TRADUZIONE C-ASSEMBLER DI STATEMENT DI CONTROLLO:

IF-THEN-ELSE

IF-THEN

Davide Bertozzi

If-then-else statement

```
If (condizione)
        do_something;
else do_something_else;
```

If-then-else statement

```
If (condizione)
        do_something;
else do_something_else;
```

STRUTTURA DEL CODICE ASSEMBLER

SOLUZIONE 1:

```
PREPARA I REGISTRI PER LA
VERIFICA DELLA CONDIZIONE

BRANCH ….., ….., LABEL_RAMO_THEN
…….
J Exit:
LABEL_RAMO_THEN:
…….
Exit:
```

SOLUZIONE 2:

```
PREPARA I REGISTRI PER LA
VERIFICA DELLA CONDIZIONE

BRANCH ….., ….., LABEL_RAMO_ELSE
…….
J Exit:
LABEL_RAMO_ELSE:
…….
Exit:
```

Quale soluzione?

- Dipende da:
  - C’è conoscenza sul flusso di esecuzione più probabile? Se si, allora associa quel flusso…
    - al ramo che non fa la jump (eseguendo così meno istruzioni), e/o
    - al ramo che viene predetto correttamente dal «branch predictor» della macchina sottostante (evitando di far fare alla macchina lavoro inutile)

Sono informazioni di cui può tenere conto il compilatore.
Per le esercitazioni di questo corso, la soluzione è indifferente.

If-then-else statement

```
int main()
{
int i=3;
Int j=6;
Int k;
                    if (i < j)
                            k = 0;
                        else
                            k = 1;
printf("%d", k);
return 0;
}
```

Codice Assembler

```
.data
i: .word 3
j: .word 6

.text
 #Prepara la condizione
lw $s0, i
lw $s1, j
#Verifica della condizione, con «branch»
slt $t0, $s0, $s1 #$t0 vale 1 se (i<j)
bne $t0, $zero, ramothen
#Ramo «else»
li $s0, 1
j Exit
#Ramo «then»
ramothen:
li $s0, 0
#Uscita dallo statement condizionale
Exit:
move $a0, $s0
li $v0, 1
syscall
li $v0, 10
syscall
```

If-then-else statement

```
If (condizione)
        do_something;
……..
```

If-then-else statement

```
If (condizione)
        do_something;
.......
```

STRUTTURA DEL CODICE ASSEMBLER

PREPARA I REGISTRI PER LA
VERIFICA DELLA CONDIZIONE: «TRUE» O «FALSE»?

BRANCH ….., ….., LABEL_CONDIZIONE_FALSE
ISTRUZIONI IN CASO DI CONDIZIONE «TRUE»
LABEL_CONDIZIONE_FALSE:
CODICE SUCCESSIVO ALLO STATEMENT CONDIZIONALE

Non occorre inserire alcuna «JUMP»!

If-then statement

```
int main(){
int i=3;
Int j=6;
Int k;
                    if (i < j)
                            k = 1;
        printf("%d", k);
return 0;
}
```

Codice Assembler

```
.data
i: .word 33
j: .word 66

.text
lw $s0, i
lw $s1, j
#Salta se la condizione è falsa
slt $t0, $s0, $s1 #$t0 vale 1 se (i<j)
beq $t0, $zero, ramofalse
li $s2, 1
ramofalse:
move $a0, $s2
li $v0, 1
syscall

li $v0, 10
syscall
```