CIRC2 : Assembleur, Numération, et Circuits

Licence informatique – semestre 4

Section 1

Les instructions de branchement

Le registre RIP

- registre spécial de 64 bits
  - EIP pour sa partie basse de 32 bits
  - IP pour sa partie basse de 16 bits
- contient l'adresse de la prochaine instruction à exécuter
- incrémenté automatiquement lors de l'exécution d'une instruction

Le registre RIP

- registre spécial de 64 bits
  - EIP pour sa partie basse de 32 bits
  - IP pour sa partie basse de 16 bits
- contient l'adresse de la prochaine instruction à exécuter
- incrémenté automatiquement lors de l'exécution d'une instruction

En modifiant le contenu de RIP on peut changer le cours de l'exécution du programme.

Les sauts inconditionnels

`JMP <adresse>`

- Jump to adress : remplace le contenu de RIP par `<adresse>`
- `<adresse>` peut être un label ou une constante

```
1          .text
2 debut:   movl $2, %eax
3          jmp suite
4 suite2:  addl $10, %ebx
5          jmp fin
6 suite:   movl %eax, %ebx
7          jmp suite2
8 fin:     addl %ebx, %eax
```

Un `<label>`

- dans une section `.data` contient l'adresse mémoire d'une donnée.
- dans une section `.text` contient l'adresse mémoire d'une instruction.

L'instruction de comparaison

`CMP <op1>,<op2>`

- `CMP` effectue une comparaison entre `<op1>` et `<op2>`
- le résultat de la comparaison est placé dans le registre spécial `rflags` (il sera présenté plus en détail plus loin)

Les branchements conditionnels

Ils se placent après une instruction `CMP <op1>, <op2>`.
Ils sont de la forme `Jcond <adresse>`.
Ils se lisent *saute à <adresse> si <op2> cond <op1>*

- `JE` : Equal (si les deux opérandes étaient égales)
- `JNE` : Not Equal
- `JB` et `JNB` : Below (si <op2> est plus petite que <op1>) et Not Below
- `JBE` et `JNBE` : Below or Equal et Not (Below Or Equal)
- `JA` et `JNA` : Above (si <op2> est plus grande que <op1>) et Not Above
- `JAE` et `JNAE` : Above or Equal et Not (Above Or Equal)

Exemple

`var_a` et `var_b` sont les adresses mémoires de deux entiers `long`.
On souhaite copier la plus grande des deux valeurs à l'adresse `var_c`.

```
1 maximum:
2         movl  var_b, %eax          # %eax <- var_b
3         cmpl  var_a, %eax
4         jae   bmax                 # si %eax >= var_a
5         movl  var_a, %eax
6         jmp   suite
7 bmax:   movl %eax, var_c
8 suite:  ...
```