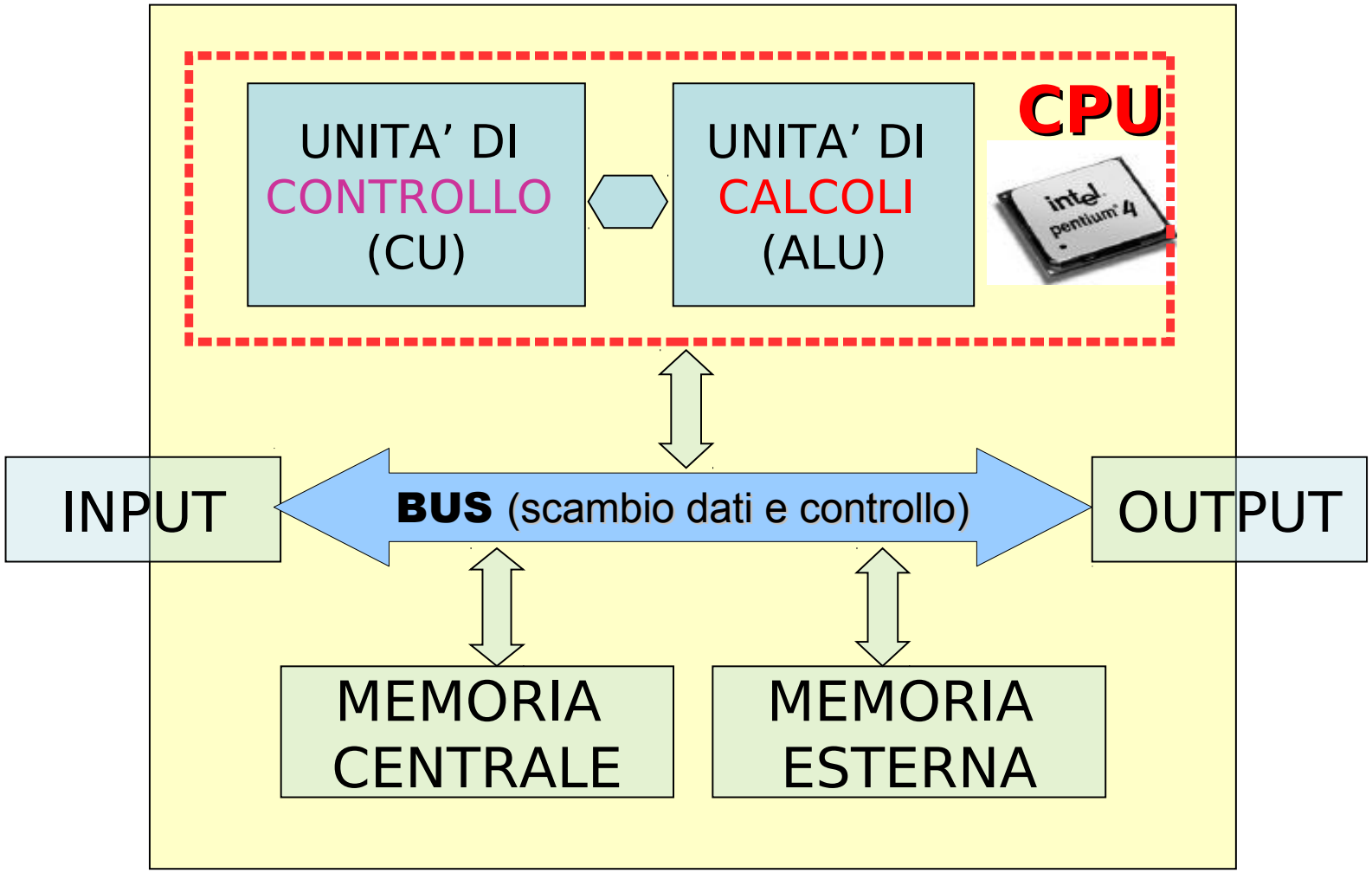
Corsi di Laurea in Biologia e Biologia Molecolare
Università di Padova, AA 2012/2013

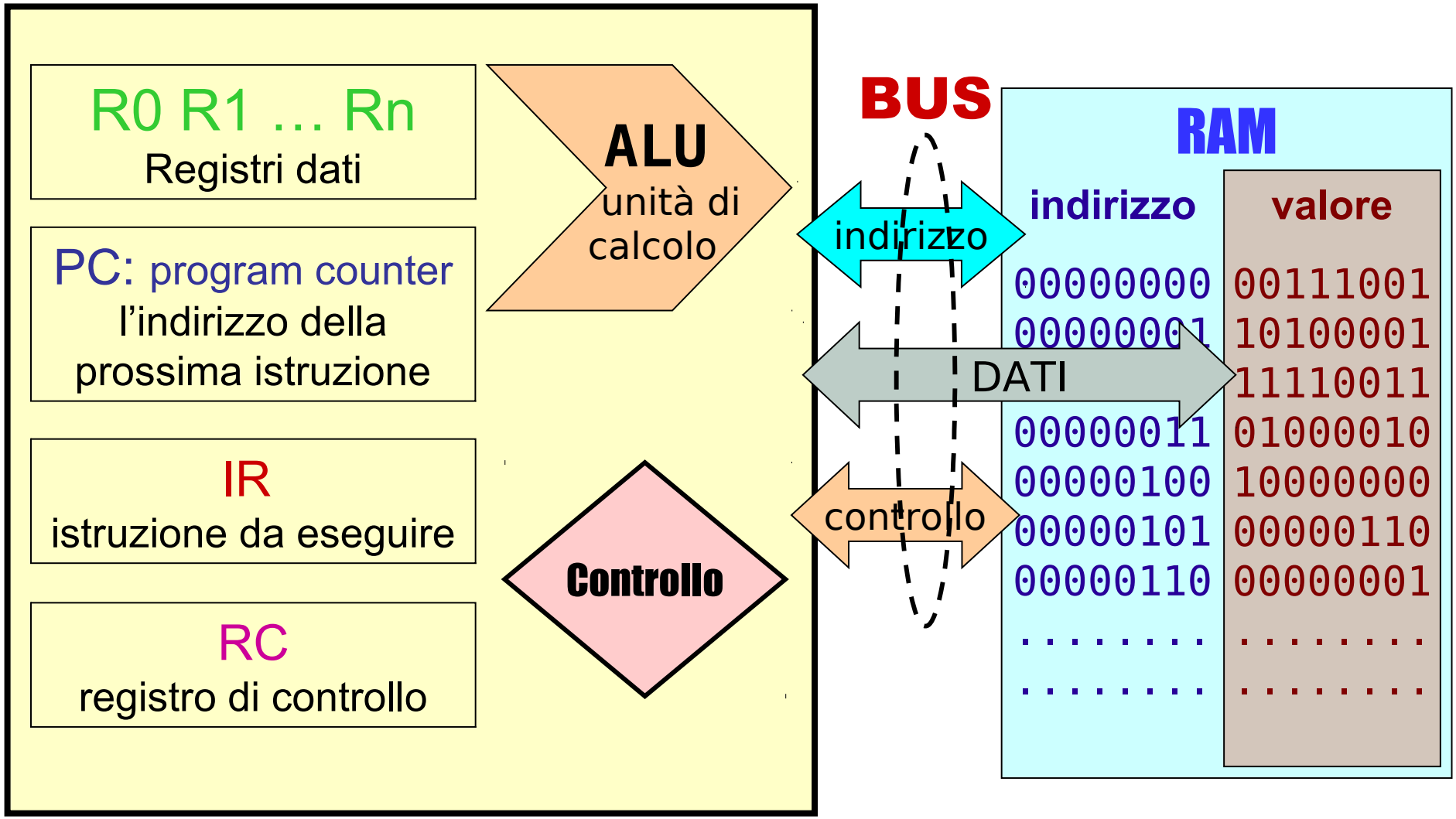
Elaborazioni al livello del CPU

www.stoianov.it

La struttura del Calcolatore



CPU


R0 R1 … Rn
Registri dati
PC: program counter
l'indirizzo della
prossima istruzione
IR
istruzione da eseguire
RC
registro di controllo
ALU
unità di
calcolo
Controllo
BUS
indirizzo
DATI
controllo
RAM
indirizzo
valore
00000000 00111001
00000001 10100001
11110011
00000011 01000010
00000100 10000000
00000101 00000110
00000110 00000001

Il CPU esegue istruzioni

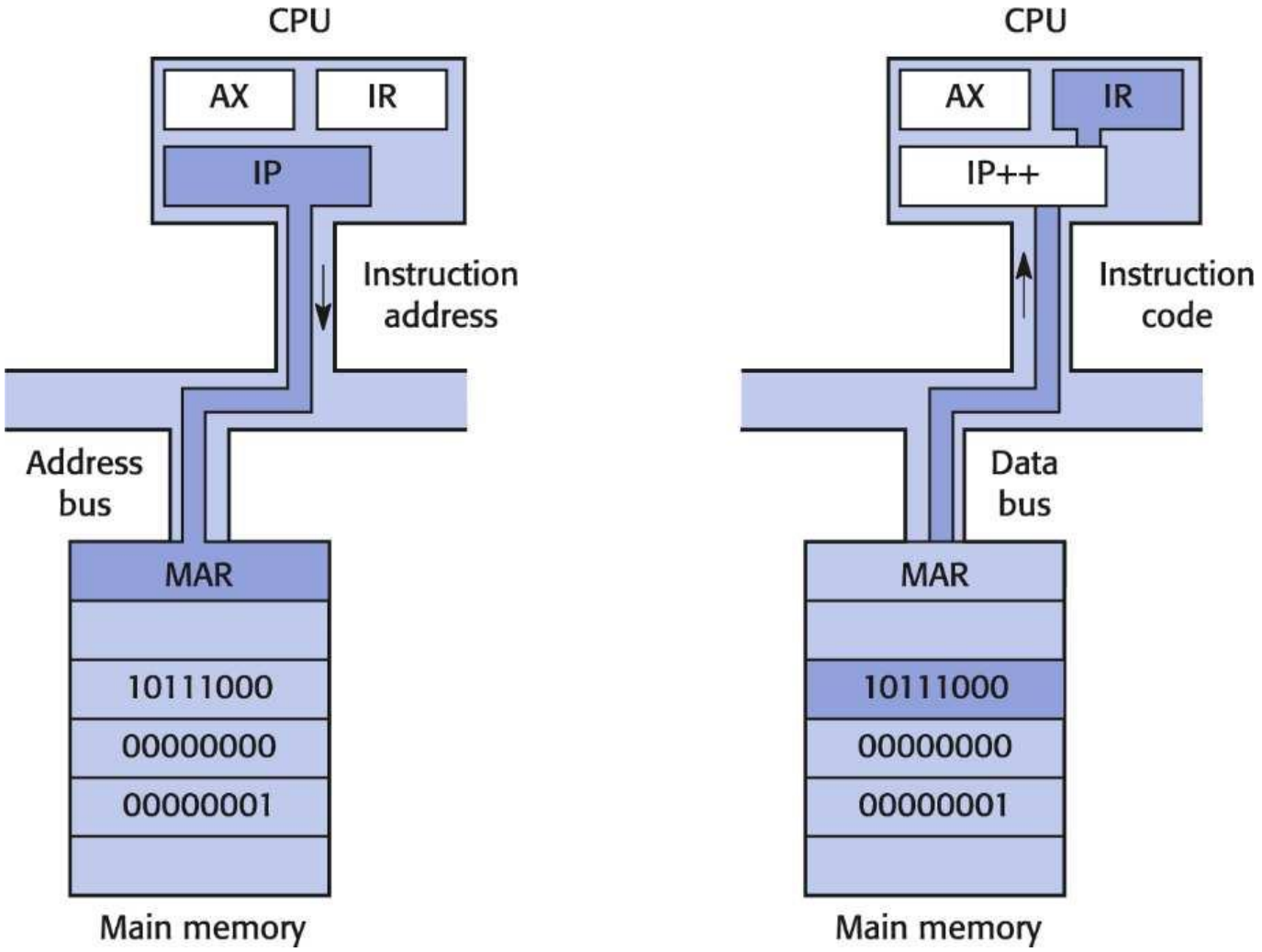
Il processore esegue istruzioni in modo sequenziale, una a volta, con un processo continuo a tre passi: **F**etch-**D**ecode-**E**xecute

i) *Preleva* un'istruzione dalla memoria con indirizzo indicato nel ***PC*** e ne pone nel ***IR.*** Poi incrementa il **PC**.

ii) *Decodifica* l'istruzione nel ***IR,*** individuando un circuito hardware da attivare.

iii) Il circuito selezionato *esegue* l'istruzione, recuperando eventuali dati (dalla memoria o registri), poi depositando il risultato (nella memoria o in un registro).

Fetch (dettaglio)



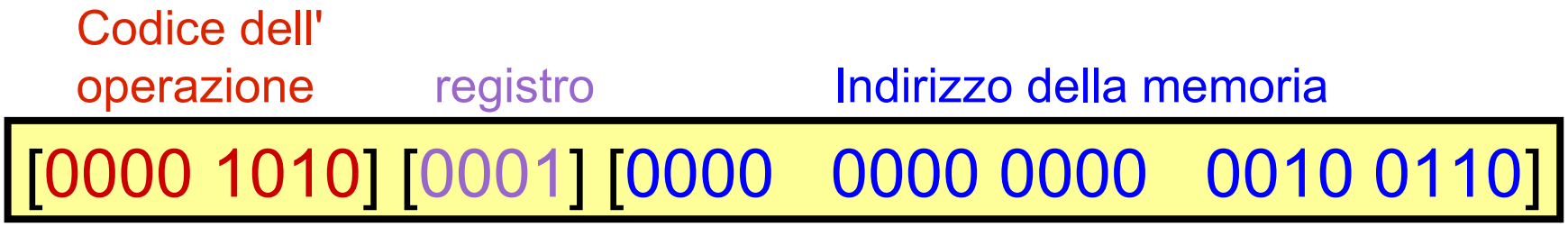
Codifica delle istruzioni

Una sequenza di bit (ad es., 4 byte) che contiene:

1) Il codice binario di un operazione da eseguire

2) parametri dell'operazione:
(registri dati, valori, indirizzi nella memoria)

Esempio:



Categorie di istruzioni

- **Trasferimento** dati (tra memoria e registri)
- **Calcolo**, aritmetica e logica (per es., somma, …)
- **Controllo** del flusso (confronto, salto, …)
- **Input / Output** dai dispositivi I/O

Istruzioni
Trasferimento dati

Codice operazione *indice-registro* *Indirizzo della memoria*

```
[0000 0000][rrrr][aaaa aaaa aaaa aaaa aaaa]
```

- carica in un registro-dati ***rrrr*** il contenuto della cella di memoria con indirizzo ***a..a***

```
[1000 0000][rrrr][xxxx xxxx xxxx xxxx xxxx]
```

- carica nel registro dati ***rrrr*** *il valore* ***xx..xx***

```
[0000 0001][rrrr][aaaa aaaa aaaa aaaa aaaa]
```

- carica nella cella di memoria con indirizzo ***a..a*** il contenuto del registro dati ***rrrr***

Istruzioni Aritmetiche

0000 0010 somma i valori binari di tipo ***integer*** contenuti nei registri ***iiii*** e *jjjj*. Il risultato resta nel registro ***iiii***

Operazione registro1 registro2 bit non-utilizzati

```
[0000 0010][iiii][jjjj] [xxxxxxxx xxxxxxxx]
```

0000 0100 sottrazione (integer)

0000 0110 moltiplicazione (integer)

0000 1000 divisione (integer)

Operazioni aritmetiche di dati tipo ***float*** (numeri reali)

0000 0011 somma (float)
0000 0101 sottrazione (float)
0000 0111 molitiplicazione (float)
0000 1001 divisione (float)

Istruzioni Aritmetiche tra registri e costanti

1000 0010 aggiunge al contenuto del registro ***iiii*** una costante contenuta nella istruzione stesa

Operazione registro dati valore (20 bit)

`[1000 0010][iiii][xxxx xxxx xxxx xxxx xxxx ]`

1000 0100 sottrazione (integer)

1000 0110 moltiplicazione (integer)

1000 1000 divisione (integer)

erazioni aritmetiche di dati tipo ***float*** (numeri reali)

1000 0011 somma (float)
1000 0101 sottrazione (float)
1000 0111 molitiplicazione (float)
1000 1001 divisione (float)

Un programmino nel codice binario

```
0000 0000  0000 0000  0000 0000 0100 0000
0000 0000  0001 0000  0000 0000 0100 0001
0000 0110  0000 0001  0000 0000 0000 0000
0000 0001  0000 0000  0000 0000 0100 0010
```

Che cosa fa ? ... :-(

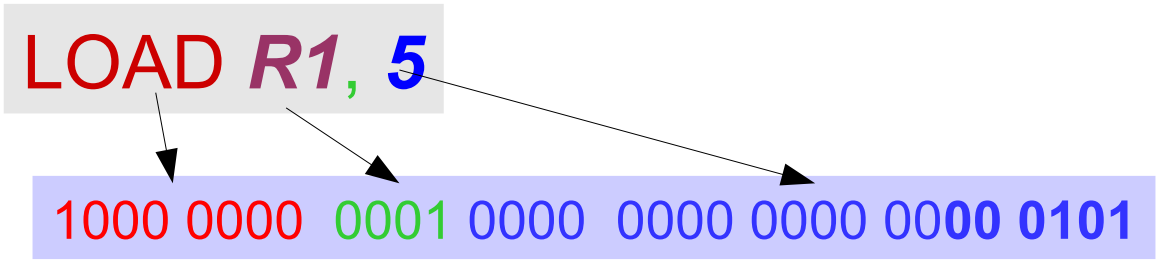
Codice **Mnemonico**

- Programmare nel codice binario è disumano.

- Molto più facile se usare codici mnemonici:
  - istruzioni (ad es., “0000 0001”=STORE)
  - Registri (0001 = R1)

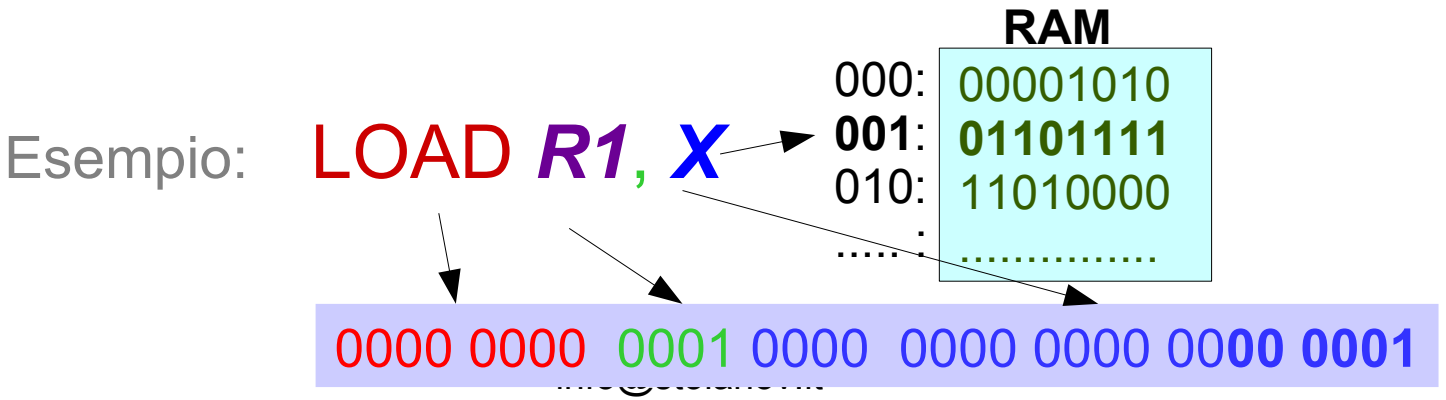
- Ma il CPU capisce solo il codice binario … allora ?
- Un programma – **compilatore** – traduce il codice mnemonico nel codice binario del CPU.

Esempio:

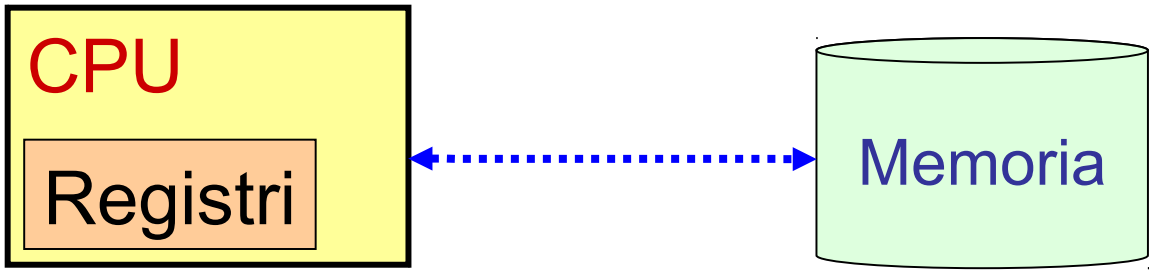


Riferimenti mnemonici degli indirizzi nella RAM

- Nei codici mnemonici, anche gli *`indirizzi nella RAM`* (di dati ed istruzioni) sono riferiti per nome:
  - I riferimenti mnemonici degli indirizzi della RAM che contengono dati si chiamano **variabili**.
  - I riferimenti mnemonici degli indirizzi nella RAM che contengono istruzioni del CPU si chiamano **etichette**.
- I **compilatori** traducono l'insieme di codici memonici di { istruzioni, variabili e etichette} in codice binario.



Istruzioni mnemonici per trasferimento dati



- LOAD ***R***, ***Var*** carica nel registro-dati ***R*** il contenuto della variabile ***Var*** (una cella di memoria)
- LOAD ***R***, ***cost*** carica nel registro-dati ***R*** la costante ***cost***
- LOAD ***R***, ***R1*** carica nel registro-dati ***R*** il contenuto del registro dati **R1**

- STORE ***R***, ***var*** copia il contenuto del registro dati ***R*** nella variabile ***Var*** (una cella di memoria)

Istruzioni mnemonici per l'Aritmetica binaria

- ADD ***R1***, ***R2*** somma i valori di tipo ***integer*** *(interi)* contenuti nei registri ***R1*** e ***R2***. Il risultato resta nel registro ***R1***

ADD R1 R2 bit non-utilizzati

```
[0000 0010][iiii][jjjj] [xxxxxxxx xxxxxxxx]
```

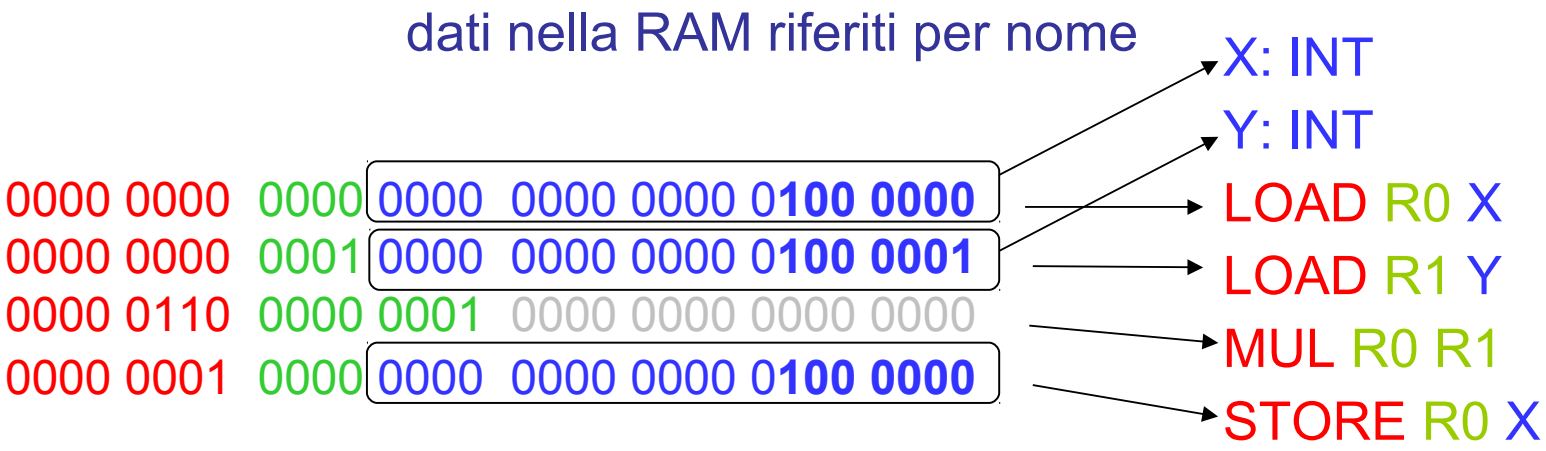
- ADD ***R***, ***cost*** *aggiunge al contenuto del registro* ***R*** *la costante* ***cost***

ADD R cost

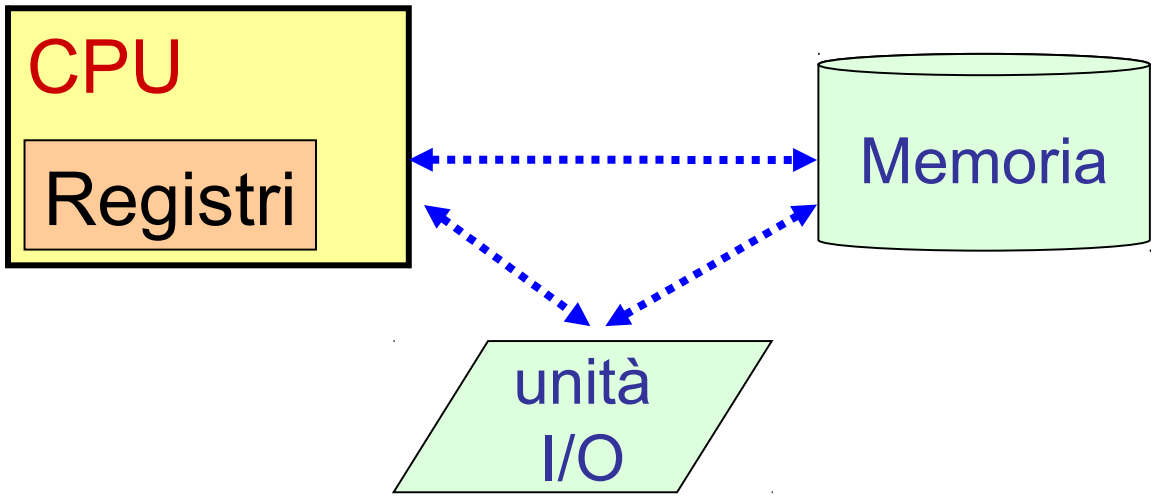
```
[1000 0010][iiii][ xxxx xxxx xxxx xxxx xxxx ]
```

- SUB ***R1***, ***R2*** e SUB ***R1***, ***cost*** sottrazione (integer)
- MUL ***R1***, ***R2*** e MUL ***R1***, ***cost*** moltiplica (integer)
- DIV ***R1***, ***R2*** e DIV ***R1***, ***cost*** divide (integer)
- **FADD, FSUB, FMUL, FDIV**: operazioni su dati ***float***

Il “programmino” nel codice Mnemonico



Input / Output



- READ ***u***, ***Var*** legge un byte dall'unità ***u*** e lo memorizza nella variabile ***Var***
- WRITE ***u***, ***Var*** scrive il contenuto della variabile ***Var*** all'unità ***u.***
- Unità I/O: (0 stdinp: tastiera), (1 stdout: video), ecc.

Un programma con input, elaborazione, output

Calcolare la somma di due valori inseriti con la tastiera

```
X: INT;
Y: INT;
```
Definisce variabili X e Y

```
READ  STINP X;
READ  STINP Y;
```
Leggere due valori dalla tastiera ed assegnarne alle variabili X e Y

```
LOAD  R0 X;
LOAD  R1 Y;
```
Copiare X e Y in R0 e R1

```
ADD   R0 R1;
```
Calcolare la somma di R0 e R1

```
STORE R0 X;
```
Copiare la somma (da R0) in X

```
WRITE STOUT X;
```
Scrivere il valore di X al *terminal*

I **commenti** servono per poter facilmente **spiegare, capire e modificare** il programma !!

Branch e STOP

- BRANCH ***addr*** spostamento del PC all'indirizzo ***addr***

- STOP termina l'esecuzione del programma

Branch Operazione nuovo indirizzo per il PC

```
[1001 0001][xxxx][aaaa aaaa aaaa aaaa aaaa]
```

STOP Operazione bit non-utilizzati

```
[1001 0001][xxxx xxxx xxxx xxxx xxxx xxxx]
```

Un programma completo

Calcolare la somma di due variabili X e Y

```
X: INT 10;     X=10;   Definisce due variabili X e Y e
Y: INT  2;     Y=2     darne valori iniziali

LOAD R0 X;     R0=X; coppia X in R0
LOAD R1 Y;     R1=Y; coppia Y in R1

ADD R0 R1;     R0=R0+R1; calcola la somma dei registri

STORE R0 X;    X=R0; copia il risultato nella variabile X

STOP;          Ferma il programma
```

I **commenti** servono per poter facilmente
spiegare, capire e modificare il programma !!

Confronto

- COMP ***r1***, ***r2*** confronta i valori-***integer*** dei registri ***r1*** e ***r2***

  Risultato: Se Rr1<Rr2 , memorizza -1 nel registro ***RC***
  Se Rr1=Rr2 , memorizza 0
  Se Rr1>Rr2 , memorizza +1

- COMP ***r1***, ***cost*** confronta il registro ***r1*** con *la costante* ***cost***

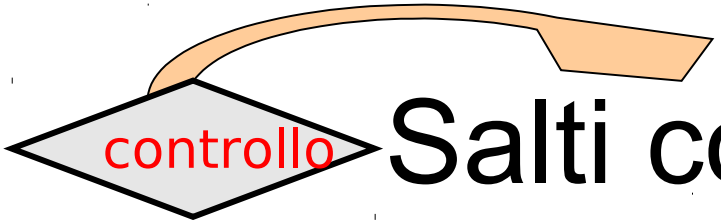
- FCOMP ***r1***, ***r2*** confronta valori del tipo ***float***

COMP Operazione registro1 registro2 bit non-utilizzati

```
[0010 0000][iiii][jjjj] [xxxxxxxx xxxxxxxx]
```

COMP Operazione registro costante

```
[1010 0000][iiii][xxxx xxxx xxxx xxxx xxxx]
```



Salti condizionali

- BRLT ***addr*** se RC=-1, spostare il PC all'***addr***
- BRLE ***addr*** se RC<=0, spostare il PC all'***addr***
- BREQ ***addr*** se RC = 0, spostare il PC all'***addr***
- BRNE ***addr*** se RC ≠ 0, spostare il PC all'***addr***
- BRGT ***addr*** se RC = 1, spostare il PC all'***addr***
- BRGE ***addr*** se RC>=0, spostare il PC all'***addr***

BRLT Operazione bit non-utilizzati nuovo indirizzo per il PC

[0100 0001] [xxxx] [aaaa aaaa aaaa aaaa aaaa]

Un programma condizionale

Ordinare i valori di due variabili X e Y

```
X:      INT 10;   Definisce variabili X e Y
Y:      INT  2;

LOAD    R0, X;    Carica X e Y nei registri R0 e R1
LOAD    R1, Y;

RC
COMP    R0, R1;   Confronta i valori di R0 e R1
BRLE    FINE;     Se ordinati, fine!
                   altrimenti …

STORE R1, X;      Copia R1 in X (per scambiare X e Y)
STORE R0, Y;      Copia R1 in Y

FINE:
STOP;             fermare il programma
```