

Datapath multiciclo

Esercitazione - Architettura degli elaboratori

Ob. Saper descrivere lo schema, disegnare l'automa di controllo dei segnali di attivazione.
Conoscere il metodo di lavoro per modificare l'automa di controllo (o lo schema) per aggiungere istruzioni

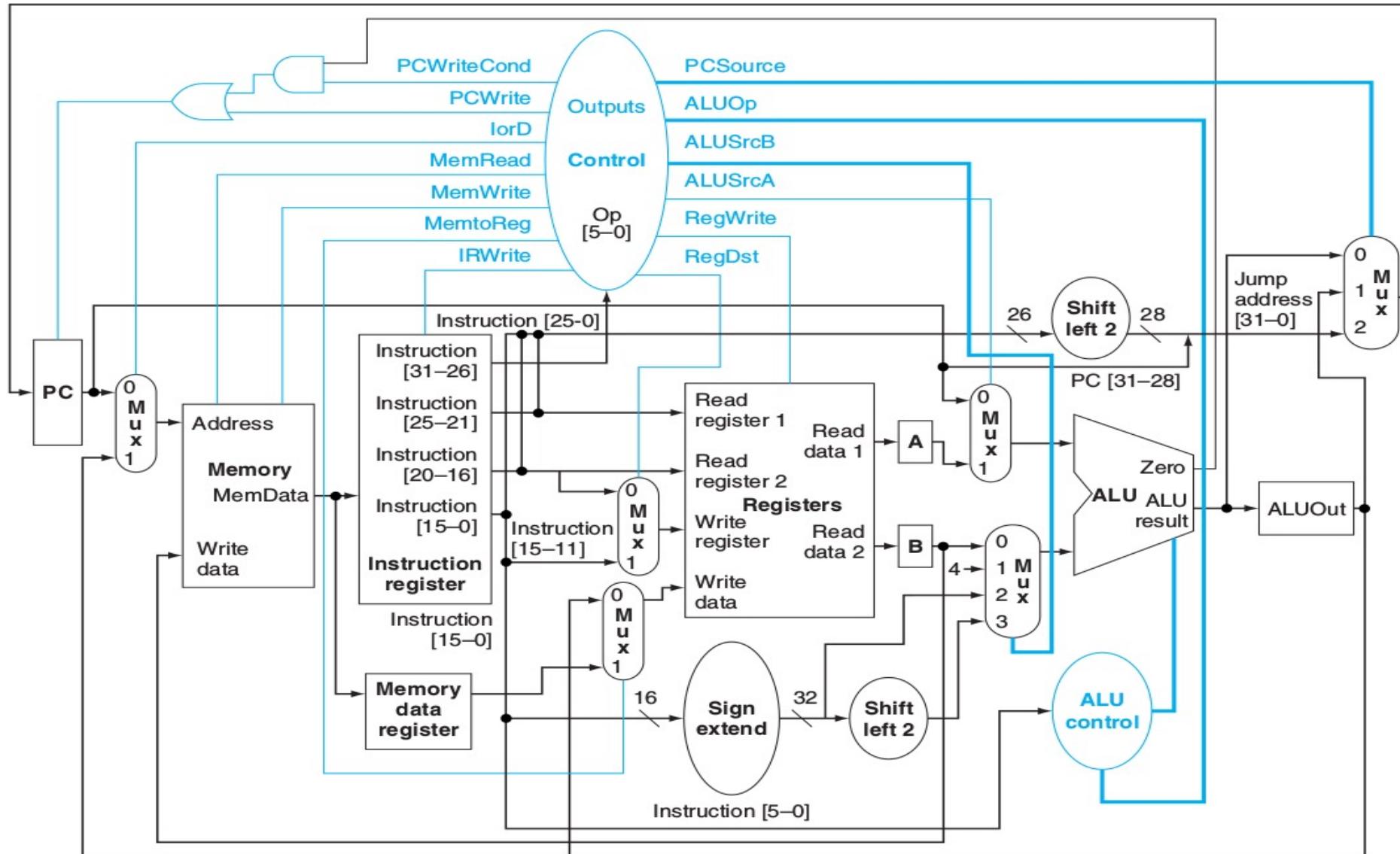


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

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?

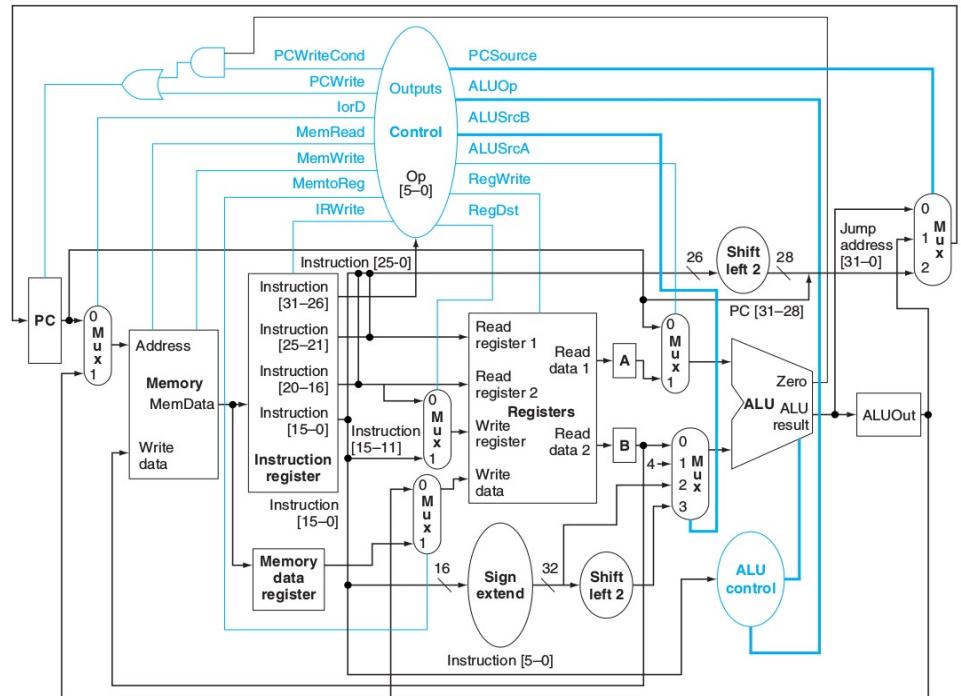


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Memoria: contiene istruzioni e dati; sulla base dell'indirizzo in input, restituisce in output l'istruzione o il dato letto. Nel caso di istruzione store memorizza il dato (WriteData) all'indirizzo in input

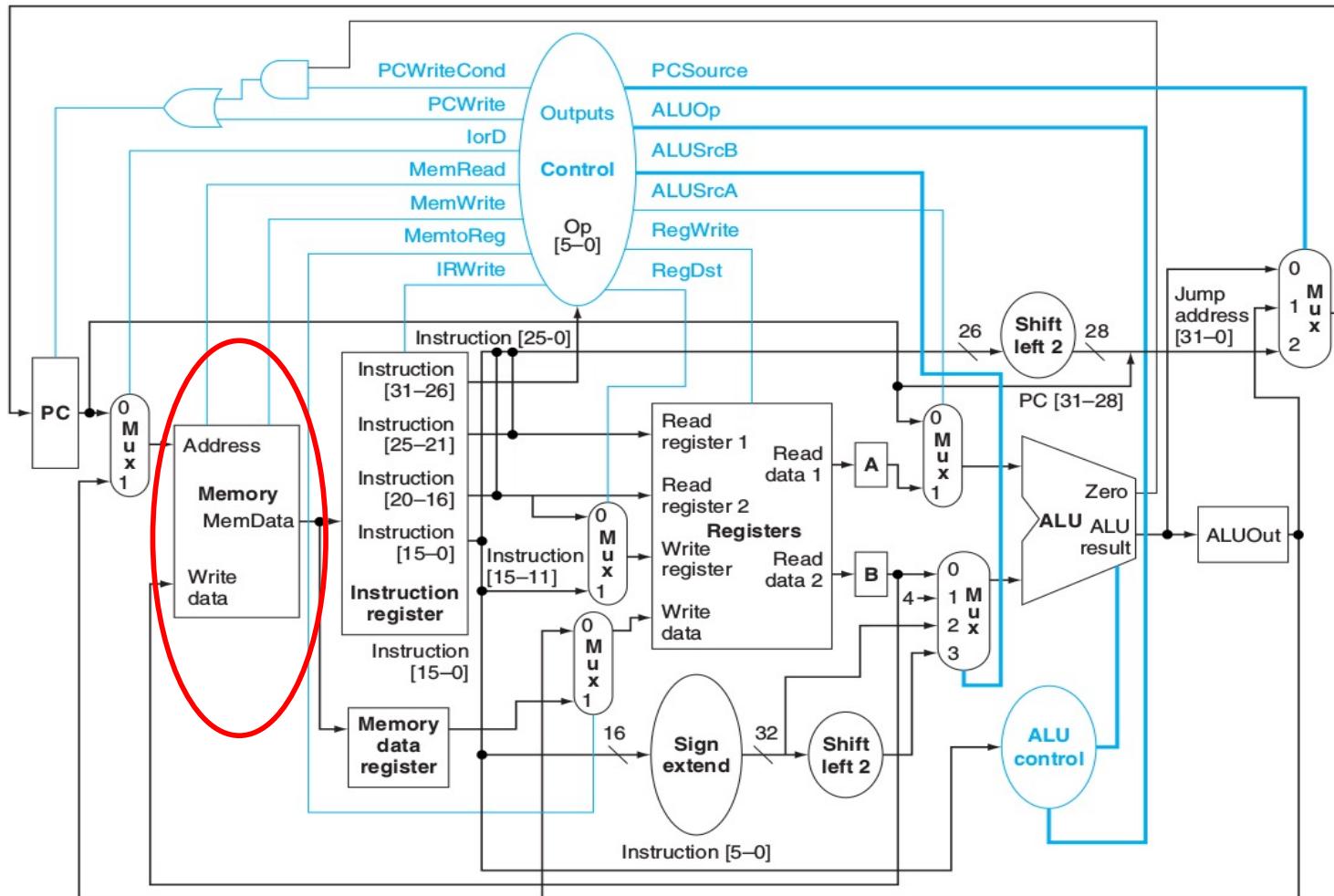


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Register file: 32 registri che contengono i dati utilizzati nel corso dell'esecuzione delle istruzioni; restituisce il contenuto dei due registri letti (indicati da ReadRegister1 e ReadRegister2) e scrive, nel caso di istruzioni load o R-type, il dato WriteData (rispettivamente, letto da memoria o output della ALU) nel registro indicato da WriteRegister

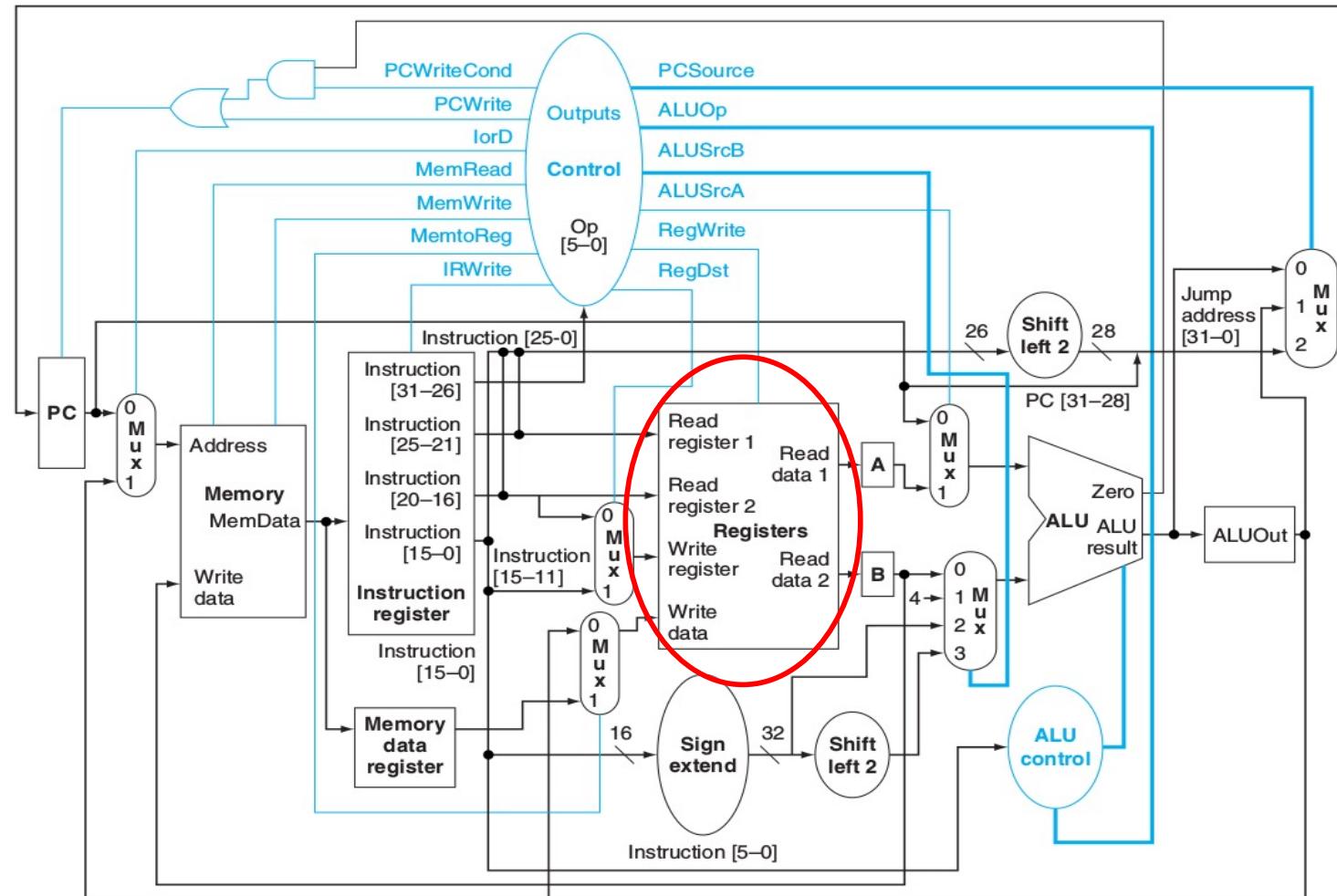


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

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 operazione aritmetico-logica 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: esegue operazione per confrontare i due registri in input

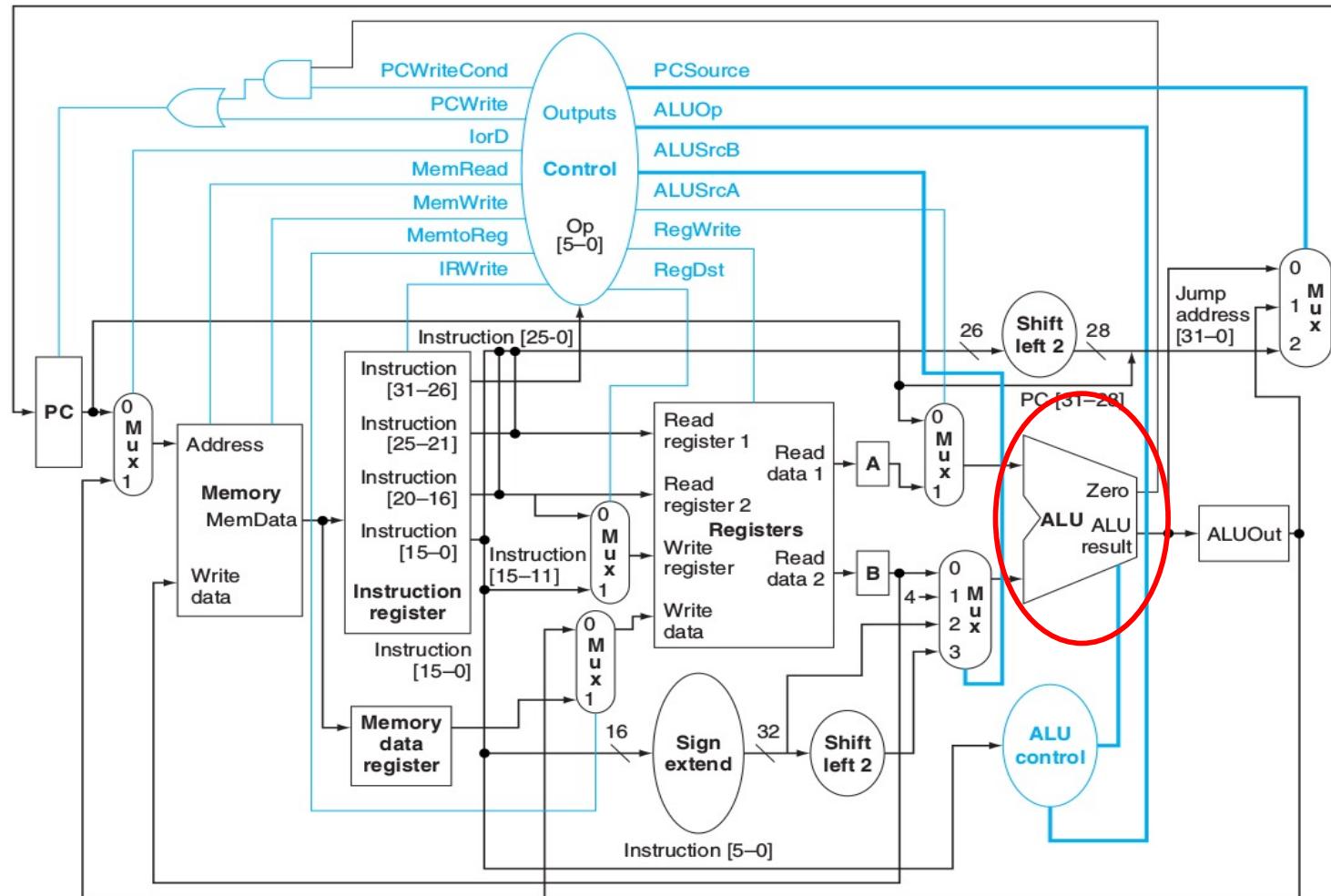


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

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

Shift Left 2: operano uno shift a sinistra dei bit in input

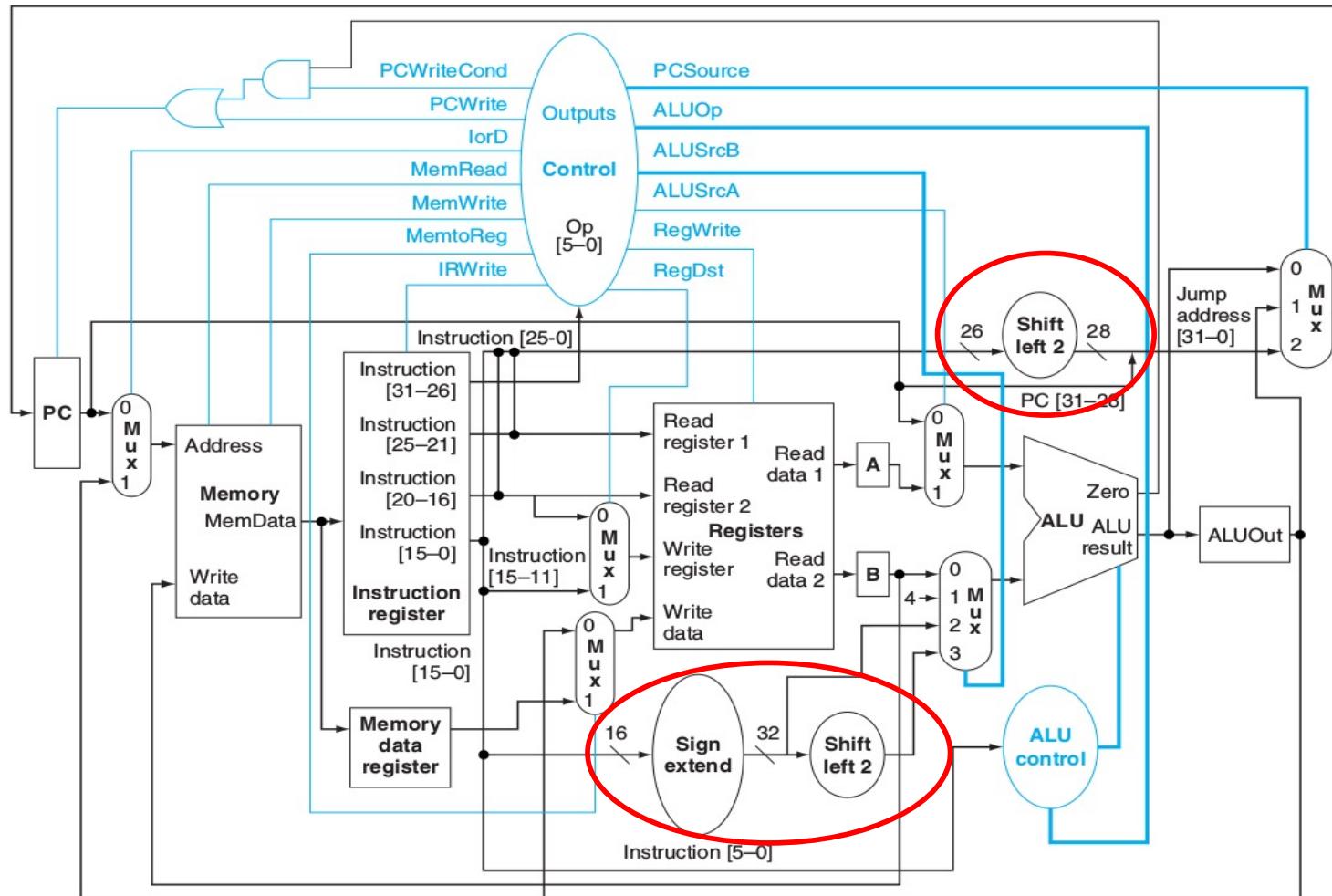


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

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. Nel caso di istruzione store, memorizza WriteData
- **Register file:** 32 registri che contengono i dati utilizzati nel corso dell'esecuzione delle istruzioni; restituisce il contenuto dei due registri letti (indicati dai ReadRegister1 e ReadRegister2) e scrive il dato WriteData nel registro indicato da 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:** operano uno shift a sinistra dei bit in input

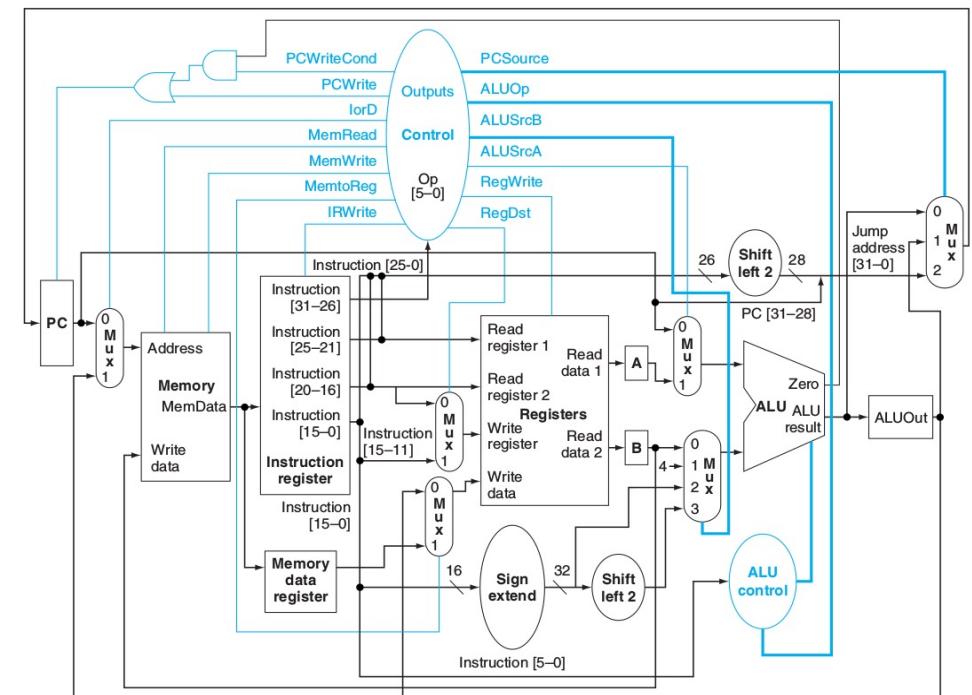


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Esercizio 1 – soluzione (registri)

- **PC:** contiene i 32 bit che indicano l'indirizzo dell'istruzione in esecuzione
- **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 ReadRegister1 e ReadRegister2 del RegisterFile
- **ALUOut:** contiene l'output della ALU

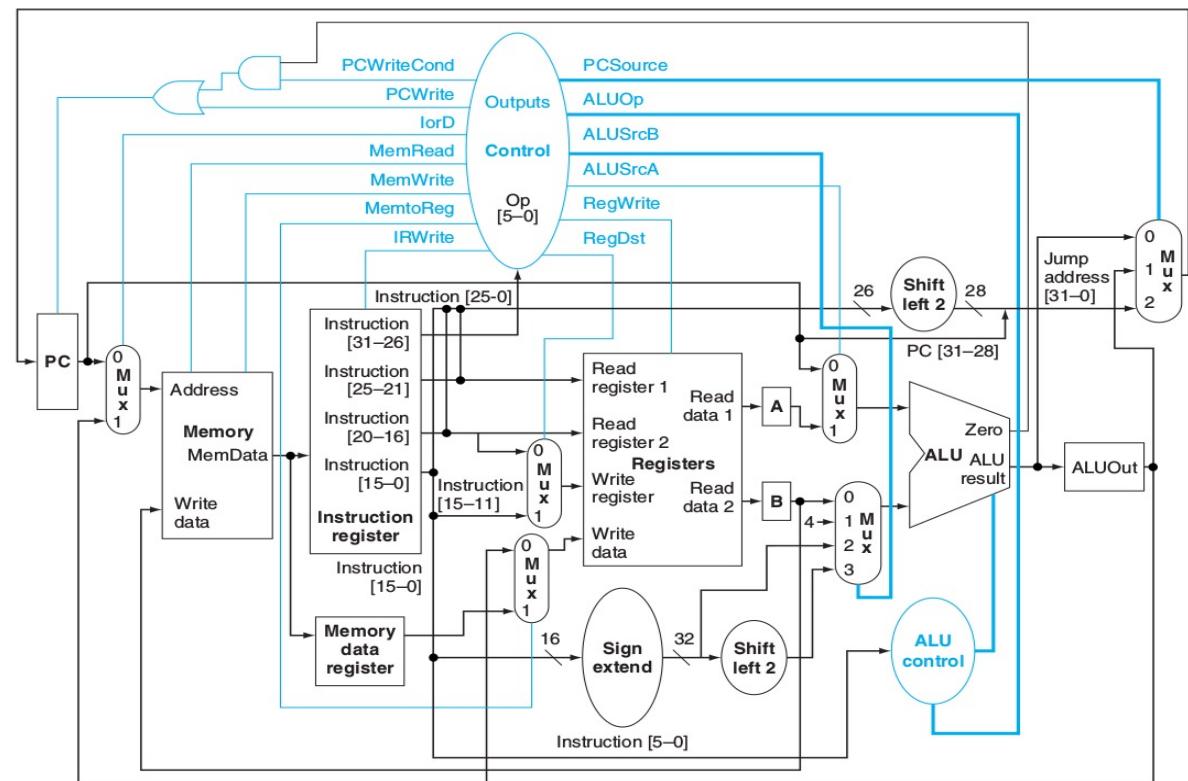


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

- 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 di istruzioni load o store)
- B: seleziona il gruppo di bit dell'IR che indica il registro in cui scrivere
 - IR[20:16] per istruzioni load
 - IR[15:11] per istruzioni R-type
- C: seleziona la sorgente per il dato da scrivere nel register file tra
 - MemoryDataRegister per istruzioni load
 - ALUOut per istruzioni R-type
- D: seleziona il primo operando dell'operazione aritmetico-logica eseguita dalla ALU tra
 - PC (per incremento +4 e calcolo valore di branch)
 - registro A (per istruzioni R-type e branch)
- E: seleziona il secondo operando dell'operazione aritmetico-logica eseguita dalla ALU
 - B (per istruzioni R-type e branch)
 - 4 (per aggiornamento del PC)
 - sign-extended di IR[15:0] (per istruzioni di accesso a memoria - lw/sw)
 - sign-extended e 2-shifted di IR[15:0] (per il calcolo del valore di branch)
- F: seleziona il valore per l'aggiornamento del PC
 - output della ALU: PC + 4
 - contenuto di ALUOut: PC + 2 shifted sign-extended branch field
 - jump target address (2 shifted left di IR[25:0] concatenato a PC[31:28])

Esercizio 1 – soluzione (multiplexer)

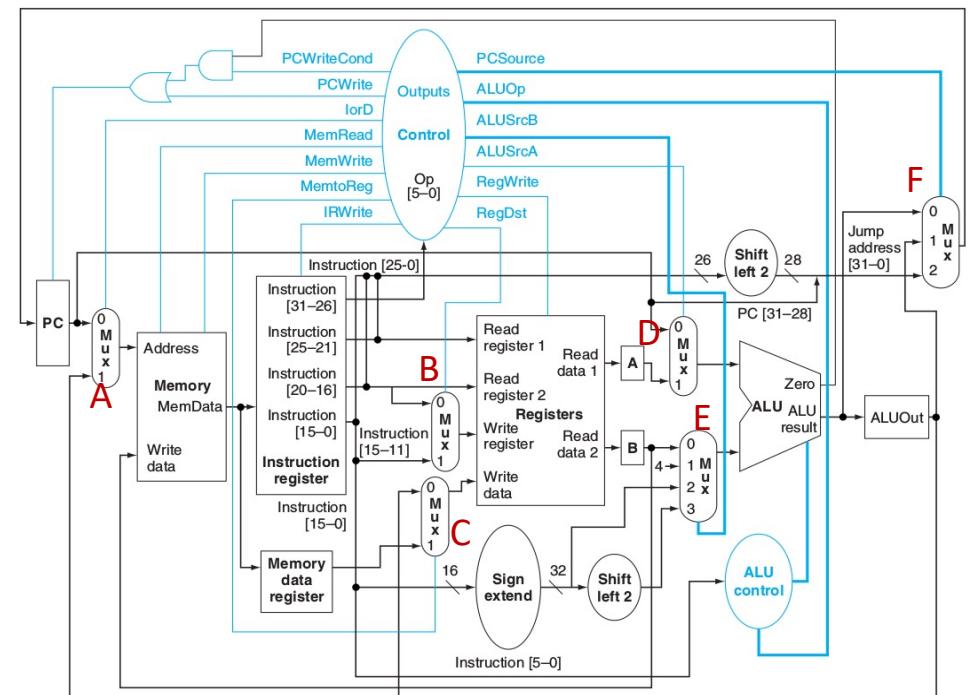


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

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?

Suggerimento: l'attivazione dei segnali di controllo per un datapath multiciclo è descritta da un automa a stati finiti

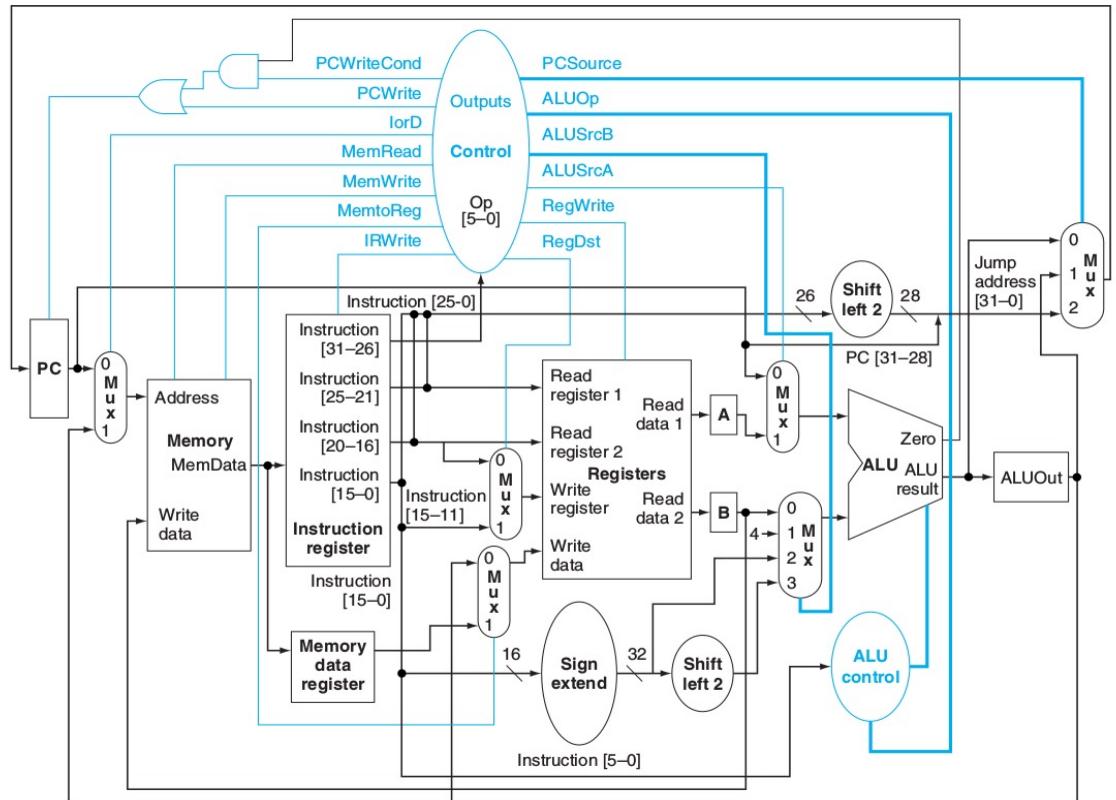


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Esercizio 2 – soluzione (1/2)

- L'attivazione dei segnali di controllo per un datapath multiciclo sono definiti dalla funzione `next_state` di un automa a stati finiti **in funzione dello stato corrente e dell'Opcode**
- I segnali di controllo sono gli output dell'automa
- **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 dell'istruzione da leggere e scrivere nell'IR
 - $MemRead = 1$: per leggere l'istruzione da memoria
 - $IRWrite = 1$: per scrivere l'istruzione letta nell'IR
 - $ALUSrcA=0$, $ALUSrcB=01$, $ALUOp=00$: per calcolare $PC + 4$
 - $PCWrite=1$ e $PCSource=00$: per scrivere il nuovo valore nel PC
- **Per tutti i tipi di istruzioni**, nella fase di decode (stato 1) sono impostati i segnali $ALUSrcA=0$, $ALUSrcB=11$ e $ALUOp=00$ 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)

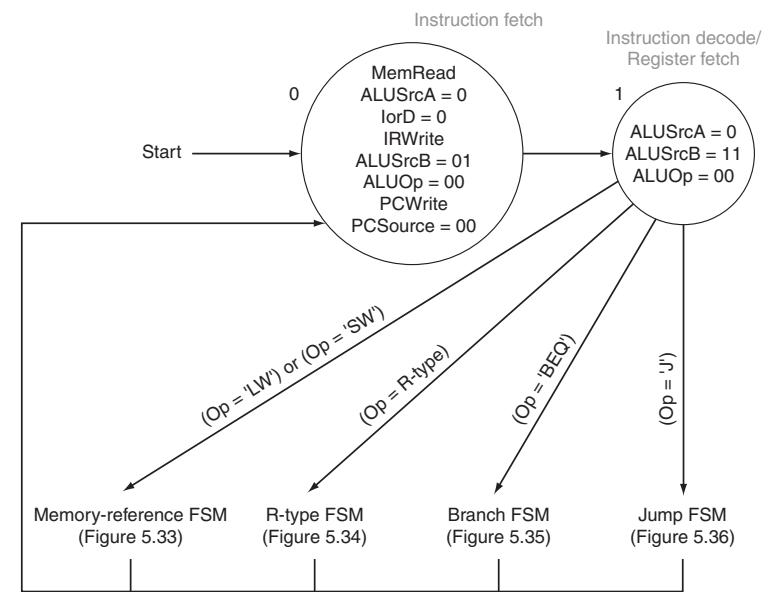
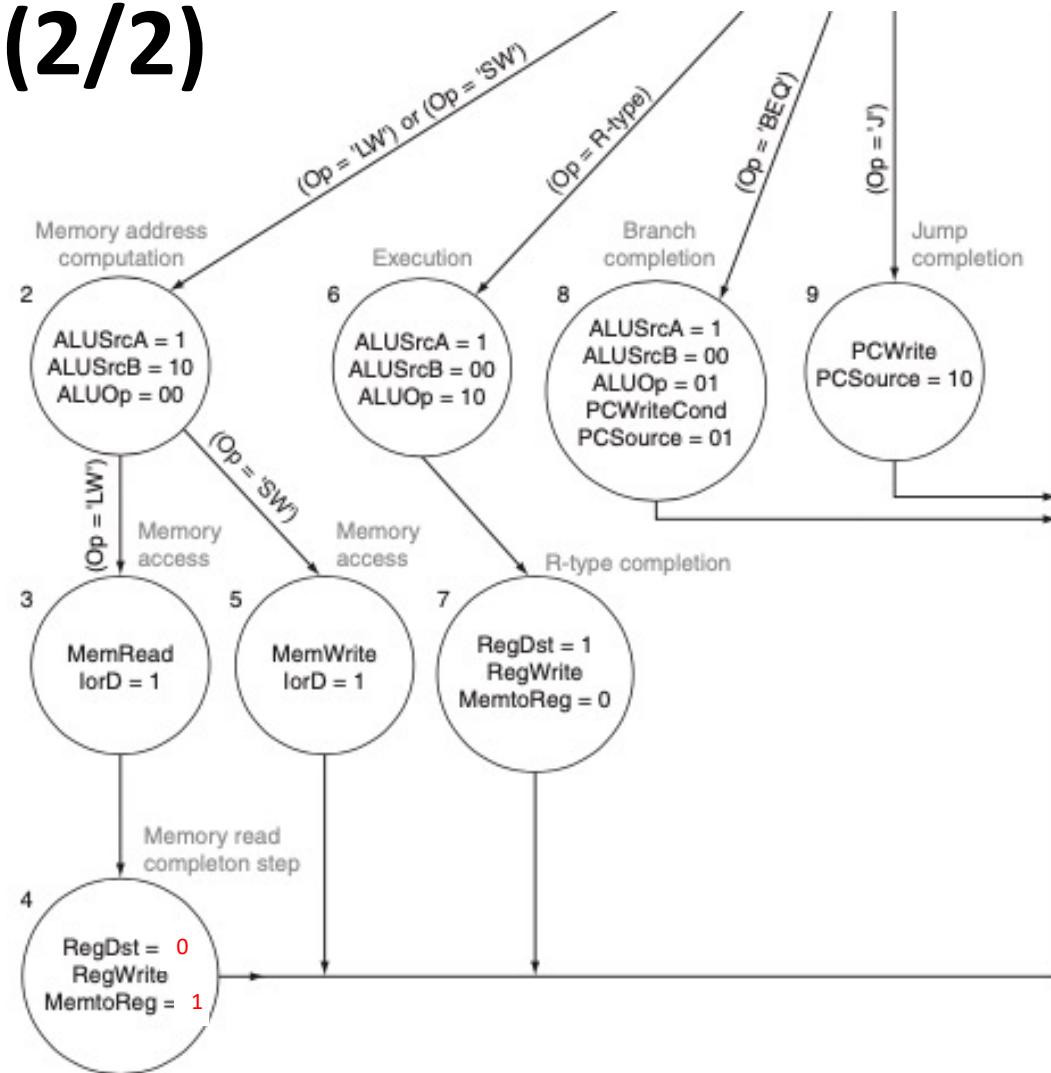


FIGURE 5.32 The instruction fetch and decode portion of every instruction is identical. These states correspond to the top box in the abstract finite state machine in Figure 5.31. In the first

Esercizio 2 – soluzione (2/2)

- le istruzioni di **jump** e di **branch**, dopo lo stato 1, richiedono ***un ciclo di clock*** per il loro completamento per la scrittura del nuovo valore del PC (nel caso di branch in funzione del risultato della sottrazione tra il contenuto di A e B)
- le istruzioni di tipo **R-type** e le istruzioni di **scrittura in memoria**, richiedono ***due cicli di clock*** per l'esecuzione di
 - una operazione della ALU (operazione R-type specificata da IR[5-0] o calcolo dell'indirizzo cui salvare il dato tramite $A = \text{RegFile}(IR[25-21]) + \text{sign_extended } IR[15-0]$)
 - una scrittura in un registro (per R-type) o in memoria (per store)
- le istruzioni di **lettura da memoria** richiedono ***tre cicli di clock*** per l'esecuzione di
 - una operazione della ALU per calcolare l'indirizzo del dato da leggere
 - una lettura da memoria
 - una scrittura nel Register file



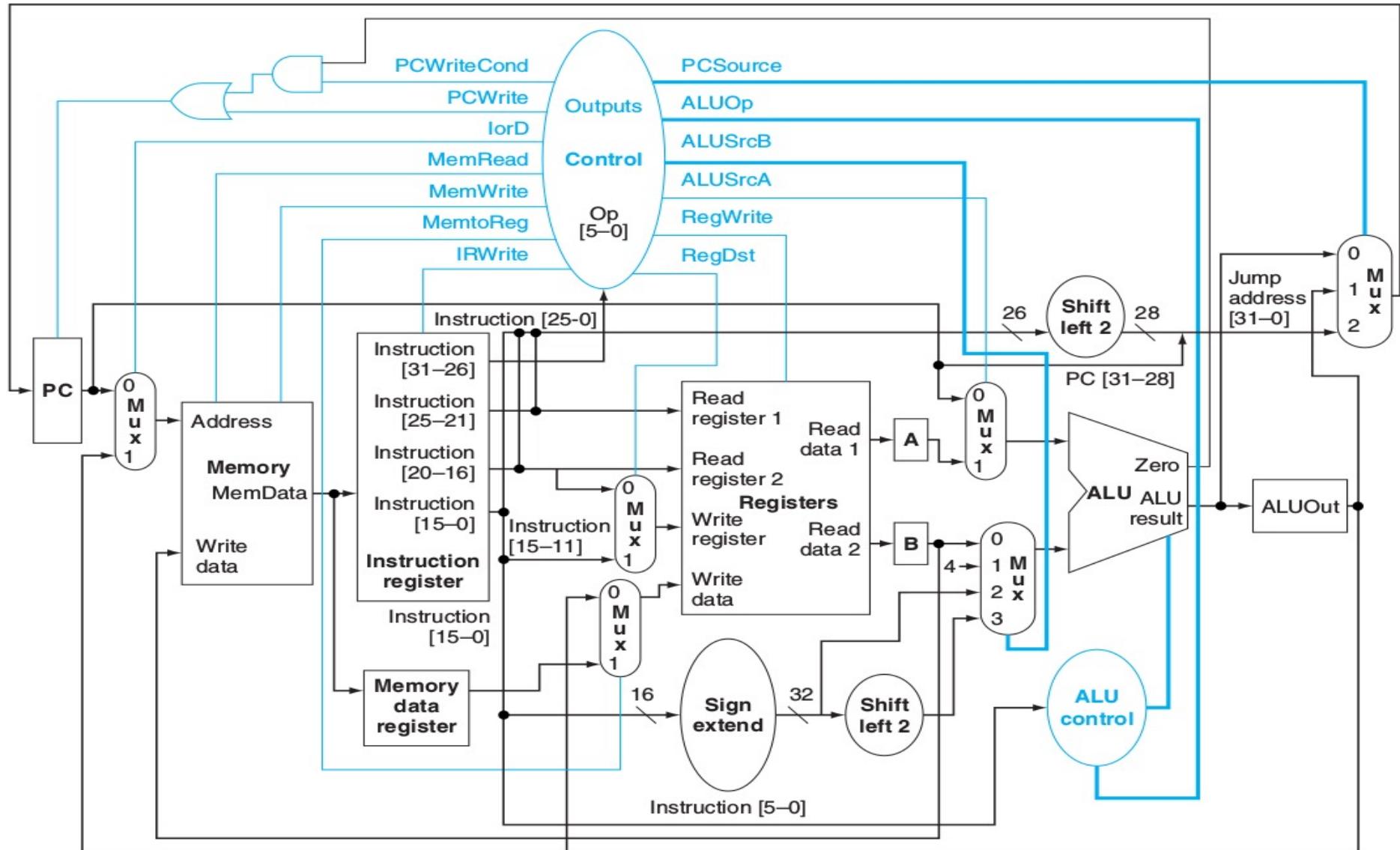


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Esercizio 3

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

Esercizio 3 – soluzione

- La ALU (nella fase di execute) è utilizzata per
 - **istruzioni R-type**: per esecuzione dell'operazione indicata dal funct_code (IR[5-0]) tra i due operandi
 - **istruzioni di accesso a memoria (lw/sw)**: per calcolare la somma tra l'offset (IR[15-0]) e il contenuto del registro base (RegFile(IR[25-21]))
 - **istruzioni di salto condizionato (branch)**: per la sottrazione dei contenuti dei due registri che compaiono nell'istruzione (IR[25-21] e IR[20-16]). *L'indirizzo di salto (branch value) è già in ALUout perché è stato calcolato nella fase di decode*
- La ALU non è usata nella fase di execute di istruzioni di **jump**

Esercizio 4

- Quanti cicli sono necessari per l'esecuzione dell'istruzione
add \$s0, \$s1, \$s2 ?

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

Addition (with overflow)

add rd, rs, rt	25-21	20-16	15-11	5-0
	0	rs	rt	rd

6 5 5 5 6

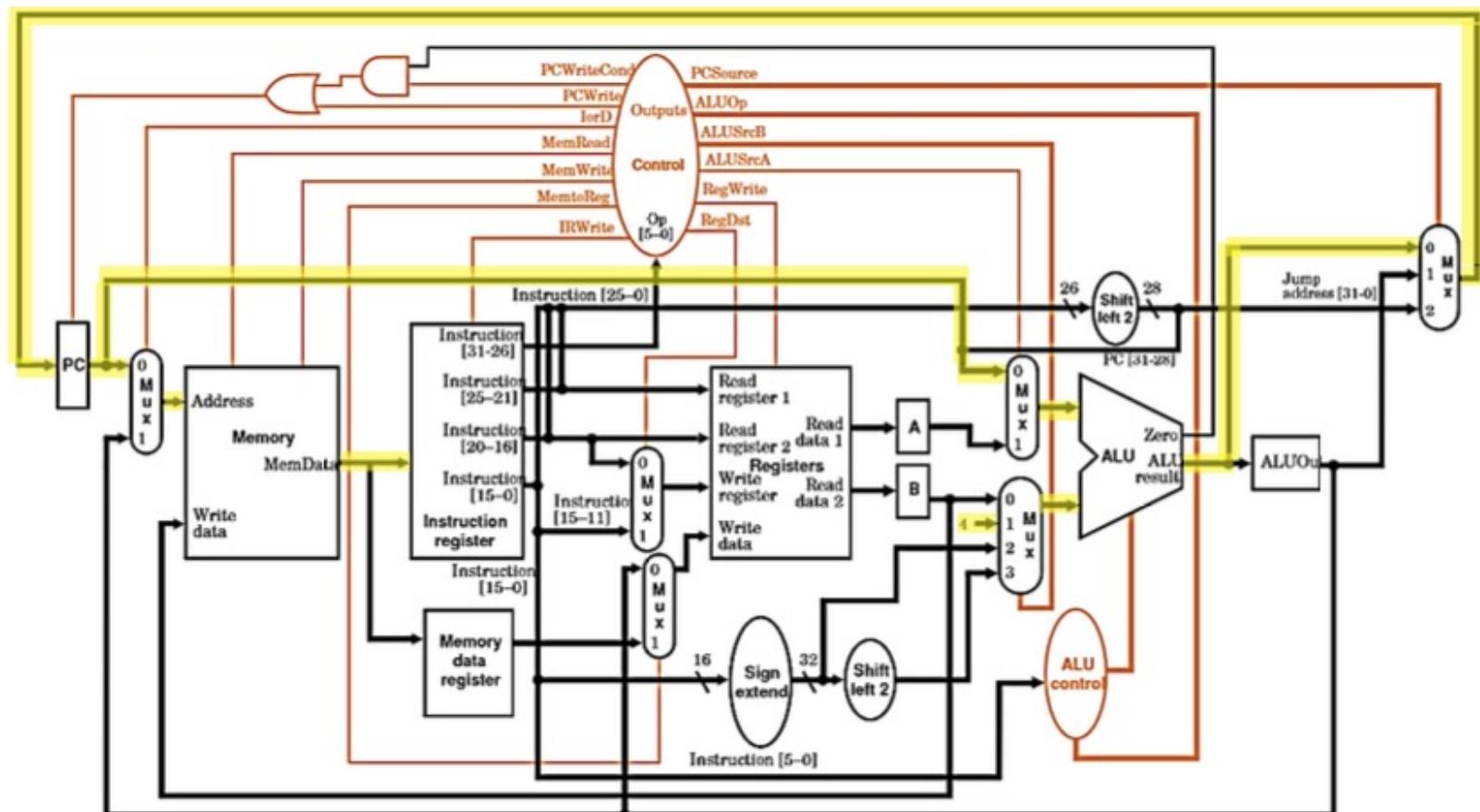
0x20

Esercizio 4 – soluzione (ciclo 1)

Fetch:

- scrittura in IR di Memory[PC]
- PC:=PC+4

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



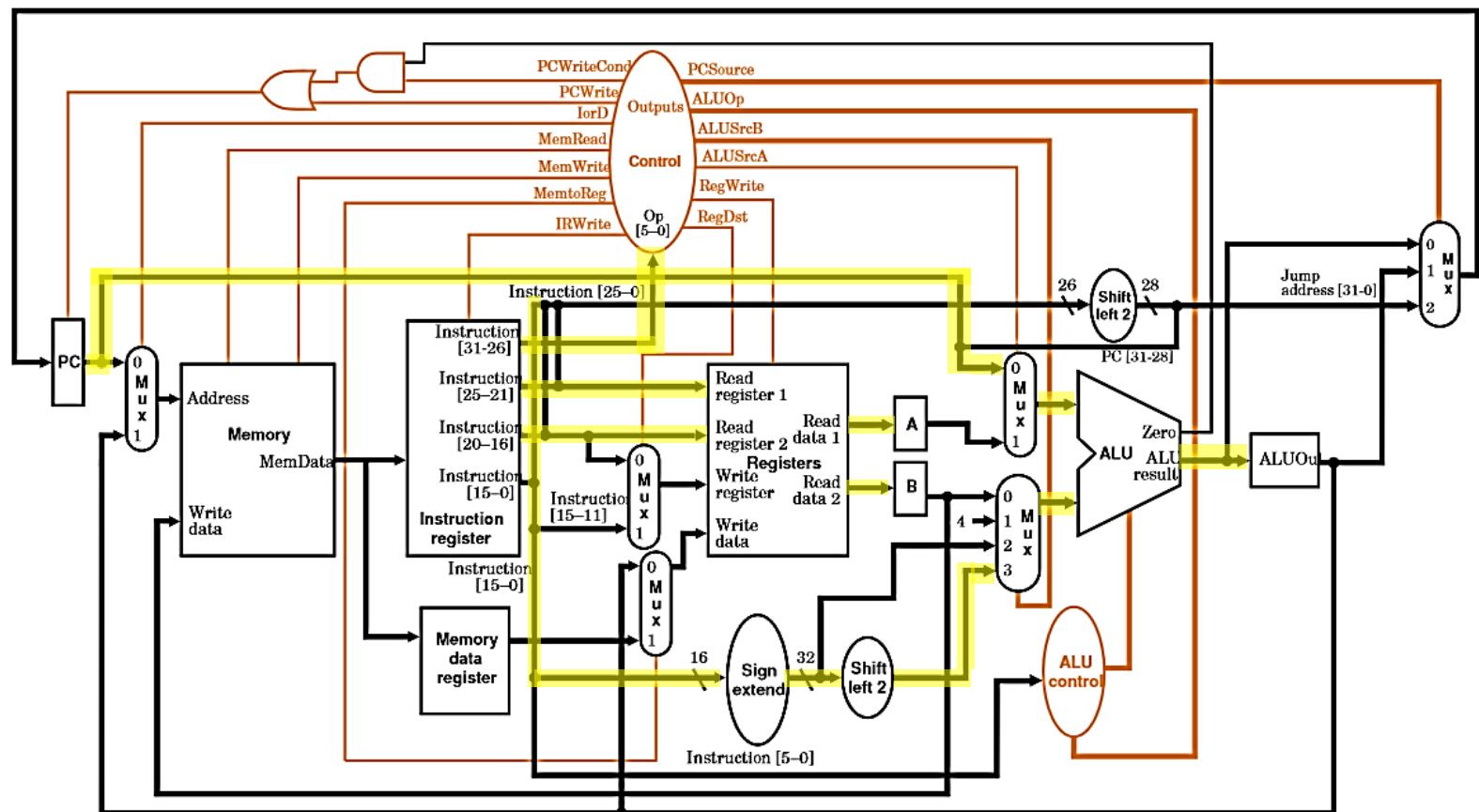
Esercizio 4 – soluzione (ciclo 2)

Decode:

- $ALUout = PC + shift_left(sign_ext(IR[15-0]))$
- $A = RegFile(IR[25-21])$
- $B = RegFile(IR[20-16])$

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



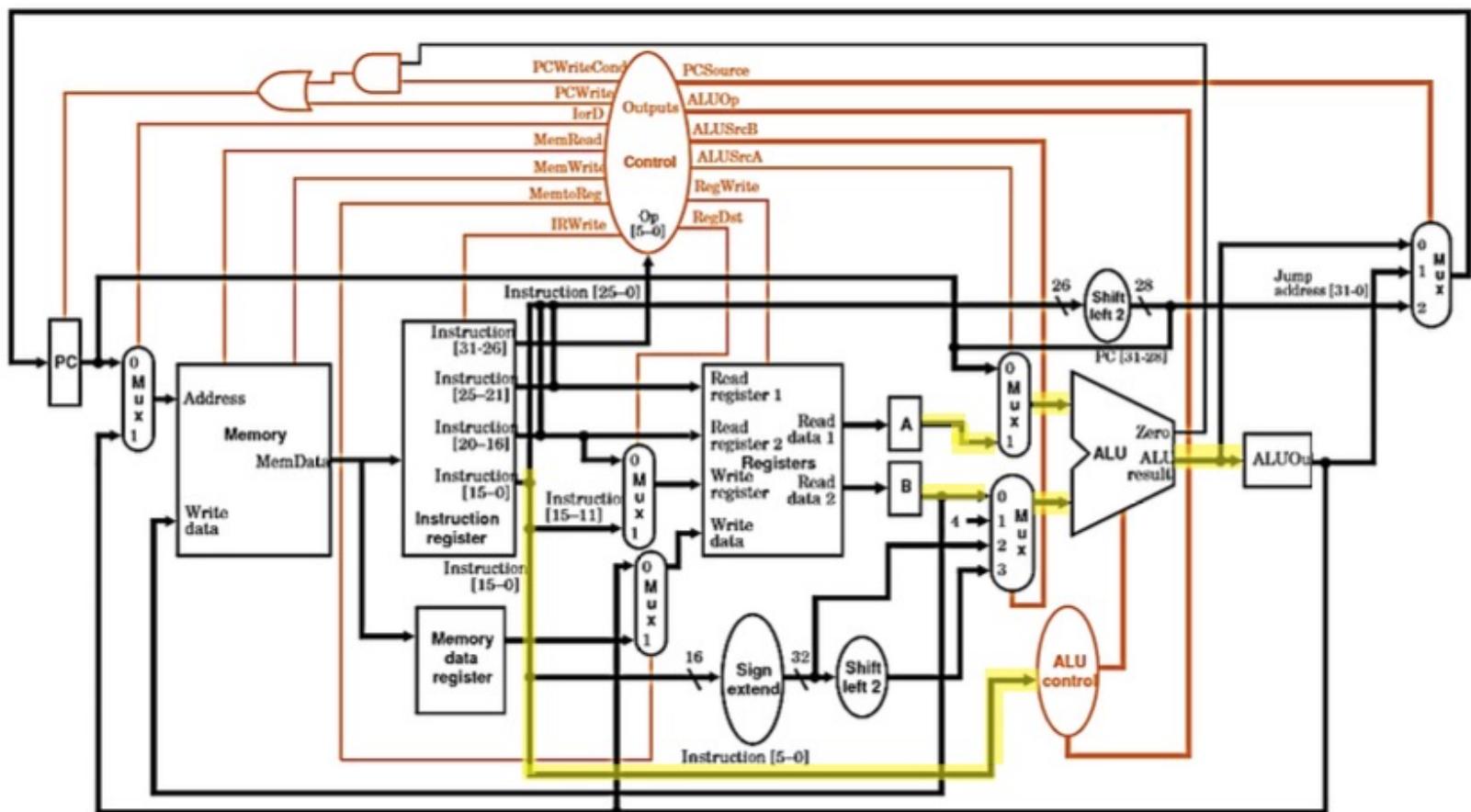
Esercizio 4 – soluzione (ciclo 3)

Execute 1:

- $ALUout = op(A, B)$

In questo caso op è la somma, in generale è determinata da $IR[5-0]$

Signal	Value
ALUOp	10
ALUSrcB	00
ALUSrcA	1

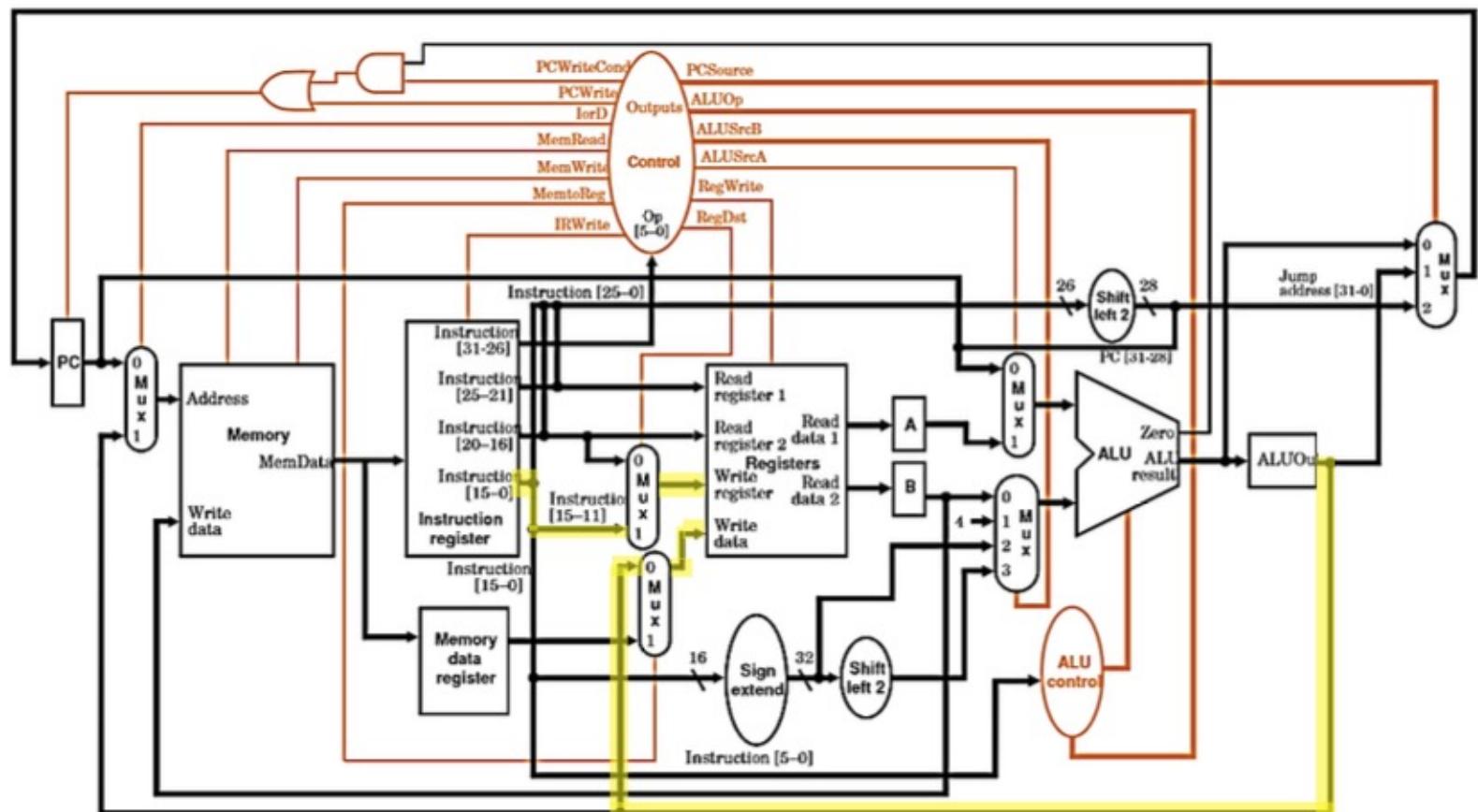


Esercizio 4 – soluzione (ciclo 4)

Execute 2:

- RegFile(IR[15-11]) = ALUout

Signal	Value
MemtoReg	0
RegWrite	1
RegDst	1



Esercizio 5

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

Esercizio 5 – soluzione

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

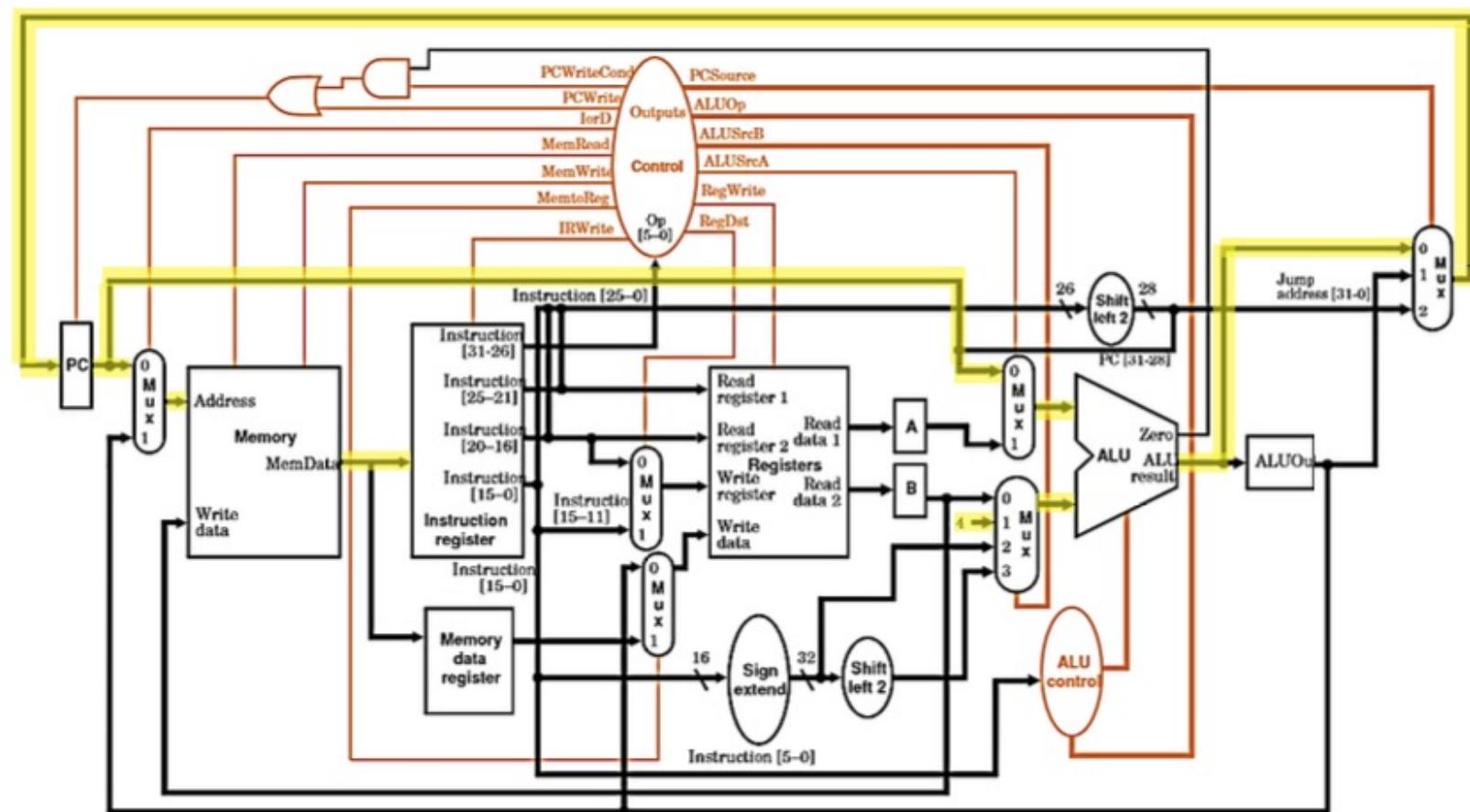
Branch on equal

beq rs, rt, label

	25-21	20-16	15-0
4	rs	rt	Offset
6	5	5	16

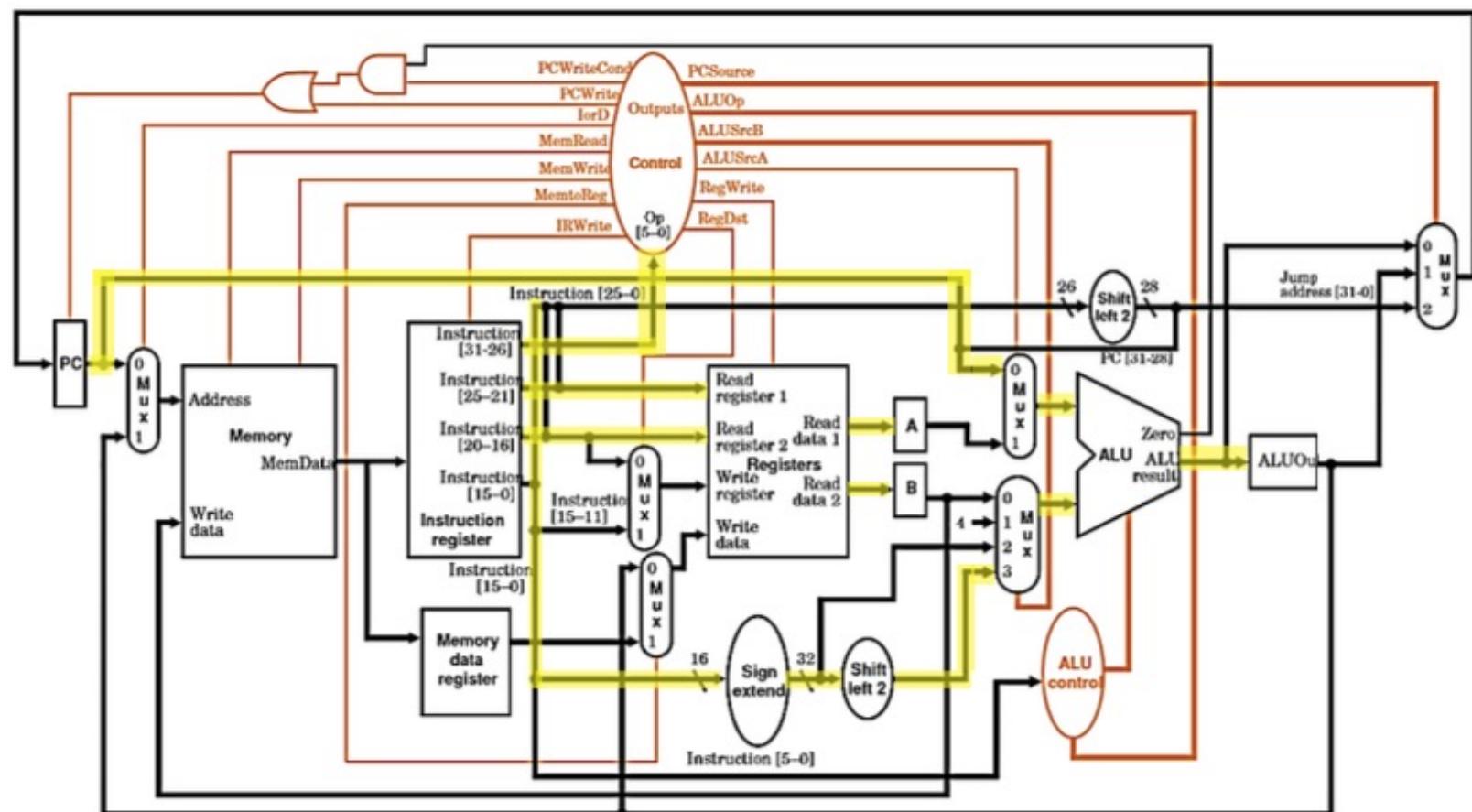
Esercizio 5 – soluzione (ciclo 1)

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



Esercizio 5 – soluzione (ciclo 2)

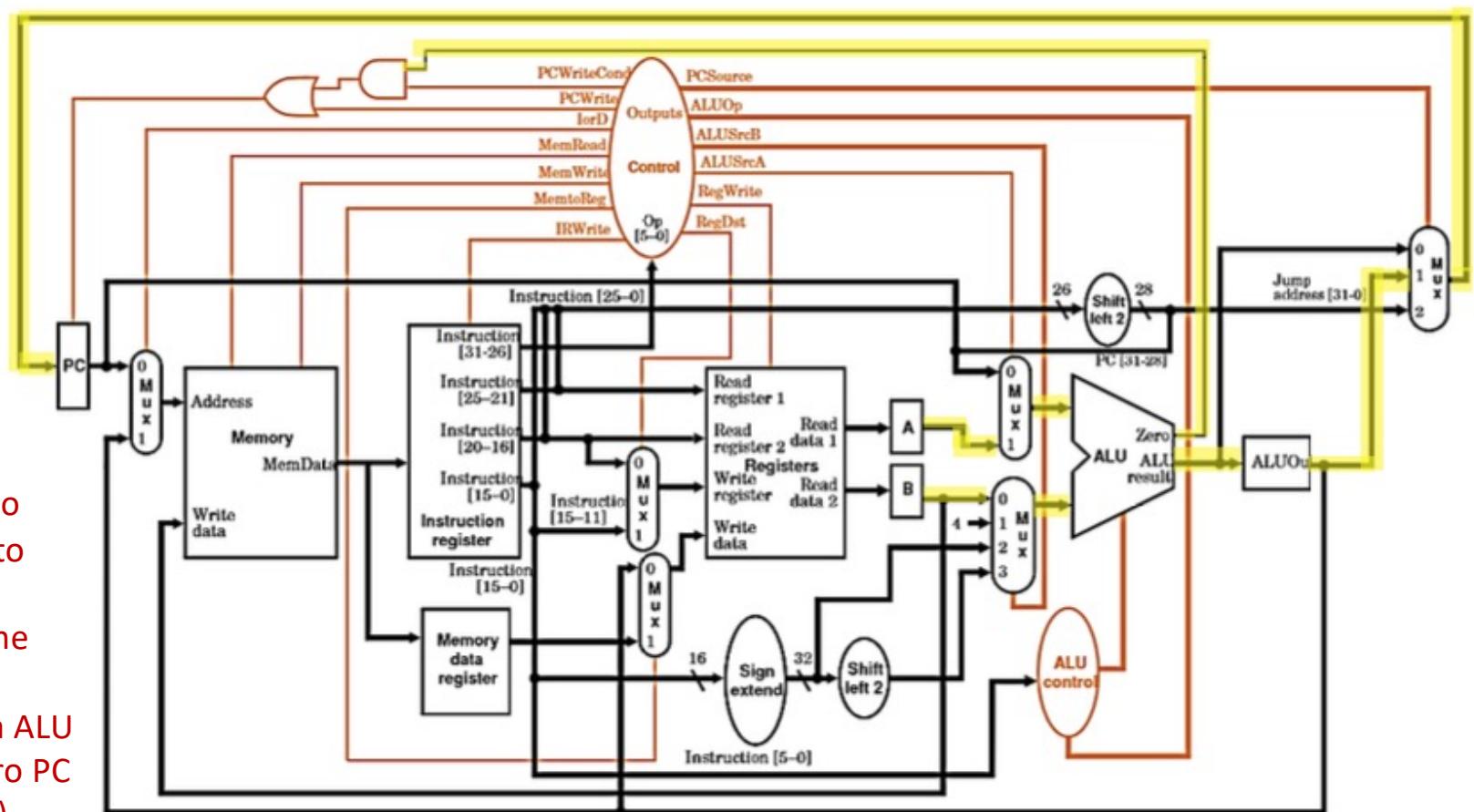
Signal	Value
ALUOp	00
ALUSrcB	11
ALUSrcA	0



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 la sottrazione 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!)



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 e la semantica dell'istruzione
 - Tabelle della fig 5.29 pag. 324 (capitolo 5 disponibile in area elearning) con dettagli su convenzioni impostazioni segnali

N.B. diverse soluzioni possibili!

Esercizio 6 – soluzione (1/2)

- L'istruzione che si vuole implementare determina l'aggiornamento del PC con i 32 bit contenuti nel registro il cui numero è indicato dai 5 bit [25-21] dell'istruzione

da pag. A-64

Jump register	25-21	20-6	5-0
jr rs	0 6	rs 5	0 15 8 6

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

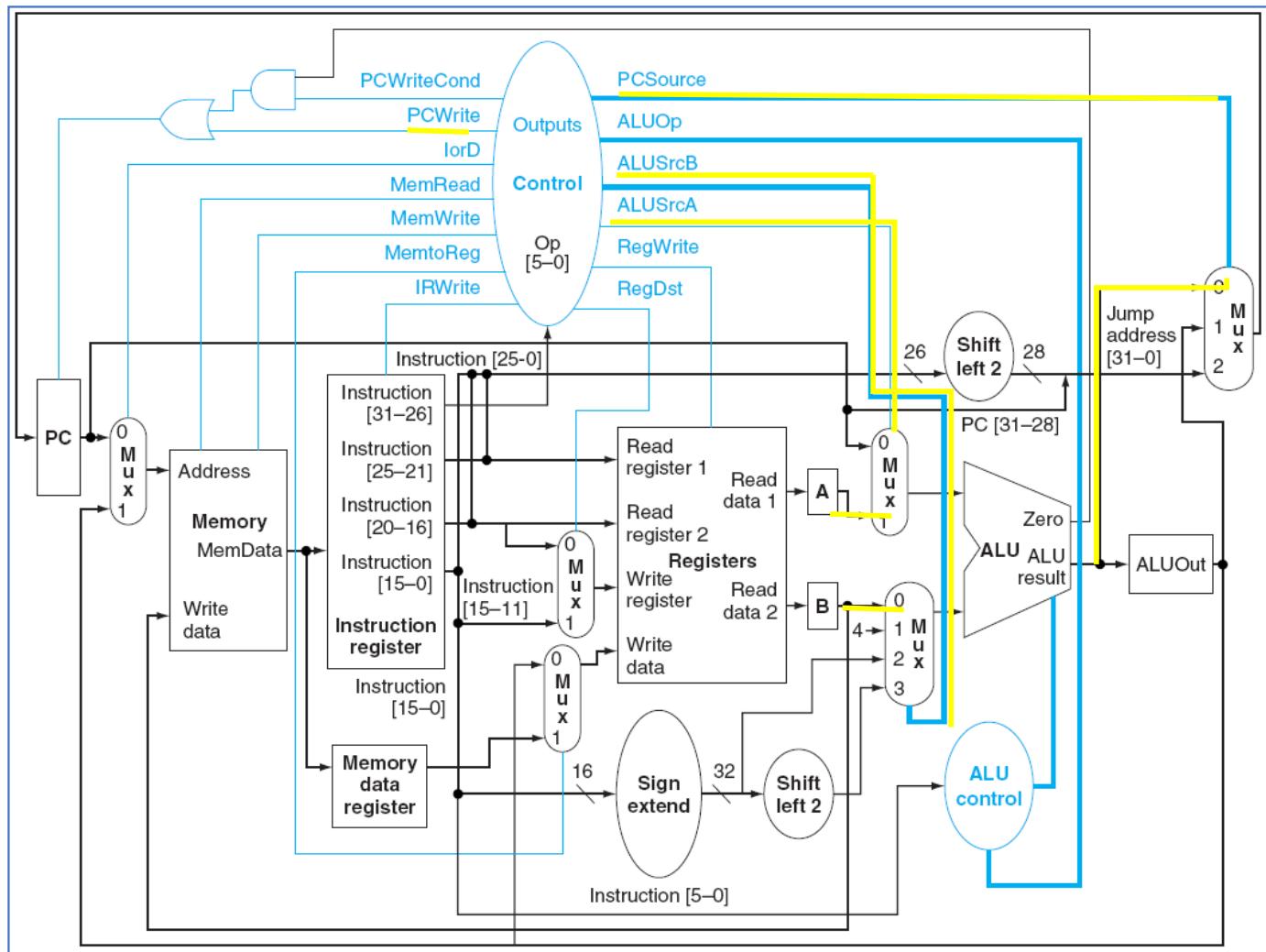
- Sappiamo inoltre che, nella fase di decode di ogni istruzione, viene scritto
 - in A il contenuto del registro il cui numero è indicato dai 5 bit [25-21] dell'istruzione
 - in B il contenuto del registro il cui numero è indicato dai 5 bit [20-16] dell'istruzione (in questo caso tali bit sono a 0 e quindi B conterrà 0, il contenuto del registro \$zero)

Esercizio 6 – soluzione (2/2)

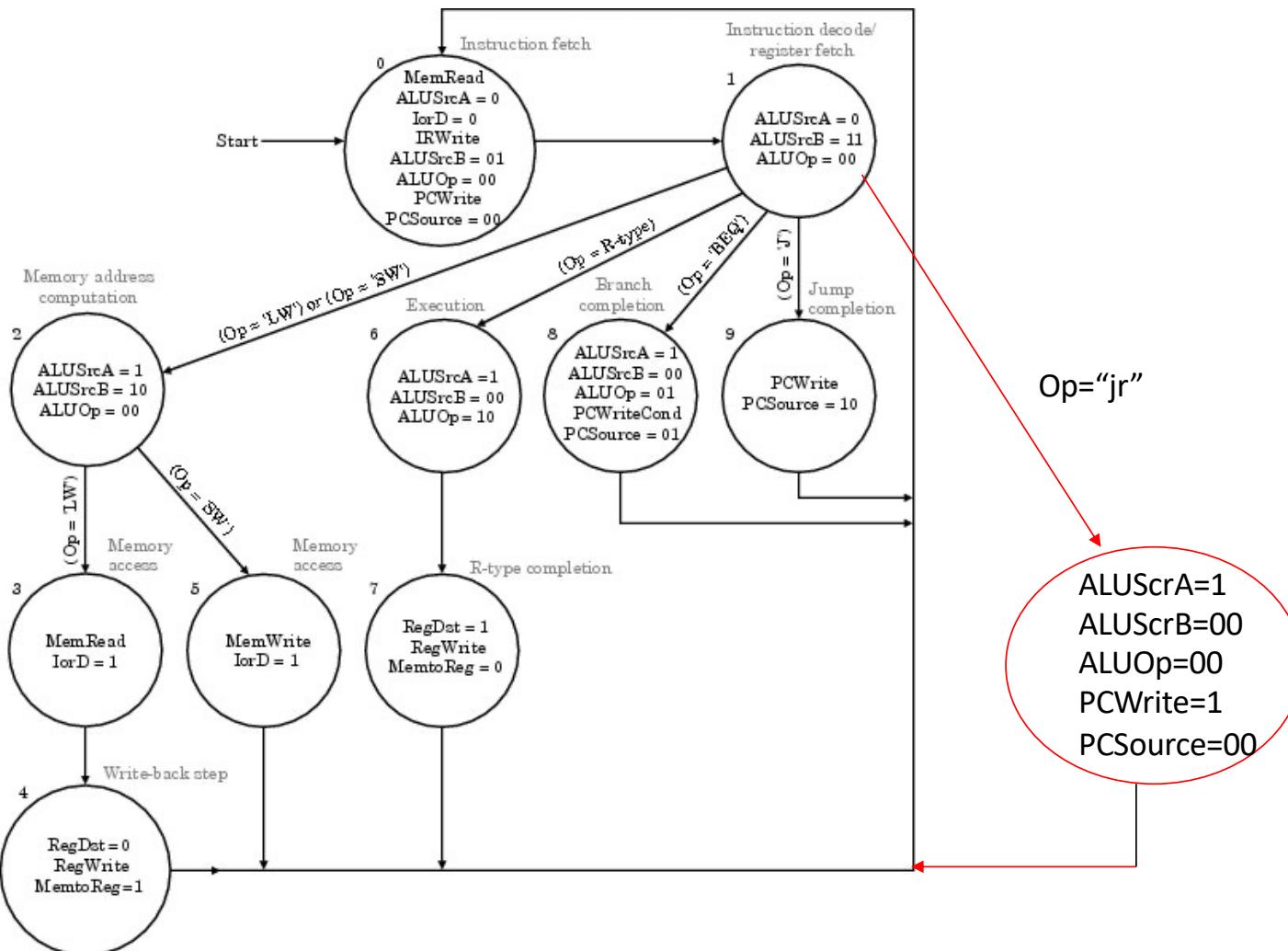
- Per realizzare l'effetto desiderato, potremo impostare i segnali di controllo in modo tale da scrivere in PC il risultato della somma tra il contenuto del registro A e del registro B
- Ovvero
 - selezionare il registro A come primo input per la ALU (ALUScrA=1)
 - selezionare il registro B come secondo input per la ALU (ALUScrB=00)
 - far eseguire una somma alla ALU (ALUOp=00)
 - abilitare la scrittura del PC (PCWrite=1)
 - selezionare l'output della ALU come valore da scrivere nel PC (PCSource=00)

Esercizio 6 – soluzione (jr rs)

ALUScrA=1
ALUScrB=00
ALUOp=00
PCWrite=1
PCSsource=00



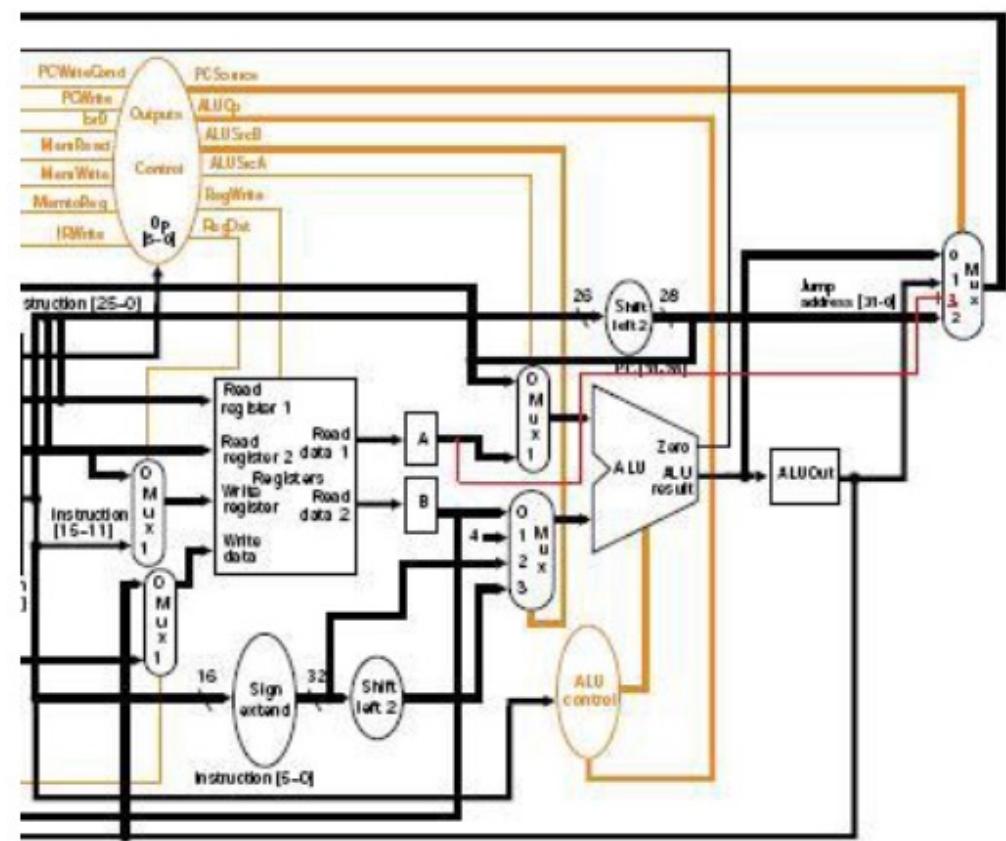
Esercizio 6 – soluzione (jr rs)



Esercizio 6 – soluzione alternativa

In alternativa potremmo 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 per la sua selezione).

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



Esercizio 7

- Modificare lo schema del datapath multiciclo e la relativa FSM, in modo da aggiungere al set di istruzioni del processore MIPS, l'istruzione ISA **addi**

Addition immediate (with overflow)

	25-21	20-16	15-0
addi rt, rs, imm	8	rs	rt
	6	5	16

Put the sum of register rs and the sign-extended immediate into register rt.

- Suggerimento: sono sufficienti modifiche alla FSM

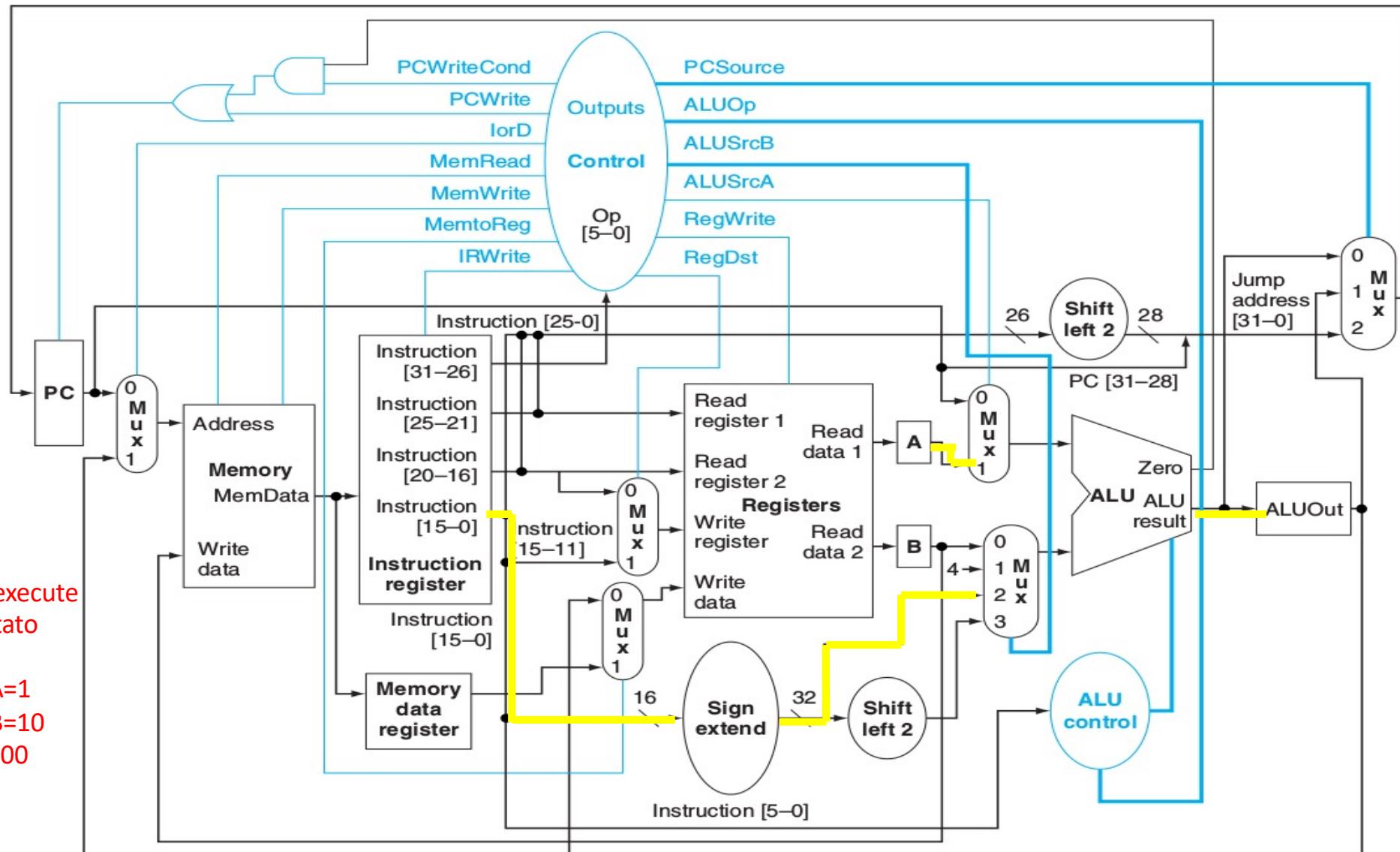


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

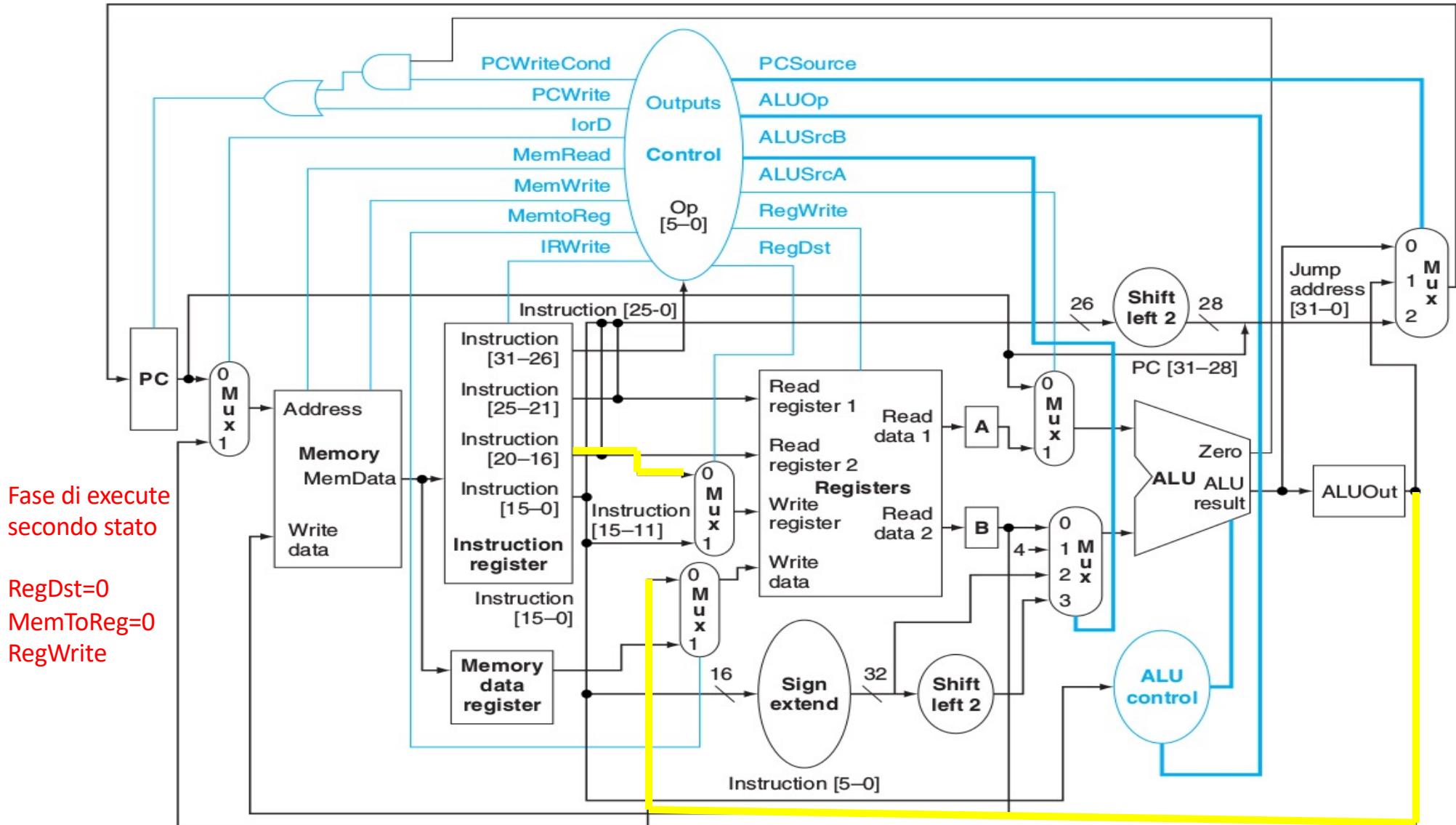
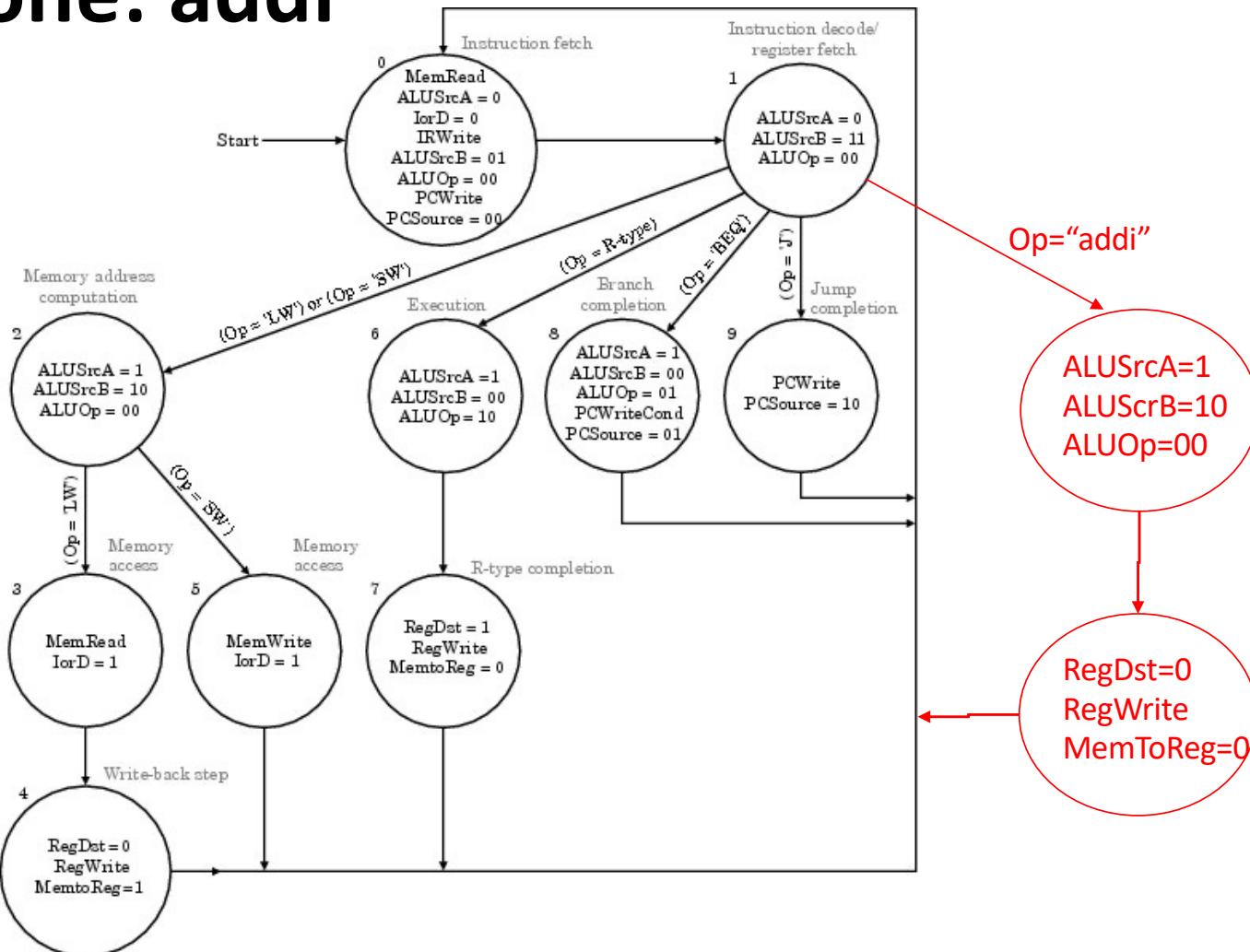


FIGURE 5.28 The complete datapath for the multicycle implementation together with the necessary control lines. The con-

Soluzione: addi



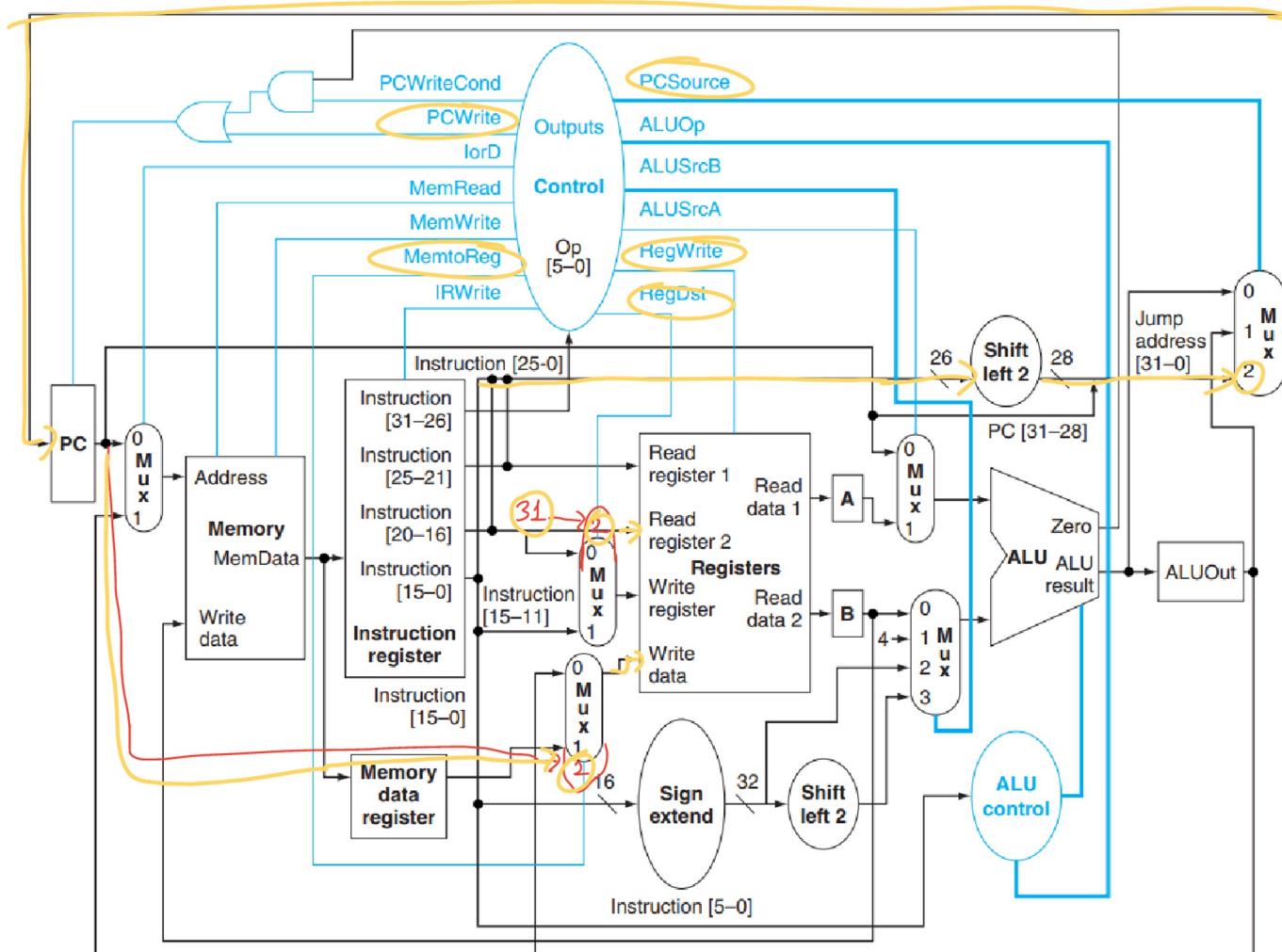
Esercizio 8

- Modificare lo schema del datapath multiciclo e la relativa FSM, in modo da aggiungere al set di istruzioni del processore MIPS, l'istruzione **jal target**
- Ricordiamo che l'esecuzione di questa istruzione ha l'effetto di
 - scrivere nel registro \$ra (\$31) il valore contenuto in PC
 - aggiornare il contenuto di PC con il valore specificato da «target» come in una comune istruzione jump

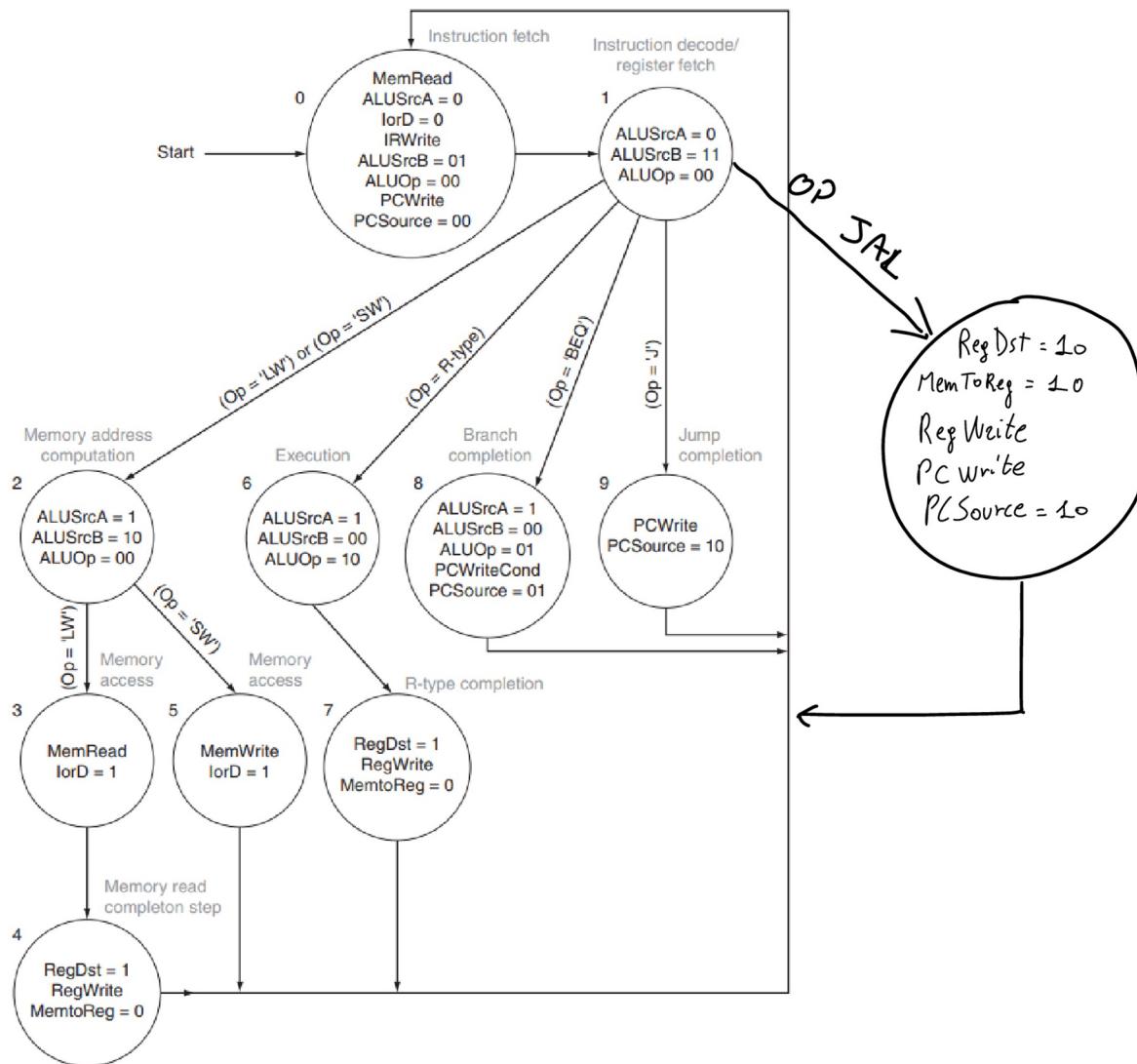
Esercizio 8 – soluzione

Possiamo effettuare entrambe le operazioni in un solo ciclo di clock

- Per quanto riguarda il salto incondizionato all'indirizzo target, l'hardware non necessita modifiche e impostiamo come in una jump PCWrite=1 e PCSource=10
- Per scrivere nel registro \$31 il valore corrente di PC, dovremo
 - dare il valore 31 all'ingresso Write Register del Register file
 - dare il valore di PC all'ingresso Write data del Register file
 - abilitare la scrittura del Register file
- Modifichiamo i due multiplexer che comandano questi due input al Register file
 - aggiungiamo un ingresso al multiplexer comandato dal segnale RegDst, assegnandogli il valore 31 come costante (e 2 come valore di selezione)
 - aggiungiamo un ingresso al multiplexer comandato dal segnale MemToReg e lo collegiamo all'uscita del PC (selezionandolo con il valore 2)



RegDst=10
 RegWrite
 MemToReg=10
 PCWrite
 PCSource=10



Esercizio 9

Facendo riferimento alla CPU multiciclo vista a lezione e alla relativa logica di controllo per l'implementazione delle istruzioni **add, and, beq, j, lw, or, slt, sub, sw**, si chiede quali siano le istruzioni durante la cui intera realizzazione (da fetch a execute) alla ALU, in termini di attivazione dei suoi ingressi di controllo viene richiesto di

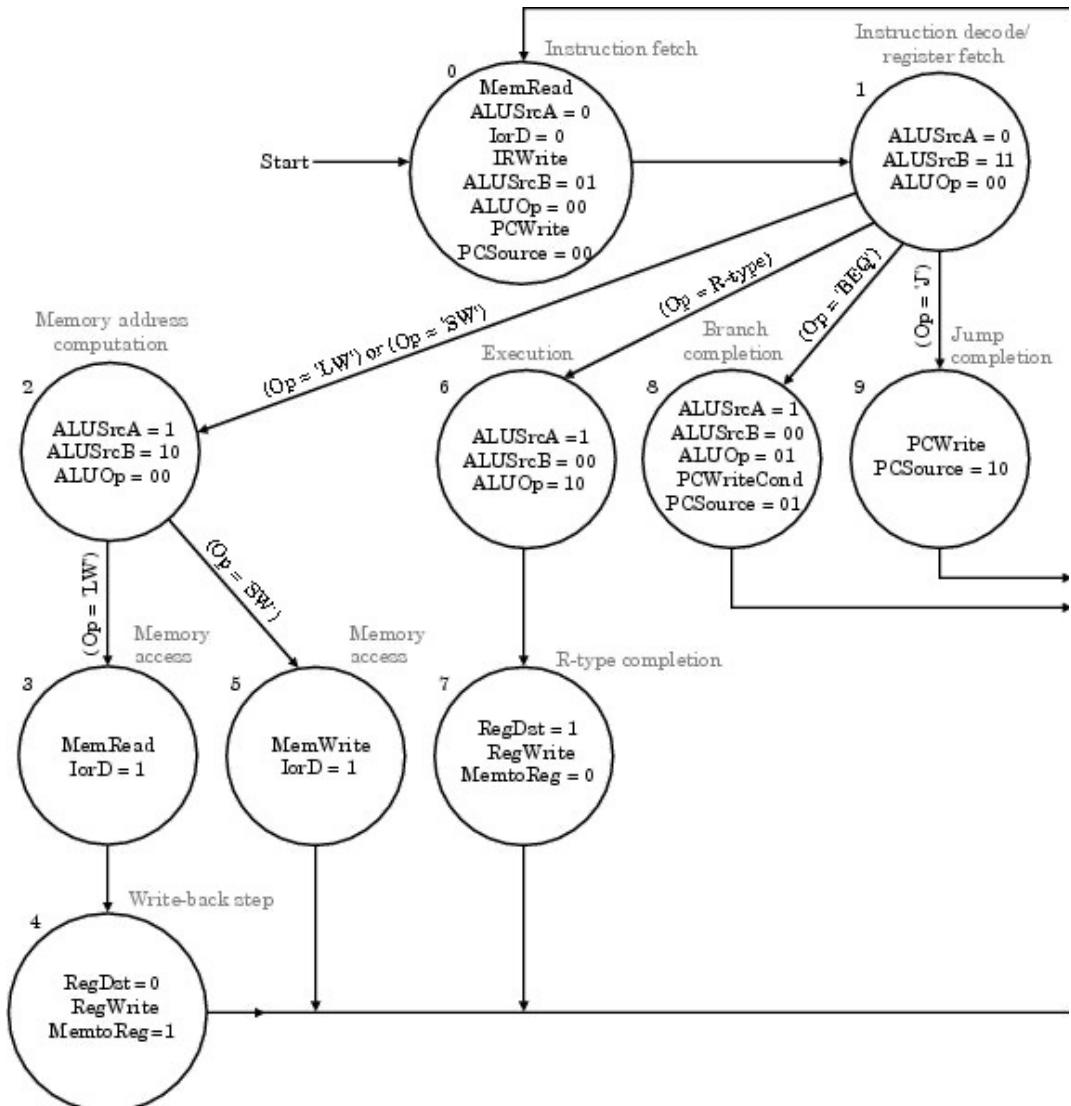
- effettuare una operazione di tipo sottrazione
- effettuare esattamente 3 operazioni di tipo somma
- utilizzare il contenuto del registro B
- ...

Esercizio 9 – soluzione

istruzioni **add, and, beq, j, lw, or, slt, sub, sw**

intera realizzazione (da fetch a execute)

- effettuare una operazione ALU di tipo sottrazione → sub, beq
- effettuare esattamente 3 operazioni ALU di tipo somma → add, lw, sw
- utilizzare il contenuto del registro B → add, and, beq, or, slt, sub



N.B.

Ulteriori esercizi relativi alla modifica del Datapath multiciclo come attività di laboratorio!

Seguono alcune anticipazioni ...

Esercizio Lab 3

- Sapendo che l'istruzione “jump register immediate” jrim imm(\$rs) salta all'indirizzo \$rs+imm e che imm è un valore a 16-bit
- Ovvero, l'istruzione è così definita:
 - 6 bit OPCODE 111111 [bit 26-31]
 - 5 bit \$rs REGISTER FIELD [bit 21-25]
 - NON SPECIFICATO \$rt [bit 16-20]
 - 16 bit IMMEDIATE [bit 0-15]
- Sullo schema del datapath multiciclo specificare quali segnali di controllo devono essere utilizzati per eseguire l'istruzione e modificare quindi la relativa macchina a stati finiti (FSM)
- Se necessarie, mostrare le minime modifiche da effettuare all'hardware per implementare l'istruzione

Esercizio Lab 4

- Modificare lo schema del datapath multiciclo e la relativa FSM, in modo da aggiungere al set di istruzioni del processore MIPS, l'istruzione **SWAP \$rs, \$rt**, che scambia il contenuto di due registri
- Si supponga che l'istruzione preveda un terzo registro \$rd con contenuto uguale a \$rs
- Trovare la soluzione che minimizzi il numero di cicli di clock necessari e che non preveda la modifica del register file durante l'esecuzione dell'istruzione

Esercizio Lab 5

- Modificare lo schema del datapath multiciclo e la relativa FSM in modo da aggiungere al set di istruzioni del processore MIPS l'istruzione **add3 \$rd, \$rs, \$rt, \$rx**
 - dove \$rd contiene la somma di \$rs, \$rt e \$rx
 - add3 è un'istruzione tipo R-Type “modificata” secondo il seguente schema
 - 6 bit OPCODE [bit 31-26]
 - 5 bit \$rd [bit 25-21]
 - 5 bit \$rs [bit 20-16]
 - 5 bit \$rt [bit 15-11]
 - 6 bit non usati [bit 10-5]
 - 5 bit \$rx [bit 4-0]
- La soluzione proposta non dovrà prevedere la modifica del Register file né aggiungere una ulteriore ALU

Jrim soluzione

Modificare lo schema del datapath multiciclo e la relativa FSM in modo da aggiungere al set di istruzioni del processore MIPS l'istruzione jrim imm(\$rs) definita come segue:

- 6 bit OPCODE [bit 26-31] (001001)
- 5 bit \$rs REGISTER FIELD [bit 21-25]
- \$rt non usato [bit 16-20]
- 16 bit IMMEDIATE [bit 0-15]

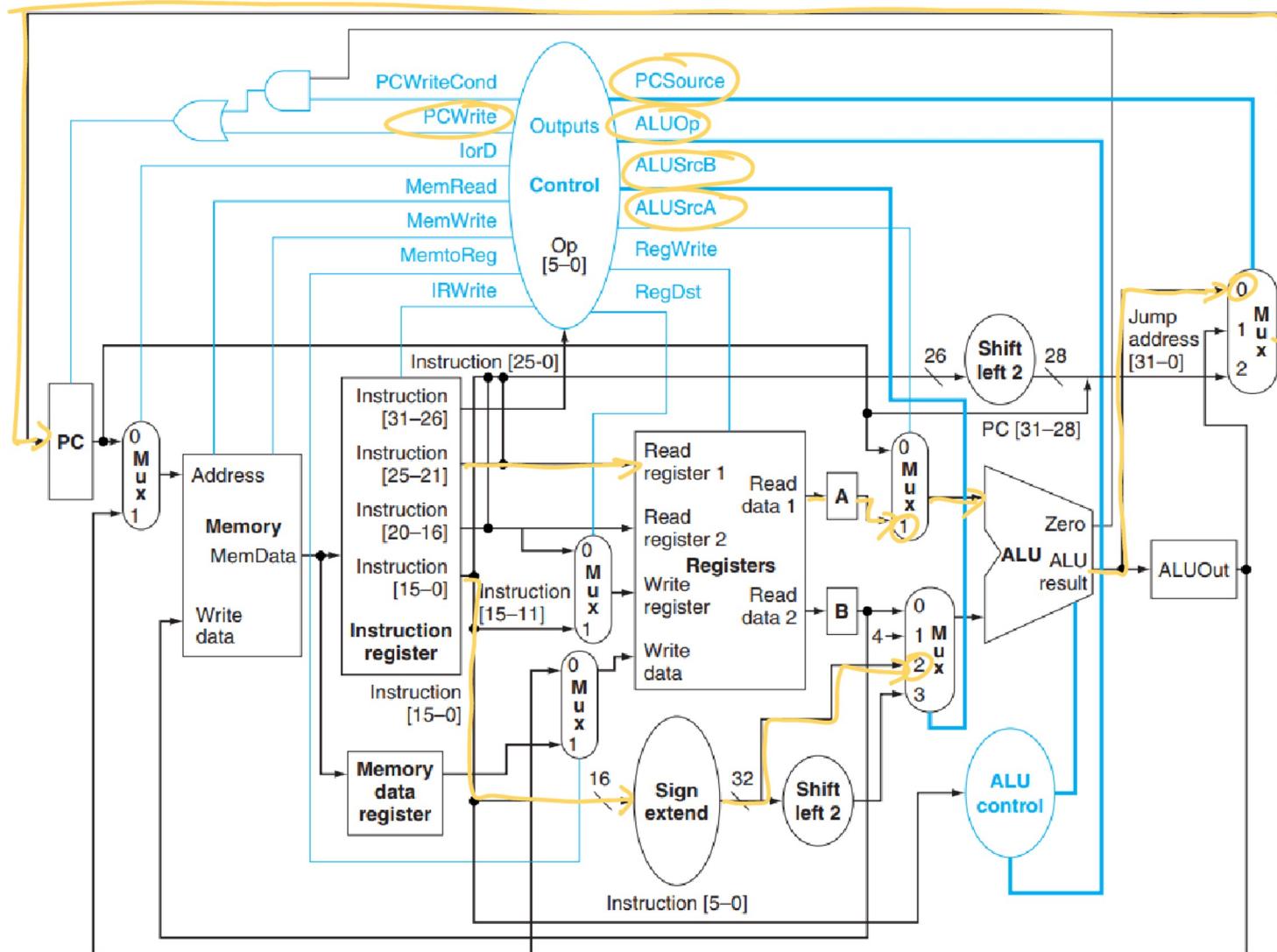
L'istruzione somma al valore contenuto nel registro \$rs il valore specificato dal campo immediate (sign-extended) e salta all'indirizzo ottenuto dalla somma

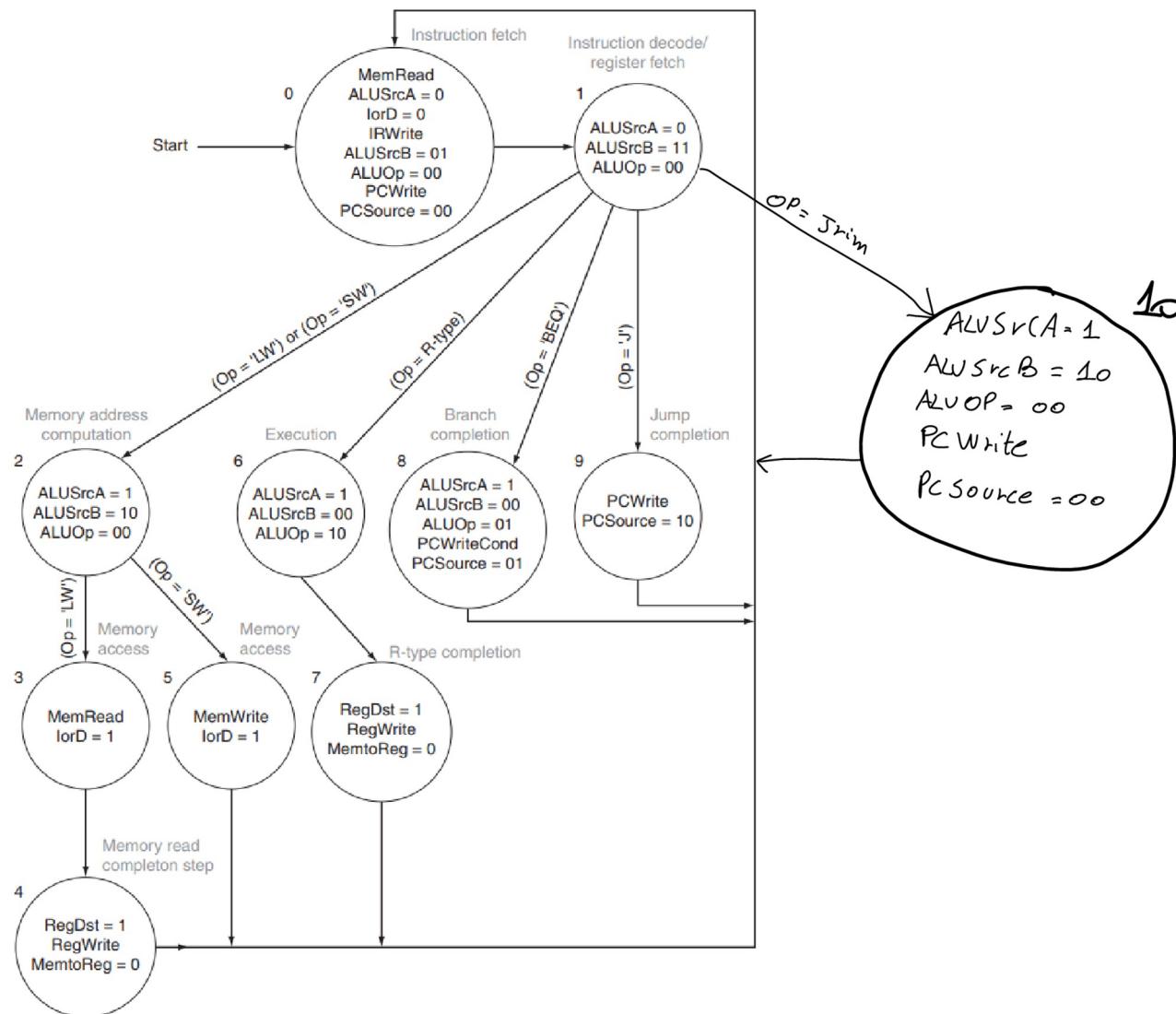
Jrim: modifiche all'hardware

- Possiamo effettuare entrambe le operazioni in un solo ciclo di clock
- Per poter effettuare la somma tra \$rs [bit 21-25] e il campo immediate [bit 0-15] è necessario che entrambi siano in input all'ALU. Osservando i multiplexer comandati dai segnali ALUSrcA e ALUSrcB notiamo che questi sono già selezionabili come input dell'ALU
- Per poter scrivere il risultato ottenuto nel PC è necessario che questo possa andare in input al registro PC. Notiamo che questo è già possibile utilizzando il multiplexer comandato dal segnale PCSource
- Non è quindi necessario effettuare modifiche hardware

Jrim: Segnali di controllo

- Per mandare in ingresso \$rs all'ALU selezioniamo "1" come segnale ALUSrcA
- Per mandare in ingresso il campo imm, con segno esteso per arrivare a 32 bit, selezioniamo "10" come segnale ALUSrcB
- Per effettuare la somma selezioniamo "00" come segnale ALUOp
- Per scrivere il valore in output dall'ALU nel registro PC selezioniamo "00" come segnale PCSource, e impostiamo PCWrite a "1"





Swap soluzione

Modificare lo schema del datapath multiciclo e la relativa FSM in modo da aggiungere al set di istruzioni del processore MIPS l'istruzione swap \$rs \$rt, che scambia il contenuto di due registri ed è definita come segue:

- 6 bit OPCODE [bit 26-31] (001001)
- 5 bit \$rs [bit 21-25]
- 5 bit \$rt [bit 20-16]
- 5 bit \$rd [bit 15-11], con \$rd = \$rs
- [bit 10-0] non utilizzati

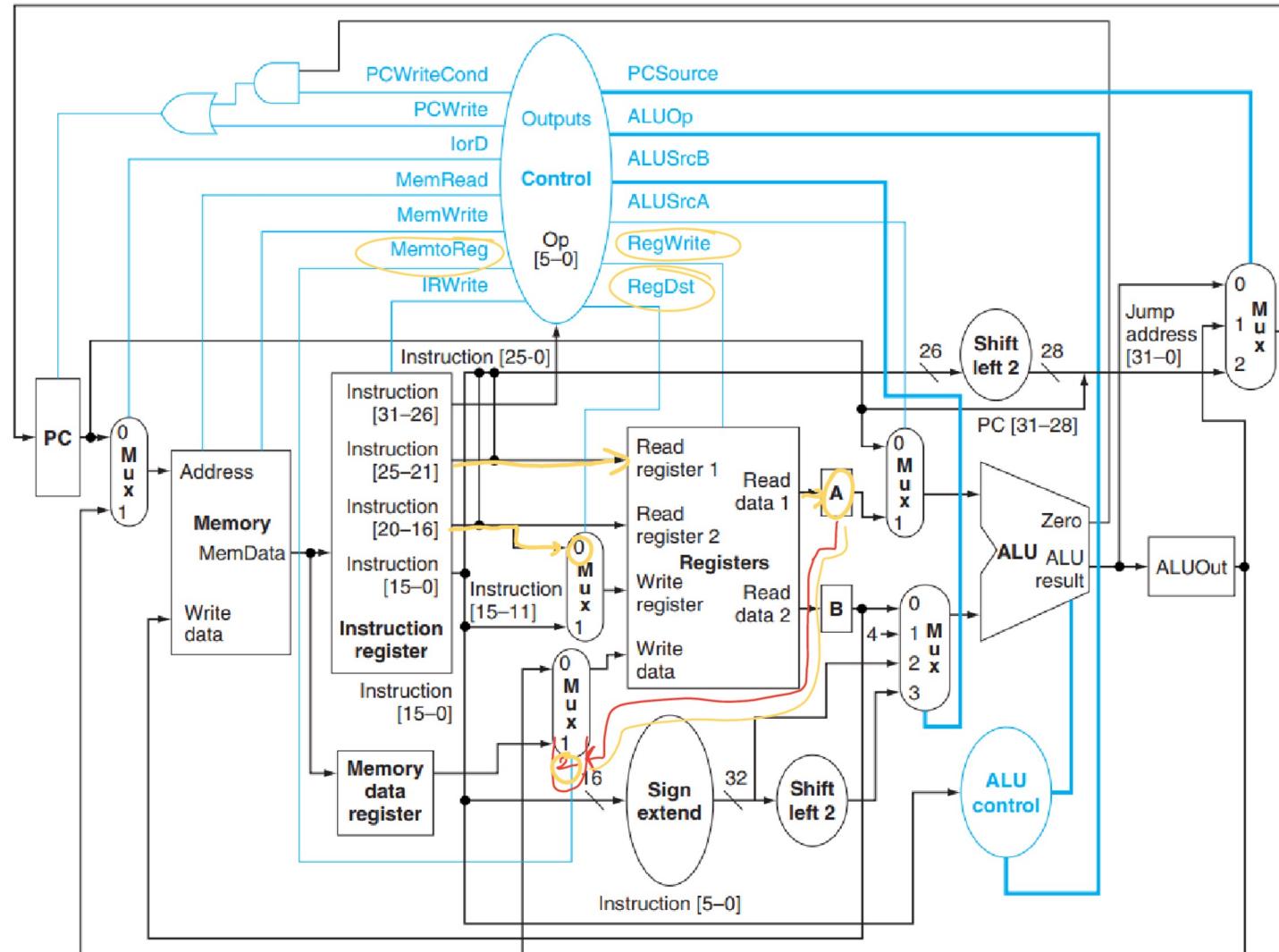
Scrive nel registro \$rt il contenuto di \$rs e nel registro \$rd (che è uguale a \$rs), il contenuto di \$rt

Swap: modifiche all'hardware

- Il valore del registro \$rs viene scritto nel registro A durante la fase di fetch. Dobbiamo quindi collegare il registro A al multiplexer collegato a “Write data” del register file, per poter scrivere il contenuto del registro \$rt
- Il valore del registro \$rt viene scritto nel registro B durante la fase di fetch. Dobbiamo quindi collegare il registro B al multiplexer collegato a “Write data” del register file, per poter scrivere il contenuto del registro in \$rd
- I campi \$rt e \$rd sono già collegati al multiplexer in input a “Write register”, non dobbiamo quindi modificarlo
- Per l'esecuzione dell'istruzione saranno necessarie due cicli

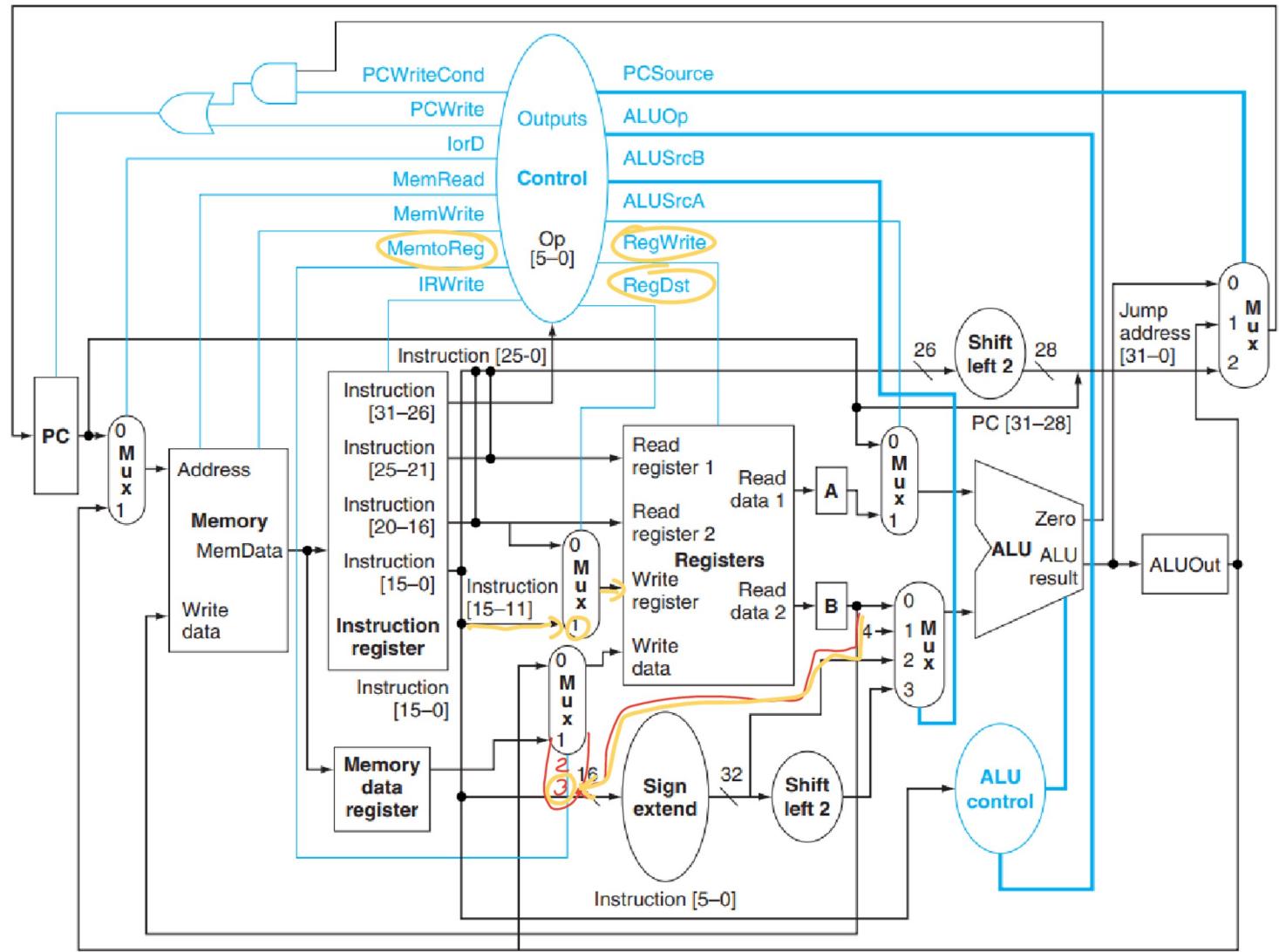
Swap: segnali di controllo, step 1

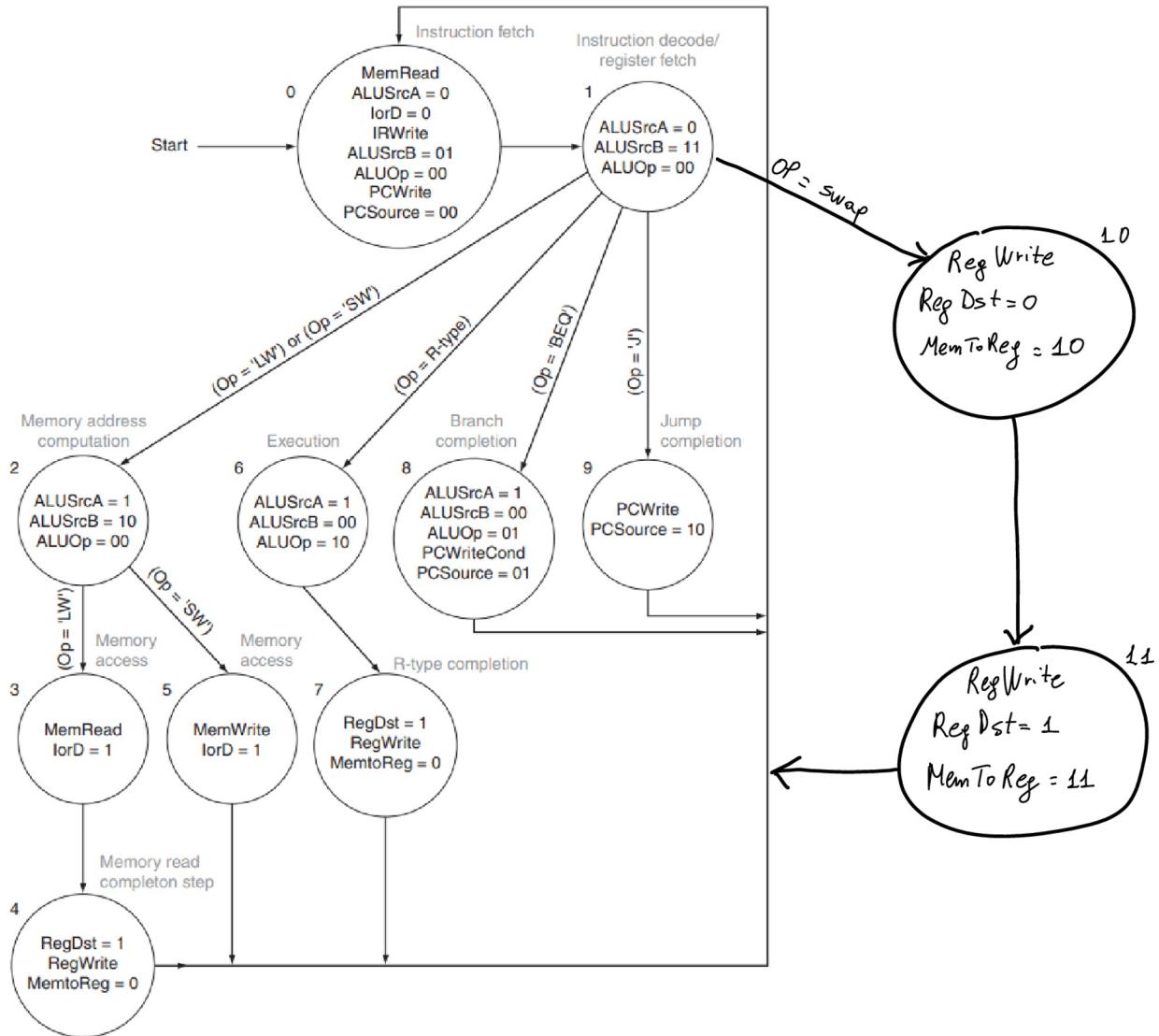
- Per avere come valore di WriteData il valore del registro A, ovvero il contenuto di \$rs, selezioniamo il nuovo ingresso al multiplexer comandato dal segnale MemToReg con valore 10
- Per scrivere nel registro \$rt, selezioniamo come valore del segnale RegDst 0
- Abilitiamo la scrittura sul register file, mettendo RegWrite = 1



Swap: segnali di controllo, step 2

- Al ciclo di clock successivo, nel registro B è ancora presente il contenuto di \$rt prima che venisse sovrascritto
- Per avere il valore del registro B come valore di WriteData, lo selezioniamo con il valore 11 del segnale MemToReg
- Per scrivere nel registro \$rd (che è stato appositamente imposto uguale a \$rs), selezioniamo con il valore 1 del segnale RegDst
- Abilitiamo la scrittura sul register file, mettendo RegWrite = 1





Add3 soluzione

Modificare lo schema del datapath multiciclo e la relativa FSM in modo da aggiungere al set di istruzioni del processore MIPS l'istruzione add3 \$rd \$rs \$rt \$rx, che somma il contenuto dei registri \$rs \$rt \$rx salvando il risultato in \$rd ed è definita come segue:

- 6 bit OPCODE [bit 26-31] (001001)
- 5 bit \$rs [bit 21-25]
- 5 bit \$rt [bit 20-16]
- 5 bit \$rd [bit 15-11]
- 5 bit \$rx [bit 4-0]

Cosa fare dopo la fase di fetch e decode?

- Scomponiamo la somma dei 3 elementi in due somme: prima sommiamo il contenuto di \$rs IR[21-25] e quello di \$rt IR[20-16]
- Sommiamo quindi al risultato ottenuto il contenuto di \$rx IR[4-0]
- Salviamo il risultato finale in \$rd IR[15-10]

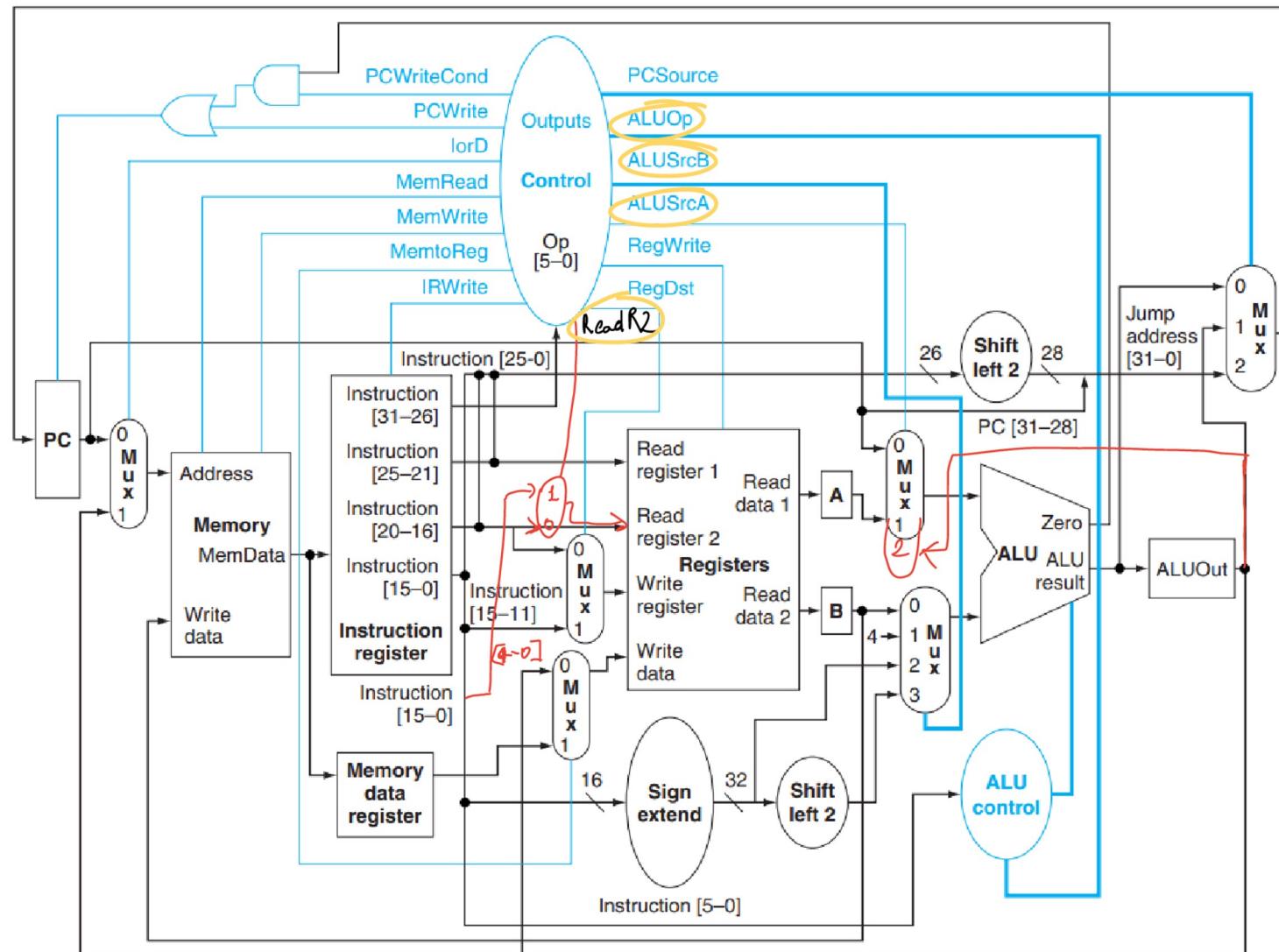
Add3: modifiche all'hardware

- Per effettuare la somma tra \$rs e \$rt non è necessaria alcuna modifica, in quanto è già possibile selezionarli come input all'ALU
- Per poter utilizzare il valore di ALUOut in una somma è necessario collegarlo ad un input dell'ALU. Aggiungiamo quindi ALUOut come ingresso al multiplexer comandato dal segnale ALUSrcA
- Per poter leggere il contenuto del registro \$rx è necessario metterlo in input al “Read register 2” del Register File. Aggiungiamo quindi un multiplexer all’input ReadRegister2, il cui segnale di controllo sarà 0 per selezionare i bit [20-16] dell’IR (come da implementazione standard) e sarà 1 per selezionare i bit [4-0]. In tutte le fasi in cui veniva usato il ReadRegister2 bisognerà impostare il segnale di controllo del multiplexer a 0, per non modificare il comportamento “standard” del datapath.
- Per salvare il contenuto di ALUOut in \$rd non è necessaria nessuna modifica hardware, in quanto è già possibile selezionare ALUOut come input di WriteData e \$rd come input di WriteRegister

Add3: Segnali di controllo, step 1

- Per il primo input dell'ALU impostiamo ALUSrcA = 01, utilizzando quindi il registro A, in cui era stato caricato nel ciclo di clock precedente il valore di \$rs
- Per il secondo input dell'ALU impostiamo ALUSrcB = 00, utilizzando quindi il registro B, in cui era stato caricato nel ciclo di clock precedente il valore di \$rt
- Impostiamo ALUOp = 00 per effettuare una somma
- Impostiamo il segnale del nuovo multiplexer ReadR2 = 1, per effettuare il fetch del registro \$rx

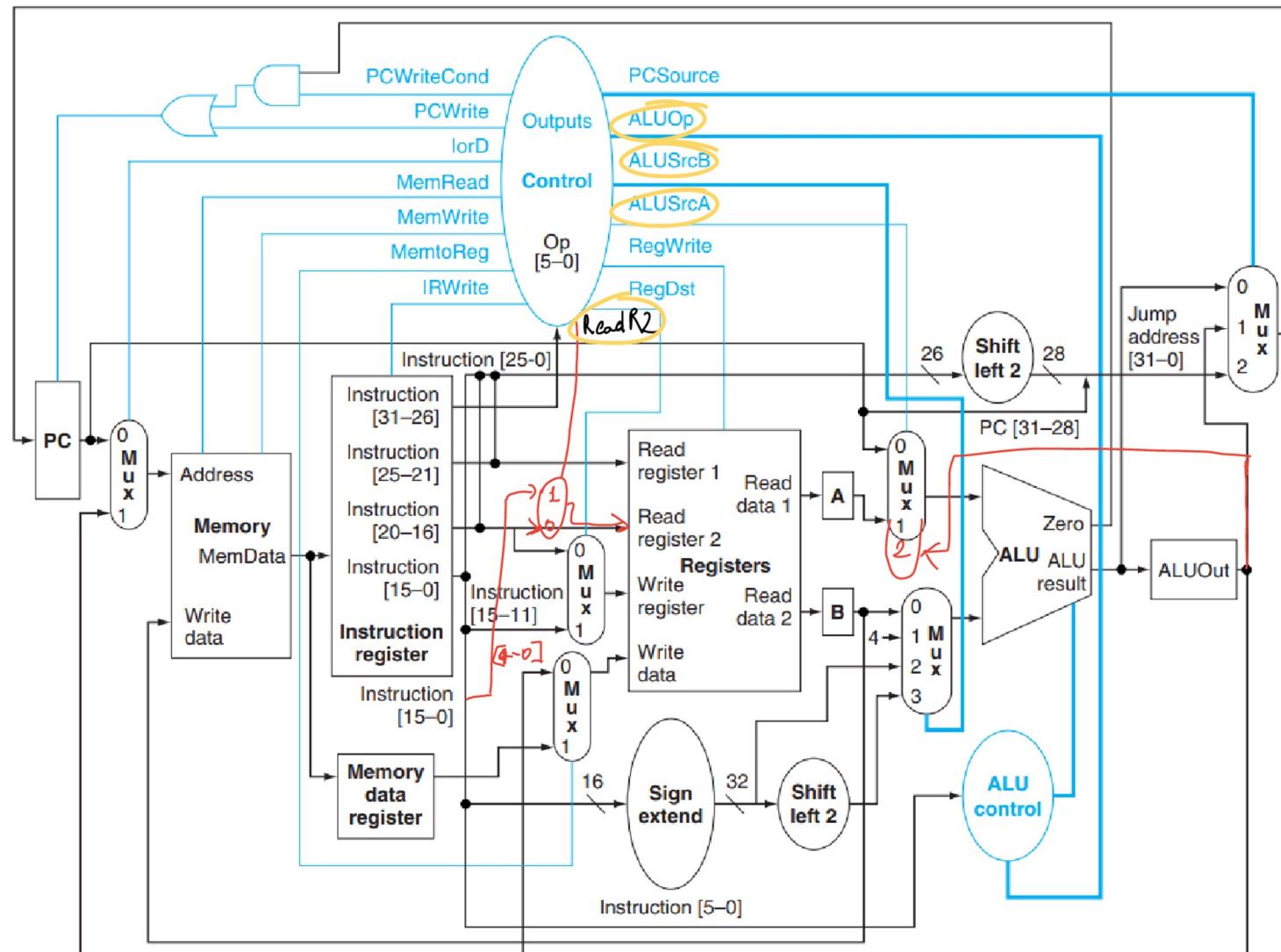
ALUSrcA = 01
 ALUSrcB = 00
 ALUOp = 00
 ReadR2 = 1



Add3: segnali di controllo, step 2

- Per il primo input dell'ALU impostiamo ALUSrcA = 10, utilizzando quindi il valore del registro ALUOut, ovvero la somma calcolata al ciclo di clock precedente
- Per il secondo input dell'ALU impostiamo ALUSrcB = 00, utilizzando quindi il registro B, in cui era stato caricato nel ciclo di clock precedente il valore di \$rx
- Impostiamo ALUOp = 00 per effettuare una somma

$\text{ALUSrcA} = 10$
 $\text{ALUSrcB} = 00$
 $\text{ALUOp} = 00$



Add3: segnali di controllo, step 3

- Infine, è necessario salvare il contenuto di ALUOut nel registro \$rd.
- Questa è una fase standard delle istruzioni R-type, possiamo quindi andare nello stato 7, ovvero impostiamo RegDst=1 per selezionare \$rd come registro destinazione, MemToReg=0 per selezionare ALUOut come valore da scrivere e RegWrite=1 per effettuare la scrittura sul register file

