Chapitre 3

Etude d'un microprocesseur 8 bits

3.1 Introduction

3.1.1 *Contexte Historique*

L'avènement des microprocesseurs a marqué un tournant majeur dans l'histoire de l'informatique. Parmi ces pionniers, le microprocesseur 8085 occupe une place significative en tant que précurseur des architectures de microprocesseurs modernes. Développé au milieu des années 1970 par Intel, le 8085 a été l'un des premiers microprocesseurs 8 bits à usage général, ouvrant la voie à une nouvelle ère de l'informatique embarquée.

3.1.2 *Importance du Microprocesseur 8085*

Le microprocesseur 8085 a joué un rôle crucial dans le développement de systèmes informatiques variés, allant de l'automatisation industrielle aux dispositifs électroniques grand public. Sa conception robuste et sa polyvalence ont contribué à façonner les fondements des microprocesseurs qui ont suivi. Ce cours vise à explorer en profondeur l'architecture, les fonctionnalités et les applications pratiques du microprocesseur 8085.

3.1.3 *Objectifs du cours*

1. Comprendre l'architecture externe et interne du microprocesseur 8085.

2. Explorer les concepts de bus et de bus multiplexé.
3. Découvrir les mécanismes de mappage et leur importance.
4. Étudier les différentes instructions et formats d'instructions.
5. Apprendre les techniques d'interfaçage avec la mémoire et les périphériques.
6. Acquérir des compétences pratiques en programmation en langage assembleur 8085.
7. Appliquer les connaissances à travers des travaux pratiques.

3.2 Architecture Externe

Le microprocesseur 8085 se présente sous la forme d'un boîtier, également appelé puce, qui adopte le format Dual In-Line (DIL) (DIL) avec 40 broches. Ce boîtier DIL est une configuration courante pour les composants électroniques, caractérisée par deux rangées de broches alignées en parallèle de chaque côté de la puce. Chaque broche a une fonction spécifique dans la connectivité du microprocesseur avec d'autres composants du système.
La Figure 3.1 présente en détail la configuration physique et logique du microprocesseur, mettant en évidence la structure Dual In-Line (DIL) composée de 40 broches. Chaque broche est clairement identifiée et nommée en fonction de sa fonction spécifique dans le fonctionnement global du microprocesseur. En complément, la Figure 3.2 offre une représentation schématique de l'architecture en bus associée à ces broches. Cette structure illustre la manière dont les signaux sont acheminés et organisés, en mettant en évidence les relations fonctionnelles entre les différents groupes de broches, tels que les lignes de données, les lignes d'adresses et les signaux de contrôle.

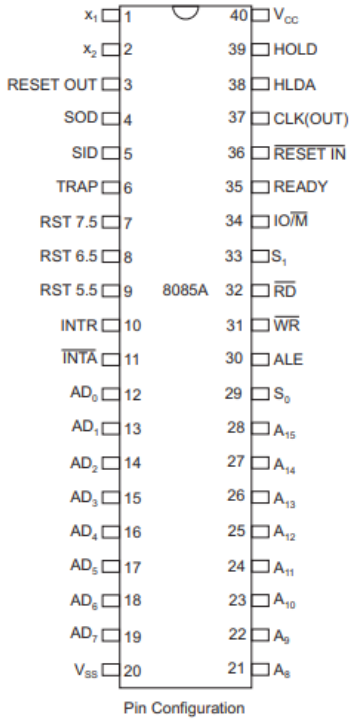

Figure 3.1: Brochage du 8085

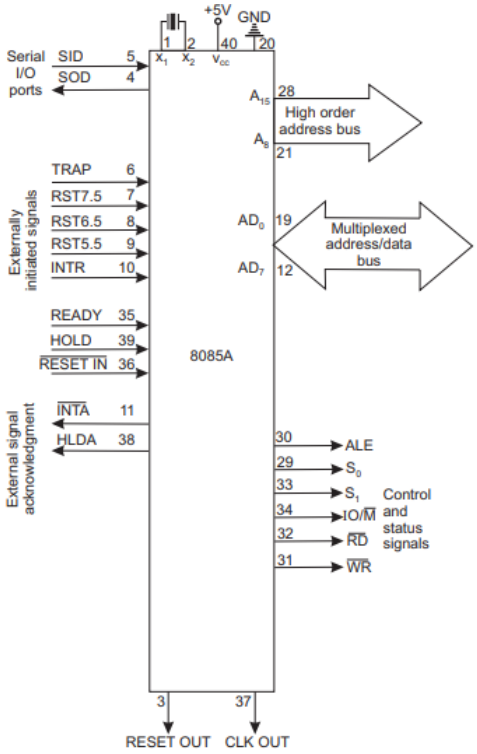

Figure 3.2: Brochage fonctionnel.

3.2.1 *Notion de Bus*

Le **bus** dans le contexte du microprocesseur 8085 est un ensemble de fils de communication qui permettent le transfert d'informations entre différentes parties du système. Il agit comme une autoroute numérique, facilitant l'échange de données, d'adresses et de signaux de contrôle entre le processeur, la mémoire et les périphériques.

3.2.1.1 Définition et Rôle du Bus

Le bus sert de canal principal pour la transmission des données.Il comprend généralement trois types de lignes :

1. **Bus d'Adresse :** Utilisé pour transmettre des adresses, spécifiant la localisation des données dans la mémoire ou les périphériques. Il transporte les adresses générées par le processeur vers la mémoire ou les périphériques.
2. **Bus de Données :** Responsable du transfert des données entre le processeur, la mémoire et les périphériques. Assure le transfert bidirectionnel des données entre le processeur et la mémoire/périphériques.
3. **Bus de Contrôle :** Transmet des signaux de contrôle qui régissent le flux d'informations et les opérations du système. Il Gère les signaux de contrôle tels que lecture/écriture, validation d'adresse, etc.

3.2.2 *Bus Multiplexe*

Le bus multiplexe est une technique où plusieurs signaux sont transmis sur un même chemin physique, mais à des moments différents. Cette approche permet de réduire le nombre de fils nécessaires pour les différentes lignes de bus, optimisant ainsi l'utilisation des ressources.

3.2.2.1 Concept de Multiplexage

Dans le bus multiplexe du 8085, une même ligne de bus est utilisée pour transmettre des adresses et des données à des moments distincts. Lors d'une opération d'écriture, le bus transporte d'abord l'adresse, puis il est réutilisé pour transmettre les données. Pour une opération de lecture, le bus transporte l'adresse, puis change de fonction pour transporter les données de la mémoire vers le processeur.

3.2.2.2 Avantages et limitations du Multiplexage

1- **Avantages :**

— **Économie de Ressources :** La technique multiplexe permet de réduire le nombre de fils nécessaires pour les bus, économisant ainsi des ressources.

— **Flexibilité :** En utilisant la même ligne pour différents types d'informations, le bus devient plus flexible et adaptable à diverses opérations.

2- **Limitations**

— **Complexité :** La gestion des opérations multiplexées peut ajouter une certaine complexité au système.

— **Vitesse Limitée :** Les opérations multiplexées peuvent ralentir la vitesse globale de transfert d'informations.

3.2.3 *Groupes de signaux du 8085*

Les signaux du 8085 peuvent être classés en sept groupes selon leurs fonctions. Ceux-ci sont :

1. Signaux d'alimentation et de fréquence
2. Bus de données et d'adresses
3. Bus de contrôle
4. Signaux d'interruption

5. Signaux d'E/S série
6. Signaux DMA (DMA)
7. Signaux de réinitialisation.

La largeur de Bus de Données (BD) et du Bus Adreesses (BA) du 8085 est respectivement de 8 bits (1 octet) et de 16 bits (2 octets). Alors que le BD est bidirectionnel par nature, le BA est unidirectionnel. Étant donné que le microprocesseur peut entrer ou sortir des données de lui-même, BD est bidirectionnel. De plus, le microprocesseur adresse/communique avec des circuits intégrés périphériques via le bus d'adresse, d'où sa nature unidirectionnelle. L'adresse sort via le BA du microprocesseur.

Le microprocesseur 8085 communique via son bus d'adresse de largeur 2 octets - le byte inférieur AD0 à AD7 (broches 12 à 19) et le byte supérieur D8 à D15 (broches 21 à 28). Ainsi, il peut adresser un maximum de 2^{16} emplacements mémoire différents. Chaque adresse (emplacement mémoire) peut contenir 1 octet de données/instruction. Par conséquent, la capacité d'adressage maximale du 8085 est de : Capacité d'adressage=2^{16} cases mémoires

3.2.4 *Démultiplexage du bus d'adresse/données et séparation des signaux de contrôle*

Le schéma de connexions est présenté ci-dessous. Le bus de commande octal 74LS244 pilote le bus d'adresse de rang supérieur A15 - A8, tandis que le registre à verrou 74LS373 pilote le bus d'adresse de rang inférieur A7 - A0 (avec l'aide du signal ALE). Le pilote bidirectionnel de bus 74LS245 pilote le bus de données D7 - D0, tandis que le 74LS138 (une puce décodeur 3 vers 8) génère, à ses broches de sortie—IOW, IOR,MEMW,MEMR, des signaux à partir des signaux IO/M,RD et WR du 8085.

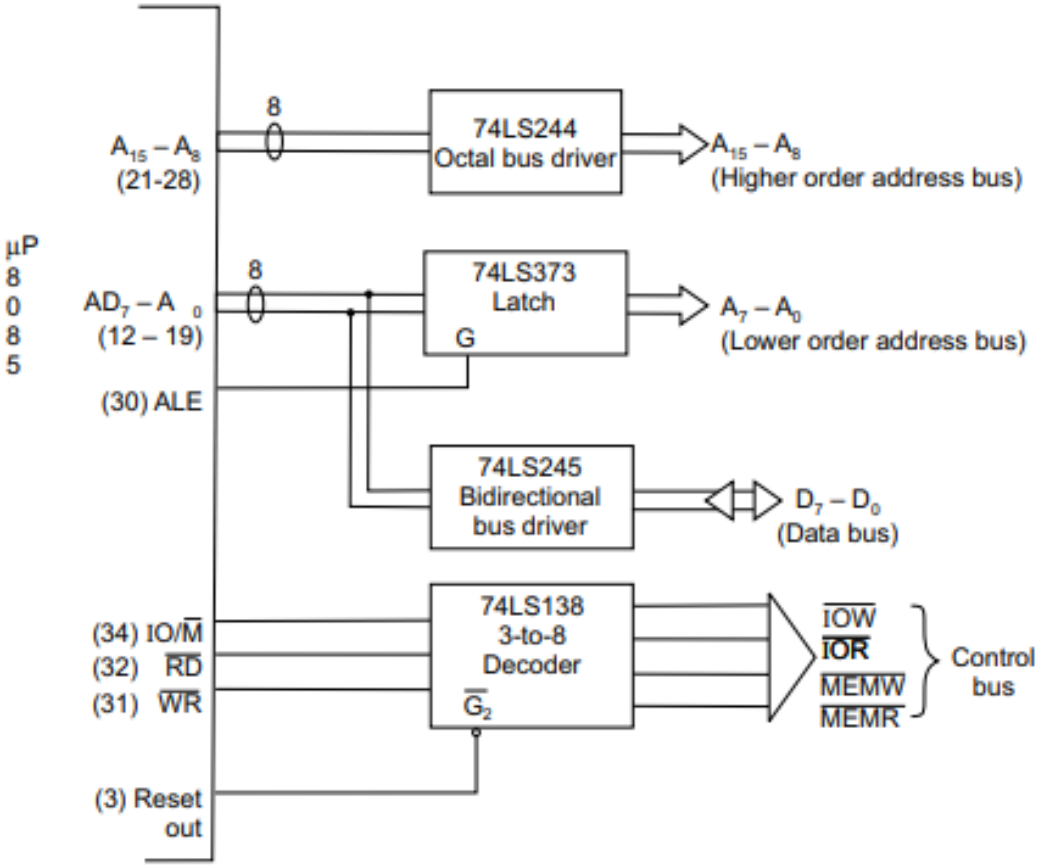

Figure 3.3: Démultiplexage du bus d'adresse/données et séparation des signaux de contrôle

3.3 Architecture Interne

l'architecture Interne est illustrée sur la figure 3.4. Les différents blocs fonctionnels du 8085 sont les suivants :

1. Registres
2. Unité arithmétique et logique (ALU)
3. Tampon d'adresse
4. Verrou d'adresse incrémenteur/décrémenteur
5. Contrôle des interruptions
6. Contrôle d'E/S série
7. Circuit de synchronisation et de contrôle
8. Décodeur d'instructions et encodeur de cycles machine.

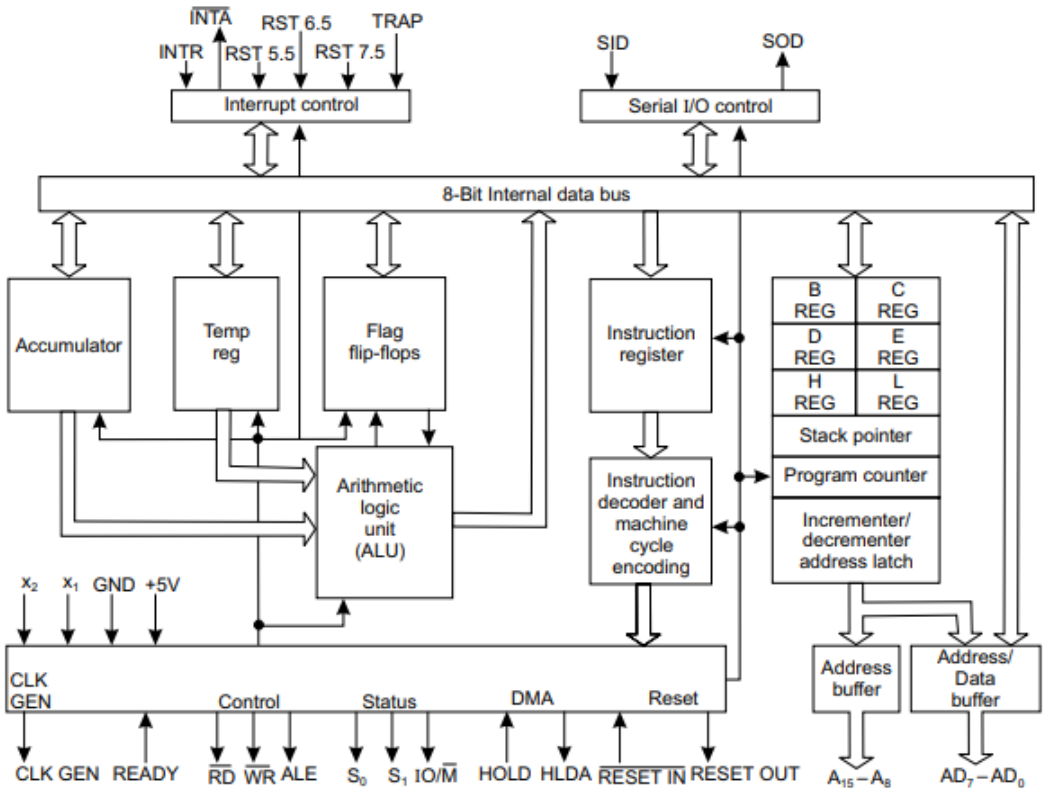

Figure 3.4: Architecture interne du 8085

Un microprocesseur qui possède n lignes de données est appelé un microprocesseur n-bits, c'est-à-dire que la largeur du bus de données détermine la taille du microprocesseur. Ainsi, un microprocesseur 8 bits comme le 8085 peut traiter 8 bits de données à la fois.Le 8085 fonctionne à une fréquence de 3 MHz, et la fréquence minimale de fonctionnement est de 500 kHz. La version 8085 A-2 fonctionne à une fréquence maximale de 5 MHz. La Figure 3.5 illustre le diagramme bloc du générateur d'horloge intégré du 8085.

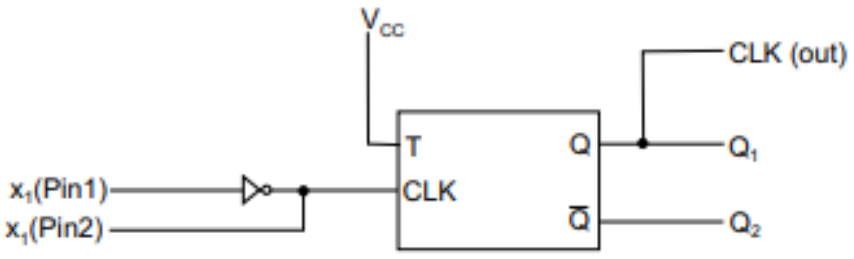

Figure 3.5: Diagramme bloc du générateur d'horloge intégré

Le générateur d'horloge intégré, les circuits accordés (LC) ou (RC), le cris-

tal piézo-électrique ou la source d'horloge externe agissent en tant qu'entrée pour générer l'horloge. Le basculement T, illustré à la figure 3.5, divise la fréquence d'entrée par 2. Ainsi, la fréquence de sortie du 8085 (obtenue à partir de la broche 37) est la moitié de la fréquence d'entrée. alors que Le signal (CLK) (sortie) obtenu à partir de la broche 37 du 8085 est utilisé pour synchroniser les dispositifs externes.

3.3.1 *Registres Internes du 8085*

Le microprocesseur Intel 8085 intègre plusieurs registres, classés en deux catégories :

1. Registres généraux (B, C, D, E, H, L) : Ces registres de 8 bits, utilisables par paires (BC, DE, HL), servent au stockage temporaire de données et d'adresses. Le couple HL joue un rôle particulier, souvent employé comme pointeur mémoire pour adresser indirectement des données.
2. Registres spéciaux :

   - Accumulateur (A) : Registre clé pour les opérations arithmétiques et logiques, stockant les résultats intermédiaires.

   - Registre d'état (Flags) : Contient des indicateurs binaires (Zero, Carry, Sign, etc.) reflétant le résultat des opérations (ex : un débordement active le flag Carry).
3. Unité de Traitement Central (CPU) : Ses fonctions principales sont doubles : d'une part, l'exécution des instructions via leur décodage et leur traitement par l'Unité Arithmétique et Logique (UAL), et d'autre part, la gestion des données, qui implique le transfert d'informations entre les registres, la mémoire et les périphériques.
4. Compteur de Programme (PR) et Registre d'Instruction (IR) Fonc-

tionnement :

PC : Pointeur 16 bits indiquant l'adresse de la prochaine instruction à exécuter (incrémenté automatiquement).

IR : Stocke l'instruction en cours de décodage après son extraction de la mémoire.

Relation avec le flux d'exécution :

Le PC assure la séquence linéaire ou sautée (via branchements) des instructions, tandis que l'IR permet à l'unité de contrôle d'identifier l'opération à effectuer.

3.4 Mapping

3.4.1 *principe et fonction du Mapping*

Dans le contexte du microprocesseur 8085, le mapping fait référence au processus d'attribution des adresses de mémoire ou des périphériques aux broches et au système de bus. Ce mécanisme permet au microprocesseur d'interagir efficacement avec les mémoires (ROM/RAM) et les périphériques d'entrée/sortie en associant des plages d'adresses spécifiques à ces composants. Le mapping mémoire établit la correspondance entre les adresses logiques générées par le processeur et les adresses physiques des composants connectés, garantissant une gestion cohérente et structurée des ressources du système.

La création d'une table de mapping facilite grandement le câblage du microprocesseur 8085 avec les mémoires et les différents périphériques. Cette table sert de guide clair pour définir les connexions nécessaires et s'assurer que les plages d'adresses et les signaux de contrôle sont correctement attribués, minimisant ainsi les erreurs et optimisant la conception globale du système.

Exemple de Mappage

Hypothèses d'un cahier de charges :

— Mémoire Morte **ROM** de 16 Koctets
— Mémoire Vive **RAM** de 8 Koctets
— Convertisseur Analogique/Numérique **CAN** de 8 octets
— Programmable Peripheral Interface **PPI** de 4 octets

L'élaboration de la table de Mapping s'effectue en deux principales étapes .

1. **Etape 1 :** À partir des capacités mémoire de chaque composant, on détermine le nombre de lignes d'adressage nécessaires pour chaque circuit. CM(ROM)= 16 koctets=2^{14}= 2^{BA}*BD/8
   BD=8 => BA=14 lignes
   De la même manière, on peut déduire les lignes d'adressage de la RAM comme suit : BA (RAM) = 13 lignes
   BA(CAN)= 3 lignes
   BA(PPI)= 2 lignes

2. **Etape 2 : Elaboration de la table de Mapping** L'élaboration de la table de mapping pour le microprocesseur 8085 constitue une étape essentielle pour organiser l'allocation des espaces mémoire et des périphériques, tout en respectant les lignes d'adressage et de décodage. Dans un système intégrant une EPROM de 16 Ko, une RAM de 8 Ko, un PPI (Programmable Peripheral Interface) de 4 octets et un CAN (Convertisseur Analogique-Numérique) de 8 octets, il est impératif d'attribuer les lignes d'adresse avec soin afin d'éviter tout chevauchement. Le 8085, avec ses 16 lignes d'adresse (A0 à A15), offre un espace d'adressage total de 64 Ko. De plus, un circuit de décodage, souvent réalisé à l'aide de portes logiques ou de décodeurs spécialisés, est indispensable pour activer chaque périphérique en fonction des bits d'adresse correspondants. Par ailleurs, les lignes non exploitées dans les plages d'adresses peuvent être réservées pour des extensions ultérieures, garantissant ainsi une évolutivité du système.

Plage d'adresses	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Circuits
00000H-3FFFH	0	0	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	ROM
4000H-5FFFH	0	1	X	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	RAM
8000H-8007H	1	0	X	X	X	X	X	X	X	X	X	X	X	0..1	0..1	0..1	CAN
C000H-C003H	1	1	X	X	X	X	X	X	X	X	X	X	X	X	0..1	0..1	PPI

Tableau 3.1: table de mapping

3.5 Jeux d'instruction du Microprocesseur INTEL 8085

La figure 3.6 représente le format d'instruction du microprocesseur 8085, un aspect clé de son architecture et de son fonctionnement. Ce format est divisé en deux parties principales :

1. **Opcode (Code d'Opération) :** La première partie de l'instruction occupe 1 octet et est utilisée pour spécifier l'opération à exécuter. Le code d'opération indique au microprocesseur quelle tâche doit être réalisée, comme une opération arithmétique (addition, soustraction), logique (AND, OR), ou une instruction de contrôle (saut, appel, retour). Chaque code d'opération est unique et correspond à une opération spécifique dans le jeu d'instructions du microprocesseur 8085.

2. **Données (Data) :** La seconde partie, optionnelle, contient les données ou les adresses nécessaires à l'exécution de l'instruction. Cette section peut varier en taille, étant soit de 0, 1, ou 2 octets selon le type d'instruction :

   0 octet : Les instructions qui n'ont pas besoin d'opérandes explicites, comme des opérations internes (NOP, HALT).

   1 octet : Les instructions nécessitant une seule donnée, comme l'immediat addressing (MVI, ANI).

   2 octets : Les instructions qui incluent une adresse ou une donnée de

16 bits, comme LXI ou JMP.

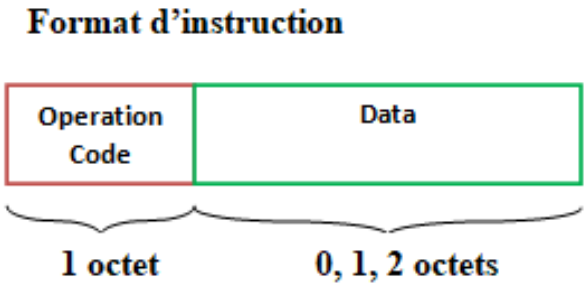

Figure 3.6: Format d'instruction du Microprocesseur 8085

Points Clés : Ce format compact et flexible permet de gérer efficacement la variété d'opérations disponibles sur le microprocesseur 8085. Le découpage clair entre le code d'opération et les données permet une interprétation rapide et précise des instructions par le processeur. L'utilisation d'un octet fixe pour le code d'opération assure la simplicité du décodage, tandis que l'ajout optionnel de données offre une flexibilité accrue dans les opérations. Ce format met en évidence la manière dont l'architecture du 8085 a été conçue pour optimiser la performance tout en minimisant l'espace mémoire requis pour coder les instructions.
Les instructions du microprocesseur 8085 sont regroupées selon leur fonction.

3.5.1 Instruction de transfert

Les instructions de transfert, également appelées instructions MOVE, permettent de transférer des données d'une source à une destination sans modifier la valeur de la source. Ces instructions sont essentielles pour manipuler les données entre les registres, la mémoire, et l'accumulateur dans le microprocesseur 8085. Elles se présentent sous plusieurs formes, avec une syntaxe adaptée au type de transfert.

1. **Transfert entre registres :**

   Syntaxe : MOV Rd, Rs Où Rd est le registre de destination et Rs le

registre source.

Exemple : MOV B, C ; Transfère le contenu du registre C dans le registre B.

2. **Transfert entre un registre et l'accumulateur :**

   Syntaxe :

   MOV A, Rs ou MOV Rs, A L'accumulateur (A) peut recevoir ou envoyer des données à un autre registre. Exemple :

   MOV A, D ; Transfère le contenu du registre D dans l'accumulateur.

   MOV E, A ; Transfère le contenu de l'accumulateur dans le registre E.

3. **Transfert entre un registre et une mémoire :**

   Syntaxe :

   MOV M, Rs ou MOV Rs, M Où M désigne l'emplacement mémoire pointé par les registres H et L (HL pair). Exemple :

   Transfert entre un registre et une mémoire :

   Syntaxe : MOV M, Rs ou MOV Rs, M Où M désigne l'emplacement mémoire pointé par les registres H et L (HL pair). Exemple :

   MOV A, M ; Charge le contenu de l'emplacement mémoire pointé par HL dans l'accumulateur.

   MOV M, A ; Sauvegarde le contenu de l'accumulateur dans l'emplacement mémoire pointé par HL.

4. **Transfert entre l'accumulateur et une mémoire :** Syntaxe :

   MOV A, M ou MOV M, A Cette instruction permet d'échanger des données entre l'accumulateur et un emplacement mémoire.

   Exemple :

   MOV A, M ; Charge le contenu de l'emplacement mémoire pointé par HL dans l'accumulateur.

   MOV M, A ; Sauvegarde le contenu de l'accumulateur dans l'emplacement mémoire pointé par HL.

5. **Transfert immédiat vers un registre ou une mémoire :**

   Syntaxe :

   MVI Rd, data ou MVI M, data Cette instruction charge directement une donnée (immédiate) dans un registre ou une mémoire.

   Exemple :

   MVI A, 0x25 ; Charge la valeur 0x25 (hexadécimale) dans l'accumulateur.

   MVI M, 0x10 ; Charge la valeur 0x10 dans la mémoire pointée par HL.

3.5.2 Instruction de Chargement et de stockage

Dans le microprocesseur 8085, les instructions de chargement et de stockage permettent de transférer des données entre le registre accumulateur (A), la mémoire et les registres. Ces instructions jouent un rôle crucial dans la manipulation des données en mémoire et en registre.

1. **Instruction de Chargement : LDA (Load Accumulator Direct)**

   L'instruction **LDA** charge le contenu d'une adresse mémoire donnée directement dans l'accumulateur.

   Syntaxe :

   LDA 16-bit_address

   Ici, 16-bit_address représente l'adresse mémoire à partir de laquelle les données sont lues.

   La taille de cette instruction est de 3 octets (1 pour l'opcode et 2 pour l'adresse).

   Exemple :

   LDA 2000H

   Dans cet exemple, le contenu de l'adresse mémoire 2000H sera chargé dans l'accumulateur (A).

2. **Instruction de Stockage : STA (Store Accumulator Direct)**

   L'instruction STA stocke le contenu de l'accumulateur dans une adresse

mémoire spécifique.

Syntaxe :

STA 16-bit_address

Ici, 16-bit_address est l'adresse mémoire où les données de l'accumulateur seront écrites.

Cette instruction est également de 3 octets.

Exemple :

STA 3000H

Dans cet exemple, le contenu actuel de l'accumulateur sera stocké à l'adresse mémoire 3000H.

3.5.3 Instructions Arithmétique et logique

Le microprocesseur 8085 est un microprocesseur 8 bits qui utilise un ensemble d'instructions pour effectuer des opérations sur des données. Les instructions arithmétiques et logiques constituent une catégorie essentielle qui permet d'effectuer des calculs et des manipulations sur les données stockées dans les registres ou la mémoire.

3.5.3.1 Instructions Arithmétiques

Le microprocesseur 8085 prend en charge uniquement des instructions pour l'addition et la soustraction.

1. **instructions d'addition**

   1.1 Instructions d'addition Sans retenus ADD (Without Carry)

   *** ADD :** Additionne le contenu d'un registre ou d'une adresse mémoire à l'accumulateur.

   Syntaxe :

   ADD R (ou) ADD M

   R : Registre (B, C, D, E, H, L).

   M : Contenu de l'adresse mémoire pointée par le registre HL.

Exemple : MOV A, B ; Charge le contenu du registre B dans l'accumulateur.

ADD C ; Additionne le contenu du registre C à l'accumulateur.

*** ADI :**

Additionne une donnée immédiate à l'accumulateur

Syntaxe :

ADI Data

Exemple :

MVI A, 10H ; Charge 10H dans l'accumulateur.

ADI 05H ; Additionne 05H à l'accumulateur.

1.2 Instructions d'addition Avec retenus ADC (With Carry)

L'instruction ADC ajoute le contenu d'un registre ou d'une mémoire, plus la retenue (flag Carry) au contenu de l'accumulateur.

*** ADC :** Additionne le contenu d'un registre ou d'une adresse mémoire ainsi que la retenue CY (flag Carry) à l'accumulateur.

Syntaxe :

ADC R (ou) ADC M

R : Registre (B, C, D, E, H, L).

M : Contenu de l'adresse mémoire pointée par le registre HL.

Exemple : MVI A, 57H ; Charge 57H dans le contenu de l'accumulateur A.

ADD C ; Additionne le contenu du registre C à l'accumulateur ainsi que le bit Cy.

*** ACI :**

Additionne une donnée immédiate à l'accumulateur ainsi que le bit Cy

Syntaxe :

ACI Data

Exemple :

MVI A, 18H ; Charge 18H dans l'accumulateur.

ACI 05H ; Additionne 05H et Cy à l'accumulateur.

2. **instructions de soustraction** Les instructions de soustraction permettent de retirer le contenu d'un registre, d'une mémoire ou d'une donnée immédiate du contenu de l'accumulateur.

   2.1 Instructions de soustraction sans prêt SUB (Without Borrow)

3.5.3.2 Instructions Logiques

Les instructions logiques permettent de manipuler les bits des données stockées dans les registres ou la mémoire. Ces instructions incluent :

1. **Opérations AND, OR, et XOR**

   Syntaxe :

   ANA : Effectue un ET logique entre l'accumulateur et un registre ou une mémoire.

   ORA : Effectue un OU logique.

   XRA : Effectue un OU exclusif (XOR).

   Exemple :

   MVi A, F0H ; Charge F0H dans l'accumulateur.

   ANA C ; Effectue un ET logique avec le registre C.

2. **Comparaison (CMP)**

   Compare le contenu de l'accumulateur avec celui d'un registre, d'une mémoire ou d'une donnée.

   Syntaxe :

   CMP R : Compare le contenu de l'accumulateur avec l'un des registres (B,C,D,E,H,L)

   CMP M : Compare le contenu de l'accumulateur avec un contenu Mémoire

   CPI data : Compare le contenu de l'accumulateur avec une donnee

Exemple :

MOV A, 05H ; Charge 05H dans l'accumulateur.

CMP B ; Compare le contenu de B avec l'accumulateur.

3. **Décalage et Complément :** CMA : Complément de l'accumulateur. RAL/RAR : Rotation des bits vers la gauche/droite.

3.5.3.3 Instructions de Branchement et d'Appel

Dans le microprocesseur 8085, les instructions de branchement et d'appel jouent un rôle essentiel pour contrôler le flux d'exécution d'un programme. Elles permettent de modifier la séquence normale d'exécution, soit en sautant à une autre partie du programme, soit en appelant une sous-routine.

1. **Instructions de Branchement** Les instructions de branchement (Jump) redirigent directement l'exécution vers une nouvelle adresse mémoire. Elles peuvent être utilisées pour implémenter des structures conditionnelles, des boucles ou des séquences spécifiques en fonction de l'état des drapeaux (flags).

   Branchement Inconditionnel (JMP)

   Syntaxe :

   JMP Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction

   Exemple :

   JMP 2000H

   Branchements conditionnel

   Les instructions conditionnelles effectuent un saut en fonction de l'état des drapeaux (flags).

   Branchement en fonction du Flag Carry (Cy)

   JC (Jump if Carry) : Saute si le drapeau de retenue (Carry flag) est défini.

   Syntaxe :

   JC Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction

si CY=1

Exemple :

JC 5000H

JNC (Jump if No Carry) : Saute si le drapeau de retenue (Carry flag) n'est pas défini.

Syntaxe :

JNC Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si CY=0

Exemple :

JNC 6000H

1.2.2 Branchement en fonction du Flag Zero (Z)

JZ (Jump if Zero) : Un Saut si le drapeau de Zero (Zero flag) est défini.

Syntaxe :

JZ Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si Z=1

Exemple :

JZ 5000H

JNZ (Jump if No zero) : un Saut si le drapeau de Zero (Zero flag) n'est pas défini.

Syntaxe :

JNZ Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si Z=0

Exemple :

JNZ 6000H

1.2.3 Branchement en fonction du Flag Signe (S)

JP (Jump if Positif) : un Saut si le résultat est positif, drapeau de Signe (signe flag) S=0.

Syntaxe :

JP Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si S=0

Exemple :

JP 5000H

JM (Jump if Negatif) : Saut si le résultat est Négatif, drapeau de Signe (signe flag) S=1.

Syntaxe :

JM Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si S=1

Exemple :

JM 5000H

1.2.3 Branchement en fonction du Flag parité (P)

JPE (Jump if Parity even) : Saute si le résultat est paire, drapeau de parité (parity flag) P=0.

Syntaxe :

JPE Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si P=0

Exemple :

JPE 5000H

JPO (Jump if Parity odd) : Un Saut si le résultat est impaire, drapeau de parité (parity flag) P=1.

Syntaxe :

JPO Addr : Un Saut vers Addr de 16 bits spécifiée dans l'instruction si P=1

Exemple :

JPO 5000H

2. **Instructions d'Appel**

   Les instructions d'appel (CALL) permettent de transférer l'exécution

vers une sous-routine, une partie du programme autonome exécutant une tâche spécifique. Contrairement au branchement, les instructions d'appel sauvegardent l'adresse de retour pour permettre de revenir au programme principal une fois la sous-routine exécutée.

2.1 Appel Inconditionnel (CALL)

Appelle une sous-routine et sauvegarde l'adresse de retour sur la pile.

Syntaxe :

CALL Addr : Appel de la sous routine pointée par Addr de 16 bits spécifiée dans l'instruction

Exemple :

CALL 2600H

2.2 Appel conditionnel

Les instructions CALL conditionnelles dans le microprocesseur 8085 permettent d'appeler une sous-routine uniquement si une condition spécifique, déterminée par l'état des drapeaux (flags), est satisfaite. Ces instructions combinent la puissance des sous-routines et des tests conditionnels, rendant le flux d'exécution encore plus flexible et efficace. Par exemple, si un résultat de calcul indique une condition particulière (comme un dépassement, une parité paire ou une valeur nulle), une sous-routine pertinente peut être exécutée. Si la condition n'est pas remplie, l'exécution du programme se poursuit normalement sans appeler la sous-routine. Cela permet d'économiser des ressources et d'exécuter des actions spécifiques uniquement lorsque cela est nécessaire, améliorant ainsi l'efficacité globale des programmes.

2.2.1 CALL en fonction du Flag Carry (Cy)

CC (CALL if Carry) : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si CY=1.

Syntaxe :

CC Addr : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si CY=1

Exemple :

CC 5000H

CNC (Jump if No Carry) : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si CY=0.

Syntaxe :

CNC Addr : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si CY=0.

Exemple :

CNC 6000H

2.2.2 CALL en fonction du Flag Zero (Z)

CZ (Jump if Zero) : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si Z=1.

Syntaxe :

CZ Addr :Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si Z=1.

Exemple :

CZ 5000H

CNZ (CALL if No zero) : Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si Z=0.

Syntaxe :

CNZ Addr :Appel de la sous routine de L'Addr de 16 bits spécifiée dans l'instruction si Z=0

Exemple :

CNZ 6000H

2.2.3 CALL en fonction du Flag Signe (S)

CP (CALL if Positif) : Appel de la sous routine si le résultat est positif, drapeau de Signe (signe flag) S=0.

Syntaxe :

CP Addr : Appel de la sous routine si le résultat est positif, drapeau de Signe (signe flag) S=0.

Exemple :

CP 5000H

CM (CALL if Negatif) : Appel de la sous routine si le résultat est positif, drapeau de Signe (signe flag) S=1.

Syntaxe :

CM Addr : Appel de la sous routine si le résultat est positif, drapeau de Signe (signe flag) S=1.

Exemple :

CM 5000H

2.2.3 CALL en fonction du Flag parité (P)

CPE (CALL if Parity even) : CALL la sous routine si le résultat est paire, drapeau de parité (parity flag) P=0.

Syntaxe :

CPE Addr : Un CALL vers Addr de 16 bits spécifiée dans l'instruction si P=0

Exemple :

CPE 5000H

CPO (Jump if Parity odd) : CALL la sous routine si le résultat est impaire, drapeau de parité (parity flag) P=1.

Syntaxe :

CPO Addr : Un CALL vers Addr de 16 bits spécifiée dans l'instruction si P=1

Exemple :

CPO 5000H

3. **Instructions de Retour**

Les instructions d'appel (CALL) permettent de transférer l'exécution vers une sous-routine, une partie du programme autonome exécutant une tâche spécifique. Contrairement au branchement, les instructions d'appel sauvegardent l'adresse de retour pour permettre de revenir au programme principal une fois la sous-routine exécutée.

3.1 Retour inconditionnel (RET)

L'instruction RET (Return) inconditionnelle permet de revenir à l'adresse sauvegardée sur la pile, qui correspond à l'instruction située juste après l'appel de sous-routine (CALL). Cette instruction est utilisée lorsque la sous-routine a terminé son exécution, et aucun test ou condition n'est requis pour retourner au programme principal.

Syntaxe :

RET : Revient à l'adresse sauvegardée sur la pile.

3.2 Retour conditionnel

Les instructions RET conditionnelles permettent de revenir au programme principal uniquement si une condition spécifique, déterminée par l'état des drapeaux, est remplie. Si la condition n'est pas satisfaite, l'exécution continue dans la sous-routine. Ces instructions offrent une flexibilité supplémentaire pour gérer des scénarios où le retour dépend de l'état des calculs effectués dans la sous-routine.

3.2.1 Retour en fonction du Flag Carry (Cy)

RC (Return if Carry) : Retour si le drapeau Carry est défini Cy=1.

Syntaxe :

RC : Retour si le drapeau Carry est défini.

RNC (return if No Carry) : Retour si le drapeau Carry n'est pas défini Cy=0.

Syntaxe :

RNC : Retour si le drapeau Carry n'est pas défini Cy=0.

3.2.2 Retour en fonction du Flag Zero (Z)

RZ (Return if Zero) : Retour si le drapeau Zero est défini si Z=1.

Syntaxe :

RZ :Retourne si le drapeau Carry est défini Z=1.

RNZ (Return if No Zero) : Retour si le drapeau Zero est définisi Z=0.

Syntaxe :

RNZ :Retourne si le drapeau Zero est défini si Z=0.

3.2.3 Retour en fonction du Flag Signe (S)

RP (Return if Positif) : Retourne si le drapeau Signe n'est pas défini , S=0

Syntaxe :

RP : Retour si le drapeau Signe n'est pas défini si (signe flag) S=0.

RM (Return if Negatif) : Retour si le drapeau Signe est défini (signe flag) S=1.

Syntaxe :

CM Addr : Retour si le drapeau Signe est défini (signe flag) S=1.

2.2.3 Return en fonction du Flag parité (P)

RPE (Return if Parity even) : Retour si le drapeau parité est paire (Parity flag) P=0.

Syntaxe :

RPE :Retour si le drapeau parité est paire si (Parity flag) P=0.

CPO (Return if Parity odd) : Retour si le drapeau parité est impaire (Parity flag) P=1.

Syntaxe :

CPO : Retour si le drapeau parité est impaire (Parity flag) P=1.

4. **Instruction de contrôle** Ces instructions contrôlent certaines fonctions spécifiques du processeur, telles que l'arrêt, la gestion des interruptions ou l'absence d'action. Elles modifient les différentes opérations exécutées par le processeur.

NOP (No Operation) : Aucune opération n'est effectuée. L'instruction est simplement chargée et décodée, mais aucune action n'est exécutée.

HLT (Halt) : Le processeur termine l'exécution de l'instruction en cours et suspend toute exécution ultérieure. Pour sortir de cet état d'arrêt, une interruption ou un signal de réinitialisation est nécessaire.

RIM (Read Interrupt Mask) : Cette instruction polyvalente est utilisée pour lire l'état des interruptions 7.5, 6.5 et 5.5, ainsi que pour lire le bit d'entrée des données série. Elle charge huit bits dans l'accumulateur avec des interprétations spécifiques.

SIM (Set Interrupt Mask) : Instruction polyvalente utilisée pour gérer les interruptions 7.5, 6.5 et 5.5 du 8085, ainsi que pour envoyer des données en série. Elle interprète le contenu de l'accumulateur selon un format défini.

3.6 Conclusion

Les instructions de branchement (JUMP), d'appel (CALL) et de retour (RET), qu'elles soient inconditionnelles ou conditionnelles, constituent des outils fondamentaux pour contrôler le flux d'exécution des programmes sur le microprocesseur 8085. Les branchements permettent de modifier directement la séquence d'exécution en sautant à une nouvelle adresse, tandis que les appels de sous-routines offrent une structure modulaire pour organiser les programmes en tâches spécifiques. Les retours, quant à eux, garantissent un retour fluide au programme principal après l'exécution des sous-routines. Dans leurs versions conditionnelles, ces instructions introduisent un niveau supplémentaire de flexibilité en exécutant des actions uniquement lorsque certaines conditions, basées sur l'état des drapeaux, sont satisfaites. Ensemble, ces instructions permettent de développer des programmes plus dynamiques,

optimisés et faciles à maintenir, répondant ainsi aux exigences variées des applications complexes.