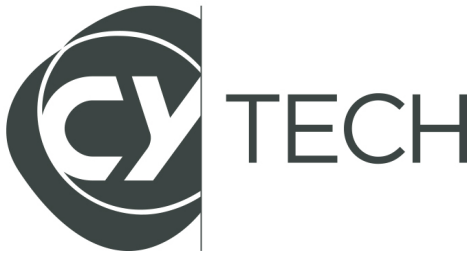
Architecture des Ordinateurs
Lecture 4: Assembleur

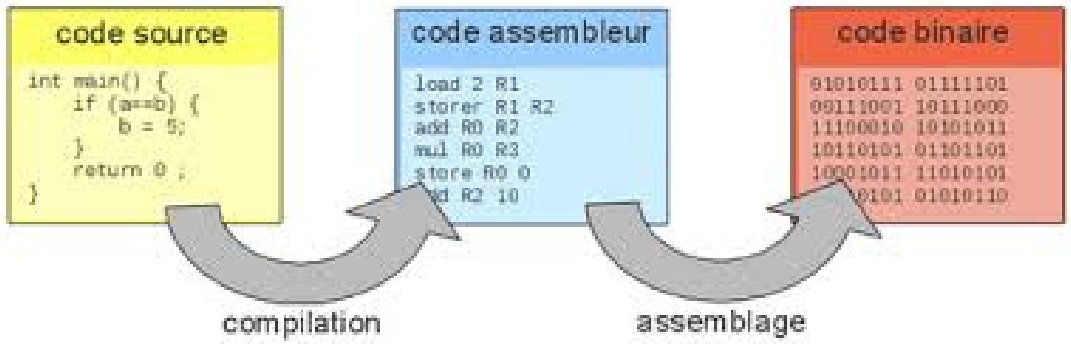
Prof. Yérali Gandica

CY-Tech Cergy Paris Université
2022



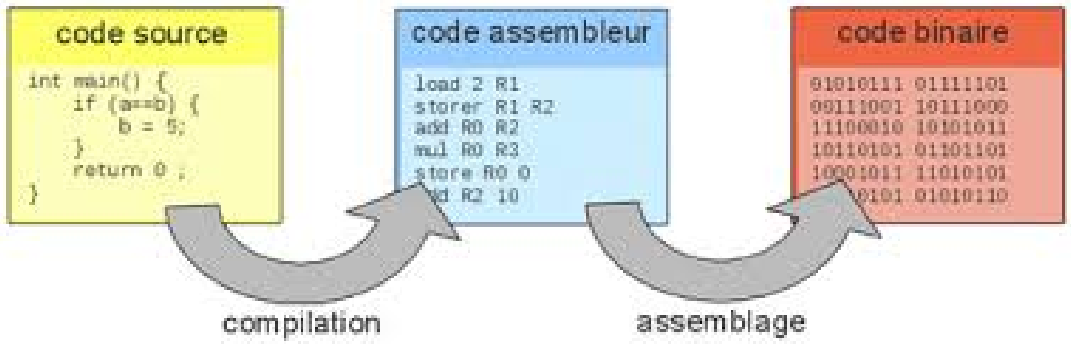
Langage d'assemblage

- Un langage d'assemblage ou langage assembleur est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- C'est le langage directement interprétable par le processeur.
- Il est défini par un ensemble d'instructions que le processeur exécute directement.
- Chaque famille de processeur utilise un jeu d'instructions différent.
- Ainsi le langage assembleur, représentation exacte du langage machine, est spécifique à chaque architecture de processeur.



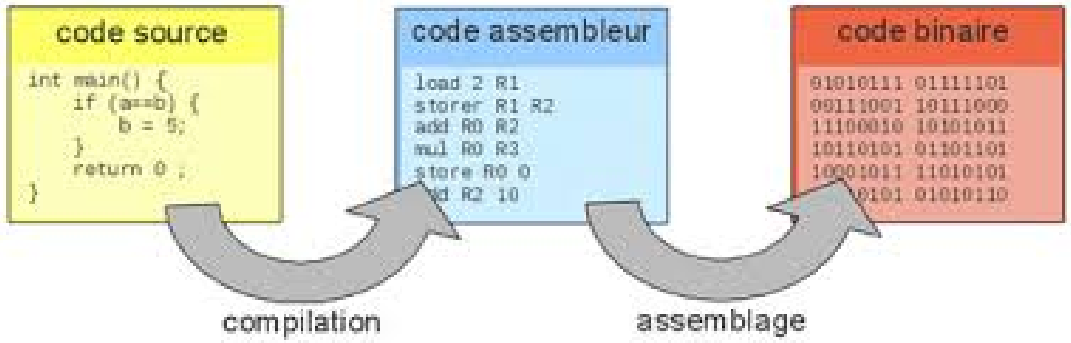
Langage d'assemblage

- Un langage d'assemblage ou langage assembleur est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- C'est le langage directement interprétable par le processeur.
- Il est défini par un ensemble d'instructions que le processeur exécute directement.
- Chaque famille de processeur utilise un jeu d'instructions différent.
- Ainsi le langage assembleur, représentation exacte du langage machine, est spécifique à chaque architecture de processeur.



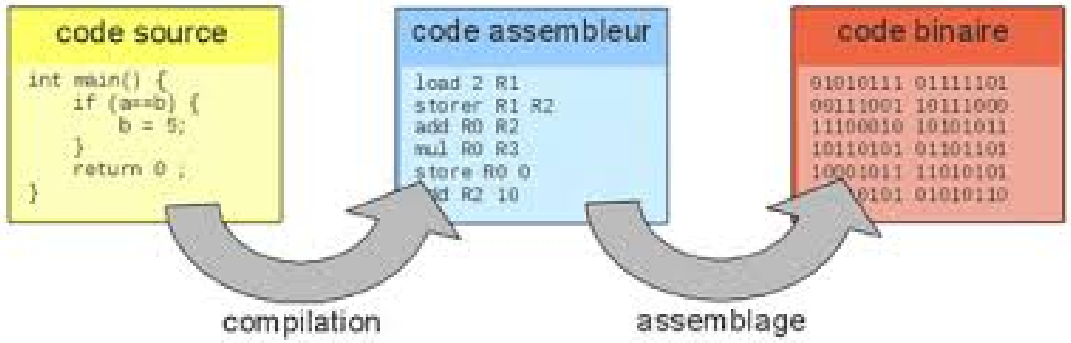
Langage d'assemblage

- Un langage d'assemblage ou langage assembleur est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- C'est le langage directement interprétable par le processeur.
- Il est défini par un ensemble d'instructions que le processeur exécute directement.
- Chaque famille de processeur utilise un jeu d'instructions différent.
- Ainsi le langage assembleur, représentation exacte du langage machine, est spécifique à chaque architecture de processeur.



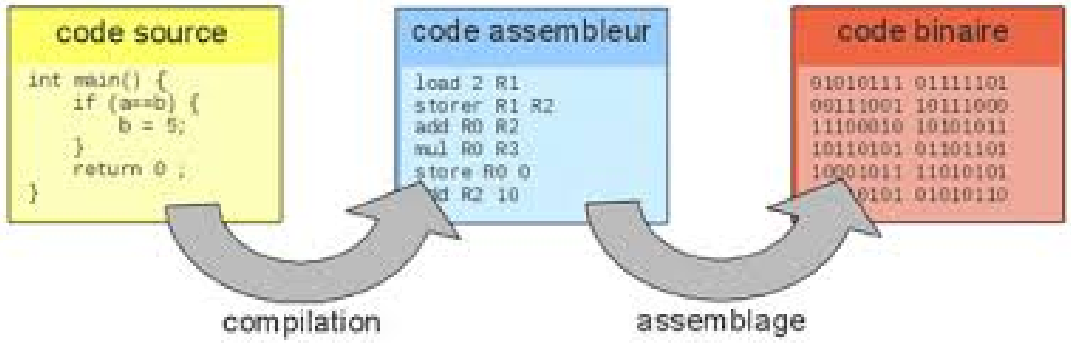
Langage d'assemblage

- Un langage d'assemblage ou langage assembleur est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- C'est le langage directement interprétable par le processeur.
- Il est défini par un ensemble d'instructions que le processeur exécute directement.
- Chaque famille de processeur utilise un jeu d'instructions différent.
- Ainsi le langage assembleur, représentation exacte du langage machine, est spécifique à chaque architecture de processeur.



Langage d'assemblage

- Un langage d'assemblage ou langage assembleur est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- C'est le langage directement interprétable par le processeur.
- Il est défini par un ensemble d'instructions que le processeur exécute directement.
- Chaque famille de processeur utilise un jeu d'instructions différent.
- Ainsi le langage assembleur, représentation exacte du langage machine, est spécifique à chaque architecture de processeur.



Langage d'assemblage

- Chaque instruction correspond à un nombre, composé de:
  - une partie codant l'opération à exécuter appelée opcode ou code opération
  - une partie pour les opérandes

- Le programme est représenté par une série de nombres qui sont conservés dans une sorte de mémoire informatique. Il y a quatre étapes que presque tous les processeurs d'architecture von Neumann utilisent dans leur fonctionnement : fetch, decode, execute, et writeback.

Langage d'assemblage

- Chaque instruction correspond à un nombre, composé de:
  - une partie codant l'opération à exécuter appelée opcode ou code opération
  - une partie pour les opérandes

- Le programme est représenté par une série de nombres qui sont conservés dans une sorte de mémoire informatique. Il y a quatre étapes que presque tous les processeurs d'architecture von Neumann utilisent dans leur fonctionnement : fetch, decode, execute, et writeback.

Execution d'une instruction

- **FETCH:** Recherche de l'instruction
- **DECODE:** Décodage de l'instruction
- **EXECUTE:** Exécution de l'opération
- **WRITEBACK:** Stockage du résultat

FETCH

La première étape, **FETCH (recherche)**, consiste à rechercher une instruction dans la mémoire vive de l'ordinateur.
L'emplacement dans la mémoire est déterminé par le compteur de programme, qui stocke l'adresse de la prochaine instruction dans la mémoire de programme.
Après qu'une instruction a été recherchée, le compteur de programme est incrémenté par la longueur du mot d'instruction.
L'instruction que le processeur recherche en mémoire est utilisée pour déterminer ce que le CPU doit faire.

DECODE

Dans l'étape **DECODE (décodage)**, l'instruction est découpée en plusieurs parties telles qu'elles puissent être utilisées par d'autres parties du processeur.
La façon dont la valeur de l'instruction est interprétée est définie par le jeu d'instructions du processeur. Souvent, une partie d'une instruction, appelée opcode (code d'opération), indique quelle opération est à faire, par exemple une addition.
Les parties restantes de l'instruction comportent habituellement les autres informations nécessaires à l'exécution de l'instruction comme par exemple des valeurs pour l'addition.

EXECUTE

Après les étapes de recherche et de décodage arrive l'étape **EXECUTE (exécution)** de l'instruction. Au cours de cette étape, différentes parties du processeur sont mises en relation pour réaliser l'opération souhaitée. Par exemple, pour une addition, l'unité arithmétique et logique (UAL) sera connectée à des entrées et des sorties.
Les entrées présentent les nombres à additionner et les sorties contiennent la somme finale. L'UAL contient le circuit électronique pour réaliser des opérations d'arithmétique et de logique simples sur les entrées (addition, opération sur les bits).
Si le résultat d'une addition est trop grand pour être codé par le processeur, un signal de débordement est positionné dans un registre d'état.

WRITEBACK

La dernière étape **WRITEBACK (écriture du résultat)**, écrit tout simplement les résultats de l'étape d'exécution en mémoire.
Très souvent, les résultats sont écrits dans un registre interne au processeur pour bénéficier de temps d'accès très courts pour les instructions suivantes.
Dans d'autres cas, les résultats sont écrits plus lentement dans des mémoires RAM, donc à moindre coût et acceptant des codages de nombres plus grands.

Assembleur

- Nous étudierons l'assembleur 80x86 (INTEL).
- Nous passerons en revue certains registres du processeur et le jeu d'instructions de l'assembleur 80x86.
- Intel 8086 est un microprocesseur 16 bits fabriqué par Intel à partir de 1978.
- Premier processeur de la famille x86 devenue l'architecture de processeur la plus répandue dans le monde des ordinateurs personnels, stations de travail et serveurs informatiques.

Assembleur

- Nous étudierons l'assembleur 80x86 (INTEL).
- Nous passerons en revue certains registres du processeur et le jeu d'instructions de l'assembleur 80x86.
- Intel 8086 est un microprocesseur 16 bits fabriqué par Intel à partir de 1978.
- Premier processeur de la famille x86 devenue l'architecture de processeur la plus répandue dans le monde des ordinateurs personnels, stations de travail et serveurs informatiques.

Assembleur

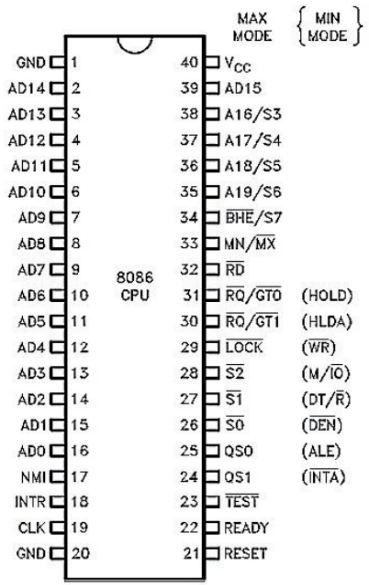
- Nous étudierons l'assembleur 80x86 (INTEL).
- Nous passerons en revue certains registres du processeur et le jeu d'instructions de l'assembleur 80x86.
- Intel 8086 est un microprocesseur 16 bits fabriqué par Intel à partir de 1978.
- Premier processeur de la famille x86 devenue l'architecture de processeur la plus répandue dans le monde des ordinateurs personnels, stations de travail et serveurs informatiques.

Assembleur

- Nous étudierons l'assembleur 80x86 (INTEL).
- Nous passerons en revue certains registres du processeur et le jeu d'instructions de l'assembleur 80x86.
- Intel 8086 est un microprocesseur 16 bits fabriqué par Intel à partir de 1978.
- Premier processeur de la famille x86 devenue l'architecture de processeur la plus répandue dans le monde des ordinateurs personnels, stations de travail et serveurs informatiques.

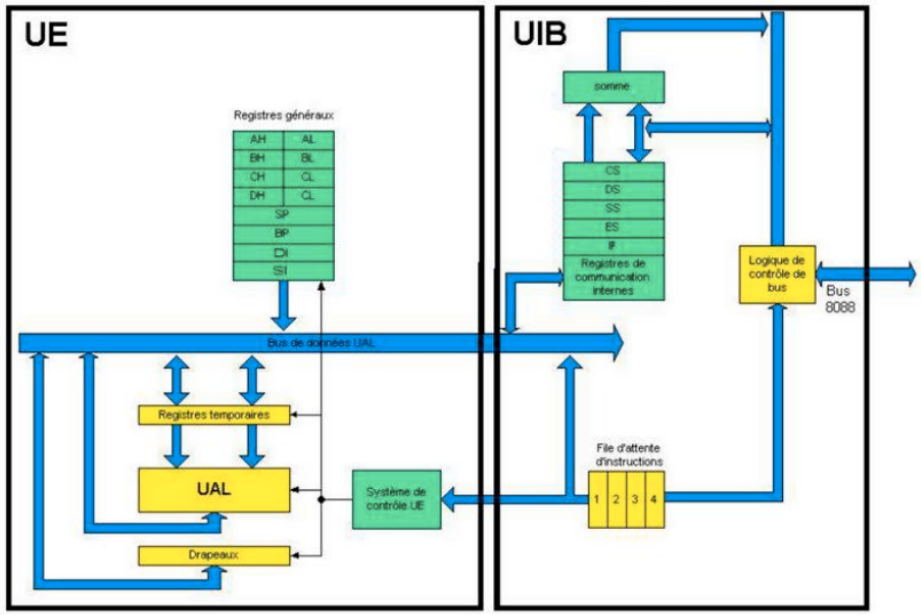
Processeur 8086

- Il se présente sous forme d'un boîtier de 40 broches alimenté par une alimentation unique de 5V.
- Il possède un bus multiplexé adresse/donnée de 20 bits.
- Le bus de donnée occupe 16 bits ce qui permet d'échanger des mots de 2 octets.
- Le bus d'adresse occupe 20 bits ce qui permet d'adresser 1 Méga octets (2 puiss 20).



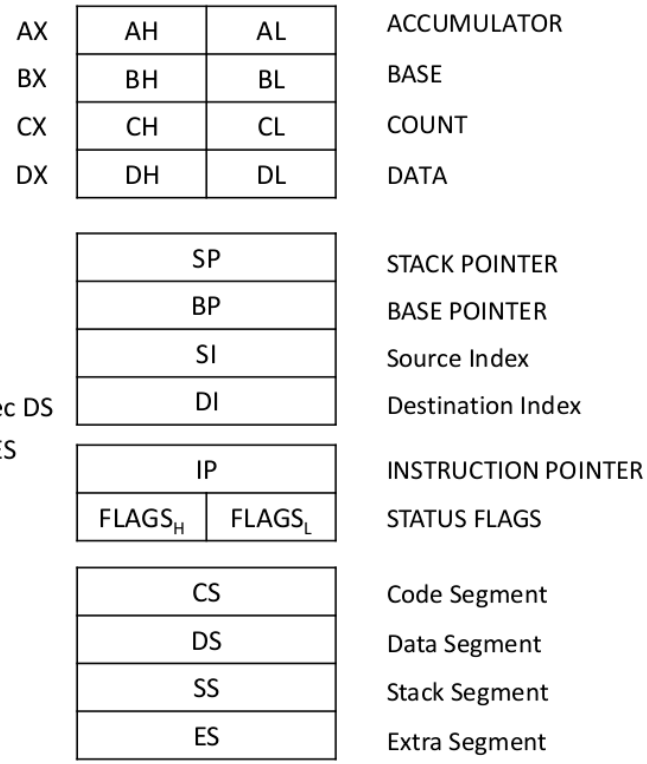
Processeur 8086

- Il possède Deux unités:
  - Unité Interface Bus (UIB) • Unité d'exécution (UE)
- Tous les registres sont de 16 bits, mais certains registres sont découpés en deux et on peut accéder séparément à la partie haute et à la partie basse.



Processeur 8086 - Registres

- **Registres généraux**
  - AX=(AH,AL), BX=(BH,BL), CX=(CH,CL), DX=(DH,DL)
  - **AX**= Accumulateur : Données d'I/O et operations arithmétiques
  - **BX**= Base: register d'adressage
  - **CX**= Compteur de boucles
  - **DX**= Data: addresses des ports pour les instructions
- **Registres pointeurs et d'index**
  - **SP** Pointeur de pile
  - **BP** pointeur de base : pointe sur la zone pile (FILO), offset avec SS
  - **SI** Source Index: pointe la mémoire, forme un offset (decalage) avec DS
  - **DI** Destination Index: pointe pointe la mémoire, offset avec DS ou ES
- **Registres de segments**
  - CS: Code Segment
  - SS: Stack Segment
  - DS:Data Segment
  - ES: Extra Segment
- **Pointeur d'instruction et register d'état**
  - Pointeur d'instruction IP
  - Registre d'état



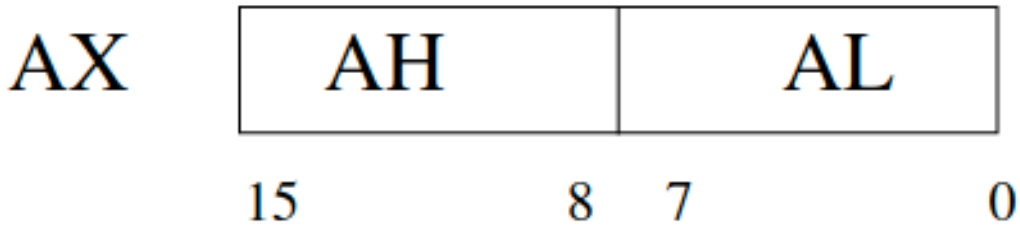
6

Processeur 8086 - Les registres généraux

Les registres généraux :ce sont des registres à usage général qui peuvent se décomposer en 2 registres de 8 bits chacun.

- AX : (registre accumulateur) registre d'usage général contenant des données. Il joue le rôle d'opérande implicite dans de nombreuses opérations : MUL, DIV, INC, . . . La partie de poids forts se nomme AH et celle de poids faibles AL (H pour high et L pour low). (Remarque : AX ne peut pas servir pour l'adressage).
- BX : (registre de base) registre d'usage général contenant des données. Comme il sert entre autre à pointer sur des adresses mémoire.
- CX : (registre de Comptage et calcul) registre d'usage général contenant des données. On peut l'utiliser dans certaines instruction comme compteur de répétition (loop). (Remarque : CX ne peut pas servir pour l'adressage).
- DX : (registre de données) registre d'usage général contenant des données. On peut l'utiliser Dans la multiplication et la division 16 bits, il sert comme extension au registre AX pour contenir un nombre 32 bits, (Remarque : DX ne peut pas servir pour l'adressage).

Tous les registres généraux:



Processeur 8086 - Les registres d'adressage (offset)

Les registres d'adressage (offset) : Ce sont des registres permettant l'adressage d'une opérande à l'intérieur d'un segment de 64 ko.

- BP (Base Pointer ou Pointeur de Base) sert de pointeur sur l'adresse mémoire correspondant à la base de la pile, et permet en fait d'atteindre n'importe quel élément de celle-ci : MOV AX, [BP+4]
- SP (Stack Pointer ou Pointeur de Pile) pointe sur le sommet de la pile. Son contenu est automatiquement géré par les instructions PUSH et POP d'empilage et de désempilage.
- SI (Source Index ou Registre d'index source) est souvent utilisé comme pointeur sur une adresse mémoire.
- DI (Destination Index ou Registre d'index destination) permet comme SI de pointer sur des adresses mémoire..

Processeur 8086 - Les registres de segment

Ce sont des registres combinés avec les registres d'adressage pour former les adresses absolues.

- CS (Code Segment) pointe sur le segment contenant le code du programme.
- DS (Data Segment) pointe sur l'adresse de début du segment qui contient les données.
- ES (Extra Segment) permet de pointer sur un segment supplémentaire défini par le programmeur.
- SS (Stack Segment) pointe sur le segment de la pile.

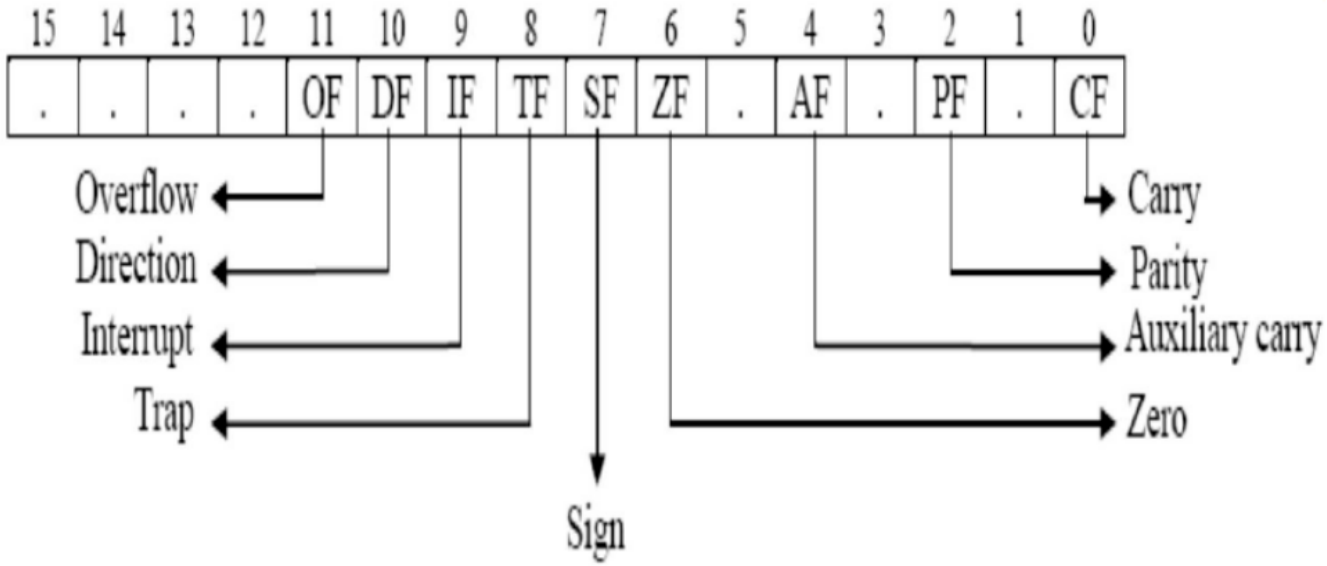
Processeur 8086 - Les registres indicateurs (Flags Register)

Le registre indicateurs (Flags Register) : il est utilisé pour stocker des états particulier du microprocesseur en cours de fonctionnement.

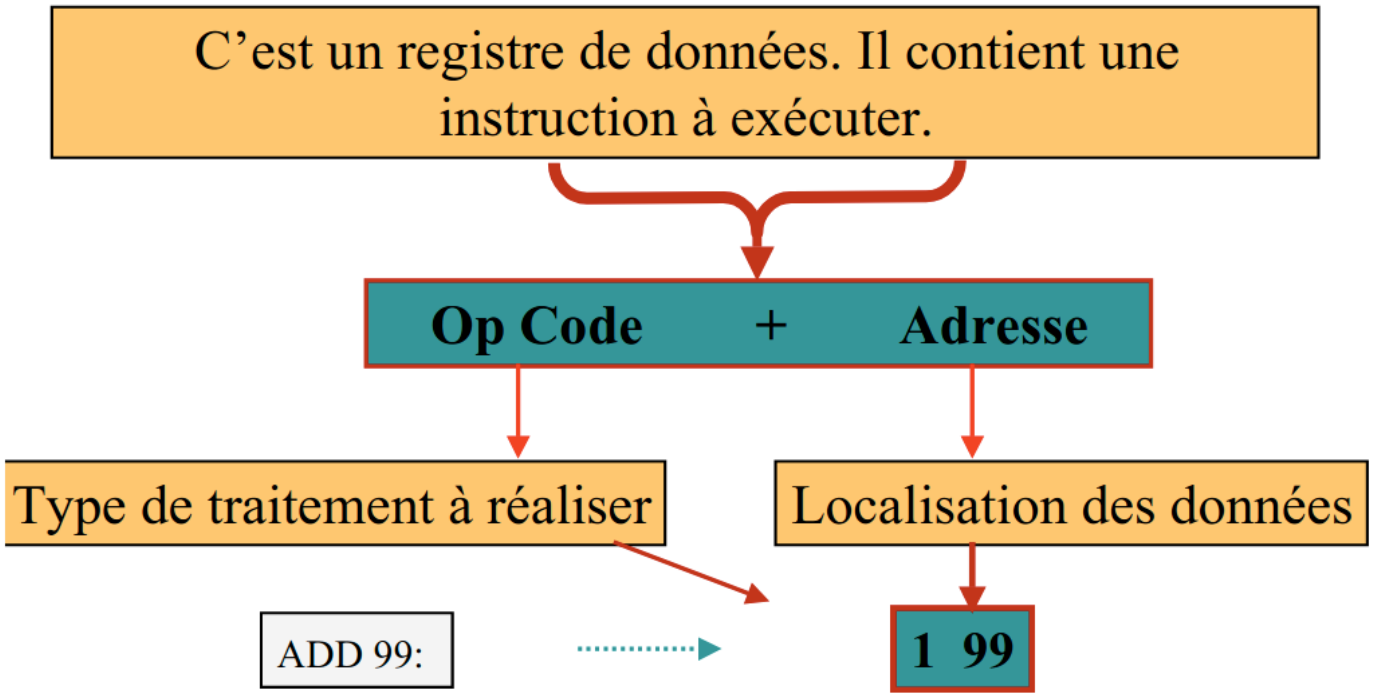
- CF : le bit de carry (retenue) est positionné pour indiquer si le calcul a engendré une retenue qui devra être reportée sur les calculs suivants.
- PF : le bit de parité. Il indique si les 8 bits de poids faible du résultat comportent un nombre pair .
- AF : le bit dit auxiliary carry (retenue auxiliaire) est positionné pour indiquer une retenue entre bits de poids faible et bits de poids forts d'un octet, d'un mot ou d'un double mot.
- ZF : le bit de zéro est positionné pour indiquer que le résultat du calcul est 0.
- SF : le bit de signe est positionné pour indiquer le signe du résultat (positif ou négatif) ;
- TF : le bit de Trap. Met le CPU en mode pas à pas pour faciliter la recherche des défauts d'exécution;
- IF : le bit d'Interruption. Il indique si le CPU autorise ou non la reconnaissance des interruptions (I = 0 $\rightarrow$Interruptions autorisées, I = 1 $\rightarrow$ Interruptions non autorisées).
- DF : bit de direction. Il fixe la direction de l'auto-inc/décrémentation de SI et DI lors des instruction de manipulation de chaînes (D = 0 $\rightarrow$Incrémentation des index, D = 1 $\rightarrow$ décrémentation des index).
- OF : le bit d'overflow est positionné pour indiquer s'il y a un débordement de capacité lors d'un calcul

Processeur 8086 - Les registres indicateurs (Flags Register)

Le registre indicateurs (Flags Register) : il est utilisé pour stocker des états particulier du microprocesseur en cours de fonctionnement.



Le registre d'instruction (IR)



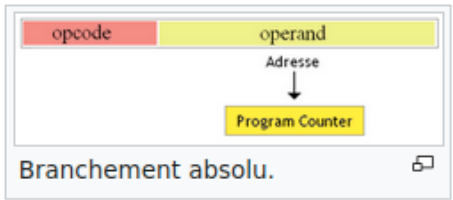
Mode d'adressage

Les modes d'adressage sont un aspect de l'architecture des processeurs et de leurs jeux d'instructions. Les modes d'adressage définis dans une architecture régissent la façon dont les instructions en langage machine identifient leurs opérandes. Un mode d'adressage spécifie la façon dont est calculée l'adresse mémoire effective d'un opérande à partir de valeurs contenues dans des registres et de constantes contenues dans l'instruction ou ailleurs dans la machine.

Modes d'adressage pour les branchements

L'adresse de la prochaine instruction à exécuter est contenue dans un registre spécial du processeur, appelé compteur ordinal (CO) (ou IP pour instruction pointer) ou compteur de programme (CP).
Les instructions de branchement (ou de saut) visent à modifier la valeur du compteur ordinal. Cette valeur est l'adresse effective de l'instruction de branchement ; elle est calculée différemment selon le mode d'adressage choisi.

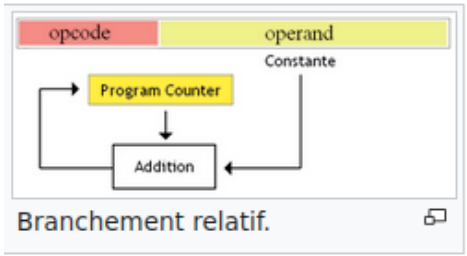
- **Adressage absolu**:l'adresse de destination est donnée dans l'instruction; le contenu du pointeur de programme est remplacé par l'adresse en question.



Les instructions de saut (ou de branchement) sont les instructions qui modifient la valeur du compteur de programme CP. Le compteur de programme représente l'adresse de la prochaine instruction à exécuter et est par défaut incrémenté par l'unité de contrôle à chaque cycle instruction. En modifiant le compteur de programme, une instruction de saut va ainsi modifier l'adresse de la prochaine instruction à exécuter.

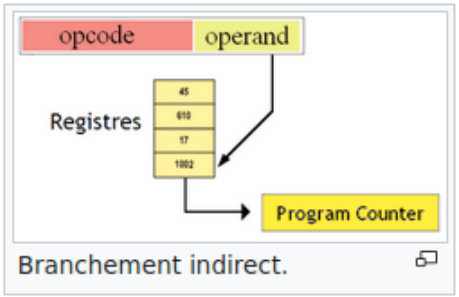
Modes d'adressage pour les branchements

- **Adressage relatif:**Comme de nombreux branchements s'effectuent vers des adresses mémoire proches de l'endroit où l'on se trouve au moment d'exécuter le branchement, on peut se contenter d'indiquer un décalage par rapport à l'adresse de la prochaine instruction. Par exemple si l'on exécute du code à l'adresse 0x42 et que l'on veut brancher 8 octets plus loin, on peut simplement indiquer +8 pour brancher à 0x4A au lieu d'indiquer cette adresse complète.



Branchement relatif.

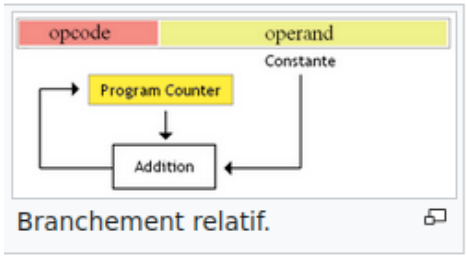
- **Adressage indirect**: Dans certains cas, l'adresse à laquelle brancher n'est pas connue à la compilation, ou alors celle-ci peut varier selon les circonstances. certains processeurs ont réglé ce problème en incorporant un mode d'adressage indirect pour le code.
  Avec ce mode d'adressage, l'adresse vers laquelle brancher est stockée dans un registre, incorporé dans la représentation binaire de l'instruction.



Branchement indirect.

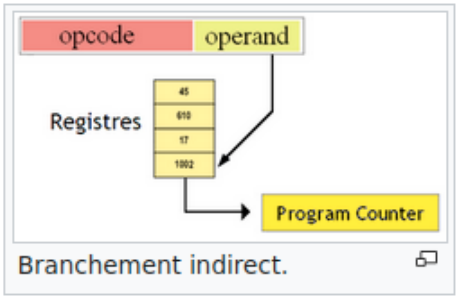
Modes d'adressage pour les branchements

- **Adressage relatif:**Comme de nombreux branchements s'effectuent vers des adresses mémoire proches de l'endroit où l'on se trouve au moment d'exécuter le branchement, on peut se contenter d'indiquer un décalage par rapport à l'adresse de la prochaine instruction. Par exemple si l'on exécute du code à l'adresse 0x42 et que l'on veut brancher 8 octets plus loin, on peut simplement indiquer +8 pour brancher à 0x4A au lieu d'indiquer cette adresse complète.



Branchement relatif.

- **Adressage indirect**: Dans certains cas, l'adresse à laquelle brancher n'est pas connue à la compilation, ou alors celle-ci peut varier selon les circonstances. certains processeurs ont réglé ce problème en incorporant un mode d'adressage indirect pour le code.
  Avec ce mode d'adressage, l'adresse vers laquelle brancher est stockée dans un registre, incorporé dans la représentation binaire de l'instruction.

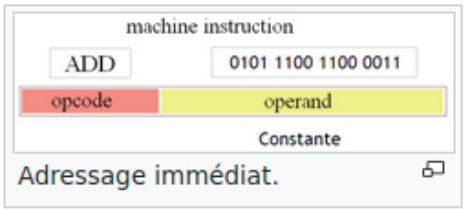


Branchement indirect.

Modes d'adressage pour les données

De nombreuses instructions font référence à des données se trouvant à différents endroits de la machine : registres, pile d'exécution, etc.

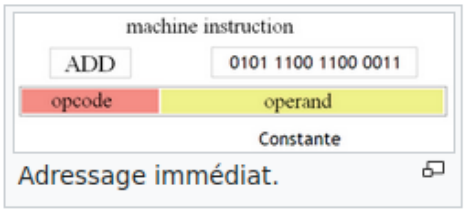
- **Adressage implicite**:Certaines opérations ne peuvent être réalisées que sur une donnée se trouvant en un endroit bien précis du processeur (par exemple, **l'accumulateur ou la pile**). Dans ce cas, il n'est pas nécessaire de spécifier l'adresse du registre en question et on parle d'adressage implicite.
  * **un accumulateur** est un registre spécial, incorporé dans certaines architectures de processeur, où les résultats intermédiaires de l'UAL (arithmetic logical unit), sont versés.
  Sans accumulateur il faudrait verser le résultat de l'UAL dans la mémoire centrale, puis le recharger pour effectuer l'opération suivante dont le résultat serait à son tour versé dans la mémoire centrale, etc.
  * **Une pile** (en anglais stack) est une structure de données fondée sur le principe: dernier arrivé, premier sorti.
- **Adressage immédiat**:la donnée est intégrée directement dans la représentation binaire de l'instruction. Ce mode d'adressage permet d'optimiser la gestion des constantes connues à la compilation.



Modes d'adressage pour les données

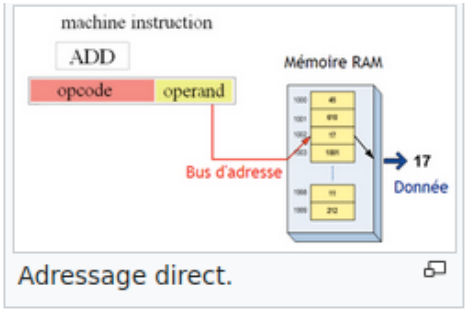
De nombreuses instructions font référence à des données se trouvant à différents endroits de la machine : registres, pile d'exécution, etc.

- **Adressage implicite**:Certaines opérations ne peuvent être réalisées que sur une donnée se trouvant en un endroit bien précis du processeur (par exemple, **l'accumulateur ou la pile**). Dans ce cas, il n'est pas nécessaire de spécifier l'adresse du registre en question et on parle d'adressage implicite.
  * **un accumulateur** est un registre spécial, incorporé dans certaines architectures de processeur, où les résultats intermédiaires de l'UAL (arithmetic logical unit), sont versés.
  Sans accumulateur il faudrait verser le résultat de l'UAL dans la mémoire centrale, puis le recharger pour effectuer l'opération suivante dont le résultat serait à son tour versé dans la mémoire centrale, etc.
  * **Une pile** (en anglais stack) est une structure de données fondée sur le principe: dernier arrivé, premier sorti.
- **Adressage immédiat**:la donnée est intégrée directement dans la représentation binaire de l'instruction. Ce mode d'adressage permet d'optimiser la gestion des constantes connues à la compilation.



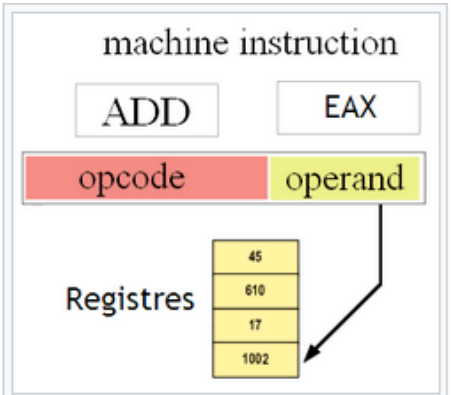
Modes d'adressage pour les données

- **Adressage direct**: Dans ce mode d'adressage, on donne l'adresse de la donnée en mémoire (RAM, ROM ou port d'E/S s'il est intégré à la mémoire). Ce mode d'adressage permet d'indiquer n'importe quel endroit dans la mémoire, le prix à payer étant que l'on doit spécifier l'adresse concernée dans son intégralité.
  Ce mode d'adressage permet parfois de lire ou d'écrire une donnée directement depuis la mémoire sans devoir la copier dans un registre.



Adressage direct.

- **Adressage registre ou inhérent** Le processeur dispose d'un certain nombre de registres de travail. Ces registres servent à stocker des données, pour qu'elles puissent servir d'opérandes pour nos instructions machines. De nombreuses instructions y font référence.



Adressage inhérent.

Jeu d'instructions 80x86

- Instructions de transfert de données
- Instructions arithmétiques
- Instructions de bits (logiques)
- Instructions de saut de programme
- Instructions de chaine de caractères
- Instructions de contrôle de processus
- Instructions d'interruptions

Instructions de transfert de données (usuelles)

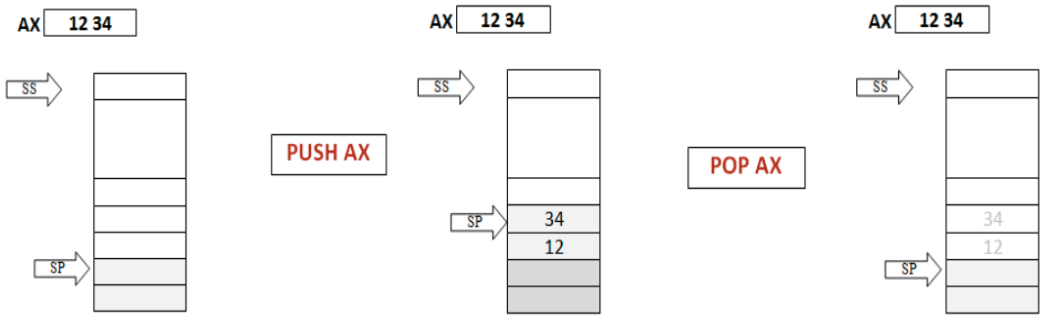
- mov R1, R2: source (R2) vers destination (R1);
- load R1, var: mémoire (var) vers registre (R1);
- store var, R1: registre (R1) vers mémoire (var);
- xchg R1, R2: échange des contenus de R1 et R2;
- clear R1: mise à zéro des bits de R1;
- set R1: mise à un des bits de R1;
- push R1: placer R1 en sommet de pile (Stack Pointer);
- pop R1: placer le sommet de pile dans R1;

Instructions de transfert de données

Exemple : MOV *Destination, Source*

```
MOV AX, BX    ; Copie du contenu d'un registre 16 bits vers un registre 16 Bits :    (BX) → AX
MOV AH, CL    ; Copie du contenu d'un registre 8 bits vers un registre 8 bits :      (CL) → AH
MOV AX, Val1  ; Copie du contenu d'une case mémoire 16 bits vers AX :                (Val1) → AX
```

Exemple : PUSH & POP



push X: empiler une donnée X: la placer au sommet de la pile et faire avancer le registre SP vers la celulle mémoire suivante.
pop X: placer dans X la donnée au sommet de la pile désigné par SP et faire avancer le registre SP vers la celulle mémoire précédente. *AX:accumulateur

Instructions arithmétiques

- add R1, R2: addition (R1 $\leftarrow$ R1 + R2);
- sub R1, R2: soustraction (R1 $\leftarrow$ R1 - R2);
- mul R1, R2: multiplication (R1 $\leftarrow$ R1 * R2);
- div R1, R2: division entière (R1 $\rightarrow$ R1 / R2);
- neg R1: complément à 2 (R1 $\leftarrow$ Ca2(R1));
- inc R1: incrémentation (R1 $\leftarrow$ R1 + 1);
- dec R1: décrémentation (R1 $\leftarrow$ R1 - 1);

Instructions logiques

Usage	Nom	Fonction
Logique	NOT	*Inversion logique sur un octet ou un mot*
	AND	*ET logique*
	OR	*OU logique*
	XOR	*OU exclusif*
	TEST	*ET logique sans résultat, affecte uniquement les flags*
Décalages	**SHL**	*Décalage logique à gauche*
	SAL	*Décalage arithmétique à gauche*
	SHR	*Décalage logique à droite*
	SAR	*Décalage arithmétique à droite*
Rotation	ROL	*Rotation à gauche*
	ROR	*Rotation à droite*
	RCL	*Rotation à gauche à travers le bit de retenue*
	RCR	*Rotation à droite à travers le bit de retenue*

Un décalage arithmétique prend en compte le bit de signe lors d'un décalage à droite pour permettre un calcul correct pour la division par deux sur les nombres négatifs codés en complément à 2.

Instructions de comparaison:

Les instructions de comparaison effectuent un calcul prédeterminé en oubliant le résultat du calcul.
Leurs opérandes ne sont donc jamais modifiés. Seuls les drapeaux sont modifiés.

- cmp R1, R2: comparaison par soustraction (R1 - R2) avec modification des drapeaux (C, Z, N, V). R1 et R2 ne sont pas modifiés.
- test R1, R2: comparaison par et bit à bit (R1 & R2) avec modification des drapeaux (C, Z, N, V); R1 et R2 ne sont pas modifiés.

Les instructions de comparaison s'utilisent avec les instructions de saut conditionnel et sont utiles à la traduction des structures conditionnelle et itérative.

Instructions de saut:

JC : (Saut si retenue)	Si CF=1 alors IP = IP + déplacement
JE/JZ : (Saut si égal/Si zéro)	Si ZF=1 alors IP = IP + déplacement
JNC : (Saut si pas de retenue)	Si CF=0 alors IP = IP + déplacement
JNE/JNZ : (Saut si Non Egal / Non Zéro)	Si ZF=0 alors IP = IP + déplacement
JNO : (Saut si pas de débordement)	Si OF=0 alors IP = IP + déplacement
JNP : (Saut si pas de parité / si parité impaire)	Si PF=0 alors IP = IP + déplacement
JNS : (Saut si pas de signe)	Si SF=0 alors IP = IP + déplacement
JO : (Saut si débordement)	Si OF=0 alors IP = IP + déplacement
JP/JPE: (Saut si parité (paire))	Si PF=1 alors IP = IP + déplacement
JS : (Saut si signe (négatif))	Si SF=1 alors IP = IP + déplacement

Instructions de commande du processeur

Type	Nom	Fonction
Indicateur (FLAGS)	STC	*Met à 1 la retenue CF*
	CLC	*Met à 0 la retenue CF*
	CMC	*Complémente la retenue*
	STD	*Met à 1 la retenue DF*
	CLD	*Met à 0 la retenue DF*
	STI	*Met à 1 l'autorisation d'interruption*
	CLI	*Met à 0 l'autorisation d'interruption*
Synchronisation	HLT	*Halte jusqu'à l'interruption*
	WAIT	*Attente jusqu'à ce que la broche TEST passe à 0*
	ESC	*Pour un coprocesseur*
	LOCK	*Verrouillage des bus pendant la prochaine instruction*
Sans opération	**NOP**	*Pas d'opération*

Langage machine

- Une instruction de langage machine correspond à une instruction possible du processeur.
- Elle contient : • un code correspondant à opération à réaliser, • les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire
- Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient
  ... 01ebe8140637274737475666662e6305f5f43544f525f4c
  5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49535
  45f5f05f5f646f5f676c6f62616c5f64746f72735f6757806
  36f6d706c657465642e3635333106474 6f725f69 ...
- C'est une suite d'instructions que l'on peut traduire directement de façon plus lisible : mov AX, BX C'est ce qu'on appelle l'assembleur. L'assembleur est donc une représentation du langage machine.

Langage machine

- Une instruction de langage machine correspond à une instruction possible du processeur.
- Elle contient : • un code correspondant à opération à réaliser, • les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire
- Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient
  ... 01ebe8140637274737475666662e6305f5f43544f525f4c
  5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49535
  45f5f05f5f646f5f676c6f62616c5f64746f72735f6757806
  36f6d706c657465642e3635331064746f725f69 ...
- C'est une suite d'instructions que l'on peut traduire directement de façon plus lisible : mov AX, BX C'est ce qu'on appelle l'assembleur. L'assembleur est donc une représentation du langage machine.

Langage machine

- Une instruction de langage machine correspond à une instruction possible du processeur.
- Elle contient : • un code correspondant à opération à réaliser, • les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire
- Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient
  ... 01ebe814063727473747566662e6305f5f43544f525f4c
  5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49535
  45f5f05f5f646f5f676c6f62616c5f64746f72735f6757806
  36f6d706c657465642e3635331064746f725f69 ...
- C'est une suite d'instructions que l'on peut traduire directement de façon plus lisible : mov AX, BX C'est ce qu'on appelle l'assembleur. L'assembleur est donc une représentation du langage machine.

Langage machine

- Une instruction de langage machine correspond à une instruction possible du processeur.
- Elle contient : • un code correspondant à opération à réaliser, • les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire
- Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient
  ... 01ebe814063727473747566662e6305f5f43544f525f4c
  5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49535
  45f5f05f5f646f5f676c6f62616c5f64746f72735f6757806
  36f6d706c657465642e3635331064746f725f69 ...
- C'est une suite d'instructions que l'on peut traduire directement de façon plus lisible : mov AX, BX C'est ce qu'on appelle l'assembleur. L'assembleur est donc une représentation du langage machine.

Langage machine

- Une instruction de langage machine correspond à une instruction possible du processeur.
- Elle contient : • un code correspondant à opération à réaliser, • les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire
- Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient
  ... 01ebe8140637274737475666662e6305f5f43544f525f4c
  5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49535
  45f5f05f5f646f5f676c6f62616c5f64746f72735f6757806
  36f6d706c657465642e3635331064746f725f69 ...
- C'est une suite d'instructions que l'on peut traduire directement de façon plus lisible : mov AX, BX C'est ce qu'on appelle l'assembleur. L'assembleur est donc une représentation du langage machine.

Les Interruptions

- L'objectif est de pouvoir prendre connaissance que le périphérique sollicite le processeur. Cette sollicitation arrive de façon totalement asynchrone.
- Deux modes sont possibles :
  - Une méthode par scrutation (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
  - Une méthode par interruption permet au périphérique lui-même de faire signe au processeur de sa présence.
- Une interruption est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption).

Les Interruptions

- L'objectif est de pouvoir prendre connaissance que le périphérique sollicite le processeur. Cette sollicitation arrive de façon totalement asynchrone.
- Deux modes sont possibles :
  - Une méthode par scrutation (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
  - Une méthode par interruption permet au périphérique lui-même de faire signe au processeur de sa présence.
- Une interruption est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption).

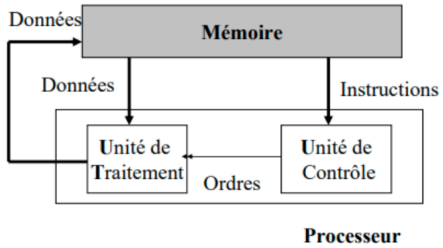
Les Interruptions

- L'objectif est de pouvoir prendre connaissance que le périphérique sollicite le processeur. Cette sollicitation arrive de façon totalement asynchrone.
- Deux modes sont possibles :
  - Une méthode par scrutation (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
  - Une méthode par interruption permet au périphérique lui-même de faire signe au processeur de sa présence.
- Une interruption est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption).

Rappel - Architecture Séquentielle

- Les machines séquentielles (un seul processeur) sont construites autour des microprocesseurs (standardisés).
  Les limites de l'exécution séquentielle:
  - Capacités d'accès à la mémoire
  - Performance
  - Temps d'exécution nécessaire
  - Débit de traitement: nombre de traitement exécutable par unité de temps.

  $->$ Nécessité de paralléliser

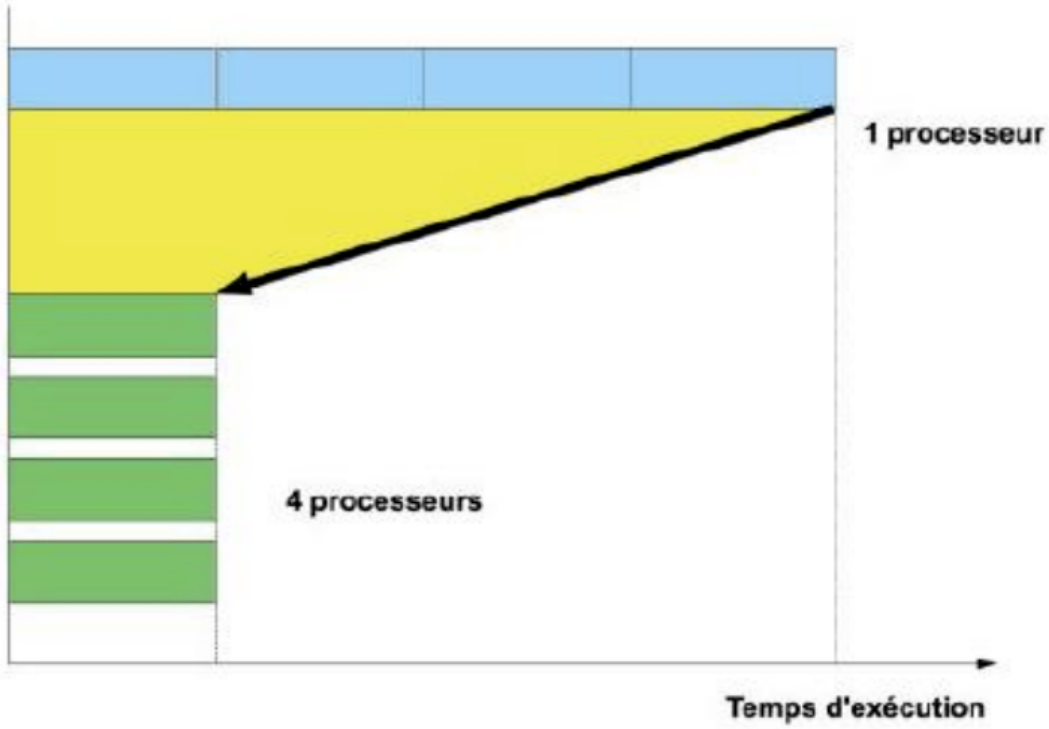


Architectures parallèles

- Une machine parallèle est essentiellement un ensemble de processeurs qui coopèrent et communiquent.
- Durant l'exécution, toutes les unités échangent des informations à travers une ressource supplémentaire: le réseau de communication interne.
- Objectif:
  - Traiter des problèmes plus grands et/ou plus complexes.
- Performances:
  - Accélération et efficacité sont une mesure de la qualité de la parallélisation.
- Le parallélisme se manifeste actuellement de plusieurs manières : en juxtaposant plusieurs processeurs séquentiels ou en exécutant simultanément des instructions indépendantes.

Multicoeur

- Processeur multi-coeur : c'est un processeur possédant plusieurs cœurs de calcul physiques fonctionnant simultanément.
- Architecture de la plupart des processeurs actuels !



Fonctionnement: les points important

Il existe des mécanismes complexes pour augmenter les performances :

- Pipelining
- Processeur superscalaire
- Hyperthreading
- Mémoire distribuée

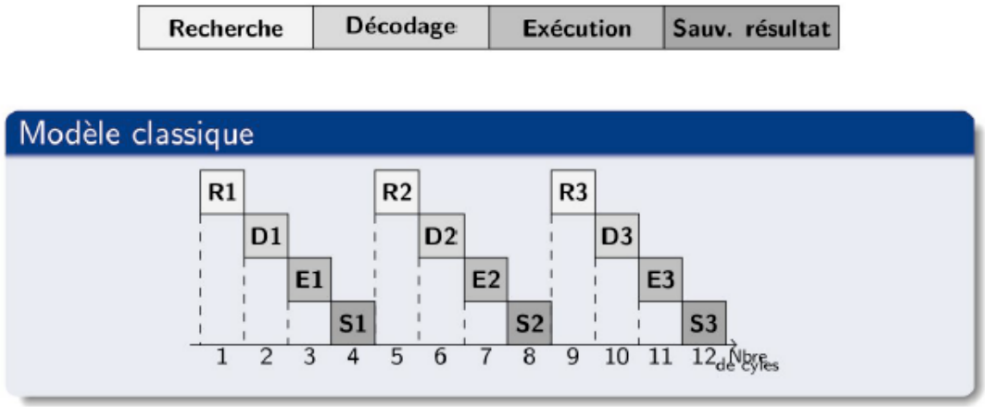
Pipeline

Il existe des mécanismes complexes pour augmenter les performances :

- L'exécution d'une instruction est décomposée en une succession d'étapes. Une instruction est découpée dans un pipeline en petits morceaux appelés étage de pipeline.
- Chaque étape correspond à l'utilisation d'une des fonctions du processeur.
- La technique du pipeline améliore le débit des instructions plutôt que le temps d'exécution de chaque instruction.
- La technique du pipeline exploite le parallélisme entre instructions d'un flot séquentiel d'instructions. Elle présente l'avantage de pouvoir, contrairement à d'autres techniques d'accélération, être rendue invisible du programmeur.

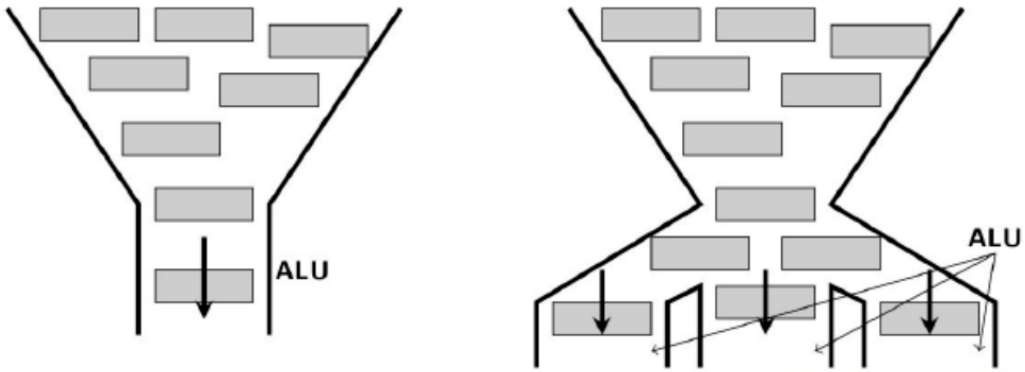
Pipeline

Exemple de pipeline à 4 étages:



- Le même travail est effectué sur des tâches différents.
- S'il y a 4 étapes dans la chaine, 4 tâches peuvent être traités simultanément et si ces étapes prennent le même temps, la chaine est continuellement occupée

Architecture superscalaire



Principe

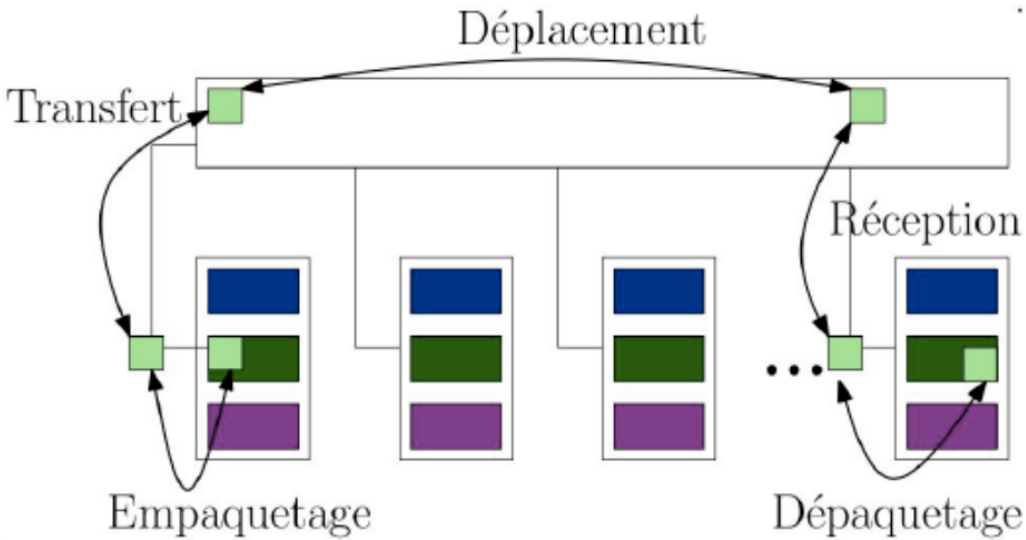
- Exécuter plusieurs instructions en même temps.
- Doter le microprocesseur de plusieurs unités de traitement travaillant en parallèle.

Hyperthreading

- SMT = Simultaneous MultiThreading, plus connu sous le nom d'Hyperthreading chez Intel.
- Principe : créer deux coeurs logiques sur une seule puce, chacun doté de ses propres registres.
- Ces deux unités partagent les éléments du coeur physique comme le cache et le bus système.
- Dégradation des performances individuelles des threads mais amélioration des performances de l'ensemble.
- Ce fonctionnement est généralement activé par défaut.

Mémoire distribuée

On dispose de N “ordinateurs” reliés par un réseau travaillant sur la même tâche.



Mémoire distribuée

- Avantages:
  - Puissance de calcul élevée
  - Augmentation proportionnelle des CPU et de la mémoire
  - Accès rapide à la mémoire locale
  - Coût raisonnable

  - Inconvénients
  - Programmation complexe : gestion explicite mémoire, échange, synchronisation
  - Temps d'accès à la mémoire non-locale très lent

Langage de programmation
Assembleur 80x86

- On apprendra quelques fonctions de ce langage.
- Pour cela, veuillez l'installer https://www.nasm.us/ sur vos ordinateurs.
- Sur Ubuntu:
  sudo apt install nasm
- ...et n'oubliez pas d'apporter vos ordinateurs aux deux derniers TD.

Langage de programmation
Assembleur 80x86

- On apprendra quelques fonctions de ce langage.
- Pour cela, veuillez l'installer https://www.nasm.us/ sur vos ordinateurs.
- Sur Ubuntu:
  sudo apt install nasm
- ...et n'oubliez pas d'apporter vos ordinateurs aux deux derniers TD.

Langage de programmation
Assembleur 80x86

- On apprendra quelques fonctions de ce langage.
- Pour cela, veuillez l'installer https://www.nasm.us/ sur vos ordinateurs.
- Sur Ubuntu:
  sudo apt install nasm
- ...et n'oubliez pas d'apporter vos ordinateurs aux deux derniers TD.

Langage de programmation
Assembleur 80x86

- On apprendra quelques fonctions de ce langage.
- Pour cela, veuillez l'installer https://www.nasm.us/ sur vos ordinateurs.
- Sur Ubuntu:
  sudo apt install nasm
- ...et n'oubliez pas d'apporter vos ordinateurs aux deux derniers TD.

Langage de programmation
Assembleur 80x86

- Tutoriel : https://riptutorial.com/assembly
- Par exemple pour compiler et exécuter: helloworld.asm
- nasm -felf64 helloworld.asm
- ld helloworld.o -o helloworld
- ./helloworld

Langage de programmation
Assembleur 80x86

- Tutoriel : https://riptutorial.com/assembly
- Par exemple pour compiler et exécuter: helloworld.asm
- nasm -felf64 helloworld.asm
- ld helloworld.o -o helloworld
- ./helloworld

Langage de programmation
Assembleur 80x86

- Tutoriel : https://riptutorial.com/assembly
- Par exemple pour compiler et exécuter: helloworld.asm
- nasm -felf64 helloworld.asm
- ld helloworld.o -o helloworld
- ./helloworld