

3- Execution de l'assembleur

Comment un programme est-il exécuté ? L'ordinateur a un programme qui décode et exécute les programmes en mémoire

L'unité de contrôle fait, en boucle infinie:

- chercher la prochaine instruction en mémoire (instruction fetch)
- décoder l'instruction (instruction decode)
- chercher les opérandes en mémoire (data fetch)
- exécuter l'instruction (execute)
- stocker les résultats (retire)
- recommencer.

3- Schémas de prog. assembleur

If..then..else:

```
if (b1 == b2) {
    S1
} else {
    S2
}
```



```
bne $t1,$t2, Else
```

```
S1
```

```
j Endif
```

```
Else: S2
```

```
Endif:
```

3- Schémas de prog. assembleur

Expressions complexes:

$t4 = (t1 * t2) / (t3 - t5) * 2$



Décomposition avec des temporaires

mul \$t1,\$t1,\$t2

sub \$t3,\$t3,\$t5

div \$t1,\$t1,\$t3

muli \$t4,\$t1,2

3- Schémas de prog. assembleur

while (..) { .. }:

\$0 est égal à 0.

```
while (p) {
    S
}
```



Entry: **beq** \$t0, \$0, **Exit**

S

j Entry

Exit:

3- Schémas de prog. assembleur

Tableau d'entiers (4
octets/élément)

A[0] = t1

A[1] = t2

A[2] = t3



A: ...

Add \$t0, \$0,A

sw \$t1, (\$t0)

sw \$t2, 4(\$t0)

sw \$t3, 8(\$t0)

!! Ne permet pas de faire A[i] avec
un mode d'adressage du type
\$t4(\$t0)

3- Schémas de prog. assembleur

Parcours de tableau

```
for (i=0; i<100; i++) {  
    A[i] = 0;  
}
```



```
add $t0, $0, 100
```

```
add $t1, $0, 0
```

```
add $t2, $0, A
```

```
L: sw $0, ($t2)
```

```
add $t1, $t1, 1
```

```
add $t2, $t2, 4
```

```
bne $t1, $t0, L
```

Calcul avec arithmétique sur
pointeurs ! (\$t2)

3- Programmation assembleur

Proposez une traduction assembleur
pour le programme

```
for (i=200; i>0; i--) {  
    if (x!=0) x = x*2-10;  
    else x=x+3;  
    A[B[i]] = B[i] +x;  
}
```



3- Conclusion

A retenir sur la programmation assembleur

- Pas de structures de contrôle évoluées (for, while, if..then..else), que des branchements
 - La programmation assembleur est difficile
 - Programmes assembleurs difficiles à lire
- Pas de structures de données (tableaux, structures)

Nécessaire de s'imposer des règles de programmation (schémas de traduction par ex comme utilisé par compilateur)

Programmation assembleur nécessaire pour

- Développement système d'exploitation, drivers matériels
- Quand compilateur pas assez mur