

architettura degli elaboratori lezione 5

Appunti di Davide Scarlata — 2024/2025

 **Professore:** Claudio Schifanella

 **Email:** claudio.schifanella@unito.it

 **Corso:** C

 **Moodle:** <https://informatica.i-learn.unito.it/course/view.php?id=3106>

 **Data lezione:** 10/03/2025

Accesso agli Array in Assembly

Codice C:

```
int d, i, j;
int v[10];

j = 5;
v[i + d] = v[j + 2];
```

Codice Assembly:

```
addi x21, x0, 5           # j = 5

# Calcolo v[j + 2]
addi x6, x21, 2           # x6 = j + 2
slli x6, x6, 2            # x6 = (j + 2) * 4 (offset in byte)
add  x6, x6, x19           # x6 = indirizzo di v[j + 2]
lw   x6, 0(x6)            # x6 = valore di v[j + 2]

# Calcolo v[i + d]
add  x7, x9, x5            # x7 = i + d
slli x7, x7, 2            # x7 = (i + d) * 4
add  x7, x7, x19           # x7 = indirizzo di v[i + d]
sw   x6, 0(x7)            # v[i + d] = x6
```

 **Note:**

- x6 contiene l'indirizzo (e poi il valore) di `v[j + 2]`
- x7 contiene l'indirizzo di `v[i + d]`

Operazioni Logiche


Esempi:


```
and  x9, x22, x19    # x9 = x22 & x19
andi x9, x22, 5      # x9 = x22 & 5

or   x9, x22, x19    # x9 = x22 | x19
ori  x9, x22, 5      # x9 = x22 | 5

xor  x9, x22, x19    # XOR tra x22 e x19
xori x9, x22, 5      # XOR tra x22 e 5

not  x5, x6          # Pseudoistruzione NOT
```

 Le istruzioni `and`, `or`, `xor` → **formato R**

 Le istruzioni `andi`, `ori`, `xori` → **formato I**

`and x5, x6, x7`

x6

Sorgente

x7

Maschera

x5

Risultato

`or x5, x6, x7`

x6

Sorgente

x7

Maschera

x5

Risultato

Salti Condizionati in RISC-V

Istruzioni principali:

- `beq rs1, rs2, L1` → Salta a L1 se `rs1 == rs2`
 - `bne rs1, rs2, L1` → Salta a L1 se `rs1 != rs2`
 - `bltu rs1, rs2, L1` → Salta a L1 se `rs1 < rs2` (senza segno)
 - `bgeu rs1, rs2, L1` → Salta a L1 se `rs1 >= rs2` (senza segno)
-

Esempi Pratici

`if-else` in C:

```
if (i == j) {  
    f = g + h;  
} else {  
    f = g - h;  
}
```

Assembly:

```
bne x22, x23, ELSE      # Se i != j, salta a ELSE  
add x19, x20, x21        # f = g + h  
beq x0, x0, ENDIF        # Salto incondizionato  
ELSE: sub x19, x20, x21  # f = g - h  
ENDIF:
```

`for` loop in C:

```
for (i = 0; i < 100; i++) {  
    ...  
}
```

Assembly:

```
add x19, x0, x0          # i = 0  
addi x5, x0, 100         # x5 = 100  
FOR: bge x19, x5, ENDFOR # Se i >= 100 esce  
...                      # corpo del ciclo  
addi x19, x19, 1         # i++
```

```
beq x0, x0, FOR          # Salta all'inizio  
ENDFOR:
```

🔧 **Nota:** Condizione corretta: `bge x19, x5, ENDFOR`
(perché `x19` è `i`)