

Introduction Générale:

Introduction :

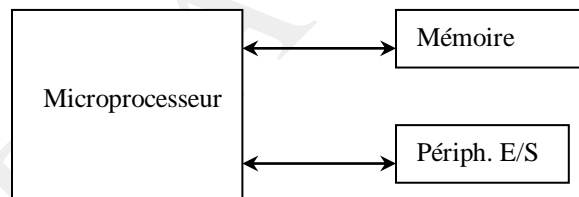
Le microprocesseur est entrain d'avoir un grand impact dans la vie de tous les jours de l'homme et joue un rôle important dans les sociétés industrielles.

Un CPU construit sur un circuit intégré à partir de la technologie VLSI est appelé Microprocesseur. Un ordinateur dont le CPU est un microprocesseur est appelé micro-ordinateur. Un microprocesseur combiné avec une mémoire et les périphériques d'E/S forment un micro-ordinateur.

Définition :

Un microprocesseur est un instrument ou un appareil logique programmable, à utilité multiple qui lit des instructions sous forme binaire à partir d'un périphérique de stockage appelée mémoire. Ensuite le microprocesseur reçoit des données sous forme binaire; et les traite selon les instructions et fournit le résultat vers un périphérique de sortie.

Toute machine programmable peut être représentée par les trois composantes suivantes : Microprocesseur – Mémoire – Périphériques d'E/S comme montré par la figure suivante :



Ces trois composantes prises ensemble forment un système pour réaliser une tâche bien précise

I- Évolution du Microprocesseur :

Le 1er microprocesseur a été introduit en 1971 par Intel Corporation USA. C'était un microprocesseur de 4-bit appelé Intel 4004. En 1972 Intel introduit le premier microprocesseur de 8-bit appelé Intel 8008. La technologie PMOS a été utilisée dans la fabrication de Intel 4004 et 8008. En 1973 un autre microprocesseur beaucoup plus rapide appelé Intel 8080 a été mis sur le marché. C'est la technologie NMOS qui a été utilisée dans sa fabrication.

NB : Les termes 4-bit et 8-bit utilisés pour les microprocesseurs méritent une explication. Un microprocesseur de 4-bit peut traiter 4 bits en parallèle, à la fois (d'un trait). Si les données à traiter sont plus de 4 bits, le microprocesseur va prendre jusqu'à 4 bits pour

les traiter d'abord, puis prendre encore les 4 prochains bits pour les traiter etc..... De la même manière, un microprocesseur de 8-bit peut gérer 8 bits de données en parallèle à la fois.

Aujourd'hui un certain nombre de microprocesseur de 8-bit sont disponibles sur le marché comme : Motorola MC6809, Zilog's Z80 et Z800, les séries 6500 de MOS Technology, les NSC800 de National Semiconductor, le SCL6502 de Semiconductor Complex etc.....

Les microprocesseurs de 16-bit sont les suivants après les microprocesseurs de 8-bit. Les exemples de microprocesseurs de 16-bit sont : Intel 8086, 80186, 80286, Motorola's MC68000, les séries de Zilog's Z8000, les TMS 9900 de Texas Instruments, le 9940 de Fairchild, le LSI 11 de Digital Equipment etc.....

Les microprocesseurs les plus récents ont un mot de longueur de 32-bit. Exemple : Intel 80386 (1985) et 80486, Motorola's 68030 (1987) et 68040, National Semiconductor NS32032 etc.....

Ces microprocesseurs fonctionnent à des fréquences différentes et adressent des capacités de mémoires différentes. Exemple : Intel 80386 est disponible à des fréquences de 20, 25, et 33 Mhz et peut adresser jusqu'à 4 GB de mémoire.

Tableau 1.1 et 1.2 donnent les résumés des microprocesseurs Intel et Motorola respectivement.

Remarque : Actuellement le microprocesseur le plus récent est le Pentium (80586) qui est de 64-bit.

II- Organisation d'un système basé sur le microprocesseur :

La figure 2.1 montre le schéma d'un système simple. Les composantes principales sont : le CPU (Central Processing Unit), la mémoire et les périphériques d'Entrée/Sortie. Ces composantes sont interconnectées par l'intermédiaire de 3 bus: le bus d'adresse, le bus de données et le bus de contrôle.

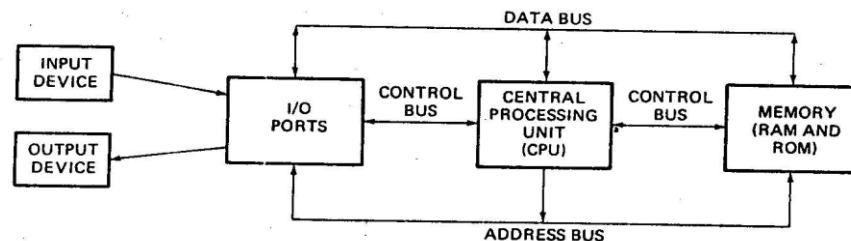


FIGURE 2-1 Block diagrams of a simple computer or microcomputer.

la mémoire :

La mémoire est composée d'un ROM et d'un RAM. Elle peut être aussi une disquette ou un disque dur ou un disque laser. Le premier rôle de la mémoire est de stocker (recevoir) les instructions sous forme de codes binaires appelées programme que l'ordinateur doit exécuter. Le second rôle de la mémoire est des stocker (recevoir) les données sous forme binaire sur lesquelles l'ordinateur doit agir.

NB : La mémoire est subdivisée en parties égales. Les instructions sont stockées au niveau de ces parties bien précises appelées Emplacements. Chaque emplacement a une adresse unique. En général, les informations que reçoit la mémoire sont sous forme d'octets (8-bit). Chaque emplacement de mémoire ne peut contenir qu'un seul octet ou byte. Et l'adresse de l'emplacement de mémoire peut être écrite sous forme de nombres binaires ou hexadécimaux.

Exemple :

8000	AF
8002	74H
8002	ABH
8003	78H

Les périphériques d'entrées / sorties (Input/Output) :

Les entrées/sorties permettent à l'ordinateur de prendre des informations ou d'envoyer des informations au monde extérieur. Les périphériques comme le clavier, l'écran, les imprimantes, les modem sont connectés à la section Entrée/Sortie. Ceci va permettre à l'utilisateur et à l'ordinateur de communiquer entre eux. Les instruments physiques utilisés pour mettre en interface les bus de l'ordinateur et les systèmes extérieurs sont souvent appelés ports. Un port pour un ordinateur fonctionne de la même manière que les ports maritimes ou les aéroports pour un pays. Un port d'entrée permet à des données qui proviennent d'un clavier ou d'un convertisseur A/D ou d'une autre source d'être reçues par l'ordinateur sous le contrôle du CPU. Un port de sortie permet d'envoyer des données à partir de l'ordinateur vers d'autres périphériques comme l'écran, l'imprimante ou un convertisseur D/A etc....

Le Central Processing Unit (CPU) :

Le Central Processing Unit contrôle les opérations de l'ordinateur. Il lit les instructions sous forme binaire, décode les instructions en une série d'actions simples et exécute ces actions.

Le Bus d'adresse

Le bus d'adresse est constitué de 13, 20, 24 signaux sous forme de lignes parallèles. Ces lignes existent entre le microprocesseur et la mémoire. On sait déjà que la mémoire est subdivisée en emplacements et chaque emplacement a une adresse unique. C'est à partir des lignes du Bus d'adresse que le microprocesseur envoie l'adresse de l'emplacement de mémoire pour pouvoir stocker ou écrire des informations à cet emplacement.

Si le CPU a N lignes d'adresses; Il peut adresser jusqu'à 2^N emplacements de mémoires. Exemple : Un CPU de 16 lignes d'adresse peut adresser jusqu'à 216 ou 65536 emplacements de mémoire. Un CPU de 20 lignes d'adresse peut adresser jusqu'à 220 ou 1048576 emplacements de mémoire. Un CPU de 24 lignes d'adresse peut adresser jusqu'à 224 ou 16777216 emplacements de mémoire.

Lorsque le microprocesseur lit à partir ou écrit des données vers un port; l'adresse du port est envoyée à partir du Bus de données.

Le Bus de données

Le bus de données est constitué de 8, 16 32 ou plus de lignes (conducteurs) parallèles. Comme indiqué par les flèches dans les deux sens au niveau de la figure 2.1. Le bus de données est bi-directionnel. Ce qui veut dire que le processeur peut lire (recevoir) des données ou des informations à partir de la mémoire ou d'un port par l'intermédiaire du bus de données ou écrire (envoyer) des données ou des informations à la mémoire ou à un port par l'intermédiaire du bus de données.

Le nombre de données que le processeur peut traiter à la fois correspond au nombre de ligne du bus de données. Ce nombre de ligne est appelé Largeur du bus et caractérise le microprocesseur. Et cette largeur du bus désigne un microprocesseur; comme par exemple, un microprocesseur de 8-bit, ou un microprocesseur de 16-bit ou un microprocesseur de 32-bit; ici 8, 16, 32 est le nombre de lignes du bus de données.

Le Bus de contrôle

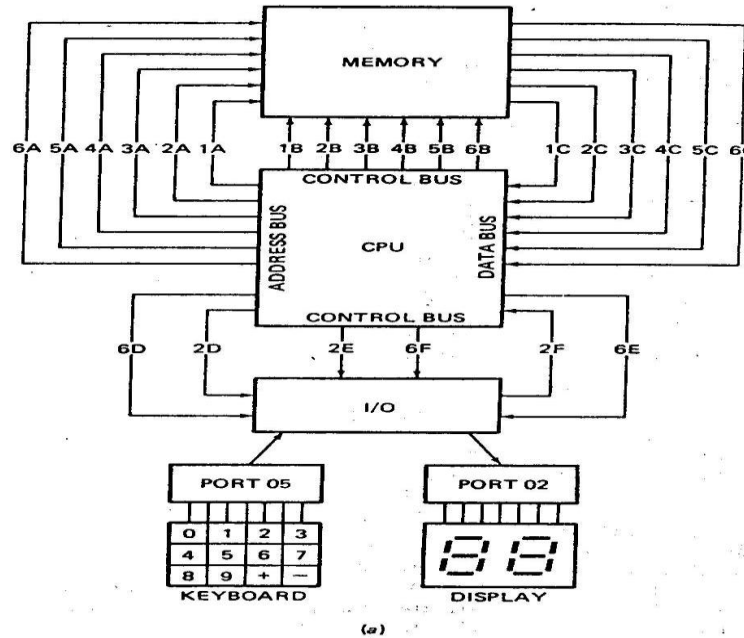
Pour un bon fonctionnement du microprocesseur; il doit y avoir des lignes de contrôle. Cette ensemble de lignes de contrôle est appelé Bus de contrôle. Le Bus de contrôle est constitué de 4 à 10 lignes. Le bus de contrôle est utilisé pour synchroniser les opérations du microprocesseur avec les périphériques externes. Des exemples de signaux de contrôle typiques sont : le Memory Read, le Memory Write, le I/O Read, le I/O Write

III- Principe de Fonctionnement de base d'un microprocesseur :

Exemple : l'exécution d'un programme de trois instructions par le microprocesseur.

Pour comprendre clairement comment le microprocesseur, la mémoire et les ports d'E/S fonctionnent ensemble comme tout système, nous allons décrire de simples actions dont l'ordre doit être suivi pour exécuter un programme simple. Les 3 instructions du programme sont:

- 1- Envoyer un nombre (une valeur) à partir d'un clavier connecté à un port dont l'adresse est 05H.
- 2- Additionner 7 à la valeur qui vient d'être lue à l'intérieur du microprocesseur.
- 3- Sortir le résultat vers un écran connecté à un port dont l'adresse est 02H.
- 4- La figure 2.2a montre les actions sous forme de schéma que l'ordinateur va réaliser pour exécuter ces 3 instructions.



Pour cet exemple on va considérer que le CPU va tirer les instructions et les données une à une. On considère aussi que les instructions, sous forme binaire sont dans des emplacements de mémoire continus (qui se suivent) commençant par l'adresse 00100H.

Le tableau suivant donnent les codes binaires qui seront nécessaires et sont dans des emplacements ou locations de mémoires successives pour exécuter le programme.

Adresse de la location ou emplacement de mémoire	Contenu de la mémoire (en binaire)	Contenu de la mémoire (en hexadécimal)	Opérations effectuées ou commentaires
00100H	11100100	E4	Faire entrer à partir de..
00101H	00000101	05	Port dont l'adresse est 05
00102H	00000100	04	Additionner avec
00103H	00000111	07	07H
00104H	11100110	E6	Faire sortir vers..
00105H	00000010	02	Port dont l'adresse est 02

Résumé des étapes :

Programme

- 1- entrer la valeur à partir du port 05H
- 2- ajouter 7 à cette valeur
- 3- Sortir le résultat à travers le port 02H

Support de cours Microprocesseurs 2010-2011

Étapes :	Opérations
1A	le CPU envoie l'adresse de la 1ère instruction à la mémoire
1B	le CPU envoie le signal de contrôle de lecture de la mémoire pour activer la mémoire
1C	L'instruction de 1 octet est envoyé de la mémoire vers le CPU à partir du bus de données
2A	Adresse l'emplacement de mémoire suivant pour obtenir le reste de l'instruction
2B	le CPU envoie le signal de contrôle de lecture de la mémoire pour activer la mémoire
2C	L'adresse du port e 1 octet est envoyée de la mémoire vers le CPU sur le bus de données.
2D	Le CPU envoie l'adresse du port sur le bus d'adresse
2E	le CPU envoie le signal de contrôle de la mémoire pour activer la mémoire
2F	Les données à partir du port sont envoyées vers le CPU sur le bus de données
3A	le CPU envoie l'adresse de l'instruction suivante à la mémoire
3B	le CPU envoie le signal de contrôle de la mémoire pour activer la mémoire
3C	L'instruction de 1 octet est envoyé de la mémoire vers le CPU à partir du bus de données
4A	Le nombre 07H est envoyé de la mémoire vers le CPU à partir du bus de données
4B	le CPU envoie le signal de contrôle de lecture de la mémoire pour activer la mémoire
4C	le CPU envoie le signal de contrôle de la mémoire pour activer la mémoire
5A	le CPU envoie l'adresse de l'instruction suivante à la mémoire
5B	le CPU envoie le signal de contrôle de lecture de la mémoire pour activer la mémoire
5C	L'instruction de 1 octet est envoyé de la mémoire vers le CPU à partir du bus de données
6A	le CPU envoie l'adresse de l'instruction suivante à la mémoire pour obtenir le reste de l'instruction
6B	le CPU envoie le signal de contrôle de lecture de la mémoire pour activer la mémoire
6C	L'adresse du port e 1 octet est envoyée de la mémoire vers le CPU sur le bus de données.
6D	Le CPU envoie l'adresse du port sur le bus d'adresse
6E	Le CPU envoie les données au niveau du port à partir du bus de données
6F	le CPU envoie le signal de contrôle d'écriture pour activer le port..

IV- Les opérations d'un microprocesseur

A partir de la discussion suivante, on peut résumer les opérations d'un microprocesseur de la manière suivante :

- 1- Le CPU tire (lit) les instructions ou les données de la mémoire en envoyant une adresse par le biais du bus d'adresse ; en envoyant un signal de lecture de mémoire par le biais du bus de contrôle. L'instruction ou la donnée concernée (adressée) est envoyée de la mémoire au CPU par le biais du bus de données.
- 2- Le CPU peut envoyer (écrire) des données au niveau de la mémoire en envoyant une adresse par le biais du bus d'adresse, en envoyant un signal d'écriture de mémoire par le biais du bus de contrôle. Les données concernées (adressées) vont être envoyées par le biais du bus de données vers la mémoire.
- 3- Pour lire ou recevoir des données à partir d'un port, le CPU va envoyer l'adresse du port par le biais du bus d'adresse ; va envoyer un signal de lecture de port d'E/S par le biais du bus de contrôle. Et les données provenant du port vont arriver au CPU par le biais du bus de données.
- 4- Pour écrire ou envoyer des données vers un port, le CPU va envoyer l'adresse du port par le biais du bus de données; va envoyer un signal d'écriture de port d'E/S par le biais du bus de contrôle. Et enfin va envoyer les données à écrire au niveau du port par le biais du bus de données.
- 5- Le CPU tire les instructions de chaque programme de manière séquentielle, décode l'instruction et l'exécute.

V- Les applications du microprocesseurs :

Les applications du microprocesseur sont nombreuses et augmentent de jour en jour.

Voici quelques exemples d'utilisation du microprocesseur

- Traitement de texte
- Le système de télétexte
- Les réservations d'avions
- Les applications industrielles et commerciales. Les systèmes basés sur le microprocesseur peuvent être utilisés comme le contrôle automatique d'une centrale électrique, le contrôle automatique de la vitesse des moteurs
- Les applications générales : Les systèmes basés sur le microprocesseur peuvent être utilisés pour contrôler la consommation de l'essence d'une voiture, les feux rouges, les températures, les robots etc....
- Gestion de base de données
- Vidéo conférence, caméra de surveillance

NB :

Les applications des microprocesseurs sont principalement de 2 catégories :

- les systèmes programmables
- les systèmes fixes

Dans les systèmes programmable comme les micro computers ou micro-ordinateurs, les SDK 85, le microprocesseur est utilisé pour le calcul et le traitement des données. Ces systèmes incluent des microprocesseurs universelles capables de traiter de très grandes

quantités de données, des périphériques de stockage (comme les disques durs et des CD ROM) ; des périphériques de sortie comme les imprimantes. Le PC (ordinateur) est une parfaite illustration des systèmes programmables.

Dans les systèmes préétablis et fixes, le microprocesseur fait parti d'un produit final et que l'utilisateur ne pourra pas le reprogrammer. Les microprocesseurs utilisés dans ces systèmes sont généralement catégorisés comme des **MICROCONTROLLEURS**.

Exemples de systèmes fixes, préétablis : les machines à laver, les contrôleurs de feux rouges, les testeurs automatiques.

Conclusion générale :

Les microprocesseurs et micro-ordinateurs renfermes des applications qui ont des conséquences grandissantes et un grand impact dans notre vie de tous les jours. Ils sont devenus incontournables pour une société moderne.

L'Architecture d'un microprocesseur :

Introduction:

Le Central Processing Unit (CPU) est la partie la plus importante d'un micro-ordinateur ou d'un système basé sur le microprocesseur (CPU).

Pour pouvoir programmer le système il faut nécessairement comprendre l'organisation physique interne du microprocesseur

Le Diagramme Schématique du CPU est montré par la figure suivante :

A. L U	Accumulateur
	Registres universels (ou à utilité générale)
	Registres spéciaux ou registres de contrôle
Unité de contrôle et de synchronisation	

En général le C.P.U est disponible sur circuit intégré et comprend cinq sections avec chaque section ayant une fonction bien précise

1. UAL (Unité Arithmétique et Logique)
2. Les registres universels ou à utilité générale
3. Les registres de contrôle ou spéciaux
4. Unité de contrôle et de synchronisation
5. L'Accumulateur

Chaque fabricant ou concepteur de microprocesseur a sa propre variété avec une architecture et une organisation spécifiques, mais les microprocesseurs peuvent avoir des caractéristiques communes et des modes de fonctionnement identiques.

L'architecture général d'un microprocesseur a été schématiquement représentée par la figure suivante.(voir figure)

(fig. 1.6).

I. L'Unité Arithmétique et Logique (U A L)

C'est au niveau de l'U.A.L que le microprocesseur effectue les opérations arithmétiques (Addition, Soustraction, Division etc....) et logiques. (Comparaison, AND, OR, EXOR etc...). L'unité peut être subdivisée en 3 composantes principales :

- l'Accumulateur
- le Régistre d'état
- l'UAL même.

1 L'Accumulateur

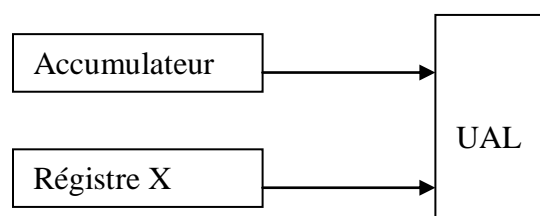


Fig 1.6 : Accumulateur

L'accumulateur est un registre spécial ou un emplacement temporaire de 8-bit qui est utilisé pour presque toutes les opérations arithmétiques et logiques. Durant l'addition de deux nombres de 8-bit, l'un des opérands doit être obligatoirement l'accumulateur; l'autre opérant peut être gardé au niveau de la mémoire ou dans l'un des autres registres. Les résultats des opérations arithmétiques et logiques effectuées par l'U.A.L sont gardés au niveau de l'Accumulateur. Au niveau du microprocesseur tous les données d'entrée / sortie passent par l'Accumulateur. Donc c'est un registre clé indispensable dont la taille correspond à la taille du mot de données (nombre du bus de données).

Au niveau d'un microprocesseur il y'a un registre temporaire appelé le registre X. Ce registre contient l'autre operant.

NB : Dans certains microprocesseurs il y a plus d'un Accumulateur.

2 Le Registre d'état ou (Statut Register) ou Flag (Drapeau)

Le registre d'état est aussi un registre spécial où la condition et l'état de la plus récente opération logique ou arithmétique est gardé. Il est aussi appelé un flag. Le flag contient des informations comme :

- la « parité » (Parity)
- la « retenue » (carry)
- «l'auxiliaire carry »
- le zéro
- le signe
- le « débordement » (Overflow)
- etc....

Les informations contenues dans les registres d'État sont utilisés au niveau des programmes pour les prises de décision, les boucles, les branchements.

Zéro : Si l'opération effectuée est nulle, le zéro flag est fixe a 1 sinon il est réinitialisé à 0.

Carry :Si la retenue générée a partir du M S B (Most significant bit ou bit qui le poids logique le plus fort) comme le résultat d'une certaine opération, le carry flag est fixe a 1 sinon il est réinitialisé a 0

Sign : Si le résultat d'une opération produit 1 comme M S B (Most significant bit ou bit qui le poids logique le plus fort) au niveau de l'Accumulateur, le signe flag est fixé à 1 sinon il est réinitialisée à 0.

Parity : Si le résultat d'une opération compte un nombre de bits pair ,le parity flag est fixé à 1 sinon il est réinitialisée à 0.

Auxiliaire carry : (half carry) : Si une opération produit une retenue (carry) à l'intérieur des premiers 4 bits de bas niveau logique (poids logique faible), l'auxiliaire flag est fixé sinon il est réinitialisé à 0. Il est utilisé pour l'arithmétique en B C D.

Overflow (Débordement) : la soustraction est effectuée en utilisant la représentation de complément par 2 (2nd complément) des nombres. Si le résultat d'une opération produit un débordement alors le flag overflow est fixé à 1 sinon il est réinitialisé à 0. Un débordement se produit quand le résultat d'une opération dépasse la longueur de l'Accumulateur

II. Les Registres Universels (d'utilité générale)

Les microprocesseurs ont un certain nombre de registres pour stocker des données de petite quantité appelées registres universels ou à utilité générale (General Purpose Régistre). Ces registres sont utilisés généralement comme second opérant (l'accumulateur est le 1er opérant) dans les opérations arithmétiques et logiques et aussi pour garder (stocker) les résultats immédiats. Ces registres sont accessibles aux utilisateurs.

III. Les Registres de Contrôle :

Pour pouvoir réaliser des opérations spéciales des registres spéciaux sont disponibles au niveau des microprocesseurs. L'accumulateur et le flag mais aussi

- le Program Counter (PC)
- le Stack Pointer (SP)
- le Régistre d'Index (IX)
- etc...
- le registre du Program Statut Word (PSW) ou le flag, le Pointeur de données, le Timer mode Selector (sélecteur de la mode de chronométrage) ou TMS, l'interruption Mask ou l'interrupteur de sélection de priorité etc....

Différents microprocesseurs ont différentes variétés de ces types de registres. Quelques-uns de ces registres les plus communs sont décrits ci-dessous.

1 Le Program Counter (PC) ou Compteur de Programme

Le Program Counter (PC) ou Compteur de Programme est en général un registre de 16-bit pour un microprocesseur de 8-bits. Les programmes ou les instructions à exécuter sont stockés de manière séquentielle au niveau de la mémoire. Le PC est utilisé pour cibler l'adresse de l'instruction suivante qui doit être tirée de la mémoire pour exécution, en d'autres termes le PC contient l'adresse de l'emplacement de la mémoire de la prochaine instruction à exécuter. C'est lui qui guide l'exécution des instructions.

2 Le Stack Pointer ou le Pointeur de la Pile

Le Stack Pointer (SP) est un registre de 16-bits pour un microprocesseur de 8-bits. Le Stack est une portion de mémoire (ou une des successions d'emplacements de mémoire) où des informations ou des données sont stockées sous forme de PILE ou LIFO (Last In First Out). Le Stack Pointer contient toujours l'adresse de l'emplacement de mémoire de la donnée à la tête de la pile (Stack).

Pour placer une donnée au niveau de la tête de la pile, on utilise l'instruction « PUSH » et au contraire tirer une donnée de la pile on utilise l'instruction « POP »

Le SP permet au microprocesseur de savoir où il s'était arrêté lorsqu'il se déplace entre programme principale et sous programme.

3 Le registre d'index

Le Régistre d'Index permet de s'adresser à un emplacement de mémoire de manière indirecte lorsqu'il pointe à cet emplacement de mémoire.

Exemple: Si l'emplacement de mémoire dont l'adresse est 2478H est chargé au niveau d(IX). Une instruction comme « MOV A, (IX) » va copier le contenu 2478H (pointée par le registre IX) au niveau de l'accumulateur.

IV. L'unité de contrôle et de synchronisation (timing and control unit) :

Cette unité contrôle toutes les unités internes et externes du microprocesseur. Elle synchronise les opérations entre unités pour éviter des collisions ou des dysfonctionnements. Cette unité a un certain nombre d'entrées et sorties externes appelées bus de contrôle. Ce sont l'entrée du Clock Pulse; la sortie du Clock Pulse; le Reset In; le Reset Out; le Interrupt In ;le Interrupt Out; le Wait In; le Hold In; le Hold Acknowledgement Out ;le Read In; le Synchronisation Out; le Read Out; le I/O Request; le Memory Request etc.... Toutes ces entrées ou sorties ne sont pas disponibles dans tous les microprocesseurs. Elles diffèrent selon le type de microprocesseur.

CONCLUSION

Les microprocesseurs et les microcontrôleurs trouvent des applications de plus en plus dans chaque sphère de la vie de tous les jours. Les caractéristiques des différents microprocesseurs varient de manière importante, certains sont plus favorables que d'autres pour des applications particulières. Quelques exemples typiques d'applications sont les jeux, les équipements électroménagers, les instruments et terminaux intelligents, la prise de décision, la conception, le contrôle, l'automatisation, la communication, les robots etc.....

L'Architecture du Microprocesseur 8085

Introduction

Par la conception de leur architecture, il y a plusieurs types de microprocesseurs. Si le concept de base est à peu près la même chose ; ils diffèrent par leur architecture, leur opération et leur instructions (types de recommandes). Donc il serait illusoire d'essayer d'étudier tous les types de microprocesseur. Mais une bonne compréhension d'un d'entre eux peut aider à comprendre les autres.

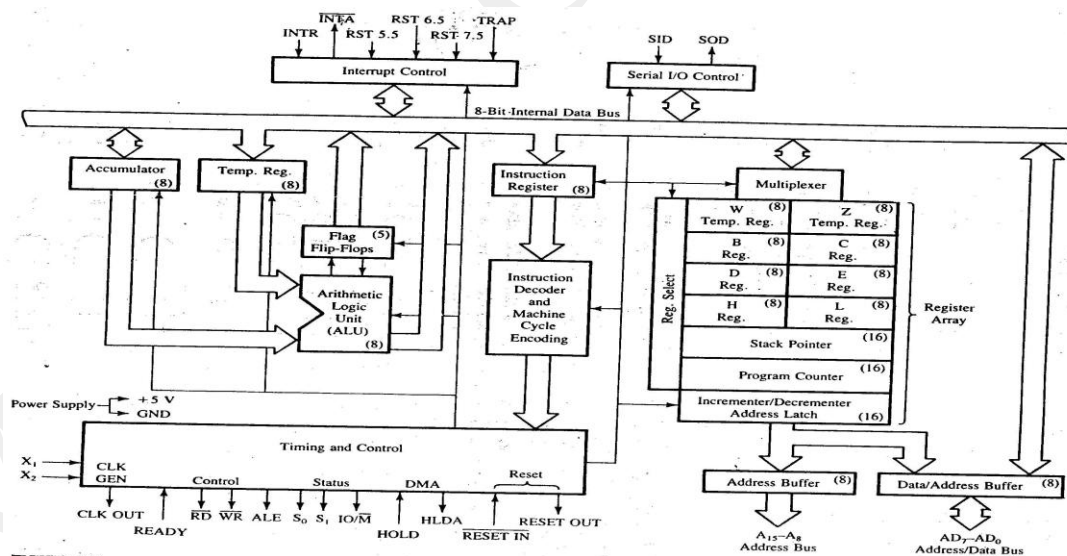
Pour ce cours nous avons choisit l'un des microprocesseurs les plus populaires : INTEL 8085. C'est un Microprocesseur de 8-bit et son architecture est simple.

I. L'Architecture du 8085 :

INTEL 8085 est un microprocesseur de 8-bit. C'est un circuit intégré de 40 pieds (pins). Il comporte une alimentation unique de 5 V DC pour son opération. Son Clock pulse Impulsion d'horloge) est de près de 3 MHZ. Il compte 80 instructions de base et 246 opcodes.

La figure 2 1 et 3.7 montrent l'architecture interne du 8085 avec 3 sections importantes :

- une Unité Arithmétique et Logique (U.A.L)
- une Unité de Contrôle et de synchronisation
- et plusieurs registres.



1 L'Unité Arithmétique et Logique :

L'Unité Arithmétique et logique a la fonction de calculatrice. Elle effectue les opérations arithmétiques et logiques. Elle effectue les opérations arithmétiques et logiques suivants:

- Addition
- Soustraction
- AND
- Exclusif OR
- Complément (NOT)
- Etc ...

Elle est composée de l'accumulateur, des registres temporaires, des circuits arithmétiques et logiques et du registre flag.

Pour réaliser toutes ses opérations, l'accumulateur est toujours l'un des opérants. Le résultat des opérations est aussi gardé au niveau de l'Accumulateur. Le registre Accumulateur est désigné par A. L'autre opérant est soit un registre temporaire ou un emplacement de mémoire.

2 Les registres

La figure 2.1 et 3.7 montre les registres d'Intel 8085, qui sont les suivants:

- Un accumulateur de 8-bit (registre A)
- Six registres universels ou à utilité générale: les registres B, C, D, E, H et L.
- Un Stack Pointer ou SP de 16-bit.
- Un Program Counter ou PC de 16-bit.
- Un Registre d'instruction
- Un Registre de statut.
- Un registre temporaire.

a. Les registres universels ou à utilité générale

Le 8085 contient 6 registres universels ou à utilité générale de 8-bit. Les registres sont désignés par B, C, D, E, H, et L. Ils peuvent être utilisés pour manipuler des données de 8-bit ou de 16-bit. Pour des données de 16-bit, 2 de ces registres sont combinés en paire et appelé Registre Paire. Les Registres Paires qui sont valides pour le 8085 sont BC, DE et HL.

Ces registres universels sont programmables, ce qui veut dire qu'un programmeur peut les utiliser pour déplacer des données à partir de ces registres en utilisant des instructions ou des commandes.

Exemple: L'instruction MOV B,C copie la donnée au niveau du registre C vers le registre B.

b. Le Stack Pointer ou SP :

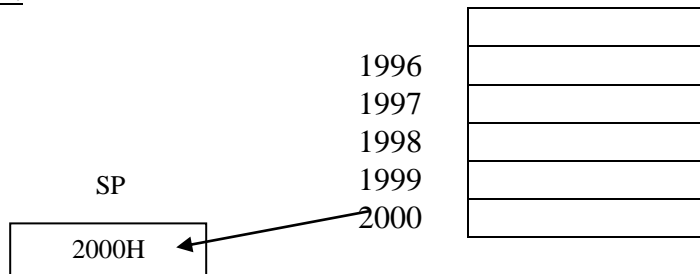
Le stack Pointer est un registre de 16-bit utilisé pour stocker ou pointer à l'adresse à la tête de la pile.

Avec l'instruction PUSH, une donnée de 2 octets est envoyée et gardée au niveau de la pile et le SP indique ou pointe à l'adresse du dernier octet envoyé au niveau de la pile.

Avec une opération PUSH, le SP est d'abord décrémenté, et l'octet de haut niveau logique est envoyé (poussé) dans la pile; ensuite, le SP est encore décrémenté et l'octet de bas niveau logique de la donnée est envoyé dans la pile. Dans une opération de POP, c'est l'octet de bas niveau logique qui est d'abord retiré et le SP est

incrémenté, ensuite l'octet de haut niveau logique est retiré et le SP est encore incrémenté.

Exemple :



c. Le Program Counter (compteur de programme) ou PC.

Le PC est un registre de 16 bit. Ce registre comme le SP est un pointeur de mémoire. Les emplacements de mémoire sont des adresses de 16-bit; c'est pourquoi c'est un registre de 16 bit.

Le microprocesseur utilise ce registre pour ordonner l'exécution des instructions. La fonction du PC est de pointer à l'adresse de l'emplacement de mémoire où la prochaine instruction doit être retirée. Le PC n'est pas accessible au programmeur ou à l'utilisateur.

Exemple:



d. Le registre d'état ou de statut (Flag) :

Le flag est un registre de 8-bit dont 5 utilisés. L'organisation du Flag est monté ci-dessous:

D7	D6	D5	D4	D3	D2	D1	D0
Sign Flag (S)	Zero flag (Z)		Auxiliary flag (AC)		Parity Flag (P)		Carry Flag (CY)

Le microprocesseur utilise le flag pour effectuer les opérations de test sur l'aspect des données.

Carry (CY): Le Carry flag est fixée à 1 lorsqu'il y a une retenue ou carry au cours d'une opération arithmétique ou logique (débordement). Sinon il est fixée à 0.

Parity (P): Le Parity flag est fixée à 1 lorsque le résultat d'une opération contient un nombre pair de 1. Sinon il est fixée à 0.

Auxiliary (AC) : L'Auxiliary flag est fixée à 1 lorsqu'un carry ou retenue est généré entre le 3^{ème} bit et le 4^{ème} bit lors d'une opération arithmétique ou logique. Sinon c'est fixé à zéro.

Zero(Z) : Le Zero flag est fixé à 1 lorsque le résultat d'une opération arithmétique ou logique est zéro. Sinon elle est fixée à zéro.

Sign (S) : Le Sign Flag est fixée à 1, si le bit le plus significatif (qui a le plus de poids logique) lors d'une opération arithmétique et logique est à 1 sinon il est fixé à zéro.

Exemple : considérons que l'Accumulateur contient la donnée CB et le registre B contient E9. Ajoutons CB à E9.

Et vérifions les conditions du flag.

CB = 11001011

E9 = 11101001

110110100

S	Z		AC		P		CS
1	0		0		1		1

NB comme constatée, au niveau du registre d'état seuls 5 bits sont définis. Les autres 3 bits ne sont pas définis. Mais parfois tous les 8 bits peuvent être combinés et utilisés pour d'autres instructions. Dans ce cas le flag (5 bits + 3 bits) est appelé PSW (Program Status Word).

Parfois le PSW et l'Accumulateur peuvent être associés comme un registre de 16 bits et utilisés dans des opérations de Stack (Pile) voir schéma

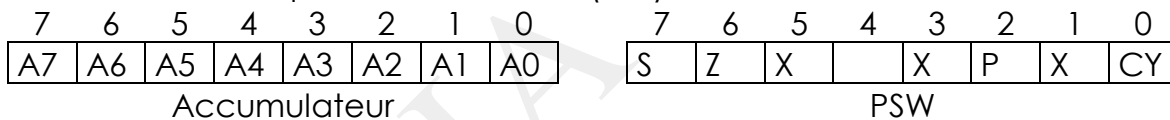


fig. : Organisation du PSW

e. Le Registre d'Instruction (IR) :

le Registre d'Instruction est un registre de 8-bit. L'instruction (à exécuter) est d'abord tirée de l'emplacement de mémoire pour être placée au niveau du Registre d'Instruction.

C'est à partir du Registre d'Instruction que l'unité de décodage; décode l'instruction en plusieurs étapes et l'Unité de Contrôle génère les signaux nécessaires pour son exécution.

Le Registre d'Instruction n'est pas programmable et n'est pas accessible à travers des instructions.

3 Les bus de données et d'adresses.

le Bus d'Adresse est un groupe de 16 lignes ou conducteurs généralement désignés de A0 à A15. Le Bus d'Adresse est unidirectionnel; les bits circulent dans un seul sens à partir du microprocesseur vers les périphériques. Le microprocesseur utilise le bus

d'adresse pour effectuer sa première fonction: identifier un périphérique ou un emplacement de mémoire.

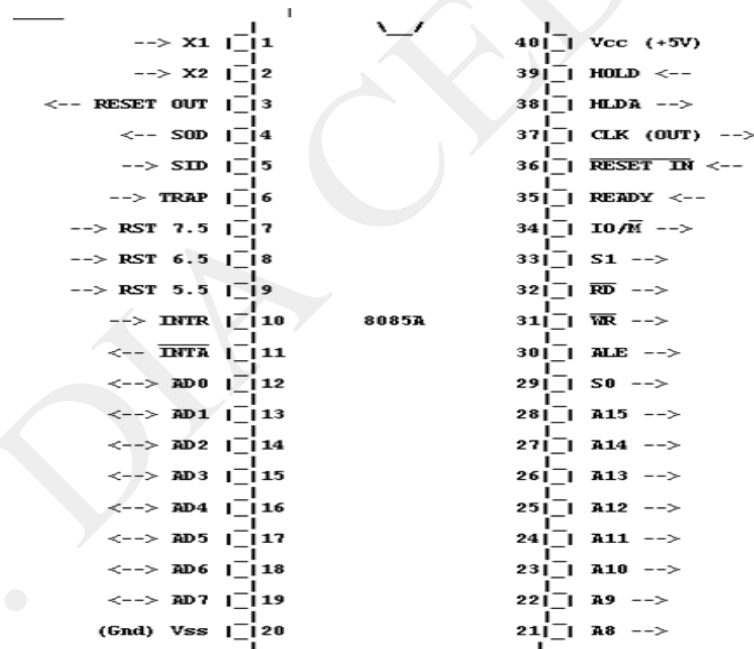
Le microprocesseur 8085 a un bus d'adresse de 16-lignes ou conducteurs, donc capable d'adresser $2^{16}=65536$ (plus connu comme 64k) d'emplacements.

Intel 8085 est un microprocesseur de 8-bit, son Bus de données est un groupe de 8 lignes ou conducteurs, ce qui veut dire que 8 bits de données peuvent être transmis en parallèle en même temps à partir du microprocesseur.

II. La Configuration des Pins (Pieds) du 8085 :

La figure 3.1 et figure 2.2 montre les pieds (pins) logiques du microprocesseur 8085. Les signaux peuvent être divisés en six groupes:

- (1) Bus d'Adresse
- (2) Bus de Données
- (3) les signaux de contrôle et de statut
- (4) les signaux unités de l'extérieur
- (5) les ports d'entrée et de sortie.
- (6) les signaux de l'alimentation et de la fréquence



1. Le Bus d'Adresse

A₈ – A₁₅ : Les pieds A₈ - A₁₅ représentent le Bus d'Adresse et sont utilisés pour l'adresse des 8 bits les plus significatifs (plus de poids logique) d'une mémoire ou d'un périphérique d'E/S.

2. Le Bus de données

AD₀ – AD₇ : Les pieds AD₀ – AD₇ représentent le Bus de données et sont utilisés pour recevoir ou envoyer des données d'une mémoire ou d'un périphérique d'E/S.

NB: le bus de données et d'adresse sont multiplexés au niveau de AD0 – AD7

3. Les signaux de contrôle et de statut

Le groupe de signaux inclus 2 signaux de contrôle \overline{RD} et \overline{WR} , trois signaux de statut (IO/M, S1 et S0) pour identifier la nature de l'opération. Et un signal spécial (ALE) pour indiquer le début de l'opération.

- RD et WR
- IO/M
- S1 et S0
- ALE

ALE: Address Latch Enable. Le signal ALE est soit de niveau 0 ou de niveau 1. Quand il est de niveau 1; il indique que les bits sur le bus AD7- AD0 sont des bits d'adresse. Mais quand ALE est de niveau 0, AD7 – AD0 est utilisé comme Bus de données.

\overline{RD} : C'est un signal de contrôle de lecture. Il est activé quand il est de bas niveau logique. Quand il est de bas niveau logique, il indique que la mémoire ou les périphériques d'E/S sélectionnés sont à lire.

\overline{WR} : C'est un signal de contrôle d'écriture activé quand il est de bas niveau logique. Quand il est de bas niveau logique, il indique que les données au niveau du bus de données sont à écrire ou envoyer vers la mémoire où vers les périphériques d'entrées et sorties sélectionnés.

IO / M: C'est un signal de statut qui doit distinguer et différencier les opérations destinées à la mémoire et les opérations destinées aux périphériques d'entrées et sorties. Quand il est de Haut niveau logique il indique une opération de périphériques d'E/S et quand il est de bas niveau logique, il indique une opération de mémoire.

S1 et S2: Ces signaux sont des signaux de statiques similaires au signal IO / M. Ils permettent d'identifier diverses types d'opération comme montré au niveau du tableau suivant. Mais ils sont rarement utiliser pour les petits systèmes.

4. Les signaux initier de l'extérieur et les interruptions

Ready (input): Ce signal est utilisé pour prolonger le temps d'écriture et de lecture du Microprocesseur. Pour permettre à un périphérique lent d'être prêt à envoyer où recevoir des données. Si Ready est de haut niveau logique(1); ceci veut dire que le périphérique est prêt. Mais si Ready est désactivé (0) le microprocesseur attend jusqu'à ce qu'ils redeviennent actif.

Hold (Input): Ce signal indique (quand il est de haut niveau logique) qu'un autre périphérique (Exemple un DMA controller) demande à utiliser le Bus d'adresse ou de données. Quand le microprocesseur reçoit un signal Hold, il libère les Bus dès que l'exécution de l'instruction courante est terminée. Le microprocesseur recommence à utiliser les bus à la fin du signal.

HOLD (Output): Ou Hold Acknowledgment. Ceci est un signal de réponse au signal Hold. Après le retrait du signal Hold ; HOLD est désactivé (0).

INTR (Input) : Ou Interrupt Request: C'est un signal utilisé lors d'une demande d'une interruption. Quand INTR est actif, le microprocesseur suspend l'exécution normale du programme pour répondre à l'interruption

INTA (Output) : Interrupt Acknowledgment: Utilisé pour répondre au signal à une interruption. INTR par le microprocesseur après la réception de cette dernière.

RST 5.5-- RST 6.5 – RST 7.5 et TRAP : Ou Restart Interrupt. Ce sont des signaux d'interruption. Quand ils sont utilisés, il transfère le contrôle du programme vers un emplacement de mémoire spécifique.

Reset In (Input) : Quand le RESET IN est de bas niveau; il remet le Program Counter à zéro. Et le microprocesseur est réinitialisé.

Reset Out (Output): Le signal indique que le microprocesseur est entrain d'être réinitialisé. Ce signal peut être utilisé pour réinitialiser les autres périphériques.

5. Les Ports Séries d'E/S

Le 8085 comporte 2 signaux. de communication en série qui sont : SID (Serial Input Data) et SOD (Serial Output Data).

SID (Input) : C'est une ligne ou pin d'entrée où des données peuvent être envoyées au microprocesseur en série (entrée en série). La donnée reçue à partir de SID est envoyée et stockée au niveau du 7^{ème} bit de l'accumulateur en utilisant l'instruction RIM.

SOD : C'est une ligne ou pin de sortie de donnée par où le microprocesseur envoie des données en série vers d'autres périphériques (sortie en série). Donc le contenu du 7^{ème} bit de l'accumulateur est transféré à la sortie en utilisant l'instruction SIM.

6. L'alimentation

Les signaux de l'alimentation et de la fréquence sont :

Vcc : Alimentation + 5V.

Vss : Relié à la terre.

X1 X2: Ce sont des terminaux où doit être connecté un circuit oscillatoire (RL ou LC) dont la fréquence va déclencher le circuit intérieur du microprocesseur pour produire un clock pulse nécessaire à son fonctionnement.

CLK (OUT) : C'est un clock pulse de sortie du microprocesseur qui a la même fréquence que le clock pulse interne du microprocesseur. Les utilisateurs peuvent l'utiliser comme système de clock pulse pour d'autres périphériques.

Programmer le 8085 : Introduction Générale

Introduction:

Un ordinateur ne peut faire que ce que le programmeur lui demande de faire. Pour effectuer une tâche particulière le programmeur écrit une séquence d'instructions appelée programme. Le programme est gardé au niveau de la mémoire RAM. Le processeur prend les instructions une à une à partir de la RAM et les exécute. Il exécute toutes les instructions dans la RAM pour produire le résultat voulu.

I. Les types de langages

1- Le langage machine :

un ordinateur utilise les nombres binaires pour ses opérations et ne comprend que les informations composées de **0s** et de **1s**. Les programmes écrits sous la forme de 0s et de 1s sont appelés les PROGRAMMES DE LANGAGE MACHINE (MACHINE LANGUAGE PROGRAMME).

Exemple :

Au niveau de l'exemple suivant ou deux nombres placés dans les emplacements de mémoire 2501 à 2503 sont additionnées et le résultat gardé au niveau de l'emplacement 2503.

DONNÉS :

Adresse mémoire

0010010100000001

0010010100000010

Données

00010101

00100000

PROGRAMME

Codes machines	Commentaires
00100001	Obtient l'adresse du 1 ^{er} nombre c a d 2501H dans le registre paire HL
00000001	
00100101	
01111110	Obtient le 1 ^{er} nombre dans l'accumulateur
00100011	Incrémenter le contenu du registre paire HL
10000110	Ajouter le 1 ^{er} nombre au 2 ^{ème} nombre
00110010	Garder le résultat dans 2503H
00000011	
00100101	
01110110	Stop

Inconvénients

- Difficile à comprendre et à vérifier
- Les programmes sont longs
- L'écriture du programme est difficile et épuisante

- Beaucoup d'erreurs d'inattention

Écrire les codes machine avec le système Hexadécimal:

Pour rendre les choses moins difficiles pour le programmeur, les codes machines peuvent être écrits avec le système hexadécimal. Ainsi les erreurs peuvent être plus faciles à détecter.

PROGRAMME

Codes machines	Commentaires
21	Obtient l'adresse du 1 ^{er} nombre c a d 2501H dans le registre paire HL
01	
25	
7 ^E	Obtient le 1 ^{er} nombre dans l'accumulateur
23	Incrémenter le contenu du registre paire HL
86	Ajouter le 1 ^{er} nombre au 2 ^{ème} nombre
32	Garder le résultat dans 2503H
03	
25	
76	Stop

DONNES :

Adresse mémoire

2501

2502

Données

0F

14

2- Le langage Assembleur

Bien que les instructions peuvent être écrites en langage hexadécimal, c'est encore assez difficile de comprendre un programme écrit en hexadécimal. Ainsi les fabricants de microprocesseurs, ont développé des langages faciles à comprendre en utilisant des **symboles en alphanumérique** au lieu de 0s et de 1s. Chaque code en alphanumérique est appelé **MNEMONIC** (qui veut dire en GREC compréhensible). Des exemples de ces codes sont **ADD** pour l'addition, **SUB** pour la soustraction ou **CMP** pour la comparaison etc.... Un programme écrit à l'aide de MNEMONICS est appelé LANGAGE ASSEMBLEUR (ASSEMBLY LANGUAGE).

Exemple :

Un exemple de programme en langage assembleur pour l'addition de 2 nombres placés dans 2 emplacements de mémoires successifs 2501H et 2502H est montré ci-dessous. La somme est gardée au niveau de l'emplacement de mémoire 2503H

PROGRAMME

Mnemonics	Opérant	Commentaires
LXI	H, 2501H	Obtient l'adresse du 1 ^{er} nombre c a d 2501H dans le registre paire HL
MOV	A, M	Obtient le 1 ^{er} nombre dans l'accumulateur
INX	H	Incrémenter le contenu du registre paire HL
ADD	M	Ajouter le 1 ^{er} nombre au 2 ^{ème} nombre
STA	2503H	Garder le résultat dans 2503H
HLT		STOP

DOONNEES

2501 – 49H

2502 – 56H

RESULTAT

2503

Remarque:

- Chaque microprocesseur a un langage assembleur qui lui est propre.
- De manière évidente un programme écrit avec un langage Assembleur doit être traduit ou convertit en langage machine avant d'être exécuté. La conversion est faite à l'aide d'un logiciel. Un programme ou logiciel qui convertit ou traduit un programme écrit en langage assembleur sous forme de programme en langage machine est appelé **ASSEMBLEUR**.

Avantage du langage assembleur :

L'avantage du langage assembleur est que l'exécution des programmes prend moins de temps comparé aux programmes sous forme de langage évolué (voir ci-dessus)

Inconvénients :

- Difficile et prend beaucoup de temps
- Le langage assembleur est orienté vers la machine (ordinateur). Ceci veut dire que le programmeur doit connaître en détail l'architecture du micro-ordinateur : ses registres, ses instructions les connexions des équipements aux ports etc....
- Le programme du langage assembleur n'est pas portable c a d un programme écrit pour tel micro-ordinateur ne peut pas être utilisé sur un autre micro-ordinateur ; Chaque ordinateur a son propre langage assembleur.

3- Le langage évolué

Les difficultés liées aux langages assembleurs peuvent être dépassés en utilisant des langages de programmations qui tendent à être « machine - indépendant » appelés **LANGAGES EVOLUES**. Les Exemples de langages de programmation évolués sont : BASIC, PASCAL, C, FORTRAN etc.... Les instructions écrites dans les langages évolués sont appelées **DECLARATIONS**. Les langages évolués contiennent un ensemble de lois et de symboles tirés de l'Anglais ou d'une autre langue

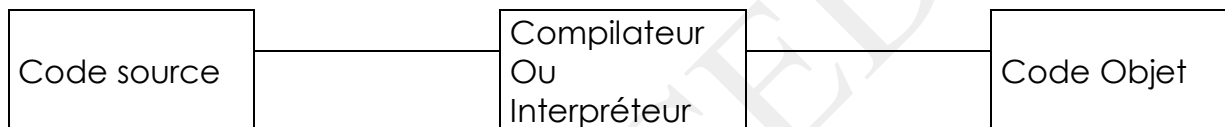
internationale. Ces langages sont orientés vers la procédure et non vers la machine (ordinateur). Lorsqu'un programme est écrit sous forme de langage évolué, on a pas besoin de connaître l'architecture de l'ordinateur. Et un programme écrit à partir d'un type d'ordinateur peut être facile à exécuter à partir d'un autre type d'ordinateur (il est portable).

Exemple de langage évolué comme Basic.

```
Input a
Input b
Let c = a + b
Print c
End
```

Bien sur on a besoin aussi de convertir le programme sous forme de langage évolué en langage ou code machine. Ici un autre programme appelé **COMPILATEUR** ou **INTERPRETEUR** est utilisé pour convertir ou traduire un langage évolué en code ou langage machine pour l'opération de l'ordinateur.

Chaque langage de programmation évolué a son propre compilateur ou interpréteur.



II. Classification des instructions du 8085

La totalité des instructions que le microprocesseur 8085 peut exécuter est appelé l'ensemble des instructions du 8085. Un programmeur peut écrire un programme sous forme de langage assembleur en utilisant ses instructions. L'ensemble des instructions du 8085 pourrait être divisé en cinq groupes :

- le groupe des opérations de transfert (copie) de données
- le groupe des opérations arithmétiques
- le groupe des opérations logiques
- le groupe des opérations de branchement (boucles)
- le groupe des opérations de contrôle d'entrées / sorties et de machines.

1- Le groupe de transfert de données :

Ce groupe de transfert de données copie les données à partir d'un emplacement appelé SOURCE vers un autre emplacement appelé DESTINATION.

- Exemple:

```
MOV
IN
OUT
LXI
```

2- Le groupe Arithmétique :

Ces instructions effectuent des opérations arithmétiques comme l'addition, la soustraction, l'incrémentation et la décrémentation.

- Exemple:

ADD

SUB

INR

DCR

3- Le groupe logique :

Ces instructions effectuent plusieurs opérations logiques avec le contenu de l'accumulateur.

Exemple:

- - AND, OR, EX-OR : ANA, ORA, XRA
- - COMPARE : CMP
- - COMPLEMENT : CMA

4- Le groupe de branchement :

Ce groupe d'instructions perturbe l'exécution ou la progression normale d'un programme de manière inconditionnelle ou conditionnelle.

Exemple:

- - JUMP :
- - CALL
- - RETURN
- - RESTART

5- Le groupe de machine :

Ces instructions contrôlent le fonctionnement de la machine.

Exemple:

- - HALT :
- - CALL
- - etc....

III. Le Format des instructions :

Une instruction est une commande . Chaque instruction compte deux parties : la première partie qui contient la tâche à effectuer appelée OPERATION CODE OU OPCODE (code de l'opération) et la deuxième est la donnée sur laquelle on agit et est appelée OPERANT.

Exemple: MVI B, 31H

Selon la nature de l'opérant, la taille du mot ou le nombre d'octets de l'instruction nous avons la classification suivante.

- une instruction de 1 mot ou de 1 octet (byte)
- une instruction de 2 mots ou de 2 octets (bytes)

- une instruction de 3 mots ou de 3 octets (bytes)

a- Les instructions d'un octet :

Une instruction composée d'un seul octet met l'opcode et l'opérant dans un seul octet.

Exemple :

MOV C, A

ADD B

CMA

Chaque instruction nécessite **un emplacement de mémoire**.

b- Les instructions de 2 octets :

Dans une instruction composée de 2 octets, le 1^{er} octet représente l'opcode et le second octet représente l'opérant

Exemple :

MVI A, 32H

ADI 07

OUT 02

Une **instruction de 2 octets** qui nécessite **2 emplacements de mémoire** pour être gardé au niveau de la mémoire.

c- Les instructions de 3 octets :

Dans une instruction composée de 3 octets, le 1^{er} octet représente l'opcode, et les deux autres octets suivant représentent une adresse de 16-bit. Il faut bien noter que le 2nd octet est la première partie de l'adresse et le 1^{er} octet est la seconde partie de l'adresse

Exemple :

JMP 2085H

LHI H, 4080H

Une **instruction de 3 octets** qui nécessite **3 emplacements de mémoire** pour être gardé au niveau de la mémoire.

IV. Les modes d'adressage des instructions

Dans cette instruction la source peut être un registre, un port d'entrée, ou bien un nombre de 8-bit. Les sources et les destinations sont en fait des opérants. Les différentes manières ou formes d'indiquer des opérants sont appelées

MODES D'ADRESSAGE DES INSTRUCTIONS. L'ensemble des instructions du 8085 contient les modes d'adressage suivants :

- l'adressage registre
- l'adressage immédiat
- l'adressage registre indirect
- l'adressage implicite

1- L'adressage registre :

Dans le mode d'adressage registre, les opérants sont des registres universels ou d'utilité générale (A, B, C, D, E, H, H)

Exemple : MOV A, B
 ADD B

2- L'adressage immédiat :

Dans le mode d'adressage immédiat, l'opérant est précisée à l'intérieur de l'instruction même, la donnée qui agit comme opérant est explicitement précisé dans l'instruction même.

Exemple : MV A, 05
 ADI A, 06
 LDA 9542

3- L'adressage registre indirect:

Dans le mode d'adressage registre indirect, l'**adresse** de l'opérant est précisée par un registre pair.

Exemple : LXI H, 2500H LXI H, 2500
 ADD M MOV A, M

4- L'adressage implicite :

Dans le mode d'adressage implicite, l'opérant est sous entendu

Exemple : Ces instructions où l'accumulateur n'est pas cité.

CMA
RAL
RAR
etc...

V. Comment écrire, assembler et exécuter un programme simple :

Un programme est une séquence d'instructions exécutées pour demander à l'ordinateur d'effectuer une tâche bien précise. Pour écrire un programme il faut diviser le problème en petites étapes en fonction des opérations que le 8085 peut effectuer. Ensuite, il faut traduire ces étapes en instructions. Voici un exemple pour écrire un programme simple pour l'addition de deux nombres avec le langage du 8085.

1- Énoncé du problème :

Écrire les instructions nécessaires pour charger les nombres hexadécimaux 32H et 48H dans les registres A et B respectivement. Additionner les nombres et afficher la somme au niveau du LED connecté au port de sortie PORT1.

2- Analyse du problème :

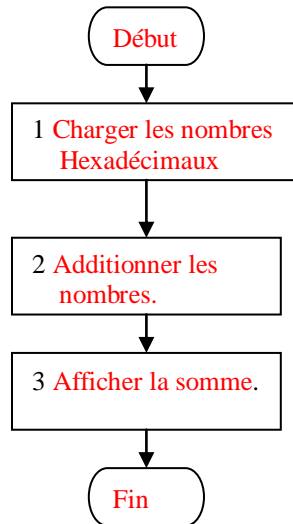
Même si ceci est un simple problème ; il est nécessaire de diviser le problème en petites étapes pour connaître et examiner la procédure d'écriture d'un programme. En général la manière dont le programme est énoncé donne des indications sur les étapes nécessaires. Ces étapes sont les suivantes :

- 1- Charger les nombres dans les registres
- 2- Additionner les nombres

- 3- Afficher la somme au niveau du port de sortie PORT1
- 4- fin

3- L'organigramme :

Les étapes décrites au niveau de l'analyse du problème et la séquence (l'ordre) peuvent être représentées par un schéma ou diagramme appelé organigramme. Ceci est montré par la figure ci-contre .



4- Le programme en langage assembleur :

Pour écrire le programme en langage assembleur, nous avons besoin de traduire les blocs montrés dans l'organigramme en des opérations du 8085 et puis en **Mnémoniques**.

Bloc 1	<u>MVI A, 32H</u>	Charger 32H dans le registre A
	<u>MVI B, 48H</u>	Charger 48H dans le registre B
Bloc 2	<u>ADD B</u>	Additionner les 2 octets et garder la somme dans le registre A
Bloc 3	<u>OUT 01H</u>	Afficher le contenu de l'accumulateur au port 01
Bloc 4	<u>HLT</u>	fin

5- Du langage assembleur au code hexadécimal (Hex code)

Pour convertir les mnémonics en code hexadécimal (Hex code) nous avons besoin de consulter les codes au niveau de l'ensemble des instructions du 8085. Ceci est appelé **ASSEMBLER MANUELLEMENT**.

Mnémonics	Hex code	commentaires
MVI A, 32H	3E	Charger 32H dans le registre A
	32	
MVI B, 48H	06	Charger 48H dans le registre B
	48	
ADD B	70	Additionner les 2 octets et garder la somme dans le registre A
OUT 01H	D3	Afficher le contenu de l'accumulateur au port 01
	01	
HLT	76	Fin

6- Écriture (chargement) dans la mémoire et conversion du code hexadécimal en code binaire :

Pour écrire ou entrer le programme au niveau de la mémoire RAM d'un micro-ordinateur et afficher le résultat, nous devons **CONNAITRE LES ADRESSES DE LA MEMOIRE ET L'ADRESSE DU PORT DE SORTIE**. Considérons que la mémoire RAM s'étale sur 2000H à 20FFH et que le système a un port de sortie de LED avec une adresse 01H.

Adresse Memoire	Hex code
2000	3E
2001	32
2002	06
2003	48
2004	70
2005	D3
2006	01
2007	76

7- Exécuter le programme :

Pour exécuter le programme nous avons besoin de dire au microprocesseur par où le programme commence en entrant l'adresse de mémoire 2000H. Puis on presse sur la touche EXECUTE pour débiter l'exécution du programme. Dès que la touche EXECUTE est pressée, le microprocesseur va charger 2000H au niveau du PC (Program counter) et le contrôle de programme est transféré du programme du moniteur à notre programme.

Le microprocesseur commence à lire les codes machines un à un et les exécute jusqu'à ce qu'il rencontre l'instruction HLT.

VI. Survole de l'ensemble des instructions du 8085 :

Pour pouvoir développer des logiciels et écrire des programmes pour contrôler des systèmes ; une bonne compréhension de l'ensemble des instructions du 8085 est essentielle. Pour Intel 8085 il y a 74 codes d'opération résultant sur 246 instructions. Ces instructions peuvent être regroupées de la manière suivante selon leur type :

Les notations suivantes sont utilisées dans la description des instructions :

R = registre de 8-bit du 8085 (A, B, C, D, E, H, L)

M= registre mémoire (emplacement)

RS= Registre source (A, B, C, D, E, H, L)

Rd= registre destination (A, B, C, D, E, H, L)

Rp= Registre pair (BC, DE, HL, SP)
() = contenu de

(Voir photocopie)

VII. Comment reconnaître le nombre d'octets qui compose une instruction :

Comme on commence à être familier avec les instructions, et commence à les utiliser dans des programmes qui doivent être assemblés manuellement. Pour bien faire l'assemblage manuel, nous avons besoin de connaître le nombre d'octets nécessaires pour chaque instruction.

En généra, il n'existe pas de règles établies pour connaître le nombre d'octets dans une instruction ; mais il existe quelques guides pouvant être décelés en examinant l'ensemble des instructions.

Les instructions d'un octet

Nous avons une instruction de 1 octet quand les opérants ne sont que des registres.

Exemple: Mov A, B
ADD C

Les instructions de 2 octets :

Nous avons une instruction de 2 octets quand les opérants contiennent un octet.

Exemple :

- MVI A, 20H
- ADI 28H

Les instructions de 3 octets :

Nous avons une instruction de 3 octets quand un des opérants est de 2 octets comme une adresse.

Exemple :

- LXI H, 3000H
- LDA 579FH

Introduction aux instructions du 8085 :

Introduction:

Chaque instruction dans le programme est une commande en binaire au microprocesseur pour effectuer une tâche ou une opération. Il n'est pas nécessaire de connaître toutes les instructions du 8085 pour comprendre les programmes simples. En essayant de comprendre les instructions simples qui sont fréquemment utilisées, le débutant peut comprendre de nouvelles instructions en apprenant les programmes. Dans ce chapitre nous introduisons les instructions de bases.

I. Les opérations de transfert de données :

Instructions : Les instructions de transfert de données copient les données à partir d'une source vers une destination sans modifier le contenu de la source. Le contenu précédent de la destination est remplacé par le contenu de la source.

Remarque importante : Les instructions de transfert de données n'affectent pas les flags

Opcod e	Opérant	Exemple
<u>MOV</u>	<u>Rd, Rs</u>	<u>MOV B,C</u>
<u>MVI</u>	<u>Rd, 8-bit</u>	<u>MOV B, 87H</u>
<u>OUT</u>	<u>8-bit, adresse du port</u>	<u>OUT 02H</u>
<u>IN</u>	<u>8-bit, adresse du port</u>	<u>IN 00H</u>
<u>HLT</u>		<u>Hlt</u>

Exemple 1:

Charger l'accumulateur avec l'octet de donnée 82H et sauvegarder la donnée dans le registre B.

Instructions MVI A, 82H
MOV B, A

Exemple 2 :

Écrire les instructions nécessaires pour lire les constants ON/OFF connectées au port d'entrée dont l'adresse est 00H et activer (alimenter) les périphériques connectés au port de sortie dont l'adresse est 01H comme montré par le schéma de la figure 6.1.

Solution:

Instructions IN 00H
OUT 01H
HLT

Etude de cas : Programme de transfert de données d'un registre à un port de sortie:

1- Énoncé du problème :

Charger le nombre hexadécimal 37H dans le registre B, et afficher le nombre au port de sortie nommé PORT1.

2- Analyse du problème :

Bien que c'est un problème simple, nous allons le diviser en petites étapes.


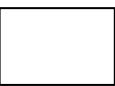




Étape 1 : Charger le registre b avec le nombre

Étape 2: Envoyer le nombre au port de sortie

3- L'organigramme :

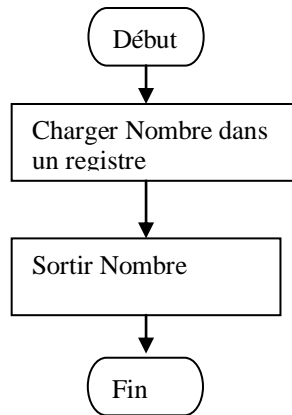
Les étapes décrites au niveau de l'analyse du problème et la séquence (l'ordre) peuvent être représentées par un schéma ou diagramme appelé organigramme.

(REMARQUE : Un organigramme inclut toutes les étapes décrites ; il doit représenter une approche logique des étapes incluses dans le problème. **Comme règle générale, un organigramme n'inclut pas comment effectuer les étapes et quelles sont les registres utilisés.** Ces étapes décrites dans l'organigramme sont traduites en langage assembleur ensuite. Voici les symboles souvent utilisés durant la réalisation d'un organigramme.

	Flèches pour indiquer la direction (le sens) de la progression de l'exécution du programme
	Rectangle pour représenter une procédure ou une opération
	Diamant ou losange pour représenter une prise de décision
	Signe ovale pour représenter le début et la fin d'un programme
	Rectangle avec cotés à deux barres pour représenter une procédure prédéfinie comme un sous programme (subroutine)
	Petits cercles fléchés pour représenter une continuation vers une page différente

)

La figure ci-contre montre la procédure des étapes décrites précédemment dans un organigramme



4- Le programme en langage assembleur :

<u>MVI B, 37H</u>
<u>MVI A, B</u>
<u>OUT 01H</u>
<u>HLT</u>

2^{ème} Étude de cas : Programme de transfert de données pour contrôler des périphériques de sortie.

Énoncé du problème :

Un micro-ordinateur est conçu pour contrôler plusieurs appareils et lampes dans une maison. Le système a un port de sortie dont l'adresse est 01H. et plusieurs unités sont connectées aux bits D0 à D7 comme montré par la figure 6.4 (voir figure). Une matinée froide, vous voulez allumer ou démarrer la radio, la cafetière et le chauffage. Écrire les instructions nécessaires pour le micro-ordinateur. Assumer que la mémoire RAM commence par 2000H.

Analyse du problème :

Le port de sortie montré à la figure 6.4 est un flip flop (bascule) D. Quand les données sont envoyées au port de sortie, elles sont relayées par le flip flop D. un bit de donnée de logique 1 alimente approximativement 5V comme sortie et peut activer un relais.

Pour allumer la radio, la cafetière et le chauffage, il faut fixer D6, D5 et D4 à la logique 1 et les autres bits à la logique 0.

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	1	0	0	0	0

$$01110000 = 70H$$

Le port de sortie requiert 70H, et 70H pourrait être envoyé au port de sortie en chargeant l'accumulateur avec 70H.

Le programme :

Mnémonics	Commentaire
<u>MVI A, 70H</u>	<u>Charger l'accumulateur avec les bits nécessaires pour allumer les périphériques</u>
<u>OUT 01H</u>	<u>Envoyer le même agencement de bits vers le port 01 et démarrer les périphériques</u>
<u>HLT</u>	<u>Fin du programme</u>

II. Les opérations Arithmétiques :

Le microprocesseur 8085 effectue différents types d'opérations arithmétiques, comme l'addition, la soustraction, l'incrémentation, et la décrémentation.

Instructions : Les instructions arithmétiques (sauf INR et DCR) :

- Considère implicitement que l'accumulateur est un des opérants
- Modifient tous les flags selon les conditions des données du résultat dans l'accumulateur.
- Place le résultat dans l'accumulateur
- N'affecte pas (ne change pas) le contenu du registre opérant.

Les instructions INR et DCR

- Affectent (changent) le contenu du registres spécifié
- Affectent tous les flags sauf le carry (CY)

• Opcode	Opérant	Exemples
• <u>ADD</u>	<u>Rg</u>	<u>ADD B</u>
• <u>ADI</u>	<u>8-bit</u>	<u>ADI, 87H</u>
• <u>SUB</u>	<u>Rg</u>	<u>SUB C</u>
• <u>SUI</u>	<u>8-bit</u>	<u>SUI 43H</u>
• <u>INR</u>	<u>Rg</u>	<u>INR C</u>
• <u>DCR</u>	<u>Rg</u>	<u>DRC E</u>

1. L'Addition :

Le 8085 effectue des additions avec des nombres binaires de 8-bit et garde le résultat au niveau de l'accumulateur. Si la somme est plus large que les huit bit (FF), le carry flag est fixé.

Exemple 1:

Le contenu de l'accumulateur est 93H et le contenu du registre C est B7H. Additionneur les 2 contenus

Solution :

Instructions ADD C

			CY	D7	D6	D5	D4	D3	D2	D1	D0
(A)	93H	=		1	0	0	1	1	0	0	1
(B)	B7	=		1	0	1	1	0	1	1	1
Som. (A)	4AH	=	1	0	1	0	0	1	0	1	0

Conditions des flags

S = 0 Z = 0 CY = 1

Donc la somme est 4AH, il y a débordement et la carry est fixé

Étude de cas : Programme pour les opérations arithmétiques Addition et incrémentation:

Énoncé du problème :

Écrire un programme pour effectuer les opérations suivantes et vérifier la sortie (le résultat)

- 1- Charger le nombre 8BH dans le registre D
- 2- Charger le nombre 6FH dans le registre C
- 3- Incrémenter le contenu du registre C par 1.
- 4- Ajouter le contenu des registres C et D et afficher le résultat de la somme au niveau PORT1.

Le programme :

Instructions	
Opcode	Opérant
<u>MVI</u>	<u>D,8B</u>
<u>MVI</u>	<u>C,6F</u>
<u>INR</u>	<u>C</u>
<u>MOV</u>	<u>A,C</u>
<u>ADD</u>	<u>D</u>
<u>OUT</u>	<u>PORT1</u>
<u>PORT1</u>	
<u>HLT</u>	

2. La Soustraction :

Le 8085 effectuent une soustraction en utilisant la méthode du second complément avec des nombres binaires de 8-bit et garde le résultat au niveau de l'accumulateur. Une soustraction peut être effectuée soit en utilisant l'instruction SUB pour soustraire le contenu d'un registre source (A,B, C, D, E, H, L) ou l'instruction SUI pour soustraire un nombre de 8-bit du contenu de l'accumulateur. Dans les deux cas c'est toujours (A-X ; X étant l'autre registre ou le nombre de 8-bit)

Le 8085 effectue intérieurement les étapes suivantes pour exécuter l'instruction SUB ou SUI

- Etape 1 : Convertir le nombre à soustraire en son 1er complément
- Etape 2 : Ajouter ensuite 1 au 1er complément pour obtenir le 2ème complément
- Etape 3 : Ajouter le 2ème complément au contenu de l'accumulateur
- Etape 4 : complémenter le Carry flag

Exemple 1:

Le contenu de l'accumulateur est 97H et le contenu du registre B est 65H. Soustraire le contenu du registre B du contenu de l'accumulateur

Solution :

Instructions SUB B

				CY	D7	D6	D5	D4	D3	D2	D1	D0
	(B)	65H	=	<div></div>	0	1	1	0	0	1	0	1
Etape 1	1 ^{er} complément de	65H	=		1	1	0	0	1	0	1	0
Etape 2	2 ^{ème} complément de	65H	+		0	0	0	0	0	0	0	1
					1	0	0	1	1	0	1	1
	Pour soustraire 97H – 65H ajouter	65H	=		1	0	0	1	0	1	1	1
	97H au 2 ^{ème} complément											
Etape 4			=	<div>1</div>	0	0	1	1	0	0	1	0
Etape 4	Complémenter le carry	CY		<div>0</div>	0	0	1	1	0	0	1	0

Résultat : (A) 32H

Conditions des flags

S = 0 Z = 0 CY = 0

Si la réponse est négative, ceci va être montré par le second complément de la magnitude (valeur absolue) du résultat actuel :

Exemple : Si la soustraction ci-dessus était effectuée comme 65H-97H, la réponse serait le 2^{ème} complément de 32 avec le carry (Borrow) fixé.

NB : LE CARRY AU NIVEAU DE LA SOUSTRACTION EST TRES IMPORTANTE. SI LE CARRY FLAG EST FIXE (CY=1). CECI VEUT DIRE QUE LE RESULTAT DE LA SOUSTRACTION EST NEGATIF. LE FAIT QUE LE CARRY SOIT FIXE A 1 INDIQUE AINSI QUE LE RESULTAT AU NIVEAU DE L'ACCUMULATEUR EST SOUS FORME DE 2^{EME} COMPLEMENT. DONC POUR RENDRE LE RESULTAT IL FAUT Y AJOUTER UN SIGNE MOINS (-) POUR OBTENIR LE VRAI RESULTAT.

Etude de cas Programme : Soustraction de deux nombres.

Enoncé du problème :

Ecrire un programme pour effectuer les opérations suivantes :

- Charger le nombre 30H dans le registre B
- Soustraire 39H de 30H
- afficher le résultat de la somme au niveau PORT1.

Programme :

Le programme pour la soustraction de 2 nombres non signés est représenté au niveau de la figure suivante 6.6 :

Instructions	
Opcode	Opérant
<u>MVI</u>	<u>B,30H</u>
<u>MVI</u>	<u>C,39H</u>
<u>MOV</u>	<u>A,B</u>
<u>SUB</u>	<u>C</u>
<u>OUT</u>	<u>PORT1</u>
<u>PORT1</u>	
<u>HLT</u>	

Pour effectuer l'instruction SUB le microprocesseur va effectuer intérieurement les opérations suivantes :

				CY	D7	D6	D5	D4	D3	D2	D1	D0
	(B)	39H	=		0	0	1	1	1	0	0	1
Etape 1	1 ^{er} complément de	39H	=		1	1	0	0	0	1	1	0
Etape 2	2 ^{ème} complément de	39H	+		0	0	0	0	0	0	0	1
					1	1	0	0	0	1	1	1
	Pour soustraire 30H – 39H ajouter 30H au 2 ^{ème} complément de 39H	30H	=		0	0	1	1	0	0	0	0
Etape 3			=	0	1	1	1	1	0	1	1	0
Etape 4	Complémenter le carry	CY		1	1	1	1	1	0	1	1	1

Conditions des flags

S = 1 Z = 0 CY = 1

Comme le carry est fixé à 1 ; le nombre F7H est le 2nd complément de (39H-30H)=09H

III. Les opérations Logiques :

Le microprocesseur est généralement une puce programmable, il peut effectuer toutes les opérations logiques à travers son ensemble d'instruction. Les instructions du 8085 incluent les instructions logiques comme AND, OR, EXOR et NOT (COMPLEMENT).

Instructions : Les instructions Logiques:

- Considèrent implicitement que l'accumulateur est un des opérants
- Réinitialisent le carry flag. L'instruction COMPLEMENT est une exception. Elle n'affecte aucun flag.
- Modifient les flags Z, P et S selon les conditions du résultat
- Place le résultat au niveau de l'accumulateur
- N'affecte pas (ne change pas) le contenu du registre opérant.

- Opcode	Opérant	Exemples
- ANA	Rg	ANA C
- ANI	8-bit	ANI, 24H
- ORA	Rg	ORA E
- ORI	8-bit	ORI 43H
- XRA	Rg	XRA B
- XRI	8-bit	XRI 4AH
CMA		CMA

1- Logique AND:

La procédure pour effectuer les opérations logiques à travers l'instruction est un peu différente de la porte logique physique. La porte AND contient 2 entrées et une sortie. De l'autre côté, l'instruction ANA engage 8 portes; 16 entrées, 8 sorties.

Exemple : Assumons que le registre B contient 77H et l'accumulateur contient 81H. Quel est Le résultat de l'opération And entre les registres A et B.

Solution :

- Instructions **ANA B**
- (A) 81H= 10000001
- (B) 77H= 01110111
- A And B = 00000001
= 01H
- Conditions des flags:
- S = 0 Z = 0 CY = 0

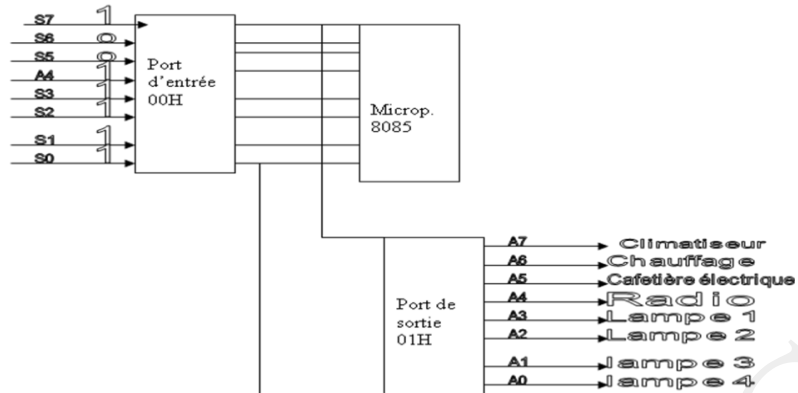
Étude de cas : Masquage des données avec la logique AND

Énoncé du problème :

Pour économiser de l'énergie et éviter un surchargement électrique pendant un après midi chaud. Mettre en place les procédures suivantes pour contrôler les appareils électriques à travers toute la maison (Voir figure 6.8 ref 84 à partir de photocopie). Assumer que le control des contacts est placé au niveau de la cuisine et est accessible à tout un chacun dans la maison. Écrire les instructions nécessaires pour :

- 1 – Allumer ou démarrer le climatiseur (air conditionner), si le contact S7 (switch) d'un port d'entrée 00H est en position on (activé)
- 2 – Ignorer tous les autres contacts (switches) du port d'entrée même si quelqu'un tente d'allumer ou de démarrer d'autres appareils électriques.

Figure 6.8



Analyse du programme :

Dans ce problème la position d'un seul contact nous intéresse : S7 ; qui est connecté à ligne D7 du bus de données. Assumer que différentes personnes ont activé ou allumer ou mis en position ON les contacts du climatiseur (air conditionner) (S7), la radio (S4), et les lampes (S3, S2, S1, S0). Si le microprocesseur lit le port d'entrée (IN 00H), l'accumulateur va recevoir la donnée qui est l'octet 9FH. Cette instruction peut être aussi simulée par l'instruction MVI A, 9FH. Mais si vous êtes intéressés seulement de savoir si le contact S7 est en position ON ; **vous pouvez masquer les bits D6 à D0 en appliquant la logique AND** aux données d'entrées avec un octet qui a 0 aux positions D6 à D0 et 1 au niveau de la position du bit D7.

D7	D6	D5	D4	D3	D2	D1	D0	
1	0	0	0	0	0	0	0	= 80H

Après que les bits D0 jusqu'à D6 soient masqués, l'octet qui reste peut être envoyé vers le port de sortie pour démarrer le climatiseur (Air Conditionner)

Résultat (Sortie) du programme :

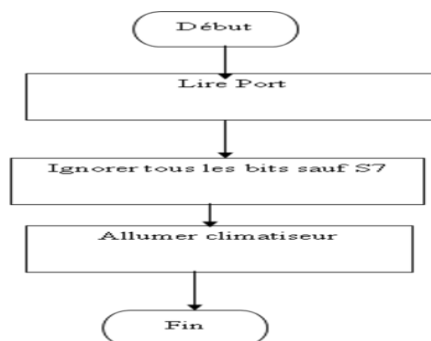
L'instruction ANI 80H applique la logique AND entre la donnée et l'accumulateur.

	(A) =	1	0	0	1	1	1	1	1	(9FH)
AND (Octet pour masquer)		1	0	0	0	0	0	0	0	(80H)
	(A) =	1	0	0	0	0	0	0	0	

Conditions des flags

S = 1

Z = 0 CY = 0



Programme :

Instructions		Commentaires
Opcode	Opérant	
<u>MVI</u>	<u>A, Donnée</u>	<u>Cette instruction simule l'instruction IN 00H</u>
<u>ANI</u>	<u>80H</u>	<u>Masquer tous les bits excepté D7</u>
<u>OUT</u>	<u>01H</u>	<u>Démarrer le climatiseur si S7 est en position ON</u>
<u>HLT</u>		<u>Fin du programme</u>

Dans cet exemple la sortie ou résultat est le même que l'octet de donnée (80H) pour masquer ; parce que le contact S7 (ou le bit D7) est en position ON. Si le contact S7 est en position OFF, la sortie serait zéro.

L'OPERATION « MASQUER » EST SOUVENT UTILISEE POUR ELIMINER LES BITS NON VOULUS DANS UN OCTET. L 'octet pour éliminer (masquer) (qui doit être en opération AND) est déterminé en plaçant des 0s dans les positions des bits qui doivent être masqués et des 1s dans les autres positions des bits qui restent.

2- OR, EXCLUSIVE-OR ET NOT:

Les instructions ORA et ORI représentent la porte logique OR avec 8 portes OR de 2 entrées. Ce processus est similaire à la logique AND expliquée dans la section précédente. L'instruction (XRA et XRI) effectue la fonction logique de la porte Exclusive OR pour huit bits. Et l'instruction CMA applique la fonction logique NOT aux bits de l'accumulateur.

Exemple 1 :

Assumons que le registre B contient 93H et l'accumulateur contient 15H. Donner le résultat des instructions ORA B, XRA B, ET CMA.

1- L'instruction ORA B va effectuer l'opération suivante

	(B) =	1	0	0	1	0	0	1	1	(93H)
OR	(A) =	0	0	0	1	0	1	0	1	(15H)
	(A) =	1	0	0	1	0	1	1	1	(97H)

Conditions des flags

S = 1 Z = 0 CY = 0

Le résultat 97H sera placé au niveau de l'accumulateur et le flag sera réinitialisé et les autres flags seront modifiés pour refléter les conditions des données dans l'accumulateur.

2- L'instruction XRA B va effectuer l'opération suivante

	(B) =	1	0	0	1	0	0	1	1	(93H)
X-OR	(A) =	0	0	0	1	0	1	0	1	(15H)
	(A) =	1	0	0	1	0	1	1	0	(86H)

Conditions des flags

S = 1 Z = 0 CY = 0

Le résultat 86H sera placé au niveau de l'accumulateur et les flags seront modifiés comme montrés.

3- L'instruction CMA aura le résultat suivant

	(A) =	0	0	0	1	0	1	0	1	(15H)
CMA	(A) =	1	1	1	0	1	0	1	0	(EAH)

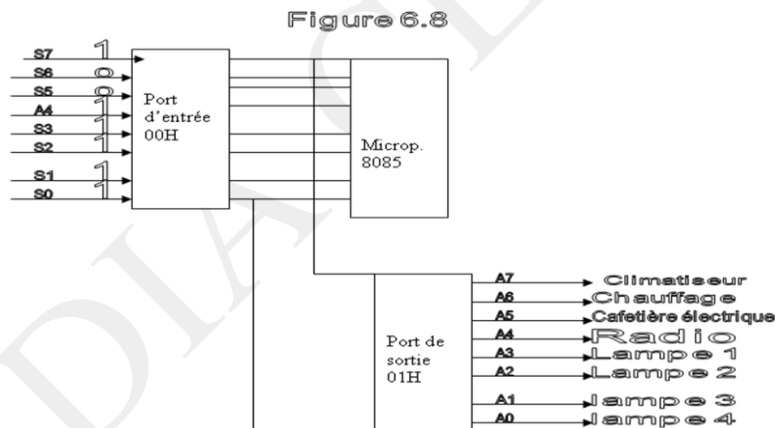
Le résultat 86H sera placé au niveau de l'accumulateur et aucun flag ne sera modifié.

3- Activer ou réinitialiser des bits spécifiques:

Parfois à des moments différents on aimerait activer ou réinitialiser un bit spécifique sans affecter les autres bits. **LA PORTE LOGIQUE OR PEUT ETRE UTILISEE POUR ACTIVER LE BIT (LE METTRE AU NIVEAU LOGIQUE 1). LA PORTE LOGIQUE AND PEUT ETRE UTILISEE POUR REINITIALISEE LE BIT (LE METTRE AU NIVEAU LOGIQUE 0).**

Exemple 1:

Au niveau de la figure 6.8 ref 84 maintenir la radio allumée sans affecter le fonctionnement des autres appareils ; même si quelqu'un éteint (met en position OFF) le contact S4.



Solution :

Pour maintenir la radio allumée sans affecter les autres appareils, le bit D7 doit être activé en appliquant la logique OR, entre le résultat de la lecture du port d'entrée et l'octet de donnée 10H comme suivant :
L'instruction IN va lire la position des contacts, comme montré avec D7 à D0, et l'instruction ORI va activer le bit D4 sans affecter les autres bits.

IN 00H	(A) =	D7	D6	D5	D4	D3	D2	D1	D0
ORI 10H	=	0	0	0	1	0	0	0	0
	(A) =	D7	D6	D5	1	D3	D2	D1	D0

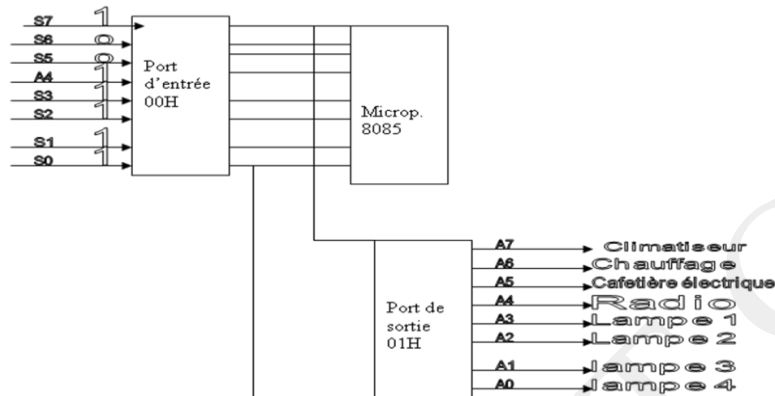
Conditions des flags

CY = 0 ; les autres vont dépendre des données.

Exemple 2:

Au niveau de la figure 6.8 ref 84 Considérer que c'est l'hivers et éteignez le climatiseur (Air conditionner) sans affecter les autres appareils.

Figure 6.8



Solution :

Pour éteindre le climatiseur, il faut réinitialiser (mettre au niveau logique 0) le bit D7 en appliquant la logique AND entre le résultat de la lecture du port d'entrée et l'octet de donnée et l'octet de donnée 7FH comme suivant :

L'instruction ANI réinitialise le bit D7 sans affecter les autres bits

IN 00H	(A) =	D7	D6	D5	D4	D3	D2	D1	D0
ANI 7FH	=	0	1	1	1	1	1	1	1
	(A) =	0	D6	D5	D4	D3	D2	D1	D0

Conditions des flags

CY = 0 ; les autres vont dépendre des données.

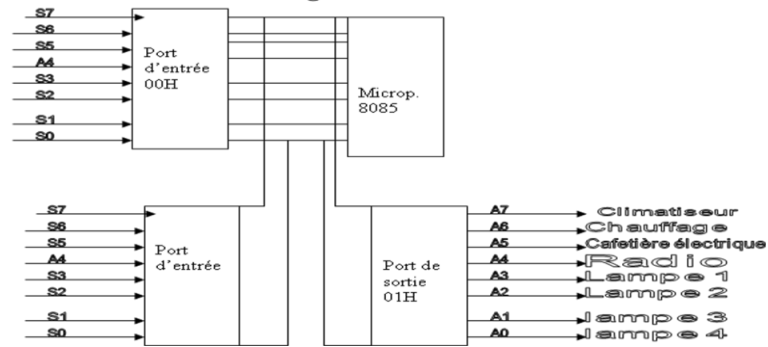
4- Etude de cas de programme: Appliquer l'instruction OR à des données provenant de deux ports d'entrées

Énoncé du problème :

Un deuxième port d'entrée avec huit contacts et ayant l'adresse 01H (voir fig. 6.9 ref 186) est connecté au microprocesseur montré à l'exemple précédent (fig. 6.8) pour aussi bien contrôler les mêmes appareils et lampes à partir de la chambre à coucher que à partir de la cuisine.

Écrire les instructions nécessaires pour allumer les appareils à partir de n'importe lequel des ports d'entrées.

Figure 6.9



Analyse du programme :

Pour allumer les appareils à partir de n'importe lequel des ports ; le microprocesseur doit lire les contacts des deux ports et appliquer l'instruction OR entre les différentes positions des contacts des deux ports. Considérons que les positions des contacts dans un port d'entrée (situé au niveau de la chambre à coucher) correspond à l'octet de donnée 91H et les positions des contacts dans le 2nd port d'entrée (situé dans la cuisine) correspond à l'octet de donnée A8H. La personne qui se trouve dans la chambre à coucher veut allumer ou démarrer le climatiseur, la radio, et la lampe de la chambre à coucher ; et la personne qui se trouve dans la cuisine veut allumer ou démarrer le climatiseur, la cafetière, et la lampe de la cuisine.

En appliquant l'instruction OR entre ces deux octets ; le microprocesseur peut allumer ou démarrer les appareils nécessaires.

Programme :

Adresse de la mémoire	Code machine	Instructions		Commentaires
		Opcode	Opérant	
XX00	06	MVI	B, 91H	Cette instruction simule la lecture du port d'entrée 01H
XX01	91			
XX02	0E	MVI	C, A8H	Cette instruction simule la lecture du port d'entrée 00H
XX03	A8			
XX04	78	MOV	A, B	transférer le contenu de B dans A pour appliquer la logique OR (c'est pas possible directement entre B, et C).
XX05	B1	ORA	C	Combiner les positions des contacts des registres B et C au niveau de l'écran.
XX06	D3	OUT	PORT1	Allumer les appareils et les lampes
XX07	PORT1			
XX08	76H	HLT		Fin du programme

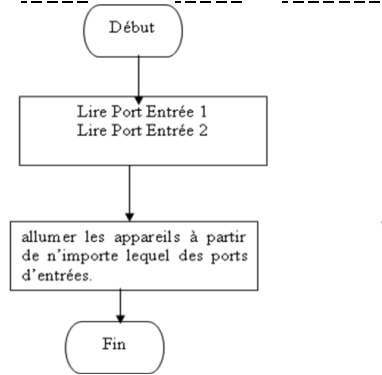
Résultat (Sortie) du programme :

En appliquant la logique OR entre les octets de données dans B et dans C :

(B)	→	(A) =	1	0	0	1	0	0	0	1	(91H)
OR											
(C) =		(A) =	1	0	1	0	1	0	0	1	(A8H)
		(A) =	1	0	1	1	1	0	0	1	(9BH)

Conditions des flags

S = 1 Z = 0 CY = 0



Programme :

Instructions		Commentaires
Opcode	Opérant	
MVI	B, 91H	Cette instruction simule la lecture du port d'entrée 01H
MVI	C, A8H	Cette instruction simule la lecture du port d'entrée 00H
MOV	A,B	transférer le contenu de B dans A pour appliquer la logique OR (c'est pas possible directement entre B, et C)
ORA	C	Combiner les positions des contacts des registres B et C au niveau de l'écran.
OUT	PORT1	Allumer les appareils et les lampes
HLT		Fin du programme

IV. Les opérations de Branchement:

Les instructions de branchement sont les instructions les plus puissantes car elles permettent au microprocesseur de changer la séquences d'un programme, soit de manière inconditionnelle ou sous certaines conditions de tests effectués.

Les instructions de branchement ordonnent au microprocesseur d'aller à un emplacement de mémoire différent et le microprocesseur va continuer à exécuter les codes machines à partir de ce nouvel emplacement de mémoire. les instructions de branchement sont classées en 3 catégories :

- Les instructions JUMP
- Les instructions CALL et RETURN.
- Les instructions RESTART

Au niveau de cette section nous verrons simplement les applications de l'instruction JUMP.

Les instructions JUMP peuvent être classées en 2 catégories : Les JUMP conditionnels et les JUMPS inconditionnels.

1- JUMP inconditionnel

L'ensemble des instructions du 8085 inclus une seule instruction JUMP inconditionnelle. Le JUMP inconditionnel permet au programmeur de réaliser des boucles continues.

Instruction

Opcodes	Opérant	Description
<u>JMP</u>	<u>16-bit</u>	<u>JUMP</u> - Ceci est une instruction de 3-octet - Le 2 nd octet et le 3 ^{ème} octet représentent l'adresse de 16-bit.

Exemple :

Pour de mander au microprocesseur de se déplacer vers l'emplacement de mémoire 2000H ; on écrit :

Code machine	Mnémonics
<u>C3</u>	<u>JMP 2000H</u>
<u>00</u>	
<u>20</u>	

2- Étude de cas de programme: JUMP inconditionnel pour réaliser une boucle continue

Énoncé du problème :

Pour modifier le programme de l'exemple 2 (dans les instructions de transfert) pour lire (enregistrer) les positions des contacts continuellement et d'allumer ou de démarrer les appareils selon le cas.

Programme :

Adresse de la mémoire	Code machine	label	Instructions		Commentaires
			Opcode	Opérant	
2000	DB	START	IN	00H	Lire les contacts d'entrée
2001	00				
2002	D3		OUT	01H	Démarrer les appareils selon les positions des contacts
2003	01				
2004	C3		JMP	START	Retourner et recommencer à lire les positions des contacts encore
2005	00				
2006	20				

3- Les JUMPS Conditionnels :

Les instructions JUMPS conditionnels permettent au microprocesseur de prendre des décisions sur la base de certaines conditions indiquées par les flags. Les instructions JUMPS conditionnels vérifient les conditions du flag et prennent une décision de changer ou de ne pas changer la séquence d'un programme.

Instructions

Toutes les instructions JUMP dans le 8085 sont des instructions de 3-octets

Les instructions suivantes transfèrent la séquence du programme à l'emplacement de mémoire précisé sous les conditions requises (données.)

Opcodes	Opérant	Description
JC	16-bit	Jump on carry (si le résultat génère un carry et CY=1)
JNC	16-bit	Jump on no carry (CY = 0)
JZ	16-bit	Jump on zéro (si le résultat est zéro (nul) et Z=1)
JNZ	16-bit	Jump on no zéro (Z = 0)
JP	16-bit	Jump on Plus (si D7 = 0 et S = 0)
JM	16-bit	Jump on Minus (si D7 = 1 et S = 1)
JPE	16-bit	Jump on Even (P = 1)
JPO	16-bit	Jump on Odd (P = 0)

Eude de cas de programme: Tester à partir du carry flag

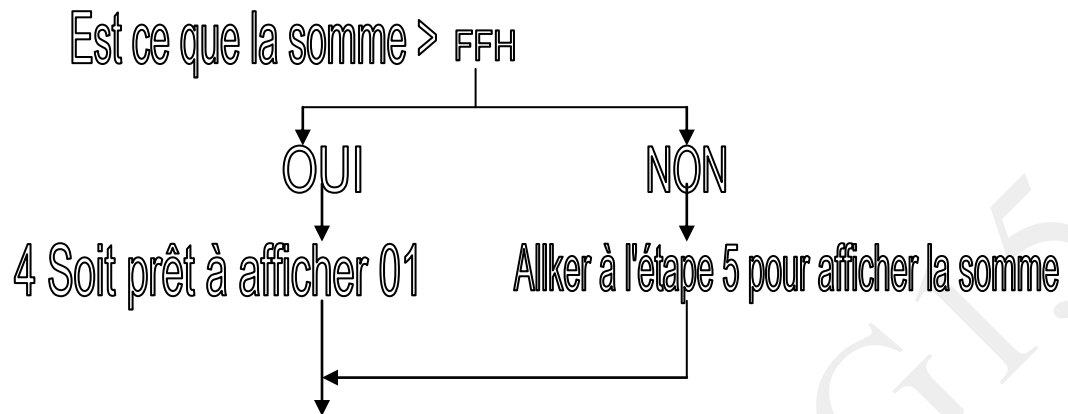
Énoncé du problème :

Charger les nombres hexadécimaux 9BH et A7H dans les registres D et E respectivement et additionner les nombres. Si le nombre est supérieure à FFH, afficher 01H au port de sortie PORT0, sinon afficher la somme.

Analyse du programme et organigramme :

Le problème peut être subdivisé en étapes montrées ci-dessous.

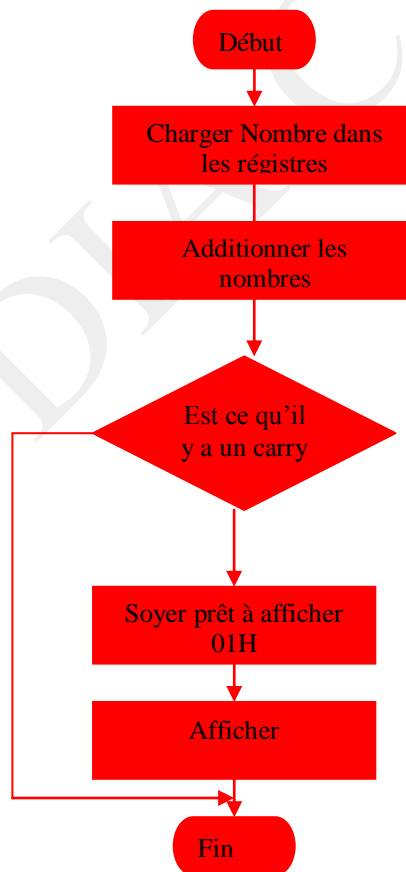
- 1- charger les nombres dans les régistres
- 2- Additionner les nombres
- 3- Vérifier la somme
- 4- être prêt à afficher 01



- 5- Affichage
- 6- Fin

Organigramme

Les 6 étapes peuvent être transformées en organigramme comme montré par la figure suivante :



programme

MVI D, 9BH

MVI E, A7H

MOV A, D

ADD E

JNC AFFICHER

MVI A, 01H

Afficher OUT 01H

HLT

Les codes machines avec les adresses des mémoires :

Considérons que notre mémoire RAM commence par 2000H. Le programme de langage assembleur pourrait être traduit de la manière suivante.

Adresse de la mémoire	Code machine	label	Instructions	
			Opcode	Opérant
<u>2000</u>	<u>16</u>	<u>START</u>	<u>MVI</u>	<u>D, 9BH</u>
<u>2001</u>	<u>9B</u>			
<u>2002</u>	<u>1E</u>		<u>MVI</u>	<u>E, A7H</u>
<u>2003</u>	<u>A7</u>			
<u>2004</u>	<u>7A</u>		<u>MOV</u>	<u>A, D</u>
<u>2005</u>	<u>83</u>		<u>ADD E</u>	
<u>2006</u>	<u>D2</u>		<u>JNC</u>	<u>DSPLAY</u>
<u>2007</u>	<u>X</u>			
<u>2008</u>	<u>X</u>			
<u>2009</u>	<u>3E</u>		<u>MVI</u>	<u>A, 01H</u>
<u>200A</u>	<u>01</u>			
<u>200B</u>	<u>D3</u>	<u>DSPLAY</u>	<u>OUT</u>	<u>00H</u>
<u>200C</u>	<u>00</u>			
<u>200D</u>	<u>76</u>		<u>HLT</u>	

Techniques de programmation avec des instructions additionnelles:

Introduction:

L'ordinateur dans son meilleur aspect dépasse les capacités de l'homme, lorsqu'on a arrive à la répétition des taches simples comme ajouter des millions de nombres.

Dans cette ordre que nous allons introduire dans cette chapitre les boucles, les compteurs et les indexes.

Les instructions nécessaires pour transférer des données de la mémoire au microprocesseur seront introduites dans ce chapitre.

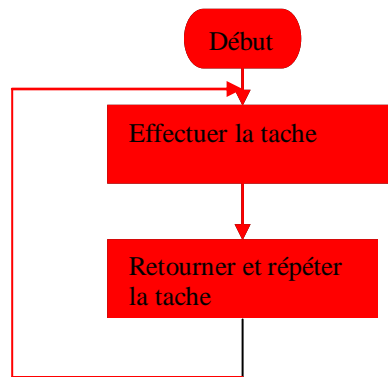
I- Les Techniques de programmation : les boucles – les compteurs et les indexes :

La technique de programmation utilisée pour ordonner au microprocesseur de répéter une tache est appelée **BOUCLE**. Cette procédure est réalisée en utilisant les instructions JUMP. En plus des techniques comme les compteurs et les indexes sont utilisés dans la mise en place des boucles. Les boucles peuvent être divisées en deux groupes.

- Les **boucles continues** qui répètent la tache de manière permanente
- Les **boucles conditionnelles** qui répètent la tache jusqu'à ce qu'une certaine condition soit remplie.

1- Les boucles continues :

Une boucle continue est mise en place en utilisant l'instruction inconditionnelle JUMP comme montré au niveau de l'organigramme ci-dessous

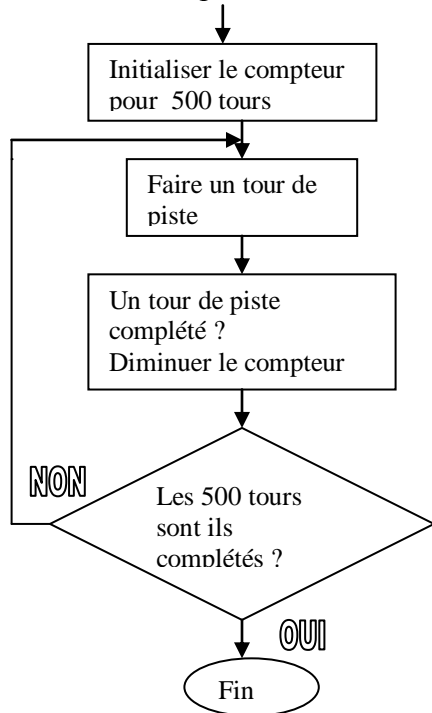


2- Les boucles conditionnelles :

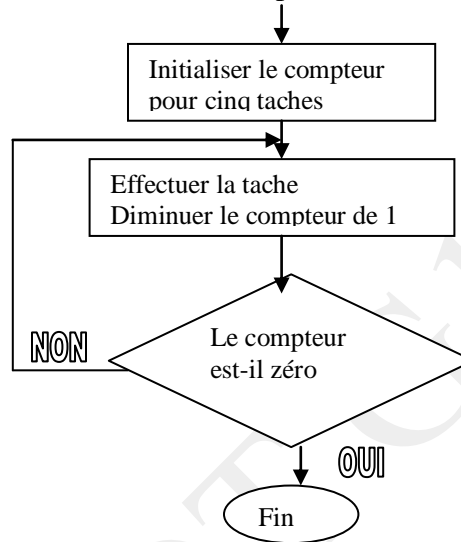
Un compteur est une application typique d'une boucle conditionnelle.

Exemple : comment le microprocesseur répète une tâche 5 fois ? Cette procédure est similaire à une voiture de course de formule 1 qui en est en son dernier tour (500) et qui devrait faire le tour de la piste 500 fois. Comment le conducteur arrive à connaître quand est ce que les cinq cents tours seront complétés ? C'est l'équipe de l'encadrement du conducteur qui met en place une méthode de décompte (compteur) et de signalement (drapeau) pour le conducteur. Ceci peut être représenté schématiquement à la figure 7.2 a. Une approche similaire est nécessaire pour le microprocesseur pour répéter la tâche 5 fois. Le microprocesseur a besoin d'un compteur et d'un flag (drapeau). Ceci peut être réalisé avec la boucle conditionnelle Comme illustré au niveau de l'organigramme 7.2 b.

7.2 a Tours de piste d'une voiture de course



7.2 b Répétition de tâches



3- Les boucles conditionnelles, les compteurs et les indexes :

Un autre type de boucle inclut les indexes (indexer) ou pointeurs en plus des compteurs (Indexer veut dire pointer ou faire référence à des objets par l'intermédiaire de nombres séquentiels (qui se suivent) comme dans une bibliothèque avec les livres sont ordonnés selon des codes (chiffres). Ceci est appelé **Indexe ou Pointeur**. De manière similaire, les octets de données sont stockés dans des emplacements de mémoire, et ces données sont référencées par emplacement de mémoire.

Exemple :

Écrire les étapes nécessaires pour additionner 10 octets de données gardés dans des emplacements de mémoires commençant par un emplacement donné et afficher la somme. Dessiner un organigramme.

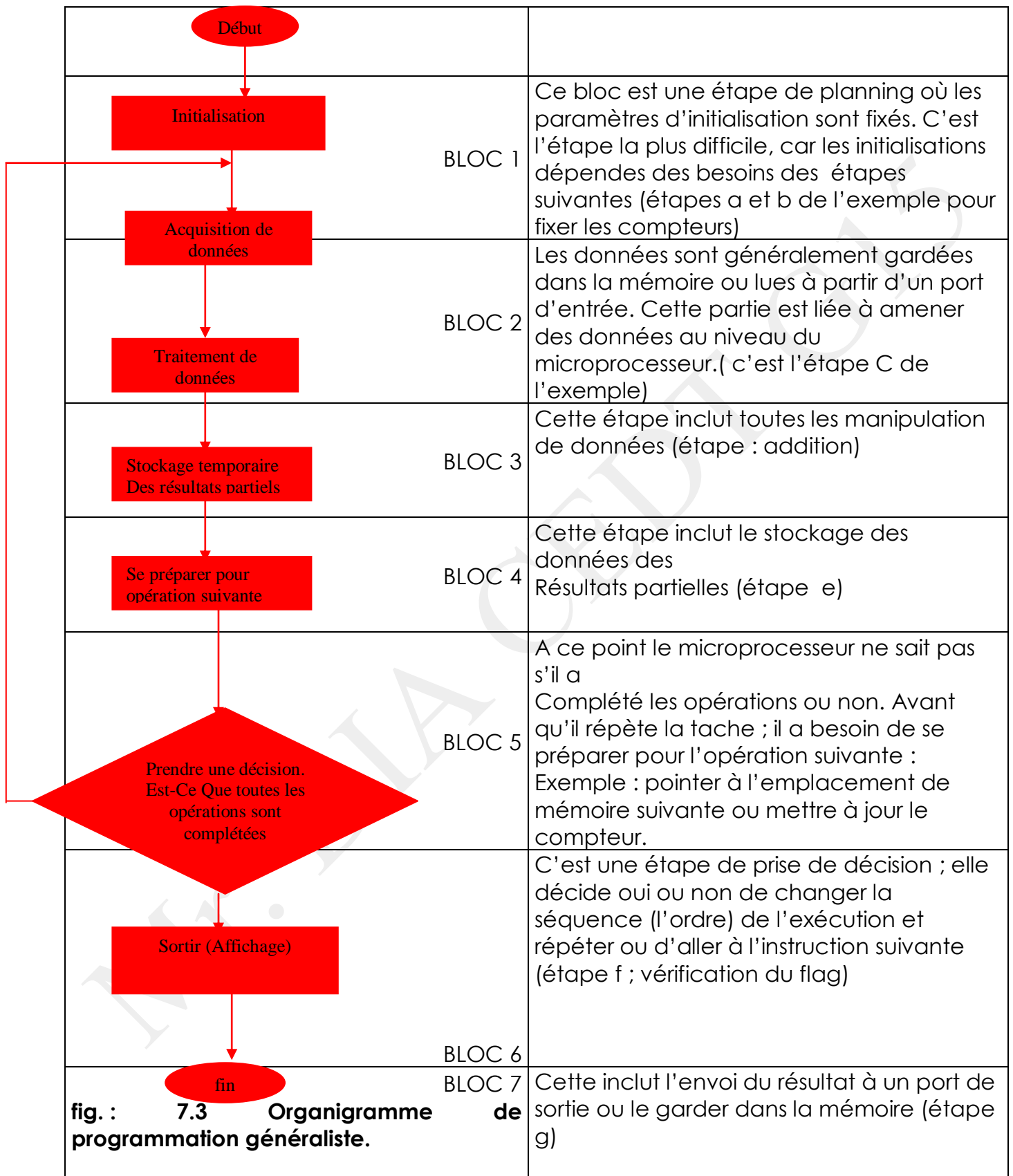
Procédure :

Le microprocesseur a besoin de :

- D'un compteur pour compter 10 octets de données.
- Un indexe ou un pointeur de mémoire pour localiser où les octets de données sont gardés

- c. Transférer la donnée d'un emplacement de mémoire vers le microprocesseur (UAL)
- d. Effectuer l'addition
- e. De registres pour le stockage temporaire des réponses partielles.
- f. Un flag pour indiquer la fin de la tâche
- g. De garder ou de faire sortir le résultat

Ces différentes étapes peuvent être représentées sous la forme d'un organigramme comme montré par la figure 7.3 ci dessous. Cet organigramme généraliste peut être utilisé pour résoudre plusieurs problèmes..



II- Les instructions additionnelles de transfert de données de 16-bits

Pour cette section nous allons introduire les instructions en relation avec le transfert de données avec le microprocesseur et la mémoire ; et les instructions pour des données de 16-bit. Leurs opcodes sont les suivants :

1. Charger les données de 16-bit dans les registres pairs (LXI)

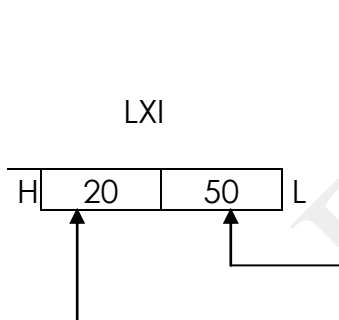
Les instructions LXI effectue une fonction similaire aux instructions MVI ; sauf que les instructions LXI charge des données de 16-bit dans les registres pairs (BC, DE, HL, SP) et dans le registre du Stack Pointer. Ces instructions n'affectent pas les flags.

Exemple :

Écrire les instructions nécessaires pour charger le nombre de 16-bit 2050H dans le registre pair HL en utilisant les opcodes LXI et MVI et expliquer la différence entre les deux instructions.

Instructions

La figure ci-dessus montre le contenu des registres et les instructions nécessaires pour l'exemple.



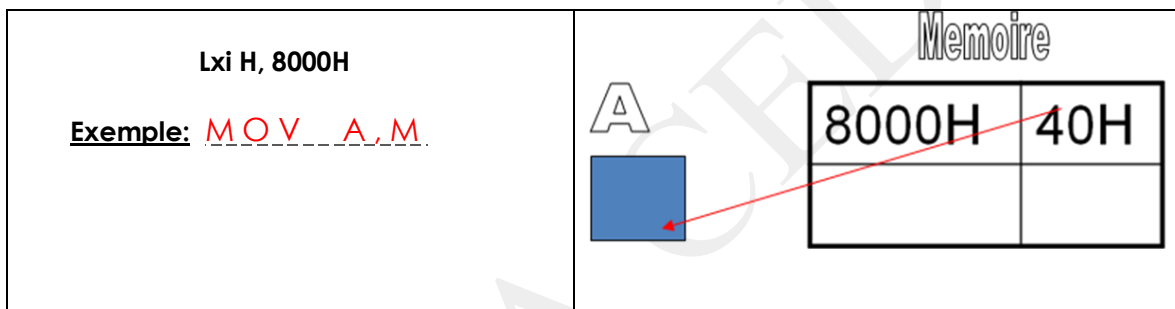
Codes Machines	Mnemonics	Commentaires
<u>21</u>	<u>LXI H, 2050H</u>	<u>Charger les registres HL</u>
<u>50</u>		<u>50 dans le registre L</u>
<u>20</u>		<u>et 20H dans le registre H</u>

Codes Machines	Mnemonics	Commentaires
26	MVI H, 20H	Charger 20H dans le registre H
20		
2E	MVI L, 50H	Charger 50H dans le registre L
50		

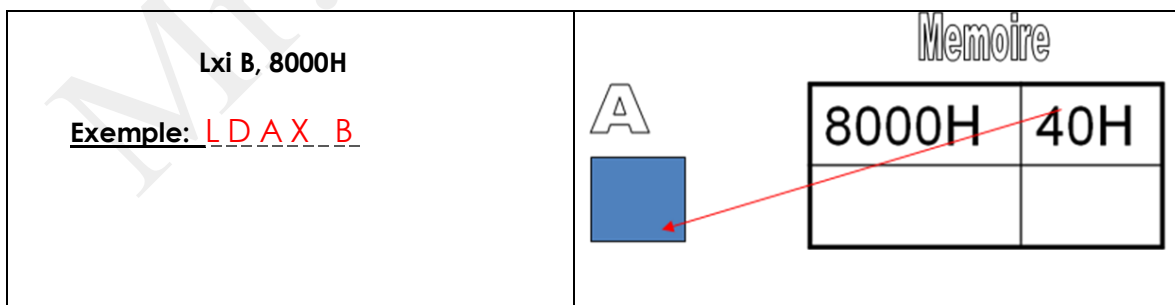
1- Transfert (Copie) de données à partir de la mémoire vers le microprocesseur

Le jeu d'instruction du 8085 inclut 3 instructions de transfert de données vers la mémoire. Ces instructions n'affectent pas les flags.

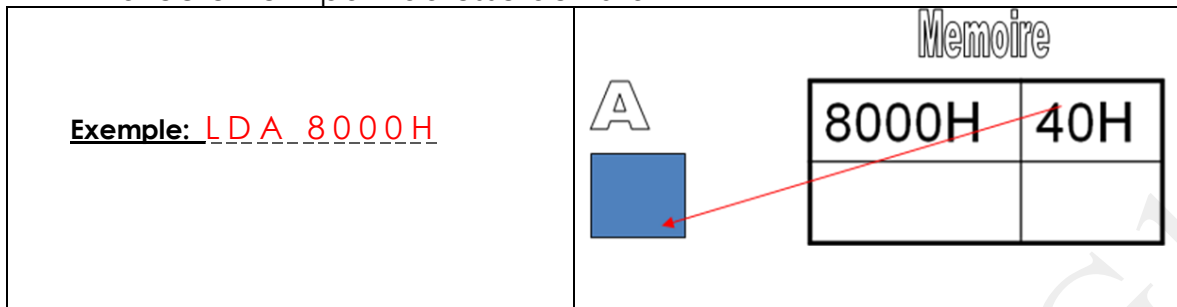
- MOV R, M : (Déplacer à partir de la mémoire vers le registre)
 - L'adresse de l'emplacement de mémoire est spécifié par le contenu du registre pair HL.



- LDAX B et LDAX D : Load Accumulateur Indirect
 - Elle copie l'octet de donnée à partir de l'emplacement de mémoire dans l'accumulateur
 - L'adresse de l'emplacement de mémoire est spécifié par le contenu du registre pair BC ou DE.



- LDA 16-bit Load Accumulateur Indirect
 - Elle copie l'octet de donnée à partir de l'emplacement de mémoire précisé directement par l'adresse de 16-bit



Exemple :

L'emplacement de mémoire 2050H contient l'octet de données F7H. Écrire les instructions pour transférer l'octet de donnée dans l'accumulateur en utilisant 3 opcodes différents MOV, LDAX, et LDA.

2050H	F7
-------	----

Solution :

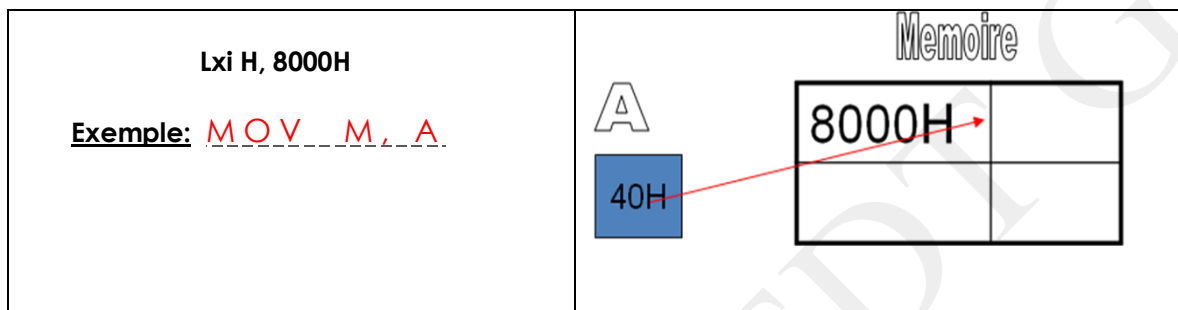
- 1- LXI H, 2050H
- MOV A,M
- 2- LXI B, 2050H
- LDAX B
- 3- LDA 2050H

♦

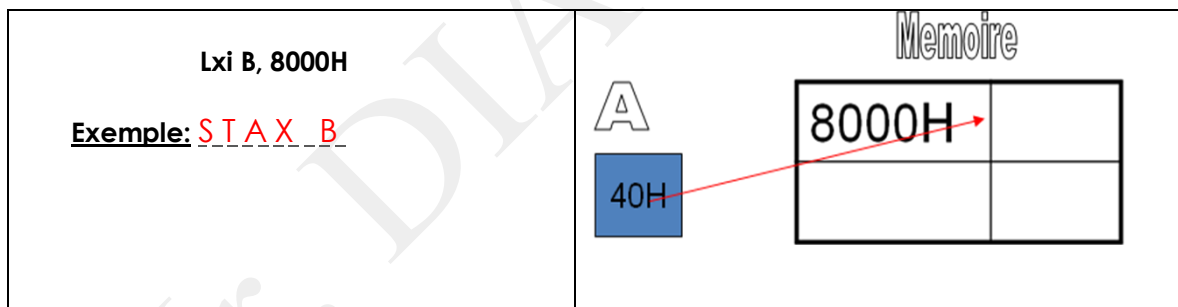
2- Transfert (Copie) de données à partir du microprocesseur vers la mémoire ou directement vers la mémoire

Les instructions pour copier des données à partir du microprocesseur vers la mémoire sont similaires à la section précédente.

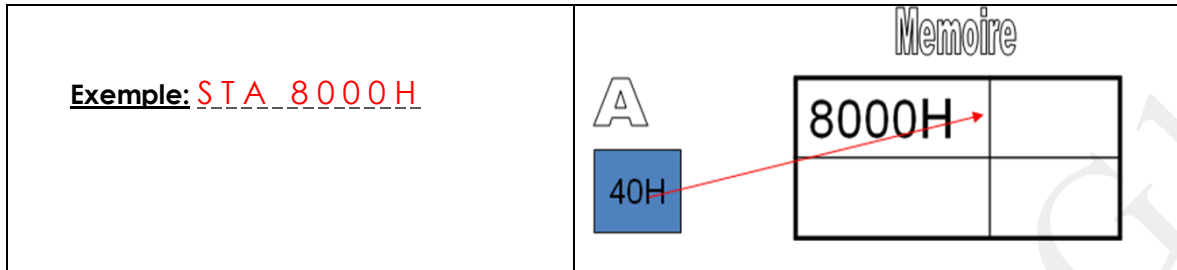
- MOV M,R : (Déplacer à partir du registre vers la mémoire)
 - l'adresse est précisée par le contenu du registre pair HL



- STAX B et STAX D : Store Accumulateur Indirect
 - Elle copie les données à partir de l'accumulateur dans l'emplacement de mémoire dont l'adresse est précisée par le contenu du registre pair BC ou DE



- STA 16-bit Store Accumateur Indirect
 - Elle copie les données à partir de l'accumulateur dans l'emplacement de mémoire dont l'adresse est précisée par l'opérant de 16-bit.



Exemple :

- 1- Le registre B contient 32H. Utiliser les instructions MOV, STAX pour copier le contenu du registre B dans l'emplacement de mémoire 8000H en utilisant le mode d'adressage indirecte.

B
32H

- 2- L'accumulateur contient F2H . Copier (A) au niveau de l'emplacement de mémoire 8000H en utilisant le mode d'adressage direct.

A
F2H

- 3- Charger F2H directement au niveau de L'emplacement de mémoire 8000H en utilisant le mode d'adressage indirect.

Solution :

♦

LXI H,
2050H
MOV M,B

LXI D, 8000H
MOV A,B
STAX D

♦

STA 8000H

♦

LXI H, 8000H

`MVI M, F2H`

3- Les opérations arithmétiques de 16-bit ou avec les registres pairs.

Ce sont les instructions incrémenter / décrémente de 16-bit.

Ces instructions qui sont décrits ci-dessous n'affectent pas les flags.

- INX Rp – Incrémenter le Registre Pair

Les instructions concernées sont : INX B – INX D - INX H – INX SP

- DCX Rp – Décrémenter le Registre Pair

Les instructions concernées sont : DCX B – DCX D - DCX H – DCX SP

Exemple :

Ecrire les instructions pour charger le nombre 2050H dans le registre pair BC.

Incrémenter le nombre en utilisant l'instruction INX B et vérifier si l'instruction INX B équivalent aux instructions INR B et INR C.

Solution :



B 20 50 C

`LXI B, 2050H`

B 20 51 C

`INX B`



B 20 50 C

`INR B`

B 21 51

`INR C`

Étude de cas : Programme de transfert de données d'un registre à un port de sortie:

Énoncé du problème :

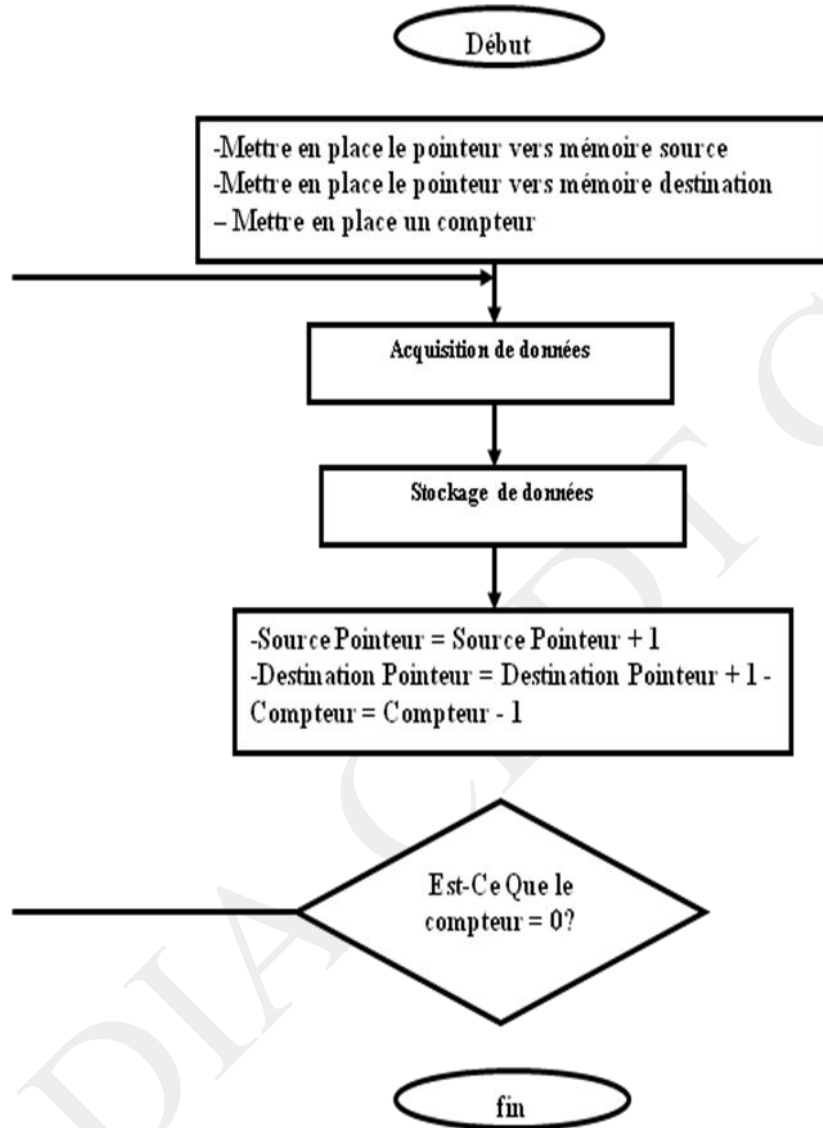
16 octets de données sont placées dans les emplacements de mémoire entre XX50H et XX57H. Transférer le bloc entier de données aux nouveaux emplacements de mémoires commençant par XX70H.

Données (H)

37- 12- F2- 82- 57- 5A- 7F- DA- E5- 8B- A7- C2- B8- 10- 19- 98

Analyse du problème :

Le problème peut être analysé en termes des blocs montrés dans l'organigramme de la figure suivante.



Le Bloc 1 est le bloc de l'initialisation. Ce bloc met en place 2 pointeurs de mémoire et un compteur. Le bloc 5 s'occupe de la mise à jour des pointeurs de mémoire et du compteur. La déclaration montrée dans le bloc apparaît bizarre si elles sont lues comme des équations algébriques. La déclaration Pointeur = Pointeur + 1 veut dire que la nouvelle valeur du Pointeur est obtenue en incrémentant la valeur précédente de 1. Ici l'organigramme est montré en détail, mais pour un long programme l'organigramme pourrait être simplifié.

5- Le programme en langage assembleur :

Adresse de la mémoire	Code Hex	Label	Instructions		Commentaires
			Opcode	Opérant	
XX00	21	START	LXI	H, XX50H	Fixer HL comme un Pointeur pour la mémoire source
XX01	50				
XX02	XX				
XX03	11		LXI	D, XX70H	Fixer DE comme un Pointeur pour la destination
XX04	70				
XX05	XX				
XX06	06		MVI	B, 10H	Fixer B pour compter 16 octets
XX07	10				
XX08	7E	NEXT	MOV	A, M	Obtenir L'octet de donnée à partir de la source
XX09	12		STAX	D	Stocké l'octet de donnée à la destination
XX0A	23		INX	H	Pointer HL à l'emplacement source suivant
XX0B	13		INX	D	Pointer DE à la destination suivante
XX0C	05		DCR	B	Un transfert est complété
XX0D	C2		JNZ	NEXT	Si le compteur n'est pas égale à zéro, retourner pour transférer l'octet suivant
XX0E	08				
XX0F	XX				

XX10	76		HLT		Fin du Programme
------	----	--	-----	--	------------------

Données	
XX00	37
↓	↓
XX00	98

4- Les opérations arithmétiques relatives à la mémoire

Les instructions relatives à la mémoire sont de la forme ;

ADD M/ SUB M : Ajouter/Soustraire le contenu de l'emplacement de mémoire à /du contenu de l'accumulateur.

INR M/DCR M : Incrémenter / Décrémenter les contenus de l'emplacement de mémoire.

Exemple 1 :

Ecrire les instructions nécessaires pour ajouter le contenu des remplacements de mémoire 2040H dans (A). et soustraire le contenu de l'emplacement de mémoire 2041H, de la première somme. Assumer que l'Accumulateur contient 30H, et l'emplacement 2040H contient 7FH.

Charger le nombre de 16-bit 2050H dans le registre pair HL en utilisant les opcodes LXI et MVI et expliquer la différence entre les deux instructions.

Solution :

La figure ci-dessus montre le contenu des registres et les instructions nécessaires pour l'exemple. Il faut d'abord préciser l'emplacement de mémoire en chargeant 2040H dans le registre pair HL

		Codes Machines	Mnemonics	Commentaires
	LXI	21	LXI H, 2040H	Charger les registres HL
		40		50 dans le registre L
		20		et 20H dans le registre H
		86	ADD M	
		23	INx H	
		96	SUB M	

Exemple 2 :

Charger 59H dans l'emplacement de mémoire 2040H et décrémenter le contenu de l'emplacement de mémoire.

Charger 90H dans l'emplacement de mémoire 2041H et décrémenter le contenu de l'emplacement de mémoire.

Solution :

La figure ci-dessus montre le contenu des registres et les instructions nécessaires pour l'exemple

H	LXI		L	Codes Machines	Mnemonics	Commentaires
	20	50		21	LXI H, 2040H	Charger les registres HL
				40		50 dans le registre L
				20		et 20H dans le registre H
				36	MVI M, 59H	
				59H		
				86	INR M	
				36	MVI M, 90H	
				90		
				35	DCR M	

Étude de cas de Programme : Addition avec carry

Énoncé du problème :

Six octets de données sont gardées dans des emplacements de mémoire commençant à partir de XX50H. Additionner tous les octets de données. Utiliser le registre B pour garder toute retenue (carry) générée durant l'addition des données au niveau de ports de sortie ou bien garder la somme au niveau de deux emplacements de mémoire consécutifs XX70H et XX71H.

Donnée (H)

A2, FA, DF, E5, 98, 8B.

Analyse du problème

Ce problème peut être analysé en relation avec l'organigramme au niveau de la figure suivante

Fig 7.12 page 227

- 1- Les instructions arithmétiques relatives à la mémoire peuvent être utilisées.
- 2- Le bloc de traitement de données peut être étendu pour tenir compte de la retenue.

Au niveau du premier bloc de l'organigramme de la figure, l'accumulateur et le registre qui contient le carry (B) doivent être réinitialisés pour être sûrs qu'ils sont vides et éviter des erreurs.

Après l'addition il est nécessaire de vérifier si l'opération génère un carry ou non (bloc 3). Si oui le registre carry est incrémenté par sinon il est contourner. Voir l'instruction ADC au niveau de la liste des instructions.

Programme

Le programme :

III- Les opérations logiques :Rotate

Ce sont les instructions relatives à la rotation des bits de l'accumulateur.

Les opcodes sont les suivants :

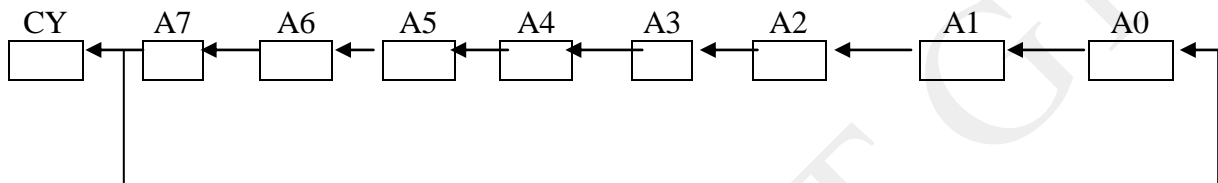
- R_L_C : Rotate Accumulator left
- R_A_L : Rotate Accumulator through carry

- RRC : Rotate Accumulator Right through carry
- RAR : Rotate Accumulateur Right

1- Les instructions :

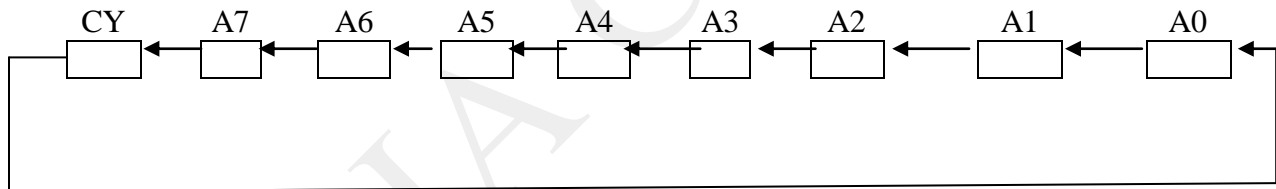
Ce groupe contient quatre instructions : deux sont pour la rotation à gauche et deux autres pour la rotation à droite. Seul le carry flag sera affecté lors de ces rotations.

a- RLC : Rotate Accumulateur Left :



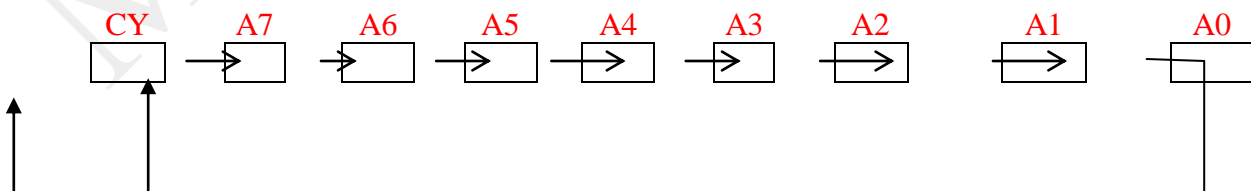
- Chaque bit est déplacé à position adjacente (suivante) vers la gauche et A7 devient A0.
- Le carry flag est modifié selon la nature du bit A7.

b- RAC : Rotate Accumulator Left through carry:

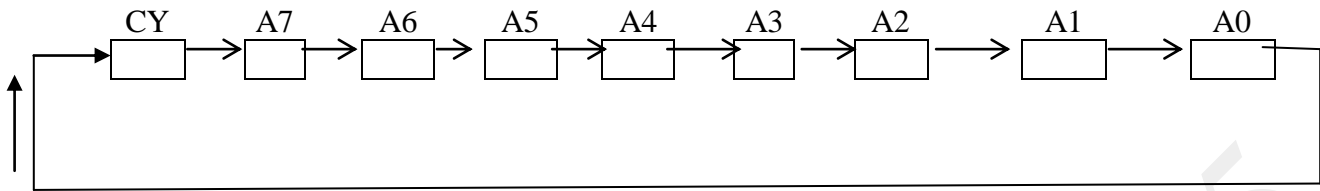


- Chaque bit est déplacé à position adjacente (suivante) vers la gauche et A7 devient le carry. Et le carry sera déplacé à A0.
- Le carry flag est modifié selon la nature du bit A7.

c- RRC : Rotate Accumulator Right:



d- RAR : Rotate Accumulator Right:



- Chaque bit est déplacé à position adjacente (suivante) vers la droite et A0 devient le bit du carry.
- Le carry flag est modifié selon la nature du bit A0.

Exemple :

Assumer que l'accumulateur contient AAH et CY = 0. Montrer le contenu de l'accumulateur après l'exécution de l'instruction RAR deux fois.

Solution :

Avant l'exécution de l'instruction	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	0	1	0	1	0	1	0	→AAH
Après l'exécution du 1 ^{er} RAR	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	1	0	1	0	1	0	1	0	0	→54H
Après l'exécution du 2 ^{ème} RAR	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	0	1	0	1	0	0	1	→A9H

Exemple :

Assumer que l'accumulateur contient AAH et CY = 0. Montrer le contenu de l'accumulateur après l'exécution de l'instruction RLC deux fois.

Solution :

Avant l'exécution de l'instruction	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	0	1	0	1	0	1	0	→AAH
Après l'exécution du 1 ^{er} RLC	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	1	0	1	0	1	0	1	0	1	→55H
Après l'exécution du 2 ^{ème} RLC	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	0	1	0	1	0	1	0	→AAH

Exemple :

Assumer que l'accumulateur contient 81H et CY = 0. Montrer le contenu de l'accumulateur après l'exécution des l'instructions RRC et RAR..

Solution :

Avant l'exécution de l'instruction	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	0	0	0	0	0	0	1	→81H
Après l'exécution du 1 ^{er} RRC	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	0	1	1	0	0	0	0	0	0	→C0H
Avant l'exécution de l'instruction	CY	A7	A6	A5	A4	A3	A2	A1	A0	
										→81H
Après l'exécution du 2 ^{ème} RLC	CY	A7	A6	A5	A4	A3	A2	A1	A0	
	1	0	1	0	0	0	0	0	0	→40H

IV- Les opérations logiques :Compare

Le jeu d'instruction du 8085 contient deux types d'instructions de comparaison : CMP et CPI

- CMP : Comparer avec le contenu de l'accumulateur
- CPI : Comparer immédiatement avec le contenu de l'accumulateur.

Instruction	Exemple	Commentaire
<u>CMP R</u>	CMP B	Elle <u>compare la donnée dans le registre (B) au contenu de</u> l'accumulateur
<u>CMP M</u>	CMP M	Elle <u>compare la donnée dans la mémoire au contenu de</u> l'accumulateur
<u>CPI 8-bit</u>	CPI 20H	Elle <u>compare la donnée citée (20H) au contenu</u> de l'accumulateur

Support de cours Microprocesseurs 2010-2011

Et indique si la donnée est >, < ou = au contenu de l'accumulateur en modifiant les flags cependant les contenus ne sont pas modifiés.

Si $(A) > (R/M)$	Z=0	CY=0
Si $(A) < (R/M)$	Z=0	CY=1
Si $(A) = (R/M)$	Z=1	CY=0

Exemple :

Écrire une instruction pour charger l'octet de données 64H dans l'accumulateur. Et vérifier si l'octet de données au niveau de l'emplacement de mémoire 2050H est égale au contenu de l'accumulateur. Si les données sont égales. Allez à l'emplacement OUT1

Solution:

Supposons que nous avons 9AH au niveau de l'emplacement 2050H.

LXI H, 2050H

MVI A, 64H

CPM M

JZ OUT

Les Compteurs et les temps de temporisation:

A- Le cycle de l'instruction

Une instruction est une commande assignée à l'ordinateur pour effectuer une opération ou une tâche.

Les étapes nécessaires pour qu'un processeur ou CPU accomplisse le retrait d'une instruction et les données nécessaires de la mémoire et de l'exécuter sont appelées le CYCLE DE L'INSTRUCTION (INSTRUCTION CYCLE ou IC). Le cycle de l'instruction est constitué du CYCLE DE RETRAIT ou RAPPEL (FETCH CYCLE ou FC) et du CYCLE DE L'EXECUTION (EXECUTION CYCLE ou EC). Les étapes nécessaires pour tirer un opcode de la mémoire constituent le FETCH CYCLE; alors que les étapes nécessaires pour tirer des données (s'il existe) de la mémoire et d'effectuer les opérations précisées dans une instruction constituent l'EXECUTION CYCLE.

Le temps nécessaire pour l'exécution d'une instruction (IC) est donnée par :

$$\underline{IC = FC + EC}$$

B- Les cycles de machine et les états (MACHINES CYCLES and STATES)

Les étapes nécessaires pour effectuer une opération d'accès à la mémoire ou un périphérique d'entrée/sortie (Retrait-Lecture-Ecriture) constitue un MACHINE CYCLE. Dans une machine cycle une opération de base comme le retrait d'un opcode ou la lecture ou l'écriture d'une mémoire ou d'une unité d'E/S est effectuée. Un CYCLE D'INSTRUCTION EST CONSTITUE DE PLUSIEURS MACHINES CYCLES. L'opcode d'une instruction est retirée durant la première machine cycle d'une instruction.

La plupart des instructions d'1 OCTET nécessite seulement 1 MACHINE CYCLE pour le retrait d'un opcode et l'exécution d'une instruction.

Les instructions de 2 ou 3 OCTETS nécessitent PLUS D'1 MACHINES CYCLES.

Exemple : La figure 3.6 montre le cycle d'instruction pour *MVI r, donnée* : Elle 2 machines cycles : un pour le retrait de l'opcode et l'autre pour la lecture de la donnée de la mémoire et son exécution. (voir figure 3.6).

La subdivision d'une opération durant un clock cycle est appelé STATES ou T-STATES. Les subdivisions sont des états internes ou internal states synchronisés avec le

clock du système. DONC UN CLOCK CYCLE DU SYSTEM EST APPELE STATE.

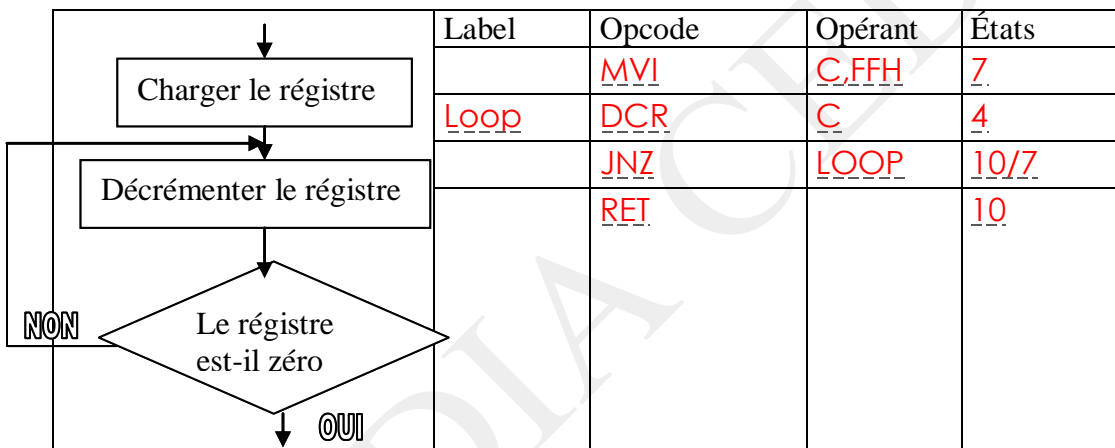
C- Les compteurs et les temps de temporisation

Concevoir un compteur est une fréquente application de la programmation. En effet on a toujours besoin d'un compteur pour un chronométrage précis entre deux évènements qui doivent se passer successivement. Les compteurs et les temps de temporisation sont utilisés en contrôle automatique industriel pour marquer le temps nécessaire avant d'initier le signal de contrôle. Ces compteurs et ces temps de temporisation trouvent aussi leurs applications au niveau des feux rouges, ou montres ou réveils électroniques.

Le temps de temporisation ou le compteur est mis en place de manière très simple en chargeant avec un registre le nombre approprié et l'incrémenter ou le décrémenter.

1. Le temps de temporisation utilisant un registre

La figure suivante montre l'organigramme et le temps de temporisation en utilisant un registre.



Le registre est chargé avec le compte FF (255 décimal) qui est décrémenter et le programme est exécuté autour de la boucle jusqu'à ce que le contenu du registre C soit à Zéro. Après le contrôle retourne au programme principal.

Comment calculer le temps de temporisation :

Si on regarde bien le programme on peut distinguer deux parties :

- Le temps d'exécution de la boucle (Time inside Loop): T_L
- Le temps d'exécution des instructions en dehors de la boucle (time outside Loop): T_o

Donc le temps total de la temporisation est :

$$TD = T_o + T_L$$

Voici une table donnant un récapitulatif de chaque instruction, de son nombre d'états du nombre de fois qu'elle est exécutée.

Instructions	États	Nombre d'exécution	Nombre total d'états par instruction
MVI	7T	<u>1</u>	<u>7T X 1</u>

DCR	4T	<u>N</u>	<u>4T X N</u>
JNZ	7/10T	<u>(N-1) + 1</u>	<u>10T X (N-1) + 7T X 1</u>
RET	10T	<u>1</u>	<u>10T X 1</u>

NB : Pour l'instructions JMP le nombre d'états est un peu particulier. Car le nombre d'états comme indiqué (7/10) varie selon le sens de l'exécution. Le nombre d'états pour JNZ est d 10 lors que le sens de l'exécution de JNZ vers DCR pour une réexécution de la boucle, alors que le nombre d'états n'est que de 7 lorsque l'exécution va de l'instruction JNZ vers RET.

Donc :

$$T_D = T_O + T_L$$

$$T_O = T \times (7 \times 1) + T \times (10 \times 1)$$

$$T_O = 17T$$

$$T_L = T \times (4N + 10 (N - 1) + 7)$$

$$T_L = T \times (14N - 3)$$

$$T_D = 17T + T \times (14N - 3)$$

$$T_D = T (17 + 14N - 3)$$

Considérons que la fréquence du clock pulse du microprocesseur est de 2 Mhz

$$T = \frac{1}{F} = \frac{1}{2Mhz} = 0,5 \times 10^{-6} s$$

$$N = FFH = 255d$$

$$T_D = 0,5 \times 10^{-6} s (17 + 14 \times 255 - 3)$$

$$T_D = 1792 \times 10^{-6} s$$

NB : les états de 7/10 peuvent être négligé et prendre 10états sur le nombre de fois que JNZ va être exécuté.

2. Le temps de temporisation utilisant deux registres

Le temps de temporisation peut être augmenté de manière significative en mettant en Oplace une boucle utilisant un registre pair avec un nom d 16-bit (FFFFH au maximum). Voici un exemple :

Label	Opcode	Operand	Etats
	<u>LXI</u>	<u>B, 2384H</u>	<u>10</u>
<u>LOOP</u>	<u>DCX</u>	<u>B</u>	<u>6</u>
	<u>MOV</u>	<u>A, C</u>	<u>4</u>
	<u>ORA</u>	<u>B</u>	<u>4</u>
	<u>JNZ</u>	<u>LOOP</u>	<u>10/7</u>
	<u>RET</u>		<u>10</u>

Nb : l'instruction ORA est utilisée pour fixer le flag Zéro car l'instruction DCX n'affecte pas le flag.

Voici une table donnant un récapitulatif de chaque instruction, de son nombre d'états du nombre de fois qu'elle est exécutée.

Instructions	Etats	Nombre d'exécutions	Nombre total d'états par instruction
LXI	10T	<u>1</u>	<u>10T X 1</u>
DCX	6T	<u>N</u>	<u>6T X N</u>
MOV	4T	<u>N</u>	<u>4T X N</u>
ORA	4T	<u>N</u>	<u>4T X N</u>
JNZ	10/7T	<u>(N-1) + 1</u>	<u>10T X (N-1) + 7 TX 1</u>
RET	10T	<u>1</u>	<u>10T X 1</u>

Donc :

$$T_D = T_O + T_L$$

$$T_O = T \times (7 \times 1) + T \times (10 \times 1)$$

$$T_O = 17T$$

$$T_L = T \times (6 \times N + A \times N + A \times N + 10 \times (N - 1) + 7 \times 1)$$

Si nous négligeons l'ajustement de l'instruction au niveau de JNZ

$$T_L = T(6N + 4N + 4N + 10N)$$

$$T_L = 24TN$$

$$T_D = 17T + 24TN$$

$$T_D = T(17 + 24N)$$

Considérons que la fréquence du clock pulse du microprocesseur est de 2 Mhz

$$T = \frac{1}{F} = \frac{1}{2\text{Mhz}} = 0,5 \times 10^{-6} \text{ s}$$

$$N = 2384H = 9092_d$$

$$T_D = 0,5 \times 10^{-6} \text{ s} (17 + 24 \times 9092)$$

$$T_D = 109112 \times 10^{-6} \text{ s}$$

3. Le temps de temporisation utilisant deux registres ; une boucle à l'intérieur d'une boucle.

Un temps de temporisation similaire à celui du registre pair peut être réalisé en utilisant deux boucles ; l'une à l'intérieure de l'autre.

Au niveau de l'exemple suivant le registre C est utilisé par la boucle e l'intérieur (loup1) et le registre B est utilisé par la boucle externe (loop 2).

La figure suivante montre l'organigramme et le temps de temporisation en utilisant un registre.

Label	Opcode	Operand	États
	MVI	B, 38H (N ₂)	7
Loop 2	MVI	C, FFH (N ₁)	7
Loop 1	DCR	C	4
	JNZ	LOOP 1	10/7
	DCR	B	4
	JNZ	LOOP 2	10/7
	RET		10

Voici une table donnant un récapitulatif de chaque instruction, de son nombre d'états du nombre de fois qu'elle est exécutée.

Instructions	États	Nombre d'exécutions	Nombre total d'états par instruction
MVI	7	1	7 x 1
MVI	7	N ₂	7 x N ₂
DCR	4	N ₂ x N ₁	4 (N ₂ x N ₁)
JNZ	10/7	[(N ₁ - 1) + 1] x N ₂	[(N ₁ - 1) x 10 + 7 x 1] x N ₂
DCR	4	N ₂	4 x N ₂
JNZ	10/7	(N ₂ - 1) + 1	[10 x (N ₂ - 1) + 1 x 7]
RET	10	1	10 x 1

Donc :

$$T_D = T_O + T_L$$

$$= T_O + T_{L1} + T_{L2}$$

$$T_O = T \times (7 \times 1) + T \times (10 \times 1)$$

$$T_O = 17T$$

$$T_{L1} = T(14N_1 + [10(N_1 - 1) + 7])$$

$$= T(14N_1 - 3)$$

$$N_1 = FFH = 255_d$$

$$T_{L1} = 0,5 \times 10^{-6} s (14 \times 255 - 3)$$

$$= 1783,5 \times 10^{-6} s$$

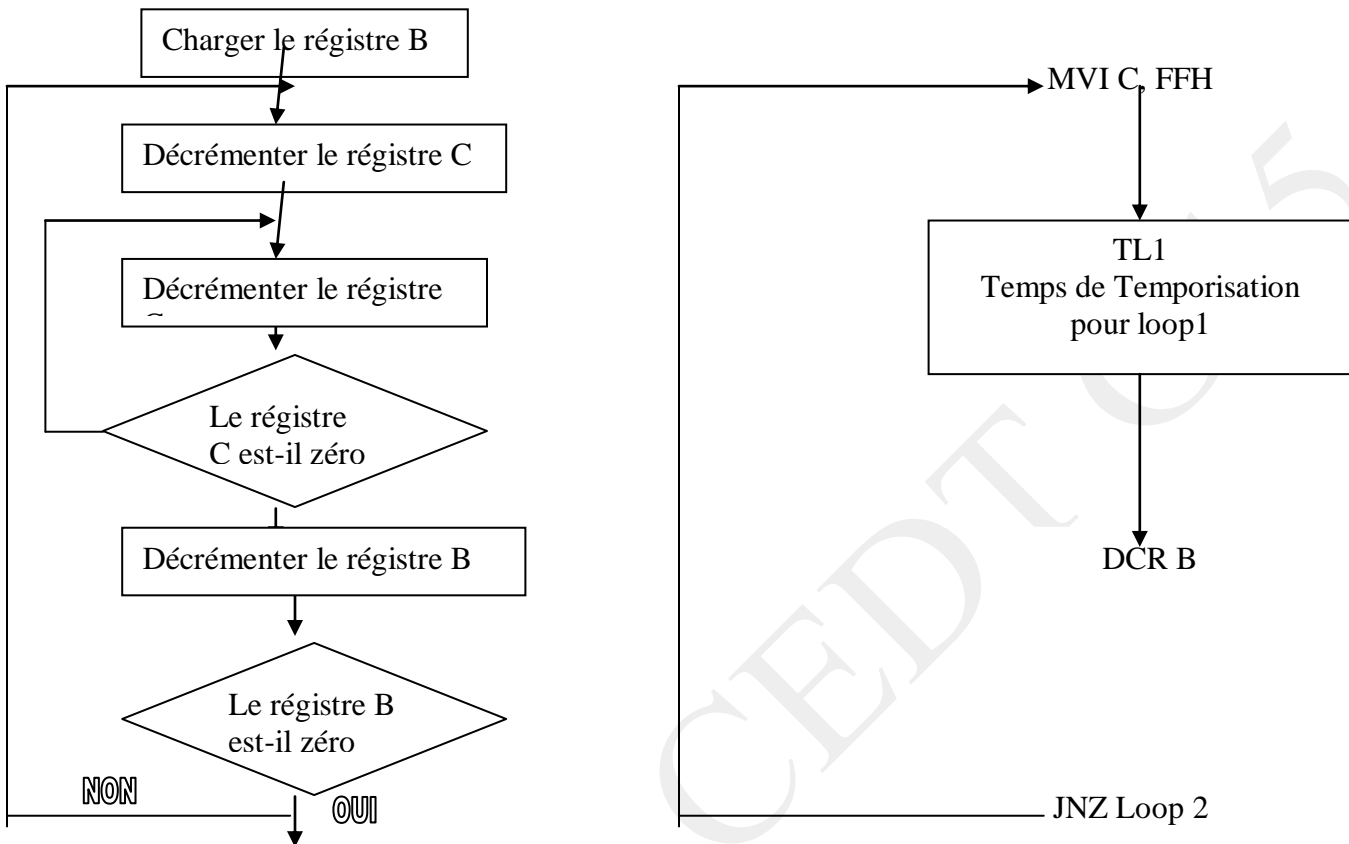
$$T_{L2} = T(7N_2 + N_2 T \frac{TL1}{T} + 4N_2 + 10[(N_2 - 1) + 1 \times 7])$$

$$= N_2 [21T + TL_1 - \frac{3}{N_2}]$$

Si on néglige - $\frac{3}{N_2}$

$$T_{L2} = 56 [21 \times 0,5 \times 10^{-6} s + 0,5 \times 10^{-6} s + 1783,5 \times 10^{-6} s \times 0,5 \times 10^{-6} s]$$

$$T_{L2} = 100.46 \text{ ms}$$



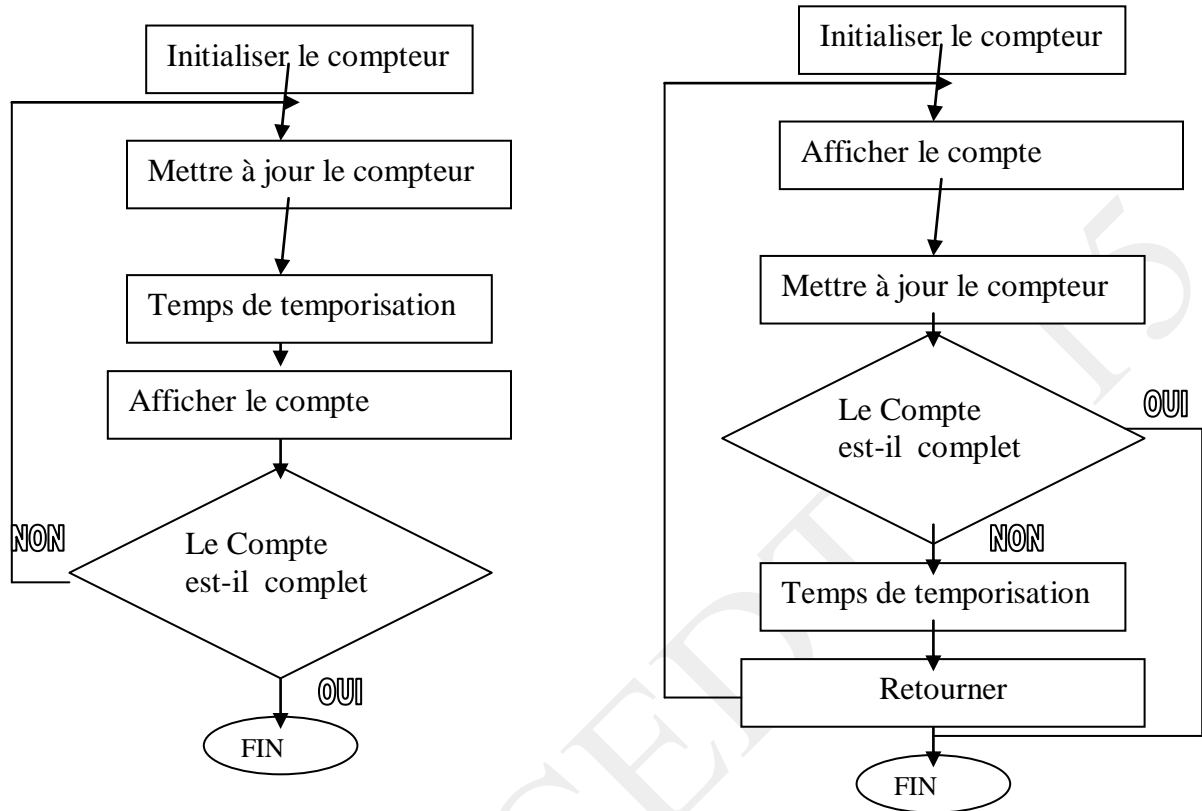
4. Autres techniques pour réaliser des temps de temporisation.

Comme déjà vu il peut être difficile de calculer le temps de temporisation en utilisant un programme ; et leur précision n'est pas toujours garantie.

Cependant dans de véritables applications, il existe des timers (chronomètres). Par Intel 8253, est un timer programmable sous forme de circuit intégré ; avec une très bonne précision mais l'inconvénient est qu'un circuit intégré supplémentaire va être utilisé augmentant la cherté du système.

5. Conception d'un compteur à l'aide d'un temps de temporisation

La figure suivante montre les différentes étapes de la conception d'un compteur à l'aide du temps de temporisation.



La figure montre les différents blocs ; mais leur ordre pourrait être modifié en fonction de la nature du problème comme montré par la figure suivante.. En effet le compteur peut être après ou avant l'initialisation.

Les Piles et les Sous-Programmes:

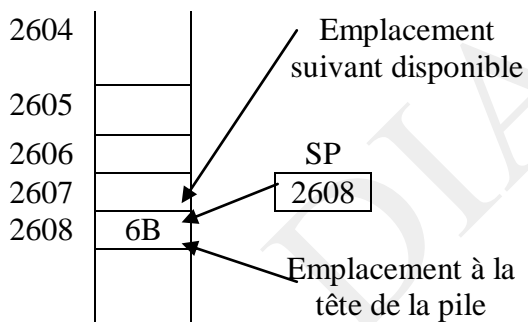
A- La Pile (Stack)

La pile au niveau du système du micro-ordinateur 8085 est une suite d'emplacement de mémoire RAM appelé ou désigné par un programmeur au niveau du programme principal.

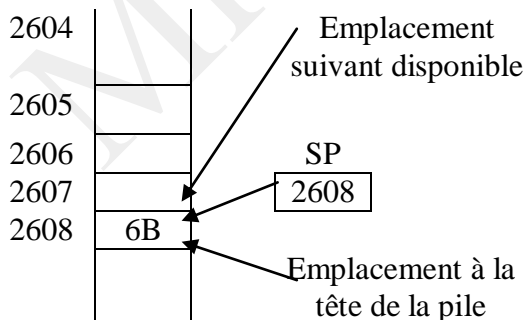
La tête de la pile est définie dans le programme en utilisant l'instruction LXI SP. Ceci va ainsi charger une adresse de 16-bit dans le registre Stack Pointer (SP) du Microprocesseur.

Les données sont stockées au niveau de la pile sous le principe de LIFO (Last In First Out). Le registre SP détient toujours l'adresse de l'emplacement de mémoire qui est à la tête de la pile.

L'instruction PUSH est utilisée pour stocker des données des registres pairs vers la pile ; deux octets à la fois, dans l'ordre inverse (voir figure). Tandis que l'instruction POP est utilisée pour transférer des données de la pile vers les registres pairs. (Voir figure).

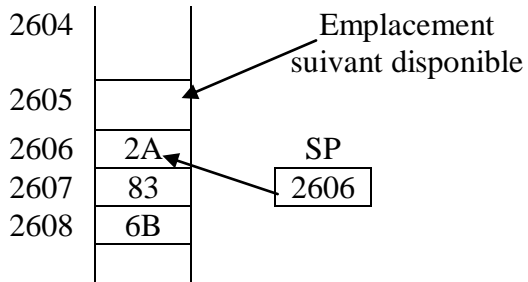


Push B

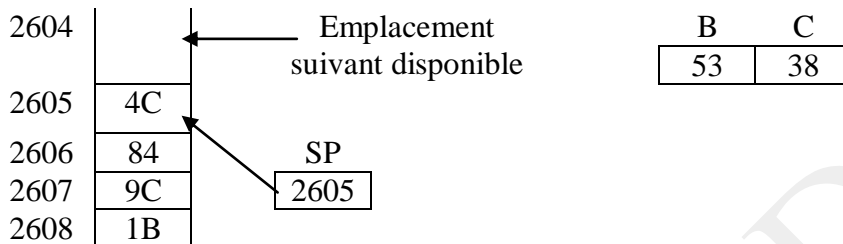


B	C
83	2A

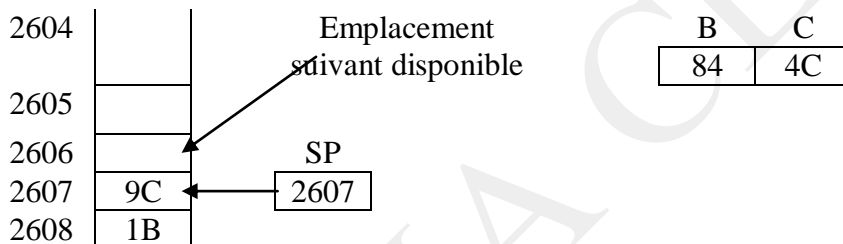
Après Push B



POP B



Après POP B



Le tableau suivant montre les instructions nécessaires utilisées avec les piles.

Instructions :

Opcode	Opérant	Description
LXI	SP, 16-bit	Charger le Registre Pair Pointer de la tête de la pile avec une adresse de 16-bit
PUSH	Rp	Stoker le contenu du registre pair au niveau de la pile - C'est une instruction d'1 octet
PUSH	B	Elle copie le contenu du registre pair précisé au niveau de la pile comme décrit en bas.
PUSH	D	

PUSH	H	Le registre qui pointe à la tête de la pile est décrémenté, et les contenus du registre de haut niveau (cad le registre B) sont copiés au niveau de l'emplacement par le registre qui pinte à la pile.
PUSH	PSW	Le registre qui pointe à la tête de la pile est encore décrémenté, et les contenus du registre de bas niveau (cad le registre C) sont copiés au niveau de cet emplacement. Les opérants B, D, H représente respectivement les registres BC, DE, et HL. L'opérant PWS représente les registres A et le flag.
POP	Rp	Retirer le contenu à la de la pile et l'envoyer au niveau du registre pair. - C'est une instruction d'1 octet
POP	B	Elles copie le contenu des deux premiers emplacements de mémoire à la tête de la pile au niveau du registre pair précisé.
POP	D	
POP	H	D'abord le contenu de l'emplacement de mémoire indiqué par le pointeur de la pile est copié au niveau du registre de bas niveau (cad le registre L), ensuite le pointeur de la pile est décrémenté par 1 Le contenu de l'emplacement de mémoire suivant est copié au niveau du registre de haut niveau (cad le registre H), ensuite le pointeur de la pile est encore décrémenté par 1
POP	PSW	Elle décrémente le contenu de la mémoire et par 1 et non l'adresse de la mémoire

B- Les sous Programmes (Subroutines)

Lorsque l'on écrit un programme, un même groupe d'instructions effectuant une certaine fonction peut apparaître plusieurs fois dans le programme.

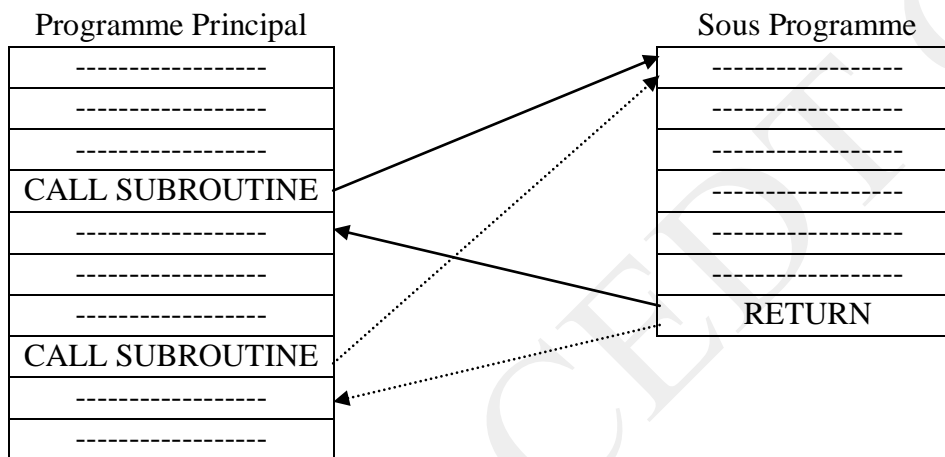
Le concept de sous programme est alors utilisé pour éviter la répétition des petits programmes. Donc un Sous-programme (Subroutine) est un

groupe d'instructions écrites séparément du programme principal pour effectuer une fonction précise qui se répète plusieurs fois au niveau du programme principal.

Le microprocesseur du 8085 contient des instructions pour mettre en place les sous programmes : **CALL** (Appeler un sous programme) et **RET** (Retourner au sous programme vers le programme principal).

Il peut y avoir plusieurs sous programmes différents dans un même programme principal.

La figure suivante montre la forme CALL-RETEURN.



Instructions :

Opcod e	Opérant	Description
CALL	16-bit Adresse de la mémoire du sous-programme	Appeler le sous programme de manière inconditionnelle C'est une instruction de 3 octets Garde le contenu du Program Counter (la prochaine instruction) au niveau de la pile Décrémente le registre Pointeur de la pile par 2. Se déplace inconditionnellement au niveau de l'emplacement précisé par le 2 ^{ème} et les 3 ^{èmes} octets. Cette instruction est complétée par l'instruction RETURN au niveau du sous programme.
RET		Retourner du sous programme vers le programme principal de manière inconditionnelle. - C'est une instruction d'1 octet - Insérer les deux octets au niveau de la tête de la pile au niveau du Program Counter et incrémenter le registre Pointeur de la pile par 2.

Exemple : La Multiplication en utilisant le Microprocesseur 8085.

En calculant les expressions $X = A \times B - C \times D$ considérant que A, B, C et D sont des nombres Hexadécimal et $X < FFH$ et $FFH > A \times B > C \times D$. Donc la multiplication est effectuée deux fois. Pour éviter une répétition du même jeu d'instruction dans le même programme, un programme peut être développé séparément pour la multiplication et appelé chaque fois que ce sera nécessaire.

Programme :

Adresse de la mémoire	Code machine	Instructions		Commentaires
		Opcode	Opérant	
4200	21 20 42	LXI	H, Adresse de A	Initialiser l'adresse 4220H où la donnée A est gardée
4203	11 21 42	LXI	H, Adresse de B	Initialiser l'adresse 4221H où la donnée B est gardée
4206	CD 00 43	CALL	MULTI	Appeler le sous programme de la multiplication où l'adresse commence par 4300H
4209	32 24 42	STA	4224	Garder la donnée A x B de l'Accum vers 4224H
420C	21 22 42	LXI	H, Adresse de C	Initialiser l'adresse 4222H où la donnée C est gardée
420F	11 23 42		H, Adresse de D	Initialiser l'adresse 4223H où la donnée D est gardée
4212	CD 00 43	CALL	MULTI	Appeler le sous programme de la multiplication où l'adresse commence par 4300H
4215	4F	MOV	A, C	Transférer la donnée A x D de Accum vers Reg C
4216	3A 24 42	LDA	4224	Charger la donnée A x B qui gardée au niveau de l'emplacement de mémoire 4224H au niveau de L'Accum.
4219	91	SUB	C	Soustraire contenu du Reg C de l'Accum.
421A	21 24 42	LXI	H, 4224	Initialiser 4224H où le résultat doit être gardé.
421D	77	MOV	M, A	Transférer le contenu de l'Accum. au niveau de 4224H
421E	76	HLT		Halte
4220	A			
4221	B			
4222	C			
4223	D			

Dans ce programme les données d'entrées sont stockés au niveau des emplacement de mémoire à partir de 4220H à 4223H. et le résultat après la compilation du programme sera obtenu à partir de l'emplacement de mémoire 4224H

Sous Programme de la Multiplication :

Adresse de	Code	label	Instructions	Commentaires
------------	------	-------	--------------	--------------

Support de cours Microprocesseurs 2010-2011

la mémoire	machine		Opcode	Opérant	
4300	46		MOV	B, M	Transférer la 1 ^{ère} donnée à partir de la mémoire vers le Reg. B
4301	EB		XCHG		Échanger les contenus de HL avec le s contenus de DE
4302	4E		MOV	C, M	Transférer la 2 ^{ème} donnée à partir de la mémoire vers le Reg. C
4303	AF		XRA	A	Réinitialiser (mettre à zéro) le carry et l'Accum.
4304	80	LOOP	ADD	B	Ajouter le contenu du Reg. B au niveau de l'Accum.
4305	0D		DCR	C	Décrémenter le Reg C.
4306	C2 04 43		JNZ	LOOP	Si le contenu du Reg. C n'est pas égale à zéro ; allez à LOOP
4309	C9		RET		Retourner vers le programme principal