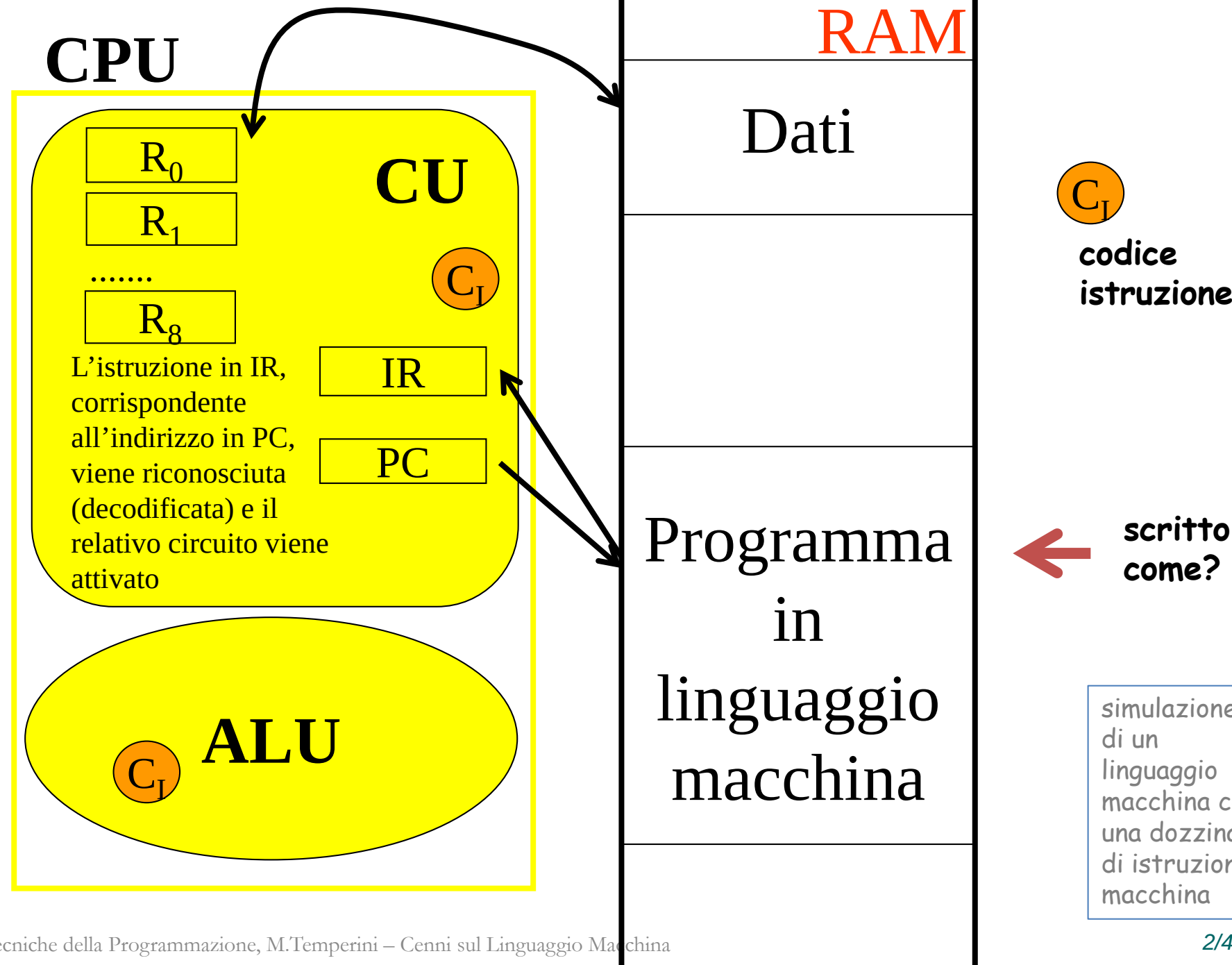
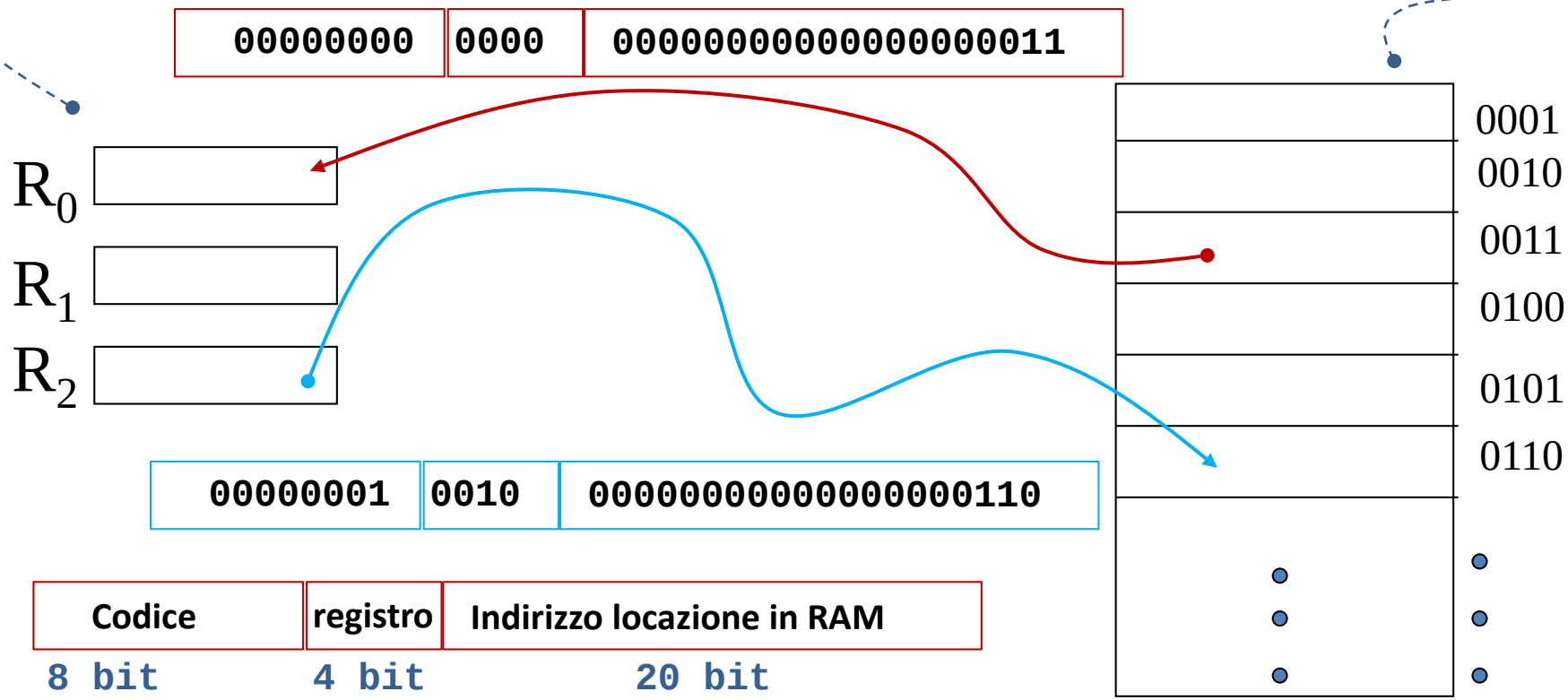
Tecniche della Programmazione

Cenni sul linguaggio macchina
e
sull'esecuzione di un programma scritto in linguaggio macchina

(ciclo di esecuzione delle istruzioni macchina da parte della CPU:
Fetch, Decode, Execute)


CPU
CU
R_0
R_1
.....
R_8
C_I
L'istruzione in IR, corrispondente all'indirizzo in PC, viene riconosciuta (decodificata) e il relativo circuito viene attivato
IR
PC
ALU
C_I
RAM
Dati
Programma in linguaggio macchina
C_I
codice istruzione
scritto come?
simulazione di un linguaggio macchina con una dozzina di istruzioni macchina

Istruzioni di trasferimento: registri ⇔ RAM

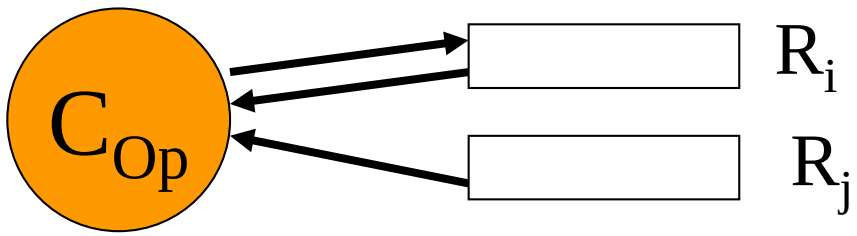


"Load" **Codice 00000000:** copia nel registro specificato il contenuto della locazione specificata

"Store" **Codice 00000001:** copia nella locazione specificata il contenuto del registro

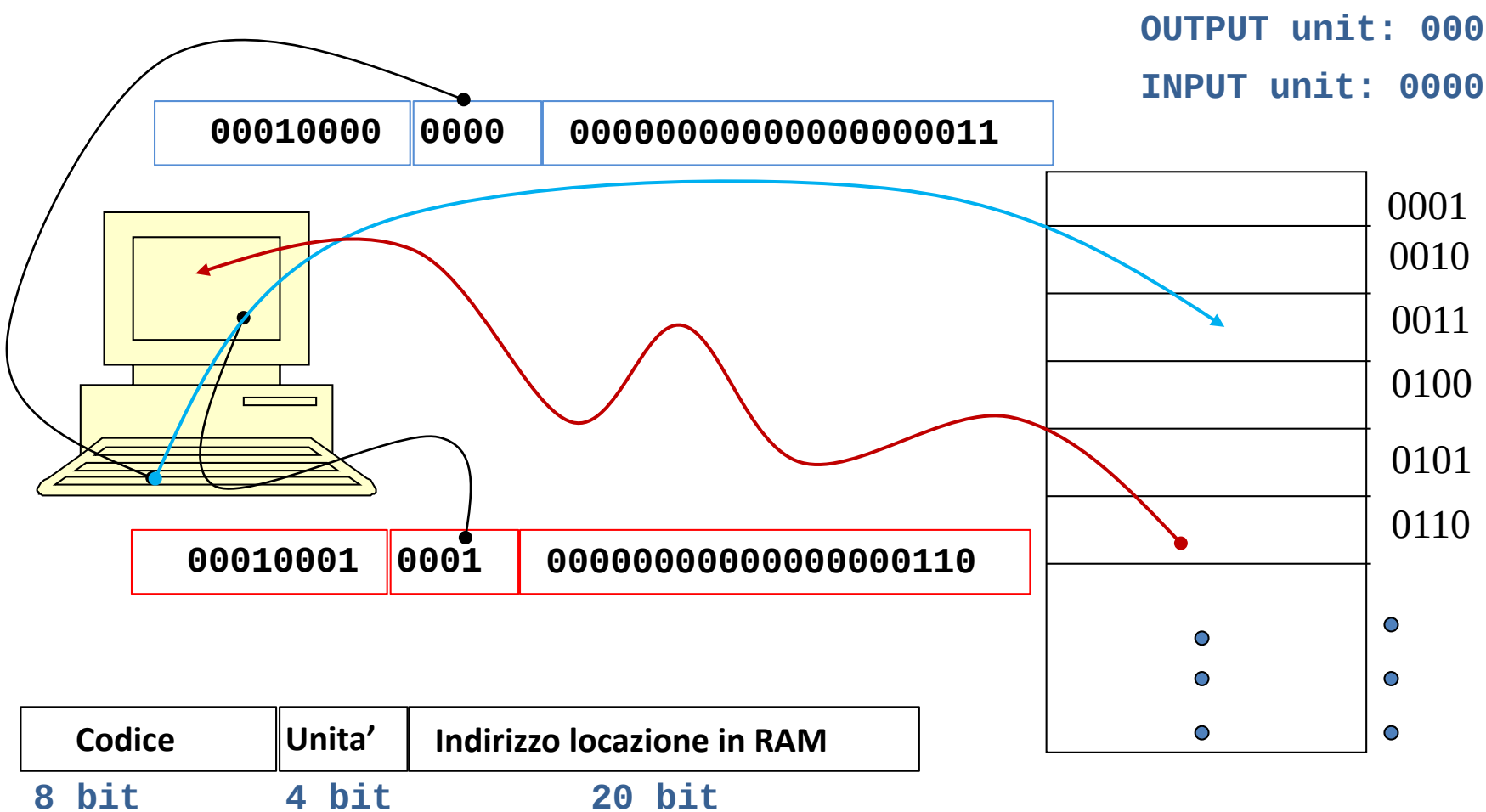
Istruzioni aritmetiche

eseguono somma, differenza, moltiplicazione e divisione usando i registri come operandi



ADD	**00000010**	**SUB**	**00000100**
MULT	**00000110**	**DIV**	**00001000**
MOD	**00001010**		

Istruzioni di I/O: unita' I/O ⇔ RAM



Codice	Unita'	Indirizzo locazione in RAM
8 bit	4 bit	20 bit

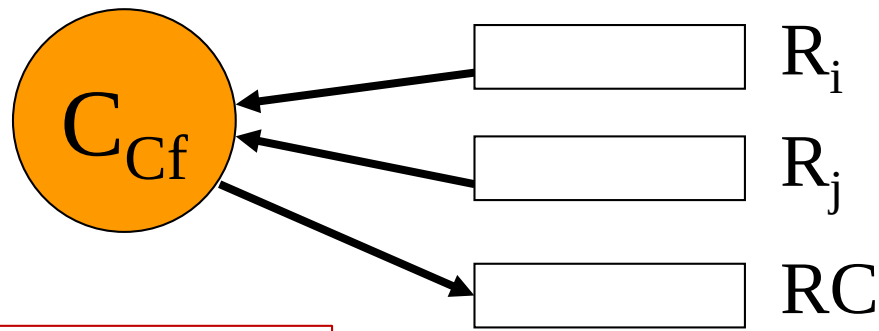
"write" **Codice 00010001:** trasferisce all'unita' di output specificata il dato presente nella locazione specificata

"read" **Codice 00010000:** legge un dato dall'unita' di input specificata e lo copia nella locazione specificata

Istruzione di confronto

confrontano il contenuto di 2 registri R_i ed R_j:

- se $R_i < R_j$ viene messo -1 nel registro RC
- se $R_i == R_j$ (uguali) viene messo 0 in RC
- se $R_i > R_j$ … 1 in RC



COMP 00100000

C_{Cf}

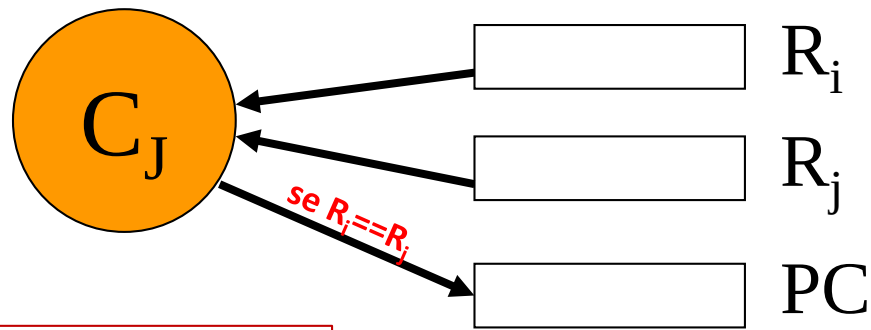
esecuzione istruzione corrispondente al codice

NB usiamo il simbolo == per significare "i valori sono uguali"

Istruzione di salto (branch, jump)

confrontano il contenuto di 2 registri R_i ed R_j:

- se $R_i == R_j$ l'indirizzo viene messo nel registro PC, per cui l'istruzione in quell'indirizzo sara' la prossima da eseguire.
  Senno', il PC rimane com'era.



8 bit 4 bit 4 bit …

```
JMP   00100001
```

C_J

esecuzione istruzione corrispondente al codice

STOP

termina il programma

Codice: **STOP** **10000001**

8 bit **Non usati**

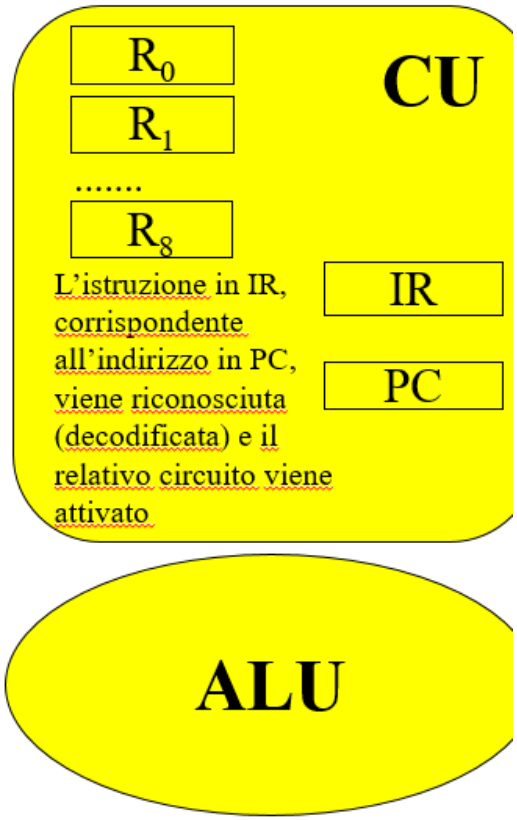
esempio

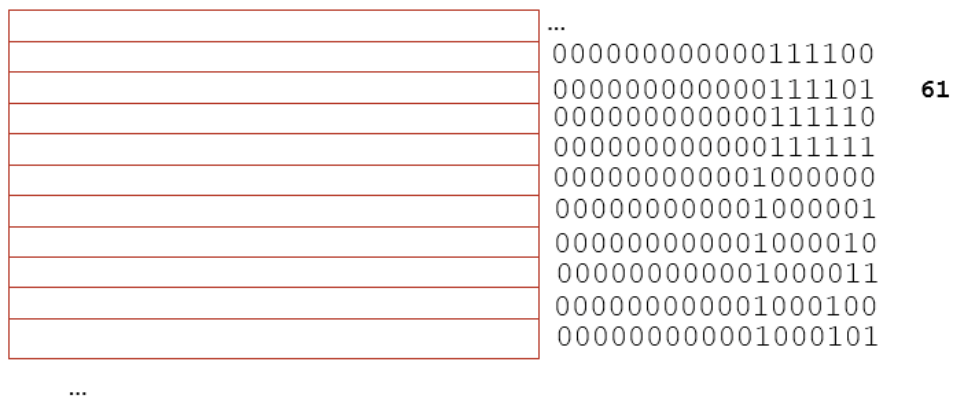
scriviamo un programma macchina che:

- trasferisce il contenuto di 2 parole di indirizzo 64 e 68 della RAM nei registri R_0 ed R_1
- li somma
- trasferisce la somma nella parola di indirizzo 60 della RAM

Algoritmo

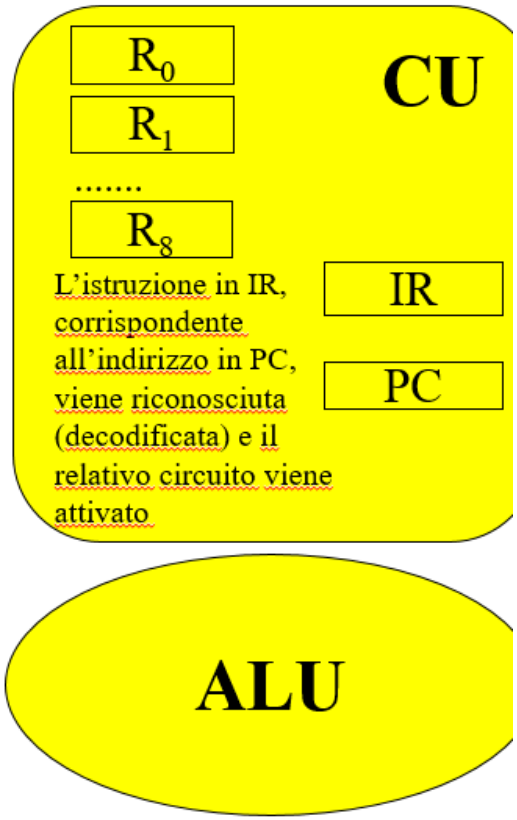
- Si trasferisce il contenuto della locazione 1000000 (indirizzo 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina

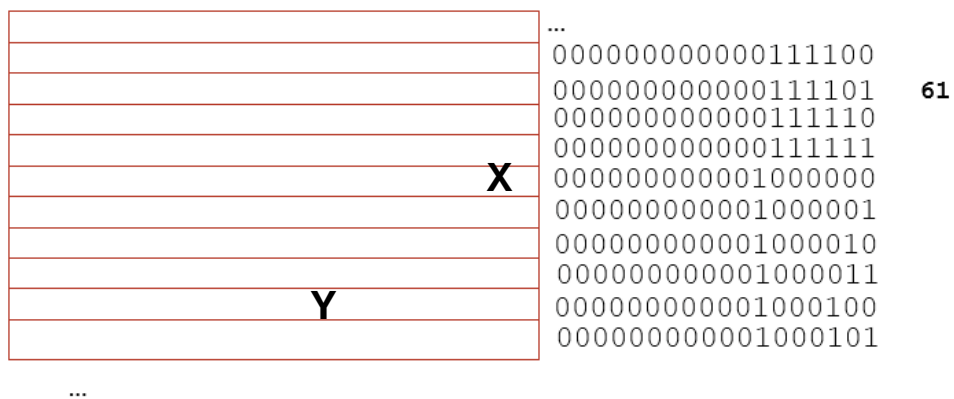




Algoritmo

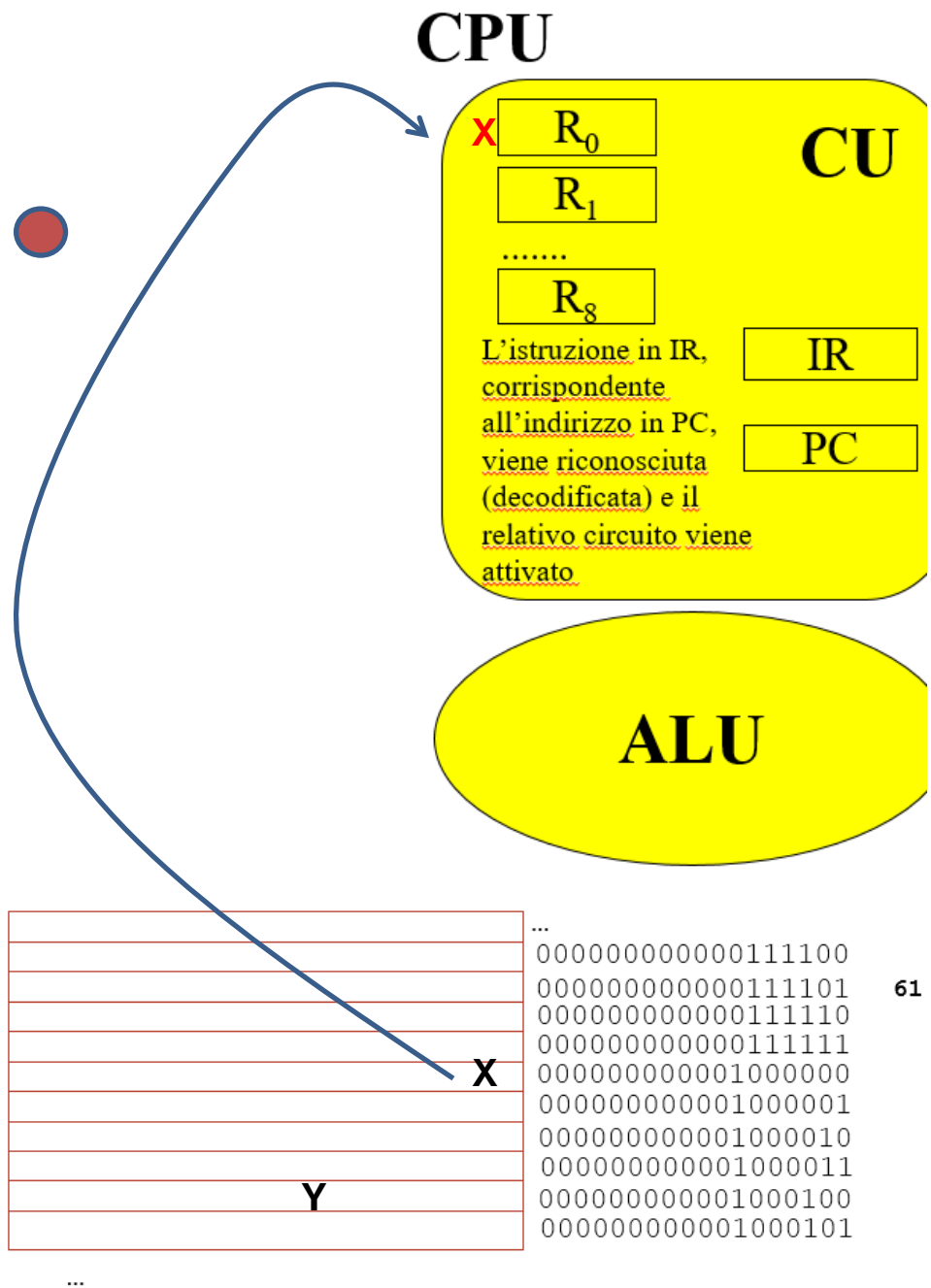
- Si trasferisce il contenuto della locazione 1000000 (indirizzo 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina





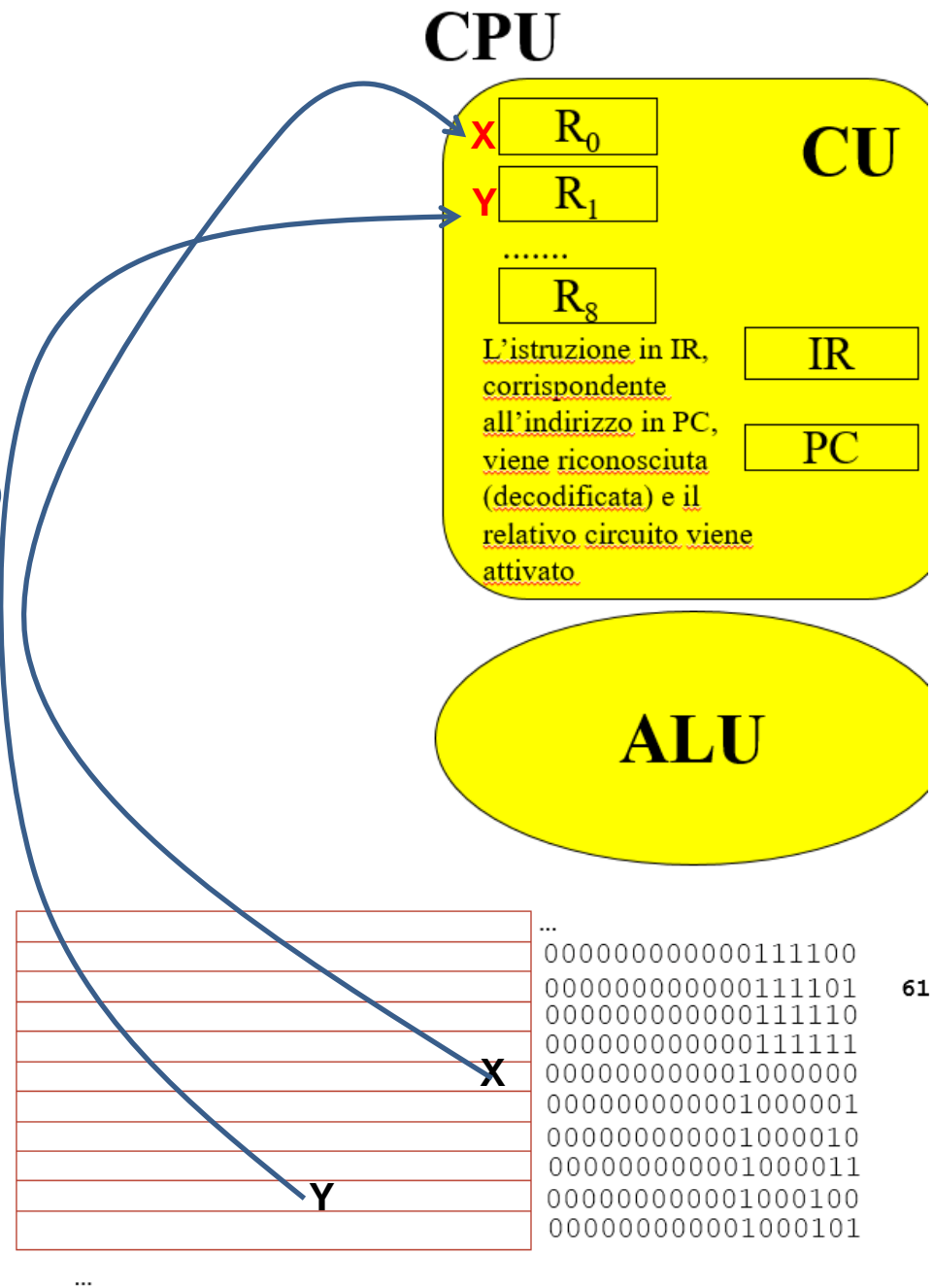
Algoritmo

- Si trasferisce il contenuto della locazione 1000000 (cioe' 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina



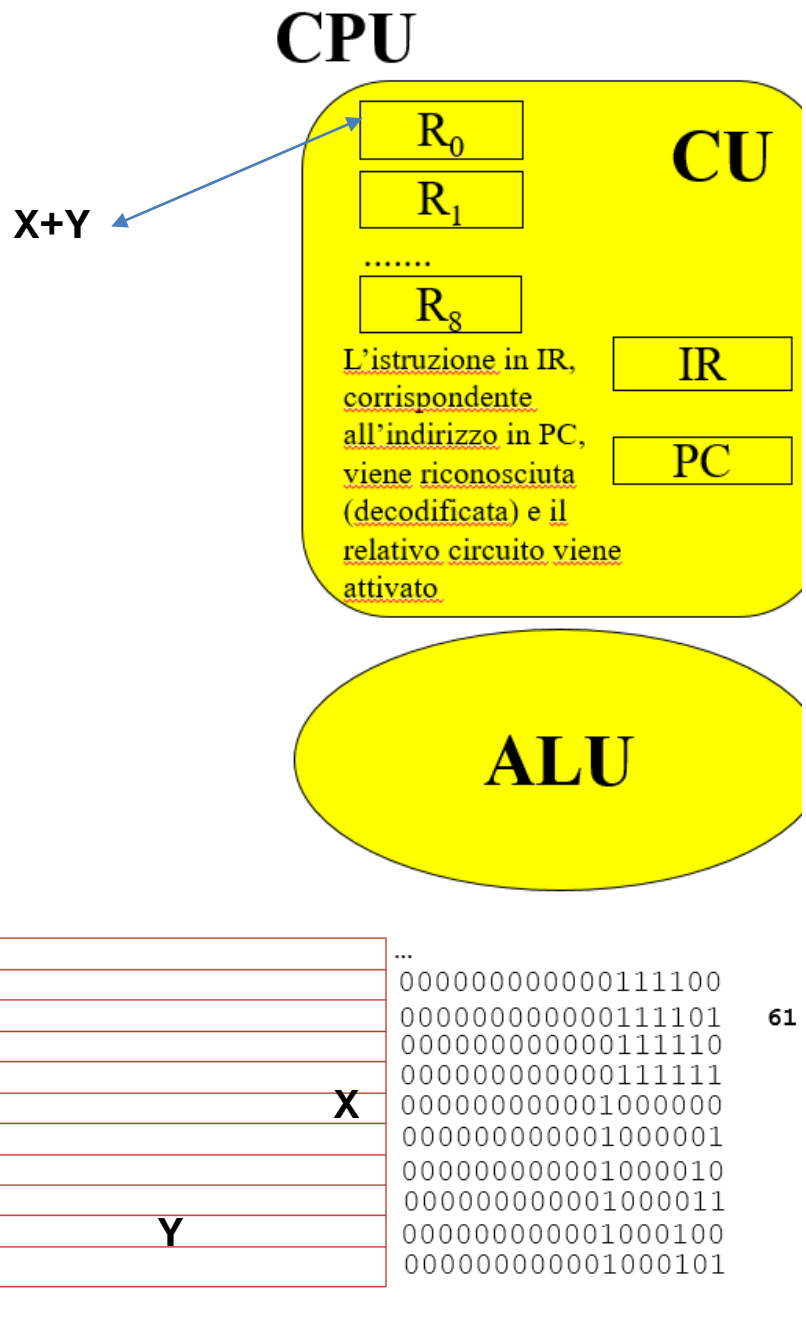
Algoritmo

- Si trasferisce il contenuto della locazione 1000000 (cioe' 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina



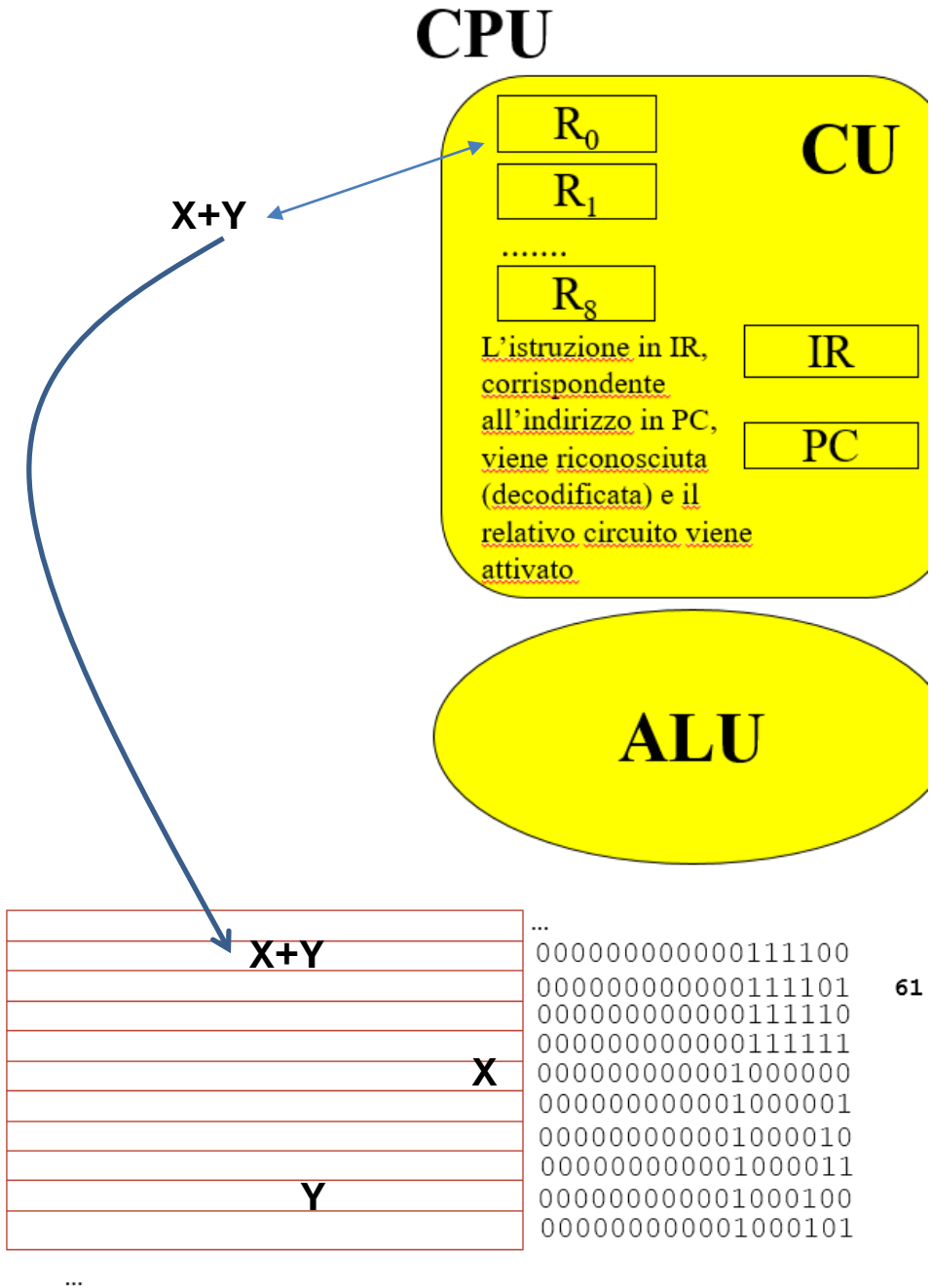
Algoritmo

- Si trasferisce il contenuto della locazione 1000000 (cioe' 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina



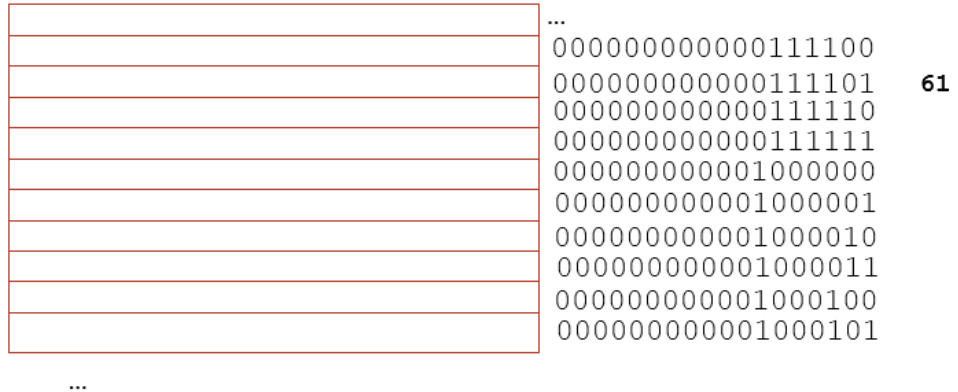
Algoritmo

- Si trasferisce il contenuto della locazione 1000000 (cioe' 64) nel registro R0 (ind. 0000);
- Si trasferisce il contenuto della locazione 1000100 (cioe' 68) nel registro R1 (ind. 0001);
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100 (cioe' 60)
- E poi si termina



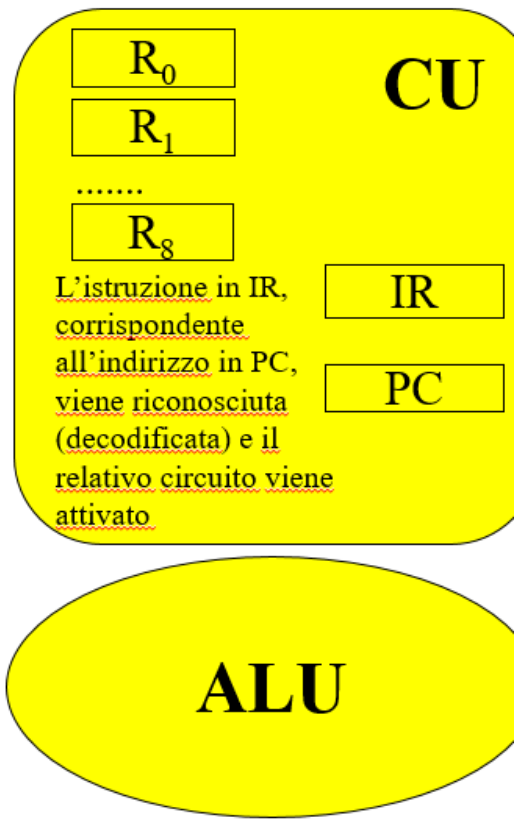
Traduzione dell'algoritmo

- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100



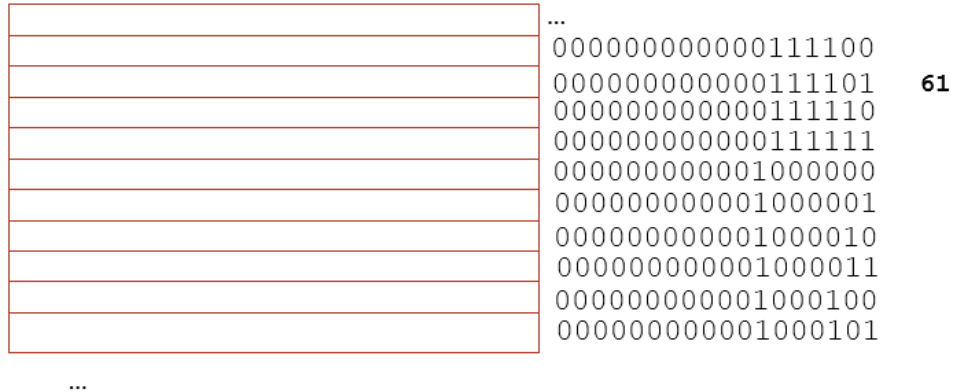
Programma in LM

00000000	0000	00000000000001000000
00000000	0001	00000000000001000100
00000010	0000	0001
00000001	0000	00000000000000111100
10000001	10000010000000000000000000000000	



Algoritmo

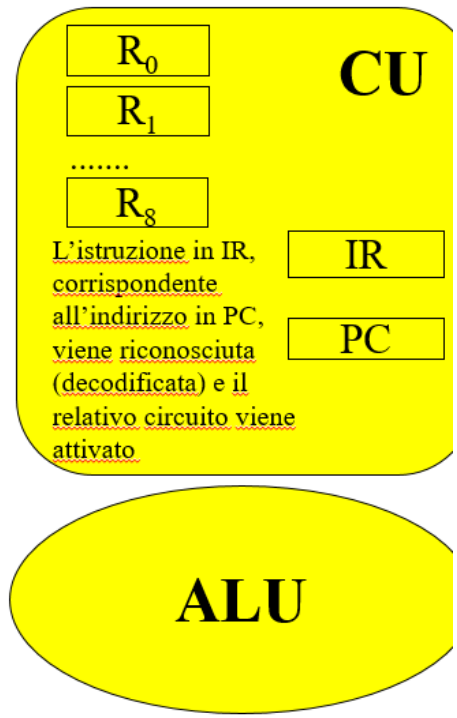
- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100



Programma in LM

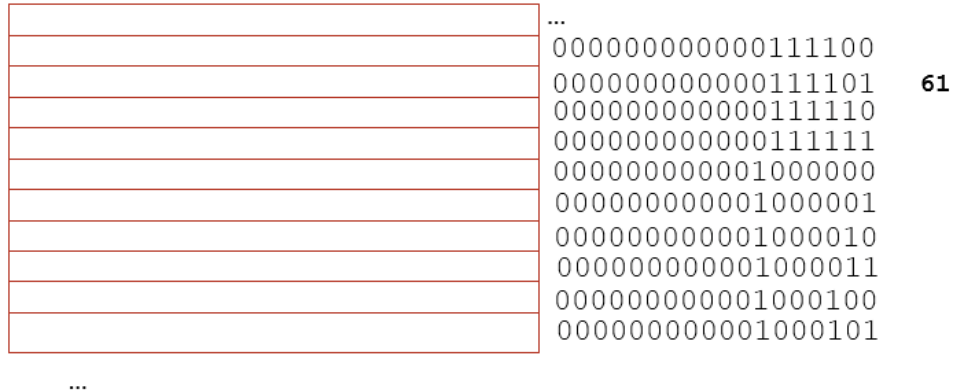
00000000	0000	00000000000001000000
00000000	0001	00000000000001000100
00000010	0000	0001
00000001	0000	00000000000000111100
10000001	10000001000000000000000000000000	

Load 64 in R0



Algoritmo

- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100

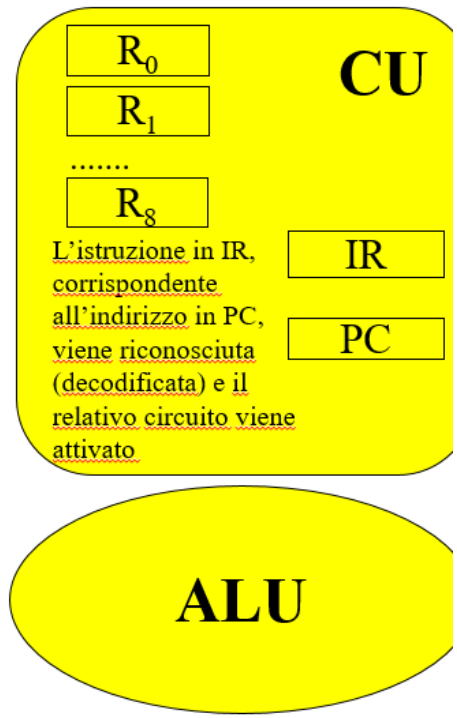


Programma in LM

00000000	0000	00000000000001000000
00000000	0001	00000000000001000100
00000010	0000	0001
00000001	0000	00000000000000111100
10000001	10000001000000000000000000000000	

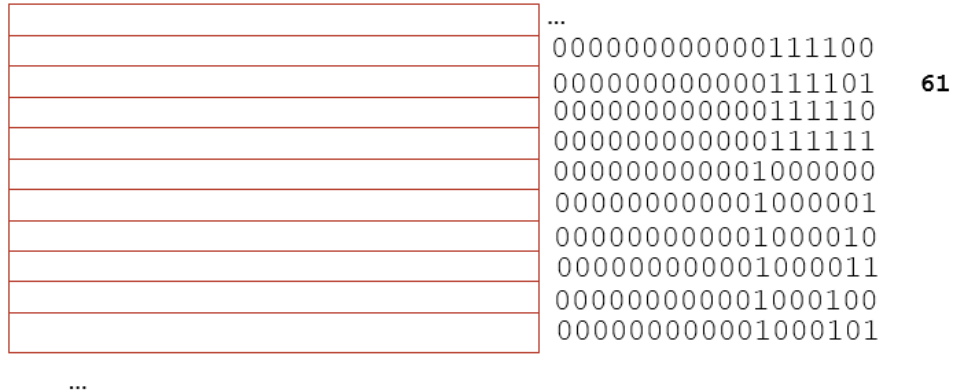
Load 64 in R0

Load 68 in R1



Algoritmo

- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100



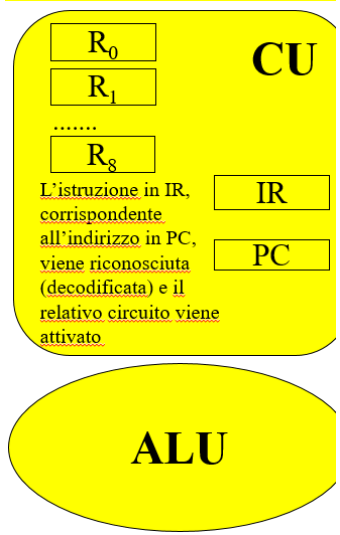
Programma in LM

00000000	0000	0000000000001000000
00000000	0001	0000000000001000100
00000010	0000	0001
00000001	0000	0000000000000111100
10000001	10000001000000000000000000000000	

Load 64 in R0

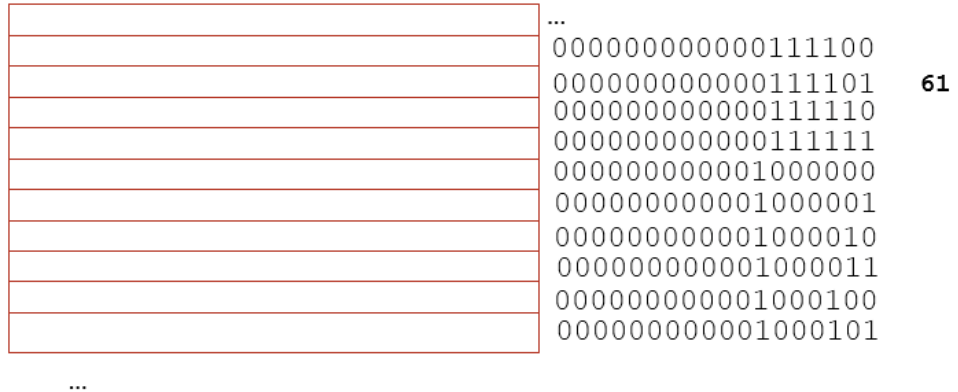
Load 68 in R1

Somma R0+R1 in R0



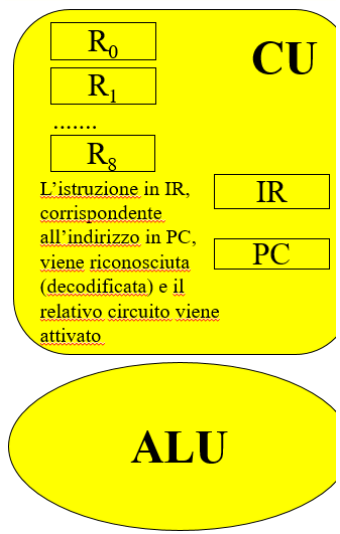
Algoritmo

- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100



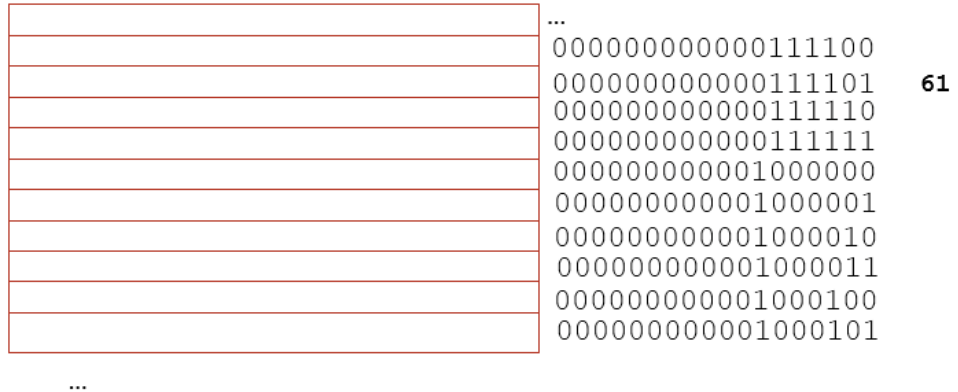
Programma in LM

00000000	0000	00000000000001000000	Load 64 in R0
00000000	0001	00000000000001000100	Load 68 in R1
00000010	0000	0001	Somma R0+R1 in R0
00000001	0000	00000000000000111100	Store R0 in ind.60
10000001	10000001000000000000000000000000		



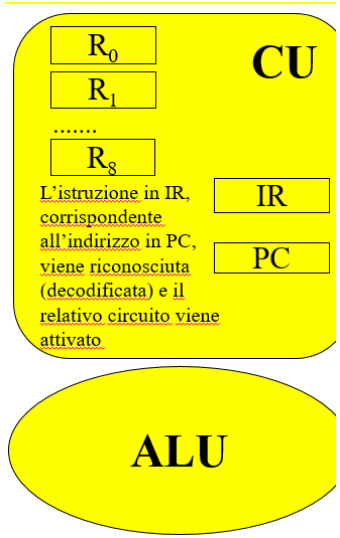
Algoritmo

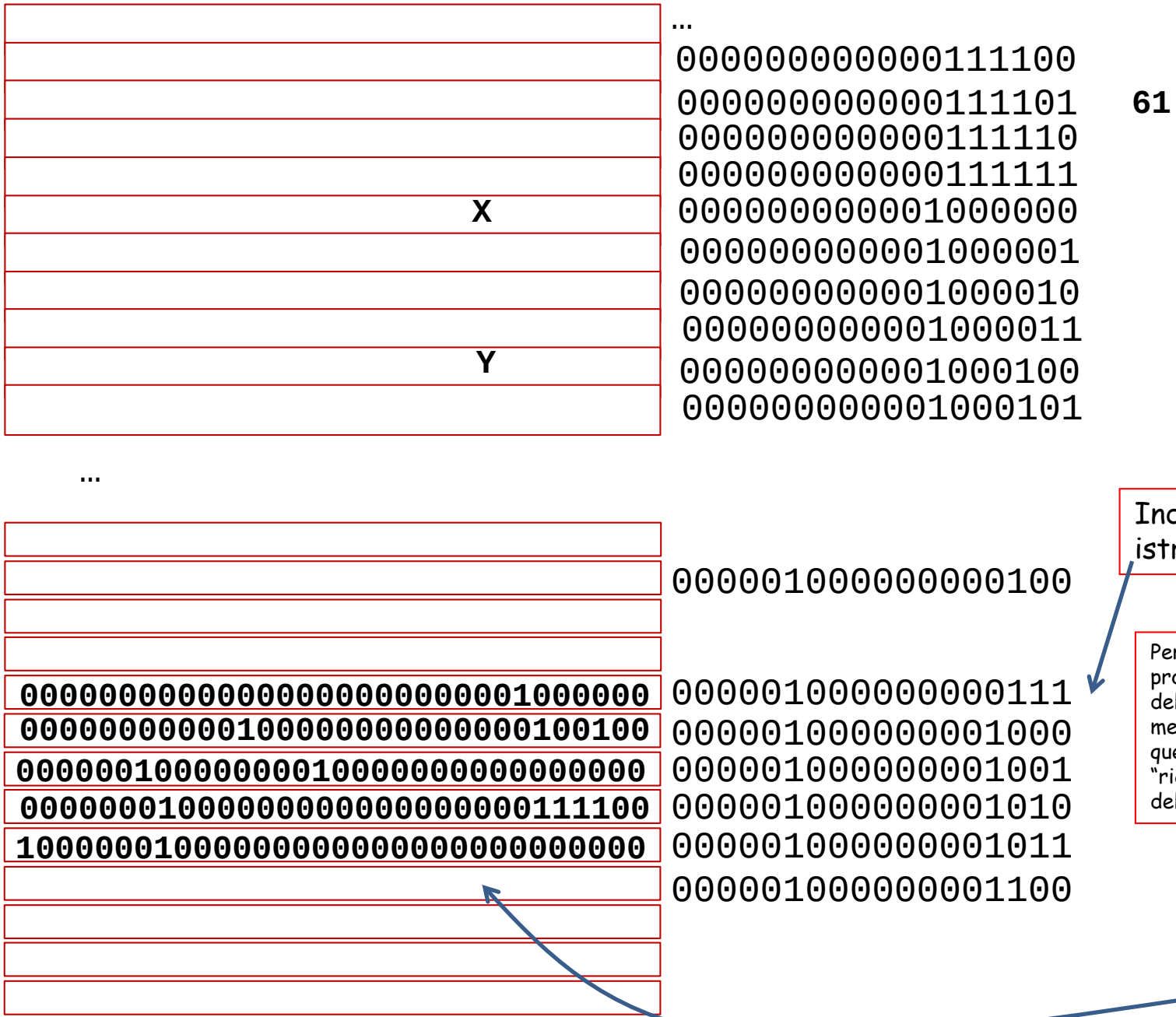
- Si trasferisce il contenuto della locazione 1000000 nel registro 0000;
- Si trasferisce il contenuto della locazione 1000100 nel registro 0001;
- Si esegue la somma tra R0 ed R1 (avendo che il risultato viene memorizzato nel registro R0)
- Si trasferisce il contenuto del registro R0 nella locazione di indirizzo 00111100

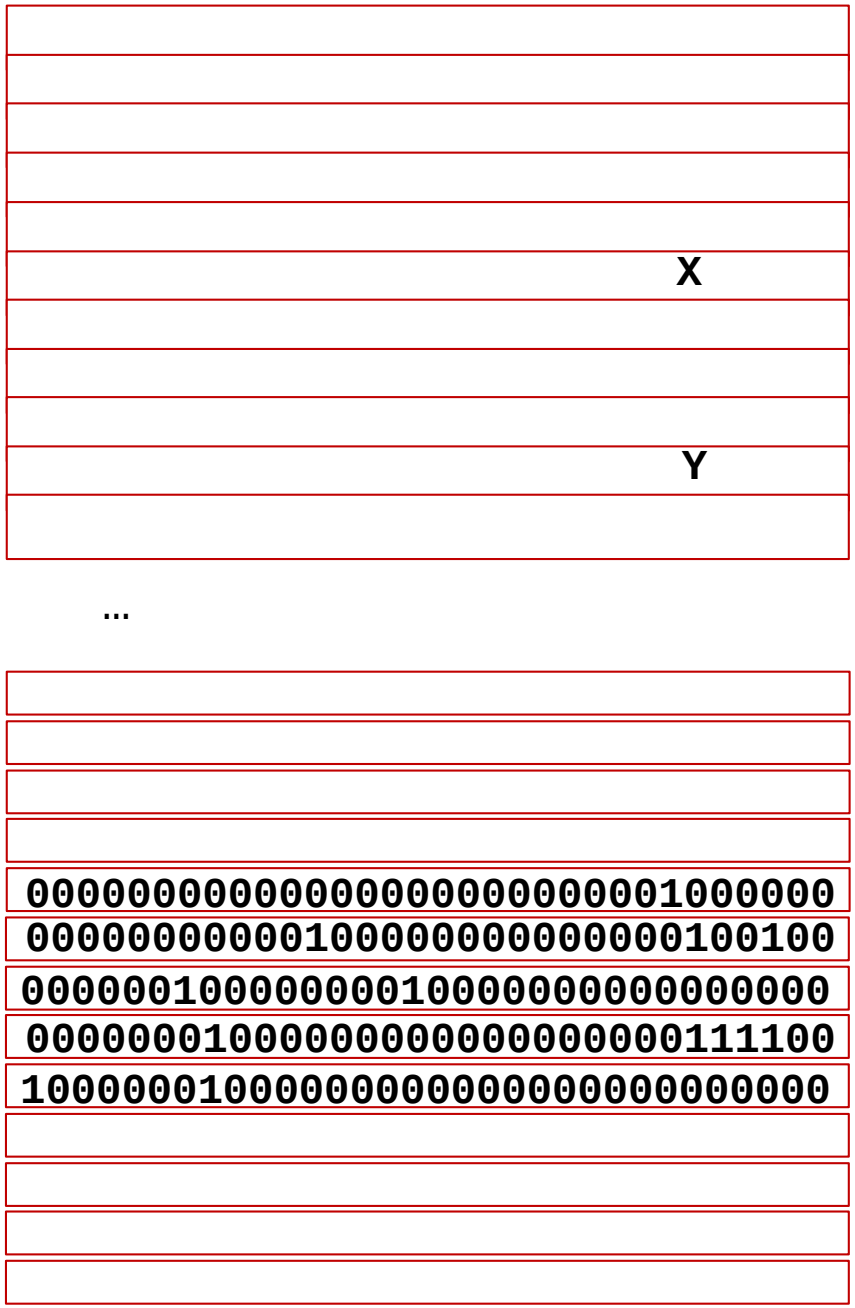


Programma in LM

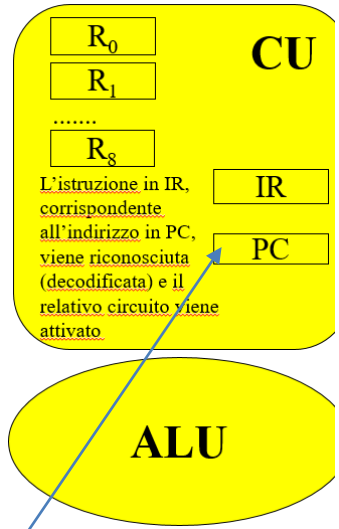
00000000	0000	0000000000001000000	Load 64 in R0
00000000	0001	0000000000001000100	Load 68 in R1
00000010	0000	0001	Somma R0+R1 in R0
00000001	0000	0000000000000111100	Store R0 in ind.60
10000001	10000001000000000000000000000000		Stop




...
0000000000000111100
0000000000000111101
61
0000000000000111110
0000000000000111111
X
0000000000001000000
0000000000001000001
0000000000001000010
0000000000001000011
Y
0000000000001000100
0000000000001000101
...
0000010000000000100
0000000000000000000000001000000
0000010000000000111
0000000000010000000000000100100
0000010000000001000
0000001000000001000000000000000
0000010000000001001
0000000100000000000000000111100
0000010000000001010
1000000100000000000000000000000
0000010000000001011
0000010000000001100
Indirizzi delle istruzioni
Per iniziare ad eseguire il programma, l'indirizzo della prima istruzione va messo nel PC della CPU. E' questa l'operazione di "richiesta di esecuzione del programma", o "RUN"
Programma in linguaggio macchina ...



61



Inizialzzazione dell'esecuzione

RUN

0000010000000000111 → PC

Il programma che inizia li' viene eseguito ripetendo ciclicamente, una volta per ogni istruzione, le seguenti fasi:

FETCH

ISTR in PC → IR

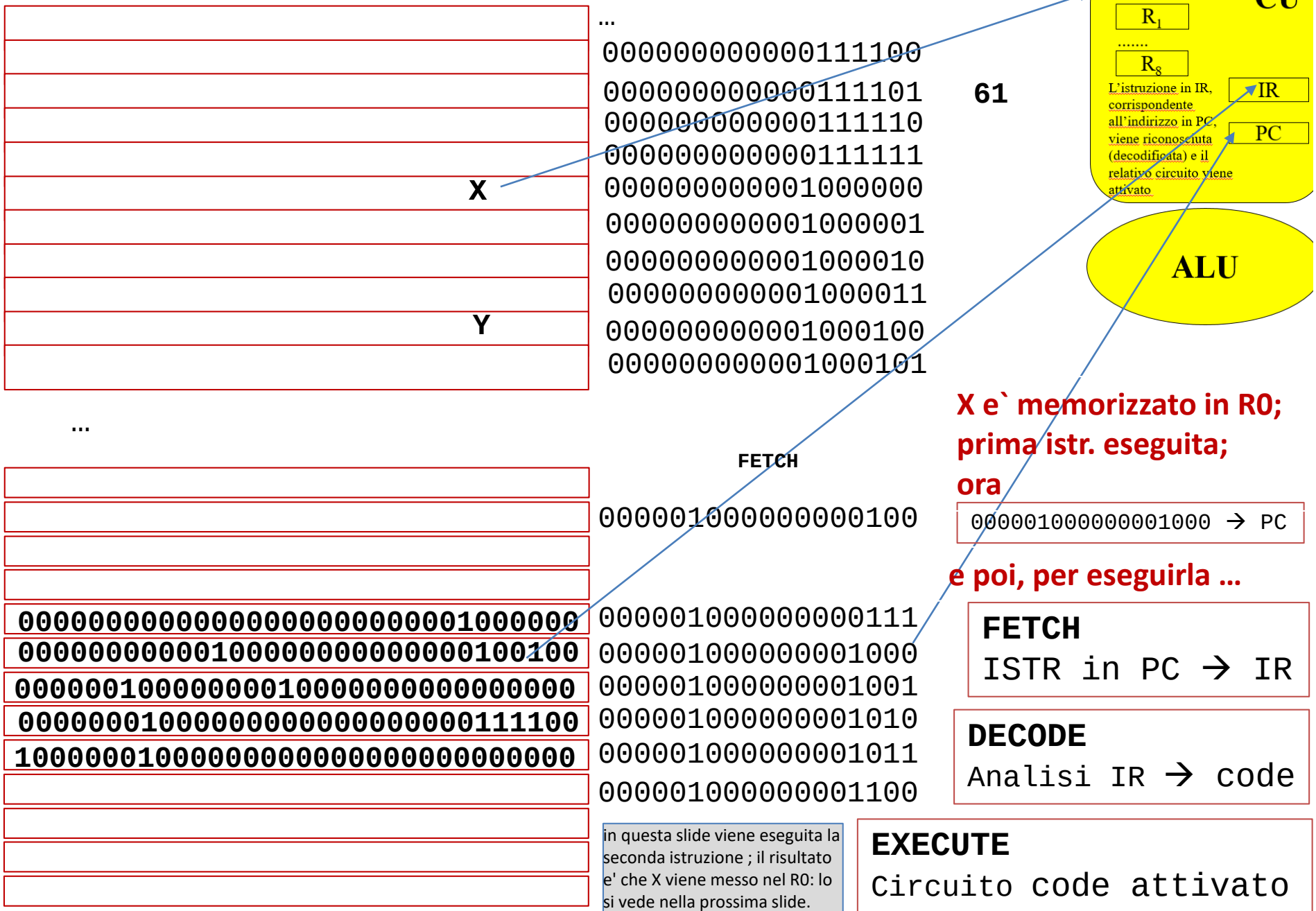
DECODE

Analisi IR → code

EXECUTE

Circuito code attivato

in questa slide viene eseguita la prima istruzione ; il risultato e' che X viene messo nel R0: lo si vede nella prossima slide.



X e` memorizzato in R0; prima istr. eseguita; ora

`0000010000000001000 → PC`

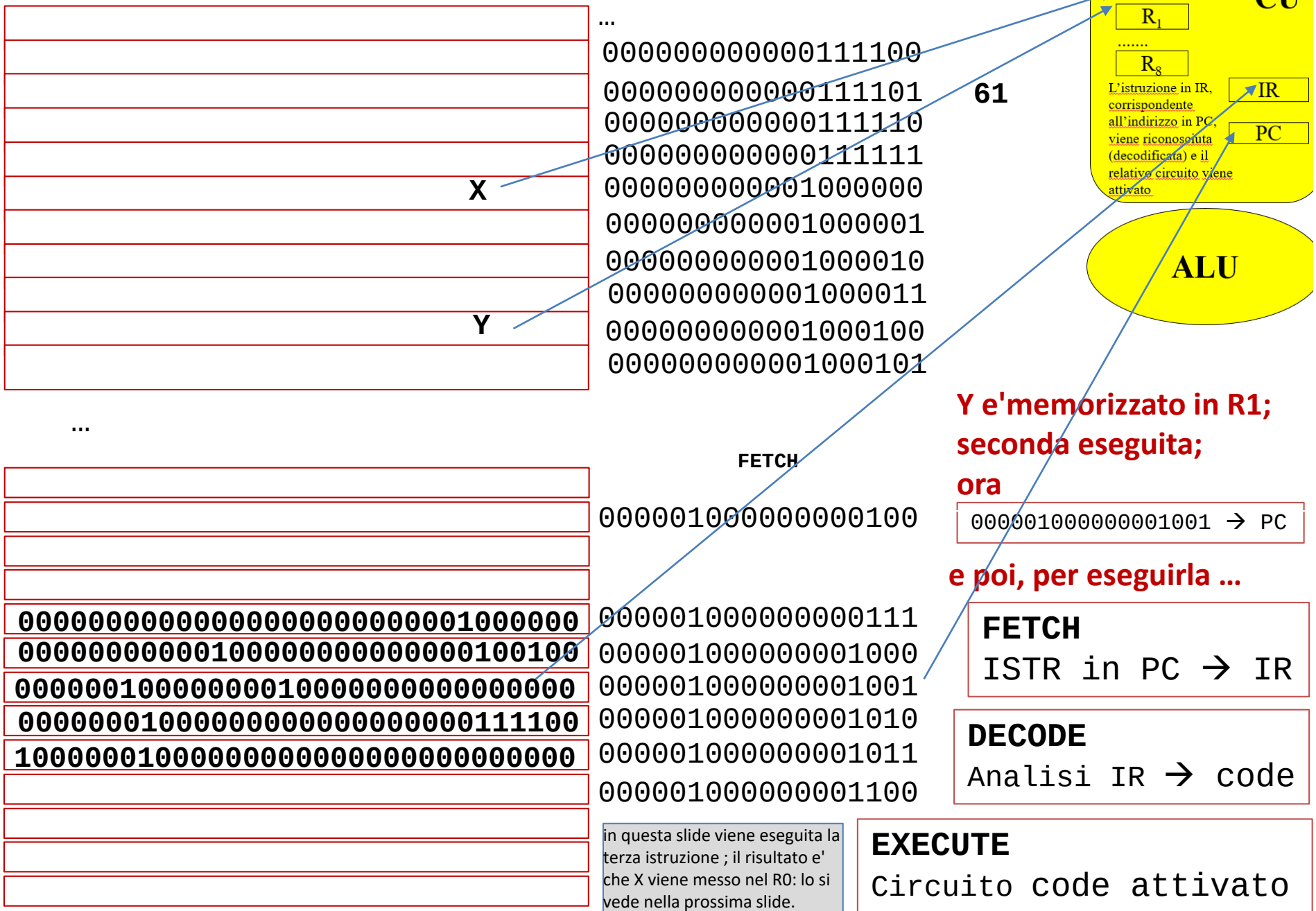
e poi, per eseguirla ...

FETCH
`ISTR in PC → IR`

DECODE
`Analisi IR → code`

EXECUTE
`Circuito code attivato`

in questa slide viene eseguita la seconda istruzione ; il risultato e' che X viene messo nel R0: lo si vede nella prossima slide.



Y e'memorizzato in R1;
seconda eseguita;
ora

```
000001000000001001 → PC
```

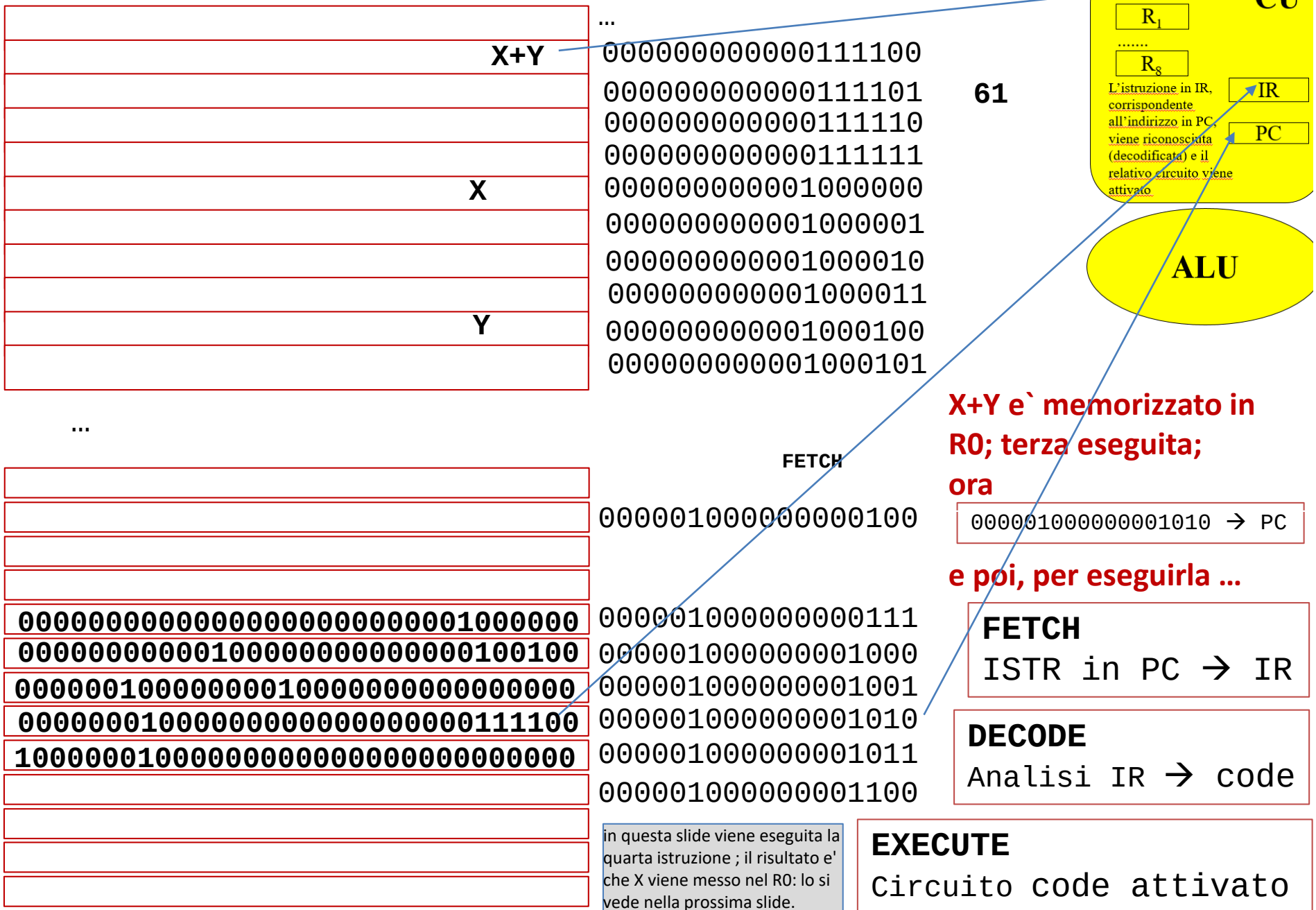
e poi, per eseguirla ...

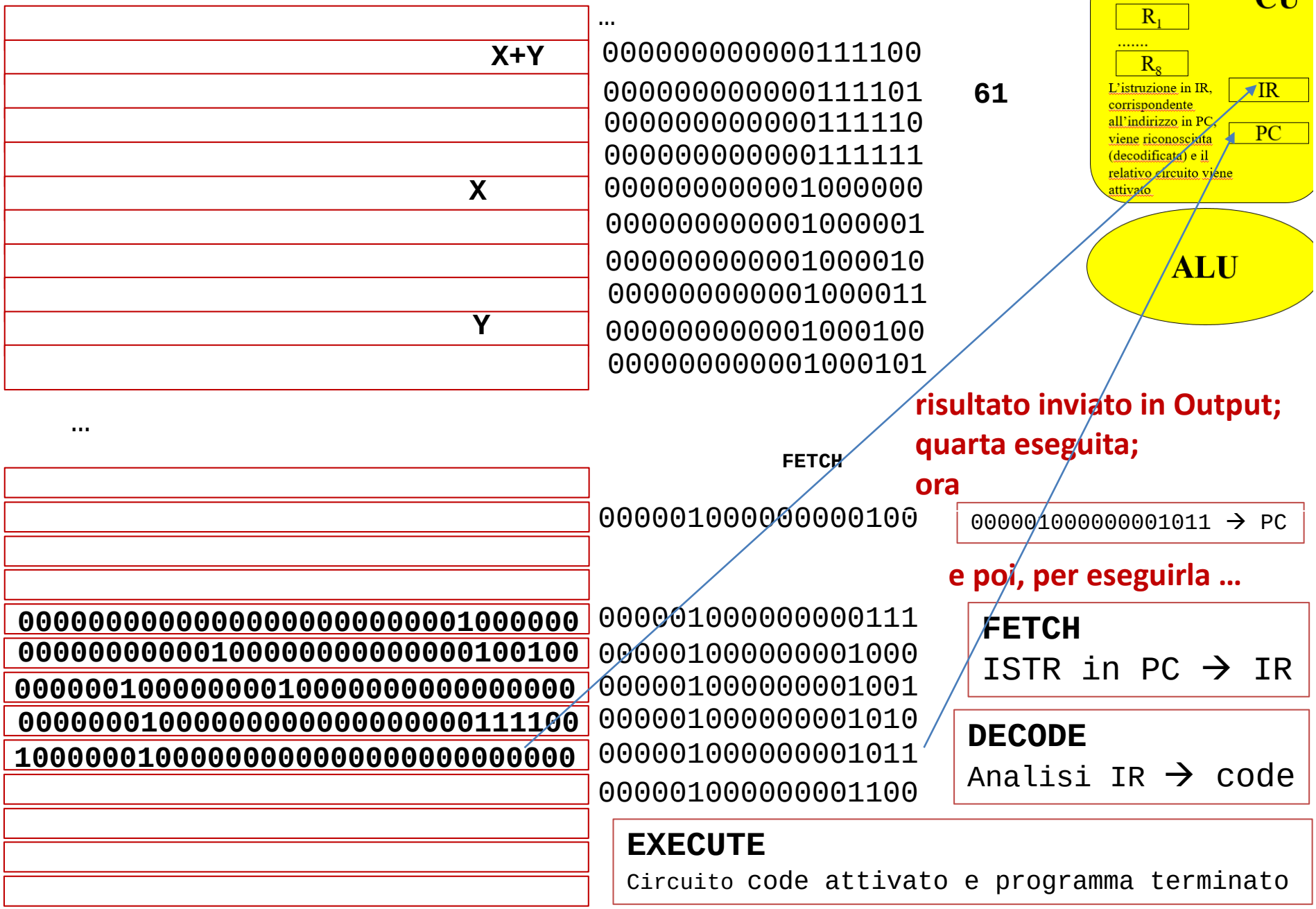
FETCH
ISTR in PC → IR

DECODE
Analisi IR → code

EXECUTE
Circuito code attivato

in questa slide viene eseguita la terza istruzione ; il risultato e' che X viene messo nel R0: lo si vede nella prossima slide.


...
X+Y
X
Y
...
FETCH
0000000000000111100
0000000000000111101
0000000000000111110
0000000000000111111
0000000000001000000
0000000000001000001
0000000000001000010
0000000000001000011
0000000000001000100
0000000000001000101
61
0000010000000000100
0000000000000000000000001000000
0000000000010000000000000100100
0000001000000001000000000000000
0000000100000000000000000111100
1000000100000000000000000000000
0000010000000000111
0000010000000001000
0000010000000001001
0000010000000001010
0000010000000001011
0000010000000001100
R_0
R_1
.....
R_8
CU
L'istruzione in IR, corrispondente all'indirizzo in PC, viene riconosciuta (decodificata) e il relativo circuito viene attivato
IR
PC
ALU
X+Y e` memorizzato in R0; terza eseguita; ora
0000010000000001010 → PC
e poi, per eseguirla ...
FETCH
ISTR in PC → IR
DECODE
Analisi IR → code
EXECUTE
Circuito code attivato
in questa slide viene eseguita la quarta istruzione ; il risultato e' che X viene messo nel R0: lo si vede nella prossima slide.


...
X+Y
X
Y
...
0000000000000111100
0000000000000111101
0000000000000111110
0000000000000111111
0000000000001000000
0000000000001000001
0000000000001000010
0000000000001000011
0000000000001000100
0000000000001000101
61
R0
R1
.....
R8
CU
L'istruzione in IR, corrispondente all'indirizzo in PC, viene riconosciuta (decodificata) e il relativo circuito viene attivato
IR
PC
ALU
FETCH
0000010000000000100
risultato inviato in Output;
quarta eseguita;
ora
0000010000000001011 → PC
e poi, per eseguirla ...
0000000000000000000000001000000
0000000000010000000000000100100
0000001000000001000000000000000
0000000100000000000000000111100
1000000100000000000000000000000
0000010000000000111
0000010000000001000
0000010000000001001
0000010000000001010
0000010000000001011
0000010000000001100
FETCH
ISTR in PC → IR
DECODE
Analisi IR → code
EXECUTE
Circuito code attivato e programma terminato

Cosa succede?
"Ciclo di esecuzione delle istruzioni"

Tre FASI, ripetute fino a quando si arriva a STOP (cioe', in altre parole, ripetute fintantoche' non si arriva a STOP)

- **FETCH** il processore "si procura" l'istruzione da eseguire (ne ha l'indirizzo nel PC, chiede, e la mette nell'IR)

- **DECODE** l'istruzione in IR viene decodificata, cioe' si decide che istruzione e'

- **EXECUTE** l'istruzione in IR viene eseguita, cioe' si attiva il circuito corrispondente (e qui puo' essere che altre componenti del calcolatore vengano coinvolte)

Si', vabbe', ma "in dettaglio?

Quando il programma viene mandato in esecuzione,
- avviene il caricamento (**Loading**) in memoria centrale
- avviene lo **Start** dell'esecuzione, copiando l'indirizzo della prima istruzione nel registro PC (Program Counter).

Poi vengono eseguite ripetutamente le tre fasi ...

FETCH
- la CU (Contro Unit) accede PC, così sa qual è l'indirizzo della prossima istruzione da eseguire;
- il contenuto di PC viene copiato in MAR (Memory Address Register)
- la CU chiede l'accesso alla locazione cha ha indirizzo dato da MAR, attraverso il bus controllo (questa locazione contiene l'istruzione da eseguire)
- la memoria invia il contenuto della locazione (l'istruzione) tramite il bus dati, copiandola nel registro MDR (Memory Data Register)
- la CU copia MDR in IR: adesso IR contiene l'istruzione da eseguire
- PC viene incrementato di 1, così ora punta a quella che dovrebbe essere la prossima istruzione da eseguire

tieni d'occhio le side precedent, con l'evoluzione della esecuzione e le immagini della CPU

Si', vabbe', ma "in dettaglio?

Quando il programma viene mandato in esecuzione,
- avviene il caricamento (Loading) in memoria centrale
- avviene lo START dell'esecuzione, copiando l'indirizzo della prima istruzione nel registro PC (Program Counter).

Poi vengono eseguite ripetutamente le tre fasi ...

FETCH
- la CU (Contro Unit) accede PC, cosi' sa qual e' l'indirizzo della prossima istruzione da eseguire;
- il contenuto di PC viene copiato in MAR (Memory Address Register)
- la CU chiede l'accesso alla locazione cha ha indirizzo dato da MAR, attraverso il bus controllo (questa locazione contiene l'istruzione da eseguire)
- la memoria invia il contenuto della locazione (l'istruzione) tramite il bus dati, copiandola nel registro MDR (Memory Data Register)
- la CU copia MDR in IR: adesso IR contiene l'istruzione da eseguire
- PC viene incrementato di 1, cosi' ora punta a quella che dovrebbe essere la prossima istruzione da eseguire

DECODE
- la CU analizza IR e capisce qual e' il codice dell'istruzione; di conseguenza sa anche quali operandi aspettarsi (numero di registro, indirizzo in RAM ... nulla ...)
- la CU recupera gli operandi con altre richieste sul bus controllo, ricevendo risposta sul bus dati.
- I valori ricevuti vanno nei registri della ALU che sono coinvolti nell'operazione che sta per essere eseguita

Si', vabbe', ma "in dettaglio?

Quando il programma viene mandato in esecuzione,
- avviene il caricamento (Loading) in memoria centrale
- avviene lo START dell'esecuzione, copiando l'indirizzo della prima istruzione nel registro PC (Program Counter).

Poi vengono eseguite ripetutamente le tre fasi ...

FETCH
- la CU (Contro Unit) accede PC, cosi' sa qual e' l'indirizzo della prossima istruzione da eseguire;
- il contenuto di PC viene copiato in MAR (Memory Address Register)
- la CU chiede l'accesso alla locazione cha ha indirizzo dato da MAR, attraverso il bus controllo (questa locazione contiene l'istruzione da eseguire)
- la memoria invia il contenuto della locazione (l'istruzione) tramite il bus dati, copiandola nel registro MDR (Memory Data Register)
- la CU copia MDR in IR: adesso IR contiene l'istruzione da eseguire
- PC viene incrementato di 1, cosi' ora punta a quella che dovrebbe essere la prossima istruzione da eseguire

DECODE
- la CU analizza IR e capisce qual e' il codice dell'istruzione; di conseguenza sa anche quali operandi aspettarsi (numero di registro, indirizzo in RAM ... nulla ...)
- la CU recupera gli operandi con altre richieste sul bus controllo, ricevendo risposta sul bus dati.
- I valori ricevuti vanno nei registri della ALU che sono coinvolti nell'operazione che sta per essere eseguita

EXECUTE
- la CU coordina l'esecuzione, coinvolgendo le parti necessarie (le periferiche di I/O per le operazioni di lettura/scrittura, la ALU per le operazioni di competenza, bus e memoria per gli accessi e memorizzazioni eventuali
- l'esito dell'operazione viene registrato nella PSW (Program Status Word). I bit della PSW dicono, ad esempio, se l'ultima operazione aritmetica ha restituito 0, se c'è stato un riporto, se c'è stato un errore, se c'è un'interrupt (sospensione del programma)

Tecniche della Programmazione, lez. 2, addendum - LM

- Esercizio

esercizio

scrivere un programma macchina che:

- legge da input i dati di un triangolo
- produce in output l'area del triangolo letto

Poi eseguirlo in modo simulato come abbiamo fatto per l'esempio visto durante questa lezione

La soluzione segue ... potete trarne suggerimenti ...

Primo suggerimento: 2 e` gia` memorizzato in 0…0111101 (alternativa ... lo si legge da input)

Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

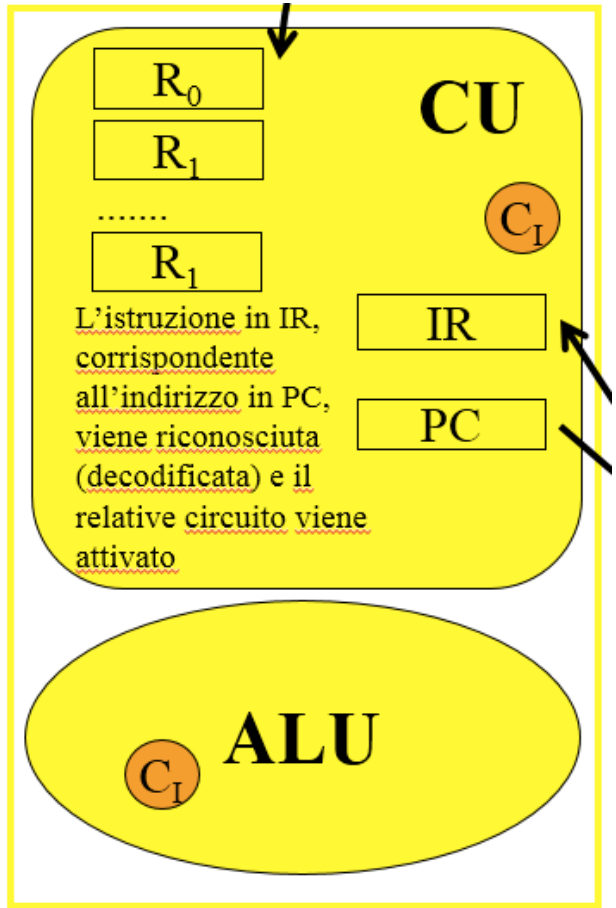
• trasferire il contenuto di 0111101 in R1

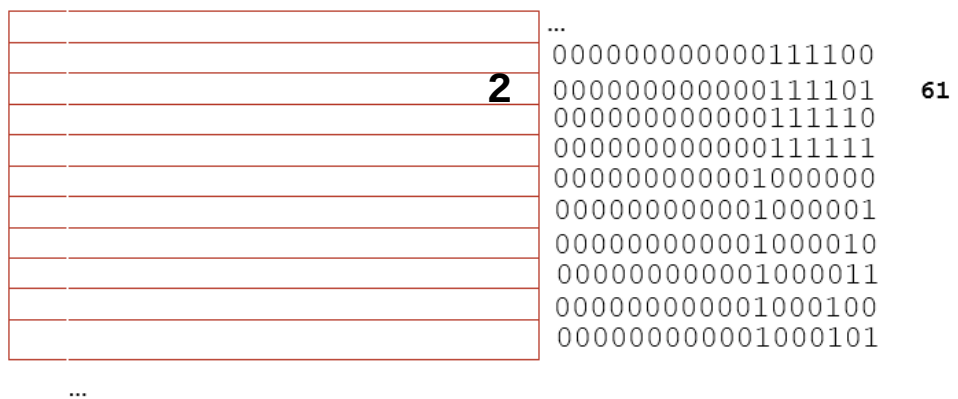
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

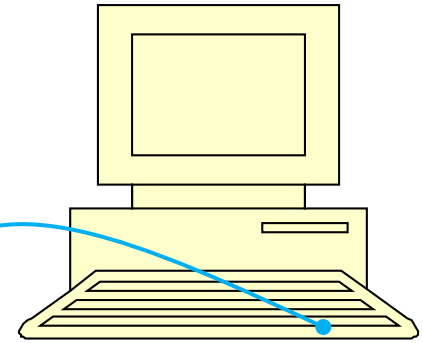
• E poi si termina

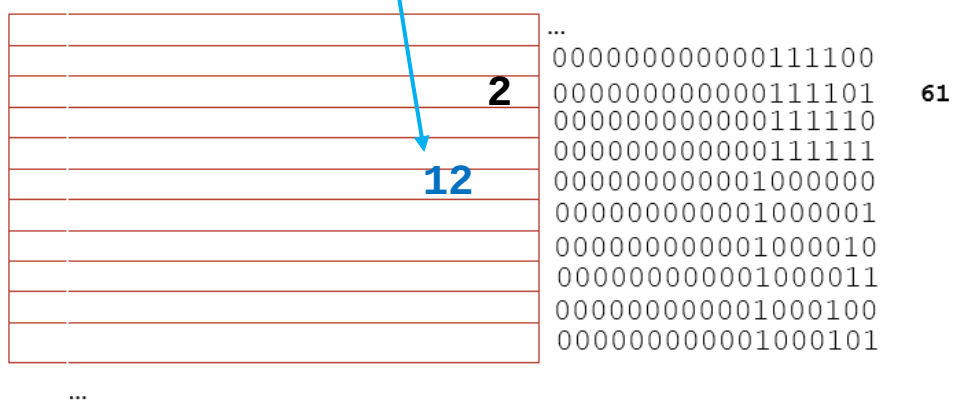




Algoritmo

- ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;
- leggere l'altezza e trasferirla nella locazione 1000100;
- trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);
- Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)
- trasferire il contenuto di 0111101 in R1
- Eseguire divisione (ris. in R0)
- trasferire R0 nella locazione 00111100
- mandare in output (STAMPARE) la locazione 00111100
- E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

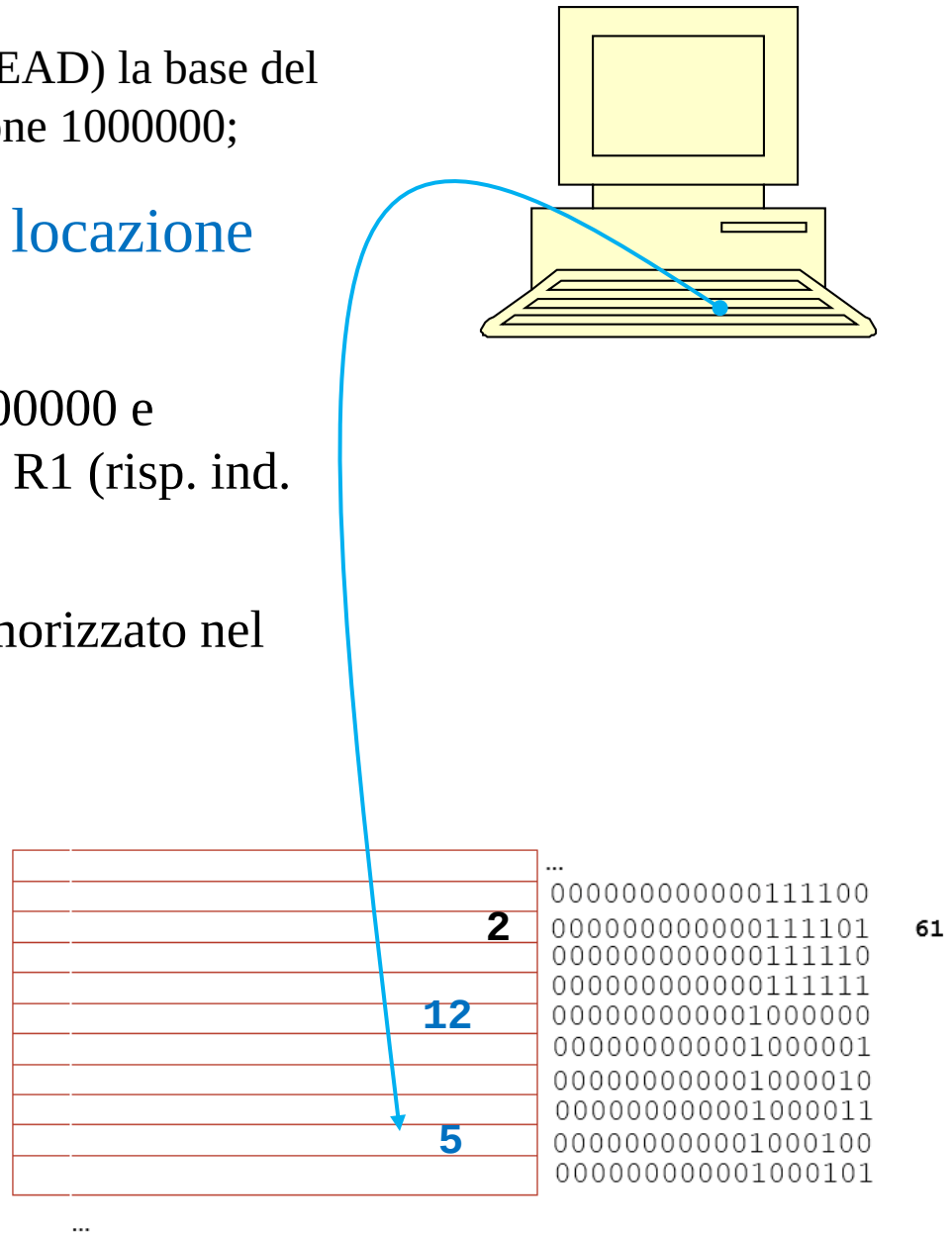
• trasferire il contenuto di 0111101 in R1

• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina



Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

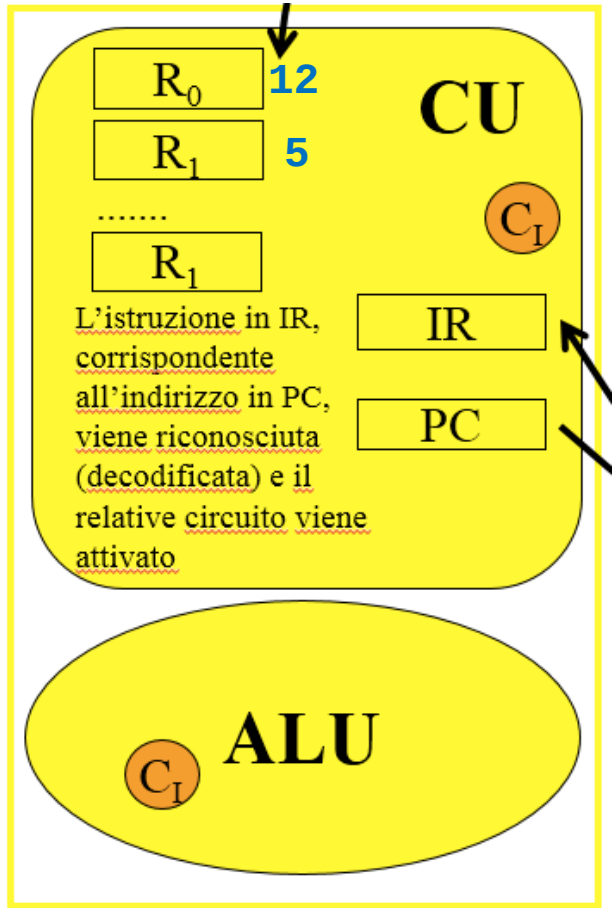
• trasferire il contenuto di 0111101 in R1

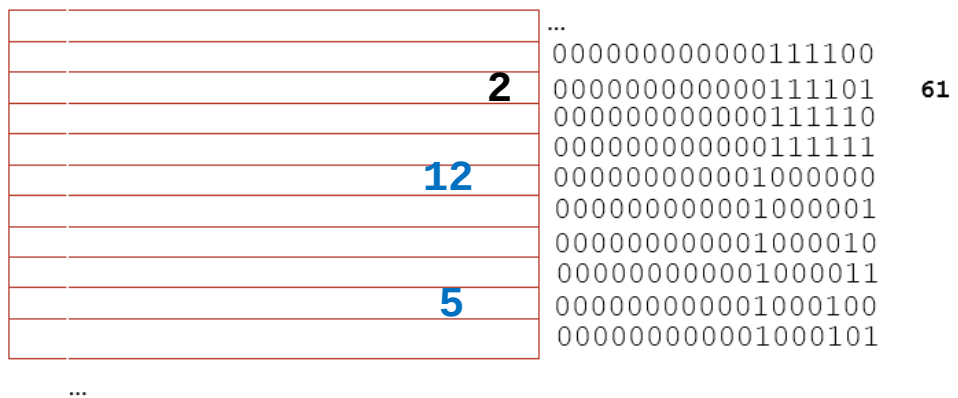
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

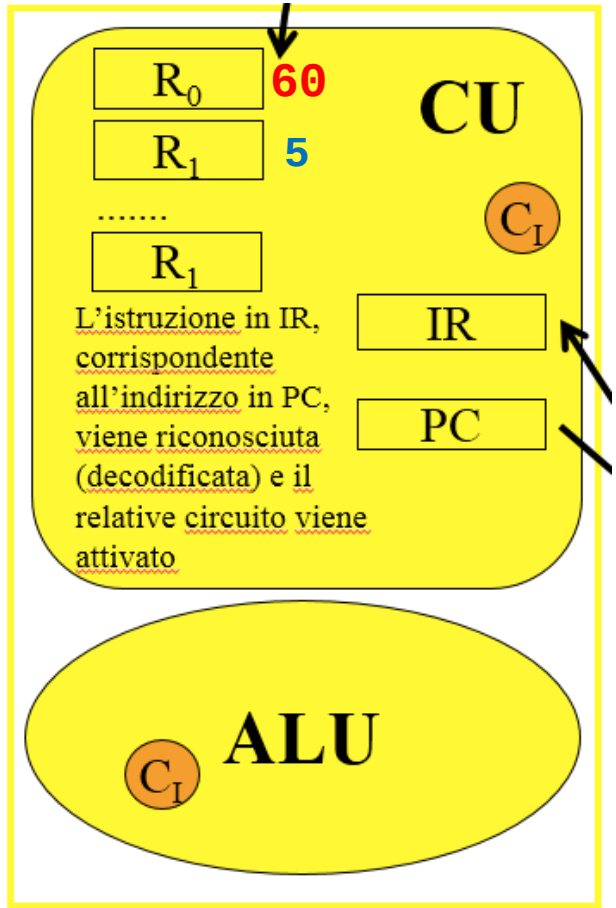
• trasferire il contenuto di 0111101 in R1

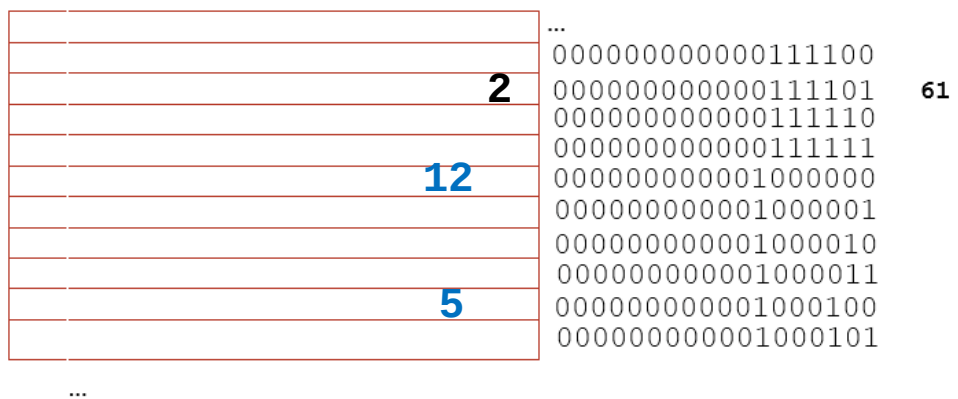
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

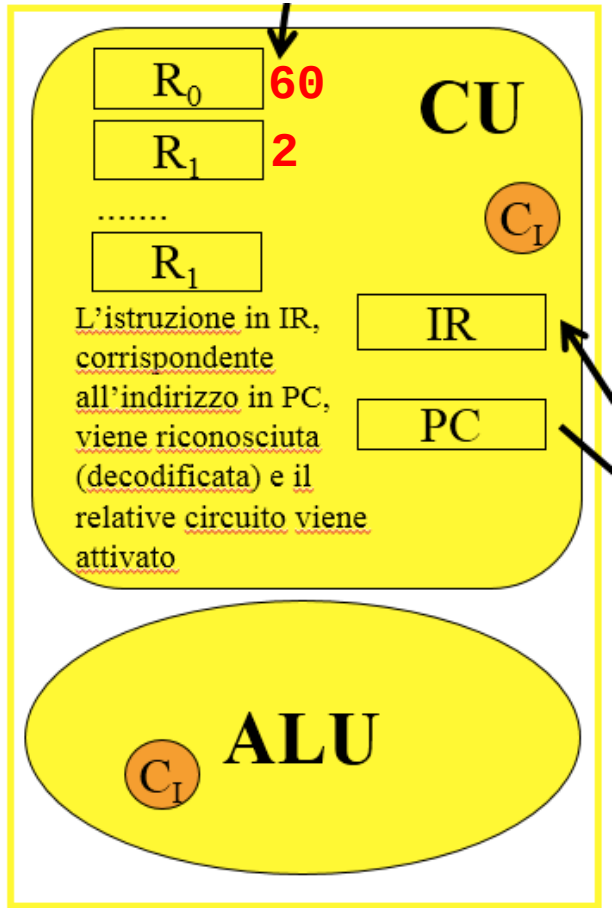
• trasferire il contenuto di 0111101 in R1

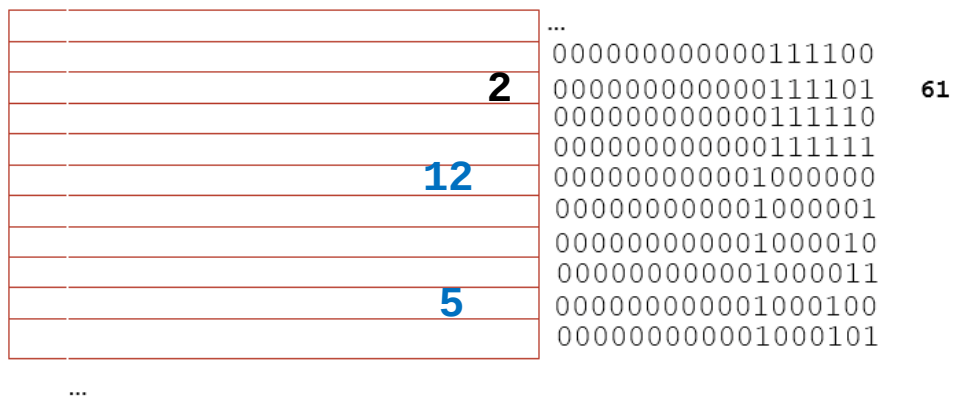
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

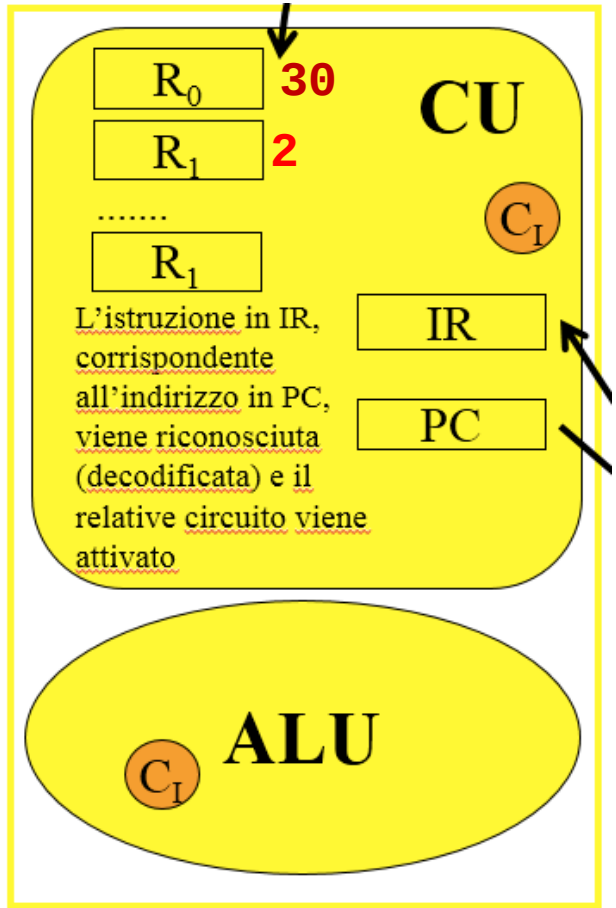
• trasferire il contenuto di 0111101 in R1

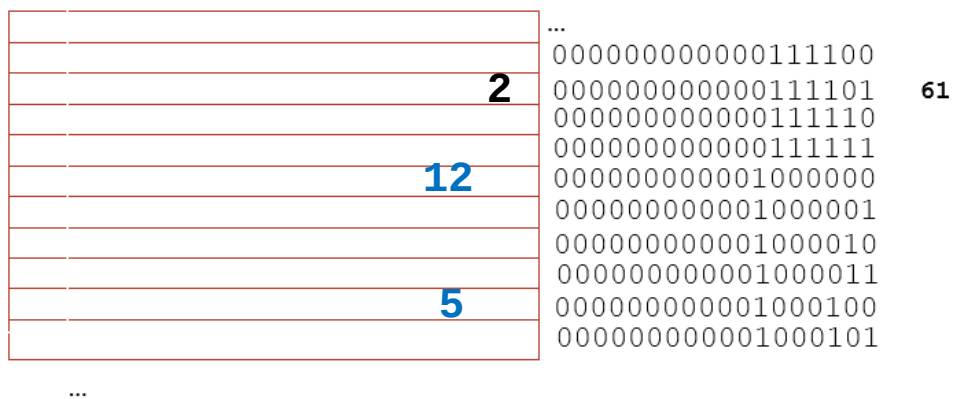
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

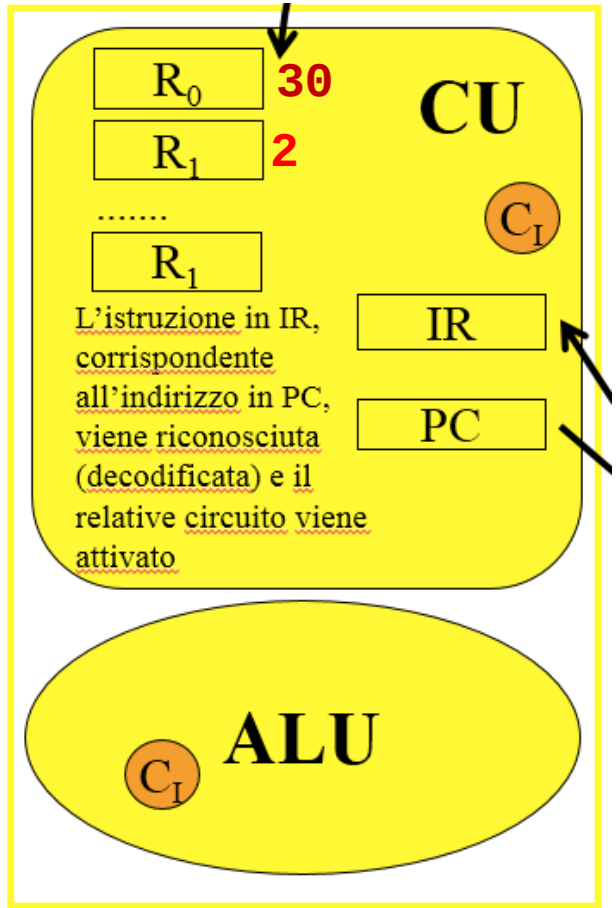
• trasferire il contenuto di 0111101 in R1

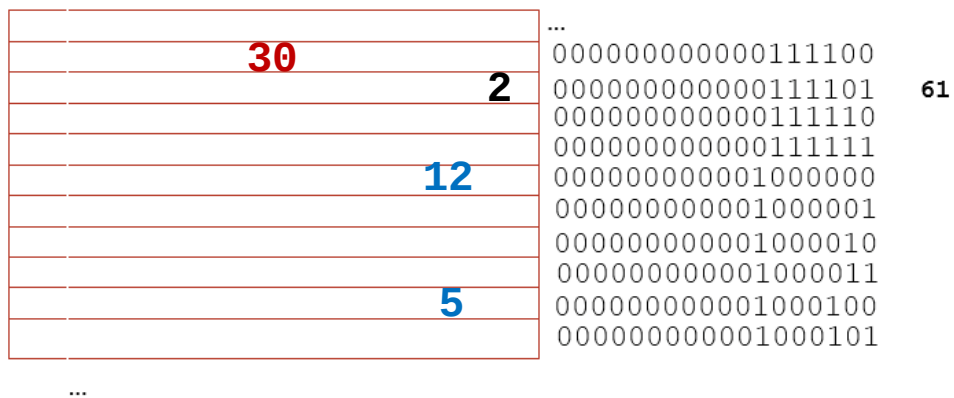
• Eseguire divisione (ris. in R0)

• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina





Algoritmo

• ricevere dall'unita' di INPUT (LEGGERE, READ) la base del triangolo, Trasferendone il valore nella locazione 1000000;

• leggere l'altezza e trasferirla nella locazione 1000100;

• trasferire il contenuto delle locazioni 1000000 e 1000100, rispettivamente nei registri R0 e R1 (risp. ind. 0000 e 0001);

• Eseguire prodotto tra R0 ed R1 (ris. memorizzato nel registro R0)

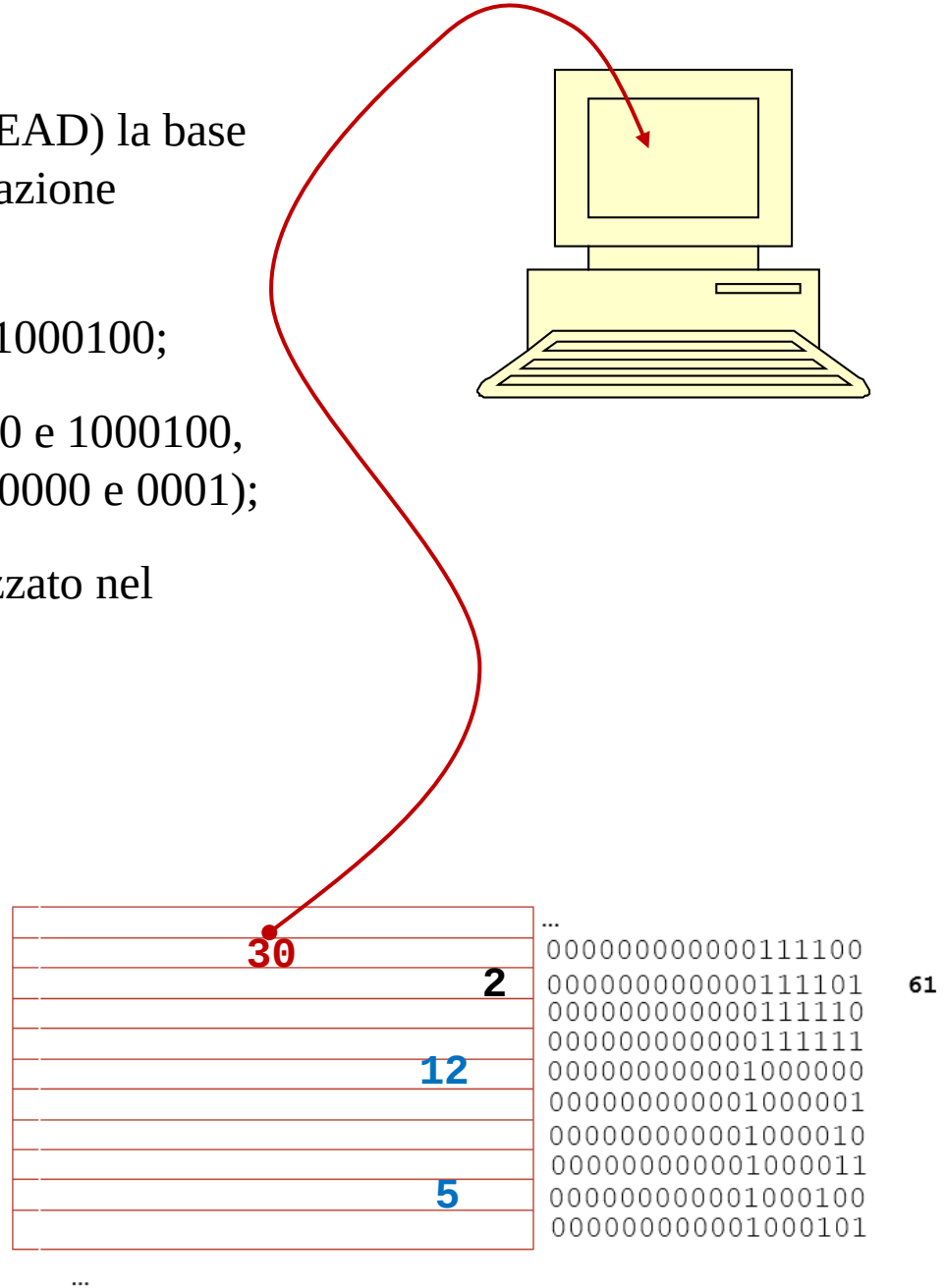
• trasferire il contenuto di 0111101 in R1

• Eseguire divisione (ris. in R0)

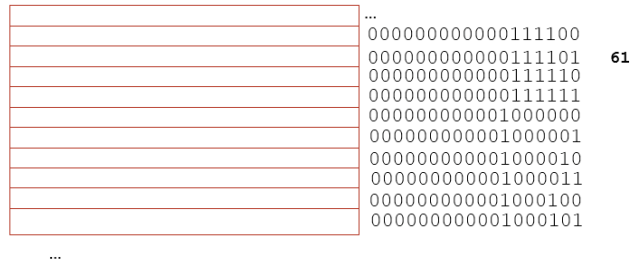
• trasferire R0 nella locazione 00111100

• mandare in output (STAMPARE) la locazione 00111100

• E poi si termina

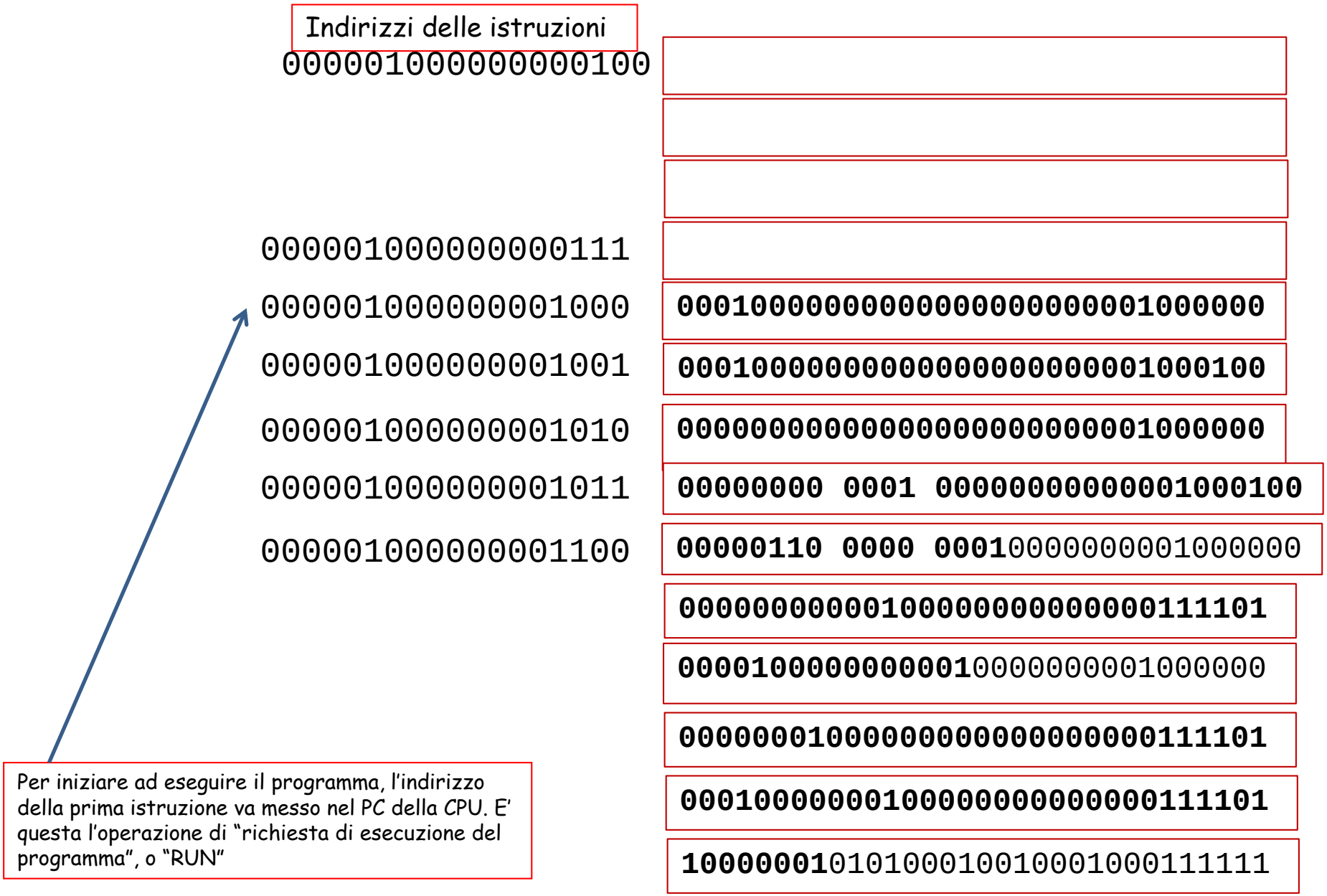


Programma (in LM)
… sulla carta



00010000	0000	0000000000001000000	Read in 64
00010000	0000	0000000000001000100	Read in 68
00000000	0000	0000000000001000000	Load da ind. 64 in R0
00000000	0001	0000000000001000100	Load da ind. 68 in R1
00000110	0000	0001	Prodotto R0 R1 in R0
00000000	0001	0000000000000111101	Load ind. 61 1in R1 (il 2)
00001000	0000	0001	divisione R0 R1 in R0
00000001	0000	0000000000000111100	Store R0 in ind.60
00010000	0001	0000000000000111100	Write da ind. 60
10000001	10000001000000000000000000000000		Stop

Programma (in LM) … in memoria



E ora eseguire il programma come farebbe la CPU

Quando il programma viene mandato in esecuzione,
- avviene il caricamento (Loading) in memoria centrale
- avviene lo START dell'esecuzione, copiando l'indirizzo della prima istruzione nel registro PC (Program Counter).

Poi vengono eseguite ripetutamente le tre fasi ...

FETCH
- la CU (Contro Unit) accede PC, cosi' sa qual e' l'indirizzo della prossima istruzione da eseguire;
- il contenuto di PC viene copiato in MAR (Memory Address Register)
- la CU chiede l'accesso alla locazione cha ha indirizzo dato da MAR, attraverso il bus controllo (questa locazione contiene l'istruzione da eseguire)
- la memoria invia il contenuto della locazione (l'istruzione) tramite il bus dati, copiandola nel registro MDR (Memory Data Register)
- la CU copia MDR in IR: adesso IR contiene l'istruzione da eseguire
- PC viene incrementato di 1, cosi' ora punta a quella che dovrebbe essere la prossima istruzione da eseguire

DECODE
- la CU analizza IR e capisce qual e' il codice dell'istruzione; di conseguenza sa anche quali operandi aspettarsi (numero di registro, indirizzo in RAM ... nulla ...)
- la CU recupera gli operandi con altre richieste sul bus controllo, ricevendo risposta sul bus dati.
- I valori ricevuti vanno nei registri della ALU che sono coinvolti nell'operazione che sta per essere eseguita

EXECUTE
- - la CU coordina l'esecuzione, coinvolgendo le parti necessarie (le periferiche di I/O per le operazioni di lettura/scrittura, la ALU per le operazioni di competenza, bus e memoria per gli accessi e memorizzazioni eventuali
- - l'esito dell'operazione viene registrato nella PSW (Program Status Word). I bit della PSW dicono, ad esempio, se l'ultima operazione aritmetica ha restituito 0, se c'è stato un riporto, se c'è stato un errore, se c'è un'interrupt (sospensione del programma)