Datapath

Esercitazione 7

Architettura degli elaboratori

1

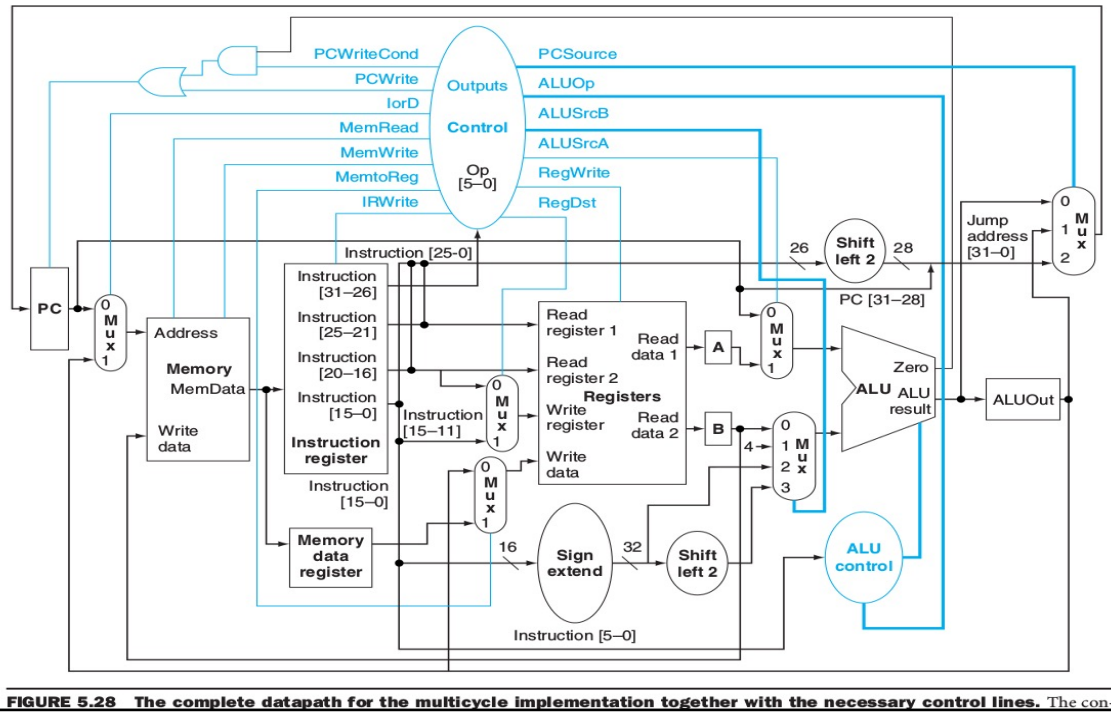


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

2

Esercizio 1

In riferimento allo schema completo del datapath multiciclo (fig. 5.28):

- Quali sono e quale funzione svolge ciascuna delle unità funzionali?
- Quali sono e quale funzione svolgono ciascuno dei registri?
- Qual è il ruolo di ciascuno dei multiplexer?

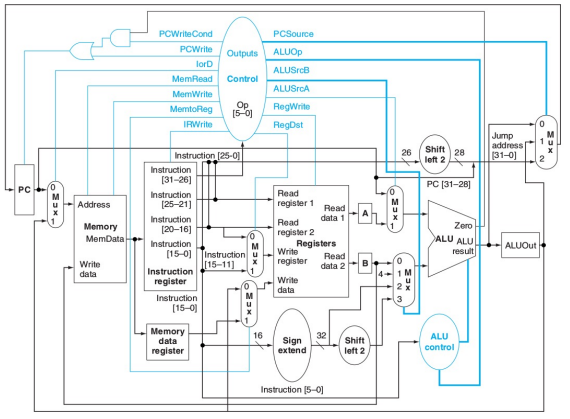


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

3

Memoria: contiene istruzioni e dati; sulla base dell'indirizzo in input, restituisce in output l'istruzione o il dato letto

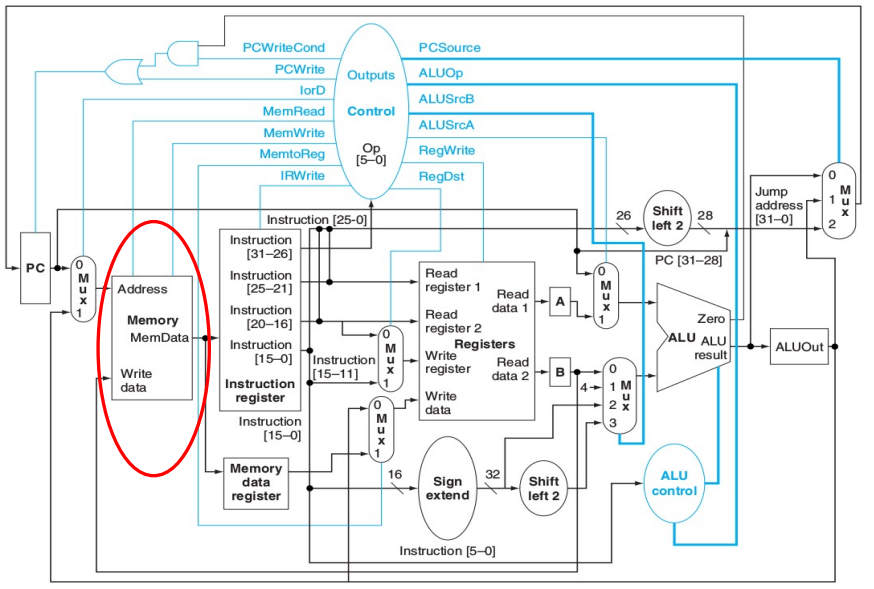


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

4

Register file: 32 registri che contengono i dati utilizzati nel corso dell'esecuzione delle istruzioni; restituisce in output il contenuto dei due registri letti (indicati dai due input – ReadRegister1 e ReadRegister2) e scrive, se il segnale di scrittura è attivo, il dato in input WriteData nel registro indicato dall'indirizzo in input (WriteRegister)

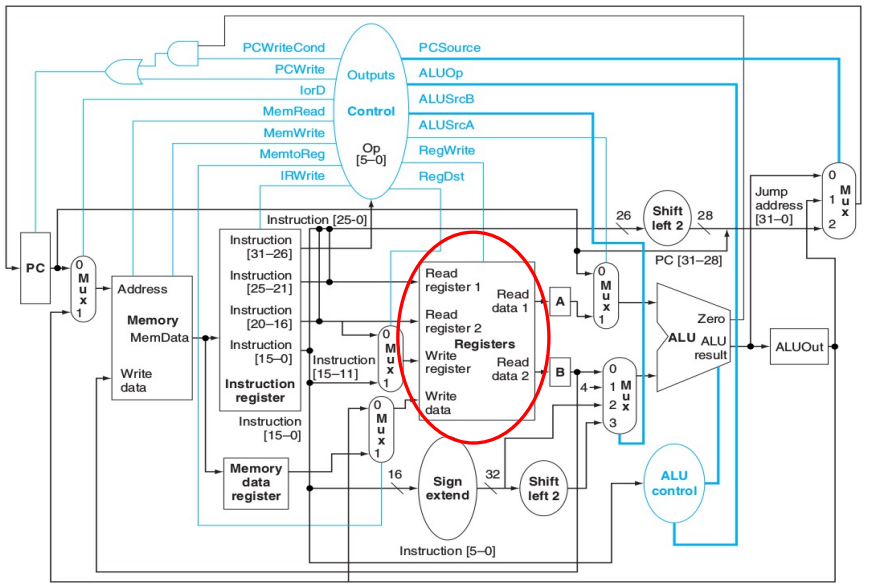


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

5

ALU: per tutte le istruzioni: incrementa il PC (PC+4) e calcola il valore di branch (che sarà usato solo nel caso di istruzioni di branch) Inoltre:

- Istruz. R-type: esegue operazioni aritmetico-logiche su due operandi in funzione del function_code indicato dai 6 bit meno significativi dell'istruzione
- Istr. accesso a memoria: calcola l'indirizzo di memoria cui accedere
- Istr. branch: confronta i due registri in input

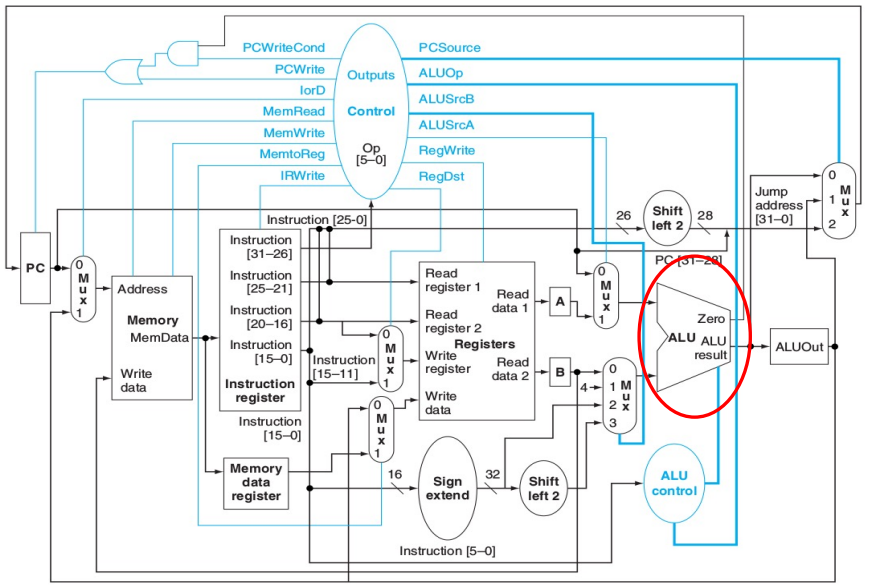


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

6

Sign Extended: opera l'estensione di segno dai 16 bit in input ai 32 bit in output

Shift Left 2: opera uno shift a sinistra dei bit in input (con l'effetto di moltiplicare per 4 l'input)

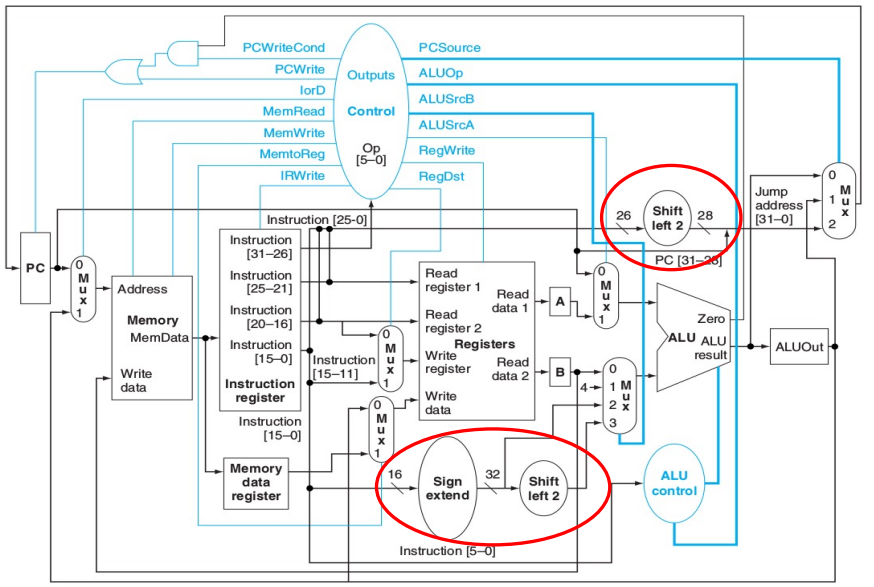


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Esercizio 1 – soluzione (Unità funzionali)

- **Memoria**: contiene istruzioni e dati; sulla base dell'indirizzo in input, restituisce in output l'istruzione o il dato letto
- **Register file**: 32 registri che contengono i dati utilizzati nel corso dell'esecuzione delle istruzioni; restituisce in output il contenuto dei due registri letti (indicati dai due input – eadRegister1 e ReadRegister2) e scrive, se il segnale di scrittura è attivo, il dato in input WriteData nel registro indicato dall'indirizzo in input (WriteRegister)
- **ALU**:
  - per istruzioni R-type: esegue operazioni aritmetico-logiche su due operandi in funzione del function_code indicato dai 6 bit meno significativi dell'istruzione
  - per istruzioni di accesso a memoria: calcola l'indirizzo di memoria cui accedere
  - per istruzioni branch: confronta i due registri in input
  - per tutte le istruzioni: incrementa il PC (PC+4) e calcola il valore di branch (che sarà usato solo nel caso di istruzioni di branch)
- **Sign Extended**: opera l'estensione di segno dai 16 bit in input ai 32 bit in output
- **Shift Left 2**: opera uno shift a sinistra dei bit in input (con l'effetto di moltiplicare per 4 l'input)

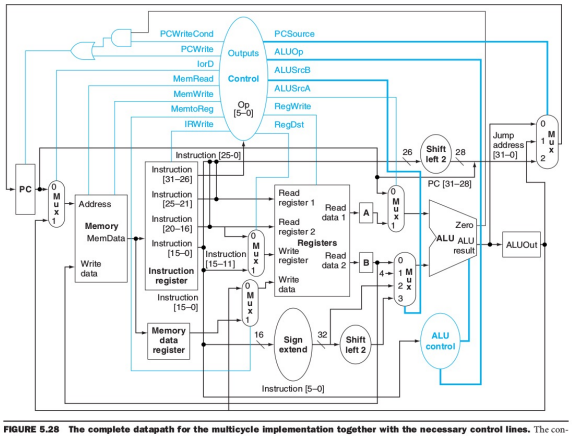


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Esercizio 1 – soluzione (registri)

- **PC**: contiene i 32 bit che indicano l'indirizzo dell'istruzione da eseguire
- **Instruction register**: contiene i 32 bit che corrispondono alla codifica dell'istruzione prelevata da memoria per l'esecuzione
- **Memory Data Register**: contiene il dato letto da memoria prima della sua scrittura nel registro destinazione (e.g. nel caso di esecuzione di istruzione load)
- **A e B**: contengono i valori letti dai registri del RegisterFile
- **ALUOut**: contiene l'output della ALU

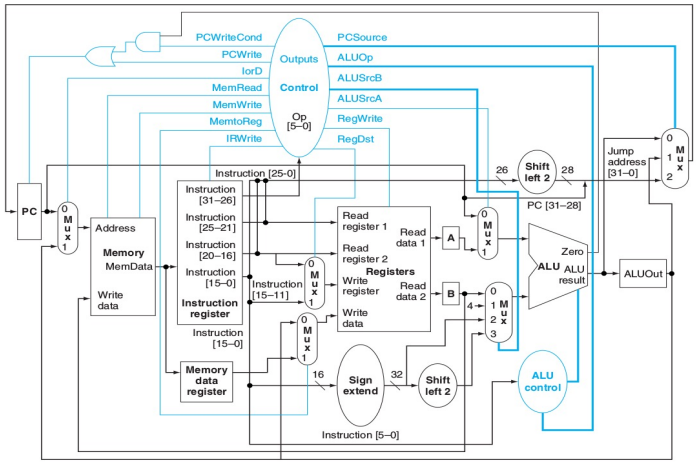


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

9

- **A**: seleziona l'indirizzo di memoria a cui accedere tra:
  - PC (nel caso si voglia leggere un'istruzione)
  - output della ALU (nel caso di esecuzione dell'istruzione load)
- **B**: seleziona il gruppo bit dell'IR che indica il registro in cui scrivere per i due tipi di istruzione I-type (IR[20:16]) e R-type (IR[15:11])
- **C**: seleziona la sorgente per il dato da scrivere in memoria tra
  - ALUOut (per istruzioni R-type)
  - MemoryDataRegister (per istruzioni load)
- **D**: seleziona il primo operando dell'operazione aritmetico-logica eseguita dalla ALU tra PC (per incremento +4) e registro A (per istruzioni R-type)
- **E**: seleziona il secondo operando dell'operazione aritmetico-logica eseguita dalla ALU
  - B (per istruzioni R-type)
  - 4 (per aggiornamento del PC)
  - sign-extended IR[15:0] (per istruzioni di accesso a memoria - lw/sw)
  - sign-extended 2-shifted (j)
- **F**: seleziona il valore per l'aggiornamento del PC
  - output dell'ALU: PC + 4
  - contenuto di ALUOut: 2 shifted 26-bit beq field
  - jump target address (IR[25:0] shifted left 2 bits e concatenato con PC + 4[31:28])

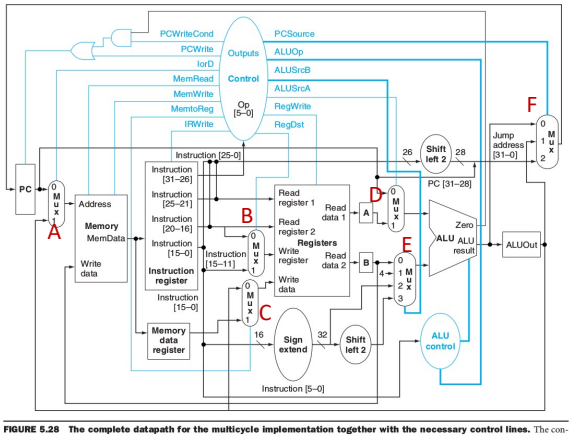


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

10

Esercizio 2

In riferimento allo schema completo del datapath multiciclo (fig. 5.28), in che fase (fetch, decode, execute), e come, sono impostati i segnali di controllo per le diverse classi di istruzioni?

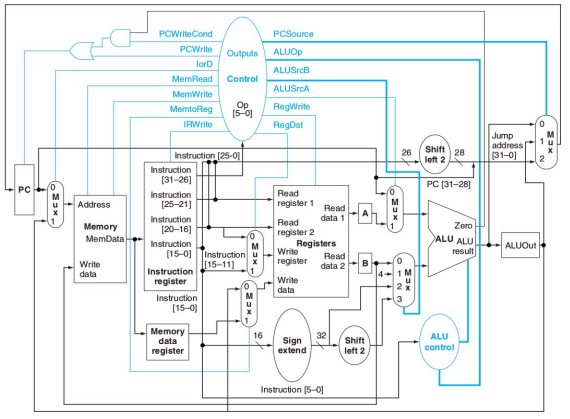


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

11

Esercizio 2 – soluzione

- L'attivazione dei segnali di controllo per un datapath multiciclo può essere descritta da un automa a stati finiti
- I segnali di controllo (next_state per l'automa) sono definiti **in funzione dello stato corrente e dell'Opcode**
- **Per tutti i tipi di istruzioni**, nella fase di fetch (stato 0) sono impostati i seguenti segnali:
  - IorD=0 per selezionare PC come sorgente per l'indirizzo di memoria
  - MemRead: per leggere un'istruzione da memoria
  - IRWrite: per scrivere l'istruzione letta nell'Instruction register
  - ALUSrcA, ALUSrcB, ALUOp, PCWrite, e PCSource sono impostati per calcolare PC + 4 e memorizzare il nuovo valore nel PC
- **Per tutti i tipi di istruzioni**, nella fase di decode (stato 1) sono impostati i segnali ALUSrcA, ALUSrcB e ALUOp per calcolare il valore di branch
- Nella fase di execute sono impostati i segnali di controllo, **in funzione del tipo di istruzione in esecuzione** (v. slide successiva)

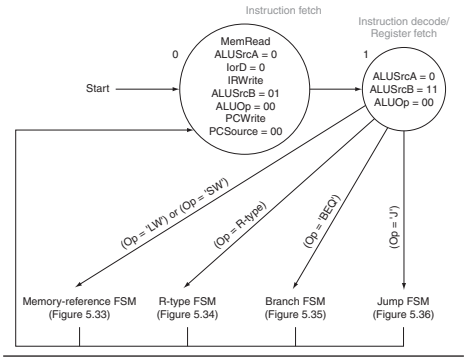
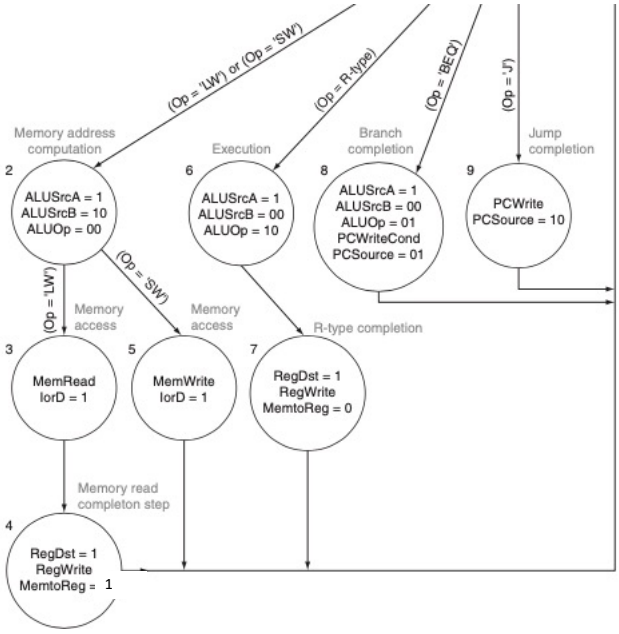


FIGURE 5.32 The instruction fetch and decode portion of every instruction is identical. These states correspond to the top box in the abstract finite state machine in Figure 5.31. In the first

12

- le istruzioni di jump o branch, dopo lo stato 1, richiedono un ciclo di clock per il loro completamento
- le istruzioni di tipo R-type e le istruzioni di memorizzazione (store), richiedono due cicli di clock per l'esecuzione di una sequenza di due stati d'impostazione dei segnali
- le istruzioni di lettura da memoria richiedono tre cicli di clock per l'esecuzione di una sequenza di tre stati d'impostazione dei segnali



13

Esercizio 3

Per quali classi di istruzioni, e come, è utilizzata la ALU (nella fase di execute)?

14

Esercizio 3 – soluzione

- La ALU (nella fase di execute) è utilizzata per
  - **istruzioni R-type**: per esecuzione dell'operazione indicata tra i due operandi
  - **istuzioni di accesso a memoria (lw/sw)**: per calcolare la somma tra l'offset e il contenuto del registro base
  - **istruzioni di salto condizionato (branch)**: per il confronto dei valori contenuti nei due registri che compaiono nella condizione. L'indirizzo di salto è già in ALUout perché è stato calcolato nella fase di decode
- La ALU non è usata nella fase execute di istruzioni jump

15

Esercizio 4

- Quanti cicli sono necessari per l'esecuzione dell'istruzione

  add $s0, $s1, $s2 ?

16

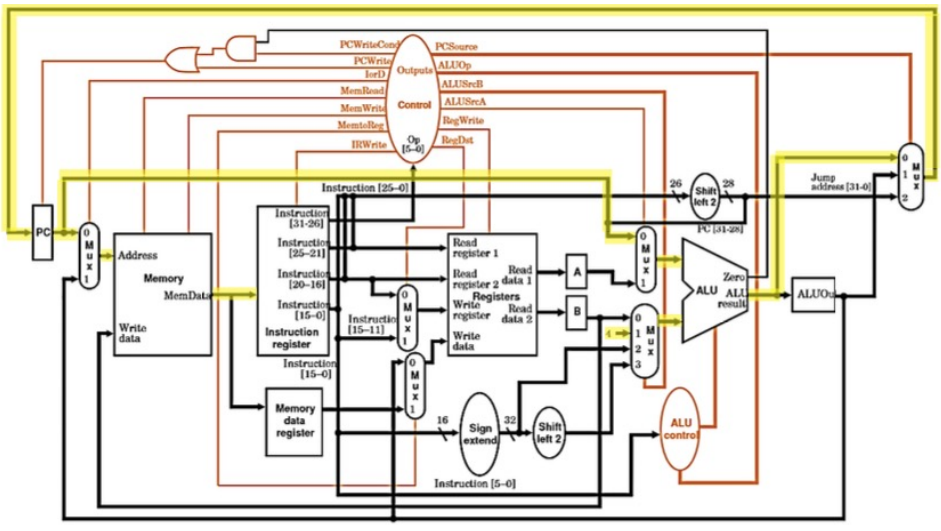
Esercizio 4 – soluzione

- L'istruzione add $s0, $s1, $s2 è di tipo R-type e richiede quindi 4 cicli per essere completata
- Nelle slide successive sono riportati i dettagli di ciascuna fase

17

Esercizio 4 – soluzione (ciclo 1)

Signal	Value
PCWrite	1
IorD	0
MemRead	1
MemWrite	0
IRWrite	1
PCSource	00
ALUOp	00
ALUSrcB	01
ALUSrcA	0
RegWrite	0

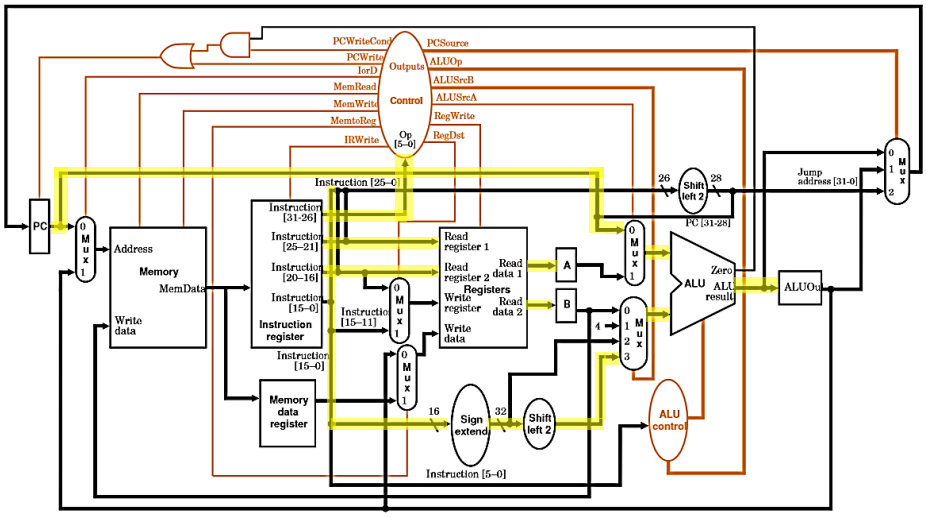


18

Esercizio 4 – soluzione (ciclo 2)

Signal	Value
ALUOp	00
ALUSrcB	11
ALUSrcA	0

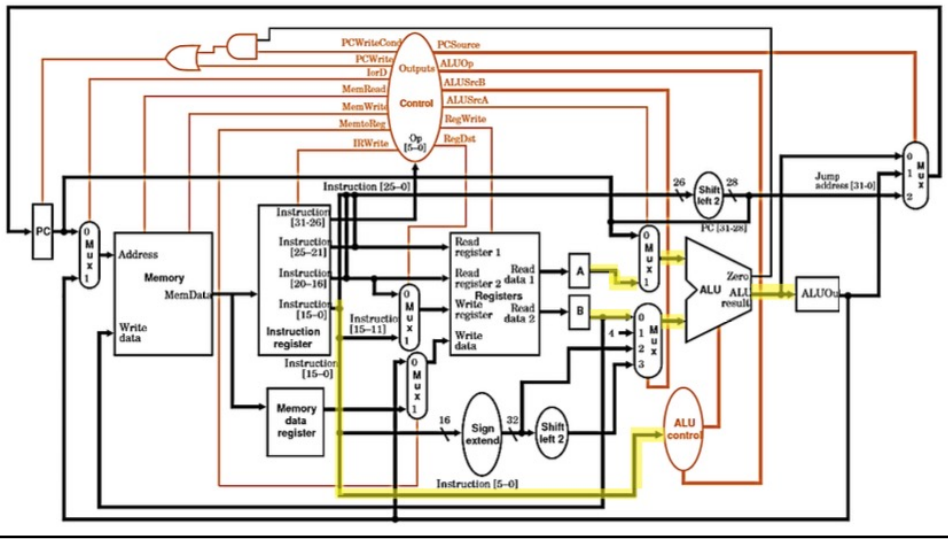
N.B. anche se in questo caso (istruzione R-type) non sarà necessario, è comunque calcolato il valore di branch che viene utilizzato per l'aggiornamento del PC nel caso di istruzioni di salto condizionato



19

Esercizio 4 – soluzione (ciclo 3)

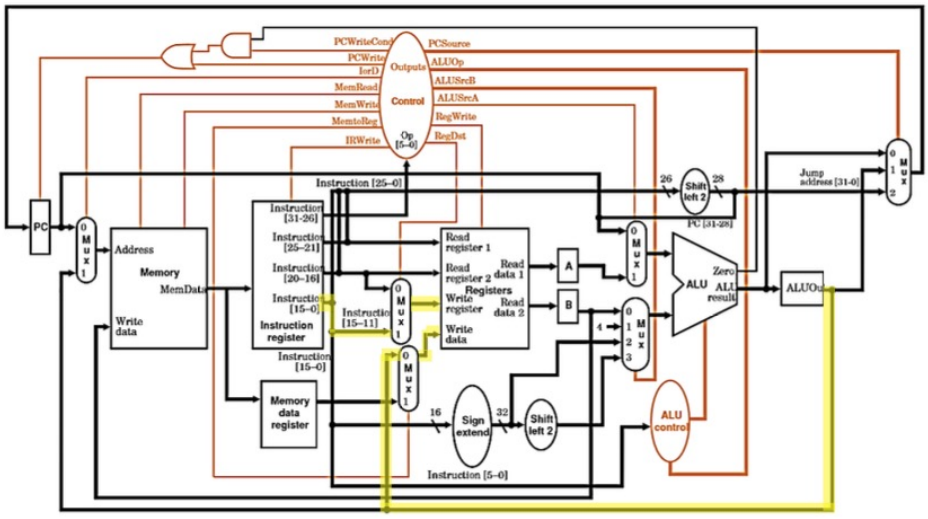
Signal	Value
ALUOp	10
ALUSrcB	00
ALUSrcA	1



20

Esercizio 4 – soluzione (ciclo 4)

Signal	Value
MemtoReg	0
RegWrite	1
RegDst	1



21

Esercizio 5

- Quanti cicli sono necessari per l'esecuzione dell'istruzione beq $s0, $s1, imm ?

22

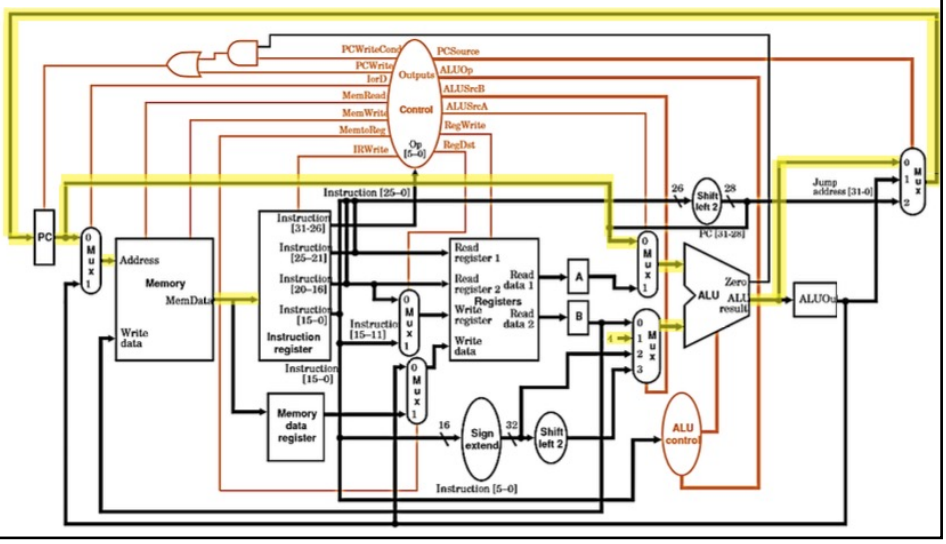
Esercizio 5 – soluzione

- L'istruzione beq $s0, $s1, imm richiede 3 cicli per essere completata
- Nelle slide successive sono riportati i dettagli di ciascuna fase

23

Esercizio 5 – soluzione (ciclo 1)

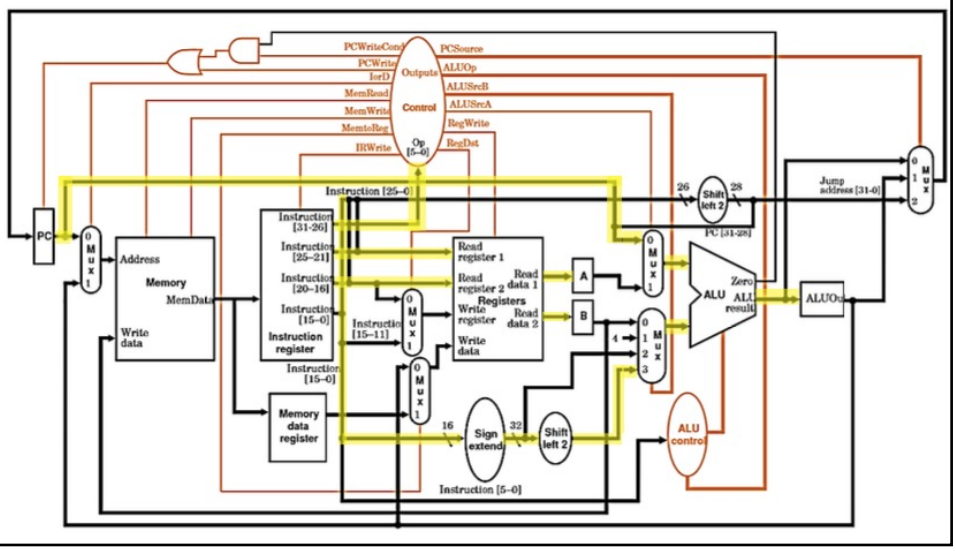
Signal	Value
PCWrite	1
IorD	0
MemRead	1
MemWrite	0
IRWrite	1
PCSource	00
ALUOp	00
ALUSrcB	01
ALUSrcA	0
RegWrite	0



24

Esercizio 5 – soluzione (ciclo 2)

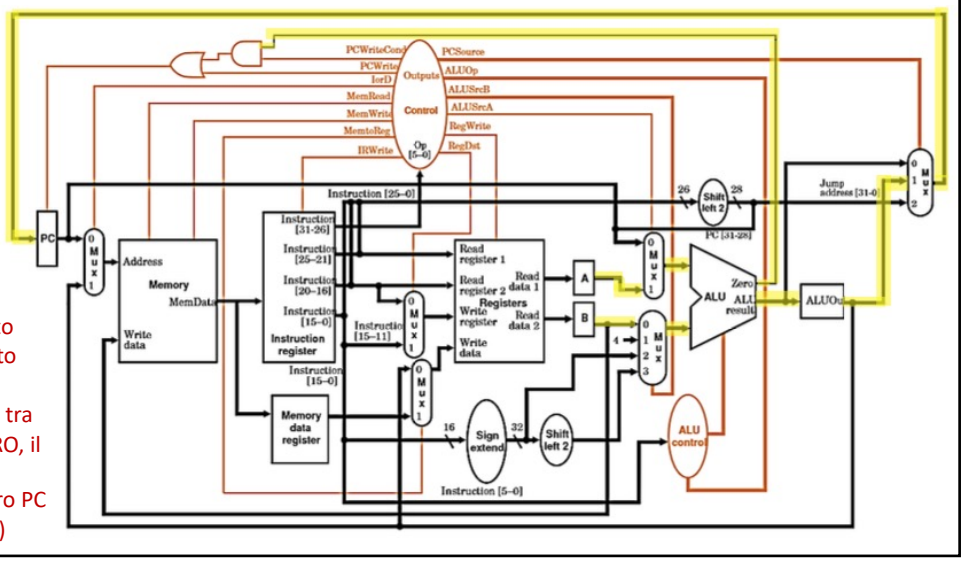
Signal	Value
ALUOp	00
ALUSrcB	11
ALUSrcA	0



25

Esercizio 5 – soluzione (ciclo 3)

Signal	Value
ALUOp	01
ALUSrcB	00
ALUSrcA	1
PCSource	01
PCWriteCond	1



N.B. il valore di branch calcolato nel ciclo precedente viene usato per l'aggiornamento del PC
La ALU è usata per il confronto tra A e B e solo se il risultato è ZERO, il segnale di output della ALU consente la scrittura del registro PC (v. porta AND in alto a sinistra!)

26

Esercizio 6

- Aggiungere l'istruzione jr rs al set delle istruzioni del datapath multiciclo. Specificare le modifiche da effettuare al datapath e le modifiche da effettuare nella FSM relativa

- Strumenti:
  - Appendice A: informazioni sul formato dell'istruzione e sulla semantica dell'istruzione (cioè qual è l'effetto della sua esecuzione)
  - Due tabelle fig 5.29 pag. 324 (capitolo 5 disponibile in area elearning): informazioni per convenzioni impostazioni segnali

N.B. diverse soluzioni possibili!

27

Esercizio 6 – soluzione

- L'istruzione che si vuole implementare (v. pagina A-64 del libro di testo) determina l'aggiornamento del PC con i 32 bit contenuti nel registro il cui indirizzo è indicato dai 5 bit [25:21] dell'istruzione

Jump register

jr rs

da pag. A-64

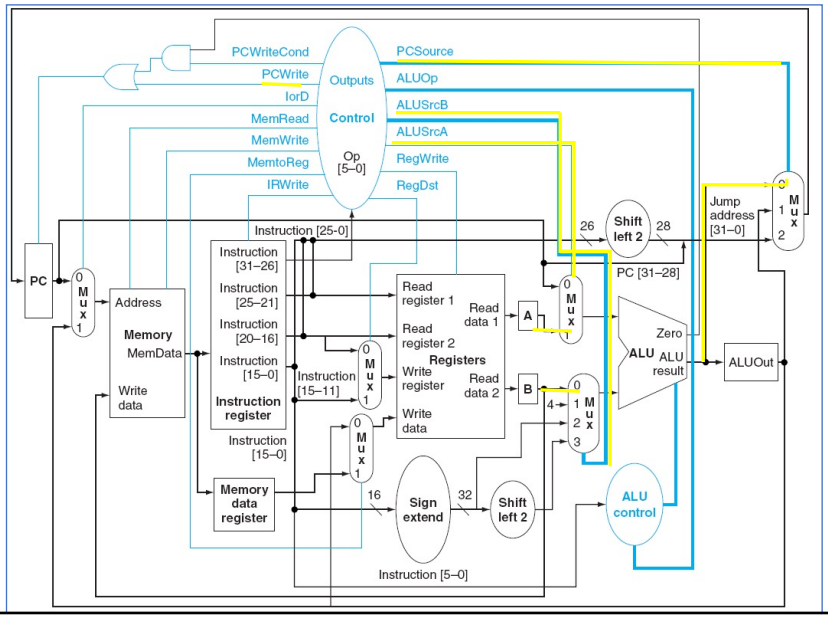
Unconditionally jump to the instruction whose address is in register rs.

- Tenendo conto che per ogni istruzione, nella fase di decode, viene scritto
  - in A il contenuto del registro il cui indirizzo è indicato dai 5 bit [25:21] dell'istruzione
  - in B il contenuto del registro il cui indirizzo è indicato dai 5 bit [20:16] dell'istruzione
- Per realizzare l'effetto desiderato, potremo impostare i segnali di controllo in modo tale da scrivere in PC il risultato della somma tra il contenuto del registro A e il registro B:
  - abilitare la scrittura del PC (PCWrite=1)
  - selezionare l'output della ALU come valore da scrivere nel PC (PCSource=0)
  - selezionare il registro A come primo input per la ALU (ALUScrA=1)
  - selezionare il registro B come secondo input per la ALU (ALUScrB=00)
  - sommare (ALUOp=00)

28

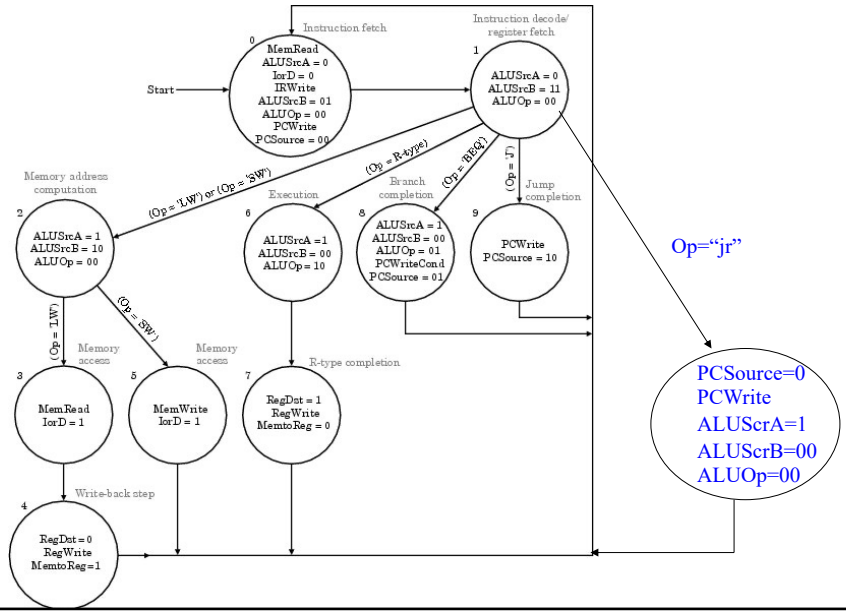
Esercizio 6 – soluzione (jr rs)

PCSource=0
PCWrite
ALUScrA=1
ALUScrB=00
ALUOp=00



29

Esercizio 6 – soluzione (jr rs)

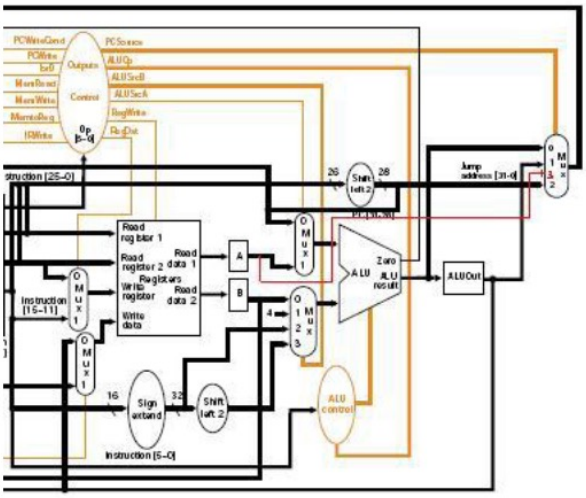


30

Esercizio 6 – soluzione alternativa

Soluzione alternativa può essere aggiungere una connessione diretta tra il registro A e il multiplexer che seleziona l'input per l'aggiornamento del PC (e associare il valore $3_{10}=11_2$ a PCSource, segnale di attivazione del multiplexer per la sua selezione).

Tuttavia, tale soluzione implica una modifica dell'hardware del processore e quindi più onerosa rispetto alla prima soluzione proposta



31