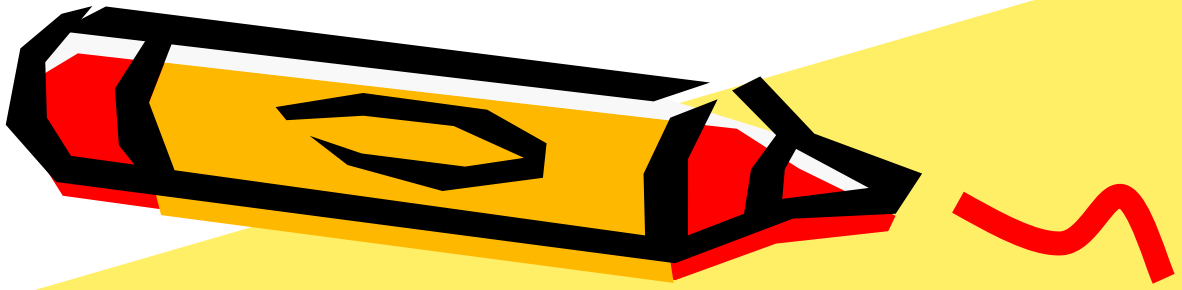


Cours
Architectures des Ordinateurs
2ème Année Licence en Informatique

Partie 2

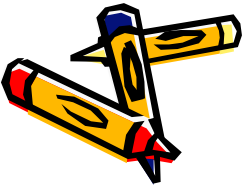
Introduction au langage machine du processeur 8086

❑Jeu d'instructions

➢Types d'instructions

1 / Instructions d'affectation : Déclenchent un transfert de données entre deux registres du processeur, entre l'un des registres du processeur et la mémoire principale, ou bien, d'une constante vers un registre du processeur ou vers un emplacement mémoire.

- ❖ **transfert CPU → Mémoire Principale (MP) (= lecture en MP) ;**
- ❖ **transfert CPU → Mémoire Principale (MP) (= écriture en MP) ;**
- ❖ **transfert CPU → CPU ;**
- ❖ **transfert CPU → Constante ;**
- ❖ **transfert constante → Mémoire Principale (MP) (= écriture en MP) ;**

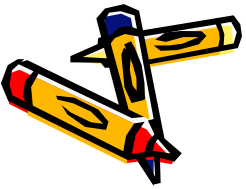


Introduction au langage machine du processeur 8086

2 / Instructions arithmétiques et logiques : Opérations entre les données de deux registres du processeur, entre une donnée de l'un des registres du processeur et une constante ou une valeur contenue dans un emplacement mémoire, ou bien, entre une donnée contenue dans un emplacement mémoire et une constante.

Exemples :

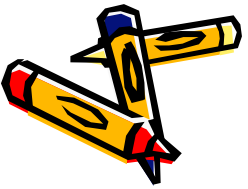
- ✓ **addition : AX → AX + donnée ;**
- ✓ **soustraction : AX → AX - donnée ;**
- ✓ **incrémentation de AX : AX → AX + 1 ;**
- ✓ **décrémentation : AX → AX - 1 ;**
- ✓ **décalages à gauche et à droite ;**



Introduction au langage machine du processeur 8086

3/ Instructions de comparaison:

- Comparaison du contenu de deux registres
- d'un registre et d'un emplacement mémoire;
- d'un registre et une constante;
- d'un emplacement mémoire et une constante, et positionnement des indicateurs.



Introduction au langage machine du processeur 8086

4/ Instructions de branchement

La prochaine instruction à exécuter est repérée en mémoire par le registre IP.

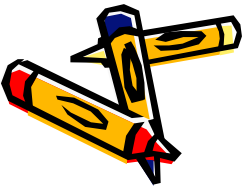
- Les instructions de branchement permettent de modifier la valeur de IP pour exécuter une autre instruction (boucles, tests, etc.).

- **On distingue deux types de branchements :**
  - ***branchements inconditionnels : IP → adresse d'une instruction ;***
  - ***branchements conditionnels :***

    Si *une condition est satisfaite,* ***alors*** *branchement,*

    |

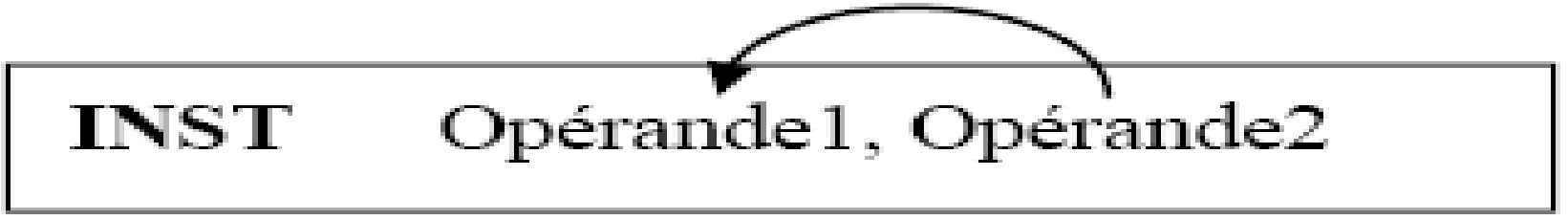
    Sinon **passage simple à l'instruction suivante.**



Introduction au langage machine du processeur 8086

❑ **Codage des instructions et mode d'adressage :** Les instructions et leurs opérandes (**paramètres**) sont stockés en mémoire principale.

- **La taille totale d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type d'instruction et aussi du type d'opérande.**
- **Chaque instruction est toujours codée sur un nombre entier d'octets, afin de faciliter son décodage par le processeur.**
- **La structure la plus générale d'une instruction est la suivante :**



L'opération est réalisée entre les 2 opérandes et le résultat est récupéré dans l'opérande de gauche.

Introduction au langage machine du processeur 8086

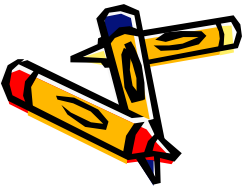
❑ **Exemple :**

ADD AX, BX	**Instruction ADD qui fait l'addition des deux registres AX et BX et retourne le résultat dans le registre AX**
MOV AX, BX	**Instruction MOV qui charge le contenu du registre BX dans le registre AX.**

Il y a aussi des instructions qui agissent sur un seul opérande ;

❑ **Exemple**

INC AX	**Instruction INC qui ajoute 1au contenu du registre AX (AX ← AX + 1).**

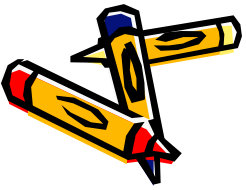


Introduction au langage machine du processeur 8086

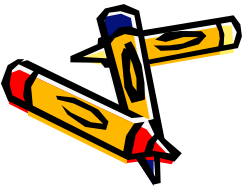
- **D'une manière générale, une instruction est composée de deux champs :**
  - **Le code opération, qui indique au processeur quelle instruction réaliser ;**
  - **Le champ opérande qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).**

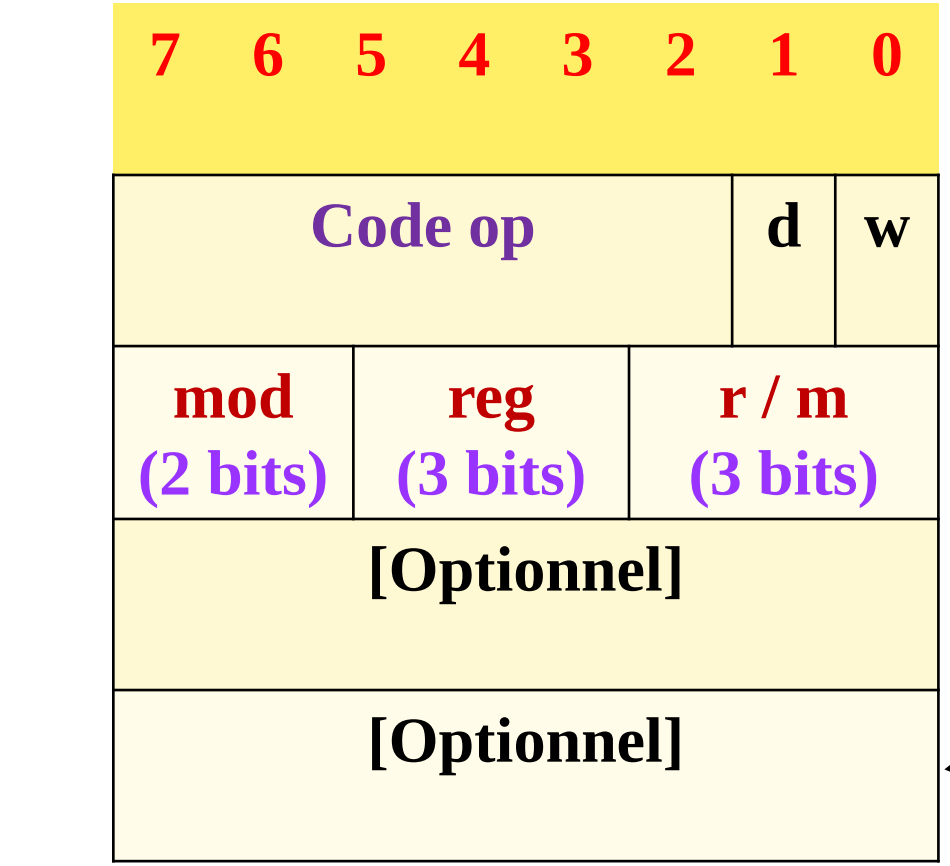
Champ Code opération	Champ Code opérande

Pour plus de détaille, la structure exacte d'une instruction 8086 est la suivante :



Introduction au langage machine du processeur 8086





1er Octet : octet de code opération

2ème Octet : octet de mode d'adressage

3ème Octet : octet bas de déplacement , d'adresse ou de la donnée

4ème Octet : octet haut de déplacement , d'adresse ou de la donnée

Introduction au langage machine du processeur 8086

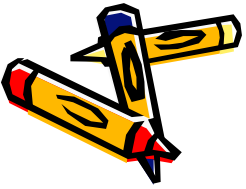
❑ **1er octet (octet de code opération) : dans ce premier octet d'instruction on trouve le code d'opération à exécuter sur 6 bits, plus deux bits d et w.**

➢ **w : indique la taille des mots et des registres traités dans cette instruction.**

- ✓ **w = 0 : ce sont des octets (8 bits) ;**
- ✓ **w = 1 : il s'agit de mots et de registres de 16 bits.**

➢ **d : indique la destination du résultat (vers un registre ou bien vers une case mémoire).**

- ✓ **d = 0 : la destination du résultat est vers une case mémoire ;**
- ✓ **d = 1 : la destination du résultat est vers un registre, s'il y a deux registres en cause, il s'agit du premier nommé dans le mnémonique, et il est codé, dans tous les cas par le champ « reg » du deuxième octet d'instruction.**



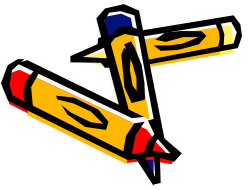
Introduction au langage machine du processeur 8086

❑ **2ème octet (octet de mode d'adressage) : Le deuxième octet de l'instruction est constitué de trois champs :**

➢ **mod (2 bits) : permet de déterminer comment interpréter le champ r/m, comme le montre le tableau_I qui va suivre,**

➢ **reg (3 bits) : contient le code du registre utilisé comme opérande dans cette instruction, les registres sont codés selon le tableau_II qui suit.**

➢ **r/m (3 bits) : permet de définir le mode de calcul d'adresse effective (AE) de la deuxième opérande, il peut aussi être considéré comme un registre lorsque mod = 11.**



Introduction au langage machine du processeur 8086

	mod=00	mod = 01	mod = 10
r/m	AE=	AE=	AE=
000	(BX) + (SI)	(BX)+(SI)+Dép de 8 bits	(BX)+(SI)+Dép de 16 bits
001	(BX) + (DI)	(BX)+(DI)+Dép de 8 bits	(BX)+(DI)+Dép de 16 bits
010	(BP) + (SI)	(BP)+(SI)+Dép de 8 bits	(BP)+(SI)+Dép de 16 bits
011	(BP) + (DI)	(BP)+(DI)+Dép de 8 bits	(BP)+(DI)+Dép de 16 bits
100	(SI)	(SI)+Dép de 8 bits	(SI)+Dép de 16 bits
101	(DI)	(DI)+Dép de 8 bits	(DI)+Dép de 16 bits
110	Dép 16 bits	(BP)+Dép de 8 bits	(BP)+Dép de 16 bits
111	(BX)	(BX)+Dép de 8 bits	(BX)+Dép de 16 bits

Tableau_I : Codes des modes d'adressage

Introduction au langage machine du processeur 8086

16 bits (w=1)	8 bits (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Tableau_II : Codes des registres

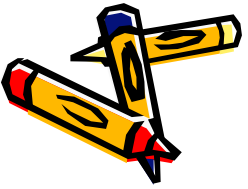
Introduction au langage machine du processeur 8086

❑**Les 2 derniers octets :** **Les 2 derniers octets sont optionnels, on les trouve uniquement lorsqu'il y a :**

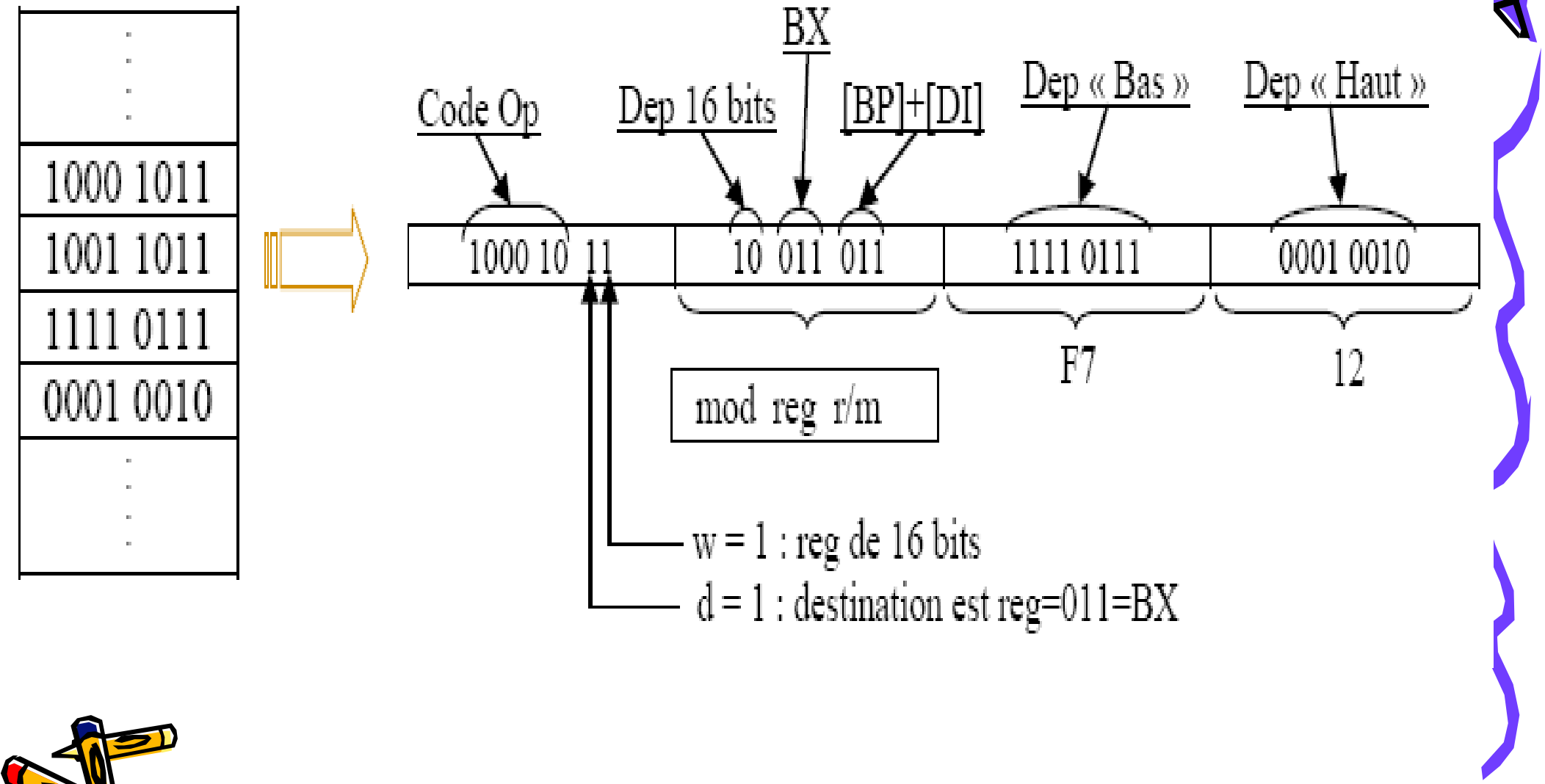
- ❖ **un adressage immédiat**
- ❖ **ou bien lorsqu'il y a un adressage direct ou indirect avec déplacement**
  - ✓ **c'est-à-dire lorsque le champs :**
    - ➢ **mod=00 et r/m=110,**
    - ➢ **ou bien mod=01 ou mod=10,**

Exemple de codage d'une instruction MOV :

Soit l'instruction MOV suivante : MOV BX,[BP+DI+12F7]



Introduction au langage machine du processeur 8086



Introduction au langage machine du processeur 8086

- **Remarque importantes**
  - **Les registres segment : (Rappel)**

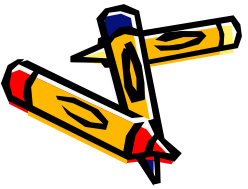
	15 ... 0
Code segment	CS
Data segment	DS
Stack segment	SS
Extra segment	ES

Introduction au langage machine du processeur 8086

- **Le 8086 a quatre registres segments de 16 bits chacun :**
  - **CS (code segment,**
  - **DS (Data segment),**
  - **ES (Extra segment)**
  - **SS (stack segment),**

- **Ces registres sont chargés de sélectionner les différents segments de la mémoire en pointant sur le début de chacun d'entre eux.**

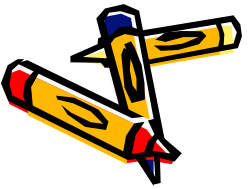
- **Chaque segment de mémoire ne peut excéder les 65535 octets.**



Introduction au langage machine du processeur 8086

❑ **Le registre CS (code segment) : Il pointe sur le segment qui contient les codes des instructions du programme en cours.**

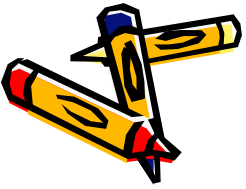
✓ **Remarque: Si la taille du programme dépasse les 65535 octets alors on peut diviser le code sur plusieurs segments (chacun ne dépasse pas les 65535 octets) et pour basculer d'une partie à une autre du programme il suffit de changer la valeur du registre CS et de cette manière on résout le problème des programmes qui ont une taille supérieure à 65535 octets.**



Introduction au langage machine du processeur 8086

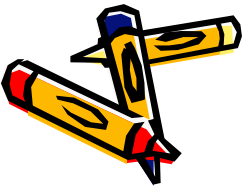
❑**Le registre DS (Data segment) : Le registre segment de données pointe sur le segment des variables globales du programme, bien évidemment la taille ne peut excéder 65535 octets (si on a des données qui dépassent cette limite, on utilise la même astuce citée dans la remarque précédente mais dans ce cas on change la valeur de DS).**

❑**Le registre ES (Extra segment) :** Le registre de données supplémentaires ES est utilisé par le microprocesseur lorsque l'accès aux autres registres est devenu difficile ou impossible pour modifier des données, de même ce segment est utilisé pour le stockage des chaînes de caractères.



Introduction au langage machine du processeur 8086

- **Le segment SS (Stack segment):** **Le registre SS pointe sur la pile: la pile est une zone mémoire ou on peut sauvegarder les registres ou les adresses ou les données pour les récupérer après l'exécution d'un sous programme ou l'exécution d'un programme d'interruption, en général il est conseillée de ne pas changer le contenu de ce registre car on risque de perdre des informations très importantes (exemple les passages d'arguments entre le programme principal et le sous programme)**

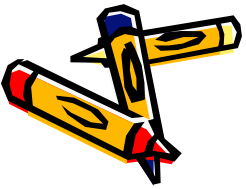


Introduction au langage machine du processeur 8086

- **Comme nous avons vu, selon la manière dont la donnée est spécifiée, c'est à dire selon le mode d'adressage de la donnée,**
  - **une instruction sera codée par 1, 2, 3 ou 4 octets.**

- **Nous distinguerons ici neuf modes d'adressage :**

  - **implicite, immédiat, direct, basé, indexé, basé et indexé, basé avec déplacement, indexé avec déplacement, basé et indexé avec déplacement.**

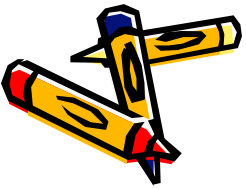


Introduction au langage machine du processeur 8086

❑ **1/ Adressage implicite : L'instruction contient seulement le code opération, sur 1 ou 2 octets. L'instruction porte sur des registres ou spécifie une opération sans opérande (exemple : "incrémenter AX" INC AX).**

❑ **2/ Adressage immédiat : Le champ opérande contient la donnée (une valeur constante sur 1 ou 2 octets).**

Exemple : mov ah,10h ; => Opcode = B410

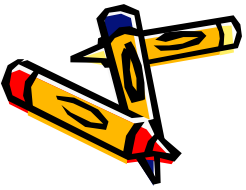


Introduction au langage machine du processeur 8086

❑ **3/ Adressage direct : Le champ opérande contient l'adresse de la donnée en mémoire principale sur 2 octets (mod = 00 et r/m = 110).**

- ✓ **Attention : dans le 80x86, les adresses sont toujours manipulées sur 16 bits, quelle que soit la taille réelle du bus.**

❑ **Exemple : mov al,[10h] ; <=> mov al,[ds:10h] => Opcode = A01000**

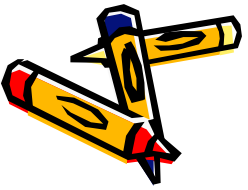


Introduction au langage machine du processeur 8086

❑ **4/ Adressage basé : l'opérande est une case mémoire dont l'adresse est donnée par un des deux registres de base BX (pour le segment de données) ou BP (pour le segment de pile).**

➢ **Exemple : mov ah,[bx] ; <=> mov ah,[ds:bx] => Opcode = 8A27**
mov al,[bp] ; <=> mov al,[ss:bp] => Opcode = 8A4600

❑ **5/ Adressage indexé : l'opérande est une case mémoire dont l'adresse est donnée par un des deux registres d'index SI ou DI,**
Exemple : mov ah,[si] ; <=> mov ah,[ds:si] => Opcode = 8A24



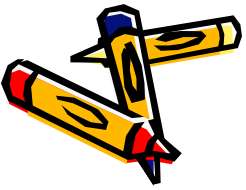
Introduction au langage machine du processeur 8086

❑ **6/ Adressage basé et indexé : l'opérande est une case mémoire dont l'adresse est donnée par un registre de base (BX ou BP) plus un registre d'index (SI ou DI),**

➢ **Exemple: mov ah,[bx+di]; <=> mov ah,[ds:bx+di] => Opcode = 8A21**
mov [bp+si],ah; <=> mov [ss:bp+si],ah => Opcode = 8822

❑ **7/ Adressage basé avec déplacement: l'opérande est une case mémoire dont l'adresse est donnée par un registre de base (BX ou BP) plus un déplacement de 8 bits ou de 16 bits,**

➢ **Exemple : mov ah,[bx+123h] ; <=>**
➢ **mov ah,[ds:bx+123h] => Opcode = 8AA72301**



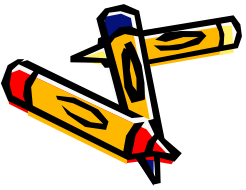
Introduction au langage machine du processeur 8086

❑ **8/ Adressage indexé avec déplacement: l'opérande est une case mémoire dont l'adresse est donnée par un registre d'index (SI ou DI) plus un déplacement de 8 bits ou de 16 bits,**

- **Exemple : mov ah,[di+123h] ; <=> mov ah,[ds:di+123h] => Opcode = 8AA52301**

❑ **9/Adressage basé et indexé avec déplacement: l'opérande est une case mémoire dont l'adresse est donnée par un registre de base (BX ou BP) plus un registre d'index (SI ou DI) plus un déplacement de 8 bits ou de 16 bits,**

- **Exemple: mov ah,[bx+si+123h]; <=> mov ah,[ds:bx+si+123h] => Opcode = 8AA02301**



Introduction au langage machine du processeur 8086

❑ **Les instructions du 8086**

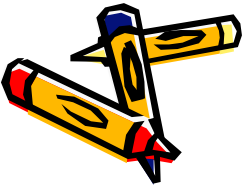
1. **Les instructions de transfert**

➢ **Instruction MOV :**

❖ **Syntaxe : MOV OpD, OpS ;**

❖ **interprétation : OpD ⟸ OpS**

➢ **Description : Transfert d'un octet ou d'un mot de 16 bits de l'opérande source OpS vers l'opérande destination OpD**



Introduction au langage machine du processeur 8086

❑**Exemples :**

- **MOV CX, AX**
- **MOV AH, DL**
- **MOV DX, 9A00**

❑ **Combinaisons possibles :**

✓ **MOV Registre1, Registre2** **copier un registre dans un autre**

✓ **MOV Registre, Mémoire** **copier le contenu d'une case mémoire dans un registre**

✓ **MOV Mémoire , Registre** **copier un registre dans une case mémoire**

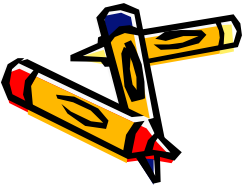
✓ **MOV Registre, immédiat** **copier une constante dans un registre**

✓ **MOV Mémoire, immédiat** **copier une constante dans une case mémoire**

Combinaisons interdites :

✓ **MOV ~~Mémoire, Mémoire~~** **copier une case mémoire dans une autre case mémoire**

❑**Les indicateurs affectés : l'instruction MOV ne modifie aucun indicateur du registre PSW.**



Introduction au langage machine du processeur 8086

➢ Instruction PUSH :

- ❖ **Syntaxe : PUSH opérande ;**
- ❖ **Description : Sauvegarde le contenu d'un registre (16 bits) ou d'une case mémoire (16 bits) dans la pile (dans le segment de pile). Opération d'empilement.**

❑ **PUSH est une action composée de :**

- ➢ **SP ← SP – 2 ;**
- ➢ **[SP] ← opérande ;**

❑ **Exemples : -- PUSH CX -- PUSH [BX]**

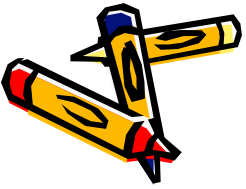
❑ **Combinaisons possibles :**

- ➢ **PUSH Registre 16 bits** — **empiler un registre de 16 bits**
- ➢ **PUSH Mémoire** — **empiler le contenu de deux cases mémoire de 8 bits**

❑ **Combinaisons interdites :**

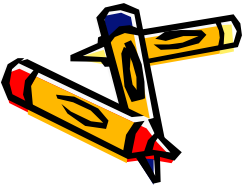
- ➢ ~~**PUSH opérande 8 bits**~~ — **empiler une opérande de 8 bits**
- ➢ ~~**PUSH immédiat**~~ — **empiler une constante**

❑ **Les indicateurs affectés: l'instruction PUSH ne modifie aucun indicateur du registre PSW.**



Introduction au langage machine du processeur 8086

- **Instruction POP :**
  - Syntaxe : POP opérande ;
  - Description : Restauration d'un mot 16 bits depuis la pile vers un registre ou une case mémoire.
- **POP est une action composée de :**
  - Opérande 16 bits ← [SP];
  - SP ← SP + 2 ;
- **Exemples : POP CX POP [12A0]**
- **Combinaisons possibles:**
  - POP Registre 16 bits dépiler vers un registre de 16 bits
  - POP Mémoire dépiler vers deux cases mémoire de 8 bits
- **Combinaisons interdites:**
  - ~~POP opérande 8 bits~~ dépiler vers un opérande de 8 bits
  - ~~POP immédiat~~ dépiler vers une constante
- Les indicateurs affectés : l'instruction POP ne modifie aucun indicateur du registre PSW.



Introduction au langage machine du processeur 8086

➢ Instruction XCHG :

- ➢ Syntaxe : XCHG OpD, OpS;
- ➢ Description : L'échange entre l'opérande source OpS et l'opérande destination OpD.

➢ XCHG est une action composée de :

- ➢ Opérande intermédiaire ← OpD;
- ➢ OpD ← OpS;
- ➢ OpS ← Opérande intermédiaire ;

➢ Exemples : XCHG CX, SP XCHG [120], DX

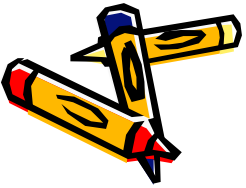
❑ Combinaisons possibles :

- ➢ XCHG Registre1, Registre2 — échange entre deux registres
- ➢ XCHG Mémoire, Registre — échange entre une case mémoire et un registre
- ➢ XCHG Registre, Mémoire — échange entre une case mémoire et un registre

❑ Combinaisons interdites :

- ➢ ~~XCHG Mémoire, Mémoire~~ — échange entre deux cases mémoire
- ➢ ~~XCHG Registre segment, operande2~~ — échange avec un registre segment

❑ Les indicateurs affectés : l'instruction XCHG ne modifie aucun indicateur du registre PSW.



Introduction au langage machine du processeur 8086

2. Les instructions Arithmétiques

❑ Instruction ADD :

- ➢ **Syntaxe : ADD OpD, OpS ;**
- ➢ **interprétation : OpD ← OpD + OpS**
- ➢ **Description : Additionne l'opérande destination OpD et l'opérande source OpS avec résultat dans l'opérande destination OpD.**

❑**Exemples : ADD CX, AX**

- ➢ **1/ ADD AH, DL** **2 / ADD DX, [9A00]**

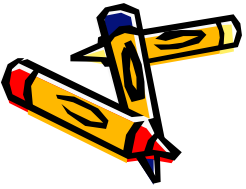
❑ **Combinaisons possibles :**

- ➢**ADD Registre1, Registre2** **Additionner les contenus de deux registres**
- ➢**ADD Registre, Mémoire** **Additionner les contenus d'un registre et d'une case mémoire**
- ➢**ADD Mémoire , Registre** **Additionner les contenus d'une case mémoire et d'un registre**
- ➢**ADD Registre, immédiat** **Additionner le contenu d'un registre et une constante**
- ➢**ADD Mémoire, immédiat** **Additionner le contenu d'une case mémoire et une constante**

❑**Combinaisons interdites:**

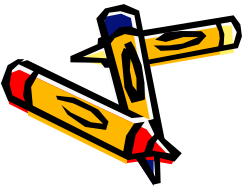
- ➢ ~~**ADD Mémoire, Mémoire**~~ **Additionner les contenus de deux cases mémoire**

❑**Les indicateurs affectés : l'instruction ADD modifie les indicateurs O, S, Z, A, P, C du PSW.**



Introduction au langage machine du processeur 8086

- **Instruction ADC :**
  - **Syntaxe : ADC OpD, OpS;**
  - **interprétation : OpD ← OpD + OpS + CF**
  - **Description : Additionne l'opérande destination OpD, l'opérande source OpS et le curry CF avec résultat dans l'opérande destination OpD.**
- **Exemples : ADC CX, AX**
  - ADC DL,[9A00]
- **Combinaisons possibles : Les même combinaisons que celles de ADD.**
- **Combinaisons interdites : Les même combinaisons que celles de ADD.**
- **Les indicateurs affectés : Les même indicateurs que ceux de ADD.**



Introduction au langage machine du processeur 8086

❑ **Instruction INC :**

- ➢ **Syntaxe : INC Op;**
- ➢ **interprétation : Op ← Op + 1**
- ➢ **Description : Ajouter 1 au contenu de l'opérande Op.**

❑ **Exemples : INC AX**

- ❖ **INC DL**
- ❖ **INC byte [9A00]**

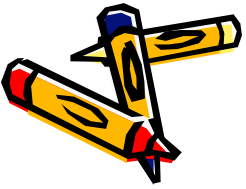
❑ **Combinaisons possibles :**

- ➢ **INC Registre** **incrémenter le contenu d'un registre**
- ➢ **INC Mémoire** **incrémenter le contenu d'une cases mémoire**

❑ **Les indicateurs affectés : l'instruction INC modifie les indicateurs O, S, Z, A, P du PSW.**

✓ **Remarque : Pour incrémenter une case mémoire, il faut préciser la taille :**

- ➢ **1) INC byte [ ]** **2) INC word [ ]**



Introduction au langage machine du processeur 8086

- **Instruction SUB :**
  - **Syntaxe : SUB OpD, OpS ;**
  - **interprétation : OpD ← OpD – OpS**
  - **Description : Soustrait l'opérande source OpS de l'opérande destination OpD avec résultat dans l'opérande destination OpD.**
- **Exemples : SUB AH, DL** **SUB DX, [9A00]**
- **Combinaisons possibles:**
  - **SUB Registre1, Registre2** **Soustraction entre deux registres**
  - **SUB Registre, Mémoire** **Soustraction entre un registre et une case mémoire**
  - **SUB Mémoire , Registre** **Soustraction entre une case mémoire et un registre**
  - **SUB Registre, immédiat** **Soustraction entre un registre et une constante**
  - **SUB Mémoire, immédiat** **Soustraction entre une case mémoire et une constante**
- **Combinaisons interdites :**
  - ~~**SUB Mémoire, Mémoire**~~ **Soustraction entre deux cases mémoire**
- **Les indicateurs affectés : l'instruction SUB modifie les indicateurs O, S, Z, A, P, C du PSW.**

Introduction au langage machine du processeur 8086

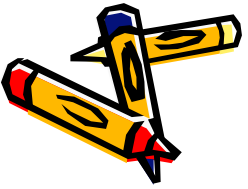
❑Instruction SBB :

- ❑ **Syntaxe : SBB OpD, OpS ;**
- ❑ **interprétation : OpD ← OpD – OpS – CF**
- ❑ **Description : Soustrait l'opérande source OpS et le curry CF de l'opérande destination OpD avec résultat dans l'opérande destination OpD.**

❑Exemples : SBB CX, AX

- ➢ SBB AH, DL
- ➢ SBB DX,[9A00]

- ➢ **Combinaisons possibles : Les même combinaisons que celles de SUB.**
- ➢ **Combinaisons interdites : Les même combinaisons que celles de SUB.**
- ➢ **Les indicateurs affectés : Les même indicateurs que ceux de SUB.**

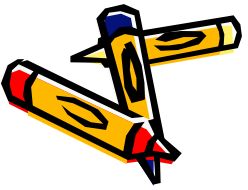


Introduction au langage machine du processeur 8086

- **Instruction DEC :**
  - **Syntaxe : DEC Op;**
  - **interprétation : Op ← Op – 1**
  - **Description : Soustraire 1 au contenu de l'opérande Op.**

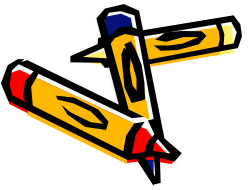
- **Exemples : DEC AX DEC byte [9A00]**

- **Combinaisons possibles : Les même combinaisons que celles de INC.**
- **Les indicateurs affectés : Les même indicateurs que ceux de INC.**



Introduction au langage machine du processeur 8086

- **Instruction NEG :**
  - **Syntaxe : NEG Op;**
  - **interprétation : Op ← – Op**
  - **Description : Forme le complément à 2 d'opérande Op.**
- **Exemples : NEG AX NEG byte [9A00]**
- **Combinaisons possibles :**
  - **NEG Registre calculer le complément à 2 du contenu d'un registre**
  - **NEG Mémoire calculer le complément à 2 du contenu d'une case mémoire**
- **Les indicateurs affectés : l'instruction NEG modifie les indicateurs C, O, S, Z, A, P du PSW.**
- **Remarque : Pour calculer le complément à 2 d'une case mémoire, il faut préciser la taille : NEG byte [ ] NEG word [ ]**

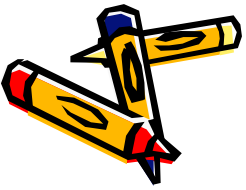


Introduction au langage machine du processeur 8086

❑**Complément à deux :** Le **complément à deux** est une représentation binaire des entiers relatifs qui permet d'effectuer les opérations arithmétiques usuelles naturellement.

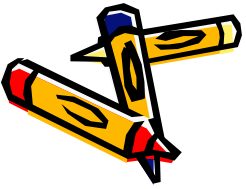
❑ La notation est utilisée sur des écritures de nombres de longueur donnée (nombres écrits couramment sur **8, 16, 32 ou 64 bits**). Dans une telle écriture, on utilise le bit de poids fort (bit le plus à gauche) du nombre pour contenir la représentation de son signe (positif ou négatif, le zéro étant considéré comme positif).

❑ On inverse les bits de l'écriture binaire de sa valeur absolue (opération binaire NON), on fait ce qu'on appelle le complément à un, On ajoute 1 au résultat (les dépassements sont ignorés).



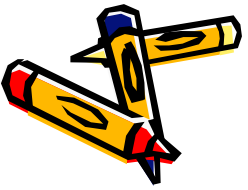
Introduction au langage machine du processeur 8086

- Cette opération correspond au calcul de $2^n - |x|$,
- où est la longueur de la représentation et |x| la valeur absolue du nombre à coder.
- Ainsi -1 s'écrit comme
- pour les nombres sur 8 bits $256\text{-}1=255=11111111_2$ bits. Ceci est à l'origine du nom de cette opération : "complément à 2 puissance n", quasi-systématiquement tronqué en "complément à 2".
- l'opération précédente effectuée sur 00000000 permet d'obtenir 00000000 ($-0 = +0 = 0$) et l'addition usuelle des nombres binaires fonctionne.
- La même opération effectuée sur un nombre négatif donne le nombre positif de départ: $2^n - (2^n - x) = x$.



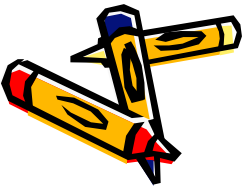
Introduction au langage machine du processeur 8086

- **Exemple : Pour coder (−4) :**
  - **On prend le nombre positif 4 : 00000100**
  - **On inverse les bits : 11111011**
  - **On ajoute 1 : 11111011 + 00000001 = 11111100**
  - **Le bit de signe est automatiquement mis à 1 par l'opération d'inversion. On peut vérifier que cette fois l'opération**
  - **3 + (−4) = (-1) se fait sans erreur :**
  - **00000011 + 11111100 = 11111111**
- **Le complément à deux de 11111111 est 00000001 soit 1 en décimal,**
- **Donc 11111111 = (−1) en décimal.**
- **Le résultat de l'addition usuelle de nombres représentés en complément à deux est le codage en complément à deux du résultat de l'addition des nombres.**
- **Ainsi les calculs peuvent s'enchaîner naturellement.**



Introduction au langage machine du processeur 8086

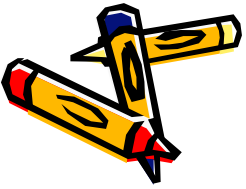
- **Instruction CMP :**
  - **Syntaxe : CMP OpD, OpS ;**
  - **interprétation : OpD – OpS**
  - **Description : Soustrait l'opérande source OpS de l'opérande destination OpD et positionne les drapeaux (indicateurs du registre d'état) en fonction du résultat sans sauvegarde de celui-ci (l'opérande destination OpD n'est pas modifié).**
- **Exemples : CMP AL, DL CMP BX,[9A00]**
- **Combinaisons possibles : Les même combinaisons que celles de SUB.**
- **Combinaisons interdites : Les même combinaisons que celles de SUB.**
- **Les indicateurs affectés : Les même indicateurs que ceux de SUB**



Introduction au langage machine du processeur 8086

- **Instruction MUL :**
  - **Syntaxe : MUL Op;**
  - **Interprétation : AX • Op × AL (Si la taille de Op = 8 bits)**
    DX:AX • Op × AX (Si la taille de Op=16 bits)
  - **Description : instruction à une seule opérande. Elle effectue une multiplication non signée entre l'accumulateur (AL ou AX) et l'opérande Op. Le résultat de taille double est stocké dans l'accumulateur et son extension (AH:AL ou DX:AX).**
- **Exemples : MUL CX** MUL Byte [9A00]
- **Combinaisons possibles :** MUL Registre — Multiplication d'un registre
  MUL Mémoire — Multiplication d'une case mémoire
- **Combinaisons interdites :** ~~MUL immédiat~~ — Multiplication d'une constante
- **Les indicateurs affectés : l'instruction MUL modifie les indicateurs O, C du PSW.**

- **Instruction IMUL : Identique à MUL excepté qu'une multiplication signée est effectuée.**

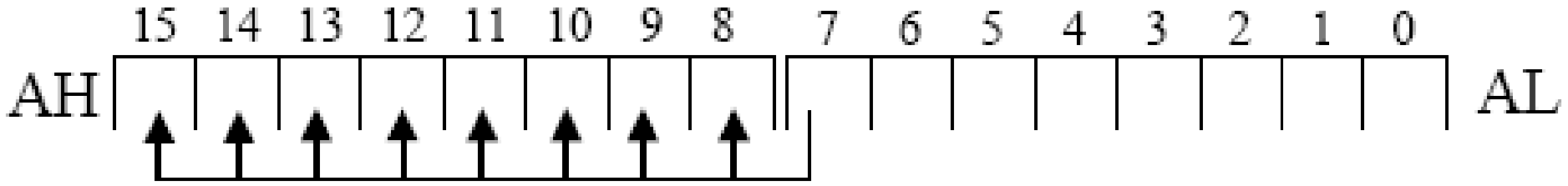


Introduction au langage machine du processeur 8086

- **Instruction DIV :**
- **Syntaxe : DIV Op;**
- **interprétation :**
  - **Si la taille de Op = 8 bits**
    - AL ← Le quotient de AX ÷ Op ; AH ←- Le reste de AX ÷ Op ;
  - Si la taille de Op = 16 bits
    - AX ← Le quotient de :AX ÷ Op ; DX ← Le reste de AX ÷ Op ;
- **Description : Effectue la division AX÷Op ou (DX:AX)÷Op16 selon la taille de Op qui doit être soit un registre soit une mémoire. Dans le dernier cas il faut préciser la taille de l'opérande, exemple : DIV byte [adresse] ou DIV word [adresse].**
- **Exemples : DIV DL DIV Byte[9A00]**
- **Combinaisons possibles :**
  - DIV Registre — **Division par un registre**
  - DIV Mémoire — **Division par une case mémoire**
- **Combinaisons interdites :**
  - DIV immédiat — Division par une constante
- **Les indicateurs affectés : l'instruction DIV ne modifie aucun indicateurs du PSW.**
- **Instruction IDIV : (Intègre DIV) Identique à DIV mais effectue une division signée.**

Introduction au langage machine du processeur 8086

- **Instruction CBW :**
- **Syntaxe : CBW**
- **Description : (Convert Byte to Word) Effectue une extension de AL dans AH. On écrit le contenu de AL dans AX en respectant le signe.**



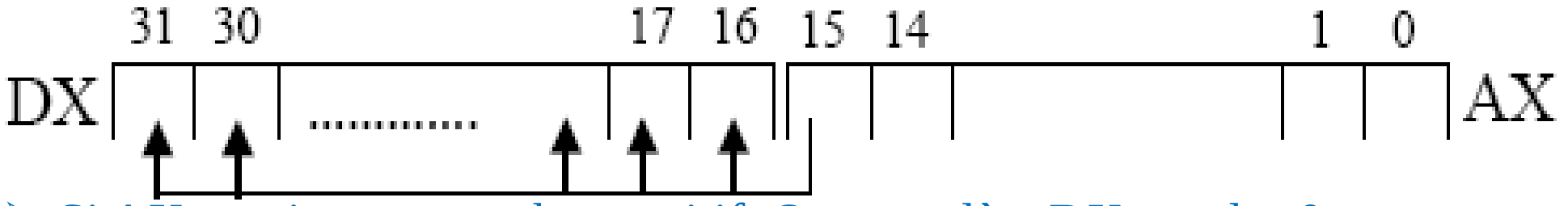
- **Si AL contient un nombre positif, On complète AH par des 0.**
- **Si AL contient un nombre négatif, On complète AH par des 1.**

Introduction au langage machine du processeur 8086

❑**Instruction CWD :**

❑ **Syntaxe : CWD**

❑ **Description : (Convert Word to Double Word) effectue une extension de AX dans DX en respectant le signe. On écrit AX dans le registre 32 bits obtenu en collant DX et AX.**



➢ Si AX contient un nombre positif, On complète DX par des 0.

➢ Si AX contient un nombre négatif, On complète DX par des 1.

➢ **Remarque : Ces deux instructions (CBW, CWD) précèdent généralement une**

➢ **instruction IDIV.**

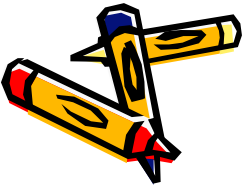
Introduction au langage machine du processeur 8086

3. Les instructions logiques

- **Instruction NOT :**
- **Syntaxe : NOT Op;**
- **interprétation : Op ← $\overline{Op}$**
- **Description : Forme le complément à 1 d'opérande Op.**
- **Effectue un NON logique bit à bit sur l'opérande Op (i.e. chaque bit de l'opérande Op est inversé).**
- **Exemples : NOT AX,** NOT byte [9A00]
- **Combinaisons possibles :**
  - NOT Registre calculer le complément à 1 du contenu d'un registre
  - NOT Mémoire calculer le complément à 1 du contenu d'une cases mémoire
- **Combinaisons interdites :**
  - ~~**NOT Immédiat**~~ calculer le complément à 1 d'une constante.
- **Les indicateurs affectés : l'instruction NOT ne modifie aucun indicateur du PSW.**

Introduction au langage machine du processeur 8086

- **Instruction AND :**
- **Syntaxe : AND OpD, OpS ;**
- **interprétation : OpD ← OpD and OpS**
- **Description : Effectue un ET logique bit à bit entre l'opérande destination OpD et l'opérande source OpS.**
- **Le résultat est stocké dans l'opérande destination OpD.**
- **Exemples : AND AH, DL AND DX, [9A00]**
- **Combinaisons possibles:**
  - AND Registre1, Registre2 ET logique entre deux registres
  - AND Registre, Mémoire ET logique entre un registre et une case mémoire
  - AND Mémoire , Registre ET logique entre une case mémoire et un registre
  - AND Registre, immédiat ET logique entre un registre et une constante
  - AND Mémoire, immédiat ET logique entre une case mémoire et une constante
- **Combinaisons interdites :**
- ~~AND Mémoire, Mémoire~~ ET logique entre deux cases mémoire
- **Les indicateurs affectés : l'instruction AND modifie les indicateurs S, Z, P du PSW.**

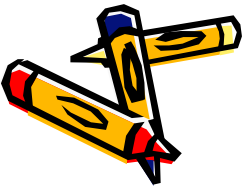


Introduction au langage machine du processeur 8086

- **Instruction OR :**
- **Syntaxe : OR OpD, OpS ;**
- **interprétation : OpD ← OpD OR OpS**
- **Description : Effectue un OU logique inclusif bit à bit entre l'opérande destination OpD et l'opérande source OpS.**
- **Le résultat est stocké dans l'opérande destination OpD.**

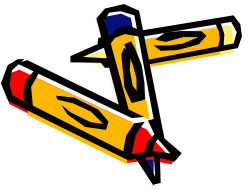
- **Exemples : OR AH,DL OR DX,[9A00]**
- **Combinaisons possibles : Les même combinaisons que celles de AND.**
- **Combinaisons interdites : Les même combinaisons que celles de AND.**

- **Les indicateurs affectés : Les même indicateurs que ceux de AND.**



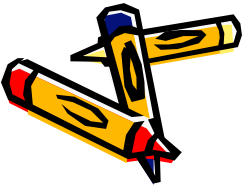
Introduction au langage machine du processeur 8086

- **Instruction XOR :**
- **Syntaxe : XOR OpD, OpS ;**
- **interprétation : OpD ← OpD ou exclusif OpS**
- **Description : Effectue un OU logique exclusif bit à bit entre l'opérande destination OpD et l'opérande source OpS.**
- **Le résultat est stocké dans l'opérande destination OpD.**
- **Exemples : XOR AH, DL XOR DX, [9A00]**
- **Combinaisons possibles : Les même combinaisons que celles de OR.**
- **Combinaisons interdites : Les même combinaisons que celles de OR.**
- **Les indicateurs affectés : Les même indicateurs que ceux de OR.**



Introduction au langage machine du processeur 8086

- **Instruction TEST :**
- **Syntaxe : TEST OpD, OpS ;**
- **interprétation : OpD et OpS**
- **Description : Effectue un ET logique bit à bit entre l'opérande destination OpD et l'opérande source OpS.**
- **Le résultat n'est pas conservé, donc l'opérande destination OpD n'est pas modifié. Seuls les flags sont affectés.**
- **Cet opérateur est souvent utilisé pour tester certains bits de l'opérande destination OpD.**
- **Exemples : TEST AH, DL TEST DX, [9A00]**
- **Combinaisons possibles : Les même combinaisons que celles de AND.**
- **Combinaisons interdites : Les même combinaisons que celles de AND.**
- **Les indicateurs affectés : Les même indicateurs que ceux de AND.**

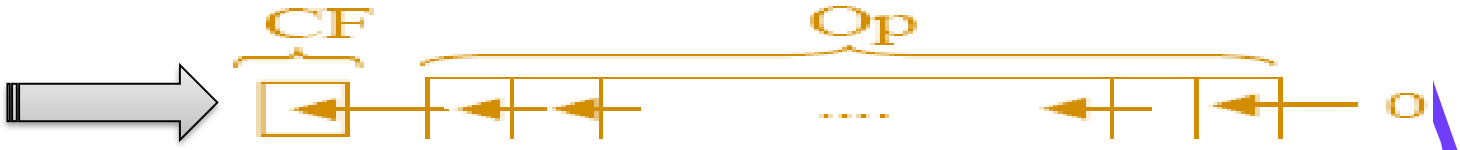


Introduction au langage machine du processeur 8086

4 Les instructions de décalage

❑**Instruction SHL ou SAL :**

❑ **Syntaxe : SHL Op, 1;**

❑ **interprétation :** 

❑**Description : Décaler l'opérande Op d'un bit vers la gauche.**

❑ **Le bit le plus à gauche passe vers le Carry Flag (CF) et le bit le plus à droite est remplacé par un zéro.**

❑ **Remarques : Pour faire un décalage de plusieurs bits, on utilise le registre CL pour indiquer le nombre de bits à décaler.**

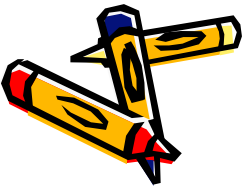
❑ SHL (**SHift Logical Left**) c'est pour le décalage logique à gauche et SAL (SHift Arithmetic Left) c'est pour le décalage arithmétique à gauche.

Introduction au langage machine du processeur 8086

❑**Exemples :** **SHL AX, 1** **MOV CL, 3**
SHL byte [9A00], 1 **SHL BX, CL**

❑**Combinaisons possibles :**

- ❑ SHL Registre, 1 décalage d'un bit du contenu d'un registre
- ❑ SHL Mémoire, 1 décalage d'un bit du contenu d'une cases mémoire
- ❑ **Combinaisons interdites :**
- ❑ ~~**SHL Immédiat, 1**~~ décalage d'une constante.
- ❑ **Les indicateurs affectés : l'instruction SHL modifie les indicateurs O, S, Z, A, P, C du PSW.**



Introduction au langage machine du processeur 8086

❑**Instruction SHR :**

❑ **Syntaxe : SHR Op, 1;**

❑ **interprétation :** 

❑ **Description : SHR (SHift Logical Right) décalage logique à droite.**

❑ **Décaler l'opérande Op d'un bit vers la droite.**

❑ **Le bit le plus à droite passe vers le Carry Flag (CF) et le bit le plus à gauche est remplacé par un zéro.**

❑ **Exemples :** **SHR AX, 1** **MOV CL, 3**
SHR byte [9A00], 1 **SHR BX, CL**

❑**Combinaisons possibles : Les même combinaisons que celles de SHL.**

❑ **Combinaisons interdites : Les même combinaisons que celles de SHL.**

❑ **Les indicateurs affectés : l'instruction SHR modifie les indicateurs O, S, Z, A, P, C du PSW.**

Introduction au langage machine du processeur 8086

❑ **Instruction SAR :**

❑ **Syntaxe : SAR Op, 1;**

❑ **interprétation :**

❑ **Description : SAR (SHift Arithmetic Right) décalage arithmétique à droite.**

❑ **Décaler l'opérande Op d'un bit vers la droite avec sauvegarde de bit de signe.**

❑ **Le bit le plus à droite passe vers le Carry Flag (CF) et le bit le plus à gauche est recopié pour garder le signe.**

❑ **Exemples :**

```
SAR AX, 1                MOV CL, 3
SAR byte [9A00], 1       SAR BX, CL
```

❑ **Combinaisons possibles : Les même combinaisons que celles de SHR.**

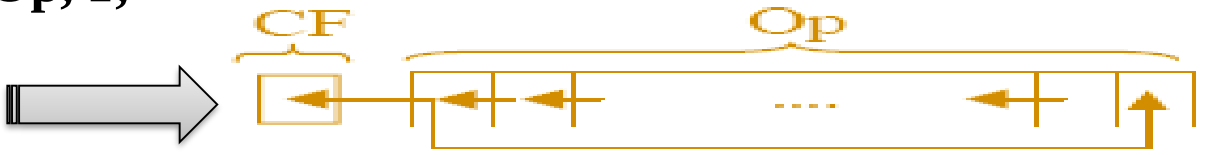
❑ **Combinaisons interdites : Les même combinaisons que celles de SHR.**

❑ **Les indicateurs affectés : l'instruction SAR modifie les indicateurs O, Z, A, P, C du PSW.**

Introduction au langage machine du processeur 8086

❑**Instruction ROL :**

❑ **Syntaxe : ROL Op, 1;**

❑ **interprétation :** 

❑ **Description : ROL (Rotate Left) Rotation à gauche.**

❑ **Effectue une rotation de l'opérande Op d'un bit vers la gauche.**

❑ **Une copie du bit le plus à gauche passe vers le Carry Flag (CF) et une copie vers le bit le plus à droite.**

❑ **Remarque : Pour faire une rotation de plusieurs bits, on utilise le registre CL pour indiquer le nombre de bits de rotation.**

❑**Exemples :** **ROL DX, 1** **MOV CL, 3**

ROL byte [9A00], 1 **ROL BX, CL**

❑ **Combinaisons possibles : Les même combinaisons que celles de SHL.**

❑ **Combinaisons interdites : Les même combinaisons que celles de SHL.**

❑ **Les indicateurs affectés : l'instruction ROL modifie les indicateurs O, S, Z, A, P, C du PSW.**

Introduction au langage machine du processeur 8086

- **Instruction ROR :**
- **Syntaxe : ROR Op, 1;**
- **interprétation :**

- **Description : ROR (Rotate Right) Rotation à droite.**
- **Décaler l'opérande Op d'un bit vers la droite.**
- **Une copie du bit le plus à droite passe vers le Carry Flag (CF) et une copie vers le bit le plus à gauche.**
- **Exemples :** **ROR DX, 1** **MOV CL, 3**
  ROR byte [9A00], 1 **ROR BX, CL**
- **Combinaisons possibles : Les même combinaisons que celles de ROL.**
- **Combinaisons interdites : Les même combinaisons que celles de ROL.**
- **Les indicateurs affectés : l'instruction ROR modifie les indicateurs O, S, Z, A, P, C du PSW.**

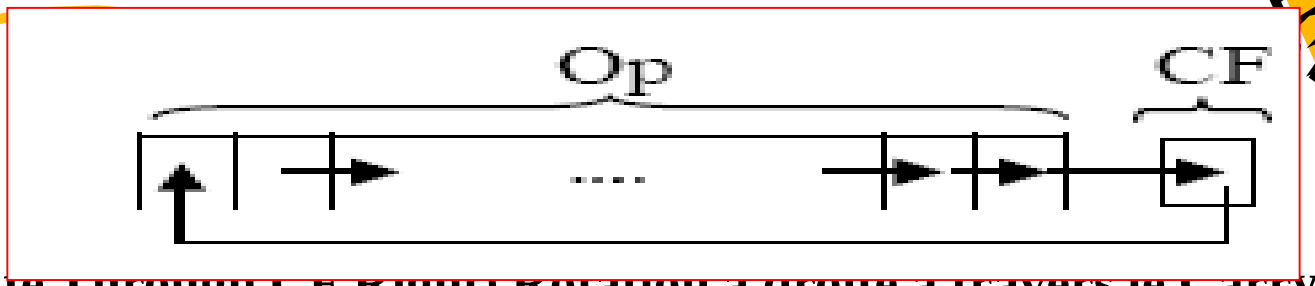
Introduction au langage machine du processeur 8086

- **Instruction RCL :**
- **Syntaxe : RCL Op, 1;**
- **Interprétation:**

- **Description : RCL (Rotate**
- **Effectue une rotation de l'opérande Op d'un bit vers la gauche. Le bit le plus à gauche passe vers le Carry Flag (CF) et l'ancien contenu du CF passe vers le bit le plus à droite.**
- **Remarque : Pour faire une rotation de plusieurs bits, on utilise le registre CL pour indiquer le nombre de bits de rotation.**
- **Exemples :** **RCL DX, 1** **MOV CL, 3**
  RCL byte [9A00], 1 RCL BX, CL
- **Combinaisons possibles : Les même combinaisons que celles de ROL.**
- **Combinaisons interdites : Les même combinaisons que celles de ROL.**
- **Les indicateurs affectés : l'instruction RCL modifie les indicateurs O, S, Z, A, P, C du PSW.**

Introduction au langage machine du processeur 8086

- **Instruction RCR :**
- **Syntaxe : RCR Op, 1;**
- **Interprétation:**



- **Description : RCL (Rotate Through CF Right) Rotation à droite à travers le Carry.**
- **Effectue une rotation de l'opérande Op d'un bit vers la droite. Le bit le plus à droite passe vers le Carry Flag (CF) et l'ancien contenu du CF passe vers le bit le plus à gauche.**
- **Remarque : Pour faire une rotation de plusieurs bits, on utilise le registre CL pour indiquer le nombre de bits de rotation.**
- **Exemples :**

RCR DX, 1	**MOV CL, 3**
RCR byte [9A00], 1	RCR BX, CL

- **Combinaisons possibles : Les même combinaisons que celles de ROL.**
- **Combinaisons interdites : Les même combinaisons que celles de ROL.**
- **Les indicateurs affectés : l'instruction RCR modifie les indicateurs O, S, Z, A, P, C du PSW.**

Introduction au langa ge machine du processeur 8086

❑Les instructions de branchement

Deux types de branchement sont possibles :

- Branchements inconditionnels

- Branchements conditionnels