LANGAGE ASSEMBLEUR

Chapitre III : Principe, Fonctionnement, Usage

Pr. Sarra ROUBI
LPE Informatique : S4

Plan Chapitre I :

- Langage de Programmation
  - *Evolution des langages*
  - *Langage Bas niveau*
- Langage Assembleur
  - *Syntaxe*
  - *Fonctionnement*
  - *Simulateur et Pratique*

PROGRAMMATION

Evolution des langages

Langages de programmation

- **Définition :**
- Langage formel servant à l'écriture de programmes exécutables par l'ordinateur
- **Catégories :**
- Langages de bas niveau :
  - *Langages machine*
  - *Langages d'assemblage*
- Langages de haut niveau ou évolués
  - *Fortran, Basic, Pascal, C, C++, Visual Basic, Visual, C++, Java…*

Langages de programmation

- Langage bas niveau :

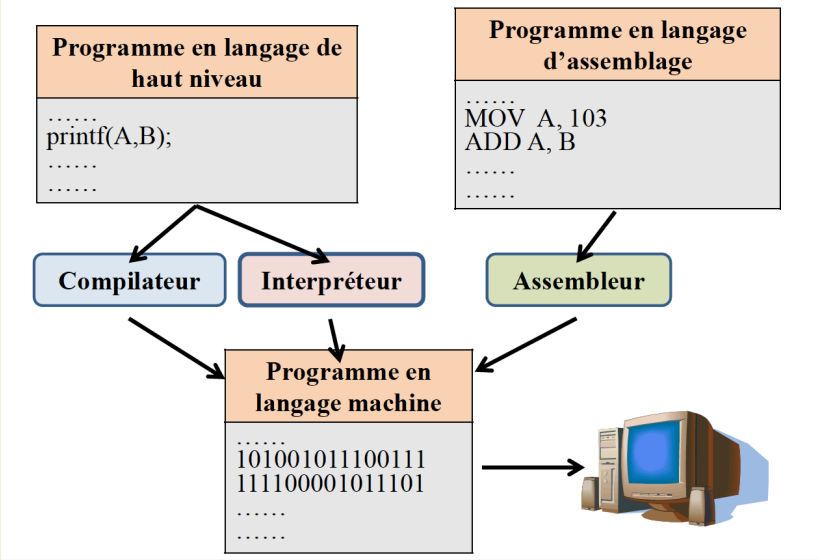
- Etroitement liés à l'ordinateur utilisé
- Difficiles à lire et à écrire (des 0 et 1)
- Fortement exposés aux erreurs
- Programmes directement exécutables par la machine ou sont à assembler

Langages de programmation

- Langage haut niveau :

- Indépendants de l'ordinateur (programmes portables)
- Faciles à utiliser (Instructions proches de la langue naturelle)
- Programmes facilement compréhensibles mais sont à compiler ou à interpréter

Langages de programmation



Langage Machine

Une instruction de langage machine correspond à une instruction possible du processeur.

- un code correspondant à opération à réaliser
- les arguments de l'opération : valeurs directes, numéros de registres, adresses mémoire

Code Op	Opérandes

Langage Machine

Si on ouvre un fichier exécutable avec un éditeur (hexadécimal), on obtient :

...
01ebe8140637274737475666 62e6305f5f43544f525f4c
5f05f5f44544f525f4c4953545f5f05f5f4a43525f4c49
53545f5f05f5f646f5f676c6f62616c5f64746f72735f6
75780636f6d706c657465642e3635333106 4746f725f69
...

Langage Machine

C'est une suite d'instructions comme 01ebe814 , que l'on peut traduire directement de façon plus lisible :

add $t7, $t3 , $sp

- C'est ce qu'on appelle l'assembleur .
- L'assembleur est donc une repr«esentation du langage machine.
- Il y a autant d'assembleurs que de type de processeurs diff«erents.

Assembleur/Processeur : CISC / RISC

C'est principalement le jeu d'instruction qui distingue les processeurs

- Les processeurs CISC (Complex Instruction Set Computer)
  - *Nombre d'instruction élevé*
  - *Les instructions réalisent souvent les transferts vers et depuis la mémoire*
  - *peu de registres*
  - *Exemples : Intel 8068, Motorola 68000*
- Les processeurs RISC (Reduced Instruction Set Computer)
  - *Peu d'instructions*
  - *Les instructions opèrent sur des registres*
  - *Registres nombreux*
  - *Exemples :Alpha, Sparc, MIPS, PowerPC*

MIPS

processeur de type RISC

- sorti dans les ann«ees 1980, utilis«e jusqu'au d«ebut 2000 (PlayStation 2)
- utilisé en informatique embarquée
- emulateurs :
  - *spim en ligne de commande,*
  - *qtspim en version graphique (pr«ef«erable).*
  - *mars impl«ementation en JAVA.*

MIPS

- Langage Assembleur:

- Voici, par exemple, quelques-unes des quelques dizaines d'instructions utilisées pour programmer un processeur x86 :
- **mov** : (move) déplace le contenu d'une case mémoire dans une autre case.
- **inc** : (increment) incrémente la valeurs contenus dans une case mémoire.
- **neg** : (negative) prend un nombre et donne son opposé.
- **cmp** : (compare) compare deux nombre.
- **jmp** : (jump) « saute » à un autre endroit du programme.

MIPS

- Exemple MIPS:

```
.data
vars: .word 5
.word 10
.text
__start:la $t0 , vars
lw $t1 , 0 ($t0 )
lw $t2 , 4 ($t0 )
saut: bge $t1 , $t2 , exit
move $a0 , $t1
li $v0 , 1
syscall
addi $t1 , $t1 , 1
j saut
exit: li $v0 , 10
syscall
```

On y trouve :
des mots clefs : .data, .word, .text
des instructions : lw, la, add, addi, bge ...
des registres : $t0, $t1, $t2
des étiquettes qui correspondent à des adresses : vars, saut,

MIPS

Trois types d'instructions

- instructions de transfert entre registres et mémoire
  - *chargement*
  - *sauvegarde*
- instructions de calcul
  - *additions*
  - *multiplications*
  - *opérations logiques*
  - *comparaisons*
  - *Sauvegarde*
- instructions de saut
  - *sauts inconditionnels*
  - *sauts conditionnels*
  - *sauvegarde*
- Appels système

MIPS

Appels système :

- MIPS permet de communiquer avec le système de façon simple par la
- commande syscall.
- La fonction utilisée est déterminée selon la valeur de $v0

$v0	commande	argument	résultat
1	print_int	$a0 = entier à lire	
4	print_string	$a0 = adresse de chaîne	
5	read_int		$v0 = entier lu
8	read_string	$a0 = adresse de chaîne, $a1 = longueur max	
10	exit		