الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

Université Ibn Khaldoun de Tiaret
Faculté des sciences Appliquées
Département de Génie Electrique

Travaux Pratiques

µ-processeurs et µ-contrôleurs

Avec rappels de cours, corrigés et programmes types

Formation : Master Electrotechnique

Code Unité : UEM 1.1

Matière : TP - µ-processeurs et µ-contrôleurs

Par : Dr. TAHRI Ahmed

Année : 2022/2023

Partie I Système à µ-processeur 8086

- Présentation générale d'un système à µ-processeur 8086
- TP1 : Prise en main d'un environnement de programmation sur µ-processeur **(01 semaine)**
- TP2 : Programmation des opérations arithmétiques et logiques dans un µ-processeur **(01 semaine)**
- TP3 : Utilisation de la mémoire vidéo dans un µ-processeur. **(01 semaine)**
- TP4 : Gestion de la mémoire du µ-processeur. **(02 semaines)**
- TP5 : Commande d'un moteur pas à pas par un µ-processeur **(02 semaines)**

Présentation générale d'un système à µ-processeur 8086

1. Présentation générale du 8086

L'Intel 8086 est un microprocesseur a 16 bits fabriqué par Intel à partir de 1978. C'est le premier processeur de la famille x86, qui est devenue l'architecture de processeur la plus répandue dans le monde des ordinateurs personnels, stations de travail et serveurs informatiques en raison du choix d'IBM de l'utiliser comme base de l'IBM PC sorti quelques années après.

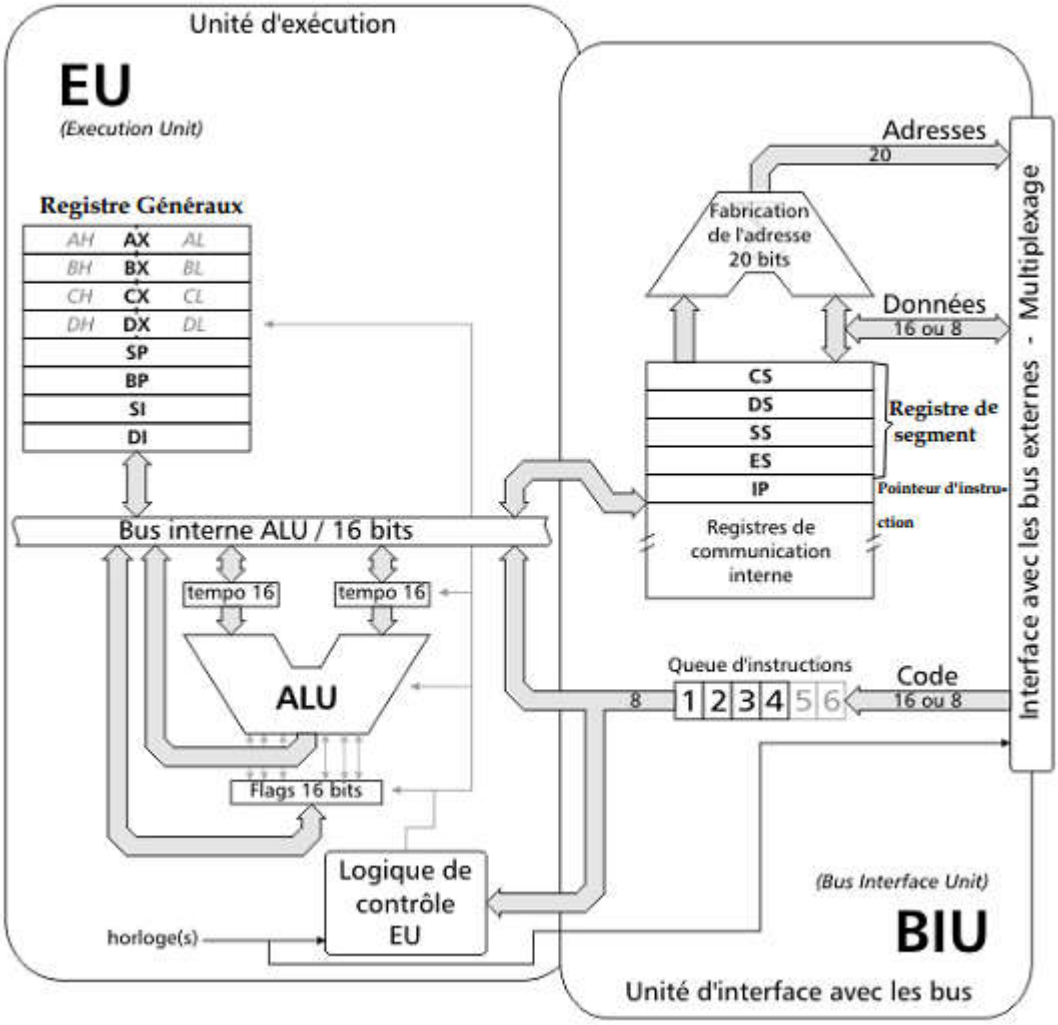

Figure 1. Registres du processeur 8086

1.1. Les registres :

La programmation en langage machine nécessite la connaissance au préalable des registres internes du processeur utilisé. En effet, en langage assembleur chaque instruction fait référence à une action élémentaire effectuée par le processeur. Il est mentionné pour chaque action qu'une des données manipulées doit obligatoirement se trouver au sein d'un registre du processeur. Nous présentons les différents registres du 8086 chacun dans sa catégorie. Les différents registres sont affichés la figure 1.

Les registres du 8086 sont séparé en cinq catégories, nommés *registres général, registres de segment* ***(segment registers)****, registres de pointer (****Pointer registers****), registres de la pile (****stack registers****) , et les registres des états(****status registers****)*, comme illustre la figure 1-3.

Les registres étant situés à l'intérieur du processeur (CPU), ils sont beaucoup plus rapides que la mémoire. L'accès à un registre ne prend généralement pas de temps. Par conséquent, vous devriez essayer de conserver les variables dans les registres.

Ces registres sont facilement accessibles aux programmeurs, et chaque registre a une fonction spéciale qui doit être clairement comprise si vous voulez écrire en langage assembleur.

1.1.1. Les registres généraux

Quatre registres à usage général (AX, BX, CX, DX) ont une largeur de 16 bits mais sont accessibles sous forme d'octet ou de mot. Ces registres de données sont normalement utilisés pour stocker des résultats temporaires qui seront exploités par des instructions ultérieures.

Accumulateur (AX) : le registre se compose de 2 registres à 8 bits AL et AH qui peuvent être combinés ensemble et utilisés comme registre 16 bits AX. AL dans ce cas contient l'octet de poids faible du mot et AH l'octet de poids fort. L'accumulateur peut être utilisé pour les opérations d'E / S et la manipulation de chaînes de caractère.

Registre de base (BX) le registre se compose de 2 registres à 8 bits BL et BH qui peuvent être combinés ensemble et utilisés comme registre 16 bits BX. BL dans ce cas contient l'octet de poids faible du mot et BH l'octet de poids fort. Le registre BX contient généralement un pointeur de données utilisé pour l'adressage indirect basé, indexé basé ou registre.

Registre de comptage CX (*count register*) : le registre se compose de 2 registres à 8 bits CL et CH qui peuvent être combinés ensemble et utilisés comme registre 16 bits CX. CL dans ce cas contient l'octet de poids faible du mot et CH l'octet de poids fort. Le registre CX peut être utilisé comme un compteur dans la manipulation de boucle et les instructions de décalage / rotation.

Registres généraux		
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

Registres de pointer
SI
DI
IP

Registres de segment
CS
DS
SS
ES

Registres d'état (status registers)	
FlagH	**Flag**L

Registres de la pile (Stack registers)
SP
BP

Figure 1-1 : les registres de 8086

Registre de donnée DX (*DATA register*) le registre se compose de 2 registres à 8 bits DL et DH qui peuvent être combinés ensemble et utilisés comme registre 16 bits DX. DL dans ce cas contient l'octet de poids faible du mot et DH l'octet de poids fort. Le registre DX peut être utilisé comme numéro de port dans les opérations d'E / S. Dans l'instruction de multiplication et de division d'entiers 32 bits, le registre DX contient un mot d'ordre élevé du nombre initial ou résultant.

1.1.2. Les registres de segment :

La plupart des registres contiennent des décalages(offsets) de données / instructions dans un segment de mémoire de 64 Ko (216). Il existe quatre segments différents de 64 Ko pour les instructions, la pile, les données et les données supplémentaires. Pour spécifier où dans 1 Mo de mémoire du processeur ces 4 segments sont situés, le processeur utilise quatre registres de segments.

Le segment de code (CS) est un registre de 16 bits contenant une adresse de segment de 64 Ko avec des instructions de processeur. Le processeur utilise CS pour tous les accès aux instructions référencées par le registre du pointeur d'instruction (IP). Le registre CS ne peut pas être modifié directement. Le registre CS est automatiquement mis à jour pendant les instructions de saut éloigné, d'appel éloigné et de retour éloigné.

Le segment de données (DS) est un registre de 16 bits contenant l'adresse d'un segment de 64 Ko avec des données de programme. Par défaut, le processeur suppose que toutes les données référencées par les

registres généraux (AX, BX, CX, DX) et le registre d'index (SI, DI) se trouvent dans **le segment de données**. Le registre DS peut être modifié directement à l'aide de l'instruction POP.

Le segment de pile ou stack (SS) est un registre de 16 bits contenant l'adresse d'un segment de 64 Ko avec une pile de programmes. Par défaut, le processeur suppose que toutes les données référencées par les registres du pointeur de pile (SP) et du pointeur de base (BP) se trouvent dans le registre du segment de pile et peuvent être modifiées directement à l'aide des instructions POP et LDS.

Extra Segment (ES) est un registre de 16 bits contenant une adresse de segment de 64 Ko, généralement avec des données de programme. Par défaut, le processeur suppose que le registre d'index de destination (DI) fait référence au segment ES dans les instructions de manipulation de chaîne. Le registre ES peut être modifié directement à l'aide des instructions POP et LES.

Encore une fois, notez que les quatre segments n'ont pas besoin d'être définis séparément. Les quatre segments peuvent se chevaucher complètement (CS = DS = SS = ES).

1.1.3. Registres de pointeurs

Les registres de cette catégorie ont tous une largeur de 16 bits et ne sont pas accessibles en tant qu'octet bas ou haut. Ces registres sont utilisés comme pointeurs de mémoire.

L'index de source (SI) est un registre de 16 bits. **SI** est utilisé pour l'adressage indirect indexé, indexé basé et de registre, ainsi qu'une adresse de données source dans les instructions de manipulation de chaînes de caractères,

L'index de destination (DI) est un registre 16 bits. DI est utilisé pour l'adressage indirect indexé, indexé basé et registre, ainsi qu'une adresse de données de destination dans les instructions de manipulation de chaîne de caractères.

Le pointeur d'instruction (IP) est un registre 16 bits. Le registre IP fonctionne toujours avec le registre CS et pointe vers le décalage (l'offset) ou l'instruction suivante.

Registres de pile

La pile est une zone de mémoire spéciale conçue pour le stockage rapide et temporaire des données du programme et les adresses de retour des sous-programmes.

Le pointeur de pile (SP) est un registre de 16 bits contenant un décalage (offset) dans le segment de pile (SS). La pile se développe vers le bas (vers les décalages de segment inférieur) à partir du pointeur de pile.

Le pointeur de base (BP) contient un décalage (offset) dans le segment de pile (SS) qui peut être utilisé comme pointeur vers une liste de paramètres.

1.1.4. Le registre d'état (Status)

15				**FlagsH**			**8**	**7**			**FlagsL**				**0**
				OF	**DF**	**IF**	**TF**	**SF**	**ZF**		**AF**		**PF**		**CF**

Indicateur de débordement (*overflow flag OF*) – **SET** (mise à 1) si le résultat arithmétique est un nombre positif trop grand ou un nombre négatif trop petit pour tenir dans l'opérande de destination.

• **Drapeau de direction (DF)** - si SET, les instructions de manipulation de chaîne décrémenteront automatiquement les registres d'index. Si EFFACÉ (mise à 0), les registres d'index seront auto-incrémentés.

• **Indicateur d'activation d'interruption (IF)** - lorsque SET ce bit, les interruptions masquables amèneront la CPU à transférer le contrôle vers un emplacement spécifié par le vecteur d'interruption.

Drapeau de piège (Trape) (TF) - si SET, une interruption en mode pas à pas se produira après l'instruction suivante. TF est EFFACÉ par l'interruption à pas à pas.

Drapeau de Signe (SF) - SET si le bit le plus significatif du résultat est négatif. EFFACÉ si le bit le plus significatif du résultat est positif.

Drapeau Zéro (ZF) - SET si le résultat opéré est zéro.

Drapeau de retenu auxiliaire (AF) - SET s'il y avait retenu ou emprunt aux bits 0-3 dans le registre AL, EFFACE autrement.

Drapeau de parité (PF) - SET si 8 bits de poids faible du résultat contiennent un nombre pair de 1 bit, EFFACE dans le cas contraire.

Drapeau de Retenu (carry) (CF) - SET s'il y a eu report ou emprunt au bit le plus significatif lors du dernier calcul de résultat.

2. Méthodes de programmation

2.1. Etapes de la réalisation d'un programme :

• Définir le problème à résoudre : que faut-il faire exactement ?

Déterminer des algorithmes, des organigrammes : comment faire ? Par quoi commencer ?

• Rédiger le programme (code source) :

– utilisation du jeu d'instructions (mnémoniques) ;

– création de documents explicatifs (documentation).

• Tester le programme en réel ;

Corriger les erreurs (bugs) ´éventuelles : déboguer le programme puis refaire des fonctionnant de manière satisfaisante.

2.2. Langage machine et assembleur :

- Langage machine : codes binaires correspondant aux instructions ;
- Assembleur : logiciel de traduction du code source écrit en langage assembleur (mnémoniques).

Réalisation pratique d'un programme :

Rédaction du code source en assembleur à l'aide d'un éditeur (logiciel de traitement ASCII) :

– édit sous MS-DOS,

– notepad (bloc-note) sous Windows,

Assemblage du code source (traduction des instructions en codes binaires) avec un assembleur :

– MASM de Microsoft,

– TASM de Borland,

– A86 disponible en shareware sur Internet, ...

Pour obtenir le code objet : code machine exécutable par le microprocesseur ;

Chargement en mémoire centrale et exécution : rôle du système d'exploitation.

Pour la mise au point (débogage) du programme, on peut utiliser un programme d'aide à la mise au point (comme DEBUG sous MS-DOS) permettant :

• l'exécution pas à pas ;

• la visualisation du contenu des registres et de la mémoire ;

• la pose de points d'arrêt ...

2.3. L'assemblage :

L'assembleur pour le microprocesseur peut être utilisé de deux manières : (1) avec des modèles qui sont uniques à un assembleur particulier, et (2) avec des définitions de segment complet qui permettent contrôle sur le processus d'assemblage et sont universels pour tous les assembleurs. Cette section présente les deux méthodes et explique comment organiser l'espace mémoire d'un programme à l'aide de l'assembleur. Il explique également le but et l'utilisation de certaines des directives les plus importantes utilisées avec cet assembleur.

2.3.1. Structure de Programme avec modèle :

==========

.model small ; ou /tiny/medium/large

.Stack <nombre>

.data

; Initialisation des données utilisées dans le programme.

; La déclaration de variable est ici.

.code

; Initialisation du segment de données,

; La logique du programme est ici.

End

2.3.2. Structure basé sur la définition segment complet :

==========

DATA_SEG SEGMENT 'DATA'

; Début de définition du segment DATA_SEG

DATA_SEG ENDS

Code SEGMENT 'CODE'

; Début de définition du segment DATA_SEG

ASSUME CS: CODE_SEG ,DS: DATA_SEG ;

ORG 100H

; Instructions

CODE_SEG ENDS ; Fin du segment CODE_SEG

END ; fin de programme

==========

Remarques :

1-Tous ce qui est après la point-virgule " ;" est considéré comme commentaire (n'est pas pris en compte par l'assembleur)