

Salti e condizioni logiche



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Condizioni logiche

seq	\$t0, \$t1, \$t2	(Set Equal To)	\$t0 = 1 se \$t1 == \$t2, altrimenti 0
slt	\$t0, \$t1, \$t2	(Set Less Than)	\$t0 = 1 se \$t1 < \$t2, altrimenti 0
sgt	\$t0, \$t1, \$t2	(Set Greater Than)	\$t0 = 1 se \$t1 > \$t2, altrimenti 0
...			

Salti e condizioni logiche



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Condizioni logiche

seq	\$t0, \$t1, \$t2	(Set Equal To)	\$t0 = 1 se \$t1==\$t2, altrimenti 0
slt	\$t0, \$t1, \$t2	(Set Less Than)	\$t0 = 1 se \$t1 < \$t2, altrimenti 0
sgt	\$t0, \$t1, \$t2	(Set Greater Than)	\$t0 = 1 se \$t1 > \$t2, altrimenti 0

...

Operazioni logiche

and	\$t0, \$t1, \$t2	(AND bit a bit)	\$t0 = \$t1 && \$t2
or	\$t0, \$t1, \$t2	(OR bit a bit)	\$t0 = \$t1 \$t2
not	\$t0, \$t1	(NOT bit a bit)	\$t0 = ! \$t1

nor \$t0, \$t1, \$t1

Handwritten notes in red:
~~_____~~ *\$t1*
~~_____~~ *\$t2*
~~_____~~ *\$t0*

Salti e condizioni logiche



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Condizioni logiche

seq	\$t0, \$t1, \$t2	(Set Equal To)	\$t0 = 1 se \$t1==\$t2, altrimenti 0
slt	\$t0, \$t1, \$t2	(Set Less Than)	\$t0 = 1 se \$t1 < \$t2, altrimenti 0
sgt	\$t0, \$t1, \$t2	(Set Greater Than)	\$t0 = 1 se \$t1 > \$t2, altrimenti 0

...

Operazioni logiche

and	\$t0, \$t1, \$t2	(AND bit a bit)	\$t0 = \$t1 && \$t2
or	\$t0, \$t1, \$t2	(OR bit a bit)	\$t0 = \$t1 \$t2
not	\$t0, \$t1	(NOT bit a bit)	\$t0 = ! \$t1

Salti condizionati

beq	\$t0, \$t1, label	(Branch if EQual)	salta se \$t0 == \$t1
ble	\$t0, \$t1, label	(Branch if Less or Equal)	salta se \$t0 <= \$t1
blt	\$t0, \$t1, label	(Branch if Less Than)	salta se \$t0 < \$t1

...

IF THEN ELSE

Esempio C

se falso

```
if (X > 0) {
```

```
    // codice da eseguire se il test è vero
```

```
} else {
```

```
    // codice da eseguire se il test è falso
```

```
}
```

```
    // codice seguente
```

IF THEN ELSE



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Esempio C

```
if (X > 0) {  
    // codice da eseguire se il test è vero  
}  
else {  
    // codice da eseguire se il test è falso  
}  
// codice seguente
```

Esempio Assembly

```
.text  
# uso il registro $t0 per la var X  
blez $t0, else      # test X <= 0  
  
# codice da eseguire se il test è vero  
  
j endIf             # esco dall'IF  
  
else:  
# codice da eseguire se il test è falso  
  
endif:  
# codice seguente
```

Ciclo DO WHILE



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Esempio C

do {

// codice da ripetere se $x \neq 0$

// il corpo del ciclo DEVE aggiornare x

} while ($x \neq 0$);

se verificato

// codice seguente

Ciclo DO WHILE



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Esempio C

```
do {
    // codice da ripetere se x != 0
    // il corpo del ciclo DEVE aggiornare x
} while (x != 0);

// codice seguente
```

Esempio Assembly

```
.text
# uso il registro $t0 per l'indice x
do:

# codice da ripetere
NOTI=0

bnez $t0, do # test x != 0

# codice seguente
```

Ciclo WHILE DO

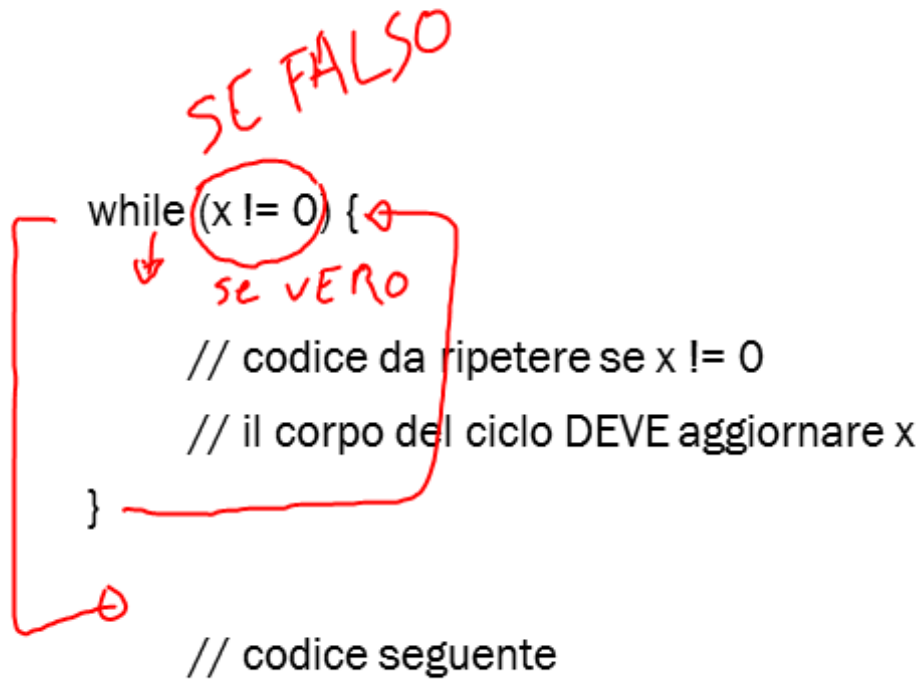
Esempio C

SE FALSO

```
while (x != 0) {  
    // codice da ripetere se x != 0  
    // il corpo del ciclo DEVE aggiornare x  
}
```

SE VERO

// codice seguente



Ciclo WHILE DO



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Esempio C

FALSO

```
while (x != 0) {
    // codice da ripetere se x != 0
    // il corpo del ciclo DEVE aggiornare x
}
```

// codice seguente

Esempio Assembly

```
.text
    # uso il registro $t0 per l'indice x
while:
    beqz $t0, endWhile # test x == 0

    # codice da ripetere

j while # loop
endWhile:
    # codice seguente
```

Ciclo FOR

Esempio C

INIZ

SE FALSO

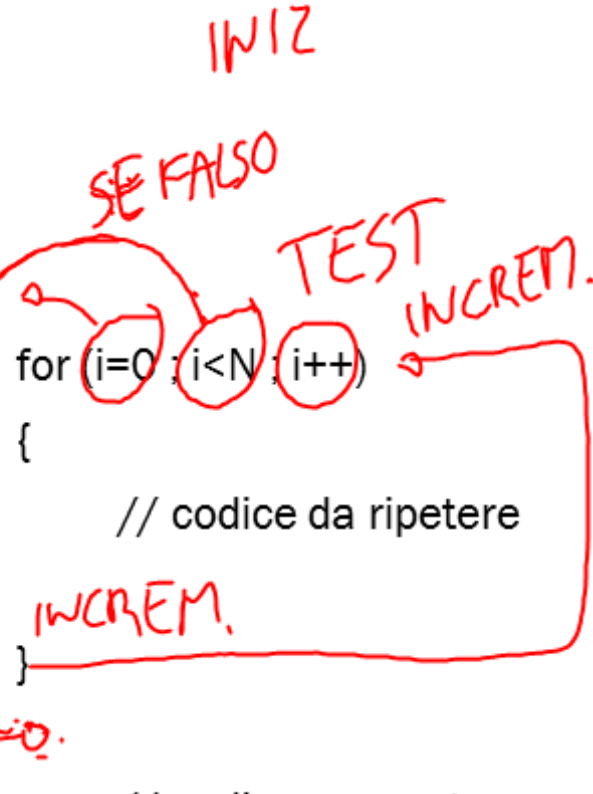
TEST

INCREM.

```
for (i=0; i<N; i++)  
{  
    // codice da ripetere  
    INCREM.  
}
```

0.

// codice seguente



Ciclo FOR

Esempio C

```
for (i=0; i<N; i++)
{
    // codice da ripetere
}
// codice seguente
```

Handwritten notes:
 $1 \oplus 1 = 0$
 $0 \oplus 0 = 0$

Diagram: A red circle highlights the entire for loop structure. A red rectangle highlights the condition $i < N$. A red arrow points from the closing brace of the loop to the comment '// codice seguente'.

Esempio Assembly

.text

```
# uso il registro $t0 per l'indice i
# uso il registro $t1 per il limite N
xor $t0, $t0, $t0      # azzero i
li $t1, N               # limite del ciclo

cicloFor:
    ble $t0, $t1, endFor # test i >= N
    # codice da ripetere
    addi $t0, $t0, 1      # incremento di i
    j cicloFor            # loop
endFor:
    # codice seguente
```

Handwritten notes:
 - A red circle highlights '\$t0' in the first two comments.
 - A red rectangle highlights '\$t1' in the second comment.
 - A red arrow points from the 'ble' instruction to the label 'cicloFor'.
 - A red circle highlights 'endFor' in the 'ble' instruction.
 - A red rectangle highlights the 'addi' instruction.
 - A red arrow points from the 'j' instruction to the label 'cicloFor'.
 - A red circle highlights 'endFor' in the label.
 - A red arrow points from the 'endFor' label to the comment '# codice seguente'.

SWITCH CASE

Esempio C

```
switch (A) {
case 0: // codice del caso 0
    break;
case 1: // codice del caso 1
    break;
// altri casi
case N: // codice del caso 3
    break;
}
// codice seguente
```

Esempio Assembly

```
.text
    sll $t0, $t0, 2      # A*4
    lw $t1, dest($t0)    # carico indirizzo
    jr $t1               # salto a registro

caso0: # codice del caso 0
    j endSwitch
caso1: # codice del caso 1
    j endSwitch
# altri casi
casoN: # codice del caso N
    j endSwitch
endSwitch:
    # codice seguente

.data
dest: .word caso0, caso1, ..., casoN
```

Es.: trova il max di un vettore

Esempio C

```
// definizione dei dati
int vettore[6] = { 11, 35, 2, 17, 29, 95 };
int N = 6;

int max = vettore[0];
// scandisco il vettore
for (i=1; i<N; i++) {
    int el = vettore[i]; // el. corrente
    if (elemento > max)
        max = elemento;
}
```

Esempio Assembly

```
.data
vettore: .word 11, 35, 2, 17, 29, 95
N: .word 6

.text
lw $t0, vettore($zero) # max ⇔ $t0
lw $t1, N # N ⇔ $t1
li $t2, 1 # i = 1

for: bge $t2, $t1, endFor
    sll $t3, $t2, 2 # i*4
    lw $t4, vettore($t3) # el. = vettore[i]
    ble $t4, $t0, else # if (el >= max)
    move $t0, $t4 # max = el.
else:
    addi $t2, $t2, 1 # i++
j for
```

syscall

Richieste al sistema operativo

Input:

- \$v0: operazione **richiesta**

- \$a0..\$a2,\$f0: eventuali **parametri**

Output:

- \$v0,\$f0: eventuale **risultato**

Syscall (\$v0)	Descrizione	Argomenti (\$a0 ...)	Risultato (<u>\$v0</u> ...)
1	Stampa Intero	<u>Intero</u>	
4	Stampa Stringa	String <u>Address</u>	
5	Leggi Intero		Intero
<u>8</u>	Leggi Stringa	<u>\$a0 = buffer</u> <u>address</u> <u>\$a1 = num chars.</u>	
10	Fine programma		

Esempio completo

Esempio C

```
// lettura di una serie  
// di interi positivi  
// terminata da 0  
// e stampa del massimo
```

```
int main() {  
    int max = 0;  
    int dato;  
    do {  
        scanf("%d", &dato);  
        if ( max == 0  
            || dato > max)  
            max = dato;  
    } while (dato != 0);  
    printf("%d", max);  
}
```

Handwritten red annotations:

- A red arrow points from the text "SYSCALL" to the `scanf` call.
- A red arrow points from the text "SYSCALL" to the `printf` call.
- A red bracket highlights the `if` statement and the `max = dato;` assignment.

Esempio completo

Esempio C

```
// lettura di una serie
// di interi positivi
// terminata da 0
// e stampa del massimo
```

```
int main() {
    int max = 0;
    int dato;
    do {
        scanf("%d", &dato);
        if ( max == 0
            || dato > max)
            max = dato;
    } while (dato != 0);
    printf("%d", max);
}
```

.text

main:

do:

endif:

Codice assembly

```
move    $t0, $zero    # $t0 ⇔ max
li      $v0, 5         # 5 ⇔ read int
syscall

seq      $t1, $t0, $zero # max == 0
sgt      $t2, $v0, $t0  # dato > max
or       $t1, $t1, $t2  # ||
beqz     $t1, endif    # if false endif
move     $t0, $v0       # max = dato
bnez     $v0, do        # se dato != 0
li      $v0, 1         # 1 ⇔ print int
move     $a0, $t0       # max
syscall

li      $v0, 10        # 10 ⇔ exit
syscall
```