

# Esercizio III-DLX

Calcolatori Elettronici L-A

1) Con riferimento al DLX con Datapath sequenziale visto durante il corso si indichi un possibile formato e il diagramma degli stati ottimizzato dell'istruzione

### **LP Ri, Offset**

L'istruzione LP Ri,Offset deve decrementare il contenuto del registro Ri di una unità e nel caso Ri risulti diverso da zero deve trasferire il controllo del programma all'indirizzo specificato dal valore immediato Offset. Nel caso Ri risulti uguale a zero il controllo del programma deve proseguire con l'istruzione successiva.

2) Con riferimento al DLX con Datapath sequenziale visto durante il corso si indichi un possibile formato e il diagramma degli stati ottimizzato dell'istruzione

### **LPR Ri,Rj**

L'istruzione LPR Ri,Rj deve decrementare il contenuto del registro Ri di una unità e nel caso Ri risulti diverso da zero deve trasferire il controllo del programma all'indirizzo specificato dal registro Rj. Nel caso Ri risulti uguale a zero il controllo del programma deve proseguire con l'istruzione successiva.

3) Indicare quali modifiche sarebbe necessario apportare allo schema del DLX pipelined visto durante il corso per realizzare l'istruzioni LP Ri, Offset in 5 stadi.

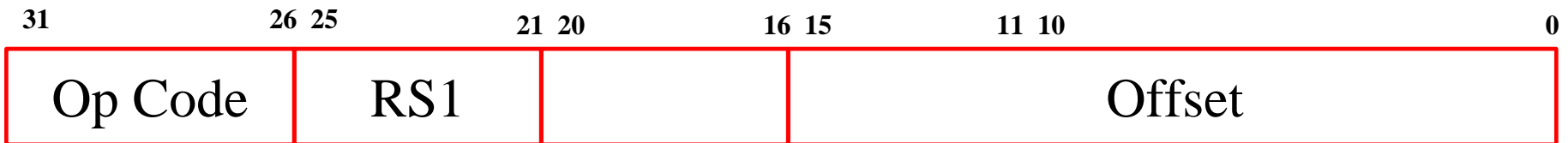
4) Indicare quali modifiche sarebbe necessario apportare allo schema del DLX pipelined visto durante il corso per realizzare l'istruzioni LPR Ri, Rj in 5 stadi.

## Punto 1)

### LP Ri, Offset

L'istruzione **LP Ri, Offset** prevede che il registro Ri sia decrementato di una unità ( $Ri <- Ri - 1$ ). Nel caso il registro Ri non sia zero il flusso di esecuzione del programma deve essere trasferito all'istruzione individuata mediante valore dello scostamento Offset. Nel caso il registro Ri sia zero deve essere eseguita l'istruzione successiva alla LP.

Il codice operativo è codificato con 6 bit, il registro implicato nell'operazione (Ri) è codificato con 5 bit (32 possibili registri) e l'offset è codificato con 16 bit.

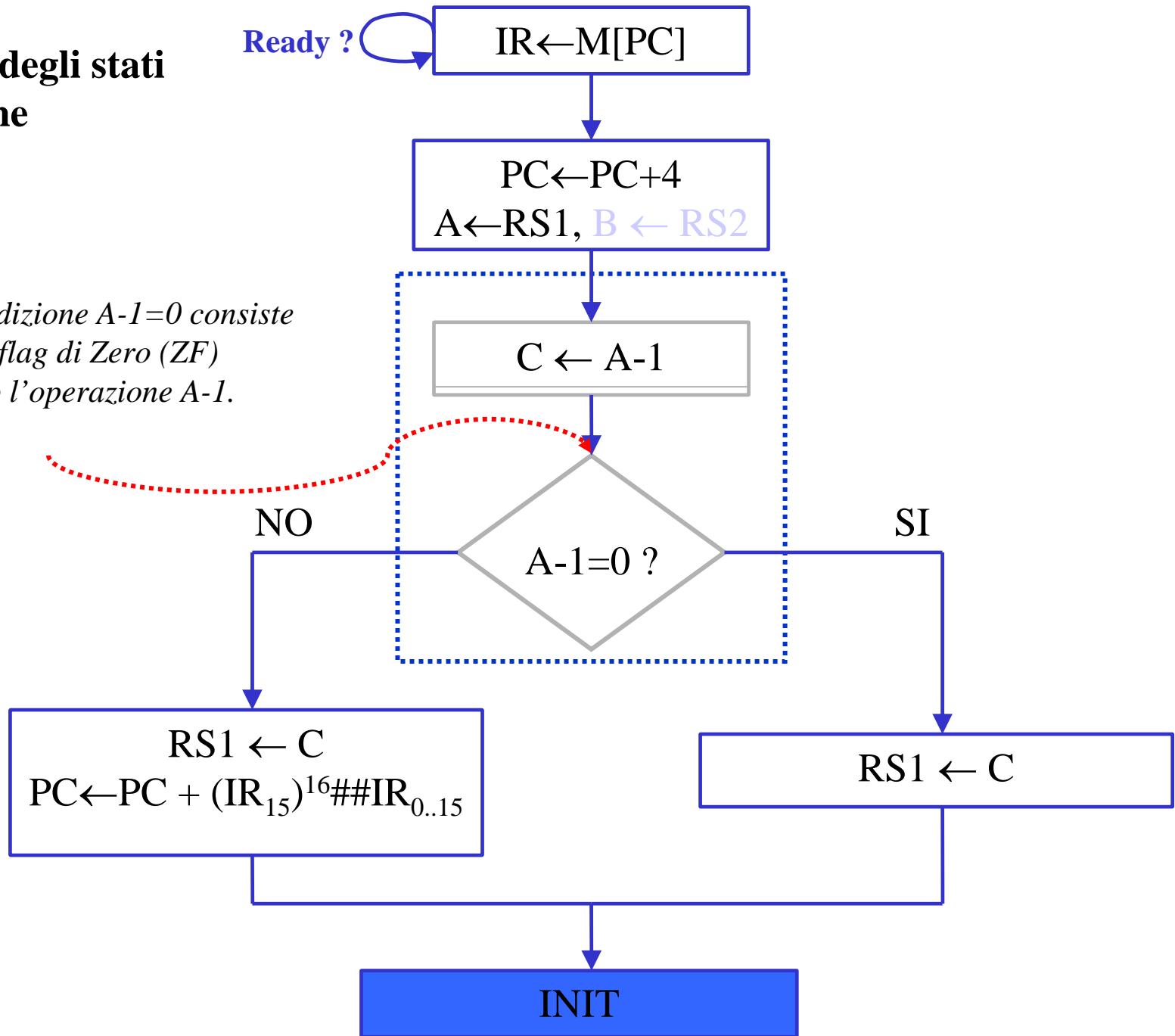


Il diagramma degli stati di questa istruzione è riportato nel lucido successivo.

# Diagramma degli stati dell'istruzione

LP Ri, Offset

*Il test della condizione  $A-1=0$  consiste nell'analisi del flag di Zero (ZF) della ALU dopo l'operazione  $A-1$ .*

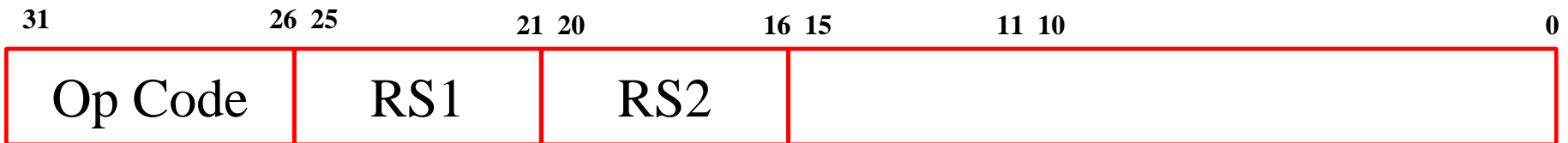


## Punto 2)

### LPR Ri, Rj

L'istruzione **LPR Ri, Rj** prevede che il registro Ri sia decrementato di una unità ( $Ri \leftarrow Ri - 1$ ). Nel caso il registro Ri non sia zero il flusso di esecuzione del programma deve essere trasferito all'istruzione individuata dall'indirizzo presente in Rj. Nel caso il registro Ri sia zero deve essere eseguita l'istruzione successiva alla LPR.

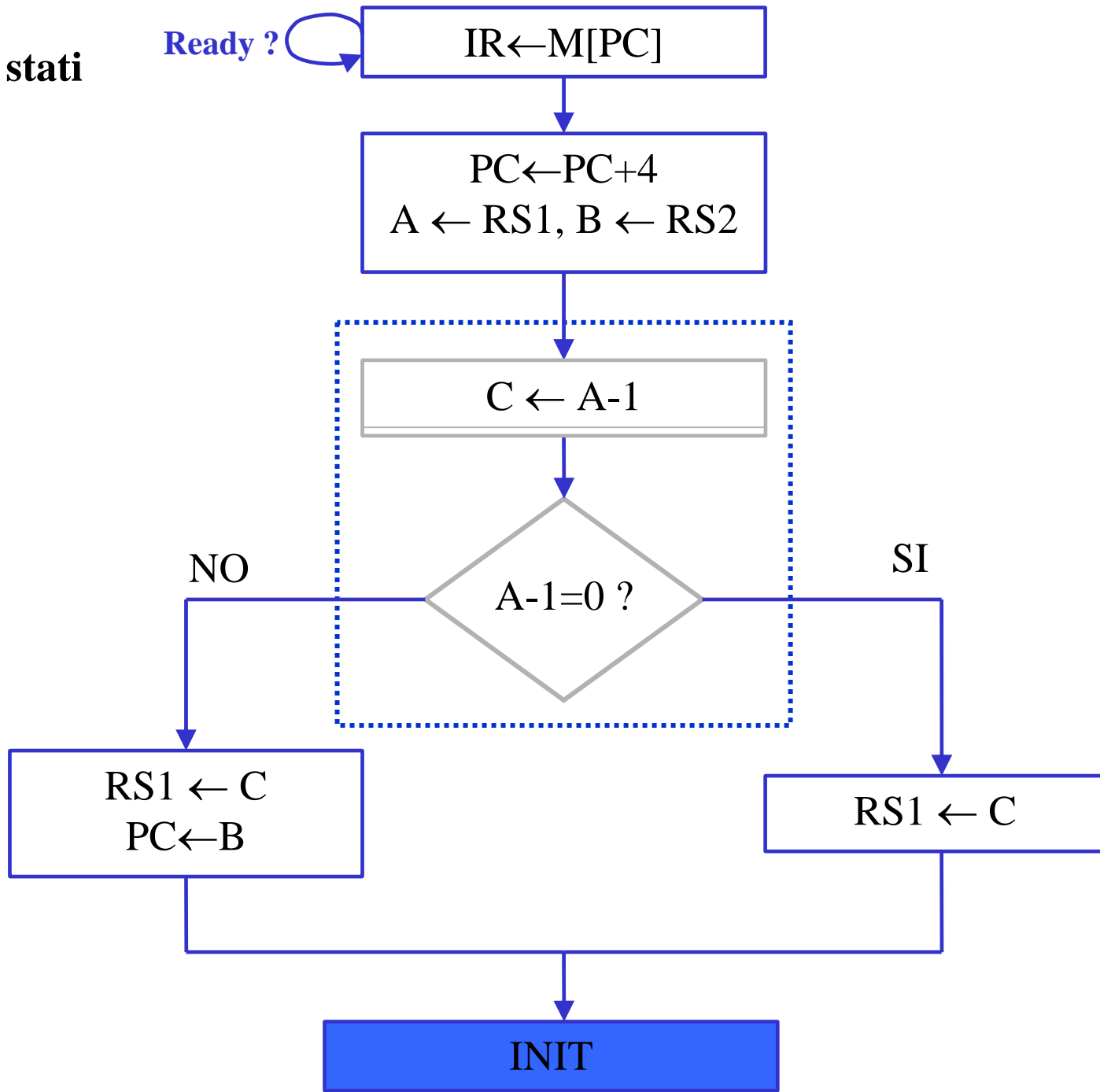
Il codice operativo è codificato con 6 bit, i registri implicati nell'operazione (Ri,Rj) sono codificati con 5 bit (32 possibili registri).



Il diagramma degli stati di questa istruzione è riportato nel lucido successivo.

**Diagramma degli stati  
dell'istruzione**

**LPR Ri,Rj**



### **Punto 3)**

Con riferimento al DLX pipelined visto a lezione. Poichè nello stadio di EX l'istruzione LP Ri,Offset esegue due operazioni che coinvolgono la ALU (decremento registro Ri e calcolo dell'indirizzo di destinazione) per poter eseguire l'istruzione LP Ri,Offset in 5 stadi sarebbe necessario introdurre nello stadio EX un'ulteriore rete logica (ALU) per poter eseguire le due operazioni in parallelo.

### **Punto 4)**

Nel caso dell'istruzione LPR Ri,Rj non è necessario apportare modifiche al datapath del DLX pipelined visto durante il corso.

5) Supponendo che R1 contenga l'indirizzo iniziale di un vettore di BYTE di 255 elementi. Si scriva il codice assembler DLX che inserisce il BYTE memorizzato nel registro R7 nei primi dieci elementi del vettore

- a) non utilizzando le istruzioni LP e LPR
- b) utilizzando una delle due nuove istruzioni (LP o LPR)

## Punto 5-a)

Codice in assembler DLX che non utilizza istruzioni LP e LPR

```
ADDI    R2,R0,9           ; R2←9, inzializzazione

CICLO:  ADD    R3,R1,R2     ; R3←R1+R2
        SB    0(R3),R7     ; mem. nella R2-esima posizione
                               ; del vettore il byte in R77..0
                               ; (MEM[R3] ←R77..0)

        SUBI   R2,R2,1     ; R2←R2-1
        BNEQZ R2, CICLO   ; se R2!=0 salta a CICLO

        SB    0(R1),R7     ; memorizza ultimo elemento
                               ; in posizione 0
```

## Punto 5-b)

Codice in assembler DLX che utilizza l'istruzione LP

```
    ADDI    R2,R0,9           ; R2←9, inzializzazione

CICLO: ADD    R3,R1,R2       ; R3←R1+R2
    SB      0(R3),R7         ; nella R2-esima posizione
                                ; del vettore il byte in R77..0
                                ; (MEM[R3] ←R77..0)

    LP      R2, CICLO

    SB      0(R1),R7         ; memorizza ultimo elemento
                                ; in posizione 0
```