MICROPROCESSEUR 16 BITS (LE 8086 DE INTEL)

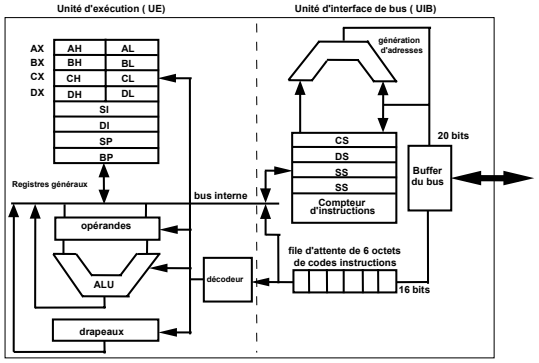
Objectifs

- Architecture générale
- Cycle d'accès au bus
- Organisation de la mémoire
- Modes d'adressages

Caractéristiques générales du 8086

- Microprocesseur 16 bits avec une architecture issue de celle du 8080.
- Il présente une capacité d'adressage étendue (24 modes).
- Jeu d'instructions très puissant (multiplication et division câblées).
- Il utilise pour l'adressage des segments spécialisés (transportabilité).
- Il peut fonctionner soit en mode minimum, soit en mode maximum.
- La mémoire de un Mo est vu comme un bloc de 512 Kilo-mots.
- les bus d'adresses et de données sont multiplexés.
- L'espace mémoires des E/S est séparé du reste de la mémoire.
- Il est totalement compatible avec le microprocesseur 8088.
- Il présente 1 entrée d'interruption logicielle et 2 entrées matérielles.
- Il dispose de circuits d'interface ou de traitements spécialisés.
- Peut reprendre les circuits périphériques des microprocesseurs 8 bits.

Architecture interne du 8086



E.U. et B.I.U.?

La première machine: L'**EU** (Execution Unit, unité d'exécution) regroupe les fonctions de traitement (unité arithmétique et logique), les registres généraux, le décodeur d'instruction et le microcode associé. Elle lit les codes opération dans une file d'attente et les exécute. La file d'attente est alimentée par le **BIU** (Bus Interface Unit, unité d'interface de bus) qui constitue la deuxième machine.

Le travail principal de **l'UIB**, est de chercher les instructions en mémoire et de les mettre dans la file d'attente. Ce travail est rompu dans deux cas particuliers.

Le premier cas est celui où l'UE, exécute une instruction nécessitant un accès mémoire. Elle demande alors à l'IUB de lire ou d'écrire une valeur en mémoire.

Le second cas est celui où l'UE rencontre un déroutement de programme (jmp / call ...). Le contenu de la file d'attente n'est plus à jour et l'UE doit demander une réinitialisation à partir de l'adresse de branchement.

La file d'attente des instructions a une taille de 6 octets, l'accès se fait toujours par mots de 16 bits sauf dans le seul cas particulier où l'adresse de réinitialisation est impaire.

Le but recherché est la **parallélisation** des deux tâches d'acquisition et d'exécution des instructions de manière à optimiser l'occupation du bus . Cette distinction est justifiée par le fait que ces deux blocs travaillent en partie indépendamment .

Lorsque l'EU a besoin d'accéder à une donnée en mémoire , il demande le transfert au BIU en nommant le segment concerné , et en fournissant le déplacement dans le segment. Le BIU **calcule** l'adresse physique sur 20 bits et réalise le transfert demandé .

Les instructions exécutées par l'EU sont issues d'une file d'attente de six octets d'instructions pour le 8086. Le BIU présente les octets d'instructions à l'entrée de la file d'attente dès que les deux conditions suivantes sont remplies :

- L'EU ne demande pas de transfert de données; cet évènement est prioritaire sur la lecture des instructions.
- Il existe des places libres dans la file d'attente qui peuvent être remplies par au moins un cycle de lecture du bus . Le 8086 a un bus de données de 16 bits: une lecture des octets d'instructions se fait dès qu'il y a au moins deux places libres .

En résumé : pendant que l'EU ne demande pas de transfert avec la mémoire , le BIU occupe ce temps libre du bus pour amener les instructions dans la file d'attente. Quand la file d'attente est vide , l'EU attend que le BIU place une instruction dans la file d'attente. Ceci arrive lorsque l'EU effectue un saut dans le programme : le contenu de la file d'attente est alors effacé (« flush » ou vidage)

Cela peut également arriver, lorsque l'exécution d'une suite d'instructions est plus rapide que le remplissage de la file d'attente par les cycles lecture du bus .

Les registres du 8086

AH	AL	ACCUMULATEUR
BH	BL	BASE
CH	CL	COMPTAGE
DH	DL	DONNEES

SP	POINTEUR DE PILE
BP	POINTEUR DE BASE
SI	INDEX DE SOURCE
DI	INDEX DE DESTINATION

IP	POINTEUR D'INSTRUCTION

Indicateurs de commande : TF | DF | IF

Indicateurs d'état : OF | SF | ZF | AF | PF | CF

CF : RETENUE
PF : PARITE
AF : RETENUE AUXILIAIRE
ZF : ZERO

SF : SIGNE
OF : DEBORDEMENT
IF : VALIDATION D'INTERRUPTION
DF : DIRECTION
TF : PIEGE

CS	SEGMENT DE CODE
DS	SEGMENT DE DONNEES
SS	SEGMENT DE PILE
ES	SEGMENT SUPPLEMENTAIRE

Les registres de travail :
Ce sont les registres avec lesquels peuvent être effectuées les additions , soustractions et opérations logiques , la deuxième opérande étant obtenue par l'un des modes d'adressages du microprocesseur . Chacun de ces registres a des fonctions particulières :

Les registres de base et d'index : BX , BP , SI , DI
Le contenu de ces registres peut être utilisé comme adresse mémoire (voir modes d'adressage du 8086) . Les deux index permettent la gestion de suite de mots; il s'agit de l'index de destination (DI) et de l'index de source (SI) . Le registre BP (base pointer) sert à la génération des adresses des données .

Le pointeur de pile : SP
Le 8086/8088 gère une pile utilisée pour les sous programmes et servant à sauvegarder les données de l'utilisateur.

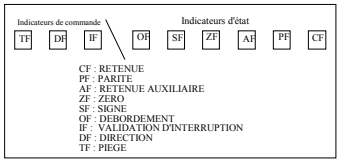
Les registres généraux : AX , BX , CX , DX
Ils sont au nombre de 4 et peuvent travailler par moitié (8 bits). Ils ont pour appellation :

- **L'accumulateur** : AX composé de AH et de AL. Les multiplications, les divisions, les traitements de chaînes, les transferts entrées-sorties, la conversion en BCD du résultat d'une opération arithmétique se font tous par son intermédiaire .
- **Base** : BX composé de BH et de BL. Ce registre sert à l'adressage en mémoire-données.
- **Le compteur** : CX composé de CH et de CL, est utilisé en compteur lors des instructions de boucles , dont les traitements de chaînes. Le registre CL est utilisé en compteur lors des instructions de décalages et de rotations multiples (sur plusieurs bits) .
- **Le registre de données** : DX composé de DH et de DL. Ce registre à des utilisations particulières, par exemples :
  * extension 32 bits pour les multiplications et les divisions de mots .
  * reste d'une division de mots (AX contient le quotient) .
  * en registre d'adresses d'entrées sorties .

Les registres de contrôle du programme :

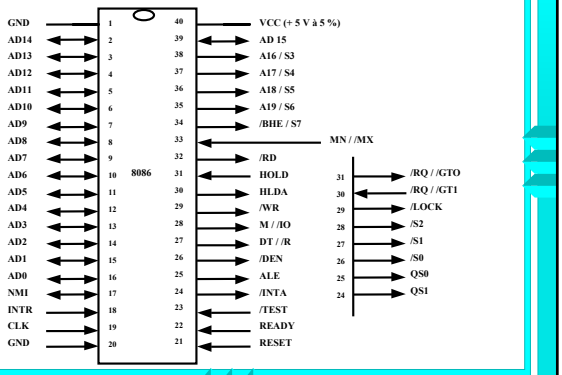
- Le pointeur d'instruction "IP" : Ce registre est mis à jour par le BIU afin qu'il pointe vers l'adresse de l'instruction suivante. Les programmes n'ont pas d'accès direct au pointeur d'instruction, mais au cours de l'exécution d'un programme, ce registre peut être modifié ou bien sauvegardé, puis restauré par la pile.

- Le registre d'état "FLAGS" : Le 8086 dispose de neuf indicateurs d'état à un bits, (voir figure suivante). (Les indicateurs TF, DF et IF ont un usage particulier). Les indicateurs d'état sont mis à jour par l'EU suite au résultat d'une opération arithmétique ou logique.



CF (carry flag) : indicateur de retenue .
PF (parity flag) : indicateur de parité .
AF (auxiliary flag) : indicateur de retenue auxiliaire . La retenue auxiliaire est la retenue générée par les 4 bits de poids faible du résultat d'une opération arithmétique . Elle permet de coder le résultat sous forme BCD .
ZF (zéro flag) : indicateur de zéro .
SF (sign flag) : indicateur de signe.
TF (trap flag) :indicateur de mode pas à pas . Si TF=1 une interruption est générée après exécution de chaque instruction du programme . Il permet la mise au point du logiciel sans gestion externe. Chaque instruction est interrompue par une exception de type pas à pas afin d'aller exécuter , à la fin de chaque instruction un sous programme d'aide à la mise au point (*debug exception*). Ce bit ne peut être modifié que par l'intermédiaire de la pile . Lorsqu'on entre dans le programme de pas à pas, le registre des indicateurs est sauvegardé et le bit TF est automatiquement remis à zéro afin d'empêcher le programme pas à pas d'être effectué lui même en mode pas à pas. Au retour, le registre des indicateurs est restitué et ainsi le bit TF se retrouve de nouveau positionné.
IF (interrupt flag) : indicateur d'autorisation d'interruption . Cet indicateur autorise (IF=1) ou interdit (IF=0) les interruptions externes demandées sur la broche INTR du circuit .
DF (direction flag) : indicateur de direction . C'est le sens de progression des registres d'index SI et DI lors des instructions de traitements de chaînes . DF=1 adresses décroissantes, DF=0 adresses croissantes
OF (overflow flag) : Le flag d'overflow est nécessaire en arithmétique signée (complément A 2), dans ce cas le bit de plus fort poids (le huitième ou le seizième) est le bit de signe et OF indique un dépassement de calculs.

Signaux du 8086



Les lignes adresses et données du 8086 sont **multiplexées** de manière à économiser le nombre de broches. Ce bus peut être utilisé directement en tant que bus local sur des petits systèmes.

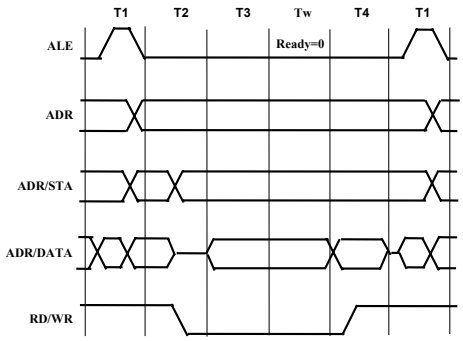
En règle générale le bus sera démultiplexé pour fournir des bus de données et d'adresses indépendants. Ceci est réalisé par de simples « latchs » commandés par le signal **ALE** (adress Latch Enable). De plus cela offre l'intérêt de bufferiser les lignes adresses et données (la sortance du 8086 étant limitée). Le signal ALE est généré directement par le 8086 en mode minimum , et par son contrôleur de bus 8288 en mode maximum.

Chaque cycle d'accès au bus comporte 4 périodes d'horloge (T1 à T4) auxquelles peuvent s'ajouter des périodes d'attente Tw (wait states) insérées entre T3 et T4. La figure suivante donne le cycle de base du 8086.

Pendant T1 , le processeur émet l'adresse sur le bus. Les transferts de données ont lieu pendant les cycles T3 et T4. Le cycle T2 permet essentiellement les changements de direction de bus pour les opérations de lecture. Des états Wait sont rajoutés entre T3 et T4 tant que le signal READY est à 0 .

La régularité des accès au bus est due à l'indépendance entre BIU et EU. Le processeur utilise 3 cycles pour son organisation interne; ainsi le bus n'est pas constamment utilisé par le 8086. Les périodes situées entre deux cycles d'accès au bus sont appelées inactives ou idle.

Cycle d 'accès au bus système



Modes de fonctionnement du 8086

Mode minimum :

le 8086 génère lui même les 3 bus, données, adresses et contrôle. Le bus de contrôle est un bus simplifié convenant à un environnement monoprocesseur. Ce mode ne permet pas l'utilisation de co-processeurs ou de processeurs spécialisés. Il n'est pas adapté à des architectures multiprocesseurs.

Mode maximum :

le 8086 génère les signaux d'état. Le contrôleur de bus décode ces signaux pour générer un bus de contrôle compatible avec les spécifications multibus. Les signaux S0, S1 et S2 sont utilisés pour identifier le type d'accès. S3 et S4 indiquent quel registre segment est utilisé pour la génération de l'adresse physique. S5 est l'image du bit d'interruption, S6 est toujours à 0 et S7 est un bit réservé.

Mode maximum / segment

S4	S3	Registre de segment concerné
0	0	ES (Extra segment)
0	1	SS (Stack segment)
1	0	CS (Code segment) ou transfert avec les entrées-sorties, ou lecture d'un vecteur d'interruption.
1	1	DS (Data segment)

Mode maximum / bus de contrôle

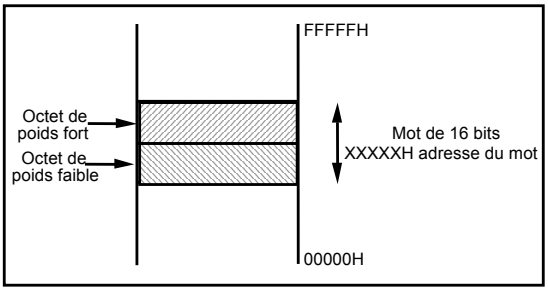
/S2	/S1	/S0	Action du microprocesseur	Sortie du 8288 correspondante
0	0	0	Acquittement d'interruption masquable	/INTA
0	0	1	Lecture d'une donnée entrées-sorties	/IORC
0	1	0	Ecriture d'une donnée entrées-sorties	/IOWC
0	1	1	Arrêt par une instruction HLT	aucune
1	0	0	Lecture d'instruction	/MRDC
1	0	1	Lecture d'une donnée mémoire	/MRDC
1	1	0	Ecriture d'une donnée mémoire	/MWTC
1	1	1	inactif	aucune

Mode maximum / file d 'instruction

QS1	QS0	Etat de la file d'attente ou action de l'unité d'exécution.
0	0	Etat inchangé. Aucun octet n'est lu dans la file d'attente.
0	1	Décodage du premier octet d'une instruction dans la file d'attente.
1	0	File d'attente vide, suite à un saut dans le programme, ou au décodage de tous les octets s'y trouvant, avant que l'unité d'interface bus ne lise les instructions suivantes.
1	1	Décodage de l'octet suivant d'une instruction (autre que le premier).

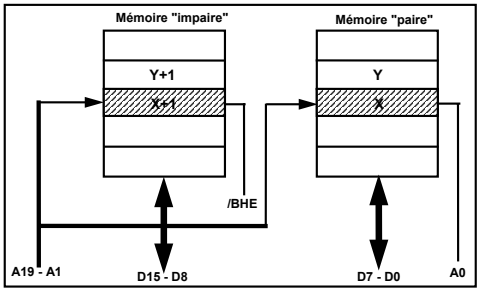
Organisation logique de la mémoire

La mémoire est organisée en 512 K-mots de 16 bits, rangés avec l'octet de poids fort à l'adresse la plus haute de la manière suivante.



Organisation physique de la mémoire

Elle est divisée physiquement en deux bancs de 512 K-octets : la mémoire paire (ou faible) et la mémoire impaire (ou fort).

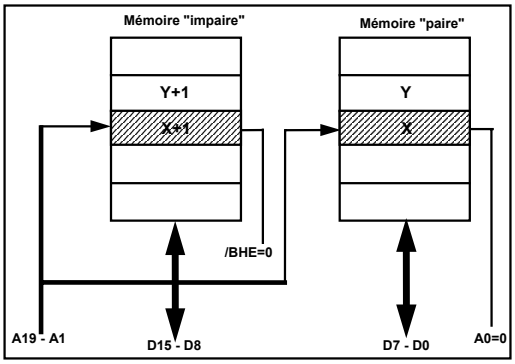


Accès à la mémoire

Pour un accès par octet un seul des bancs est validé et pour un accès par mot, les deux bancs sont utilisés simultanément. Un seul accès ou successivement deux accès suivant que l'adresse du mot est paire ou impaire. Les bits A0 et BHE sont utilisés pour de telles opérations.

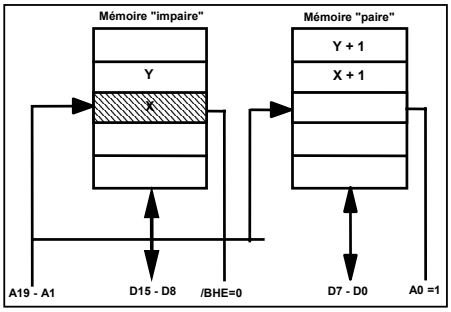
/BHE	A0	Donnée sélectionnée.
0	0	Le mot de 16 bits (adresse paire).
0	1	L'octet sur le bus D8-D15 (adresse impaire).
1	0	L'octet sur le bus D0-D7 (adresse paire).
1	1	aucune.

Accès à 16 bits à une adresse paire



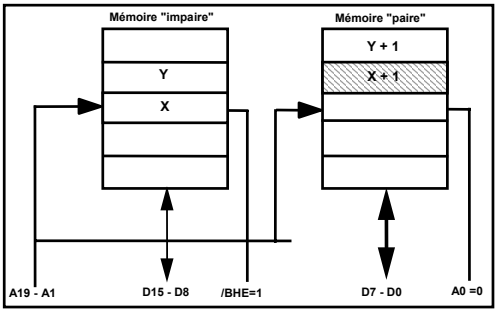
Accès à 16 bits à une adresse impaire

PREMIER TEMPS

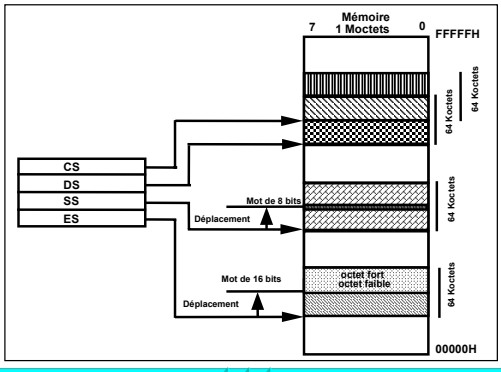


Accès à 16 bits à une adresse impaire

DEUXIEME TEMPS



Les mécanismes d 'adressages



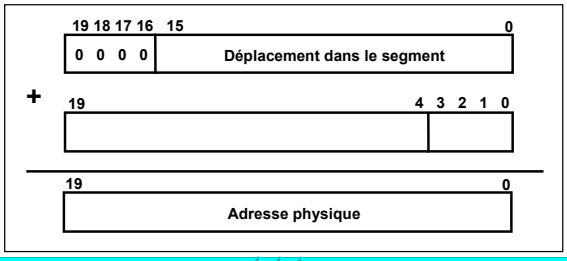
Remarque

L'accès au 32 K-mots d'entrées-sorties n'est référencé dans aucun segment. Les adresses physiques sont celles spécifiées directement par le programmeur.

Calcul de l'adresse

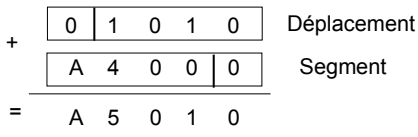
L'établissement de l'adresse physique sur 20 bits est obtenu par la formule suivante :

Adresse = 16 X registre de segment + déplacement



Exemple numérique :

registre de segment = A400H et déplacement 1010H. L'adresse physique sera égale à A5010.



Les modes d 'adressages

Le code exécutable d'un programme est toujours pointés par le contenu du registre IP par rapport au contenu du registre CS (CS:IP).

Les données stockées en pile de sauvegarde sont toujours pointées par le contenu du registre SP par rapport au contenu du registre SS (SS:SP).

Les positions mémoires représentant des données sont pointées par une adresse effective AE par rapport au contenu d'un segment qui dépend de la composition de l'adresse effective.

Les données à manipuler peuvent être dans des registres, dans la mémoire, dans la pile ou spécifiées immédiatement dans l'instruction (modes d'adressage des données).

Les modes d 'adressages

1) Adressage des registres

Concerne tout transfert ou toute opération, entre deux registres de même taille, ou sur un registre.

MOV SI , CX (CX) vers (SI) (16 bits)
ADD DH,AL Transfert de (DH) + (AL) vers (DH) (8 bits)

2) Adressage de la mémoire

Il y a 24 possibilités en tout. C'est ce qui permet le calcul du déplacement dans le segment (encore nommé AE) et le segment concerné par défaut par le mode d'adressage.

Adressage direct

Le déplacement est donné directement par le programme , et se fait dans le segment DS (par défaut) .

Déplacement = mot de 16 bits .

Soit le registre de segment DS = 1200H et le déplacement = 0081H (= VARIABLE). l'instruction :

MOV BP , VARIABLE

réalise le transfert de VARIABLE (dans le segment DS) vers (BP).

Adressage immédiat

La donnée opérande d'une opération avec un registre ou une mémoire est directement donnée par le programme. Cette donnée peut être un octet ou un mot, le registre ou la mémoire destinataire étant de la même taille .

MOV WORD PTR[SI],1

réalise le chargement de ((SI) dans le segment DS) par 0001H.

(l'expression WORD PTR ou BYTE PTR indique la taille de l'emplacement mémoire considéré, quand elle n'est pas implicitement connue , comme dans cet exemple).

Les registres segments ne peuvent pas être chargés par adressage immédiat .

Adressage basé (registre de base : BX et BP)

L'adresse est déterminée par :

base BX : le contenu de BX, auquel est éventuellement ajouté un décalage sur 8 ou 16 bits **signé**, représente le déplacement dans le **segment DS** (par défaut) .

base BP : le contenu de BP, auquel est ajouté (toujours) un décalage sur 8 bits ou 16 bits **signé**, représente le déplacement dans le **segment SS** (par défaut) .

d'où les cinq possibilités :

déplacement = (BX)
déplacement = (BX) + octet de décalage
déplacement = (BX) + mot de décalage
déplacement = (BP) + octet de décalage
déplacement = (BP) + mot de décalage

MOV ES , TABLE [BP]

réalise le transfert de ((BP) + TABLE) vers (ES) dans le segment SS.

Les modes d'adressages avec index

Adressage indexé (registre d'index : SI et DI)

Le contenu du registre d'index (SI ou DI) , auquel est éventuellement ajouté un décalage sur 8 ou 16 bits **signé** , représente le déplacement dans le **segment DS** (par défaut), d'où les six possibilités suivantes :

déplacement = (SI)
déplacement = (SI) + octet de décalage
déplacement = (SI) + mot de décalage
déplacement = (DI)
déplacement = (DI) + octet de décalage
déplacement = (DI) + mot de décalage

MOV VECTEUR[SI] , AL

réalise le transfert de (AL) vers ((SI) + VECTEUR dans le segment DS).

Adressage basé indexé

Le déplacement est déterminé par la somme du contenu du registre BX ou BP et d'un des registres d'index SI ou DI, auquel est éventuellement ajouté un décalage sur 8 ou 16 bits signé. DS et SS sont pris par défaut.

Base BX : la somme des contenus du registre BX et d'un des registres d'index (SI ou DI), à laquelle est éventuellement ajouté un décalage sur 8 ou 16 bits signé, représente le déplacement dans le segment DS (par défaut) .

Base BP : la somme des contenus du registre BP et d'un des registres d'index (SI ou DI), à laquelle est éventuellement ajouté un décalage sur 8 ou 16 bits signé, représente le déplacement dans le segment SS (par défaut) .

D'où 12 les possibilités suivantes :

déplacement = (BX) + (SI), déplacement = (BX) + (SI) + octet de décalage

déplacement = (BX) + (SI) + mot de décalage, déplacement = (BX) + (DI)

déplacement = (BX) + (DI) + octet de décalage, déplacement = (BX) + (DI) + mot de décalage

déplacement = (BP) + (SI), déplacement = (BP) + (SI) + octet de décalage

déplacement = (BP) + (SI) + mot de décalage, déplacement = (BP) + (DI)

déplacement = (BP) + (DI) + octet de décalage, déplacement = (BP) + (DI) +mot de décalage

XCH SP , TABLE[BP][SI]

réalise le transfert de (SP) vers ((BP)+(SI) + TABLE dans le segment SS), et le transfert de ((BP)+(SI) + TABLE dans le segment SS) vers (SP).

REMARQUE : Dans tout ce qui précède, les segments concernés sont ceux par défaut . Le programmeur peut référencer les déplacements à d'autre segments en les citant explicitement dans les instructions. Les codages de ces dernières occupent alors un octet supplémentaire .

Exemple :

MOV AX , [BX] réalise (AX) <-- ((BX) dans le segment DS)

Le segment DS est concerné par défaut .

Alors que :

MOV AX , ES:[BX] réalise (AX) <-- ((BX) dans le segment ES)

Le segment ES est cité explicitement .

Les contenus des registres de segments (CS , DS , SS , ES) peuvent être transférés dans un registre de travail, ou en mémoire adressée par l'un des modes décrits. L'inverse n'est possible que sur DS, SS et ES. CS ne peut être modifié que par un saut inter segment du programme. Le tableau suivant donne une récapitulation des modes d'adressage cités.

Les modes d 'adressages

Type d'adressage	Obtention du déplacement dans le segment	Segment concerné par défaut
Direct	donné directement par le programme	DS
Basé	(BX) (BX) + décalage (octet ou mot) (BP) + décalage (octet ou mot)	DS DS SS
Indexé	(SI) (SI) + décalage (octet ou mot) (DI) (DI) + décalage (octet ou mot)	DS DS DS DS
Basé indexé	(BX) + (SI) (BX) + (SI) + décalage (octet ou mot) (BX) + (DI) (BX) + (DI) + décalage (octet ou mot) (BP) + (SI) (BP) + (SI) + décalage (octet ou mot) (BP) + (DI) (BP) + (DI) + décalage (octet ou mot)	DS DS DS DS SS SS SS SS

L 'adressage des entrées-sorties

Les entrées sorties sont situées dans un espace d'adressage indépendant de la mémoire. Iles ne font l'objet d'aucune segmentation. Deux possibilités sont utilisées.

Adressage direct :

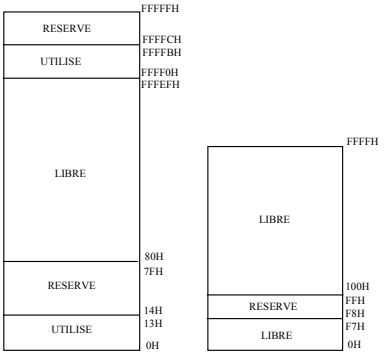
l'adresse du port est donnée directement par le programme. On n'accède qu'aux 256 premiers octets, ou aux 128 premiers mots d'entrées sorties.

```
IN AX , 0A8H
OUT 0A8H , AL
```

Adressage par registre :

l'adresse du port est contenue dans le registre DX. On accède alors aux 64 Ko, 32 K-mots, d'entrées sorties .

```
OUT DX , AL
IN AX , DX
```



L'adressage des sauts de programme

Les instructions du programme sont pointées par le pointeur d'instruction IP dans le segment CS .

Deux catégories de saut sont à distinguer :
- saut à l'intérieur du même segment , dit **intra-segment** : CS n'est pas modifié .
- saut à l'extérieur du segment , dit **inter-segment** : CS est modifié.

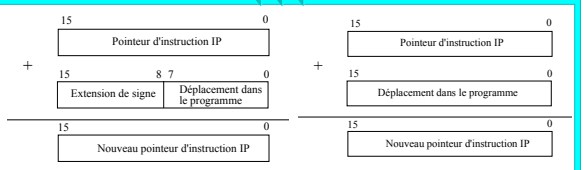
L 'adressage des sauts de programme

Saut intra-segment :

Saut conditionnel : L'adressage est dit relatif au pointeur d'instruction : cela consiste à ajouter au pointeur d'instruction IP un nombre de 8 bits , dont le signe est étendu sur 16 bits. Ce nombre est le déplacement dans le programme (pas dans le segment). Ce genre est limité à :
- 127 octets en avant ,
- 128 octets en arrière .

Exemple :
- **JBE INF_EGAL**
- **JNA NON_SUP**
- **JE EQUAL**

Saut inconditionnel : Le saut peut aussi être inconditionnel. Dans ce cas, l'adressage est soit relatif, soit indirect. Dans le cas du relatif, le nombre ajouté au pointeur d'instruction est un octet, ou un mot, qui permet alors de couvrir tout le segment.



JMP SHORT NEAR-LABEL
(+127 à -128 octets)

JMP NEAR-LABEL
(+32767 à -32768 octets)

Pour l'indirect, le pointeur d'instruction est chargé par un nombre de 16 bits. Ce nombre est soit le contenu de l'un des registres de travail 16 bits, soit le contenu d'un mot mémoire adressé par l'un des modes d'adressage décrit précédemment.

JMP AX

JMP TABLE[BX]

L 'adressage des sauts de programme (2)

Saut inter-segment :

Ce saut est exclusivement inconditionnel. L'adressage doit fournir les valeurs à charger dans le pointeur d'instruction IP et dans le registre de segment CS. Il peut être direct ou indirect.

Adressage direct

Le déplacement (destiné à IP) et le segment (destiné à CS) sont donnés directement par le programme .

- **JMP addr; addr sur 4 octets.**
- **JMP FAR_LABEL**

Adressage indirect

Le déplacement et les segments sont les contenus de deux mots mémoires consécutifs . L'adressage se fait par l'un des modes décrits pour l'adressage de la mémoire.

- **JMP DWORD PTR [BP][DI]**

La pile

La pile est implantée dans l'espace mémoire . Elle est pointée par le registre SP dans le segment SS exclusivement .

Les données à empiler ou à dépiler sont uniquement des mots . La provenance ou la destination est un registre ou un emplacement mémoire adressé par l'un des modes décrit précédemment . La pile se remplie par adresses décroissantes et se vide par adresses croissantes. L'instruction suivante : **PUSH TABLE[DI]**, réalise :

(SP) <-- (SP)-2 , ((SP) dans le seg SS)<-- ((DI)+TABLE dans le seg DS)

POP TABLE[DI] (SP = SP + 2), réalise l'opération de récupération

La pile sera traitée avec plus de détails dans le chapitre réservé à la " PROGRAMMATION DU 8086".

Les sous-programmes

Les sous programmes

Les possibilités d'adressage des sous programmes sont les mêmes que celles des sauts inconditionnels intra-segments ou inter-segments .

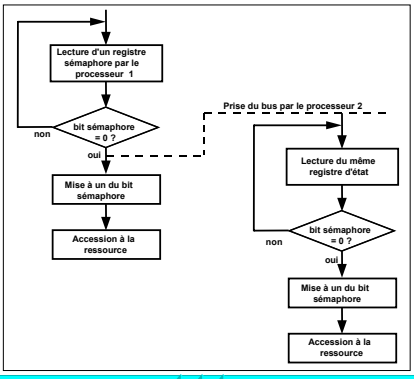
Lors de l'écriture d'un sous programme, le programmeur doit être attentif au point suivant :

- sous programme **intra-segment** : le pointeur d'instruction IP est placé dans la pile avant le saut. A la fin du sous programme l'instruction RET restaure IP .

- sous programme **inter segment** : le pointeur d'instruction IP, puis le registre de segment CS, sont placés dans la pile avant le saut . A la fin du sous programme, l'instruction RET, différente de la précédente bien qu'ayant le même nom, restaure IP et CS .

Le programmeur doit déclarer un sous programme **inter segment** ou **intra segment**, pour que l'assembleur puisse coder correctement l'instruction RET. (Voir chapitre Pragrammation du 8086).

Systèmes multiprocesseurs



Instructions du 8086

Transfert de données
- MOV
- POP
- OUT

Instructions arithmétiques
- ADD
- DIV
- MUL

Instructions logiques
- OR
- AND
- DEC

Décalages et rotations
- ROL
- RCL
- SAL

Instructions de branchement
- JMP
- CALL
- JCXZ

Manipulations de caractères
- REP MOVS
- REP MOVSB
- REP MOVSW

Instructions de commande
- INT N
- STI
- CLI

Remarque

Un ou plusieurs indicateurs d'état peuvent être affectés par l'exécution de ces instructions.