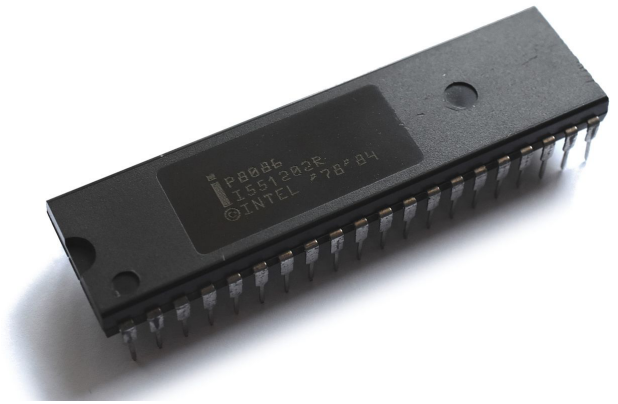
Architecture du 8086

Introduction

- Chaque type de processeur possède un jeu d'instructions spécifiques
- Ces jeux d'instructions ne sont bien souvent pas compatibles entre eux
- On peut distinguer de grands types de processeurs et de jeux d'instructions associés:
  - Les processeurs CISC (Complex Instruction Set Computer) qui ont un jeu d'instructions étendu. Les processeurs grand public font partie de cette famille
  - Les processeurs RISC (Reduced Instruction Set Computer) qui disposent d'un jeu d'instructions réduite. Ces instructions sont limitées mais rapides d'exécution

Le processeur 8086

- Il s'agit d'un des premiers processeurs grand public (lancé en 1978)
- C'est le premier processeur de la famille des processeurs Intel allant du 8086 jusqu'au Pentium
- Il contient 29000 transistors et est cadencé à une fréquence allant de 4,77 MHz à 10 MHz
- Particularités du 8086:
  - Bus d'adresse de 20 bits (5 symboles hexa) soit 1 Mo adressable
  - Bus de données de 16 bits (4 symboles hexa)
  - Des registres 16 bits

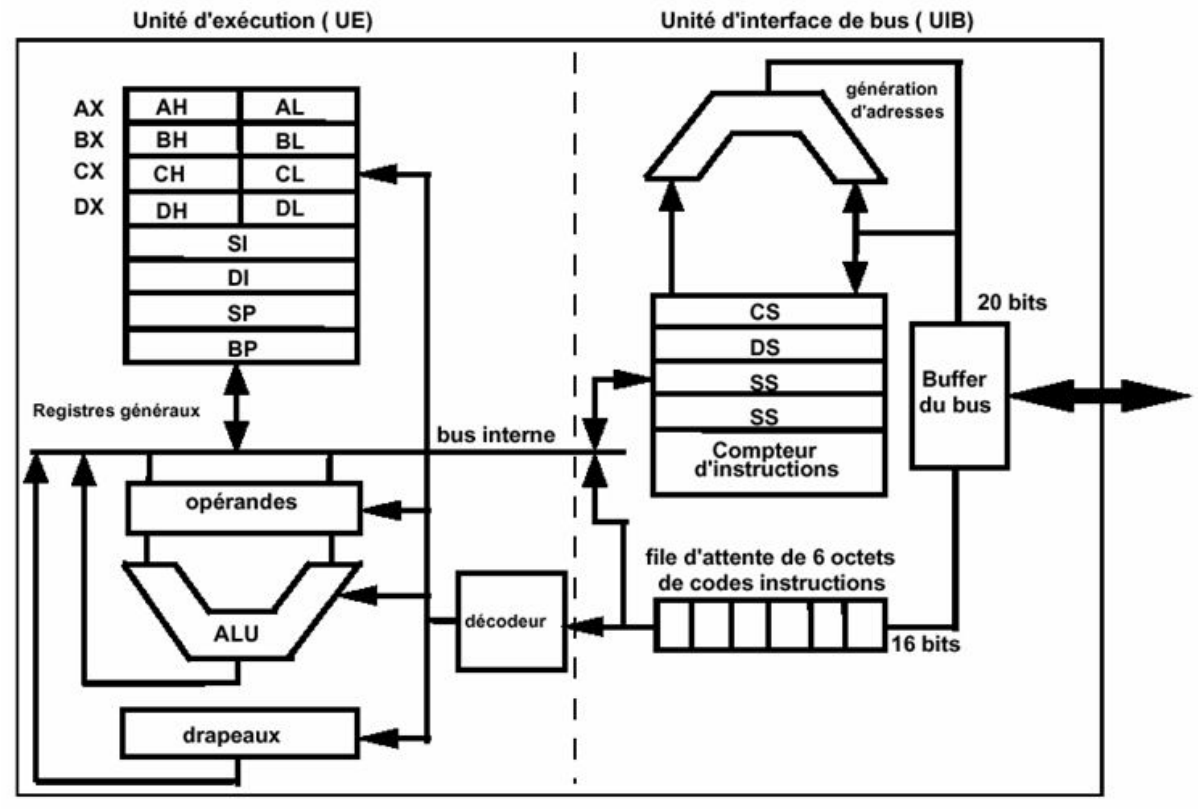


Le processeur 8086

- Les registres, en particulier le registre d'adresse n'étant que de 16 bits, on ne peut adresser en théorie que 2^{16} adresses différentes parmi les 2^{20} possibles
- Un **mécanisme d'adressage particulier** est mis en place pour adresser les 2^{20} adresses possibles: l'adresse est décomposée et stockée en deux parties dans deux registres
- Les données sont stockées en mémoire en **mode little endian**

Architecture interne du 8086

- 2 unités internes: l'unité d'exécution (UE) et l'unité d'interface avec le bus (UIB)



- Le 8086 a introduit la notion de traitement pipeline: chargement des instructions suivantes dans une file d'attente pendant qu'une instruction est traitée par l'UE

Organisation de la mémoire: les paragraphes

- Les adresses mémoire dont le dernier chiffre hexadécimal (4 bits de poids faible) est 0h divisent la mémoire en paragraphes
- Un paragraphe est donc un ensemble de 16 adresses consécutives dont la première est un multiple de 16
- Les paragraphes sont repérés par les chiffres hexadécimaux de poids forts (16 premiers bits) d'une adresse

Segmentation de la mémoire du 8086

- Le bus d'adresse de 20 bits (5 symboles hexa) permet d'adresser 2^{20} adresses physiques mais les registres ne peuvent stocker que des adresses sur 16 bits

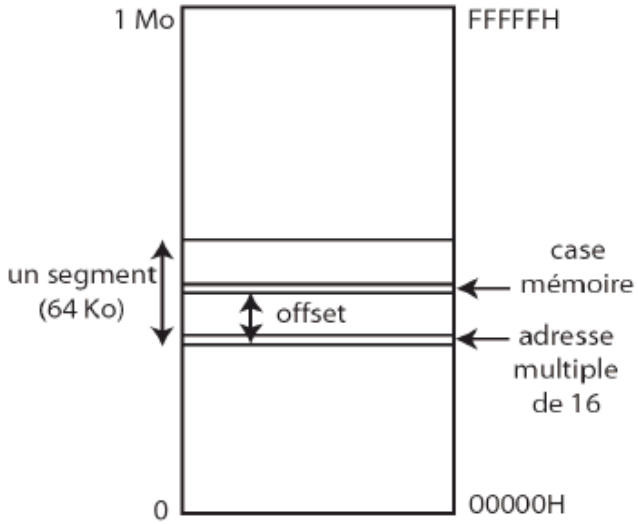
- Chaque adresse physique codée sur 5 symboles hexa est décomposée en un couple de valeurs:

  numéro de segment / déplacement dans le segment (offset)

- Ce couple est désigné sous le terme **adresse logique**

Segmentation de la mémoire du 8086

- Un segment est un ensemble de 64 Ko consécutifs, dont le premier élément coïncide avec le début d'un paragraphe (adresse physique se terminant par 0)
- Le déplacement (offset) spécifie un octet particulier dans un segment



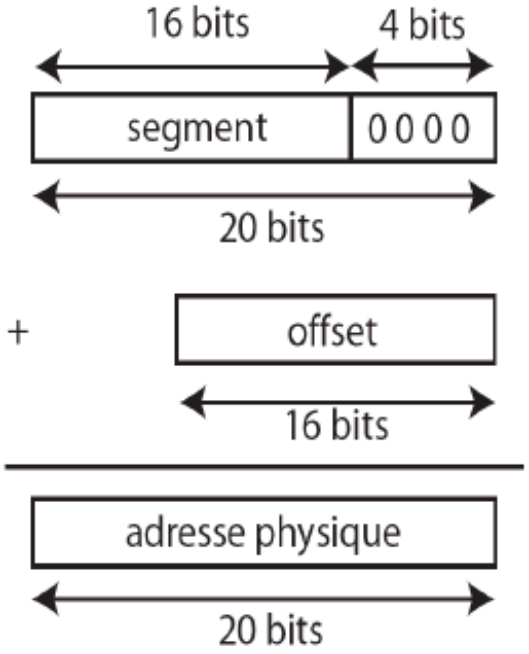
Segmentation de la mémoire du 8086

- Les valeurs de **numéro de segment** et **déplacement** sont codées sur 16 bits (valeur comprise entre 0 et 65535)
- La relation entre adresse physique et adresse logique (couple segment/déplacement) est:

  adresse physique = 16 x segment + déplacement

- A une même adresse physique peut correspondre plusieurs couples segment/déplacement (à chaque adresse physique correspond 2^{12} = 4096 adresses logiques différentes)

Segmentation de la mémoire du 8086

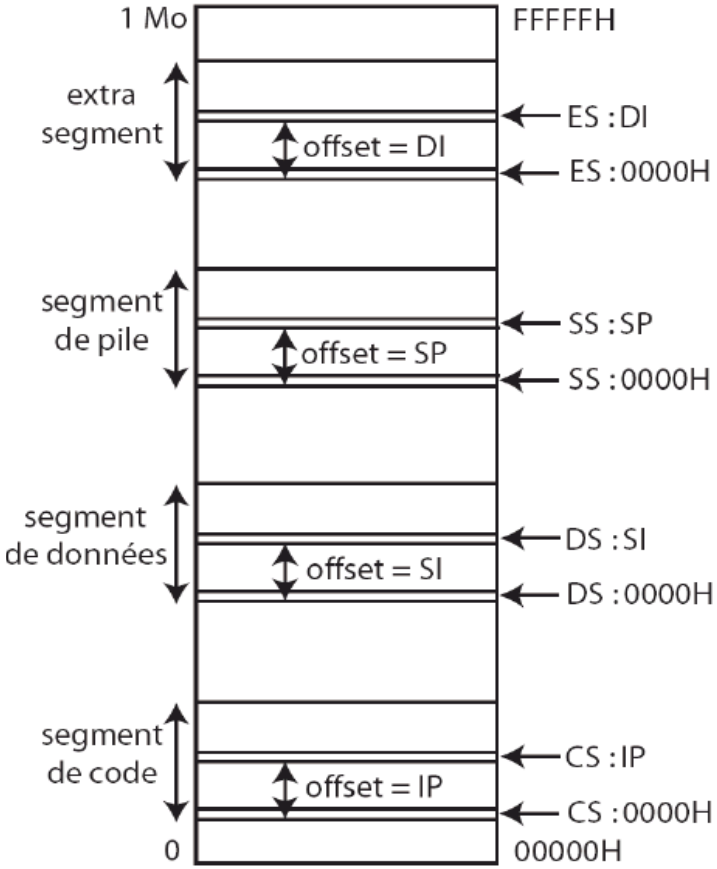


CS:IP = (4055:3192)
= 40550+3192
= 436E2

Segmentation de la mémoire du 8086

- Lors de son exécution un programme utilise plusieurs segments mémoire :
  - le **segment de code** qui contient les instructions du programme
  - le **segment de données** qui contient les données du programme
  - le **segment de pile** qui contient la pile du programme
- Ces trois segments peuvent se situer n'importe où en mémoire et même se recouvrir partiellement

Segmentation de la mémoire du 8086



Les registres du 8086

- Les registres du 8086 sont des registre 16 bits (2 octets)
- Le 8086 possède des registres permettant de contenir des numéros de segments:
  - **CS** contient le numéro de segment où est stocké le ***programme*** en cours d'exécution
  - **DS** contient le numéro de segment où sont stocké les ***données***
  - **SS** contient le numéro de segment où est stocké la ***pile***
  - **ES** contient un numéro de ***segment auxiliaire*** permettant d'adresser des données.
- Les registres CS et SS sont initialisés par la machine lors de l'exécution d'un programme exécutable
- Le registre DS doit être initialisé dans le programme

Les registres du 8086

- Le 8086 possède également plusieurs registres généraux:
  - Les registres **AX**, **BX**, **CX**, **DX** sont des registres généraux servant à stocker des données. Chacun de ces registres se décompose en deux registres de 8 bits, représentant la partie haute (AH, BH, CH, DH) et basse (AL, BL, CL, DL) de ces registres
  - **SI** et **DI** contiennent le déplacement dans le segment de données et sont généralement utilisés pour le transfert d'un ensemble d'octets dans la mémoire
  - SP et BP permettent la gestion de la pile. Ils sont associés par défaut à SS
    - **SP** est le registre de pointeur de pile qui contient le déplacement dans le segment de pile de l'adresse de la case mémoire courante de la pile
    - **BP** d'adresser une case mémoire particulière de la pile

Les registres du 8086

- Deux registres ne sont pas accessibles par une instruction:
  - Le registre d'état FLAG contenant les différents drapeaux spécifiant l'état de l'UAL
  - Le registre IP (compteur ordinal) spécifiant le déplacement dans le segment de code de l'adresse mémoire de la prochaine instruction à exécuter.
- L'adresse mémoire spécifiée par CS:IP pointe sur la case-mémoire contenant la prochaine instruction à exécuter.

Les registres du 8086

Le registre d'état (flag)

- Il s'agit d'un registre 16 bits
- Les bits du registre d'état (drapeaux) sont des indicateurs qui annoncent une condition particulière suite à une opération arithmétique ou logique
- Les instructions de branchements conditionnels utilisent ces indicateurs du registre d'état
- Le registre d'état est formé des bits suivants, dont certains ne sont pas utilisés:

15															0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

Les registres du 8086

Le registre d'état (flag)

- Bit CF (Carry Flag): positionné à 1 lorsque qu'une retenue est générée par une opération arithmétique sur des entiers naturels (entiers non signés) sur 8 ou 16 bits
- Bit PF (Parity Flag): positionné à 1 lorsque le résultat de l'opération effectuée contient un nombre pair de bits à 1
- Bit AF (Auxiliary Flag): positionné à 1 lorsqu'une retenue est générée en sortie du quarter de poids faible (4 bits de poids faible) sur le quarter de poids fort (4 bits de poids fort). Ce que l'on appelle une demi-retenue
- Bit ZF (Zero Flag): positionné à 1 lorsque le résultat de l'opération effectuée vaut 0

Les registres du 8086

Le registre d'état (flag)

- Bit SF (Sign Flag): positionné à 1 lorsque le bit de poids fort du résultat de l'opération effectuée vaut 1 (indique donc le signe du résultat lorsque l'on effectue une opérations sur des entiers signés)
- Bit OF (Overflow Flag): positionné à 1 en cas de débordement arithmétique, c.a.d si le résultat de l'opération arithmétique effectuées sur des nombres entiers signés excède la capacité de l'opérande
- Bit DF (Direction Flag): indicateur de direction utilisé par les instructions de manipulation de chaînes de caractères pour auto incrémenter (ou décrémenter) la valeur des registres index SI et DI
- Bit IF (Interrupt Flag): positionné à 1 pour autoriser la prise en compte des interruptions externes
- Bit TF (Trap Flag): positionné à 1 pour autoriser l'exécution en mode pas à pas du programme

Modes d'adressage du 8086

- Mode d'adressage = manière de spécifier l'adresse de la case mémoire à laquelle on souhaite accéder
- Dans un 8086 il y a plusieurs manières d'indiquer cet emplacement:
  - simple indication d'un offset
  - exploitation du contenu d'un registre
  - calcul de l'adresse en additionnant le contenu de plusieurs registres à une valeur constante

Modes d'adressage du 8086

- Adressage immédiat
  - le code opérande contient directement une donnée
  - met en jeu un registre et une valeur

    exemple: `MOV AX, 8`

- Adressage registre
  - le code opérande est un autre registre
  - Le contenu du deuxième registre est transférée dans celui du premier

    exemple: `MOV AX, BX`

Modes d'adressage du 8086

- Adressage direct
  - le code opérande contient l'adresse d'une donnée en mémoire
  - Ce mode d'adressage provoque un temps d'exécution de l'instruction plus long car l'accès à la mémoire principale est plus long que l'accès à un registre

exemples:

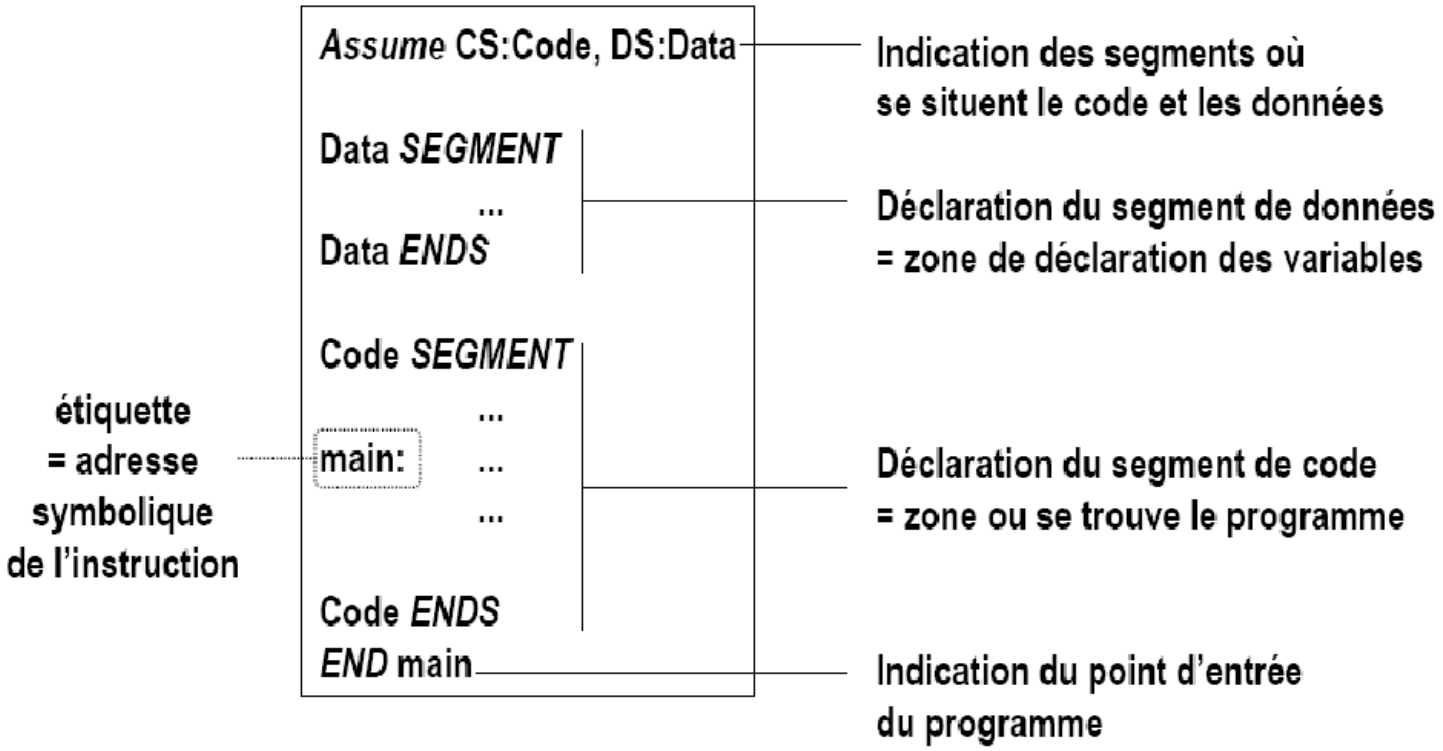
```
MOV AL, DS:[0008]
MOV AX, DS :1999
MOV AX, Variable ; (variable étant une étiquette)
```

Modes d'adressage du 8086

- Adressage indirect
  - permet d'accéder à une donnée par l'intermédiaire d'un registre (BX) qui contient son adresse
  - En adressage indirect, on affecte à AX la valeur contenue à l'adresse donnée par le registre BX

    exemple: `MOV AX, [BX]`
  - Il existe également plusieurs moyens pour réaliser un adressage indirect en fonction des registres utilisés (BP,SI,DI):
    - adressage basé: `MOV AX,[BP]`
    - adressage indexé: `MOV AX,[SI +123h]`
    - adressage basé indexé: `MOV AX,ES:[BP + SI + 123h]`

Structure d'un programme 8086



Structure d'un programme 8086

- le programme spécifie quelles sont les parties appartenant au segment de code, au segment de données ou au segment de pile
- Chaque partie est repérée par une étiquette (Data pour le segment de données ou Code pour celui de code)
- Les étiquettes peuvent être nommées de façon arbitraire (au choix du programmeur)
- Une étiquette est une manière symbolique de représenter une adresse en mémoire

Structure d'un programme 8086

- Ces étiquettes sont suivies de directives d'assemblages:
  - **SEGMENT** : spécifie le début du segment
  - **ENDS** : spécifie la fin du segment
- L'étiquette **Main** permet de spécifier le point d'entrée du programme, c'est à dire la première instruction à exécuter lors du lancement du programme
- De la même manière la directive **END Main** repère la fin du programme
- La directive **Assume** permet de spécifier quelle partie du programme constitue le code (étiquette associée à CS) et quelle partie du programme est associée aux données (étiquette associée à DS)

Déclaration de variables

- L'espace mémoire défini par le segment de données peut être initialisé afin de stocker des constantes (chaînes de caractères, valeurs numériques) ou des variables
- Avant de pouvoir être utilisées dans le programme, les variables et constantes doivent être déclarées
- Il existe plusieurs manières de déclarer les constantes et les variables en fonction de leur taille et de leur structure
- Usuellement, on utilise une déclaration du type

```
nom DB constante, [constante]
```

Déclaration de variables

```
Assume CS:Code, DS:Data

Data SEGMENT
  N1        dw      25h
  N2        dw      4
  Prod      dw      ?
            ...
  NP        dw      2,3,5,7,11,13
  Table     dw      10,20,30,40,50


TabVide     dw      100 dup (?)
  Tab1      dw      50 dup (1b)

Chaine      db      'TEXTE',0Dh,0Ah

Data ENDS
```

db = Define Byte (Byte = Octet = 8 bits)
dw = Define Word (Word = Mot = 16 bits)
dd = Define Double (Double = 32 bits)
+ valeur initiale (ou liste de valeurs) ou de ?

N1 est une variable de type WORD (dw = Define Word) initialisée à 25 (en héxa)

Table est un tableau (une suite) de WORD initialisé à 10…50 (en décimal)

TabVide est un tableau de 100 WORD non initialisé

Tab1 est un tableau de 50 WORD initialisés à 1 (en binaire)