	STA8000 (328000)
LDA	3A	3OCTETS	LDA0103 (3A0103)

MVIA 10: l'accumulateur est chargé immédiatement avec la valeur 10.

STA 8000: le contenu de l'accumulateur est stocké à l'adresse 8000.

LDA 0103: charge le contenu de la cellule mémoire d'adresse 0103 dans l'accumulateur

d) **Les instructions de manipulation d'informations**

- **Les instructions logiques:**

ORI	F6	2 octets	Ex: ORI77 (F677)
ANI	E6	2 octet	ANI55 (E655)
CMP	2F	1 octet	CMPA

ORI77: le contenu de l'accumulateur est combiné par OU logique avec 77

01110111 :77 01010101:55

Ou $\frac{10110110}{11110111}$: contenu de l'ACC. Ex: $\frac{11111111}{01010101}$

ANI 55 : opération de ET logique entre le contenu de l'ACC et la valeur 55.

CMPA: complémenter le contenu de l'accumulateur.

- **Les instructions arithmétiques:**

ADI	C6	2 octets	ADI 18 (C6 18)
SUI	D6	2 octets	SUI (D6 34)
INR	3C	1 octet	INR: incrémenter l'accumulateur
DCR	3D	1 octet	DCR: décrémenter l'accumulateur

ADI 18: la valeur 18 est additionnée immédiatement à l'accumulateur.

SUI 34: la valeur 34 est soustraite immédiatement à l'accumulateur.

II.1.4. Exemple de programme:

Quand un écrit un programme, on liste les octets instruction et on écrit les adresses à coté

Ceci facilite la lecture du programme.

Adresse	Instruction
0000	IN 01
0002	CMP
0003	OUT 02
0005	JMP 0000
0008	.....
	.....

Programme assembleur

adresse	instruction
0000	DB 01
0002	2F
0003	D3 02
0005	C3 0000
0008	.....

Programme machine

adresse	instruction
0000	DB
0001	01
0002	2F
0003	D3
0004	02
0005	C3
0006	00
0007	00
0008	....

Programme dans la mémoire

Les programmes sont en générale développés en langage assembleur, ensuite translaté en langage machine.

En entrant le programme dans ménine, le µp l'exécute aux

-Recherche des données qui sont dans la porte d'entrée d'adresse 01 et les met dans l'accumulateur.

- Complémente le contenu de l'accumulateur.
- Envoie les données de l'accumulateur vers la porte de sortie d'adresse 02.
- Contenue l'exécution des instructions en sautant à l'adresse 0000.

Remarques:

- chaque cellule ne peut contenu qu'un octet, d'où le dernier tableau qui montre comment le programme est rentré (écrit) dans la mémoire RAM.
- Pour distingue une instruction d'une donnée parmi les informations de la mémoire la 1^{ere} information est toujours une instruction (Code Opération CO) et les informations suivantes sont des données ou des adresses (code opérande OP).
- Toutes les instructions du microprocesseur 8085 seront dérivées de manière détaillée après dans cette polycopie

II.2 Fonctionnement du µp

Un système à µp est capable de stocker des informations, de réaliser des calculs, de prendre des décisions suivant le résultat de calcul, et d'arriver à la solution finale d'un problème donné. Cependant, il ne peut accompli ces différentes taches pour être dirigé. Chaque détaille doit lui être par le programmeur.

Ce dernier, écrit un programme qui est traduction d'un organigramme qui représente les différents traitements à exécuter.

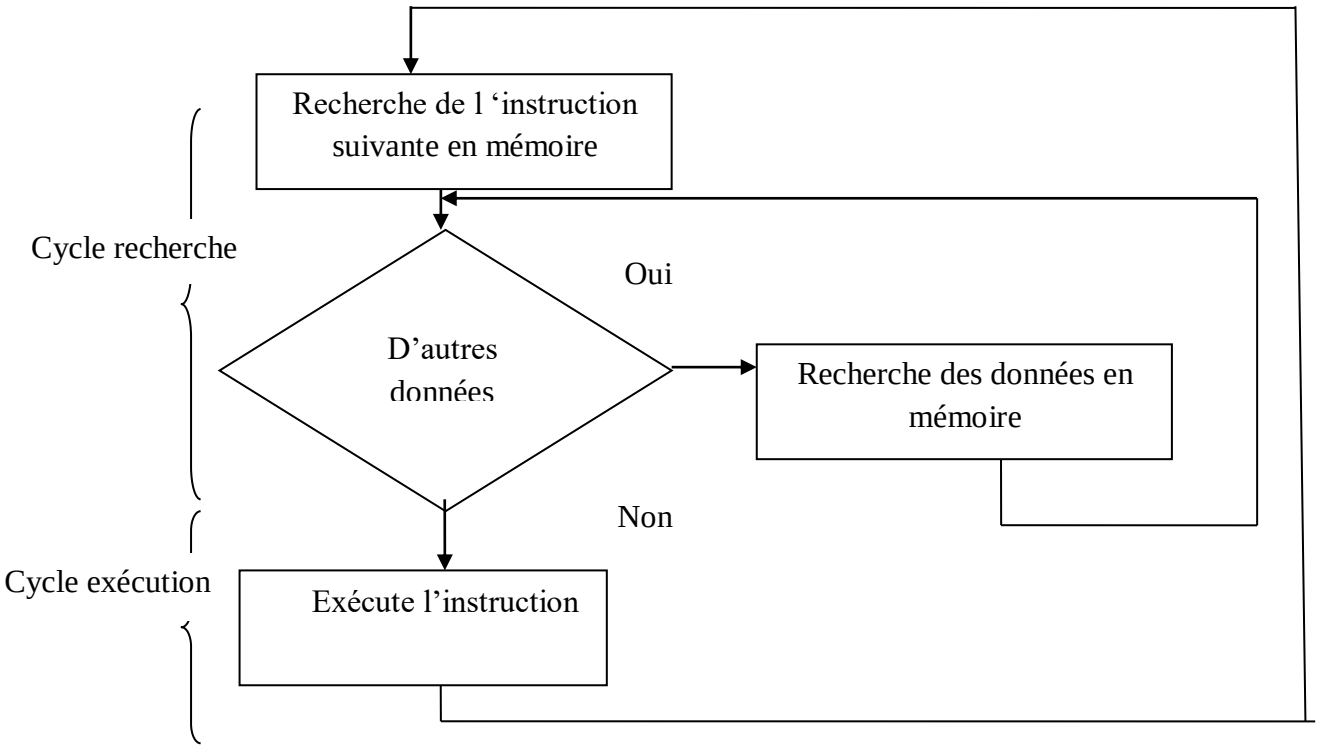
Le programme, composé d'une suite d'instructions, étant rangé mémoire, le µp doit accompli les taches suivantes:

1. appeler une instruction qu'il lit en ménine.
2. Décoder cette instruction c.à.d. la traduire en macro-commande internes, ce qu'elle lui dit de faire.
3. Il l'exécute.

II.2.1Séquencement des instructions

Le séquencèrent d'une instruction comprend deux phases

- Une phase de recherche en en mémoire et du décodage de l'instruction
- Une phase de l'exécution de m'instruction



II.2.1.a - La phase recherche

1. L'adresse de l'instruction à exécuter est stockée dans le CP.

2. Le contenu du CP est transféré sur le Bus d'adresse qui relié à la mémoire.
3. Une commande de lecture/écriture (L/C) est présentée à la ménine par le Bus.
4. Transfert du contenu de la cellule ménine adressée sur le Bus de donnée.
5. Cette information est ensuite aiguillé vers le registre d'instruction (RI) ou s'effectue le décodage et la reconnaissance du code opératoire.

La phase recherche se termine par l'incrémentation du CP qui printer vers la prochaine cellule du programme.

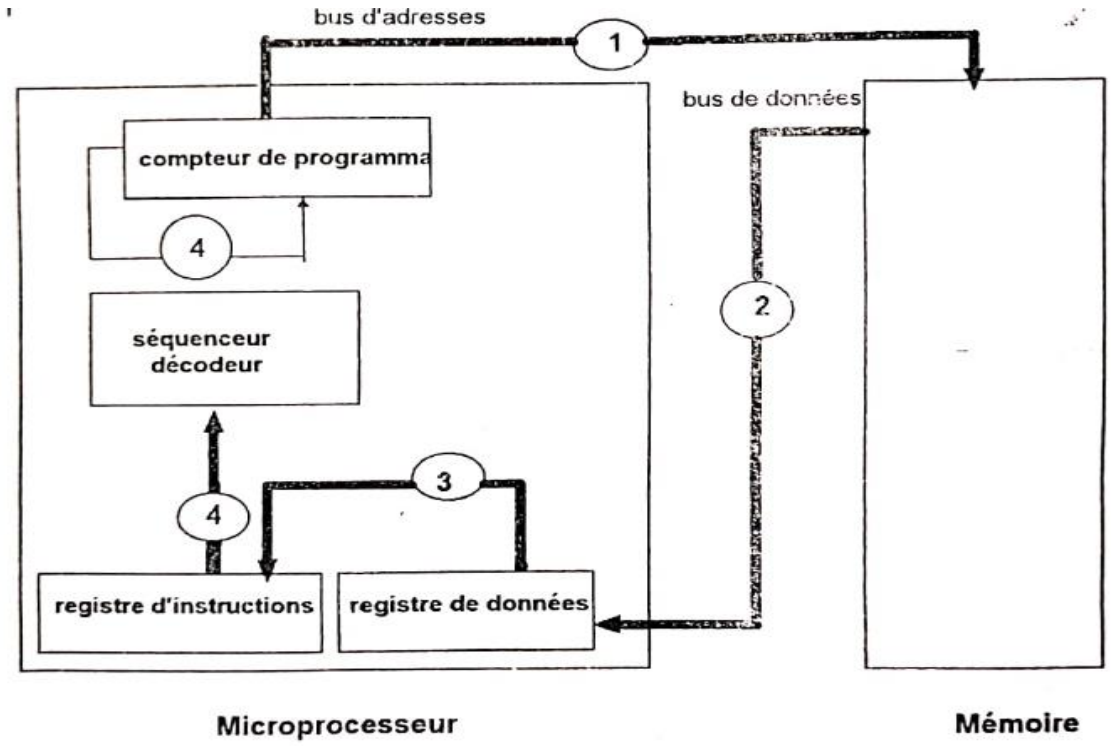


Schéma logique des liaisons : phase de recherche

- Transfert vers la mémoire de l'adresse de l'instruction
- Transfert de l'instruction dans le registre de données
- Transfert de l'instruction dans le registre d'instruction
- Incrémentation de contenu du compteur de programme et décodage de l'instruction

II.2.1.b La phase exécution:

L'instruction placée dans le RI étant analysée et décodée, l'UC décide quels sont être les différents circuits concernés par l'exécution de l'instruction (UAL, Accumulateur,…).

Ex: chargement de l'accumulateur par le contenu d'une cellule mémoire.

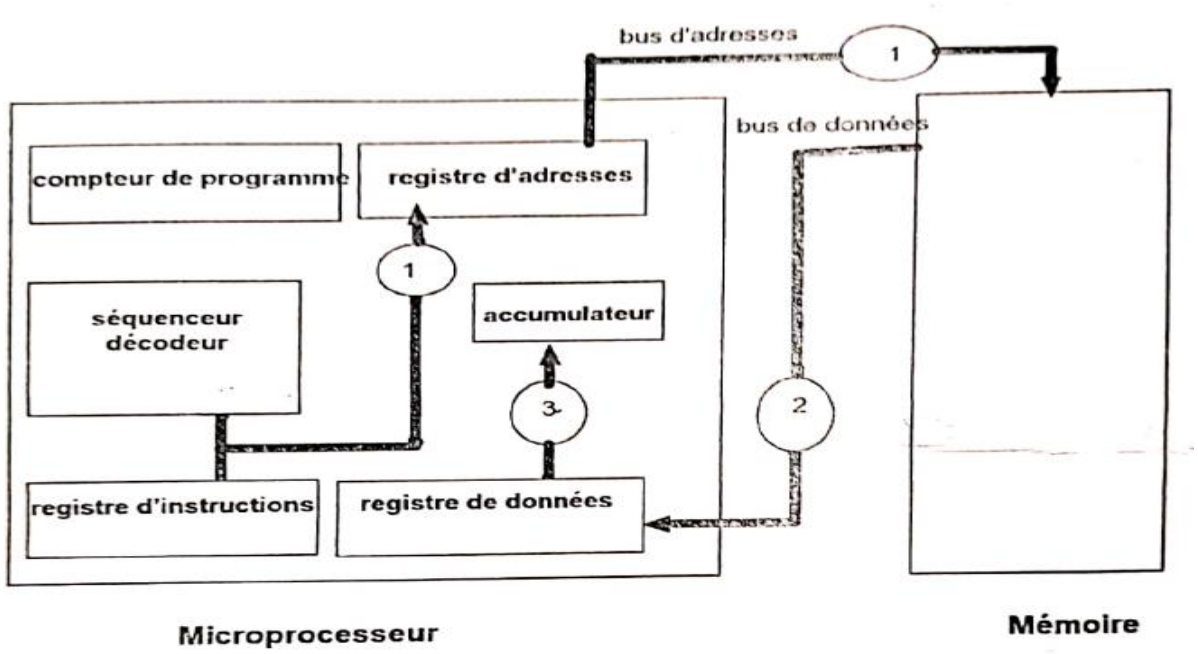
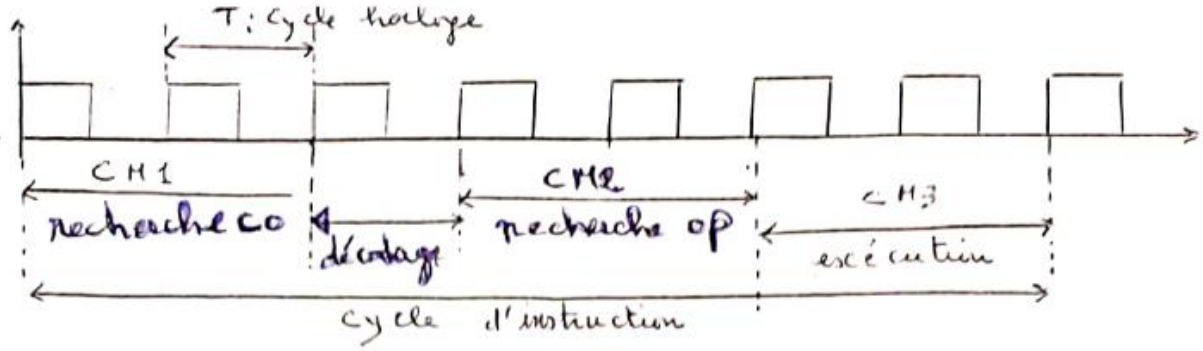


Schéma logique des liaisons : phase exécution

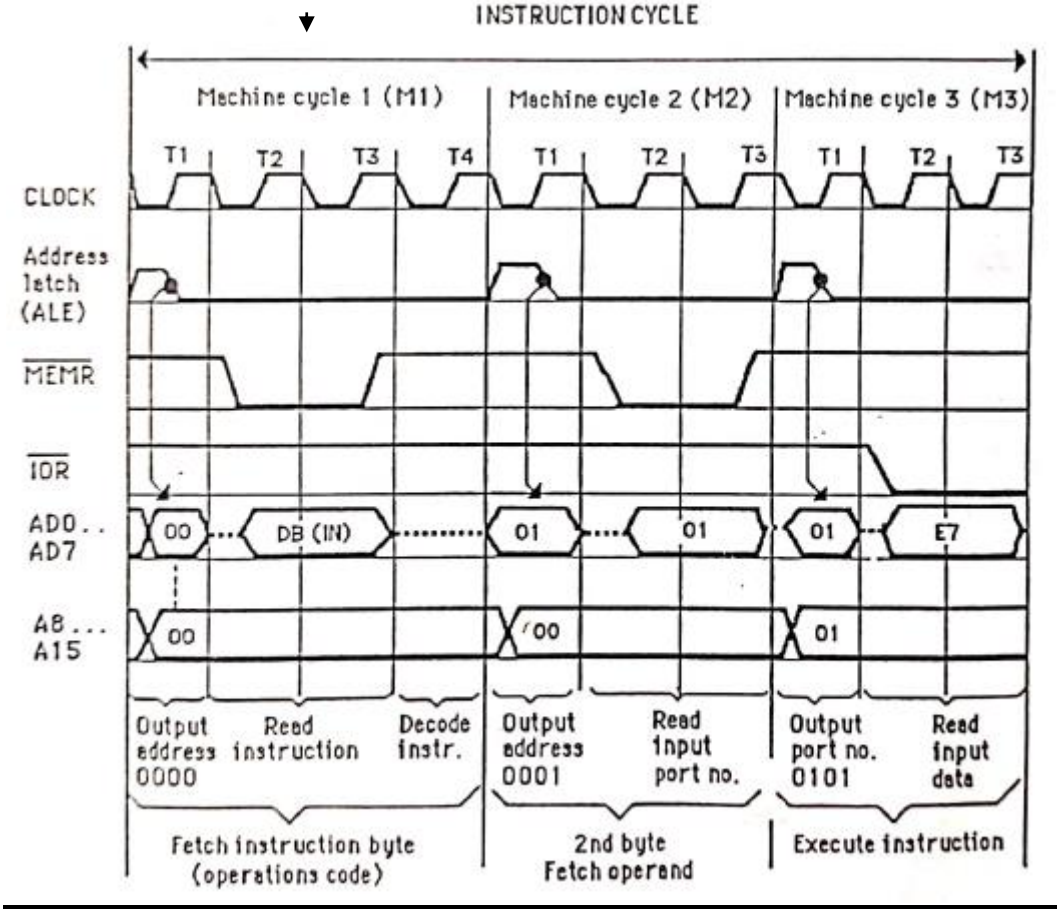
- Transfert vers la mémoire des champs adresse de l'instruction
- Transfert de mot de mémoire dans le registre de données
- Aiguillage de donnée vers le registre accumulateur

Sachant que le fonctionnement du µp est cadencé par une horloge, la réalisation d'une instruction nécessite plusieurs cycles (périodes) d'horloge appelée**Cycle instruction**. Chaque cycled'instruction est composé de plusieurs cycles machines (CM). Un cycle machine est composé de 1 à 5 cycle d'horloge.



Le temps d'exécution d'une instruction est donné par le constricteur plus la fréquence de l'horloge ↗ , plus le temps ↘

II.2.2. Diagramme temporel d'une instruction : IN01 (DB01):



II.2.3Description du diagramme temporel:

Cette instruction IN01 ou (DB01), signifie lire les données à la porte d'entrée d'adresse 01, comporte 3 cycles machines.

- **Cycle machine CM1: recherche et décodage du code opération:**

T_1: transmission de l'adresse de la case mémoire (0000). Contenu dans le CP, sur Bus d'adresse et validation de as adresse par la signale (ALE) sur le Bus de commande (BC).

T_2: l'ordre de lecture $\overline{MERE}$=0 est transmis le BC à la case ménine adressée (0000), ceci lui permet de mettre les données sur Bus de données. Le code (AB) apparait sur le BD.

T_3: le code (DB) est chargé dans le RI et le signal $\overline{MERE}$= 1, on désactive la case ménine choisie. Le CP est incrémenté →0001.

T_4 : le µp décode l'instruction qu'il vient de lire, c.à.d regarde dans la ROM la liste des instructions stockées et à partie de la détermine ce qu'est l'instruction.

L'instruction reçue et décodée dans ce cas est une instruction à 2 octets (DB01), ainsi il y a un autre octet à lire dans la ménine avant que l'instruction puisse être exécutée.

L'adresse du 2 octet (0001) est déjà dans le CP.

- **Cycle Machine CM2: recherche de l'opérande:**

T_1 : l'adresse de l'opérande (0001) est mise sur Bus d'adresse et validation de cette adresse par le signale ALE sur le BC.

T_2: l'ordre de lecture est donnée $\overline{MERE}$= 0, le µp lit le contenu de la cellule ménine adressé, qui contient l'adresse de la porte d'entrée, et le met sur le Bus de données.

T_3 : le µp minorisé intérieurement l'adresse de la porte d'entrée (01) dans un registre temporaire.

- **Cycle machine CM3 : exécution de l'instruction:**

T_1 : le µp met l'adresse de la porte d'entrée sur le Bus d'adresse (0101) et valide cette adresse par le signal ALE.

T_2 : Utilisant le signale$\overline{IOR}$= 0, le up permet à la porte d'entrée adressée de mettre ses données (E7) sur le Bus de donnée.

T_3: l'information contenue sur le Bus de données est transférée dans l'accumulateur.

L'exécution de l'instruction terminé, le µp est prêt pour l'instructeur suivant le µp répète les séquencesdirectes au dessus pour chaque instruction du programme.

Nous pouvons constater que le cycle entré de l'instruction IN 01 est divisé en 3 cycles machines et 10 impulsions d'horloge.

Si la fréquence de l'horloge est de 2MHZ T= 5µs = µp

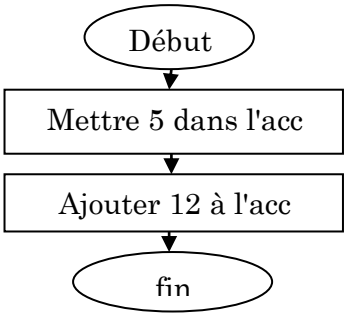
A nécessité un temps t = 10. T = µs. Pour exécuter l'instructeur IN.

Le temps d'exécution d'un programme dépende du temps d'exécution des instructions.

Pour le µp 8085, le temps d'exécution d'une instruction est de 8 µs, ainsi il peut exécuter 125.000 instruction par seconde.C'est la vitesse de travail du µp.

II.2.4. Exemples d'exécution d'un programme:

Soit à additionner les nombres 5 et 12 et le résultat reste tans l'A.



1. La 1ère instruction commande le chargement du nombre 5 dans l'accumulateur.
2. La 2ème instruction ordonne l'addition du nombre 12 au contenu de l'accumulateur.
3. Le résultat reste dans l'accumulation.

Les instructions	Code assembleur	Code machine
Changer l'acc avec 5 Additionner 12 à l'acc	MVI 05 ADI 12	3E 05 C6 12

Supposant que ce programme occupe les cellules ménines à partir de l'adresse 0102 dans la ménine centrale. Chaque cellule contient un octet.

adresse	mémoire
0102	3E
0103	05
0104	C 6
0105	12
.	.
.	.

Remarque: avant l'exécution d'un programme mettre toujours RESET pour remettre à zéro tous les registres internes du µp.

1ère étape: situation du départ du programme:

Compteur programme: CP	0102
Registre d'instruction: RI	?
Accumulateur: A	?

3E	0102
05	0103
C 6	0104
12	0105

Le CP démarré toujours avec l'adresse de la 1ère instruction dans notre cas 0102

2ème étape: lecture du 1er octet de l'instruction: code opération:

Compteur programme: CP	0103
Registre d'instruction: RI	3E
Accumulateur: A	?

3E	0102
05	0103
C 6	0104
12	0105

L'octet est ramené dans le RI Aussitôt le CP d'incrémente et pointeur vers la cellule suivante 0103.
(Le code 3 E est copié et non effacé de la ménine.

3ème étape: décodage et recherche de la donnée:

Compteur programme: CP	0104
Registre d'instruction: RI	3E
Accumulateur: A	05

3E	0102
05	0103
C 6	0104
12	0105

Le code 3E est décodé, le µp cherche la donnée 05 dans la cellule ménine d'adresse 10. La 1ère instruction est terminée le CP d'incrémente à nouveau.

4ème étape: recherche de la 2ème instruction:

Compteur programme: CP	0105
Registre d'instruction: RI	C6
Accumulateur: A	05

3E	0102
05	0103
C 6	0104
12	0105

Le µp lit le code de la 2ème instruction qui est C6 et le met de le RI.Le CP s'incrémente encore.

5ème étape: exécution de l'addition:

Compteur programme: CP	0106
Registre d'instruction: RI	C6
Accumulateur: A	17

3E	0102
05	0103
C 6	0104
12	0105

Le code C6 est décodé, Le µp doit lire la donnée 12 dans la cellule ménine d'adresse 0105 pour l'additionne à l'accumulateur. Le CP d'incrémente

L'exécution de programme est terminée.

Le microprocesseur Intel 8085 est apparu en 1976. Il est équipé d'un bus de données de 8 bits et un bus d'adresses de 16 lignes d'adresses.

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.

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. Cependant, il était plus lent que le 8080. Sa vitesse était de 1.5 million d'instructions à la seconde avec une horloge à 6.144 MHz et 4 cycles par instruction.

Le 8085 était vendu dans un pack à 40 broches, ce qui lui permettait d'utiliser un bus d'adresses sur 16 bits et un bus de données sur 8 bits, donnant un accès facile à 64 KO de mémoire. Il a sept registres de 8 bits (dont six peuvent être combinés dans trois registres de 16 bits), un pointeur de pile sur 16 bits et un compteur de programme sur 16 bits. Le 8085 possède 256 ports d'entrées/sorties accessibles par des instructions dédiées.

III 1. Architecture externe du 8085

Le microprocesseur 8085 se présente sous la forme d'un boîtier DIP (Dual In-line Package) à 40 broches. La figure III.1 montre le schéma de brochage du 8085. Tout les signaux peuvent être classés en six groupes : (1) Bus d'adresses, (2) Bus de données, (3) Bus de contrôle et signaux d'état, (4) Alimentation et générateur de fréquence, (5) Signaux d'interruptions (6) Ports d'E/S sériels.

III 1.1. Bus d'adresses et bus de données

Le nombre de lignes d'adresses du 8085 est 16 lignes A_{15}-A_0 dont le poids faible est multiplexé temporellement avec le bus de données AD_7-AD_0. 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'adresses, d'où 2^{16} = 64 Ko d'espace mémoire adressable par le 8085.

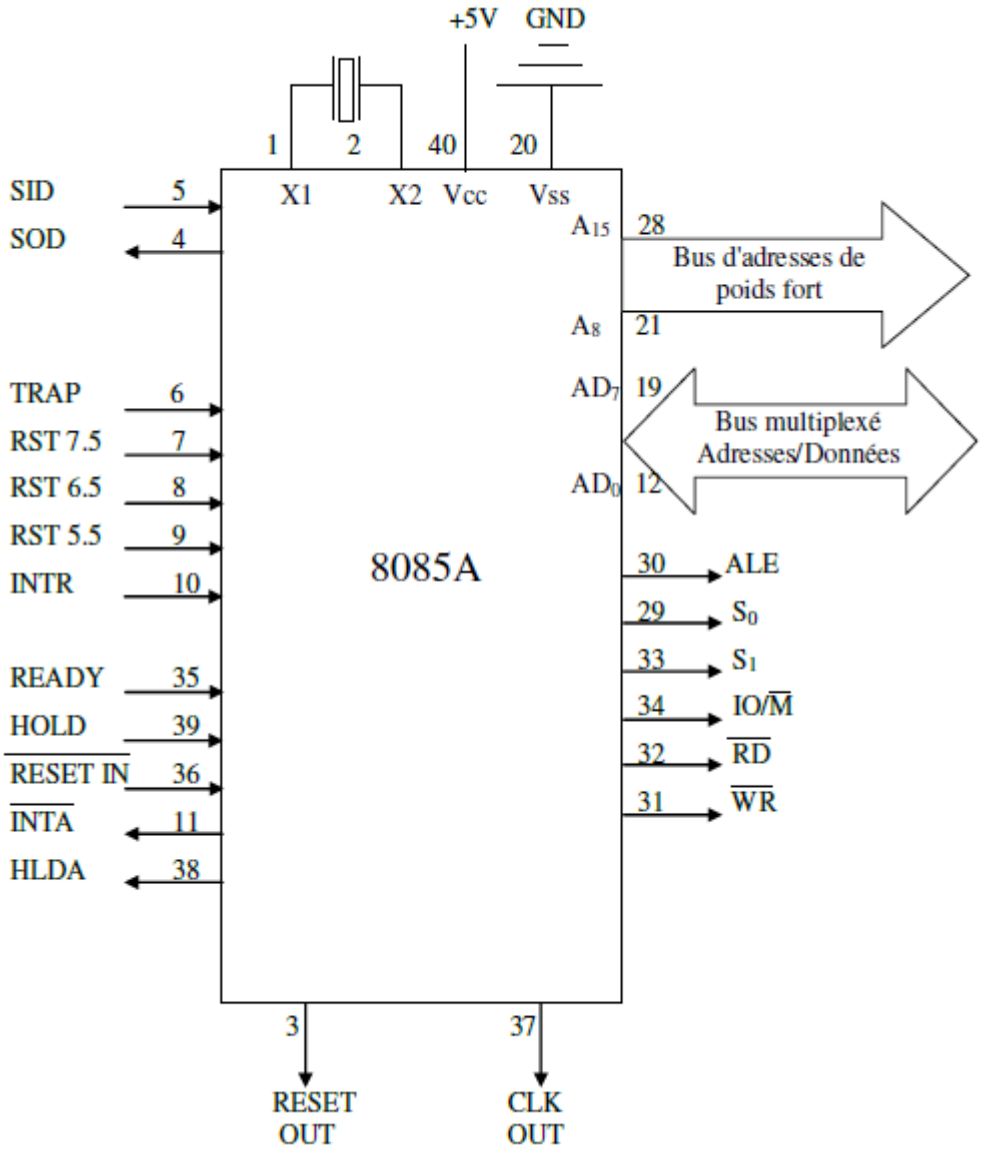


Figure III.1: Architecture externe du 8085

III 1.2. Signaux de contrôle et d'état

ALE : A l'état haut, implique que les bits présents su le bus A/D sont les lignes d'adresses. Un circuit 74373 mémorise ces bits à sa sortie.

RD : Read, signal de lecture d'une donnée, actif niveau bas.

WR : Write, signal d'écriture d'une donnée, actif niveau bas.

$\mathbf{IO/\overline{M}}$: Input-Output / Memory, indique si le 8085 adresse la mémoire (IO/M = 0) ou les entrées/sorties ($\mathbf{IO/\overline{M}} = 1$).

S_1et **S_0**: Signaux d'état indiquant le type d'opération en cours sur le bus. Rarement utilisésdans les petits systèmes.

III 1.3. Alimentation et générateur de fréquence

Vcc : Alimentation +5V

GND : Masse de référence

X_1et **X_2**: Un Quartz est relié aux deux broches pour générer un signal carré périodique. Pourgénérer une fréquence de 3MHz, le quartz doit avoir une fréquence de 6MHz.

CLK (OUT) : Cette broche peut être utilisée comme signal d'horloge pour d'autres circuits.

III 1.4. Signaux d'interruptions

Le 8085 possède cinq signaux d'interruptions.

INTR (Interrupt Request) : C'est le signal envoyé par une interface indiquant une demande d'interruption.

$\overline{\text{INTA}}$: Le 8085 répond à INTR en envoyant 0 sur la ligne $\overline{\text{INTA}}$: (Interrupt Aknowledge).

RST7.5 RST6.5 et RST5.5 : Interruptions de redémarrage. Ce sont des interruptions vectorisés qui transfèrent le contrôle à une position mémoire spécifique. Ces lignes sont prioritaire que INTR. Les priorités sont classées selon l'ordre décroissant de 7.5, 6.5 à 5.5.
TRAP : (Non Maskable Interrupt) : interruption prioritaire par rapport à INTR.

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

READY : entrée de synchronisation avec la mémoire.

RESET IN: Quand ce signal est à 0, le compteur de programme est remis à 0, les bus sont en haute impédance et le microprocesseur redémarre.

RESET OUT : Ce signal indique que le microprocesseur vient d'être redémarrer, ce signal peut être utilisé par d'autres circuits.

III 1.5. Ports d'E/S sériels

Le 8085 possède deux signaux pour l'implémentation de la transmission série : SID (Serial Input Data) et SOD (Serial Output Data). Dans la transmission série, les bits de données sont transmises sur une seule ligne l'un après l'autre comme dans le cas de la ligne téléphonique.

III 2. Démultiplexage des bus AD7-AD0

La figure III.2 montre l'utilité des latchs pour le démultiplexage de bus. Le bus AD_7-AD_0 est connecté aux entrées des latchs 74LS373. ALE est connectée à la broche de validation (G) des latchs. Quand ALE est à l'état haut, les latchs sont transparents; ça veut dire que les sorties des latchs changent en fonction des entrées. Quand le microprocesseur envoi un 0 sur ALE, l'adresse est mémorisée jusqu'au prochain ALE et la sortie des latchs représente le bus d'adresses de poids faible A_7-A_0. Le schéma de la figure III.2 indique une lecture mémoire de la valeur 4FH à partir de l'adresse 2005H

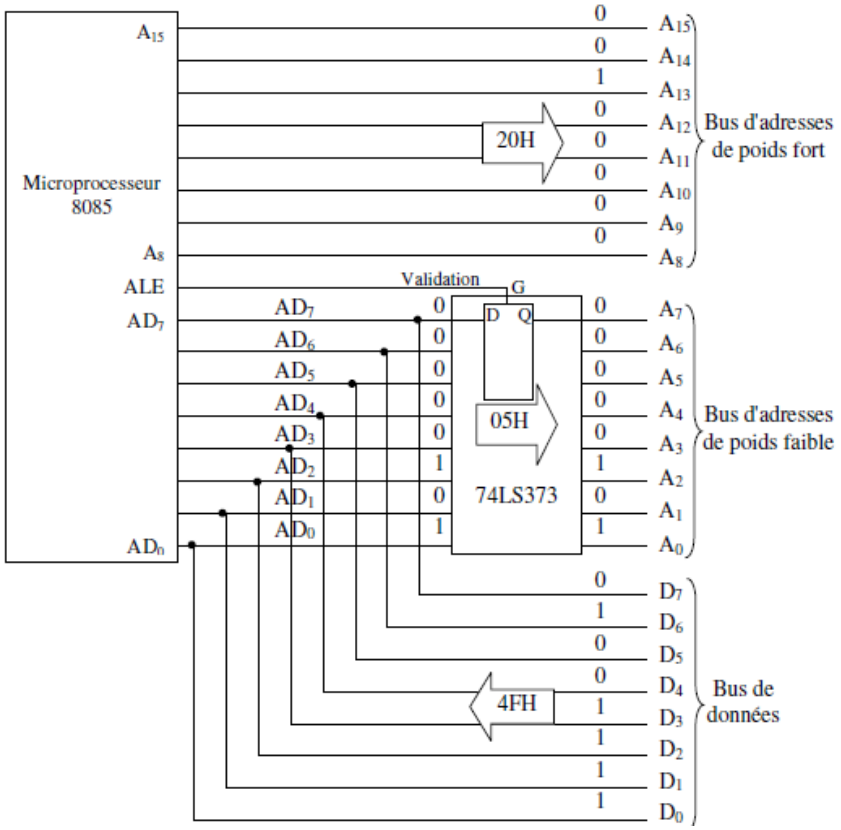


Figure III.2: Démultiplexage des bus d'adresses et de données AD_0-AD_7

III 3. Architecture interne du 8085 [16]

La structure interne du microprocesseur est d'une très grande complexité. Pour essayer de la comprendre on peut cependant la diviser en quatre grandes sections : Accumulateur, registres généraux, UAL, Registre d'instructions, Registres d'adresses, décodeur d'instructions, contrôleur d'interruptions, contrôleur des entrées / sortie série et séquenceur

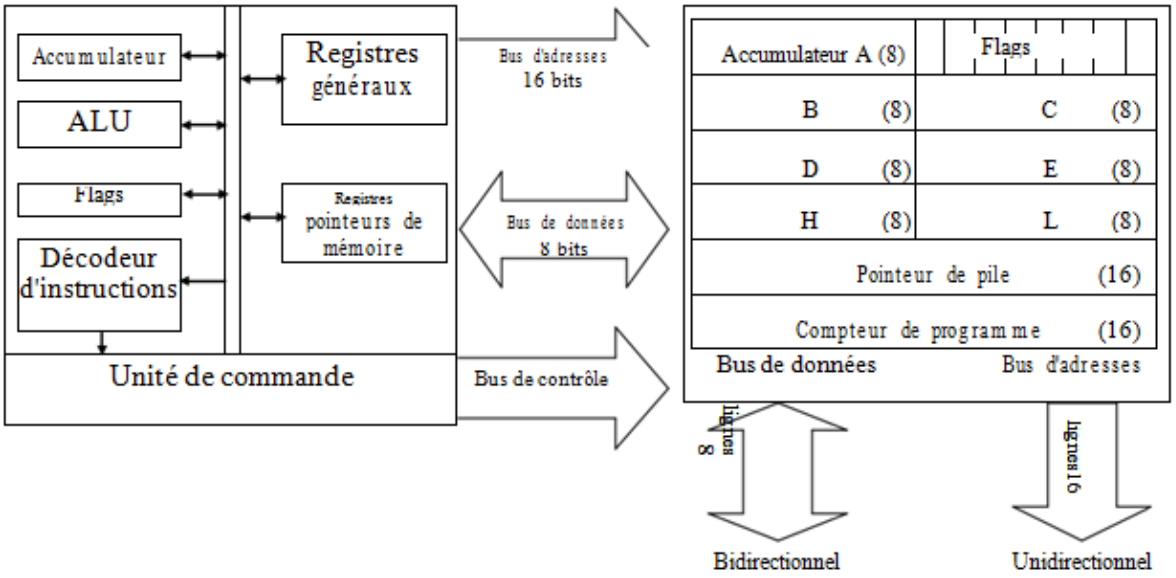


Figure III.3 : Architecture interne de base du microprocesseur 8085

Tous les blocs sont connectés avec des connexions internes appelés bus interne. Les opérations arithmétiques et logiques sont réalisées par l'unité arithmétique et logique. Les résultats sont stockés dans l'accumulateur et les drapeaux (flags). Le 8085 utilise le bus d'adresses pour envoyer les adresses vers les mémoires et périphériques, les huit lignes de données sont les lignes de transfert vers/des mémoires/interfaces d'E/S.

III.3.1. Registres de base du 8085

Nous avons déjà parlé aux registres d'un microprocesseur de façon générale, maintenant, nous allons présenter les différentes registres du microprocesseur 8085, c'est juste pour illustrer nos connaissances sur ce type de microprocesseur (8085), ces registres sont :

PC : Registres de 16 bits contenant l'adresse de l'instruction à exécuter.

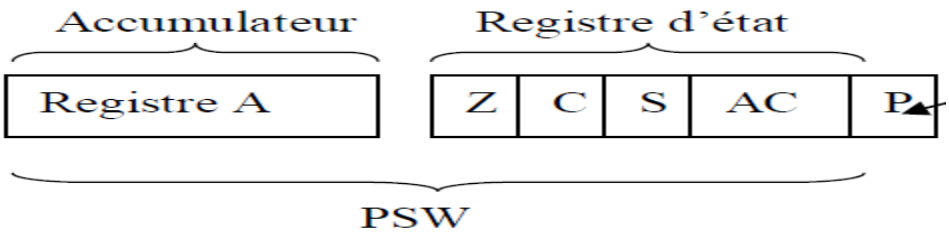
A, B, C, D, E, H et L : Registres de 8 bits multi usage. Ils ont une utilité proche de la RAM, mais étant interne au microprocesseur, ils ont des fonctionnalités additionnelles.

Leregistre A s'appelle également l'accumulateur.

SP : **Stack Pointer**, il a pour rôle d'indiquer au microprocesseur la prochaine case disponible dans la pile.

III.3.2 Registre d'état

C'est l'ensemble des drapeaux (flags) qui forme le registre d'état. Le registre d'état associé à l'accumulateur forme le contexte ou PSW (Program Status Word



Les flags réagissent après certaines instructions et permettent de savoir si le résultat de l'opération est zéro (flag Z), si il y a eu un dépassement (flag C), si le nombre de bit à 1 est pair (flag P) et le signe (flag S).

III.4. Modes d'adressage

Les façons de désigner les opérandes constituent les "modes d'adressage". Le microprocesseur 8085 possède quatre modes d'adressage :

- Mode d'adressage par registre.
- Mode d'adressage immédiat.
- Mode d'adressage direct.
- Mode d'adressage indirect.

III.4.1. Adressage par registre

Ce mode d'adressage concerne tout transfert ou toute opération, entre deux registres de 8 bits.

Dans ce mode l'opérande sera stocké dans un registre interne au microprocesseur.

Exemple :

MOV A, B ; cela signifie que l'opérande stocké dans le registre B sera transféré vers le registre A. Quand on utilise l'adressage par registre, le microprocesseur effectue toutes les opérations d'une façon interne. Donc dans ce mode il n'y a pas d'échange avec la mémoire, ce qui augmente la vitesse de traitement de l'opérande.

III.4.2. Adressage immédiat

Dans ce mode d'adressage l'opérande apparaît dans l'instruction elle- même, exemple :

MVI A,50H ; cela signifie que la valeur 50H sera stockée immédiatement dans le registre A.

III.4.3. Adressage direct

Dans ce mode on spécifie directement l'adresse de l'opérande dans l'instruction. Par exemple pour charger l'accumulateur A par le contenu de l'adresse 2000H, on doit écrire : LDA 2000H

III.4.4. Adressage indirect

Dans ce mode d'adressage l'adresse de l'opérande est stockée dans un registre qu'il faut bien évidemment le charger au préalable par la bonne adresse. L'adresse de l'opérande sera stockée dans un registre pair B, D, H ou SP.

Exemple :

LXI H,1234H ; adressage immédiat codé par 21H 34H 12

MOV A,M ; adressage indirect codé par 7E

Pour la deuxième instruction (MOV A,M), le contenu de la case mémoire dont l'adresse se trouve dans le registre pair H ((HL)=1234H) est mis dans l'accumulateur A.

On appelle jeu d'instruction l'ensemble des codes représentant les opérations qu'il peut exécuter. On peut regrouper ces instructions en

- Opérations de transferts
- Opérations arithmétiques
- Opérations logiques
- Opérations de sauts
- Opérations de lectures
- Opérations d'écriture
- Opérations de contrôles

Le jeu d'instruction se trouve en fin du support. Il n'est pas nécessaire de le connaître par cœur, par contre il faut savoir le genre d'instruction qui existe, savoir comment trouver l'instruction dont on a besoin. Cependant, il est très utile de connaître un ensemble d'instructions couramment utilisées. Ce sont ces instructions que nous allons détailler dans ce polycopie.

Remarque : Nous allons bien expliquer toutes les instructions de microprocesseur 8085 c.à.d, nous allons donner une description de chaque instruction dans un tableau avec un exemple clair qui décrive l'exécution de l'instruction.

IV.1. Instructions de transfert

Ce premier groupe d'instructions permet de transférer (copier) des données d'un endroit un autre (d'un registre ou mémoire ou interface d'E/S) appelé source vers un autre registre (ou E/S ou mémoire) appelée destination. En effet, le contenu de la source n'est pas déplacé, mais copié dans la destination sans modifier le contenu de la source. Les instructions de transfert sont résumées comme suit:

Ce groupe comprend toutes les instructions qui transfèrent des données entre un registre et un autre, ou entre les registres et les emplacements mémoire, et chaque instruction avec un exemple pour illustrer et connaitre bien le mode d'adressage (immédiate direct, registre…) ces instructions sont divisées en :

IV.1. a Instruction de copie des données

- **Copie des données d'un registre à un autre** :MOV Rd , Rs

  Rd : registre destinataire, Rs : le contenu registre source

Exemple : Copier les données de registre A= 34H au registre B :

Execution del'instruction	Résultat de l'exécution	avant execution	A	B
			34	X
Mov B,A	B=34H	aprés execution	34	34

- **Copie de données d'un emplacement ou adresse mémoire vers un autre registre** :

MOV Rd , M	Copier le contenu d'un emplacement ou adresse mémoire M (M adresse spécifiée par HL) vers un autre registre Rd

Exemple : Copier le contenu de l'emplacement du mémoire M(2080)=3Eau registre D				
Exécution de l'instruction	Résultat de l'exécution	avant execution	D	M(2080)
			X	3E
LXI H,2080 MOV D,M	HL=2080 D=3EH	aprés execution	3E	3E

- **Copie le contenu d'un registre vers une zone mémoire (emplacement) :**

MOV M , Rs	Copie le contenu de registre Rs vers une zone mémoire

Exemple : Copier le contenu de registre D= 3Cà l'emplacement du mémoire M(2071)				
Exécution de l'instruction	Résultat de l'exécution	avant execution	D	M(2071)
			3C	X
LXI H,2071 MOV M,D	HL=2071 M(2071)=3CH	aprés execution	3C	3C

IV.1.b Instructions de transfert immédiate (8bits) :

le transfert des données se fait immédiatement (huit bits) vers l'un des registres impairs (R) ou vers une zone mémoire (emplacement) (M) , ces instructions sont :

- **Instructions de transfert immédiat vers un registre**

MVI R , 8 BIT	Chargement ou transfert des données immédiatement vers le registre R

Exemple : Charger immédiatementl'accumulateur par 3EH

Exécution de l'instruction	Résultat de l'exécution	avant execution	A	8bits
MVI A,E3	A=E3H		X	3E
		aprés execution	3E	

- **Transfert immédiat des données vers un emplacement mémoire :**

MVI M , 8 BIT	Transfert ou chargement des données immédiat vers un emplacement de mémoire M

Exemple : Charger immédiatement les données EAH vers l'emplacement de mémoire M (adresse 2070) si les contenus des registres L et H sont70, 20 respectivement.

Exécution de l'instruction	Résultat de l'exécution	avant execution	M(2070)	8bits
MVI M,EA H	M(2070)=EA		X	EA
		aprés execution	EA	

Exemple :

- Charger le registre D par la donnée F5H
- Charger le registre B par la donnée 3DH
- Transférer le contenu de registre B vers l'emplacement mémoire 2090H

Exécution de l'instruction	Resultat de l'exécution	avant exécution	Après exécution
MVI D,F5H MVI B,3DH MVI H,20H MVI L,90H MOV M,B HLT	D=F5H B=3DH H=20H L=90 M(2090)=3DH	D X B X H X L X M(2090) X	F5 3D 20 90 3D

2000	MVI D
2001	F5
2002	MVI B
2003	3D
2004	MVI H
2005	20
2006	MVI L
2007	90
2008	MOV M,B
2009	HLT

Le programme dans la mémoire

IV.1.c Instruction de chargement immédiat (16bits) :

le transfert des données se fait immédiatement (16bits) vers l'un des registres pairs (R)

LXI Rp , 16 BIT	Transfert ou chargement des données (16 Bits) immédiat vers l'un des registres pairs Rp.	
Exemple : Charger le registre pair BC par la donnée E44FH		
Exécution de l'instruction	Résultat de l'exécution	avant execution B C X X
LXI B, E44FH	BC=E44FH	aprés execution E4 4F

- **Les instructions de chargement indirect** : ce sont deux instructions qui se font le chargement du contenu d'une adresse mémoire spécifié dans un registre B ou D vers l'accumulateur

LDAX Rp	Chargement du contenu de l'emplacement mémoire spécifié par l'adresse de deux registres paires BC et DE seulement vers l'accumulateur (cette instruction ne s'exécute seulement qu'avec les deux registres paires BC et DE, elle ne fonctionne pas avec le registre HL
STAX Rp	Chargement ou stockage du contenu de l'accumulateur dans l'emplacement mémoire spécifié par l'adresse de deux registres paires BC et DE seulement vers (cette instruction ne s'exécute seulement qu'avec les deux registres paires BC et DE, elle ne fonctionne pas avec le registre HL

Exemple :

Si le contenu de registre pair BC est 2035H et le contenu de l'emplacement mémoire de l'adresse AFH, charger le contenu de l'adresse 2035H vers l'accumulateur

LDAX B après l'exécution A= AF

- **Les instructions de chargement direct** : ces sont deux instructions qui se font le chargement direct de l'accumulateur par le contenu d'une adresse la mémoire

LDA address (16b)	Chargement de l'accumulateur par le contenu d'une adresse mémoire
STA address (16b)	Stockage l'accumulateur par le contenu d'une adresse mémoire

Exemple
Charger l'accumulateur par le contenu de l'adresse mémoire 2040H supposant que le contenu est 3EH

Exécution de l'instruction	Résultat de l'exécution	Avant l'exécution M(2040) **A**
LDA 2040H	A=3E	3E X
		après l'exécution 3E 3E

LHLD address (16b)	Chargement du registre pair H par le contenu de la case mémoire (le contenu de l'adresse mémoire cité sera chargé dans le registre L et le contenu de la prochaine adresse sera chargé dans le registre H)
SHLD address (16b)	Stockage du registre pair H par le contenu de la case mémoire (le contenu de l'adresse mémoire cité sera stocké dans le registre L et le contenu de la prochaine adresse sera stocké dans le registre H)

Exemple
Si les contenus des adresses mémoire2060H et 2061H sont ADH et 2BH respectivement, charger les contenus de ces adresses directement au registre HL

Exécution de l'instruction	Résultat de l'exécution	Avant l'exécution H L M
LHLD 2060 H	L=AD H=2B	X XX AD (2060) 2B (2061)
		après l'exécution H L
		2B AD

Exemple : Ecrire un programme par assembleur du microprocesseur 8085 qui permet d'effectuer les taches suivantes :

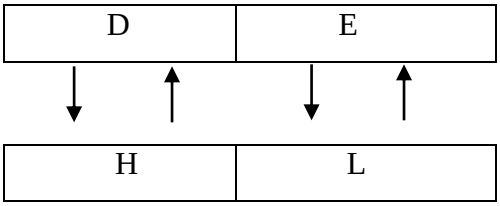
Début de programme (l'adresse mémoire 2000h)

- Charger les données 6Dh, 47h dans les adresses mémoire 3060H, 3090H, respectivement (HL)
- Charger l'accumulateur par le contenu de l'adresse mémoire 3060H
- Transférer le contenu de l'accumulateur au registre E
- Transférer le contenu de l'adresse mémoire 3090H vers le registre B

Code	Resultat
LXI H , 3060H	HL = 3060
MVI M , 6DH	M (3060)= 6D
LXI H , 3090H	HL = 3090
MVI M , 47H	M(3090) = 47
LDA , 3060H	A = 6D
MOV E , A	E = 6D
MOV B, M	B = 6D
HLT	HLT

2000	LXI H
2001	L = 60
2002	H = 30
2003	MVI M
2004	6D
2005	LXI H
2006	L = 90
2007	H = 30
2008	MVI M
2009	47
2010	LDA
2011	60
2012	30
2013	MOV E , A
2014	MOV B , M
2015	HLT

- **Instruction XCHG**:: Echange des contenus de HL et DE.(contenu de H avec D et le contenu de L avec E



Exemple

Effectuer l'échange des contenus des registres pairs HL et DE si les contenus des registres sont HL =4DE2 H,DE= 12EAh, respectivement

	D	E	H	L
Avant l'exécution	12	EA	4D	E2
Après l'exécution	4D	E2	12	EA

IV.2 Instructions entries /sorties IN/OUT

IN Port address(8b)	Recevoir un octet de données d'une interface d'entrée dans l'accumulateur
OUT Port address(8b)	Envoyer un octet de données depuis A vers une interface de sortie

Exemple

Envoyer un octet de données depuis A vers une interface de sortie (port 01)

- **OUT 01**

Exemple

- Charger la donnée 3050 dans le registre BC
- Charger la donnée 9BH dans l'accumulateur
- Stocker le contenu de l'accumulateur dans l'emplacement mémoire spécifie par le registre BC
- Charger le registre pair HL par la donnée 5D34H
- Stocker le contenu de registre pair HL dans l'emplacement mémoire 3080H et3081H

LXI B , 3050H MVI A , 9BH STA X B LXI H , 5D34H SHLD , 3080H	BC = 3050 A = 9B M(3050) = 9B HL =5D34 M(3080) = 34 M(3081) = 5D

Position memoire	Code	Opcode
2000 2001	LXI B	
2002	30	
2003	MVI A	
2004	9B	
2005	STAX B	
2006	LXI H	

2007	34	
2008	5D	
2009	SHLD	
200A	80	
200B	30	
200C	HLT	

IV.3 Instructions de gestion de la pile

Le début de la pile est défini dans le programme par l'utilisation de l'instruction LXI SP, qui charge l'adresse mémoire de la pile dans le registre pointeur de pile (SP). En général SP pointe vers le top de la pile. Par exemple si SP est chargé par 2099H (LXI SP,2099H), le stockage des données commence à partir de 2098H et continu selon l'ordre 2097H, 2096H,... La taille de la pile dépend de la taille de la mémoire disponible.

Les contenus des registres pairs peuvent être stockés dans la pile (deux octets en même temps) par l'utilisation de l'instruction PUSH. Pour transférer les données depuis la pile vers les registres POP est utilisée. Deux octets sont transférés de/vers la pile, ce qui implique que lors d'un PUSH SP se décrémente de 2 et lors d'un POP SP s'incrémente de 2.

- **Instruction PUSH et POP**

Les instructions PUSH et POP peuvent être utilisées pour échanger les contenus des registres pairs B et D ou H ou registre PSW (Program Status Word) qui est l'ensemble Accumulateur (A) et registre d'état (F).

En peut résumer :

- La précision de début de la zone la pile en utilise l'instruction (LXI, adresse de debut)
- Un stockage(Envoie)du contenu se fait par l'utilisation de l'instruction PUSH Rp
- Le transfert (récupération)des données stockées par la pile vers les registres en utilisons l'instruction POP Rp

1.	(LXI, adresse de début)	Le début la zone la pile de manière directe
2.	SPHL	Le début la zone la pile de manière indirecte
3.	PUSH Rp	Envoie) du contenu des registres Pairs
4.	POP Rp	Récupération des données des registres pairs

Remarque : SPW= les Flags et l'accumulateur

Exemple :

Si le contenu du pointeur de la pile est 2099 H et le contenu de registre pair BC est 4AB5H, stocker le contenu du registre à l'adresse du pointeur SP

Remarque ; le pointeur de la pile SP va se décrémenter chaque fois, il commence de transférer le contenu de registre B.aprés il transfert le contenu de registre C.

L'exécution de l'instruction	Resultat de l'execution	
PUSH B	1 : SP-1 2099-1= 2098	À cette adresse le contenu de registre B va transférer(High)
	2 : SP-1 2098-1= 2097	À cette adresse le contenu de registre B va transférer (low)
	3 : SP= 2097	Le pointeur de pile s'arrête ici

Avant l'exécution	C 4C B 5B	SP 2099	2099 XX 2098 XX 2097 XX
Après l'exécution	B 4C C 5B	SP 2097	2099 B5 2098 4C 2097 XX

1. Instruction POP

Exemple

Si le contenu du pointeur de la pile SP est 2090 H et le contenu de emplacement mémoire 2090h et 2091H sont F5 H et 01 H, respectivement, stocker les contenus du registre pair HL

Remarque : le pointeur de la pile SP va s'incrémenter chaque fois, il commence de transférer le contenu de registre L.aprés il transfert le contenu de registre H. (il commence de transférer le contenu de registre L (low) après il transfert le contenu de registre H (High)

Avant l'exécution	Resultat de l'execution
POP H	1 : SP= 2090

	2 : SP+1 2090+1= 2091
	3 : SP+1 2091+1 = 2092

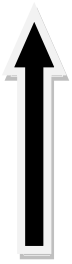
Avant l'exécution de l'instruction	H L XX XX	SP 2090	2090 2091 F5 01
Avant l'exécution de l'instruction	H L 01 F5	SP 2092	F5 01

Exemple

Transférer les données suivantes [12h, 14h, 20h,30h, F1h, CDh] dans la pile si le pointeur de la pile commence par cette adresse E222h

```
LXI SP, E222H
LXI B,1214h
PUSH B
LXI B,2030h
PUSH B
LXI B,F1CDh
PUSH B
```

Position memoire	**Data**
E21Dh	CD
E21Eh	F1
E21Fh	30
E220h	20
E221h	14
E222h	12



Comme nous avons déjà dit les instructions PUSH et POP peuvent être utilisés pour effectuer l'échange des contenus des registres pairs B et D.

Pour bien illustrer et comprendre le fonctionnement de la pile on utilise deux instructions push et pop dans cet exemple :

Architecture du programme est toujours le suivant :

PUSH B

PUSH D

POP B

POP D

La figure IV.1 montre comment une valeur est stockée dans la pile et la figure IV.2 montre comment elle est récupérée :

Pour le registre pair B=1234H et SP=2000H

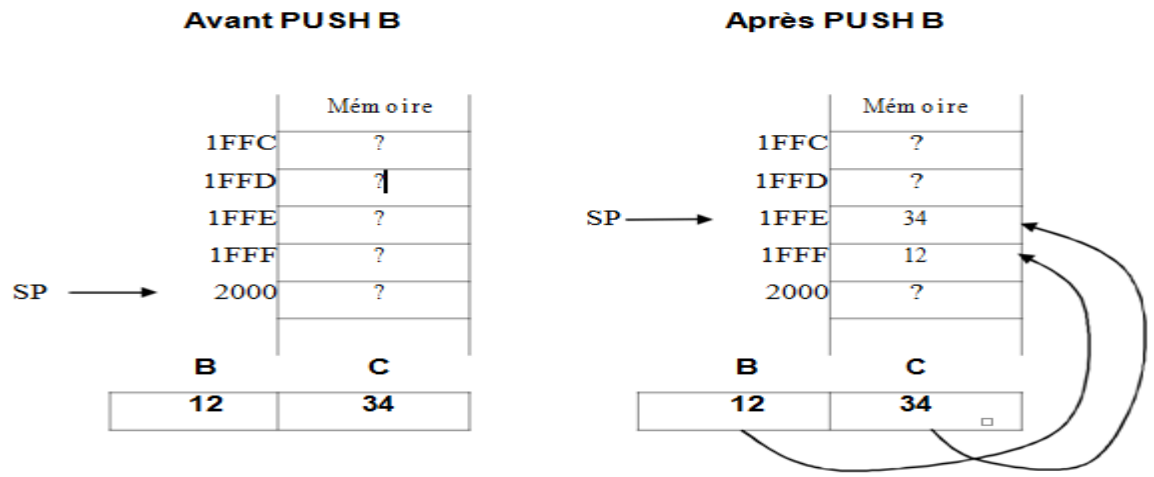


Figure IV.1 : Exécution de PUSH B

SP=2000H

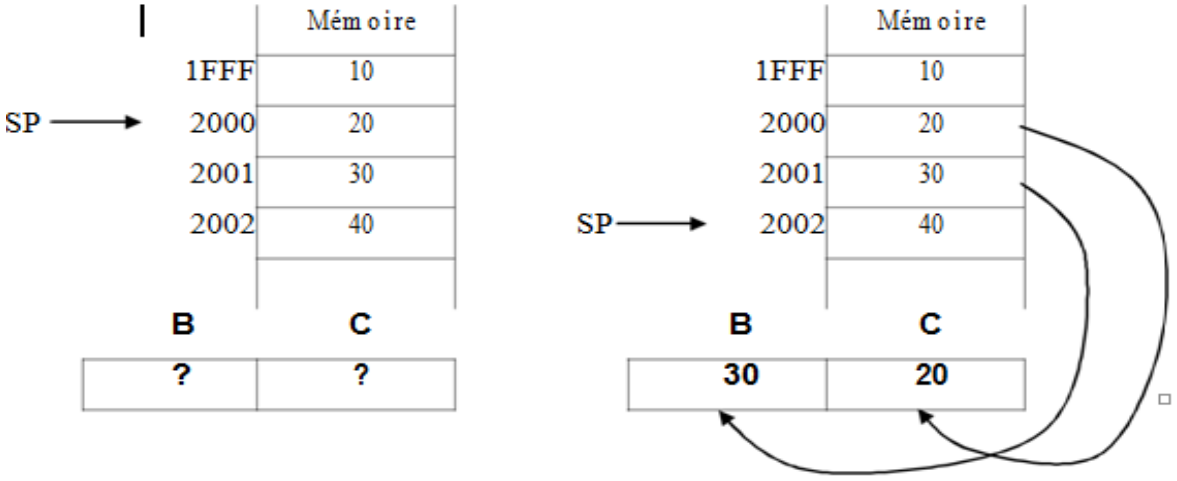


Figure IV.2 : Exécution de POP B

IV.4. Instructions arithmétiques

IV.4.1 Les instructions de l'addition (µp 8085)

- **Instruction ADD :** Additionner le contenu de l'accumulateur A avec le contenu du registre ou case mémoire, il stock le résultat dans l'accumulateur.

ADD R	Additionner le contenu de l'accumulateur A avec le contenu du registre R(8 bits), le résultat est stocké dans l'accumulateur
ADD M	Additionner le contenu de l'accumulateur A avec le contenu d'une case mémoire (l'adresse spécifie par le registre pair HL) le résultat est

	stocké dans l'accumulateur
ADI data(8b)	Additionner immédiatement les données (une valeur de 8 bits) avec le contenu de l'accumulateur, le résultat est stocké dans l'accumulateur
ADC R	Additionner le contenu du registre R(8 bits) et le retenu avec le contenu de l'accumulateur A , le résultat est stocké dans l'accumulateur
ADC M	Additionner avec le contenu d'une case mémoire (l'adresse spécifie par le registre pair HL) et le retenu avec le contenu de l'accumulateur A, le résultat est stocké dans l'accumulateur
ACI data(8b)	Additionner immédiatement les données (une valeur de 8 bits) et le retenu avec le contenu de l'accumulateur, le résultat est stocké dans l'accumulateur

instruction	**Exemple** Additionner le contenu de registre C avec le contenu de l'accumulateur A si A=3AH و C=AEH	
ADD C	0011 1010	A=3AH
	+ 1010 1110	C=AEH
	1110 1000	A=E8H
	CY=0, AC=1, P=1, Z=0, S=1	drapeaux de registre d'état

Exemple

Additionner le contenu de registre A et contenu de l'accumulateur si A=23H

Remarque : l'accumulateur est le même registre A c.à.d. l'instruction va s'additionner le l'accumulateur avec elle-même

ADD A	0010 0011	A=23H
	+ 0010 0011	A=23H
	0100 0110	A=46H
	CY=0, AC=0, P=0, Z=0, S=0	drapeaux de registre d'état

	Additionner le contenu de registre B=33H avec le contenu de registre si D=A1H	
MOV A, B ADD D	0011 0011	A=33H
	+ 1010 0011	D=A1H
	1101 0100	A=D4H
	CY=0, AC=0, P=1, Z=0, S=1	drapeaux de registre d'état

	Additionner le contenu de registre H=3EH avec le contenu de l'accumulateur A=36H Si CY=1	
ADC H	0011 0110	A=36H
	+ 1	CY =1
	0011 0110 + 0011 1110	A+CY=37H H=3EH
	0111 0101	A=75H
	CY=0, AC=1, P=0, Z=0, S=0	drapeaux de registre d'état

	Additionner immédiatement la donnée (une valeur) 25H avec le contenu de l'accumulateur A=10H	
ADI 25H	0001 0000	A=10H
	+ 0010 0101	25h
	0011 0101	A=35H
	CY=0, AC=0, P=1, S=0, Z	drapeaux de registre d'état

	Additionner immédiatement la donnée (une valeur) 76H avec le contenu de l'accumulateur si A=10H et CY=1 (retenu)	
ACI 76	0001 1101 + 1	A=1DH
		CY =1
	0001 1110 + 0111 0110	A=1EH 76H
	1001 0100	A=94H
	CY=0, AC=1, P=0, S=1, Z=0	drapeaux de registre d'état

	Additionner immédiatement la donnée (une valeur) 89H avec le contenu de l'accumulateur si A= DAH etla valeur de registre d'etat est 4EH	
Remarque ; en veut exécuter l'opération de l'addition utilisant le retenu, donc il nécessaire de trouver la valeur de retenu car il est inconnu, mais on a la valeur de registre d'état(flag) Flag 4E = 0100 111**0**→ CY=0		
D7 D6 D5 D4 D3 D2 D1 D0 **S** Z X AC X P X CY		
ACI 76	1101 1010 + 0	A=DAH + CY=0H

	1101 1010 + 1000 1001	A+CY=DAH + 89H
	1 0110 0011	A=63H, CY=1
	CY=1, AC=1, P=1, S=0, Z=0	drapeaux de registre d'état

DAD	Additionner la donnée (une valeur de 16bits) 3214 H avec le contenu de registre HL=34D2H	
Remarque: Il faut stocker la donnée de 16bits 3214 H dans l'un des registres pairs(BC) après l'instruction DAD fait l'addition de la donnée avec le contenu de registre HLet le résultat est stocké dans le même registre HL		
LXI B,3214 **DAD B**	0011010011010010 + 0011001000010100	HL=34D2H + BC=3214H
	0110011011100110	HL=66E6H

Exemple

	Additionner les données stockées dans les emplacement mémoire 2030h [20], 2040h [3E]et 2050h [30] et stocker le résultat dans le registre E
MVI A,00h	A=00h
LXI H,2030h	HL=2030h
ADD M	A=00+20=20
LXI H,2040h	HL=2040h
ADD M	A=20+3E=5E
LXI H,2050h	HL=2050h
ADD M	A=5E+30=8E
HLT	

IV.4.2instructions de subtraction

SUB R	soustraire le contenu de l'accumulateur A avec le contenu du registre R(8 bits), le résultat est stocké dan l'accumulateur
SUB M	soustraire le contenu de l'accumulateur A avec le contenu d'une case mémoire (l'adresse spécifie par le registre pair HL) le résultat est stocké dans l'accumulateur
SUI data(8b)	Soustraire immédiatement les données (une valeur de 8 bits) avec le contenu de l'accumulateur, le résultat est stocké dans l'accumulateur
SBB R	soustraire le contenu du registre R(8 bits) et le retenu avec

	le contenu de l'accumulateur A , le résultat est stocké dans l'accumulateur
SBB M	soustraire le contenu d'une case mémoire (l'adresse spécifie par le registre pair HL) et le retenu(Borrow) avec le contenu de l'accumulateur A , le résultat est stocké dans l'accumulateur
SBI data(8b)	soustraire immédiatement les données (une valeur de 8 bits) et le retenu (Borrow) avec le contenu de l'accumulateur, le résultat est stocké dans l'accumulateur

	Exemple Effectuer la soustraction de contenu de registre C avec le contenu de l'accumulateur A si A=37H و C=40H	
SUB C	0011 0111 - 0100 0000	**37** - 40
	1 1111 0111	1 F7
	CY=1, AC=0, P=0, Z=0, S=1	drapeaux de registre d'état

	Exemple : Soustraire le contenu de registre D=4EH avec le contenu de registre C=44H si report=1	
Soustraction avec l'accumulateur, il faut prendre le report en considération pour exécuter cette instruction		
MOV A,C SBB D	44 - 4E	A=44H A=A-D-Report
	1F6 - (report) 1	A=F5H
	1F5	

	CY=1, AC=0, P=0, Z=0, S=1	drapeaux de registre d'état

IV.4.3 instruction d'incrémentation

INR R	Incrémenter le contenu d'un registre R par 1 (+ 1) le résultat est stocké dans le même registre	affecte sur tous les drapeaux de registre d'état sauf CY
INR M	Incrémenter le contenu d'une case mémoire (spécifie par le registre pair HL) par 1 (+1) le résultat est stocké dans le même emplacement	
INX Rp	Incrémenter le contenu des registres paires.(HL, BC, DE et SP) par 1 (+ 1) le résultat est stocké dans le même registre	n'affecte pas

IV.4.4 instructions de decrémentation

DCR R	Décrémenter le contenu d'un registre R par 1 (+ 1) le résultat est stocké dans le même registre	L'instruction affecte tout les drapeaux de registre d'état sauf CY
DCR M	Incrémenter le contenu d'une case mémoire (spécifie par le registre pair HL) par 1 (+1) le résultat est stocké dans le même emplacement	
DCX RP	Incrémenter le contenu des registres paires.(HL, BC, DE et SP) par 1 (+ 1) le résultat est stocké dans le même registre	n'affecte pas

.

Exemple : Si le contenu de registre B=35H incrémenter le contenu de cet registre une fois

INR B	B=36h B=35+1	tout les drapeaux de registre d'état sont affecté sauf CY

Exemple : si le contenu de registre A=FFH incrémenter le contenu de l'accumulateur

INR A	A=FF+1 A=00H	Registre d'état CY= 0 n'affecte pas AC=1 P=1 Z=1 S=0

Exemple :si le contenu de registre A=FFH additionner 01 au contenu de l'accumulateur		
ADI 01	A=A+8Bit A=FF+01 A=00H	Drapeaux de l'addition sont toute affectés CY=1 AC=1 P=1 Z=1 S=0

Incrémenter le contenu de registre BC une fois si le contenu de BC=34C9H		
INX B	BC=BC+1 BC=34C9+1 BC=34CAH	registre de drapeaux n'affecte pas par l'opération

IV.5 Instructions logiques

- **Porte AND**

ANA R	And logique entre l'accumulateur A et le contenu du registre R , le résultat est stocké dans l'accumulateur	drapeaux (P,S,Z) sont affecté par le résultat, mais AC=1,CY=0 ,S=1, Z=0, P=1
ANA M	And logique entre l'accumulateur A et case mémoire.(spécifie par le registre pair HL) le résultat est stocké dans l'accumulateur	
ANI data (8b)	: And logique(immédiate) entre le contenu de l'accumulateur A et une valeur de 8bits	

- **Porte OR**

ORA R	Or logique entre l'accumulateur A et le contenu du registre R, le résultat est stocké dans l'accumulateur	Drapeaux (P,S,Z) sont affecté par le résultat, mais AC=0,CY= 0, S=1, Z=0, P=1
ORA M	Or logique entre l'accumulateur A et case mémoire.(spécifie par le registre pair HL) le résultat est stocké dans l'accumulateur	
ORI data(8b)	Or logique(immédiate) entre le contenu de l'accumulateur A et une valeur de 8bits	

- **Porte XOR**

XRA R	XOR (Ou exclusif) entre l'accumulateur A et le contenu du registre R, le résultat est stocké dans l'accumulateur	Drapeaux (P,S,Z)sont affecté par le résultat,
XRA M	XOR(Ou exclusive) entre l'accumulateur A et	

	case mémoire.(spécifie par le registre pair HL) le résultat est stocké dans l'accumulateur	mais AC=0, CY= 0,
XRI data(8b)	XOR(Ou exclusive) (immédiate) entre le contenu de l'accumulateur A et une valeur de 8bits	

- **Porte NOT**

CMA	Complémenter A : Complémenter l'accumulateur A (inverser tous les bits de A).	n'affecte pas sur les drapeaux.

Exemple : exécuter une AND entre les contenus des registres A=C9H, B=B3H		
ANA B	A = A . B	
	C9 B3	1100 1001 1011 0011 (AND)
	81	1000 0001
		Drapeaux (P, S, Z)sont affectés par le résultat, mais AC=1, CY= 0, S=1, Z=0, P=1

Exemple : Exécuter une AND avec la valeur 9Dh et le contenu de l'accumulateur A= 3E		
ANI ,9Dh	A =3E.9D	0011 1110 1001 1101 (AND)
	A=1C	0001 1100
		P=0 , Z=0 , S=0 , CY=0, AC=1
Exemple : exécuter une OR entre les contenus des registres A=C9H, B=78H		
ORA C	A=C9+78	1100 1001 0111 1000 (OR)
	A=F9	1111 1001
		Drapeaux (P,S, Z)sont affecté par le résultat, mais

		AC=0, CY= 0, S=1, Z=0, P=1

Exemple : Exécuter une OR avec la valeur 9Dh et le contenu de l'accumulateur A= 3E		
ORI ,9DH	A=3E+9D	0011 1110 1001 1101 (OR)
	A=BF	1011 1111
		Drapeaux (P, S, Z)sont affectés par le résultat, mais AC=0, CY= 0, S=1, Z=0, P=0

Exemple : exécuter une XOR entre les contenus des registres A=C9H, B=78H		
XRA C	A =C9XOR 78	1100 1001 0111 1000
	A=B1	10110001
		Drapeaux (P, S, Z)sont affectés par le résultat, mais AC=0, CY= 0, S=1, Z=0, P=1

Exemple : Complémenter le contenu de l'accumulateur (A=89H)		
CMA	A=89h	1000 1001
	A=76	0111 0110
		Le register d'état n'affecte pas

IV.5. Instructions de comparaison

CMP R ou CMP M

R : registre 8 bits et M : Adresse d'une case mémoire spécifiée par HL, c'est une instruction à un seul octet et elle compare le contenu de registre/case mémoire avec le contenu de A.

Aucun contenu n'est modifié par l'instruction CMP, seulement les flags sont affectés par l'opération de soustraction.

If A<R CY=1, Z=0 If A=R CY=0, Z=1 If A>R CY=0, Z=0 Les autres flags sont affectés depent du resultat	Compare le contenu de registre avec le contenu de l'accumulateur A, elle n'affecte pas sur les drapeaux	CMP R
	Compare le contenu de l'accumulateur A et la case mémoire (adresse de la case mémoire spécifiée par HL), elle n'affecte pas sur les drapeaux	CMP M
	Compare le deuxième octet (donnée de 8bits) avec le contenu de A.	CPI data (8b)

Exemple : Comparer le contenu de l'accumulateur A=57H et le contenu de registre B=62H		
CMP B	A-B=57-62=1F5	0101 0111 (57) 0110 0010- (62)
		1 1111 0101 (1F5)
	CY=1 Then A < B	Cy=1, Z=0, Ac=1, P=1, S=1

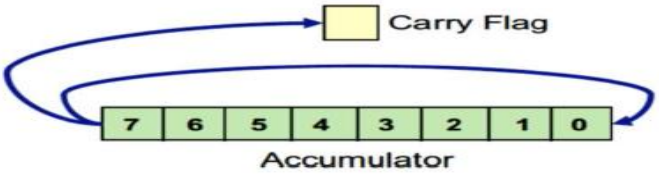
Comparer le contenu de l'accumulateur A=99H et la valeur 4CH		
CPI , 4CH	A - B =99-4C= 4D	1001 1001 (99) 0100 1100 - (4C)
		0100 1101 (4D)
		Cy=0, Z=0, Ac=0, P=1, S=0

IV.6. Instructions logiques de rotation

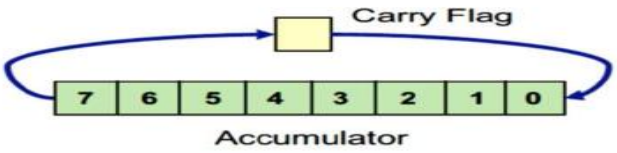
RAL	Rotation à gauche de A par CY. Chaque bit est décalé vers la gauche d'une seule position. Le bit D_7 deviendra CY et le CY est décalé vers D_0.	.affecte seulement sur le report (CY=D7)
RAR	Rotation à droite de A par CY. chaque bit est décalé vers la droite d'une seule position. Le bit D_0 deviendra CY et le CY est décalé vers D_7.	affecte seulement sur le (CY=D0)

RLC	Rotation à gauche de A. chaque bit est décalé vers la gauche d'une seule position. Le bit D_7 deviendra D_0.CY est modifié selon la valeur de D_7	Affecte sur le report CY (D0 et CY←D7)
RRC	Rotation à droite de A. chaque bit est décalé vers la droite d'une seule position. Le bit D_0 deviendra D_7. CY est modifié selon la valeur de D_0.	Affecte sur le report CY (D7 and CY←D0)

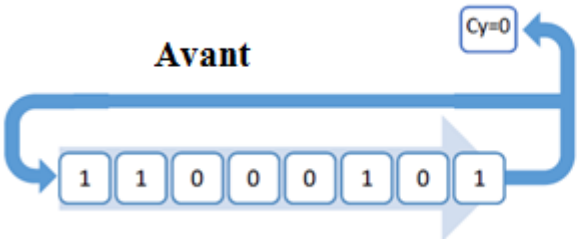
- **RLC**

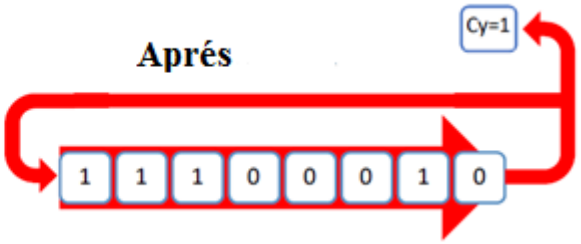


- **RAL**

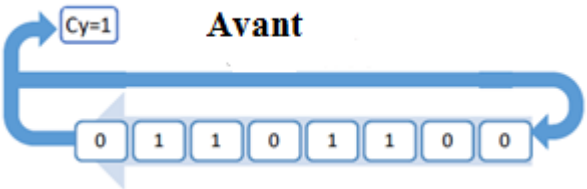


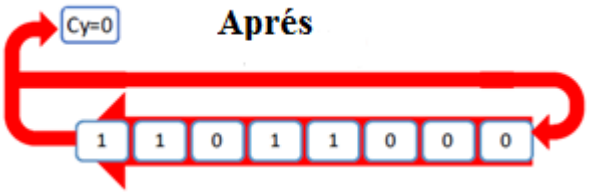
Faire la rotation à droite avec retenue si le contenu de l'accumulateur A=C5 et CY=0		
RRC	Avant A=C5et CY=0	Après A=E2 et CY=1



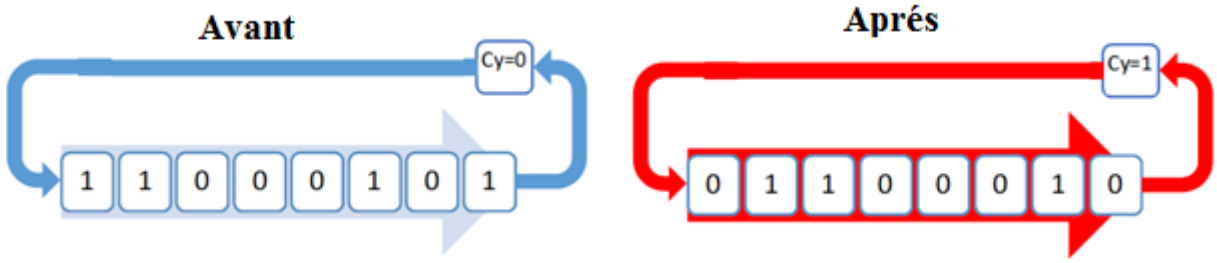


Faire la rotation à gauche avec retenue si le contenu de l'accumulateur A=6C et CY=1		
RLC	Avant A=6C et CY=0	Après A=D8 et CY=0

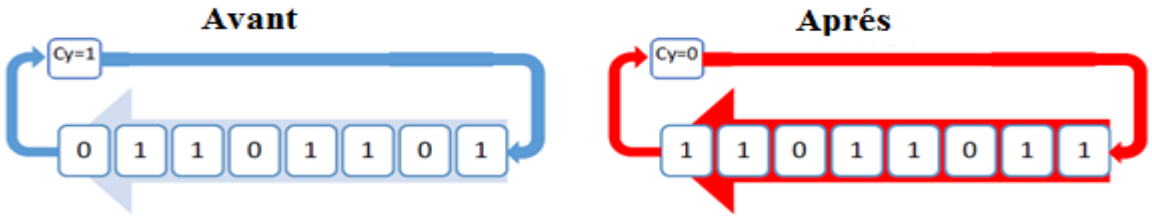




Faire la rotation à droite gauche par le retenue CY si le contenu de l'accumulateur A=C5et CY=0		
RAR	Avant A=C5et CY=0	Après A=D8 et CY=0



Faire la rotation à gauche par le retenue CY si le contenu de l'accumulateur A=6Det CY=1		
RAL	Avant A=6D et CY=1	Après A=DB et CY=0



EXEMPLE : Si le contenu de l'emplacement mémoire 3050H est C2H et le contenu de l'emplacement mémoire est 3051Hest D4H • Additionner les deux contenus mémoires • Faire la rotation à droite par le retenue • Comparer les résultats avec les données FEH	
LXI H , 3050H HL=3050	HL=3050H
MOV A, M	A=C2
INX H	HL=3051H
ADD M	A=A+M=C2+D4= 96, CY=1
RAR	A=CB, CY=0
CPI ,FEH	A<FE [cy=1, P=0, AC=0, S=1, Z=0]

IV.7 Instructions de branchement

Les instructions de branchement permettent au microprocesseur de changer l'exécution séquentielle du programme sous certaines conditions ou sans condition.

Les instructions de branchement sont classées en trois catégories:

- Les instructions de saut,
- Les instructions CALL et RETURN,
- Les instructions de redémarrage.

IV.7 .1 Les instructions de saut :

A.Saut inconditionnel

Le saut inconditionnel est un branchement systématique. L'instruction utilisée dans ce cas est :

JMP Adresse 16 bits, C'est une instruction à trois octets

Exemple

JMP 54F3h : Jump à 54F3h, Force le PC à 54F3h

La prochaine instruction exécutée sera à l'adresse 54f3h.

Remarque : l'opérande de cette instruction est une adresse, soit 16 bits. Cette instruction a donc une taille de 3 octets

Exemple :

0000h MVI A, 55h

0002h CMA

0003h JMP 0002h

adresse ROM	Instruction	PC	A
0000h	MVI A, 55h	0002h	55h
0002h	CMA	0003h	AAh
0003h	JMP 0002h	0002h	AAh
0002h	CMA	0003h	AAh
0003h	JMP 0002h	0002h	AAh
0002h	CMA	0003h	55h
...	...	...	...

B. Sauts conditionnels: le saut est conditionné

Le saut conditionnel se fera si une condition donnée est remplie. Cette condition sera évaluée grâce aux flags.

Opcode	Opérande	Description
JC	16 bits	Saut si retenue (si le résultat génère une retenue : CY=1)
JNC	16 bits	Saut si pas de retenue (Y=0)
JZ	16 bits	Saut si nul (si le résultat est nul : Z=1)
JNZ	16 bits	Saut si non nul (si le résultat est différent de 0 : Z=0)
JP	16 bits	Saut si plus grand (si D_7=0 et S=0)
JM	16 bits	Saut si plus petit (si D_7=1 et S=1)
JPE	16 bits	Saut si pair (P=1)
JPO	16 bits	Saut si impair (P=0)

Nous allons donner des exemples sur les instructions de saut conditionnels et inconditionnels dans la série d'exercices

IV.7.2 .Les instructions CALL

A. Appels inconditionnels

L'instruction : **CALL adresse (16bits)** : Cette instruction est utilisée pour invoquer inconditionnellement un sous-programme à l'adresse donnée mais avant de le déplacer, elle stocke l'instruction suivante dans la zone de pile afin qu'elle puisse retourner correctement.

B.Appels conditionnels

Instruction	Description
CC adresse (16 bits)	Appel sous programme si CY=1
CNC adresse (16 bits)	Appel sous programme si CY=0
CZ adresse (16 bits)	Appel sous programme si Z=1
CNZ adresse (16 bits)	Appel sous programme si Z=0
CM adresse (16 bits)	Appel sous programme si S=1
CP adresse (16 bits)	Appel sous programme si S=0
CPE adresse (16 bits)	Appel sous programme si P=1
CPO adresse (16 bits)	Appel sous programme si P=0

IV.7.3 Instructions de Retour

A. Retours inconditionnel.. RET

RET Retourne du sous-programme au programme principal invoqué inconditionnellement, Le processeur extrait l'adresse de retour du haut de la pile où elle était stockée lorsque le programme secondaire a été appelé

B. Retours conditionnels

RC	Appel sous programme si CY=1
RNC	Appel sous programme si CY=0
RZ	Appel sous programme si Z=1
RNZ	Appel sous programme si Z=0
RM	Appel sous programme si S=1
RP	Appel sous programme si S=0
RPE	Appel sous programme si P=1
RPO	Appel sous programme si P=0

IV.8 Boucles en assembleur

Les programmes illustrés auparavant peuvent être résolus manuellement. Cependant, des difficultés de traitement auront lieu dans le cas des taches répétitives, comme par exemple l'addition de 100 nombres ou transfert de milliers d'octets. La programmation assembleur offre au programmeur la possibilité de répéter des séquences : cela est nommé bouclage. Une boucle permet au microprocesseur de changer la séquence d'exécution et répéter les taches une nouvelle fois. Cela est accompli par l'utilisation des instructions de branchement. En plus peut être des compteurs sont ajoutés pour définir le nombre de répétition. Les boucles sont classées selon deux groupes : boucles infinies et boucles conditionnelles

IV.8.1. Boucles infinies

Ce type de boucles est utilisé pour la répétition infinie. Son principe est illustré par la figure 10.a. Un programme avec une boucle infinie ne s'arrête que lorsque le système reçoit un RESET (Voir section architecture externe du 8085).

Position mémoire	**Exemple** le programme utilisé dans le feu rouge de circulation il est continue de gérer les intersections routières
Début	programme

	JMP Début

Exemple : Ecrire un programme par la langage assembleur du microprocesseur 8085 pour exécuter de façon contenue :

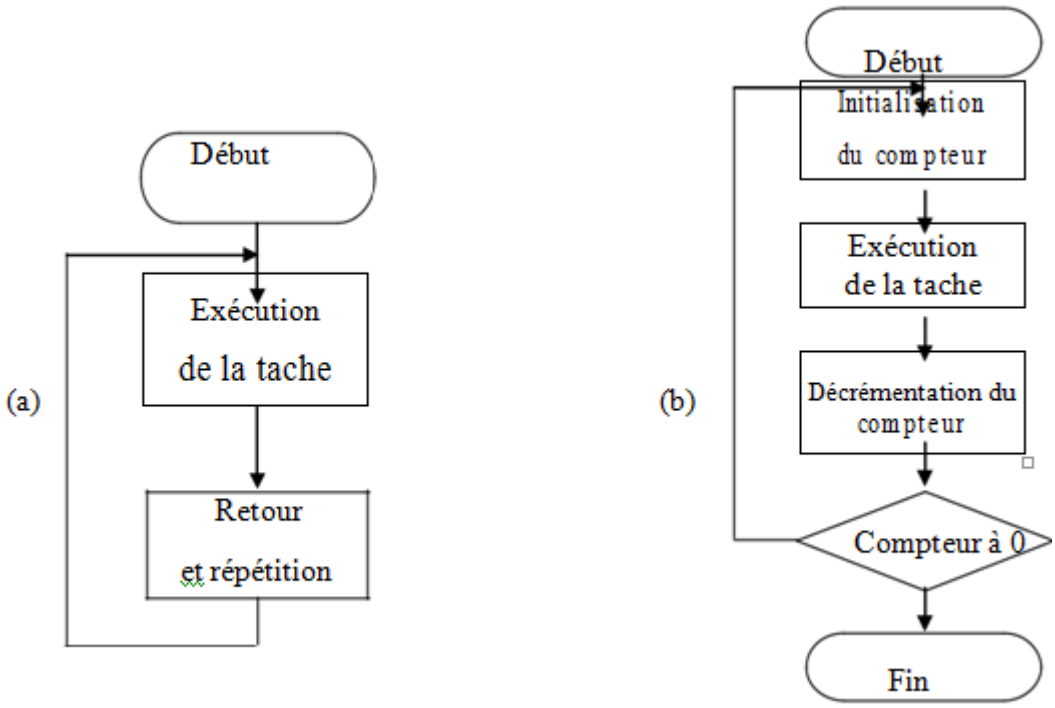
- Lecture du signal (valeur) à partir du port 01
- ajouter 05 à la valeur lue
- Envoie de la valeur obtenu à la sortie (port 02)

Adresse Mémoire		
2000	IN 01h	Lecture de la valeur d'entré et chargement de cette valeur à l'accumulateur
2002	ADI 05h	Addition immédiate de cette valeur à 05h
2004	OUT 02h	Envoie de la valeur obtenue au port 02
2006	JMP 2000	Saut au position mémoire 2000, le début de programme pour continuer l'exécution de programme autre fois

IV.8.2. Boucles conditionnelles

Une boucle conditionnelle est : utilisée pour la répétition avec certaines conditions. Elle est réalisée à base des instructions de branchement. Ces instructions vérifient l'état des flags (Zero, Carry,...) et répètent les taches concernés si les conditions sont satisfaites. Ces boucles utilisent généralement la notion de comptage (figure IV.3 b).

Emplacement memoire	
	Nombre des fois : MVI C,
Debut :	programme
	DCR C
	JNZ Start
	HLT



Exemple : Ecrire un programme qui permet d'additionner les données trouvés dans la position mémoire de 3000H à 300BH Si le résultat de l'addition ne peut aborder 8bits e stocker le résultat		
Adresse mémoire	**Instruction**	
	MVI A 00h	Initialiser l'accumulateur (compteur)
	LXI H 3000h	Adresse de la première position
	MVI C 0Bh	Nombre de répétition= adresse de la dernière position –adresse de la première position (300B-3000=0B)
Début	ADD M	Addition de contenu de la mémoire dans l'accumulateur
	INX H	Incrémentation du registre HL (+1)
	DCR C	Décrémentation du compteur par 1 (-1)
	JNZ **Début**	Saut (début) si non nul (si le résultat est différent de 0 : Z=0) (le compteur C≠0)
	INX H	Incrémentation du registre HL (+1)
	MOV M A	Stocker le resultat avec addition

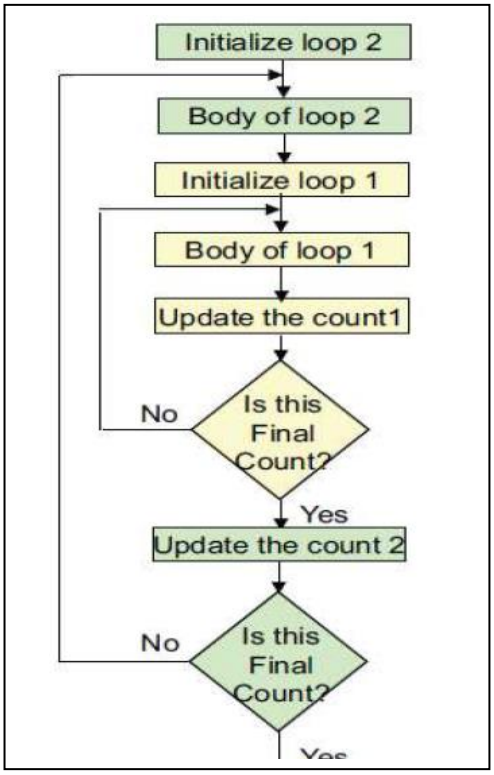
Exemple

Faire une boucle répétitive pour un nombre de fois ne dépasse (65536) (FFFFh) la capacité de registre est 16bits

Position de memoire	
	Les valeurs initiales des registres (initialisation des registres)
	Nombre des fois LXI B,
Début	Programme
	DCX B
	MOV A, B
	ORA C
	JNZ Début
	HLT

IV.8.3. Boucles imbriquées

Position mémoire	
	MVI B, Nombre de répétition de la boucle externe
BOUCLE1	MVI C, nombre de répétition de la boucle interne
BOUCLE2	LE PROGRAMME
	DCR C
	JNZ **BOUCLE1**
	DCR B
	JNZ **BOUCLE2**



REMARQUE : Nombre de répétition total de programme= nombre de répétition de la boucle externe × nombre de répétition de la boucle interne

Exemple 1

Le comptage d'une valeur a une autre de manière croissante

Ecrire un programme de comptage croissant de nombre 04h à 20h et afficher l la valeur sur le port 01h de manière continu		
BOUCLE 2	MVI A 04h	charger l4accumulateur a par la valeur

		04h
BOUCLE 1	OUT 01h	Affichage de la valeur sur le port de sortie 01
	INR A	Incrémentation de l'accumulateur A
	CPI 11h	Comparaison avec la valeur quant veut l'arrêter
	JNZ BOUCLE 1	Si la valeur =11 la valeur de Z=1
	JMP BOUCLE 2	Retour inconditionnel pour que le programme s'exécute de façon continue

Exemple 2

Le comptage d'une valeur a une autre de manière décroissante

Ecrire un programme de comptage croissant de nombre D2h à 20h et afficher l la valeur sur le port 01h de manière continu		
BOUCLE 2	MVI A D2h	charger l'accumulateur a par la valeur D2h
BOUCLE 1	OUT 01h	Affichage de la valeur sur le port de sortie 01
	DEC A	Décrémentation de l'accumulateur A
	CPI 1Fh	Comparaison avec la valeur quant veut l'arrêter
	JNZ BOUCLE 1	Si la valeur =1F la valeur de Z=1
	JMP BOUCLE 2	Retour inconditionnel pour que le programme s'exécute de façon continue

Autres instructions

CMC	Instruction complément de retenue CY	Complémenter la valeur de retenu
STC	Forcer (SET)la valeur de retenu CY=1	Forcer (SET)la valeur de retenu CY=1
NOP	N'a pas d'opération	Le microprocesseur n'a fait aucune action
HLT	Arrêt le programme	

IV.9 Transfert de langage assembleur vers langage machine

1 . Commence par la première instruction (code opération) (jeu d'instruction à la première adresse

2 .si l'instruction de type immédiate, les données après le code directement

3. Si l'instruction a une adresse (adresse de 16 bits)en commence par le poids faible LSB, après le poids fort MSB

4 . si l'instruction a une label : exemple les instructions de saut ou d'appel, on cherche à l'adresse de LABEL (on gère cette situation comme une adresse)

5. on continue avec les autres instructions de même manière

Exemple

Trouver le code machine de ce programme assembleur, le programme commence par l'adresse 3000

Adresse	Langage machine
3000	16
3001	8B
3002	0E
3003	6F
3004	79
3005	82
3006	32
3007	2E
3008	30
3009	C3
300A	02
300B	30

LABEL	MVI D 8Bh MVI C 6Fh MOV A C ADD D STA 302Eh JMP LABEL

On ne peut pas comprendre comment fonctionne un µp sans savoir ce qu'est une mémoire

V.1- Notion de mémoire:

Toutes les données et programme sont stockés dans la mémoire centrale. Les circuits mémoires sont des circuits électroniques pouvant:

- ✓ Enregistrer des informations binaires (écriture).
- ✓ Les conserver d'une façon transitoire ou permanente.
- ✓ Les restituer à la demande (lecture).

Les mémoires se divisent en deux catégories:

- Les mémoires centrales réalisées sur des circuits intégrés.
- Les mémoires périphériques: disquette, CD, ROM, bande amagnétique.

Les mémoires centrales se divisent en deux groupes:

- Les mémoires vives : RAM.
- Les mémoires mortes : ROM.

V.1.1 Les mémoires vives: RAM

RAM: Random Access Memory ou mémoire à accès aléatoire sont des mémoires dans lesquelles on peut lire ou écrire des informations dans n'importe quelle cellule mémoire aléatoirement il suffit d'adresser la cellule concernée.

(Rq: bande magnétique: il faut dérouler toute la bande: accès séquentiel)

Elles sont réalisées avec des bascules.

Les RAM stockent les programmes et données de l'utilisateur, on peut les modifier à tout moment. (SRAM, DRAM, VRAM).

V.1.2. les mémoires mortes: ROM:

ROM: Read Only Memory ou mémoire à lecture seulement.

Les mémoires ROM sont imprimée une fois pour toute par le fabricant, on peut que le lire comme les pages d'un livre.

Elles contiennent les programmes de fonctionnement du µp et les instructions.

Les mémoires ROM conservent l'information si on coupe l'alimentation → non volatile.

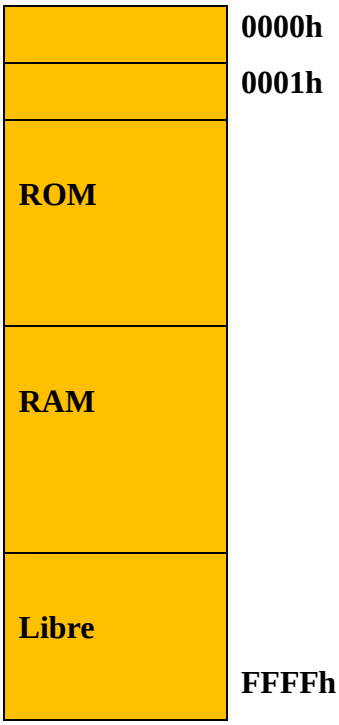
Différents types de ROM
- → PROM: ROM vierge programmable.
- → EPROM: ROM effaçable et reprogrammable.

V.2- organisation des circuits mémoires:

Les circuits mémoires sont réalisés par une association de cellule mémoire est un registre permettant d'emmagasiner des mots binaires sur 8 bits, (16bits, 32bits…)

Un mot de 8 bits est appelé octet ou byte.

Chaque cellules mémoire possède son numéro d'ordre ou adresse avec 16 Bus d'adresse on peut adresser 2^{16} = 65.536 cellules.



La capacité d'un circuit mémoire est exprimée en:

1K bits = 1024 bits (1024 = 2^{10}).

1 KO = 1024 octets

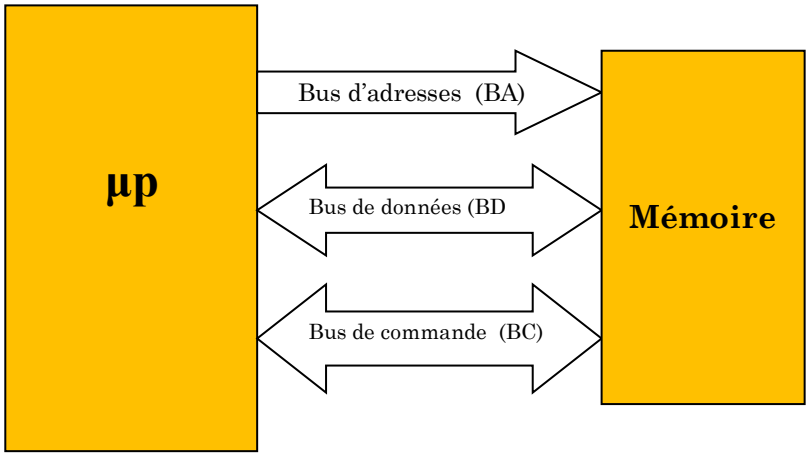
- un groupe de cellules est réservé par LO.RAM.

-un groupe de cellules est réservé par LO.ROM.

la numération est faite en hexadécimal pour raison de commodité de 0000 FFFF (= 64K). ⟶

V.3 Relation entre le µp et la mémoire:

Le µp est mes en relation avec la mémoire grâce aux Bus d'adresses, Bus données, et Bus de commande.



-BA (16 bits) spécifie l'adresse de la cellule mémoire.

- BD (8 bits) transfert les données de et vers le µp.
- BC transmet la commande (lecture/écriture) du µp à la mémoire.

V.4. Procédure de lecture et d'écriture en mémoire:

Le fonctionnement d'un µp est cadencé par une horloge, il faut bien synchroniser les commandes et les adresses pour lire on écrire des données en mémoire.

V.6. Cycle de fonctionnement

Les échanges d'une mémoire avec l'extérieur se font suivant une procédure bien définie donnée par l'état des signaux de gestion de la mémoire. **Exemple** : Cycle de lecture d'une SRAM

Pour lire le contenu d'une mémoire, il faut :

- présenter une adresse sur le bus d'adresse,
- sélectionner le boîtier par l'intermédiaire du « Chip Enable (CE) » ou du « Chip Select (CS) »,
- valider la mémoire en position lecture (R /W en position lecture _ read),
- au bout d'un temps d'accès, les données sont disponibles en sortie sur le bus de données. Les broches de sortie passent de l'état haute impédance à l'état actif

V.4.1. Chronogramme d'écriture en mémoire

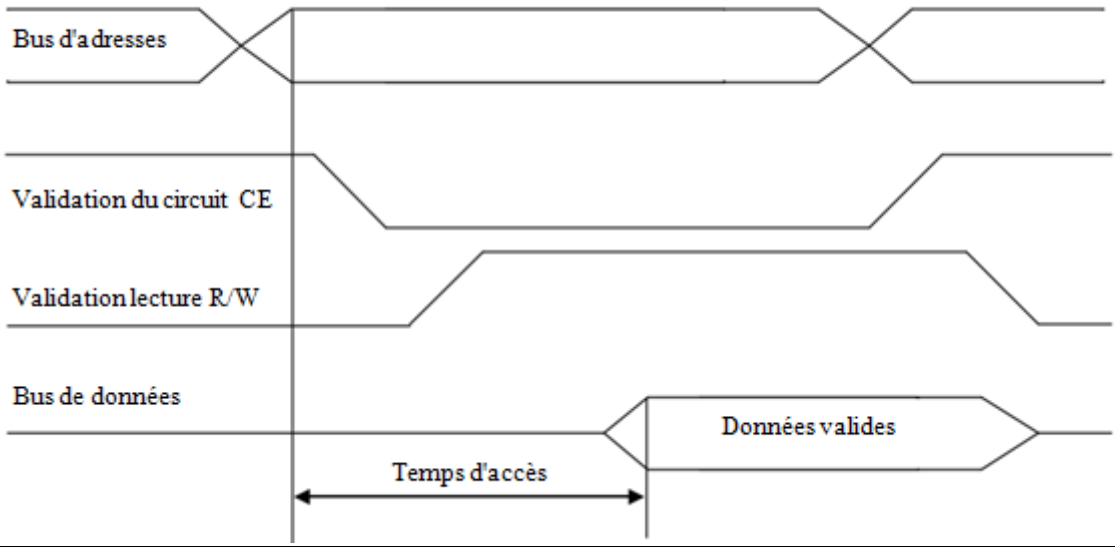


Figure V.1 : Chronogrammes d'accès mémoire

V.5. Connexion de plusieurs boitiers mémoires sur lebus d'un microprocesseur

Les boitiers m´mémoire possèdent une broche notée CS : Chip Select. Lorsque cette broche est active (état bas), le circuit peut étre lu ou écrit. Lorsqu'elle est inactive(état haut), le circuit est exclu du service : ses broches de données D0 à D7 passent `a l'´etat de haute Impédance : tout se passe comme si la m´mémoire ´était d´connectée du bus de données du microprocesseur, d'où la possibilité´e de connecter plusieurs boitiers m´mémoire sur un mêmebus : un seul signal CS doit ˆêtre actif `a un instant donné´e pour ´éviter les conflits entre lesDifférents boitiers.

On peut utiliser une boitier tout seule de 64Ko , cette boitier peut utiliser comme ROM ou RAM, mais cette conception a plusieurs inconvénients

Mémoire N1
0000-7FFF
ROM
Mémoire N2
8000-FFFF
RAM

Nous pouvons diviser la mémoire comme nous souhaitons : deux mémoires de 32ko Mémoires de64kO= 2×32KO (mémoire ROM 0000-7FFF) (RAM 8000-FFFF) Onutilise les lignes d'adresse A_0-A_{14} le dernier ligneA_{15} est utilisé pour choisir le numéro de boitier,

le A_{15} est connecté dans le circuit de CS (chip set enable) c.à.d. un décodeur de 1 vers 2

Si CS=0 le choit de la boitier 2

CS=1 le choix du premier boitier

Mémoire N1
0000-3FFF
Mémoire N2
4000-7FFF
Mémoire N3
8000-BFFF
Mémoire N4
CFFF-FFFF

En peut utiliser la conception de 4×16KO donc dans ce cas on utulise les lignes d'adresse A_0-A_{13}, il reste de lignes A_{14}-A_{15} , ces dernierslignes sont connecté dans un circuit de décodeur pour générer les CS pour choisir le numéro des boitier ; CS_1. CS_2 , CS_3 , CS_4 (un décodeur de 2vers 4la mémoire devienne (voir figure ci-dessus)

Si on utilise mémoire de 64kO= 8×8KO,une case m´mémoire est désignée par les bits d'adresses A_0 à A_{12}, il reste les lignes A13-A15, ces derniers lignes sont connecté dans un circuit de décodeur pour générer les CS (CS_1. CS_2 , CS_3 , CS_4 , CS_5. CS_6 , CS_7 , CS_8) (décodeur de 3vers8), pour choisir le numéro des boitier

Si on utilise mémoire de 64kO= 16×4KO les lignes d'adresse

A_0-A_{11}il reste les lignes A_{11}-A_{15} ces derniers lignes sont connecté

dans un circuit de décodeur pour générer les CS pour choisir le

numéro des boitier ; CS_1. CS_2 , CS_3 , CS_4 , CS_5. CS_6 , CS_7 , CS_8 , CS_9. CS_{10}, CS_{11}, CS_{12} , CS_{13}. CS_{14} , CS_{15} , CS_{16} (décodeur de 4 vers16), même chose pour les précédentes, il est plus simple que ça, chaque chip set select un numéro de mémoire ; Pour cela, il faut utiliser un circuit de décodage d'adresses.

Mémoire N1 0000-1FFF
Mémoire N2 2000-3FFF
Mémoire N3 4000-5FFF
Mémoire N4 6000-7FFF
Mémoire N5 8000-9FFF
Mémoire N6 A000-BFFF
Mémoire N7 C000-DFFF
Mémoire N8 E000- FFFF

V.5. Calcul de la taille et le temps d'exécution d'un programme

V.5. a La taille d'une instruction

La plus part des instructions de microprocesseur 8085 a un octet sauf les instructionsimmédiates qui a deux (code opération + opérande de 8 bits)()ou trois octets code opération + opérande 16 bits (adresse mémoire**)**

Type d'instruction	**Nombre des octets**
La plus part des instructions	1
instructions immédiate	2
adresse mémoire	3

exemple

instruction	**Nombre d'octets**
IN 01	2
MVI A	2
LXI H 3000	3
ACI 30	2
ADD B	1
Call 4010	3
NOP	1
HLT	1

- le Temps d'exécution total d'un programme assembleur est la somme de temps d'exécution de chaque instruction du programme
- Le 8085 a quatre types d'adressage
- Adressage implicite CMP (1 octet)

- Adressage immédiate MVI,45 (2 octets)
- Adressage direct LDA 4000 3 (octets)
- Adressage indirect LDAX B

V.5. b La relation entre le cycle machine et le nombre des impulsions d'horloge

Cycle machine :On ne peut pas connaitre le cycle machine sans connaitre le type et l'dressage de l'instruction lui-même

Le cycle machine comprend deux phases : phase de recherche et phase d'exécution de l'instruction, sans entrer dans les détailles, on aura un cycle machine dans les cas suivants :

- Lecture de code opération
- Lecture d'une adresse mémoire
- Ecriture dans la mémoire
- Lecture d'un port d'entre
- Ecriture d'un port de sortie

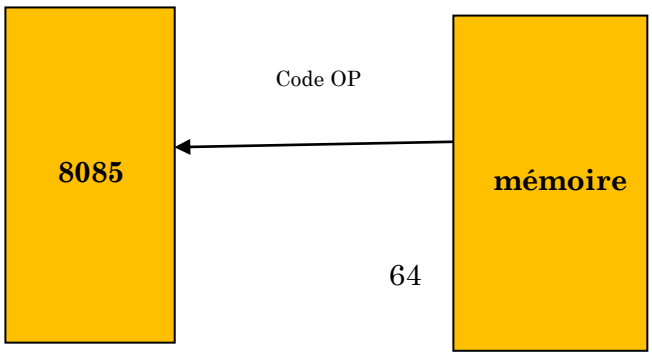
Le premier cycle toujours (lecture de code opération) a besoin 4 impulsions d'horloge (4T) et les autres cycles n'ont besoin qu'un 3 impulsions horloge

Cycle machine	Nombre d'impulsion horloge	
1	4T	Le temps minimal d'exécution
2	7T	
3	10T	
4	13T	
5	16T	Le temps maximal d'exécution

Remarque : Le nombre des impulsions horloge nécessaire est =(4T de cycle de code opération+ Nombre des autres cycles×3T)

Exemple

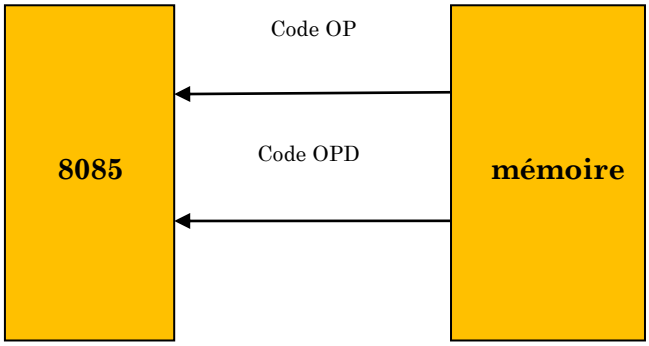
MOV A, B (transférer er le contenu de B vers l'accumulateur) cette instruction a un octet (le code machine est 78) elle permet lire le code opération 78) on a dans le cas (lecture du code opération)



Elle n'a besoin qu'un cycle machine c'à d 4 impulsions d'horloge(lecture de code opération de mémoire vers le microprocesseur

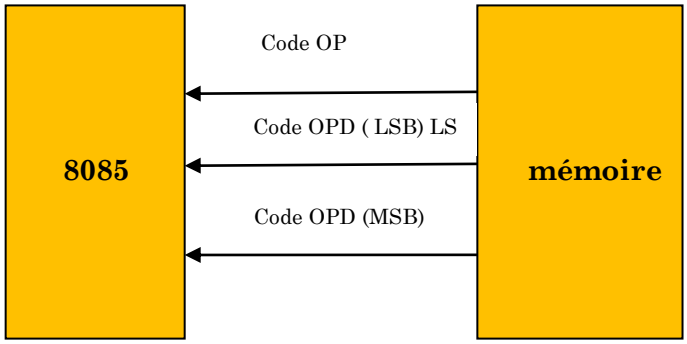
Exemple MVI A, 45H

Elle a deux octets, et deux cycles machine, l'un pour le code opération et l'autre pour la lecture code opérande(lire de mémoire)

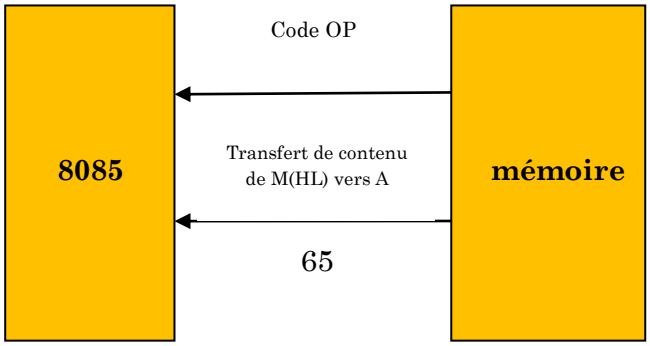


L'instruction

LXI H, F145H a trois cycle machines (l'un pour le code opération(21) , deuxième pour le LSB (45) du code opérande, le dernier pour MSB(F1)

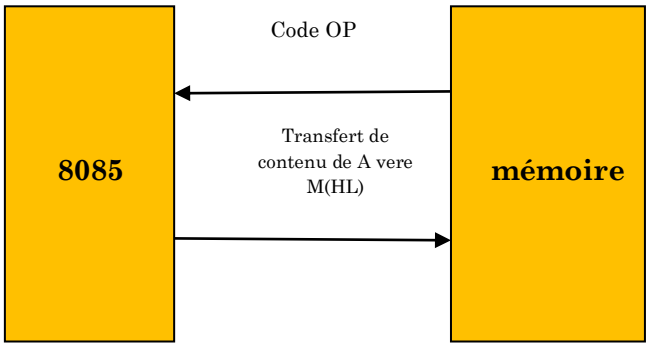


L'instruction MOV A,M cette instruction a un octet, mais elle a deux cycles machines, le premier cycle de lire le code opérations la deuxième elle connaitre l'emplacement mémoire spécifie par le regisre HL après elle transfert le contenu à l'accumulateur (lecture de mémoire) ,

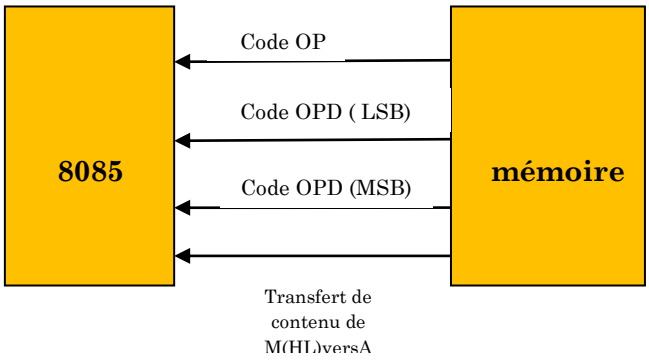


Remarque : Le nombre des impulsions horloge nécessaire est =(4T de cycle de code opération+ Nombre des autres cycles×3T)

MOV M,A elle transfert le contenu d'accumulateur vers la mémoire , elle a un octet mais deux cycle machine



LDA 3000H : Elle a 4 cycle machine (un pour la lecture du code opération, deuxième pour le code opérande LSB(00) , troisième pour la lecture du code opérande MSB(30) et le dernier pour charger le contenu de l4adresse mémoire 3000 à l'accumulateur



Exemple

Instruction	**Nombre d'octets**	**Nombre des cycles**	**Nombre d'impulsion T**
ACI data (8b)	2	2	7
ADC R	1	1	4
ADI data (8b)	2	2	7
ADD M	1	2	7
ANA M	1	2	7
Call address (16b)	3	5	16
CMC	1	1	4
DAD Rp	1	3	10
MVI M, data(8b)	2	3	10
LHLD address (16b)	3	5	16
JC address (16b)	3	2	7
JNC address (16b)	3	3	10
INR M	1	3	10
IN Port address(8b)	2	3	10
RAL	1	1	4

Remarque : vous trouvez le détail de toutes les instruction dans l'annexe de ce polycopie

Exemple :

soit le programme suivant

MVI D,7A

MVI C,2F

MOV A C

ADD D

STA 202F

- Calculer la taille de chaque instruction et la taille de programme complet
- Calculer le temps d'exécution de programme si la fréquence de l'horloge de micro processeur(cristal) est 1Mhz

  Corrigé

Language assembleur	Nombre des octets
MVI D,7A	2
MVI C,2F	2
MOV A C	1
ADD D	1
STA 202F	3
TOTAL	9

Le temps d'exécution nécessaire :

Langage assembleur	**Cycle machine**	**Impulsion d'horloge**
MVI D,7A	2	7
MVI C,2F	2	7
MOV A C	1	4
ADD D	1	4
STA 202F	4	13
	10	35

Le temps d'exécution total est : T= impulsion d'horloge / fréquence = 35 /1Mhz=35µs

Exemple

	MVI C, FF	7 T(impultion d'horloge)
LOOP**:**	DCR C	4 T
	JNZ LOOP	10 T

Tdelay= T0+TL

T0= hors boucle

TL: l'interieur du boucle

Tdelay= T total

T0=7T

TL=(14T×255)-3T=3567T

14T pour les deux instructions repetés255 fois (FF=255)mins 3Tpour la valeur final JNZ(la condition satisfie)

Tdelay= T0+TL=3567T+7= 3574Tdelay

Temps d'execution total = Tdelay/1Mhz=3,574µs

Un microcontrôleur est un circuit intégré rassemblant dans un même boitier un microprocesseur, plusieurs types de mémoires et des périphériques de communication (Entrées-Sorties).

Selon un arrêté français du 14 septembre 1990 relatif à la terminologie des composants électroniques : " Circuit intégré comprenant essentiellement un microprocesseur, ses mémoires, et des éléments personnalisés selon l'application ".

Le Motorola 68HC11, ici en boîtier PLCC, est un microcontrôleur réputé

VI.1 Structure d'un système à microprocesseur

- Un microprocesseur a besoin de certains éléments pour fonctionner :
- de la mémoire morte dite ROM (principalement pour stocker le programme) ;
- de la mémoire vive dite RAM (principalement pour stocker les variables) ;
- des périphériques (principalement pour interagir avec le monde extérieur).
- une horloge pour le cadencer (principalement à quartz)

Ces différents blocs sont reliés par 3 bus :

- le bus d'adresse qui permet au microprocesseur de sélectionner la case mémoire ou le périphérique auquel il veut accéder pour lire ou écrire une information (instruction ou donnée) ;

- le bus de données qui permet le transfert des informations entre les différents blocs ; ces informations seront soit des instructions soit des données en provenance ou à destination de la mémoire ou des périphériques ;
- le bus de contrôle qui indique si l'opération en cours est une lecture ou une écriture, si un périphérique demande une interruption etc.

Le fonctionnement est le suivant :

- À la mise en route du système, le microprocesseur va chercher dans la mémoire à l'adresse 0 (pour la plupart des processeurs) la première instruction à exécuter ;
- il stocke cette instruction dans un registre interne appelé registre d'instructions ;
- il execute cette instruction ;
- puis en consultant le registre pointeur d'instruction va chercher l'instruction suivante, etc.

Traditionnellement, ces composants sont intégrés dans des circuits distincts. Le développement d'un tel système à base de microprocesseur se trouve donc pénalisé par (liste non exhaustive) :

- La nécessité de prévoir l'interconnexion de ces composants (bus) ;
- la place occupée physiquement par les composants et les moyens d'interconnexion ;
- la consommation énergétique ;
- la chaleur dégagée ;
- le coût financier.

Les microcontrôleurs améliorent l'intégration et le coût (lié à la conception et à la réalisation) d'un système à base de microprocesseur en rassemblant ces éléments essentiels dans un seul circuit intégré. On parle alors de " système sur une puce " (en anglais : " *System on Chip* "). Dans la suite, nous nous intéressons plus particulièrement aux microcontrôleurs 8 bits, c'est-à-dire ceux dont le bus de données comportent 8 bits et le bus d'adresses 16 bits.

VI.2 Périphériques

Les périphériques sont des circuits électroniques intégrés au microcontrôleur capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :

- les convertisseurs Analogique/Numérique (donnent un nombre binaire à partir d'une tension électrique) ;
- les convertisseurs Numérique/Analogique (opération inverse) ;
- les générateurs de signaux à modulation de largeur d'impulsion (MLI, ou en Anglais, PWM pour *Pulse Width Modulation*) ;
- les timers (compteurs de temps ou d'événements) ;
- les comparateurs (comparent deux tensions électriques) ;
- les contrôleurs de bus (UART, IIC, SSP) ;
- les oscillateurs (servent de base de temps aux timers).

Le fonctionnement des périphériques peut être paramétré et commandé par le programme et/ou les entrées-sorties. Les périphériques peuvent générer une interruption qui contraint le processeur à quitter le programme principal pour effectuer une routine d'interruption.

VI.3 Jeu d'instructions

On peut classer les instructions qu'un microcontrôleur est capable d'effectuer en quelques groupes.

VI.3.1 Instructions de transfert

Le microcontrôleur passe une grande partie de son temps à transférer des octets d'un endroit à l'autre du système : d'un périphérique vers un registre interne ou vice-versa, d'un registre interne vers la mémoire RAM ou vice-versa. Ce qui ne peut en général pas être fait directement, c'est un transfert direct d'une case mémoire vers une autre ou vers un périphérique, ou une écriture en mémoire ROM, la structure du microcontrôleur rend obligatoire le passage des informations par un de ses registres internes. Remarquons que, sauf exceptions, il s'agit plutôt d'une copie que d'un transfert puisque la case mémoire d'origine garde son information (tant qu'on n'a pas écrit autre chose à la place).

VI.3.2 Instructions arithmétiques

Un microcontrôleur 8 bits n'est pas un grand mathématicien. Tout au plus est-il capable d'effectuer des additions, des soustractions, des multiplications et des divisions sur des nombres binaires de 8 bits. Toutes les opérations mathématiques complexes telles que le traitement des grands nombres, des nombres fractionnaires, des puissances, des racines

carrées, des fonctions trigonométriques, logarithmiques et exponentielles doivent être ramenées à une succession d'opérations simples portant seulement sur des octets. Des routines mathématiques (petits programmes permettant de réaliser les calculs complexes) ont été développées pour la plupart des microcontrôleurs populaires.

VI.3.4 Instructions logiques

Les microcontrôleurs sont capables d'effectuer des opérations logiques : ET, OU, XOU (*XOR*), NON (inverseur), rotations, décalages. Les opérations sont opérées simultanément sur les bits correspondant des deux registres.

La comparaison des octets A et B, qui est considérée comme une opération logique, est réalisée comme une soustraction dont on néglige le résultat ; on s'intéresse simplement au fait de savoir s'il est nul (ce qui signifie que A = B), positif (A > B) ou négatif (A < B). Ces indications sont inscrites dans des indicateurs d'états (petites mémoires d'1 bit situées dans le processeur).

VI.3.5 Instructions d'entrées/sorties

Ces instructions sont utilisées pour :

- lire l'état d'un port d'entrée (permettant l'interfaçage d'interrupteurs, de commutateurs, d'optocoupleurs, de convertisseurs analogiques/numériques, de claviers, etc.) ;
- écrire une information dans le registre d'un port de sortie, qui maintient l'information à la disposition des circuits extérieurs (leds, moteurs, relais, convertisseurs numériques/analogiques, etc.) ;
- écrire ou lire une information dans les registres d'un port série.

Signalons que dans certains microcontrôleurs les périphériques sont considérés simplement comme des cases de mémoire et ils sont gérés par les instructions de transfert (**entrées/sorties intégrées mémoire**). D'autres microcontrôleurs disposent d'instructions spécifiques pour les entrées/sorties (**entrées/sorties indépendantes**).

VI.3.6 Instructions de branchement

Il s'agit d'instructions qui altèrent le déroulement normal du programme. On distingue les sauts et les sous-routines.

- Les **sauts** provoquent un branchement du programme vers une adresse mémoire qui n'est pas contigüe à l'endroit où l'on se trouve.

- La **sous-routine** ou sous-programme est une partie de programme dont on a besoin à plusieurs endroits dans l'exécution du programme principal. Plutôt que de répéter ce sous-programme à tous les endroits où l'on en a besoin, on le place en un endroit donné (par exemple à la fin du programme principal) et on opère un branchement du programme principal vers le sous-programme chaque fois que nécessaire. La grande différence par rapport au saut, c'est qu'au moment du branchement il faut mémoriser l'adresse d'où l'on vient, afin de pouvoir y revenir une fois le sous-programme terminé. Ceci est effectué en mémorisant l'adresse de départ dans un registre ad hoc (la pile) du microcontrôleur.

  Tant les sauts que les sous-routines peuvent être :

- inconditionnels ;
- conditionnels, c'est-à-dire que le branchement n'a lieu que si une certaine condition est remplie ; généralement, la condition testée est le contenu d'un des indicateurs d'état ; ceux-ci indiquent par exemple si le contenu de l'accumulateur est nul, positif, négatif, de parité paire ou impaires.

VI.3.7 Instructions diverses

On trouve dans ce groupe :

- des instructions de gestion de la pile (zone de mémoire RAM permettant le stockage de données pendant l'exécution du programme) ;
- des instructions de contrôle du processeur : par exemple passage en mode basse consommation, contrôle des périphériques embarqués (càd. sur la même puce que le processeur) ;
- des instructions permettant de positionner des indicateurs internes du processeur.

Ces instructions varient fort selon les familles des microcontrôleurs.

VI.4 .Modes d'adressage pour les données

De nombreuses instructions font référence à des données se trouvant à différents endroits du microcontrôleur : registres internes du processeur, ROM, RAM, ports d'E/S. On appelle

modes d'adressage les différentes façons de spécifier les endroits où se trouvent les données dont on a besoin.

VI.4.1 Adressage implicite

Certaines opérations ne peuvent être réalisées que sur une donnée se trouvant en un endroit bien précis du processeur (par exemple, l'accumulateur ou la pile). Dans ce cas, il n'est pas nécessaire de spécifier l'adresse du registre en question et on parle d'adressage implicite.

VI.4.2 Adressage registre ou inhérent

Le processeur dispose d'un certain nombre de registres de travail. De nombreuses instructions y font référence ; vu leur nombre peu élevé (8, par exemple), il suffit d'un petit nombre de bits pour spécifier le registre désiré (3 dans notre cas). On parle dans ce cas d'adressage registre ou inhérent.

VI.4.3 Adressage direct

Dans ce mode d'adressage, on donne l'adresse (généralement en 16 bits) de la donnée en mémoire (RAM, ROM ou port d'E/S s'il est intégré à la mémoire). Ce mode d'adressage permet d'indiquer n'importe quel endroit dans la mémoire, le prix à payer étant que l'on doit spécifier l'adresse concernée dans son intégralité (2 à 4 octets).

Certains processeurs implémentent aussi, pour réduire l'encombrement du programme, l'adressage direct restreint : l'adresse ne comporte qu'un octet, et ce mode ne permet d'accéder qu'aux données se trouvant sur la page courante (adresses ayant donc les mêmes bits de poids fort) ou sur la page 0 (adresses ayant donc les bits de poids fort à 0).

VI.4.4 Adressage indirect à registre

Dans ce mode d'adressage, l'adresse de la donnée se trouve dans un registre spécial du processeur (du même nombre de bits que son bus d'adresses), le pointeur de données. L'avantage, par rapport à l'adressage direct, est que l'adresse peut être manipulée commodément, par exemple pour accéder à une suite de données consécutives en mémoire. Ceci est particulièrement utile lorsqu'on manipule des données stockées dans un tableau.

VI.4.5 Adressage immédiat

C'est un peu un abus de langage que de parler d'adressage dans ce cas-ci. En effet, la donnée suit tout simplement l'instruction.

VI.4.6 Adressage indexé

Ce mode est assez semblable à l'adressage indirect à registre. Il fait appel à un registre spécial appelé " registre d'index ". Certains microcontrôleurs ne supportent pas ce mode, d'autres au contraire ont 1 ou même 2 registres d'index. Deux registres d'index sont particulièrement bienvenus lorsqu'il s'agit de déplacer un bloc de données dans la mémoire RAM.

VI.5 Structure d'une instruction

Une instruction comporte généralement de 1 à 3 octets. Le premier octet est appelé code opératoire, il spécifie ce que l'on veut faire.

Le ou les octets suivants spécifient la donnée (adressage immédiat), une adresse restreinte, une adresse absolue (16 bits) ou une donnée 16 bits (peu utilisé).

Une instruction traite au maximum deux données, si l'on se limite à 3 octets pour l'instruction, une seule des données peut être en RAM.

VI.6 Modes d'adressage pour les branchements

L'adresse de la prochaine instruction à exécuter est contenue dans un registre spécial du processeur, appelé pointeur de programme ou compteur de programme.

- Adressage direct : dans ce cas, l'adresse où l'on doit aller est donnée en 16 bits ; on peut donc se rendre n'importe où dans la mémoire (programme). Au moment du branchement, le contenu du pointeur de programme est remplacé par l'adresse en question (si le branchement concerne une sous-programme, on sauvegarde le contenu du pointeur de programme).

  Comme de nombreux branchements s'effectuent vers des adresses mémoire proches de l'endroit où l'on se trouve au moment d'exécuter le branchement, on a prévu deux modes d'adressage plus compacts, ne nécessitant qu'un octet pour l'adresse :

- adressage relatif : on spécifie le nombre de pas à effectuer en avant, ou plus souvent en arrière, dans le programme ;

- adressage restreint : on spécifie l'adresse sur la page courante (octet d'adresse le plus significatif inchangé).

VI.7 Familles de microcontrôleurs

- la famille Atmel AT91 ;
- la famille Atmel AVR ;
- le C167 de Siemens/Infineon ;
- la famille Hitachi H8 ;
- la famille Intel 8051, qui ne cesse de grandir ; de plus, certains processeurs récents utilisent un cœur 8051, qui est complété par divers périphériques (ports d'E/S, compteurs/temporisateurs, convertisseurs A/N et N/A, chien de garde, superviseur de tension, etc.) ;
- l'Intel 8085, à l'origine conçu pour être un microprocesseur, a en pratique souvent été utilisé en tant que microcontrôleur ;
- le Motorola 68HC11 ;
- la famille Freescale 68HC08
- la famille Freescale 68HC12
- la famille des PIC de Microchip ;
- la famille des dsPIC de Microchip ;
- la famille des ST6 de STMicroelectronics ;
- la famille ADuC d'Analog Devices ;
- la famille PICBASIC de Comfile Technology;
- la famille MSP430 de Texas Instruments.
- la famille 8080 , z80 , Rabbit : pour memoire , le 8080 est un des grands ancetres , mais z80 et Rabbit sont encore bien vivants
- la famille PSoC de Cypress
- la famille LPC21xx ARM7-TDMI de Philips
- la famille V800 de NEC
- la famille K0 de NEC

VI.8 Applications

Les microcontrôleurs sont souvent utilisés dans l'élaboration de systèmes embarqués, nécessitant des traitements spécialisés (autoradios, téléphones portables, lecteur mp3, GPS, etc.).

Ces circuits intégrés sont également très prisés en robotique amateur et permettent de réaliser de nombreuses applications, y compris des robots autonomes, les automatismes en modélisme (maquettes de réseau ferroviaire).

VI.9 Programmation

Le langage C est le langage de prédilection (avec l'assembleur) du développement sur microcontrôleurs. Une fois le programme compilé, le fichier binaire doit être envoyé au microcontrôleur. On utilise soit :

- un programmateur de microcontrôleurs et souvent également d'EEPROM, on parle alors de programmateur universel.
- un programmateur ISP qui a l'avantage de ne pas nécessiter de sortir le microcontôleur du système électronique complet.

On peut alors utiliser le système. Toutefois, le programme qui a été envoyé peut comporter des bogues, aussi, pour parvenir à les détecter on utilisera par exemple un émulateur in-circuit.

SERIES DES EXERCICES

Exercice1

Dites si cette instruction est possible ou non. Si elle est impossible, expliquez pourquoi.
MOV H,L : OUI / NON

_____

MOV D, 23h : OUI / NON

_____

MOV E, F : OUI / NON

_____

MVI C,D : OUI / NON

_____

MVI H,99h : OUI / NON

_____

MVI B,100h: OUI / NON

_____

Exercice2

Dites si ces instructions sont possibles ou pas. Expliquez pourquoi.

1. ADI 2FFh : OUI / NON
   _____
2. ORA Ch : OUI / NON
   _____
3. ORA C : OUI / NON
   _____
4. ANA FFh : OUI / NON
   _____
5. ORB D : OUI / NON
   _____
6. ORI 00h : OUI / NON
   _____
7. ADD B,C : OUI / NON

_____

Exercice 3

Effectuer les opérations Suivantes en écrivant un programme assembleurs de chaque opération (Status de Bits S,Z,C de registre d'état)

- 4H+ 3H (ADD),
- FFH+01H (ADI) ,
- FFH-03h (SUI)

Exercise 4:

Expliquer les programmes suivants en décrivant le résultat et le statut des 3bits de registre d'état (S,Z,C) et l'accumulateur (A)

1. MVI A,05H
SUI 02H

2. MVI A,01H
DCR A

3. LXI SP,2000H
INX SP

4. LXI B,300
DCX B

5. MVI A,30H
MVI B,F0H
ADD B
ADC B

6. MVI A,30H
MVI B,F0H
ADD B
ACI 30H

7. MVI A,30H
MVI B,20H
ADI FFH
SBB B

8. MVI A,30H
ADI FFH
SBI 20H

Exercice 5 : Ecrire un programme assembleur qui permet d'effectuer de stocker la donnée 35 dans l'emplacement mémoire 8024 par différentes méthodes

Exercice 6 : Ecrire un programme assembleur qui permet d'effectuer l'addition des nombres 12F4H et10D5H et stocker le résultat dans le registre BC

Exercice 7 : Ecrire un programmeen assembleur qui permet d'effectuer la soustraction des nombres 2426H et1065H et stocker le résultat dans le registre BC

Exercice 8 : Ecrire un programme n assembleur qui permet d'effectuer l'addition des nombres 1234H et A251H et stocker le résultat dans l'adresse mémoire 3000H et 3001H

Exercice 9 : Ecrire un programme n assembleur qui permet d'effectuer la multiplication 9×12F5 et stocker le résultat dans l'adressemémoire 3000H et 3001H

Exercice 10 : Effectuer un and logique entre l'accumulateur et le registre, Expliquer les programmes suivant en décrivant le résultat et le statut des 3bits de registre d'état (S,Z,C) et l'état de l'accumulateur.

- 81Hand 77H (Syntaxe **:** ANA R ou ANA M)
  ANI : And Immédiate : And logique entre le contenu de l'accumulateur A et une valeur 8 bits.
- MVI A, 55H

ANI 01H

- MVI A,55H

CMA

- **ORA : Or Syntaxe : ORA R ou ORA M**
  Tel que R : Registre 8 bits et M : adresse d'une case mémoire
  EXEMPLE
  MVI A,81H
  MVI B,7EH
  ORA B

Exercice 11 :

Soit le programme suivant. Donnez l'état de chaque registre **après** l'exécution de l'instruction. On suppose que la première instruction est à l'adresse 0000h.

Programme en ROM	PC	A	B
MVI A,07h			
MVI B, 23h			
ADD B			
ADI 3h			
ANI 0h			

Pour connaître la valeur du PC après une instruction, il faut ajouter à la valeur du PC le nombre d'octet de l'instruction. Si le PC vaut 54h, après une instruction de 2 octets il vaudra 56h.

Exercice 12 :

Concevez un code qui écrit les valeurs 77h dans B et 88h dans C, puis qui fait l'addition de ces deux registres et qui stocke le résultat dans D.
B = 77h, C = 88h,
B + C --> D

Programme en ROM	PC	A	B	C	D

Exercice 13:

Soit le programme suivant , complétez le tableau suivant:

Programme en ROM	PC	A	Z	C	P
MVI A FEh					
MVI B 3h					
ADD B					
DCR A					
DCR A					
INC A					
INC A					
SUI 01					
SUI 01					
ADI 01					
ADI 01					
SUI 34h					
SUB A					

Exercice 14 :

1. Ecrire un programme assembleur (Microprocesseur 8085) pour trouver la sortie de l'opération logique suivante :

$Y = (A + B).\overline{(C \oplus B)}$, Si : A=C4 h, B=32 h , C=F9 h, D=E2 h

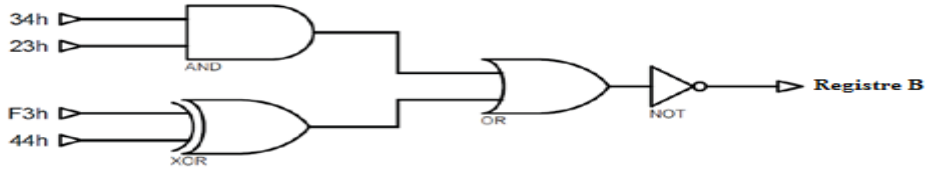
2. Complétez le tableau suivant :

Programme assembleur	Resultat
MVI A , C4H	A=C4H

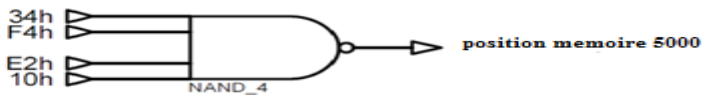
Exercice 15 :

Ecrire un programme assembleur (Microprocesseur 8085) pour trouver la sortie de l'opération logique suivante :

A)



B)



Exercice 16 :

Ecrire un programme de comptage croissant de nombre 0000h à 1000h et afficher la valeur sur les deux ports 00h et 01h

Ecrire un programme de comptage croissant de nombre 0000h à 1000h et afficher la valeur sur les deux ports 00h et 01h			
	LXI H 0000h	Chargement de la valeur initiale de registre HL (deux Registres H et L)	
	LXI B 1001h	Chargement de la valeur qui suit valeur finale	Pour l'affichage
BOUCLE	MOV A H	Copie de valeur de H à l'accumulateur	
	OUT 00h	Affichage Du Valeur Sur Le Port 01	
	MOV A L	Copie de valeur de H à l'accumulateur	
	OUT 01h	Affichage Du Valeur Sur Le Port 00	
	INX H	Incrémentation du registre HL (+1)	
	DCX B	Décrémentation du compteur B par 1 (-1)	

	MOV A B	transfert de la valeur de registre b a l'accumulateur	Pour la vérification que le registre BC=0000h
	ORA C	Faire un OR logique entre le registre C et l'accumulateur A	
	JNZ LOOP	Saur si le résultat non nul	
	HLT		

Exercice 17 :

Ecrire un programme qui permet d'effectuer la multiplication des contenus de deux cases mémoires (l'emplacement 2200h et 2201h et stocker le résultat à partir de l'adresse 2300h

Ecrire un programme qui permet d'effectuer la multiplication des contenus de deux cases mémoires (l'emplacement 2200h et 2201h et stocker le résultat à partir de l'adresse 2300ha		
Adresse mémoire	**Instruction**	
	LDA 2200H	Chargement de la première valeur
	MOV E, A	Stockage dans le registre E
	MVI D, 00	Initialisation de registre D
	LDA 2201H	Chargement de la première valeur
	MOV C, A	transfert de la deuxième valeur au registre C
	LX I H, 0000 H	Initialisation de registre HL
BACK:	DAD D	Addition de la première valeur
	DCR C	Décrémentation du compteur
	JNZ BACK	Retour au début de programme si la condition ne satisfie pas (non nul)
	SHLD 2300H	Stockage des résultats dans les deux positions mémoires 2300h et 2301

Exercice 18 :

Complétez le tableau suivant:

Programme en ROM	**PC**	**A**	**B**	**C**	**Flag Z**	**Flag S**
MVI A,50h						
MVI B, 30h						
MVI C,70h						
CMP B						
CMP C						
CMP A						
CPI 41h						
CPI 50h						
SUB B						
SUB C						

Exercice 19 : Le début de la mémoire contient les données suivantes :

3EFE060180E611E600

Trouver le code assembleur

Exercice 20 :

Trouver le code machine de ce programme assembleur, le programme commence par l'adresse 3000

LABEL	MVI D 8Bh MVI C 6Fh MOV A C ADD D STA 302Eh JMP LABEL

Exercice 21 :

Soit le programme en ROM suivant:

Remplissez le tableau suivant

Programme en ROM	**PC**	**A**	**C**
MVI A, AAh			
RAL			
RAL			
RAL			
RLC			
RLC			
RLC			
RRC			
RRC			
RRC			
RAR			
RAR			
RAR			

Exercice 22 :

Soit le programme suivant:

0000h MVI A,00h

0002h MVI B,01h

0004h ADD B

0005h JMP 000Bh

0008h MVI A,0h

Adresse Mémoire	instruction	PC	A	B

000Ah INR A

000Bh JMP 0004h
Remplissez le tableau suivant

en faisant attention particulièrement

à la prochaine instruction à exécuter.

Exercice 23

Soit le programme suivant:

0000h MVI A, 10h

0002h MVI C, 4h

0004h ADD A

0005h DCR C

0006h JNZ 0004h

Remplissez ce tableau

en faisant attention particulièrement

à la prochaine instruction à exécuter

0009h NOP

Remplissez le tableau suivant

Label	Instruction	PC	A	C	Z

Corrigés des exercices

Corrigé de l'exercice1 :

- MOV H,L : OUI
- MOV D,23h : NON, il faut utiliser l'instruction MVI
- MOV E, F : OUI
- MVI C,D : NON , il faut utiliser l'instruction MOV
- MVI H,99h : OUI
- MVI B,100h: NON, l'opérande 100h se code sur 9 bits

Corrigé de l'exercice 2 :

ADI 2FFh : NON, l'opérande est sur plus de 8 bits

ORA Ch : NON, Ch est une valeur hexadécimale, alors qu'on attend un nom de registre

ORA C : OUI

ANA FFh : NON, ANI FFh est correct

ORB D : NON, l'instruction ORB n'existe pas, seul ORA existe

ORI 00h : OUI

ADD B,C : NON, ADD n'existe pas, il n'est pas possible de faire une addition de 2 registres si on n'utilise pas le registre A.

Corrigé de l'exercice 3 :

MVI A,04H

MVI B,03H

ADD B

Ce qui donne (A)=(A)+(B)=04H+03H=07H S=0 Z=0 CY=0

MVI A,FFH

ADI 01H

Ce qui donne (A)=(A)+01H=FFH+01H=00H S=0 Z=1 CY=1

Exemple :

MVI A,FFH

MVI B,03H

SUB B

Ce qui donne (A)=(A)-(B)=FFH-03H=FCH S=1 Z=0 CY=0

Corrigé de l'exercice 4 :

Exemple 1 :

MVI A,05H

SUI 02H

Ce qui donne (A)=(A)-02H=05H-02H=03H S=0 Z=0 CY=0

Exemple2 :

MVI A,05H

INR A

Ce qui donne (A)=(A)+1=05H-1=06H S=0 Z=0

Exemple 3 :

MVI A,01H

DCR A

Ce qui donne (A)=(A)-1=01H-1=00H S=0 Z=1

Exemple 4:

LXI SP,2000H

INX SP

Ce qui donne (SP)=(SP)+1=2000H+1=2001H S=0 Z=1

Exemple 5 :

LXI B,3000H

DCX B

Ce qui donne (B)=(B)-1=3000H-1=2FFFH S=0 Z=0

Exemple 6 :

```
MVI A,30H
MVI B,F0H
ADD B          ; (A)=30H+F0H=20H et CY=1
ADC B          ; (A)=20H+F0H+1=11H
```

ACI : Add immediate 8 bits data with carry : Additionner une donnée 8 bits avec A et CY
Syntaxe : ACI donnée 8 bits

Exemple 7 :

```
MVI A,30H
MVI B,F0H
ADD B          ; (A)=30H+F0H=20H et CY=1
ACI 30H        ; (A)=20H+30H+1=51H
```

SBB : Subtract register contents with borrow : Soustraire registre 8 bits ou case mémoireet report CY de A.
Syntaxe : SBB R ou SBB M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

```
MVI A,30H
MVI B,20H
ADI FFH        ; (A)=30H+FFH=2FH et CY=1
SBB B          ; (A)=2FH-20-1=0EH
```

SBI : Subtract immediate 8 bits data with borrow : Soustraire donnée 8 bits et report CYde A.
Syntaxe : SBI donnée 8 bits

Exemple 8 :

```
MVI A,30H
ADI FFH        ; (A)=30H+FFH=2FH et CY=1
SBI 20H        ; (A)=2FH-20-1=0EH
```

Corrigé de l'exercice 5 :

Le stockage de la donnée 35 dans l'emplacement mémoire 8024

On a quatre methodes

Methode1	**Methode2**	**Methode3**	**Methode4**
LXI H 8034h MVI M 35h	LXI H 8034h MVI A 35h MOV M A	LXI D 8034h MVI A 35h STAX D	MVI A 35h STA 8034h

Corrigé de l'exercice 6 :

L'addition des nombres 12F4H et10D5H et stocker le résultat dans le registre BC

Methode 1	**Methode2**
LXI H 1254h LXI D 10D5h MOV A L ADD E MOV C A MOV A H ADC D MOV B A	MVI A 54h ADI D5h MOV C A MVI A 12h ACI 10h MOV B A

Corrigé de l'exercice 7 :

la soustraction des nombres 2426H et1065H et stocker le resultat dans le registre BC

Methode1	**Methode2**
LXI H 2426h LXI D 1065h MOV A L SUB E MOV C A MOV A H SBI D MOV B A	MVI A 26h SUI 65h MOV C A MVI A 10h SBI 10h MOV B A

Corrigé de l'exercice 8 :

l'addition des nombres 1234H et A251H et stocker le résultat dans l'adresse mémoire 3000H et 3001H

LXI H 1234h
LXI D A251h
DAD D
SHLD 3000h

Corrigé de l'exercice 9 :

La multiplication 9×12F5 et stocker le résultat dans l'adresse mémoire 3000H et 3001H

```
LXI H 0000h
LXI D 12F5h
DAD D
DAD H
DAD H
DAD H
DAD D
SHLD 3000h
```

Corrigé de l'exercice 10 :

ANA : And : And logique entre l'accumulateur A et le contenu du registre ou case mémoire.

Syntaxe : ANA R ou ANA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

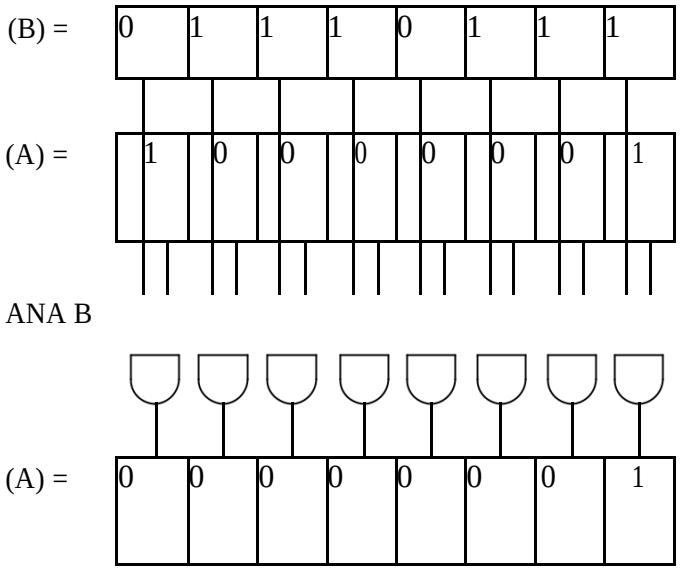
Exemple1 :

MVI A,81H

MVI B,77H

ANA B

Ce qui donne (A)=01H. S=0 Z=0 P=0



2. ANI : And Immédiate : And logique entre le contenu de l'accumulateur A et une valeur

8 bits.

Syntaxe : ANI donnée 8 bits

Exemple 2:

MVI A,55H

ANI 01H

Ce qui donne (A)=54H. S=0 Z=0 P=0

3. ORA : Or : Or logique entre l'accumulateur A et le contenu du registre ou case mémoire.

Syntaxe : ORA R ou ORA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple 3 :

MVI A,81H

MVI B,7EH

ORA B

Ce qui donne (A)=FFH. S=1 Z=0 P=1

4. ORI : Or Immédiate : Or logique entre le contenu de l'accumulateur A et une valeur 8 bits.

Syntaxe : ORI donnée 8 bits

Exemple 4 :

MVI A,55H

ORI 02H

Ce qui donne (A)=57H. S=0 Z=0 P=0

5. XRA : Xor : Ou exclusif entre l'accumulateur A et le contenu du registre ou une case mémoire.

Syntaxe : XRA R ou XRA M

Tel que R : Registre 8 bits et M : adresse d'une case mémoire

Exemple 5:

MVI A,80H

MVI B,7EH

XRA B

Ce qui donne (A)=FEH. S=1 Z=0 P=0

6. XRI : Xor Immédiate : Ou exclusif entre le contenu de l'accumulateur A et une valeur 8bits.

Syntaxe : XRI donnée 8 bits

Exemple 6 :

MVI A,55H

XRI 02H

Ce qui donne (A)=57H. S=0 Z=0 P=0

Corrigé de l'éxercice 11

Programme en ROM	**PC**	**A**	**B**
MVI A,07h	MVI A,07h	0002h	?
MVI B, 23h	MVI B, 23h	0004h	23h
ADD B	ADD B	0005h	23h
ADI 3h	ADI 3h	0007h	23h
ANI 0h	ANI 0h	0009h	23h

Corrigé de l'éxercice 12

Programme en ROM	**PC**	**A**	**B**	**C**	**D**
MVI B,77h	0002h	**?**	77h	**?**	**?**
MVI C,88h	0004h	**?**	77h	88h	**?**
MOV A,B	0005h	77h	77h	88h	**?**
ADD C	0006h	FFh	77h	88h	**?**
MOV D,A	0007h	FFh	77h	88h	FFh

Corrigé de l'éxercice 13

Programme en ROM	PC	A	Z	C	P
MVI A FEh	0002h	FEh	0	0	0
MVI B 3h	0004h	FEh	0	0	0
ADD B	0005h	01h	0	1	0
DCR A	0006h	00h	1	1	1
DCR A	0007h	FFh	0	1	1
INC A	0008h	00h	1	1	1
INC A	0009h	01h	0	1	0
SUI 01	0009h	00h	1	0	1
SUI 01	000Dh	FFh	0	1	1
ADI 01	000Fh	00h	1	1	1
ADI 01	0011h	01h	0	0	0
SUI 34h	0013h	CDh	0	1	0
SUB A	0014h	00h	1	0	1

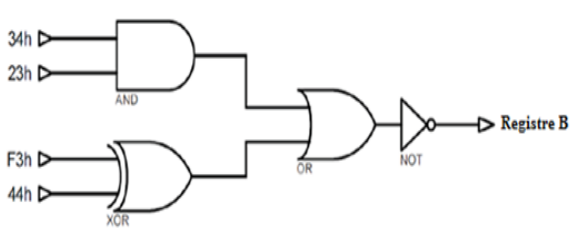
Corrigé de l'éxercice 14

Program	Result
MVI A , C4H	A=C4H
MVI B , 32H	B=32H
MVI C , F9H	C=F9H
MVI D , E2H	D=E2H
ORA B	A+B=C4+32=F6
MOV H,A	H=F6H
MOV A,C	A=F9
XRA D	A D=F9 E2=1B
CMA	A=E4
ANA H	A.H=E4.F6=E4
HLT	

Corrigé de l'éxercice 15

A) le programme assembleur de la circuit suivant :

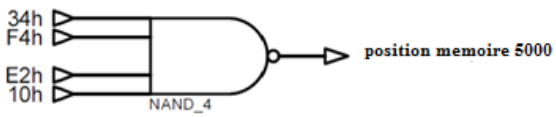
MVI A 34h
ANI 23h
MOV H A



MVI A F3h
XRI 44h
ORA H
CMA
MOV B A

B)le programme assembleur de la circuit suivant :

MVI A 34h
ANI 23h
MOV H A
MVI A F3h
XRI 44h
ORA H
CMA
MOV B A



Corrigé de l'exercice 16 :

Ecrire un programme de comptage croissant de nombre 0000h à 1000h et afficher la valeur sur les deux ports 00h et 01h			
	LXI H 0000h	Chargement de la valeur initiale de registre HL (deux Registres H et L)	
	LXI B 1001h	Chargement de la valeur qui suit la valeur finale	Pour l'affichage
BOUCLE	MOV A H	Copie de valeur de H à l'accumulateur	
	OUT 00h	Affichage Du Valeur Sur Le Port 01	
	MOV A L	Copie de valeur de H à l'accumulateur	
	OUT 01h	Affichage Du Valeur Sur Le Port 00	
	INX H	Incrémentation du registre HL (+1)	
	DCX B	Décrémentation du compteur B par 1 (-1)	
	MOV A B	transfert de la valeur de registre b a l'accumulateur	Pour la vérification que le registre BC=0000h
	ORA C	Faire un OR logique entre le registre C et l'accumulateur A	
	JNZ LOOP	Saur si le résultat non nul	
	HLT		

Corrigé de l'éxercice 17

Ecrire un programme qui permet d'effectuer la multiplication des contenus de deux cases mémoires (l'emplacement 2200h et 2201h et stocker le résultat à partir de l'adresse 2300h

Ecrire un programme qui permet d'effectuer la multiplication des contenus de deux cases

mémoires (l'emplacement 2200h et 2201h et stocker le résultat à partir de l'adresse 2300ha		
Adresse mémoire	**Instruction**	
	LDA 2200H	Chargement de la première valeur
	MOV E, A	Stockage dans le registre E
	MVI D, 00	Initialisation de registre D
	LDA 2201H	Chargement de la première valeur
	MOV C, A	transfert de la deuxième valeur au registre C
	LX I H, 0000 H	Initialisation de registre HL
BACK:	DAD D	Addition de la première valeur
	DCR C	Décrémentation du compteur
	JNZ BACK	Retour au début de programme si la condition ne satisfie pas (non nul)
	SHLD 2300H	Stockage des résultats dans les deux positions mémoires 2300h et 2301

Corrigé de l'éxercice 18

Programme en ROM	**PC**	**A**	**B**	**C**	**Flag Z**	**Flag S**
MVI A, 50h	0002h	0002h	00h	00h	0	0
MVI B, 30h	0004h	0004h	30h	00h	0	0
MVI C,70h	0006h	0006h	30h	70h	0	0
CMP B	0007h	0007h	30h	70h	0	0
CMP C	0008h	0008h	30h	70h	0	1
CMP A	0009h	0009h	30h	70h	1	0
CPI 41h	000Bh	000Bh	30h	70h	0	0
CPI 50h	000D	000D	30h	70h	1	0
SUB B	000Eh	000Eh	30h	70h	0	0
SUB C	000Fh	000Fh	30h	70h	0	1

Corrigé de l'éxercice 19

Le début de la mémoire contient les données suivantes :

3E FE 06 01 80 E6 11 E6 00

Ce code représente les instructions du programme que l'on vient d'entrer.

3E FE --> MVI A,FEh

06 01 --> MVI B,01h

80 --> ADD B
E6 11 --> ANI 11h

E6 00 --> ANI 00h

Corrigé de l' éxercice 20

Trouver le code machine de ce programme assembleur, le programme commence par l'adresse 3000

Adresse	Langage machine
3000	16
3001	8B
3002	0E
3003	6F
3004	79
3005	82
3006	32
3007	2E
3008	30
3009	C3
300A	02
300B	30

LABEL	MVI D 8Bh MVI C 6Fh MOV A C ADD D STA 302Eh JMP LABEL

Corrigé de l' éxercice 21

Programme en ROM	**PC**	**A**	**C**
MVI A, AAh	0002h	AAh	0
RAL	0003h	54h	1
RAL	0004h	A9h	0
RAL	0005h	52h	1
RLC	0006h	A4h	0
RLC	0007h	49h	1
RLC	0008h	92h	0
RRC	0009h	49h	0
RRC	000Ah	A4h	1
RRC	000Bh	52h	0
RAR	000Ch	29h	0
RAR	000Dh	14h	1
RAR	000Eh	8Ah	0

Corrigé de l'éxercice 22

Adresse Mémoire	**instruction**	**PC**	**A**	**B**
0000h	MVI A, 00h	0002h	00h	??
0002h	MVI B,01h	0004h	00h	01h
0004h	ADD B	0005h	01h	01h
0005h	JMP 000Bh	000Bh	01h	01h
000Bh	JMP 0004h	0004h	01h	01h
0004h	ADD B	0005h	02h	01h
0005h	JMP 000Bh	000Bh	02h	01h
000Bh	JMP 0004h	0004h	02h	01h
0004h	ADD B	0005h	03h	01h
0005h	JMP 000Bh	000Bh	03h	01h
000Bh	JMP 0004h	0004h	03h	01h
0004h	ADD B	0005h	04h	01h

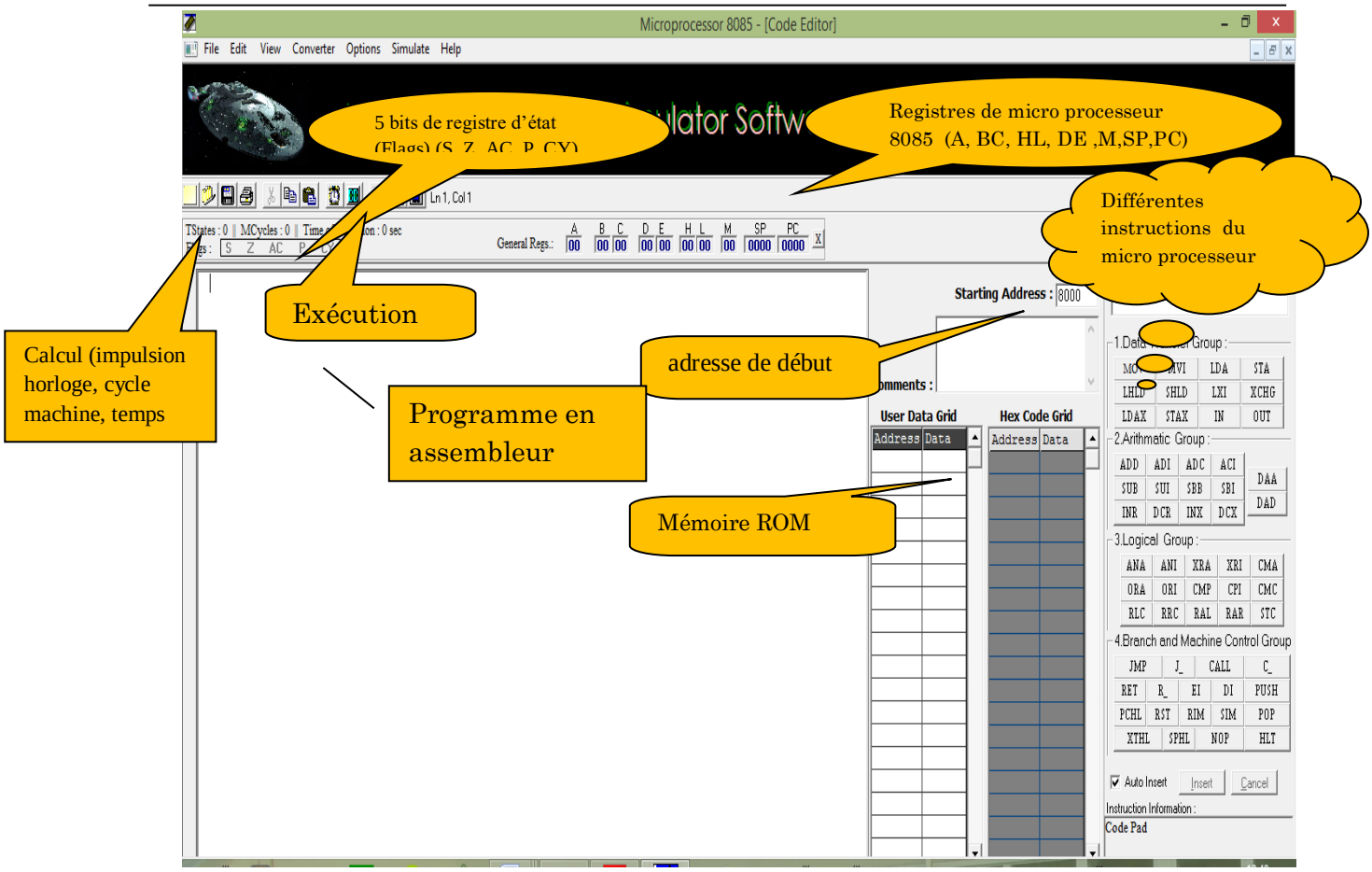
0005h	JMP 000Bh	000Bh	04h	01h
000Bh	JMP 0004h	0004h	04h	01h
0004h	ADD B	0005h	05h	01h
0005h	JMP 000Bh	000Bh	05h	01h
000Bh	JMP 0004h	0004h	05h	01h
0004h	ADD B	0005h	06h	01h

Corrigé de l'éxercice 23

Label	**Instruction**	**PC**	**A**	**C**	**Z**
0000h	MVI A,10h	0002h	20h	00h	0
00002h	MVI C,04h	0004h	20	04h	0
0004h	ADD A	0005h	40	04h	0
0005h	DCR C	0006h	40	03h	0
0006h	JNZ 0004h	0004h	40	03h	0
0004h	ADD A	0005h	80	03h	0
0005h	DCR C	0005h	80	02h	0
0006h	JNZ 0004h	0004h	80	02h	0
0004h	ADD A	0005h	00	02h	1
0005h	DCR C	0006h	00	01h	0
0006h	JNZ 0004h	0004h	00	01h	0
0004h	ADD A	0005h	00	01h	1
0005h	DCR C	0006h	00	00h	1
0007h	HLT				

Le simulateur (8085simulator) est un programme informatique simulant le fonctionnement du microprocesseur 8085 au point de vue de ces registres, de sa mémoire, des ces flags et ses éventuelles interfaces I/O.en peut donner une description rapide de ce logeciel :

- Ce simulateur "8085 Simulator" est logiciel très utile pour ceux qui commencent à apprendre le langage Assembleur. Il compile le code source et l'exécute pas à pas dans simulateur 8085 Simulator.
- Visualiser la progression de programme dans les différentes registres A, B, C, D, E, H, L, compteur de programme (CP), pointeur de pile (SP), Flags (F), mémoire, interfaces d'entrée-sortie, convertisseurs binaire - décimal - hexadécimal,... La fenêtre ressemble à la figureci- dessous



Prise en main de Microprocessor 8085simulator

- On doit saisir les programmes dans la fenêtre blanche Figure 1 : Interface du simulateur du 8085
- Une fois la saisie du programme est terminée et après la correction des erreurs syntaxiques on peut le simuler en cliquant sur le bouton "Run".
- Le simulateur résume le jeu d'instruction du microprocesseur 8085, c'est une forme d'aide pour le programmeur.
- Figure 2 : Programme assemblé. Le programme peut être exécuter tout entier (Run all At a Time) ou instruction par instruction (Step By Step). On peut visualiser les contenus de tous les registres du microprocesseur ainsi que la mémoire et les entrées sorties du système.

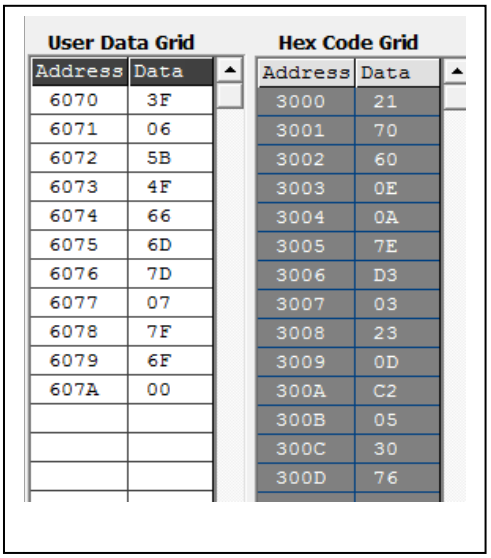
  Nous allons effectuer plusieurs Tps (comptage, décomptage, rotation, permutation des données, affichage des données à la sortie des ports, recevoir des données…..) par ce simulateur, et nous avons choisi bien des exemples adéquats pour notre besoin, en peut résumer les objectifs de ces Tps dans des points :
- Familiarisation avec le simulateur 8085 Simulator.
- Maitriser les différentes instructions : de transfert, de données instructions arithmétiques et logiques, instructions de saut et rotation, instructions de la gestion de la pile, instructions de branchement et les boucles
- Maitriser la notion de modes d'adressage.
- Configurer l'espace mémoire
- Entrer un programme
- Repérer sur l'écran les différents registres, mémoires, flags
- Comparer le résultat attendu avec la théorie

TP1: BCD à affichage à sept segments.

- Ecrire un programme en assembleur qui permet de
- l'affichage un nombre écrit en BCD vers un afficheur sept segments
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)

(Remarque :Réglez la fréquence d'horloge à 30 Hz.)

Programme en mémoire



```
LXI H 6070h
MVI C 0Ah
SHOW: MOV A M
OUT 03h
INX H
DCR C
JNZ SHOW
HLT
```

Tp2 : Lecture de donnés et stockage en mémoire

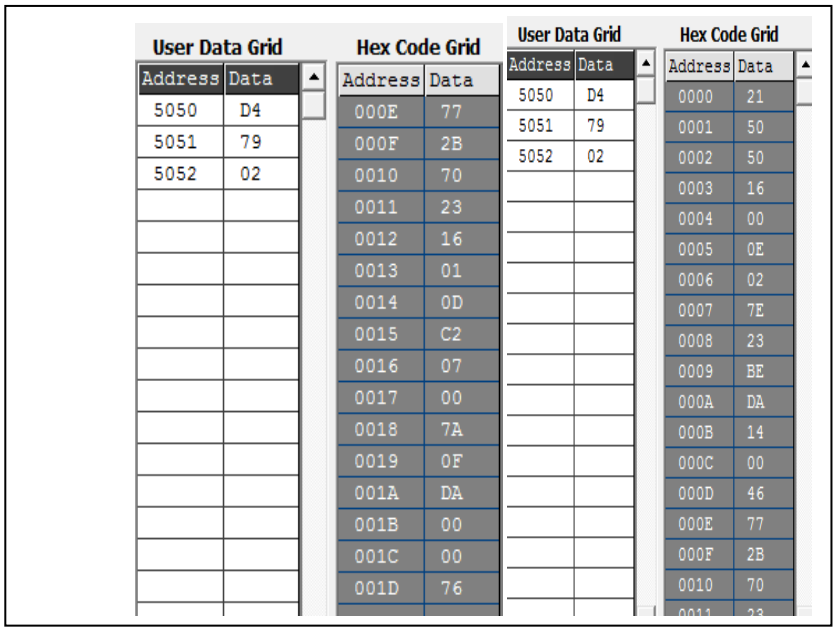
Ecrire un programme en assembleur qui permet d'effectuer ce problème :

Un ensemble de trois lectures est stocké en mémoire à partir de XX50h. Triez les lectures par ordre croissant.

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

(Remarque : réglez la fréquence d'horloge à 10 Hz)

```
START: LXI H 5050h
MVI D 00h
MVI C 02h
CHECK:  MOV A M
        INX H
        CMP M
        JC NXTBYT
        MOV B M
        MOV M A
        DCX H
        MOV M B
        INX H
        MVI D 01h
        NXTBYT: DCR C
        JNZ CHECK
        MOV A D
        RRC
        JC START
        HLT
CHECK: MOV A M
        INX H
        CMP M
        JC NXTBYT
        MOV B M
        MOV M A
        DCX H
        MOV M B
        INX H
        MVI D 01h
```



```
NXTBYT: DCR C
JNZ CHECK
MOV A D
RRC
JC START
HLT
```

TP3: Instruction arithmitiques (addition stockage transfert....)

Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

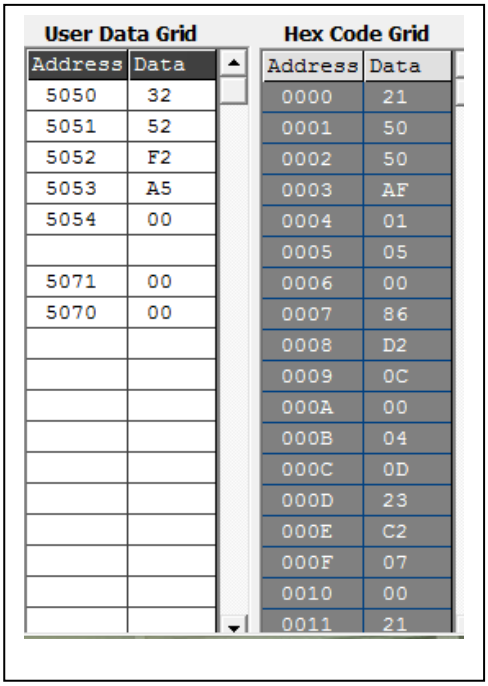
le programme ajoutera les 5 octets écrits de 5050 à 5054 et reflétera le résultat de 5071 à 5072.

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

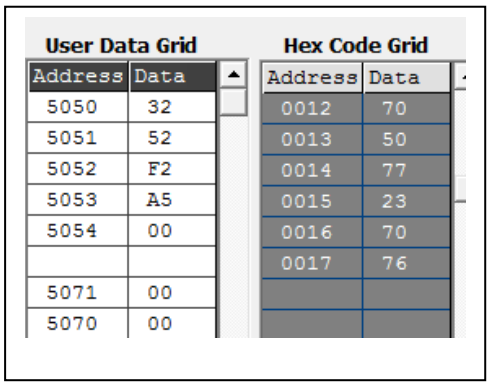
```
LXI H 5050h
XRA A
LXI B 0005h

LBL1: ADD M
JNC LBL2
INR B

LBL2: DCR C
INX H
JNZ LBL1
LXI H 5070h
MOV M A
INX H
MOV M B
HLT
```



User Data Grid		Hex Code Grid	
Address	Data	Address	Data
5050	32	0000	21
5051	52	0001	50
5052	F2	0002	50
5053	A5	0003	AF
5054	00	0004	01
		0005	05
5071	00	0006	00
5070	00	0007	86
		0008	D2
		0009	0C
		000A	00
		000B	04
		000C	0D
		000D	23
		000E	C2
		000F	07
		0010	00
		0011	21



User Data Grid		Hex Code Grid	
Address	Data	Address	Data
5050	32	0012	70
5051	52	0013	50
5052	F2	0014	77
5053	A5	0015	23
5054	00	0016	70
		0017	76
5071	00		
5070	00		

Programme en mémoire

TP4 : la plus grande et la plus petite lecture et l'affichage sur les ports de sortie

Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

Un ensemble de 10 lectures sont stockées à partir de 2050h.

Trouvez la plus grande et la plus petite lecture parmi elles et affichez-les respectivement sur le port1 et le port2

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

User Data Grid

Address	Data
2050	48
2051	32
2052	F2
2053	38
2054	37
2055	40
2056	82
2057	8A

Hex Code Grid

Address	Data
9012	23
9013	0D
9014	C2
9015	07
9016	90
9017	7A
9018	D3
9019	00
901A	7B
901B	D3
901C	01
901D	76

START: LXI H 2050h
COUNTER: MVI C 08h
STOREHIGH: MOV D M
STORELOW: MOV E M
NXTBYT: MOV A M
HIGH: CMP D
JC LOW
MOV D A
LOW: CMP E
JNC NEXT
MOV E A
NEXT: INX H
DCR C
JNZ NXTBYT
MOV A D
DISPLAYHIGH: OUT 00h
MOV A E
DISPLAYLOW: OUT 01h
HLT

User Data Grid

Address	Data
2050	48
2051	32
2052	F2
2053	38
2054	37
2055	40
2056	82
2057	8A

Hex Code Grid

Address	Data
9000	21
9001	50
9002	20
9003	0E
9004	08
9005	56
9006	5E
9007	7E
9008	BA
9009	DA
900A	0D
900B	90
900C	57
900D	BB
900E	D2
900F	12
9010	90
9011	5F

Programme en mémoire

Tp5 : Copie de bloc de données de 16 bits vers autre emplacement mémoire

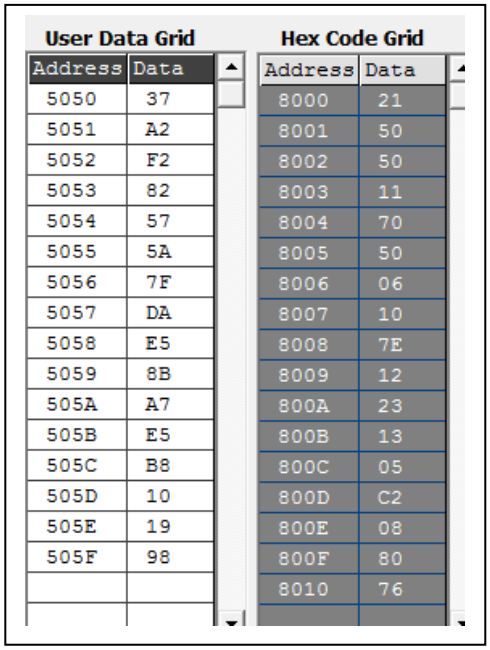
Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

On a seize octets de données sont stockés dans des emplacements mémoire de 5050h à 505Fh.

Copiez le bloc de données dans de nouveaux emplacements de mémoire à partir de 5070h

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

```
START: LXI H 5050h
       LXI D 5070h
       MVI B 10h
NEXT:  MOV A M
       STAX D
       INX H
       INX D
       DCR B
       JNZ NEXT
       HLT
```



User Data Grid		Hex Code Grid	
Address	Data	Address	Data
5050	37	8000	21
5051	A2	8001	50
5052	F2	8002	50
5053	82	8003	11
5054	57	8004	70
5055	5A	8005	50
5056	7F	8006	06
5057	DA	8007	10
5058	E5	8008	7E
5059	8B	8009	12
505A	A7	800A	23
505B	E5	800B	13
505C	B8	800C	05
505D	10	800D	C2
505E	19	800E	08
505F	98	800F	80
		8010	76

Programme en mémoire

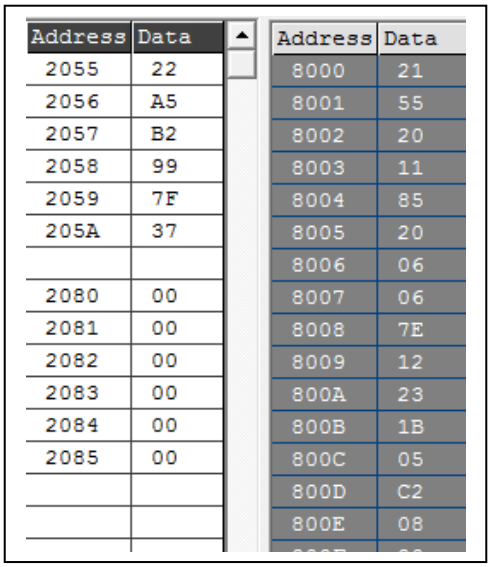
TP6: Copie de bloc de données de 6 mots (Bytes)vers autre emplacement mémoire

Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

Six octets de données sont stockés dans la mémoire à partir de 2055h. Copiez le bloc de données vers l'emplacement mémoire à partir de 2080h dans l'ordre inverse

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

```
START: LXI H 2055h
       LXI D 2085h
       MVI B 06h
NEXT:  MOV A M
       STAX D
       INX H
```



Address	Data	Address	Data
2055	22	8000	21
2056	A5	8001	55
2057	B2	8002	20
2058	99	8003	11
2059	7F	8004	85
205A	37	8005	20
		8006	06
2080	00	8007	06
2081	00	8008	7E
2082	00	8009	12
2083	00	800A	23
2084	00	800B	1B
2085	00	800C	05
		800D	C2
		800E	08

Programme en mémoire

DCX D

DCR B

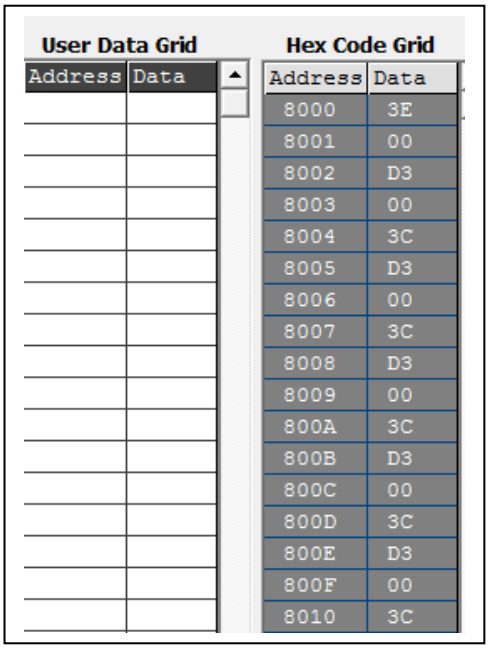
JNZ NEXT

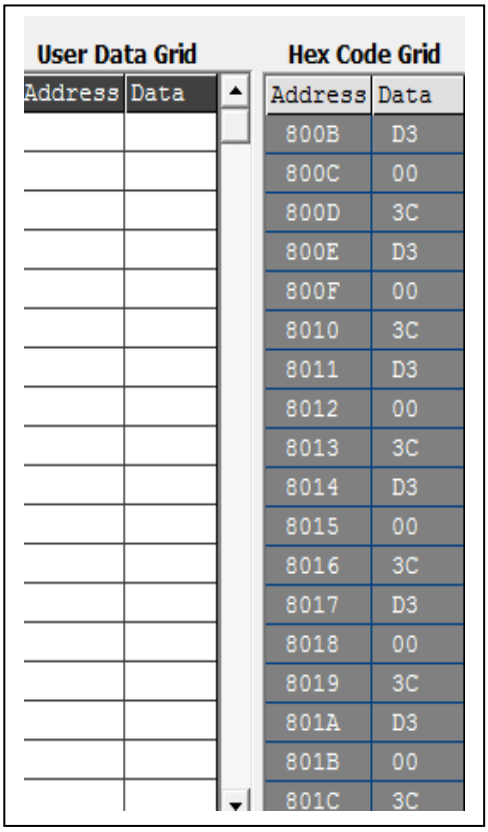
Tp7: Comptage

Ecrire un programme en assembleur qui permet d'effectuer le comptage d'un registre ou compteur et l'affichage de la valeur chaque fois au port de sortie 00 l'adresse de début est 8000H, initialisez tous les registres.

- Exécuter le programme en cliquant sur le bouton Run
- Corriger des erreurs syntaxiques
- Vérifier bien les contenus des différents registres, mémoires, les 5 bits flags(S ,Z ,AC,P,CY)
- L'état de mémoire

```
MVI A 00h
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
OUT 00h
INR A
```





Tp8 : chargement et initialisation d'une zone mémoire

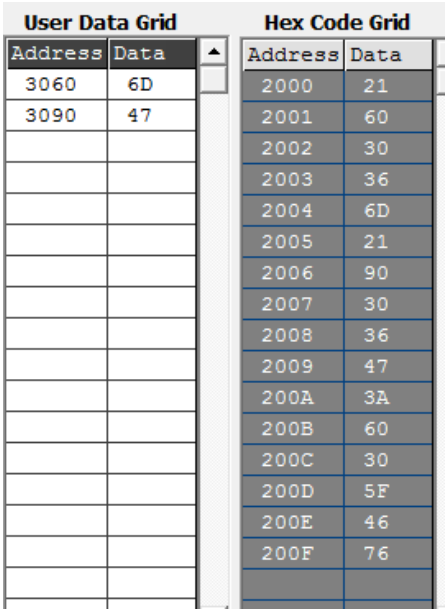
Programme en mémoire

Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

Initialiser des zones mémoire 3060h et 3090h par des données 6Dh et 47 h respectivement et transfert de ces données à l'accumulateur

```
LXI H 3060h
MVI M 6Dh
LXI H 3090h
MVI M 47h
LDA 3060h
MOV E A
MOV E M
HLT
```

Programme en mémoire



User Data Grid

Address	Data
3060	6D
3090	47

Hex Code Grid

Address	Data
2000	21
2001	60
2002	30
2003	36
2004	6D
2005	21
2006	90
2007	30
2008	36
2009	47
200A	3A
200B	60
200C	30
200D	5F
200E	46
200F	76

TP 09 : Gestion de la pile

Ecrire un programme en assembleur qui permet d'effectuer le problème suivant :

BC	0503
DE	4F4D
HL	234C

- Effectuer l'addition des contenus des registres Bet C
- créer une zone de pile commence par 8000
- Stocker tous les registres et les flags dans cette zone
- récupérer ces contenus

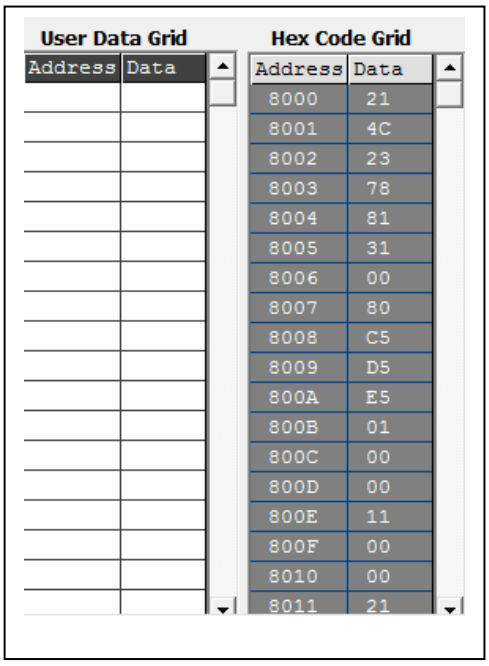
En veut utiliser la pile et sauvegarde les contenus dans la pile et en récupère les contenus aux registres, le pointeur de la pile SP commence par 8000,

LXI B 0405

LXI D 4F4Dh

LXI H 234Ch

MOV A B
ADD C
LXI SP 8000h
PUSH B
PUSH D
PUSH H
LXI B 0000h
LXI D 0000h
LXI H 0000h
POP H
POP D
POP B



User Data Grid

Address	Data

Hex Code Grid

Address	Data
8000	21
8001	4C
8002	23
8003	78
8004	81
8005	31
8006	00
8007	80
8008	C5
8009	D5
800A	E5
800B	01
800C	00
800D	00
800E	11
800F	00
8010	00
8011	21

Programme en mémoire

- Résumé des cours de ma graduation ‘’cours de microprocesseurs’’ 1999, univ de Laghouat,
- Résumé des cours de ma post graduation ‘’cours de microprocesseurs’’2008 ENP, ORAN.
- Ordinateur et microprocesseur pratique Collège De La Salle
- Cours et travaux dirigés sur : Les microprocessuers 8 bits Abid sabeur tunis 2003/2002
- Cours de microprocesseur HAGG`EGE, 2003 ISET Radés
- Systèmes a microprocesseurs cours & exercices dr nabil boukhennoufa
- Cours de Microprocesseur Esnard Aurélien ENSERB Informatique 1ère année

Liens internet

- https://www.youtube.com/watch?v=iEb72cXKUn4
- https://www.youtube.com/watch?v=bvDElRtlGPk
- https://www.youtube.com/watch?v=5hF4ziHwJBQ
- https://www.youtube.com/watch?v=2nAJLWPObJQ
- https://www.youtube.com/watch?v=ebqSGDAxdkw&list=PLOZFp92Iw5-6Am5p5UaMfwULiTxHlBsW0
- https://www.youtube.com/watch?v=ODURM5ihc7M
- https://www.youtube.com/watch?v=E0q82pxBlmk
- https://www.youtube.com/watch?v=o-AZCrc84Tc
- https://www.youtube.com/watch?v=ImxwKsFhEgc

-

LOGICIEL DU 8080 A OU 8085 A INTEL

INST.	HEXA.	INST.	HEXA.	INST.	HEXA.	INST.	HEXA.
ACI	CE	DCR L	2D	MOV C,M	4E	POP D	D1
ADC A	8F	DCR M	35	MOV D,A	57	POP H	E1
ADC B	88	DCX B	0B	MOV D,B	50	POP PSW	F1
ADC C	89	DCX D	1B	MOV D,C	51	PUSH B	C5
ADC D	8A	DCX H	2B	MOV D,D	52	PUSH D	D5
ADC E	8B	DCX SP	3B	MOV D,E	53	PUSH H	E5
ADC H	8C	DI	F3	MOV D,H	54	PUSH PSW	F5
ADC L	8D	EI	FB	MOV D,M	56	RAL	17
ADC M	8E	HLT	76	MOV E,A	5F	RAR	1F
ADD A	87	IN	DB	MOV E,B	58	RC	D8
ADD B	80	INR A	3C	MOV E,C	59	RET	C9
ADD C	81	INR B	04	MOV E,D	5A	RLC	07
ADD D	82	INR C	0C	MOV E,E	5B	RM	F8
ADD E	83	INR D	14	MOV E,H	5C	RNC	D0
ADD H	84	INR E	1C	MOV E,L	5D	RNZ	C0
ADD L	85	INR H	24	MOV E,M	5E	RP	F0
ADD M	86	INR L	2C	MOV H,A	67	RPE	E8
ADI	C6	INR M	34	MOV H,B	60	RPO	E0
ANA A	A7	INX B	03	MOV H,C	61	RRC	0F
ANA B	A0	INX D	13	MOV H,D	62	RST 0	C7
ANA C	A1	INX H	23	MOV H,E	63	RST 1	CF
ANA D	A2	INX SP	33	MOV H,H	64	RST 2	D7
ANA E	A3	JC	DA	MOV H,L	65	RST 3	DF
ANA H	A4	JM	FA	MOV H,M	66	RST 4	E7
ANA L	A5	JMP	C3	MOV L,A	6F	RST 5	EF
ANA M	A6	JNC	D2	MOV L,B	68	RST 6	F7
ANI	E6	JNZ	C2	MOV L,C	69	RST 7	FF
CALL	CD	JP	F2	MOV L,D	6A	RZ	C8
CC	DC	JPE	EA	MOV L,E	6B	SBB A	9F
CM	FC	JPO	E2	MOV L,H	6C	SBB B	98
CMA	2F	JZ	CA	MOV L,L	6D	SBB C	99
CMC	3F	LDA	3A	MOV L,M	6E	SBB D	9A
CMP A	BF	LDAX B	0A	MOV M,A	77	SBB E	9B
CMP B	B8	LDAX D	1A	MOV M,B	70	SBB H	9C
CMP C	B9	LHLD	2A	MOV M,C	71	SBB L	9D
CMP D	BA	LXI B	01	MOV M,D	72	SBB M	9E
CMP E	BB	LXI D	11	MOV M,E	73	SBI	DE
CMP H	BC	LXI H	21	MOV M,H	74	SHLD	22
CMP L	BD	LXI SP	31	MOV M,L	75	SPHL	F9
CMP M	BE	MOV A,A	7F	MVI A	3E	STA	32
CNC	D4	MOV A,B	78	MVI B	06	STAX B	02
CNZ	C4	MOV A,C	79	MVI C	0E	STAX D	12
CP	F4	MOV A,D	7A	MVI D	16	STC	37
CPE	EC	MOV A,E	7B	MVI E	1E	SUB A	97
CPI	FE	MOV A,H	7C	MVI H	26	SUB B	90
CPO	E4	MOV A,L	7D	MVI L	2E	SUB C	91
CZ	CC	MOV A,M	7E	MVI M	36	SUB D	92
DAA	27	MOV B,A	47	NOP	00	SUB E	93
DAD B	09	MOV B,B	40	ORA A	B7	SUB H	94
DAD D	19	MOV B,C	41	ORA B	B0	SUB L	95
DAD H	29	MOV B,D	42	ORA C	B1	SUB M	96
DAD SP	39	MOV B,E	43	ORA D	B2	SUI	D6
DCR A	3D	MOV B,H	44	ORA E	B3	XCHG	EB
DCR B	05	MOV B,L	45	ORA H	B4	XRA A	AF
DCR C	0D	MOV B,M	46	ORA L	B5	XRA B	A8
DCR D	15	MOV C,A	4F	ORA M	B6	XRA C	A9
DCR E	1D	MOV C,B	48	ORI	F6	XRA D	AA
DCR H	25	MOV C,C	49	OUT	D3	XRA E	AB
		MOV C,D	4A	PCHL	E9	XRA H	AC
		MOV C,E	4B	POP B	C1	XRA L	AD
		MOV C,H	4C			XRA M	AE
		MOV C,L	4D			XRI	EE
						XTHL	E3

Différentes instructions du logiciel du 8080 A ou du 8085 A de INTEL

Type d'instruction	Instruction	Opération accomplie	rég. concern. par r ou rp	Indicateurs affectés
incrémentation et décrémentation	INR M INR r DCR M DCR r INX rp DCX rp	(M) + 1 → M (r) + 1 → r (M) − 1 → M (r) − 1 → r (rp) + 1 → rp (rp) − 1 → rp	A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP BC, DE, HL, SP	Z, S, P, AC Z, S, P, AC Z, S, P, AC Z, S, P, AC aucun aucun
branchement	JMP addr CALL addr RET JC, CC, RC addr JNC, CNC, RNC addr JZ, CZ, RZ addr JNZ, CNZ, RNZ addr JP, CP, RP addr JM, CM, RM addr JPE, CPE, RPE addr JPO, CPO, RPO addr	saut inconditionnel à l'adresse addr appel sous-programme d'adresse addr retour de sous-programme saut, appel SP, retour SP si CY = 1 saut, appel SP, retour SP si CY = 0 saut, appel SP, retour SP si Z = 1 saut, appel SP, retour SP si Z = 0 saut, appel SP, retour SP si signe = 0 saut, appel SP, retour SP si signe = 1 saut, appel SP, retour SP si pair saut, appel SP, retour SP si impair		AUCUN
instructions de décalage	RLC RAL RRC RAR	déc. gauche, CY hors boucle déc. gauche, CY dans boucle déc. droit, CY hors boucle déc. droit, CY dans boucle	VOIR BAS DE LA PAGE	CY
instructions spéciales	EI DI NOP HLT	Interruption autorisée Interruption non autorisée Rien Arrêt microprocesseur		AUCUN

LOGICIEL DU 8080 A OU DU 8085 A DE INTEL

Type d'instruction	Instruction	Opération accomplie	rég. concernés par r ou rp	Indicateurs affectés
lecture périphérique écriture périphérique	IN addr OUT addr	(port) → A (A) → port		AUCUN
lecture mémoire	LDA addr LHLD addr LDAX rp MOV r,M POP rp	(M) → A (M) → HL ((rp)) → A (M) → r ((SP)) → rp	 BC, DE A, B, C, D, E, H, L PSW, BC, DE, HL	AUCUN sauf pour POP PSW
écriture mémoire	STA addr SHLD addr STAX rp MOV M, r MVI M data PUSH rp	(A) → M (HL) → M (A) → (rp) (r) → M data → M (rp) → (SP)	 BC, DE A, B, C, D, E, H, L PSW, BC, DE, HL	AUCUN
transferts de données	MOV r1, r2 MVI r data LXI rp data PCHL SPHL XCHG XTHL	(r2) → r1 data → r data → rp (HL) → PC (HL) → SP (HL) ↔ (DE) (HL) ↔ ((SP))	A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP	AUCUN
opérations arithmétiques	ADD M ADD r ADC M ADC r ADI data ACI data DAA SUB M SUB r SBB M SBB r SUI data SBI data DAD rp	(A) + (M) → A (A) + (r) → A (A) + (CY) + (M) → A (A) + (CY) + (r) → A (A) + data → A (A) + (CY) + data → A correction décim. (A) − (M) → A (A) − (r) → A (A) − (CY) − (M) → A (A) − (CY) − (r) → A (A) − data → A (A) − (CY) − data → A (HL) + (rp) → HL	A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L BC, DE, HL, SP	TOUS CY
fonctions logiques	ANA M ANA r ORA M ORA r XRA M XRA r ANI data ORI data XRI data CMP M CMP r CPI data CMA CMC STC	(A) . (M) → A (A) . (r) → A (A) + (M) → A (A) + (r) → A (A) ⊕ (M) → A (A) ⊕ (r) → A (A) . data → A (A) + data → A A ⊕ data → A (A) − (M) (A) − (r) (A) − data $(\overline{A})$ → A $(\overline{CY})$ → CY 1 → CY	A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L A, B, C, D, E, H, L	Indicateurs remis à zéro CY CY, AC TOUS CY, AC CY, AC CY, AC aucun CY CY

- Pour les opérandes ou les adresses de 16 bits, l'octet poids faible est traité le premier.
- La lettre M dans le code instruction symbolise l'adressage indirect par HL.