Lezione 21
Il Processore: Unità di Elaborazione (3)

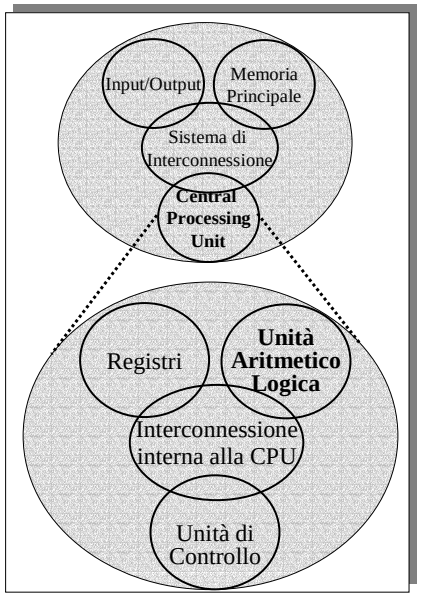
Vittorio Scarano

Architettura

Corso di Laurea in Informatica

Università degli Studi di Salerno

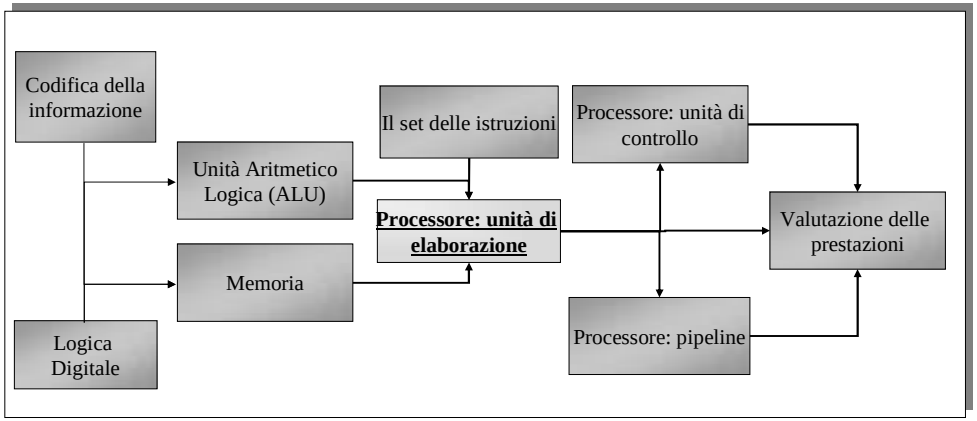
Un quadro della situazione



- Cosa abbiamo fatto…
  - Il Set di Istruzioni
- Dove stiamo andando..
  - progettazione del processore
- Perché:
  - per poter capire cosa deve offrire al programmatore il processore come istruzioni

Dove siamo nel corso…

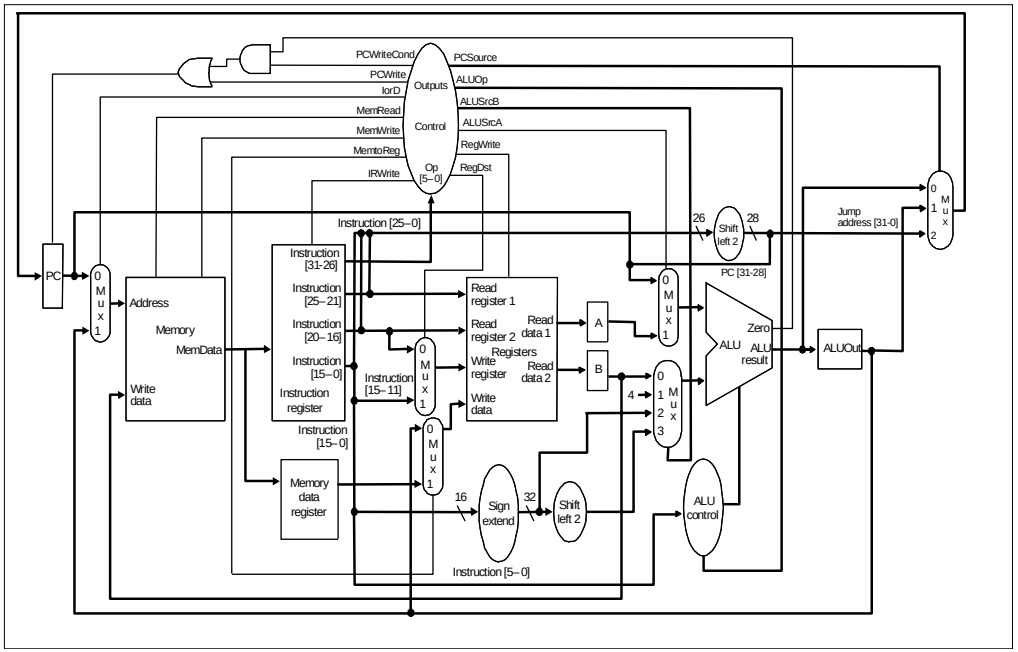
- Progettazione: unità di elaborazione



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

La unità di elaborazione multi-ciclo



I segnali di controllo a 1 bit

Segnale	Effetto quando vale 0	Effetto quando vale 1
RegDst	Registro destinazione = rt	Registro destinazione = rd
RegWrite	Nessuno	Nel registro indicato sull'ingresso *WriteRegister* viene scritto il valore *WriteData*
ALUSrcA	Il primo oper. di ALU è PC	Il primo oper. di ALU è registro A
MemRead	Nessuno	Lettura della locazione indicata dall'*Indirizzo*
MemWrite	Nessuno	Scrittura della locazione indicata dall'*Indirizzo*
MemtoReg	Il valore in *WriteData* viene da ALUOut	Il valore in *WriteData* viene da MDR
IorD	L'indirizzo proviene da PC	L'indirizzo proviene da ALUOut
IRWrite	Nessuno	Uscita memoria scritta in IR
PCWrite	Nessuno	Scrittura in PC: provenienza da *PCSource*
PCWriteCond	Nessuno	Scrittura in PC se *Zero* di ALU è 1

I segnali di controllo a 2 bit

Segnale	Val	Effetto
ALUOp	00	La Alu calcola una somma
	01	La Alu calcola una sottrazione
	10	Operazione determinata dal campo funct
ALUSrcB	00	Il secondo ingresso della ALU: proviene dal registro B
	01	…: è la costante 4
	10	…: il valore dei 16 bit meno sign. di IR, estesi a 32 bit
	11	…: il valore dei 16 bit meno sign. di IR, estesi a 32 bit, shiftati a sinistra di 2 bit
PCSource	00	In PC viene scritta l'uscita della ALU (PC+4)
	01	In PC viene scritto ALUOut (indirizzo della BEQ)
	10	In PC viene scritto indirizzo del salto incondizionato (26 bit meno significativi, scalati a sinistra di 2 bit e concatenato con i 4 bit più significativi di PC

Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

Esecuzione di una istruzione in più cicli

- L'obiettivo:
  - decomporre le istruzioni in modo da bilanciare il carico di lavoro in ogni ciclo
- Suddividiamo le istruzioni in una serie di passi
  - ogni passo coincide con un ciclo di clock
  - in modo che ogni passo contenga, al più
    - una operazione con la ALU
    - un accesso al banco dei registri
    - un accesso alla memoria
- In questa maniera il ciclo di clock sarà pari alla più lenta di queste tre operazioni

Cosa accade alla fine di ciascun ciclo

- Tutti i valori che sono richiesti nel passo successivo
  - sono memorizzati in qualche componente
- Ad esempio:
  - il program counter (PC)
  - il banco dei registri
  - la memoria
  - uno dei registri addizionali
    - A, B, MDR oppure ALUOut
    - IR (dotato di segnale di controllo per la scrittura)
      - che è l'unico dei registri addizionali che contiene lo stesso dato per tutti i passi di esecuzione di una istruzione

La scrittura nei registri

- Distinguiamo tra
  - la scrittura in un registro (PC, A, B etc.)
  - la scrittura nel banco dei registri
    - che necessita di un ciclo di clock aggiuntivo
- Infatti, il banco dei registri ha una logica di controllo ed un corrispondente tempo di accesso superiore a quello dei registri singoli
  - quindi per ridurre il ciclo di clock, dobbiamo necessariamente richiedere più cicli di clock per poter completare gli accessi al banco dei registri

I passi di esecuzione di una istruzione

- Una prima presentazione
- 5 passi:
  - passo di prelievo della istruzione
    - si carica IR e si incrementa il PC
  - passo di decodifica dell'istruzione e caricamento registri
    - si leggono i due registri e si calcola (comunque!) l'indirizzo del salto condizionato
  - passo di esecuzione, calcolo indirizzo memoria e salto
    - primo passo realmente dipendente dalla istruzione da eseguire
  - passo di accesso alla memoria o completamento istruzione R
  - passo di completamento lettura dalla memoria

Come indicare le operazioni compiute in un ciclo

- Usiamo una notazione per indicare le operazioni
  - *Register Transfer Language* (RTL)
- Semplice sintassi. Alcuni esempi:
  - IR = Memoria[PC]
    - trasferimento della locazione di memoria indicizzata dal PC in IR
  - PC = PC + 4
    - incremento del program counter di 4
  - IR[25-21]
    - indica i 5 bit di IR dal bit 25 al bit 21
  - Reg[IR[25-21]]
    - registro indirizzato dai bit di IR dal 25 al 21

Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

1 - Passo di prelievo della istruzione (a)

- Obiettivo:
  - caricare la istruzione ed incrementare il program counter
- Passo comune a tutte le istruzioni
  - non facciamo alcuna differenza
    - non potremmo, non avendo ancora letto la istruzione!
- Le operazioni eseguite:

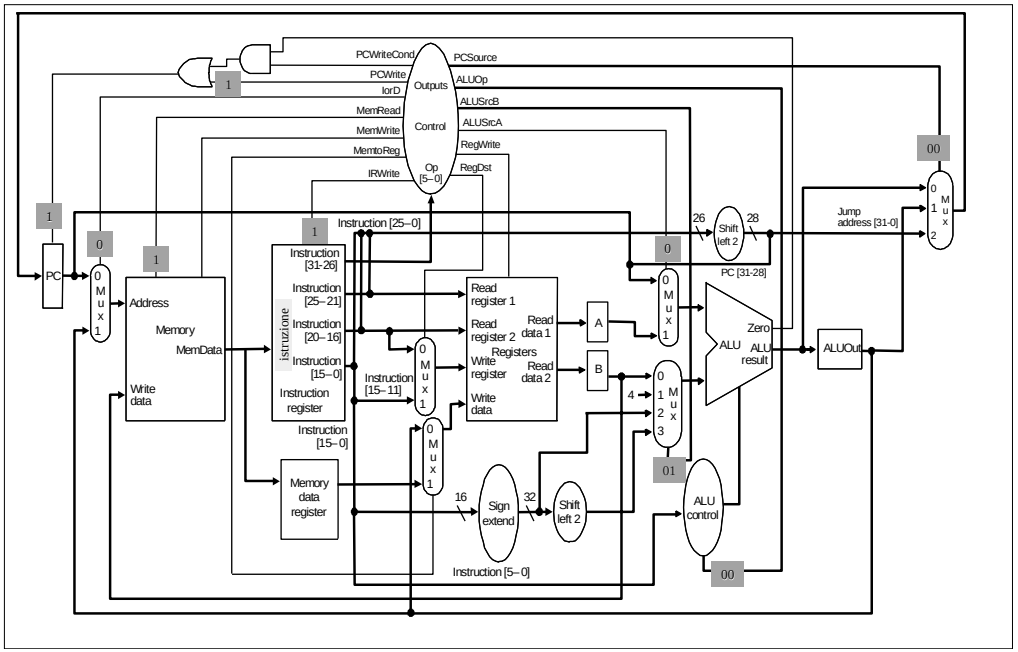
```
IR = Memoria[PC]
PC = PC + 4
```


1 - Passo di prelievo della istruzione (b)

La implementazione di questo passo richiede:

- Per la lettura della istruzione:
  - MemRead = 1
    - lettura della memoria
  - IRWrite = 1
    - scrittura del registro di istruzione (IR)
  - IorD = 0
    - per selezionare il PC come sorgente dell'indirizzo
- Per l'incremento di PC:
  - ALUSrcA = 0
    - per inviare il PC alla ALU come primo operando
  - ALUSrcB = 01
    - per inviare 4 alla ALU come secondo operando
  - ALUOp = 00
    - per fare effettuare una somma alla ALU
  - PCWrite = 1 e PCSource = 00
    - per memorizzare l'indirizzo calcolato dalla ALU (PC+4) nel PC

Il primo passo



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

2 - Passo di decodifica e caricamento registri (a)

- Obiettivo:
  - leggere i registri
  - calcolare l'indirizzo del salto condizionato
- "*Cosa? Ma non sappiamo ancora che istruzione è!*"
  - si, ma non è dannoso leggere i registri e comunque calcolare il salto
    - il calcolo può servire dopo e se non serve… non si usa
- Passo comune a tutte le istruzioni
- Le operazioni eseguite:

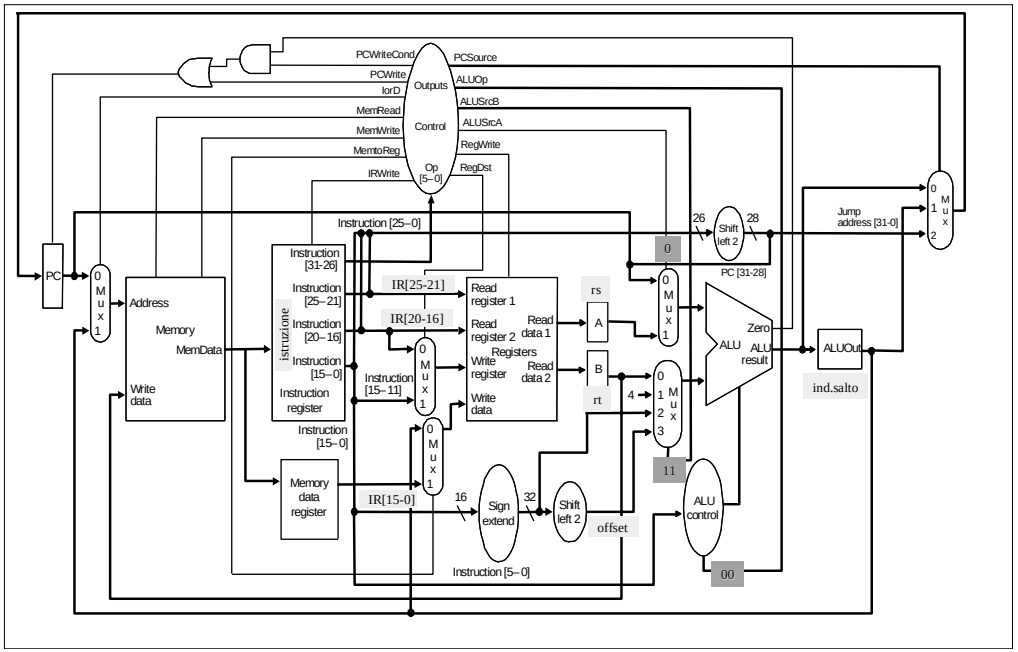
```
A = Reg [IR[25-21]]
B = Reg [IR[20-16]]
ALUOut = PC + (sign-extend(IR[15-0]) << 2)
```


2 - Passo di decodifica e caricamento registri (b)

La implementazione di questo passo richiede:

- Per la lettura dei registri:
  - basta fornire gli indirizzi
  - il banco dei registri deve venire letto ad ogni ciclo e quindi i valori di A e B vengono riscritti ad ogni ciclo di clock
- Per il calcolo del salto condizionato:
  - ALUSrcA = 0
    - per inviare il PC alla ALU come primo operando
  - ALUSrcB = 11
    - per inviare alla ALU come secondo operando il campo offset, esteso e scalato

Il secondo passo



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

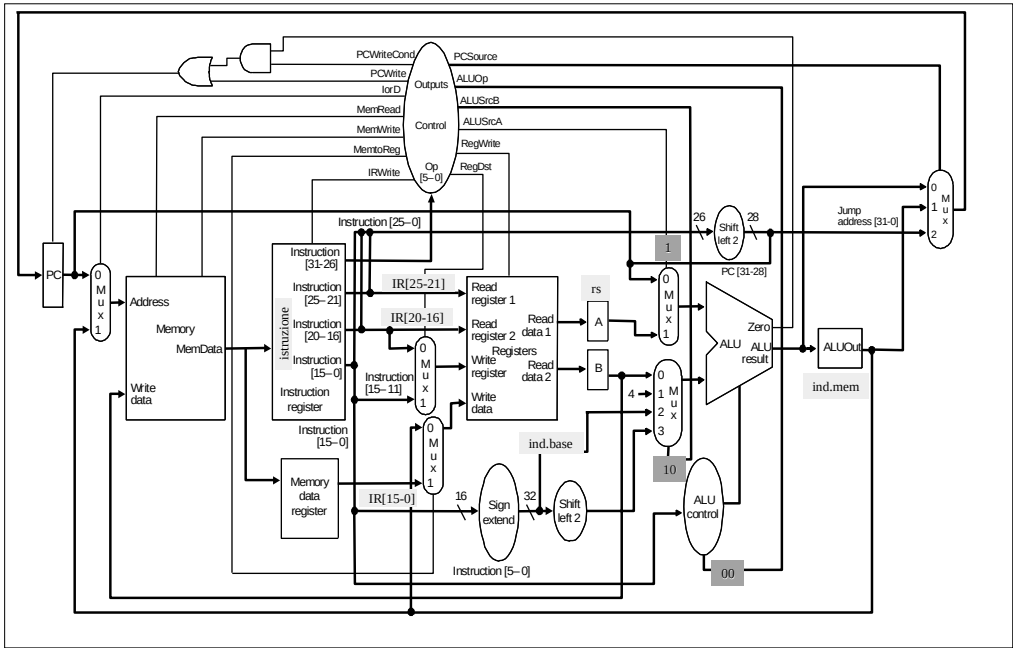
3 - Passo di esecuzione, calcolo indirizzo o salto (a)

- Primo passo in cui le operazioni compiute dipendono dalla istruzione
  - descriviamo le operazioni per ognuna delle istruzioni
- Nel caso di istruzione di accesso alla memoria
  - serve calcolare l'indirizzo della memoria (lw $8, 1000($9))

ALUOut = A + sign-extend(IR[15-0])

- Segnali di controllo
  - ALUSrcA = 1
    - per inviare il registro A alla ALU come primo operando
  - ALUSrcB = 10
    - per inviare il risultato della estens. in segno alla ALU come secondo operando
  - ALUOp = 00
    - per effettuare la somma

Il terzo passo (istruzioni di accesso a memoria)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

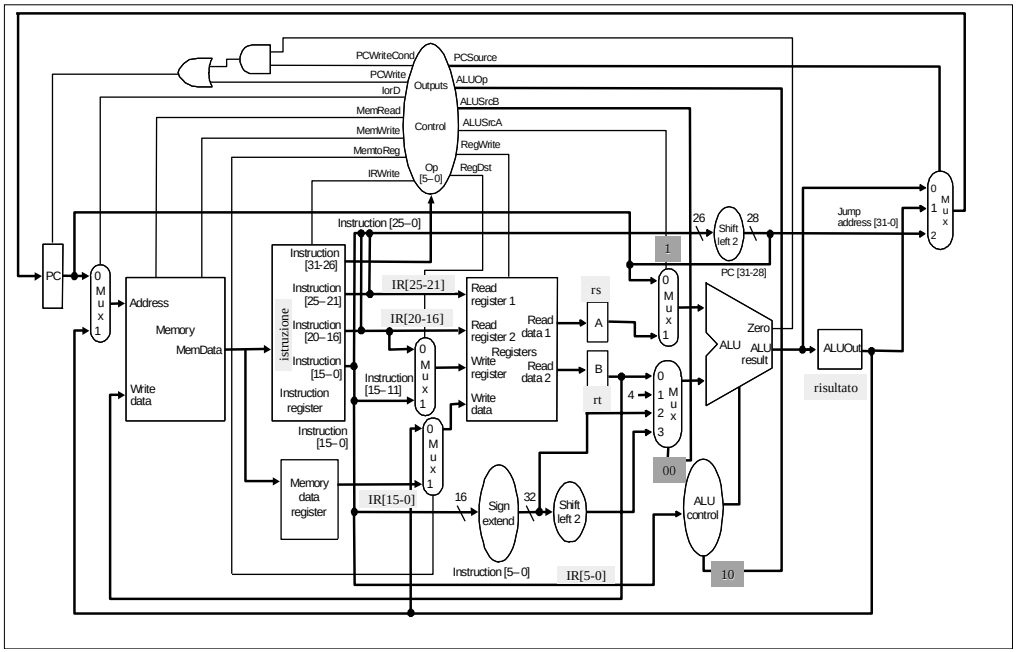
3 - Passo di esecuzione, calcolo indirizzo o salto (b)

- Nel caso di istruzione logico-aritmetica (tipo-R)
  - serve eseguire la operazione richiesta

ALUOut = A op B

- Segnali di controllo
  - ALUSrcA = 1
    - per inviare il registro A alla ALU come primo operando
  - ALUSrcB = 00
    - per inviare il registro B alla ALU come secondo operando
  - ALUOp = 10
    - per effettuare la operazione indicata a partire dai bit nel campo funct della istruzione

Il terzo passo (istruzioni di tipo R)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

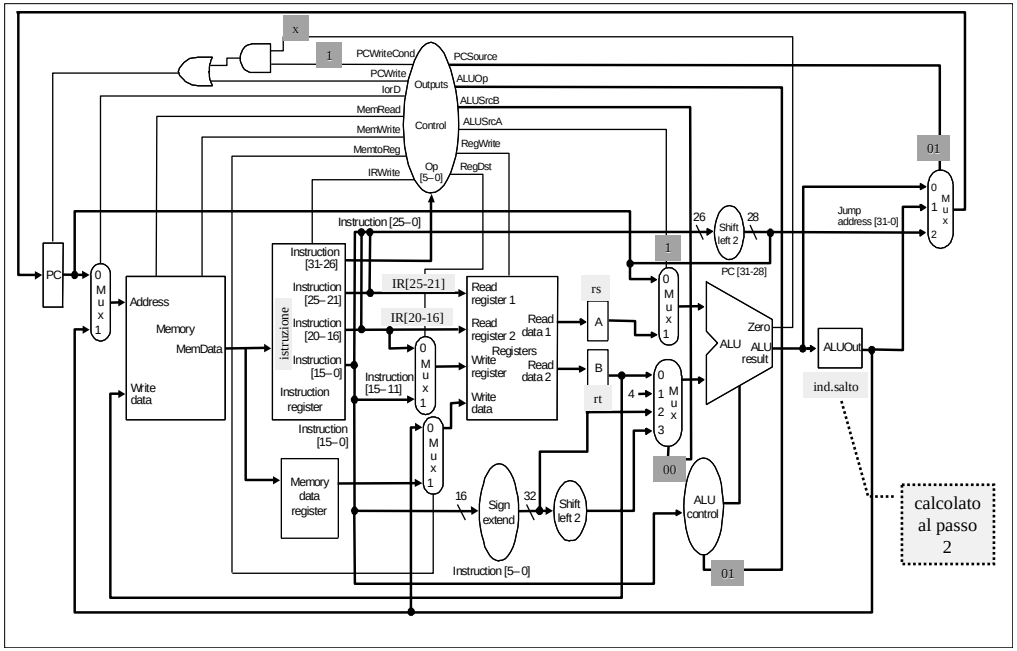
3 - Passo di esecuzione, calcolo indirizzo o salto (c)

- Nel caso di salto condizionato
  - si deve determinare la uguaglianza dei registri

```
if (A == B) PC = ALUOut
```

- Segnali di controllo
  - ALUSrcA = 1
    - per inviare il registro A alla ALU come primo operando
  - ALUSrcB = 00
    - per inviare il registro B alla ALU come secondo operando
  - ALUOp = 01
    - per effettuare la sottrazione per controllare la uguaglianza
  - PCCondWrite = 1
    - per aggiornare il PC se la uscita Zero della ALU è 1
  - PCSource = 01
    - il valore scritto nel PC viene da ALUOut (indirizzo di salto calcolato al passo precedente) (N.B.: il PC viene scritto due volte, prima come PC+4 ed eventualmente con il valore del salto)

Il terzo passo (istruzione di salto condizionato)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

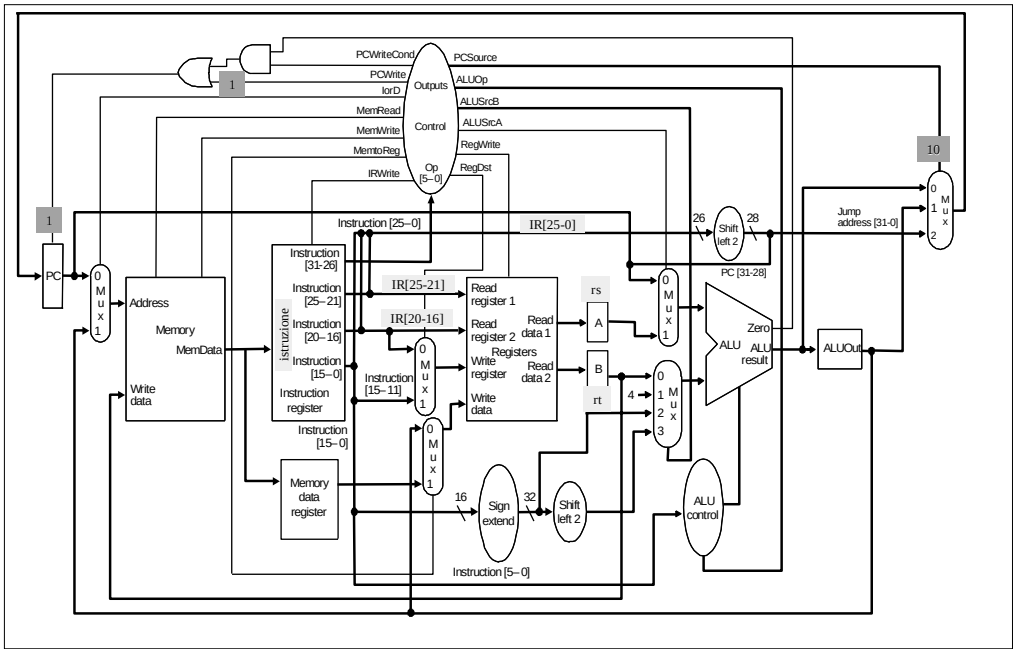
3 - Passo di esecuzione, calcolo indirizzo o salto (d)

- Nel caso di salto incondizionato
  - si deve assegnare il PC con il valore nella istruzione

```
PC = PC[31-28] || (IR[25-0]<<2)
```

- Segnali di controllo
  - PCSource = 10
    - il valore scritto nel PC viene dalla unità di estensione in segno, integrata con i 4 bit più significativi di PC
    - N.B.: anche in questo caso il PC viene scritto due volte, prima come PC+4 e poi con il valore dell'indirizzo del salto incondizionato
  - PCWrite = 1
    - per scrivere il valore nel PC

Il terzo passo (istruzione di salto incondizionato)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

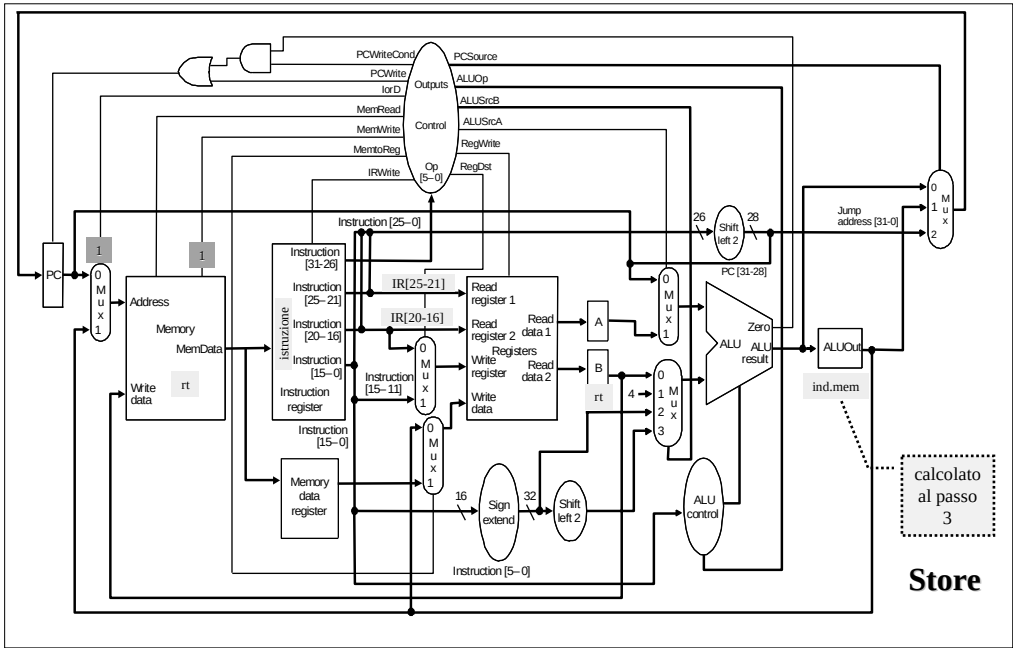
4 - Passo di accesso memoria o completamento R (a)

- Load o store accedono alla memoria (in MDR)
  - in caso di load la scrittura in registro viene al prossimo passo

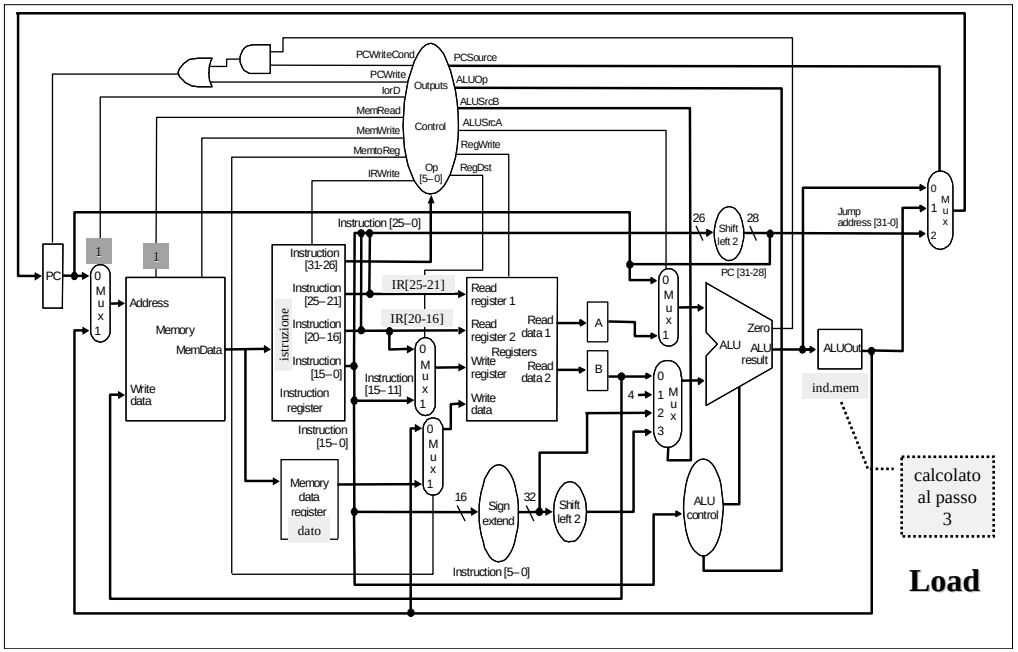
```
Memoria[ALUOut] = B        // per la store
MDR = Memoria (ALUOut) // per la load
```

- Segnali di controllo (indirizzo calcolato al passo precedente)
  - MDR viene comunque scritto ad ogni ciclo
    - nessun segnale di controllo necessario
- Store: (valore da memorizzare in B)
  - MemWrite = 1
  - IorD = 1
    - per usare come indirizzo di memoria quello proveniente dalla ALU
- Load:
  - MemRead = 1
  - IorD = 1
    - per usare come indirizzo di memoria quello proveniente dalla ALU

Il quarto passo (istruzioni di accesso a memoria)



Il quarto passo (istruzioni di accesso a memoria)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

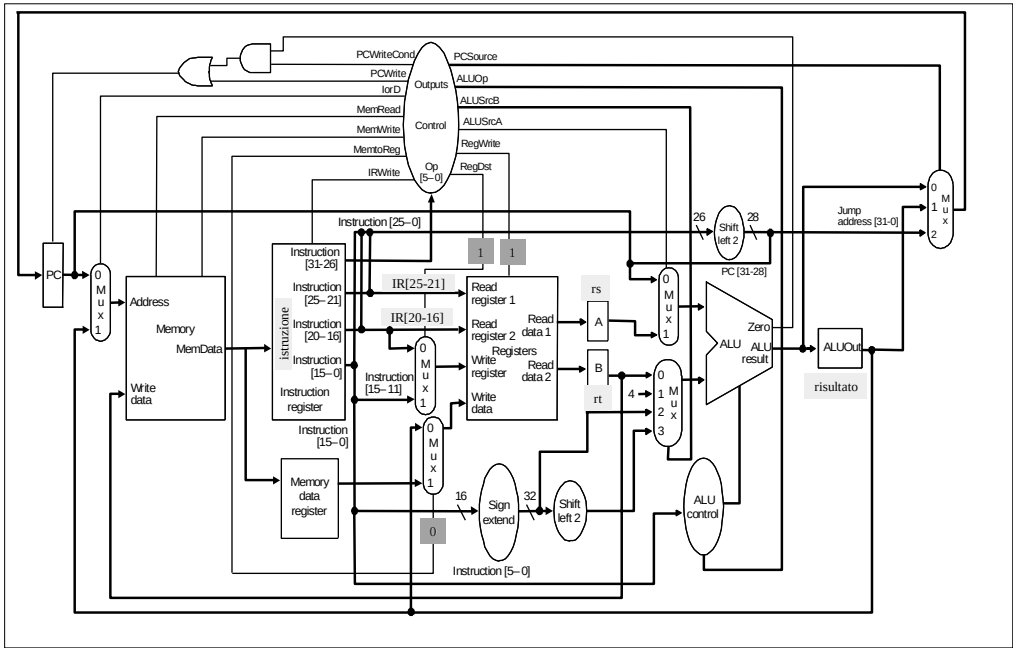
4 - Passo di accesso memoria o completamento R (a)

- Si deve scrivere il valore calcolato nel registro dest.

```
Reg[IR[15-11]] = ALUOut
```

- Il contenuto di ALUOut (calcolato nel passo precedente)
  - va scritto nel registro destinazione (rd)
- I segnali di controllo
  - RegDst = 1
    - per usare i bit 15-11 come registro destinazione
  - RegWrite = 1
    - per scrivere nel registro
  - MemtoReg = 0
    - per scrivere il dato proveniente dalla ALU anziché i dati dalla memoria

Il quarto passo (istruzioni di tipo R)



Organizzazione della lezione

- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

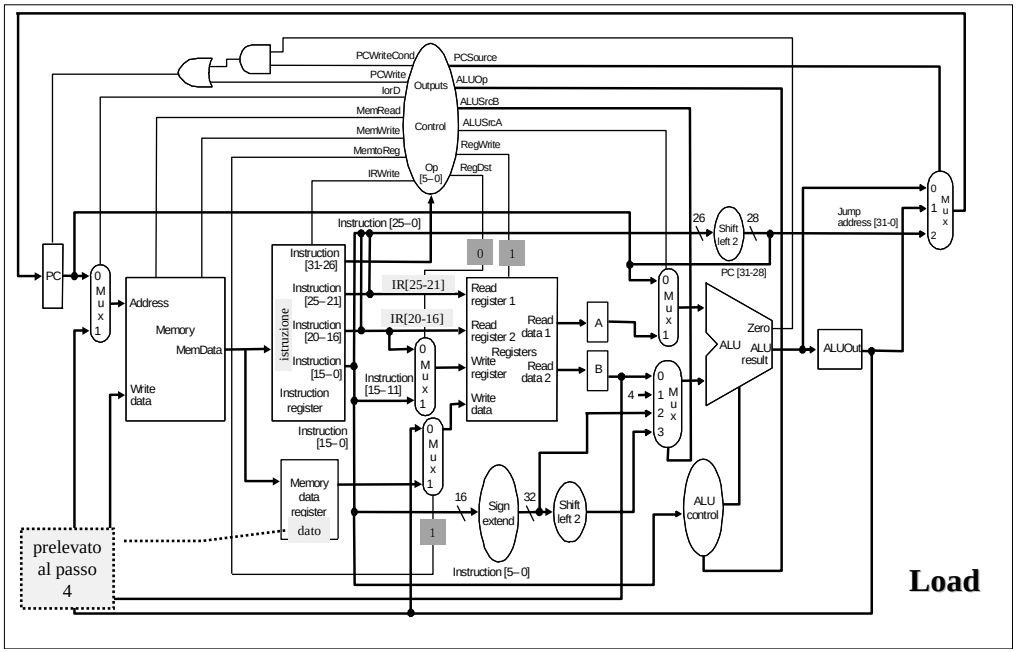
5 - Passo di completamento lettura da memoria (a)

- Si completa la lettura dalla memoria (lw)
  - il registro MDR conteneva il valore da caricare nel registro

```
Reg[IR[20-16]] = MDR
```

- Segnali di controllo
  - RegDst = 0
    - per usare i bit 20-16 come registro destinazione
  - RegWrite = 1
    - per scrivere nel registro
  - MemtoReg = 1
    - per scrivere il dato proveniente dalla memoria anziché il dato dalla ALU

Il quinto passo (istruzioni di accesso a memoria)

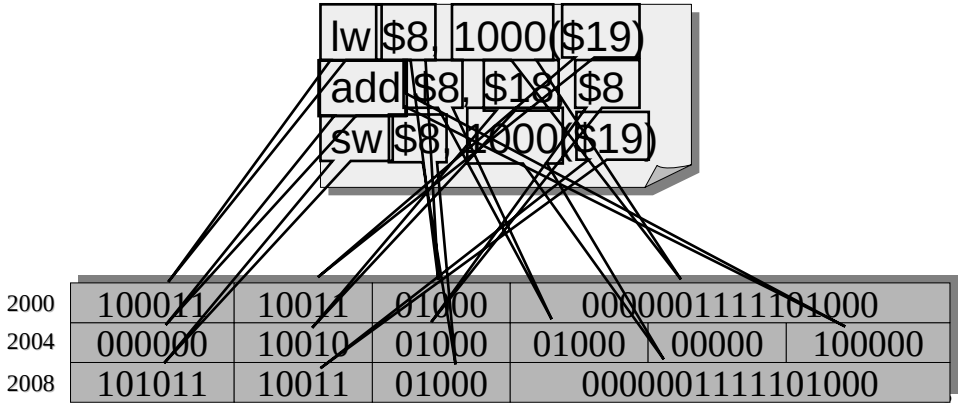


Organizzazione della lezione

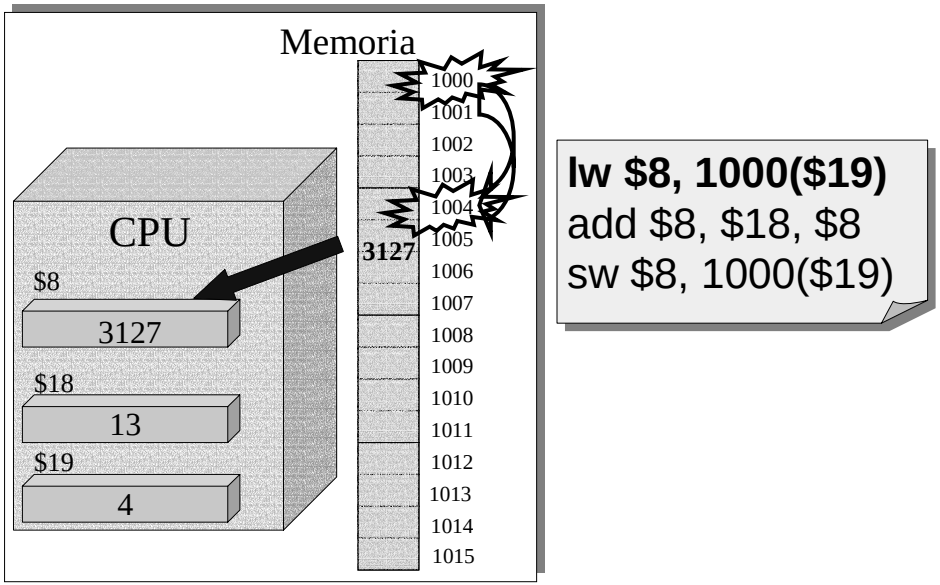
- La progettazione di una unità di elaborazione a ciclo multiplo
- La esecuzione di una istruzione in 5 passi
  1. prelievo della istruzione
  2. decodifica istruzione e caricamento i registri
  3. esecuzione, calcolo indirizzo o salto
  4. accesso alla memoria o completamento istr. tipo R
  5. completamento lettura da memoria
- Una istruzione di esempio

Un programma di esempio

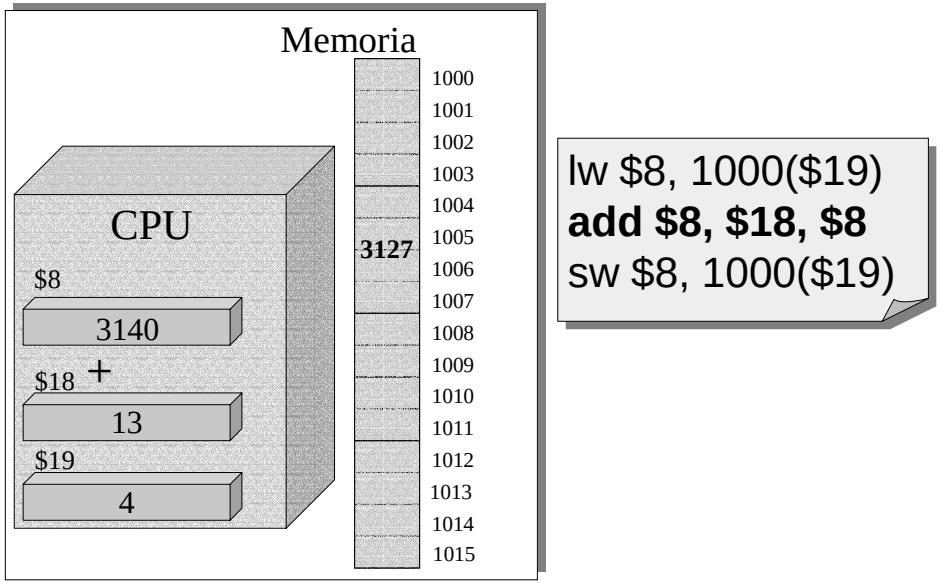
- Programma che carica un dato dalla memoria, lo incrementa e lo rimemorizza
- Supponiamo che sia memorizzato nelle locazioni di memoria a partire dalla 2000



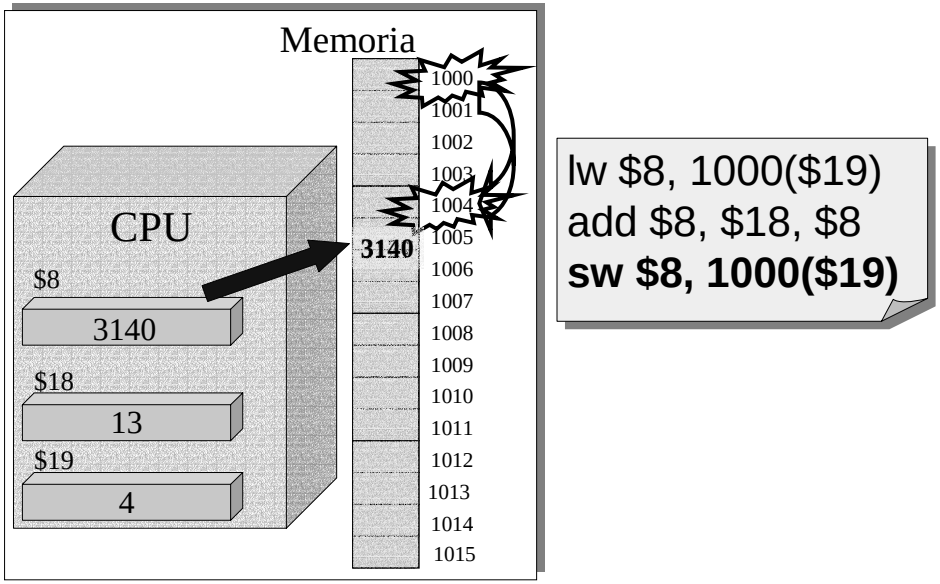
Un programma di esempio (1)



Un programma di esempio (2)



Un programma di esempio (3)



lw $8, 1000($19)

- La rappresentazione di questa istruzione è:

2000	100011	10011	01000	0000001111101000

- Analizziamo la esecuzione di questa istruzione:
  - 5 passi di esecuzione
    - passo di prelievo della istruzione
    - passo di decodifica e caricamento registri
    - passo di esecuzione, calcolo ind.memoria e salto
    - passo di accesso memoria o completamento R
    - passo di completamento lettura memoria

1 - Passo di prelievo della istruzione (a)

- Obiettivo:
  - caricare la istruzione ed incrementare il program counter
- Passo comune a tutte le istruzioni
  - non facciamo alcuna differenza
    - non potremmo, non avendo ancora letto la istruzione!
- Le operazioni eseguite:

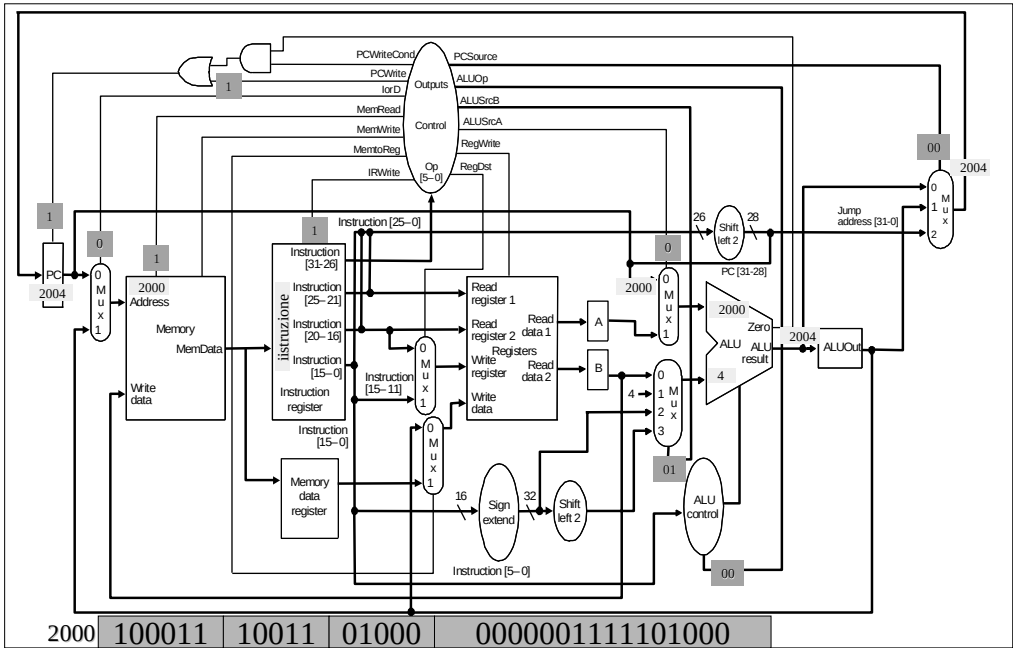
```
IR = Memoria[PC]
PC = PC + 4
```


1 - Passo di prelievo della istruzione (b)

La implementazione di questo passo richiede:

- Per la lettura della istruzione:
  - MemRead = 1
    - lettura della memoria
  - IRWrite = 1
    - scrittura del registro di istruzione (IR)
  - IorD = 0
    - per selezionare il PC come sorgente dell'indirizzo
- Per l'incremento di PC:
  - ALUSrcA = 0
    - per inviare il PC alla ALU come primo operando
  - ALUSrcB = 01
    - per inviare 4 alla ALU come secondo operando
  - ALUOp = 00
    - per fare effettuare una somma alla ALU
  - PCWrite = 1 e PCSource = 00
    - per memorizzare l'indirizzo calcolato dalla ALU (PC+4) nel PC

Il primo passo di lw $8, 1000($19)



2 - Passo di decodifica e caricamento registri (a)

- Obiettivo:
  - leggere i registri
  - calcolare l'indirizzo del salto condizionato
- "*Cosa? Ma non sappiamo ancora che istruzione è!*"
  - si, ma non è dannoso leggere i registri e comunque calcolare il salto
    - il calcolo può servire dopo e se non serve… non si usa
- Passo comune a tutte le istruzioni
- Le operazioni eseguite:

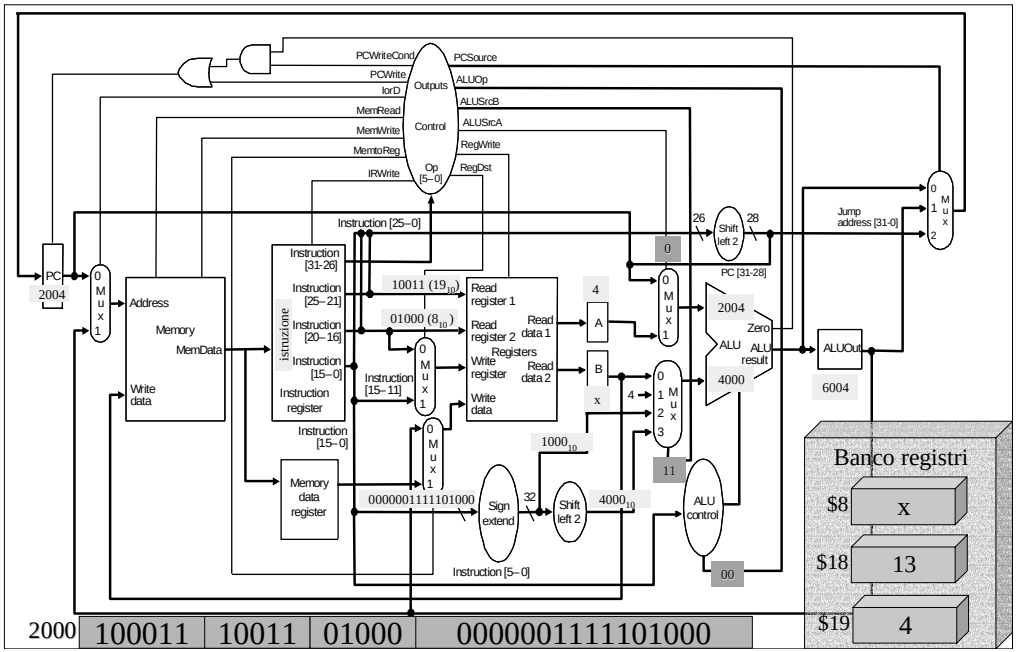
```
A = Reg [IR[25-21]]
B = Reg [IR[20-16]]
ALUOut = PC + (sign-extend(IR[15-0]) << 2)
```


2 - Passo di decodifica e caricamento registri (b)

La implementazione di questo passo richiede:

- Per la lettura dei registri:
  - basta fornire gli indirizzi
  - il banco dei registri deve venire letto ad ogni ciclo e quindi i valori di A e B vengono riscritti ad ogni ciclo di clock
- Per il calcolo del salto condizionato:
  - ALUSrcA = 0
    - per inviare il PC alla ALU come primo operando
  - ALUSrcB = 11
    - per inviare alla ALU come secondo operando il campo offset, esteso e scalato

Il secondo passo di lw $8, 1000($19)



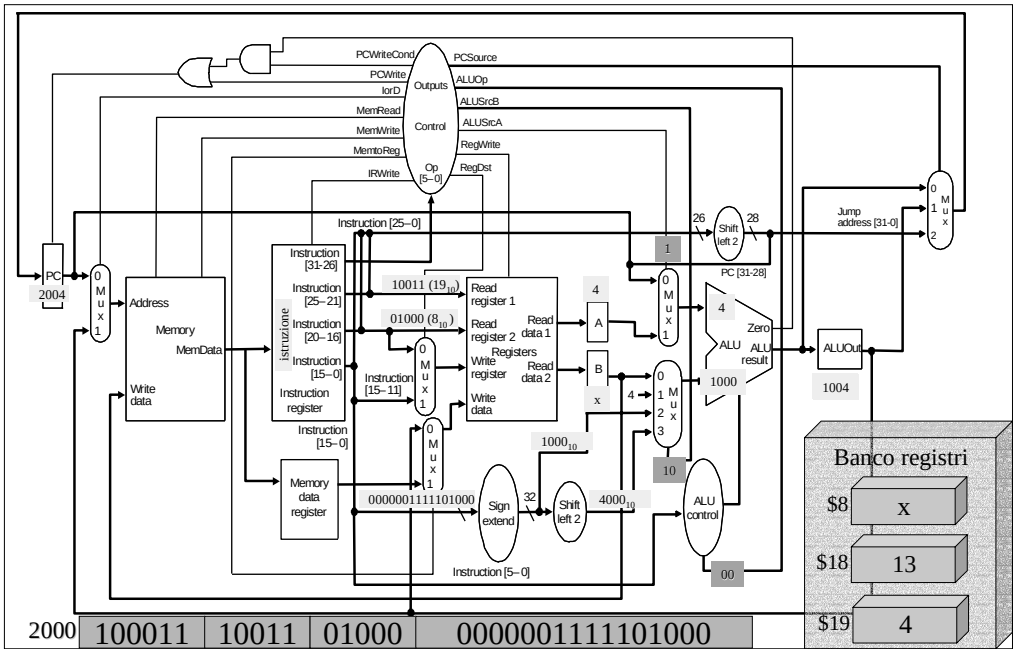
3 - Passo di esecuzione, calcolo indirizzo o salto (a)

- Primo passo in cui le operazioni compiute dipendono dalla istruzione
  - descriviamo le operazioni per ognuna delle istruzioni
- Nel caso di istruzione di accesso alla memoria
  - serve calcolare l'indirizzo della memoria (lw $8, 1000($9))

```
ALUOut = A + sign-extend(IR[15-0])
```

- Segnali di controllo
  - ALUSrcA = 1
    - per inviare il registro A alla ALU come primo operando
  - ALUSrcB = 10
    - per inviare il risultato della estens. in segno alla ALU come secondo operando
  - ALUOp = 00
    - per effettuare la somma

Il terzo passo di lw $8, 1000($19)



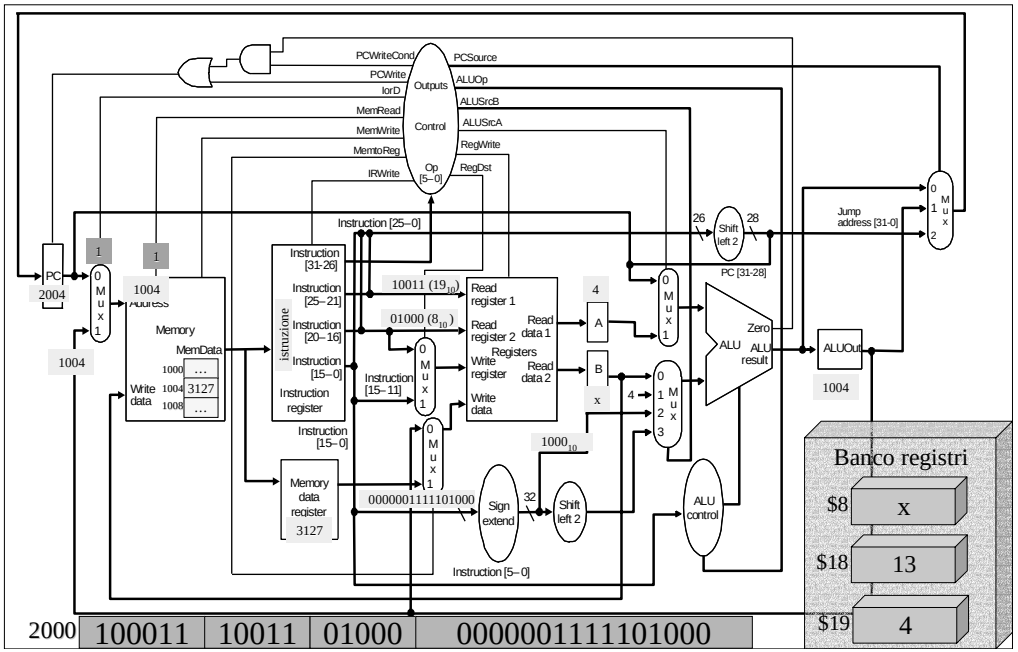
4 - Passo di accesso memoria o completamento R (a)

- Load o store accedono alla memoria (in MDR)
  - in caso di load la scrittura in registro viene al prossimo passo

```
Memoria[ALUOut] = B       // per la store
MDR = Memoria (ALUOut) // per la load
```

- Segnali di controllo (indirizzo calcolato al passo precedente)
  - MDR viene comunque scritto ad ogni ciclo
    - nessun segnale di controllo necessario
- Store: (valore da memorizzare in B)
  - MemWrite = 1
  - IorD = 1
    - per usare come indirizzo di memoria quello proveniente dalla ALU
- Load:
  - MemRead = 1
  - IorD = 1
    - per usare come indirizzo di memoria quello proveniente dalla ALU

Il quarto passo di lw $8, 1000($19)



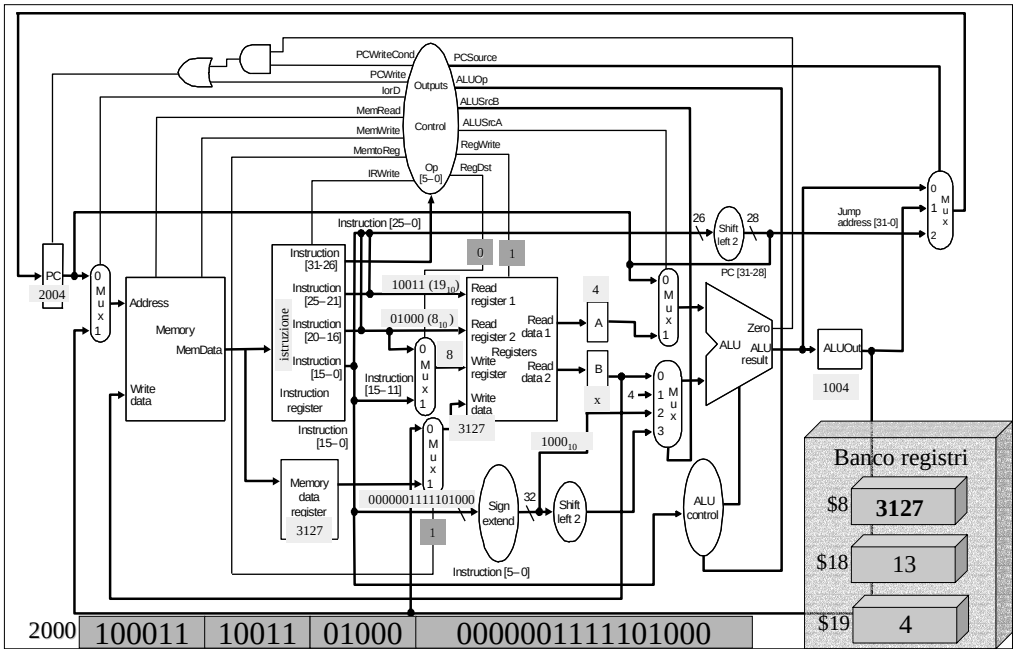
5 - Passo di completamento lettura da memoria (a)

- Si completa la lettura dalla memoria (lw)
  - il registro MDR conteneva il valore da caricare nel registro

```
Reg[IR[20-16]] = MDR
```

- Segnali di controllo
  - RegDst = 0
    - per usare i bit 20-16 come registro destinazione
  - RegWrite = 1
    - per scrivere nel registro
  - MemtoReg = 1
    - per scrivere il dato proveniente dalla memoria anziché il dato dalla ALU

Il quinto passo di lw $8, 1000($19)



Schema riassuntivo dei passi

	Azione intrapresa			
Nome del passo	Istruzioni di tipo R	Accesso Memoria	Salti condizionati	Salti incondizionati
Prelievo	IR= Memoria[PC] PC=PC+4			
Decodifica	A = Reg[IR[25-21]] B=Reg[IR{20-16]] ALUOut = PC+(sign-extend(IR[15-0])<<2			
Esecuzione, calcolo ind. , salti	ALUOut=A op B	ALUOut = A + Sign-extend(IR[15-0])	if (A==B) then PC=ALUOut	PC=PC[31-28] || (IR[25-0]<<2)
Accesso Mem compl. R	Reg[IR[15-11]]=ALUOut	Load: MDR=Memoria[ALUOut] Store: Memoria[ALUOut]=B		
Compl. lettura da memoria		Load: Reg[IR[20-16]]=MDR		

Architettura (2003-2004). Vittorio Scarano