

[ASM] Systèmes, Microprocesseurs, Architecture (1)

Par Rhaeven, relecture et adaptation par Find3r

Assembleur x86

ISA (Instruction Set Architecture) :

- Registres
- Jeu d'instructions
 - Une instruction fait entre 1 et 15 bytes

Bytecode (32 bits).

OPcode	rD	rS1	rS2
--------	----	-----	-----

OPcode	rD	immédiat (imm)
--------	----	----------------

Langage assembleur :

```
add r2, r3, r4  
r2 <- r3 + r4
```

```
mov r3, $1 // r3 <- 1
```

Langage machine : Bytecode

Assembleur : Représentation texte du bytecode

$S.c \xrightarrow{ec} .S \xrightarrow{as} \text{bytecode (ELF .O)}$

Architectures

- **CISC :**
 - Complex Instruction Set
 - → x86

- **RISC :**
 - Reduced Instruction Set
 - → MIPS
 - → ARM

CISC moderne est implémenté (pour certains cas) avec un coeur RISC et un layer de traduction

Delay Slot

```
jmp $abc  
add r1, r2, r3
```

Sur RISC, le CPU exécute les instructions 2 par 2 -> besoin d'un *nop* après les JMP

```
jmp $abc  
nop  
add r1, r2, r3
```

Instructions

- **Arithmetiques** : add, mul, ...
- **Logiques** : or | and | shift
- **Transfert**: mov, ...
- **Comparaison**: cmp %rax, \$0
- **Branchement**: jmp jcc

cc (ex: jcc) est un type de comparaison qui dépend des flags de l'opération précédente

Certaines instructions sont du sucre et n'existent pas telles quelles dans le bytecode

Registres

- %RAX
- %RBX
- %RCX
- %RDX

- %RSI
- %RDI
- %R8 -> R15
- %RSP : stack pointer
- %RBP : base pointer
- %RIP : instruction pointer
- %eflags
 - N négatif
 - Z zero
 - C carry
 - O overflow

```

1 | if (a != 0)
2 |     block A
3 | else
4 |     block B

```

```

1 |     cmp $0, %rax
2 |     jne A
3 |         block A
4 |     jmp.E
5 | A:  block A
6 | E:

```

```

mov $0, %rax
// %rax <- 0

```

mov: **mnémonique**

\$0 et %rax: **opérandes**

%xxx: **registre**

\$xxx: **immédiat**

xxx: **adresse**