6' Esercitazione

https://politecnicomilano.webex.com/meet/gianenrico.conti

Gian Enrico Conti

MIPS architecture

Outline

- **Formato Istruzioni**
  - Istruzioni per formato
- **Architettura a singolo ciclo**
  - Funzionamento
  - Segnali di controllo
- **Architettura pipelined**
  - Funzionamento
  - Esecuzione sequenziale vs pipelined
  - Conflitti all'interno della pipeline
    - Conflitti strutturali
    - Conflitti sui dati
    - Conflitti di controllo

Formato istruzioni

- **Esistono 3 tipi di formati istruzioni nel processore MIPS**
  - Istruzioni R
  - Istruzioni I
  - Istruzioni J

Nome	Campi						Commenti
Dimensione del campo	6 bit	5 bit	5 bit	5 bit	5 bit	6 bit	Tutte le istruzioni MIPS sono a 32 bit
Formato R	codop	rs	rt	rd	shamt	funz	Formato delle istruzioni aritmetiche
Formato I	codop	rs	rt	indirizzo / costante			Formato delle istruzioni di trasferimento dati di salto condizionato e immediate
Formato J	codop	indirizzo di destinazione					Formato delle istruzioni di salto incondizionato

- **I campi definiti sono**
  - codop — codice operativo
  - rs — primo registro sorgente
  - rt — secondo registro sorgente
  - rd — registro destinazione
  - shamt — shift amount
  - funz — codice funzione
  - costante — valore costante
  - indirizzo — indirizzo di destinazione

Istruzioni per formato

Istruzioni MIPS	Nome	Formato	Pseudoistruzioni MIPS	Nome	Formato
add	`add`	R	move	`move`	R
subtract	`sub`	R	multiply	`mult`	R
add immediate	`addi`	I	multiply immediate	`multi`	I
load word	`lw`	I	load immediate	`li`	I
store word	`sw`	I	branch less than	`blt`	I
load half	`lh`	I	branch less than or equal	`ble`	I
load half unsigned	`lhu`	I	branch greater than	`bgt`	I
store half	`sh`	I	branch greater than or equal	`bge`	I
load byte	`lb`	I			
load byte unsigned	`lbu`	I			
store byte	`sb`	I			
load linked	`ll`	I			
store conditional	`Sc`	I			
load upper immediate	`lui`	I			
and	`and`	R			
or	`or`	R			
nor	`nor`	R			
and immediate	`andi`	I			
or immediate	`ori`	I			
shift left logical	`sll`	R			
shift right logical	`srl`	R			
branch equal	`beq`	I			
branch not equal	`bne`	I			
set less than	`slt`	R			
set less than immediate	`slti`	I			
set less than immediate unsigned	`sltiu`	I			
jump	`j`	J			
jump register	`jr`	R			
jump and link	`jal`	J			

MIPS Reference Sheet

(https://uweb.engr.arizona.edu/~ece369/Resources/spim/MIPSReference.pdf)

Instruction Encodings

Register	000000ss ssstttt dddddaaa aafffff
Immediate	oooooss ssstttt iiiiiiii iiiiiiii
Jump	oooooii iiiiiiii iiiiiiii iiiiiiii

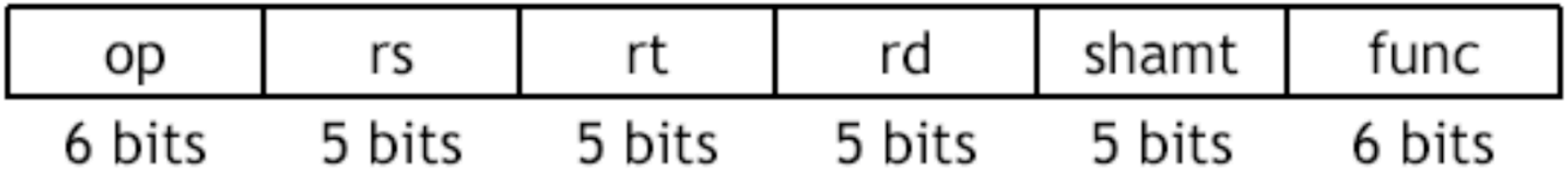
Opcode Table

Instruction	Opcode/Function	Syntax
add	100000	ArithLog
addu	100001	ArithLog
addi	001000	ArithLogI
addiu	001001	ArithLogI
and	100100	ArithLog
andi	001100	ArithLogI
div	011010	DivMult
divu	011011	DivMult
mult	011000	DivMult
multu	011001	DivMult
nor	100111	ArithLog
or	100101	ArithLog
ori	001101	ArithLogI
sll	000000	Shift
sllv	000100	ShiftV
sra	000011	Shift
srav	000111	ShiftV
srl	000010	Shift
srlv	000110	ShiftV
sub	100010	ArithLog
subu	100011	ArithLog
xor	100110	ArithLog
xori	001110	ArithLogI
lhi	011001	LoadI
llo	011000	LoadI

Instruction	Opcode/Function	Syntax
slt	101010	ArithLog
sltu	101001	ArithLog
slti	001010	ArithLogI
sltiu	001001	ArithLogI
beq	000100	Branch
bgtz	000111	BranchZ
blez	000110	BranchZ
bne	000101	Branch
j	000010	Jump
jal	000011	Jump
jalr	001001	JumpR
jr	001000	JumpR
lb	100000	LoadStore
lbu	100100	LoadStore
lh	100001	LoadStore
lhu	100101	LoadStore
lw	100011	LoadStore
sb	101000	LoadStore
sh	101001	LoadStore
sw	101011	LoadStore
mfhi	010000	MoveFrom
mflo	010010	MoveFrom
mthi	010001	MoveTo
mtlo	010011	MoveTo
trap	011010	Trap

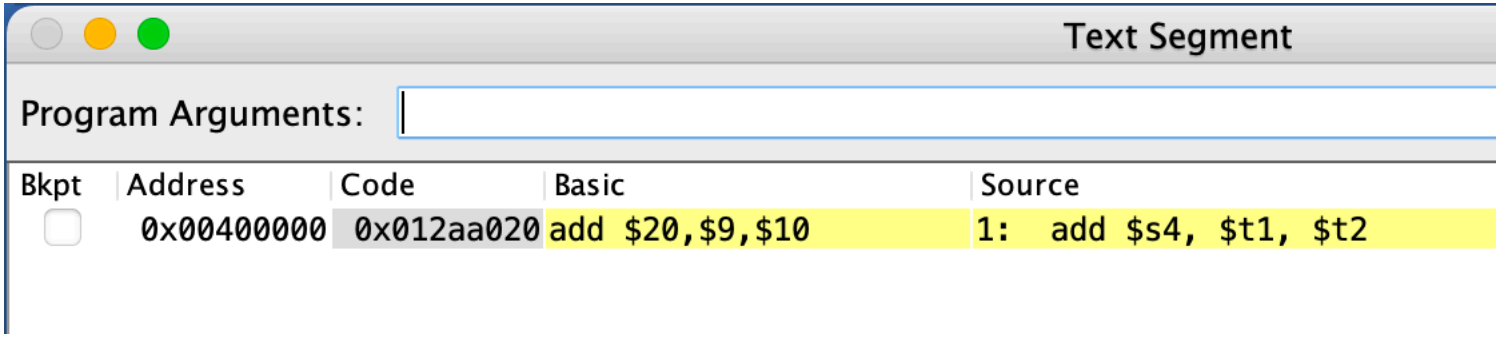
Some samples..

EXAMPLE: R-type



add $s4, $t1, $t2

In Mars:



Rd: 20

Rs:9

Rt: 10

Op = 00000

E dalla tabella prec.:

Instruction	Opcode/Function	Syntax
add	100000	ArithLog

EXAMPLE: R-type

add $s4, $t1, $t2

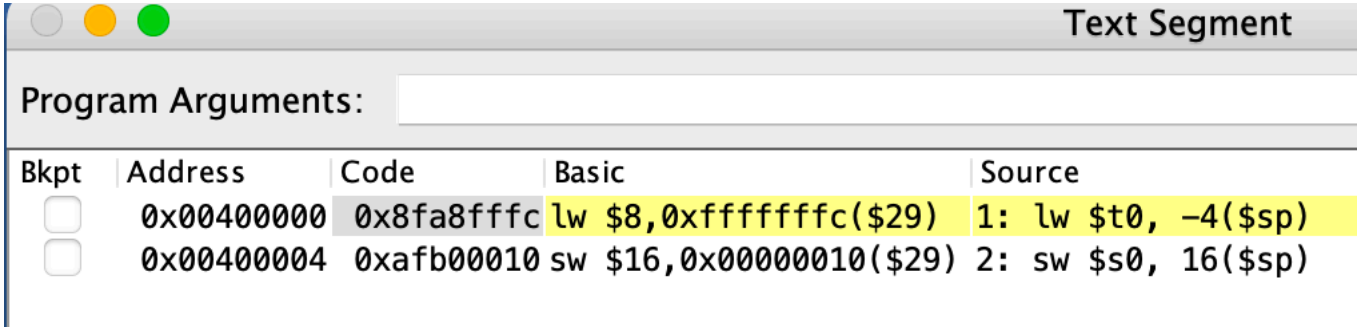

```
add $s4, $t1, $t2
```

```
Rd: 20          10100
Rs:  9          01001
Rt: 10          01010
```

EXAMPLE: lw, sw

```
lw $t0, -4(sp)
sw $s0, 16(sp)
```

In Mars:



Rd: --
Rs: (SP)$29 = 29
Rt: $8
    $10

lw	100011
sb	101000
sh	101001
sw	101011

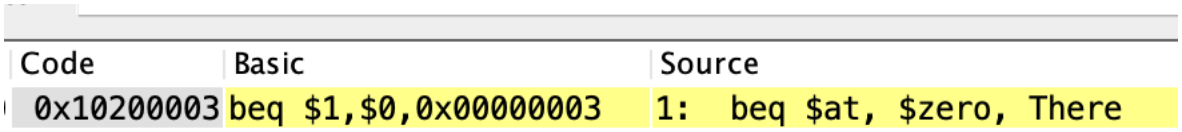
EXAMPLE: salto

beq $at, $zero, There

addi $at, $zero, 10 #dont care
addi $at, $zero, 20 #dont care

There: sub $at, $at,30 #dont care

In Mars:



Code	Basic	Source
0x10200003	beq $1,$0,0x00000003	1: beq $at, $zero, There

Rd: 0
Rs: 1

Si noti salto di 3 (3 istruz. = 12 celle)

EXAMPLE: lw, sw

```
beq $at, $zero, There
li $t1,10
li $t2,20
add $t1, $t1, $t2
There: sub $at, $at,30
```

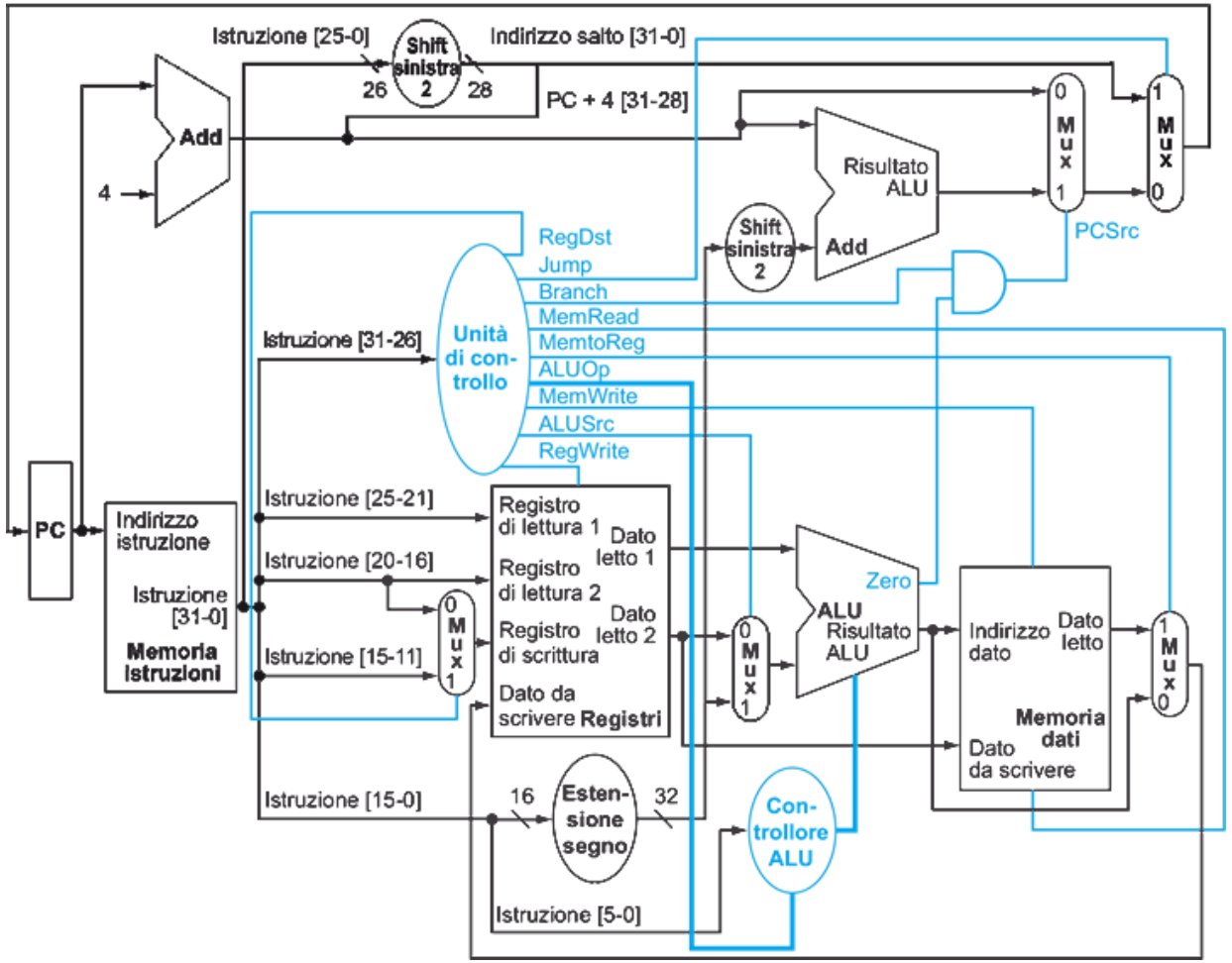
sltiu	001001	ArithLogI
beq	000100	Branch
bgtz	000111	BranchZ
blez	000110	BranchZ

Rd: 0

Rs: 1

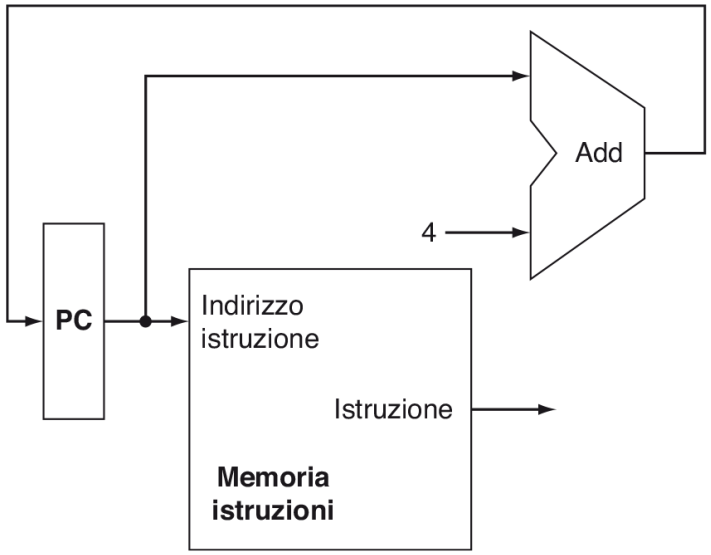
Architettura a singolo ciclo

- **Schema semplificato dell'architettura MIPS a singolo ciclo**
  - Supporto della maggior parte delle funzioni



Funzionamento (I)

- **Il funzionamento dell'architettura MIPS si divide in 5 parti principali:**
  - **FETCH**
    - Caricamento dell'istruzione nel PC dalla memoria istruzioni
    - Incremento del PC



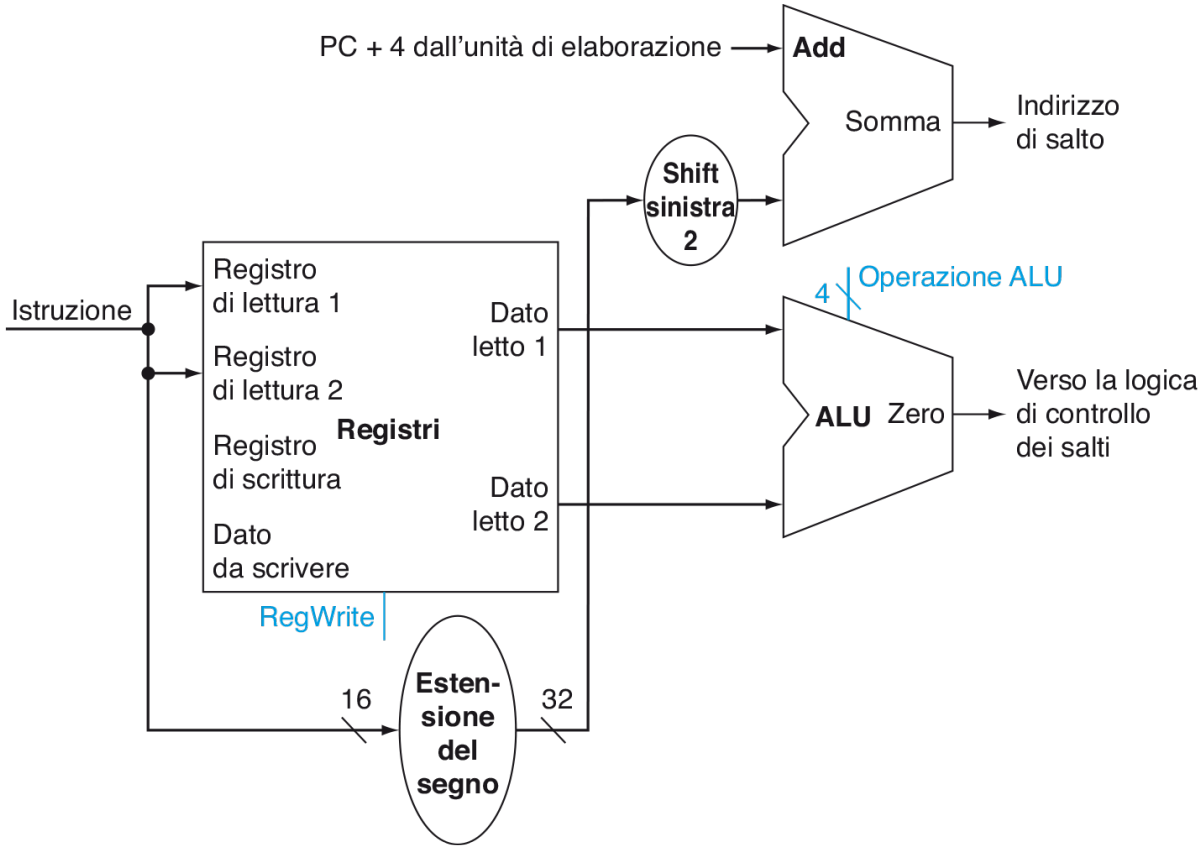
Funzionamento (II)

- **DECODE**

  - Lettura dei dati da Register File
  - Preparazione di eventuali altri operandi per la ALU (Ex. Immediati)
  - Preparazione dei segnali di controllo per l'esecuzione dell'istruzione

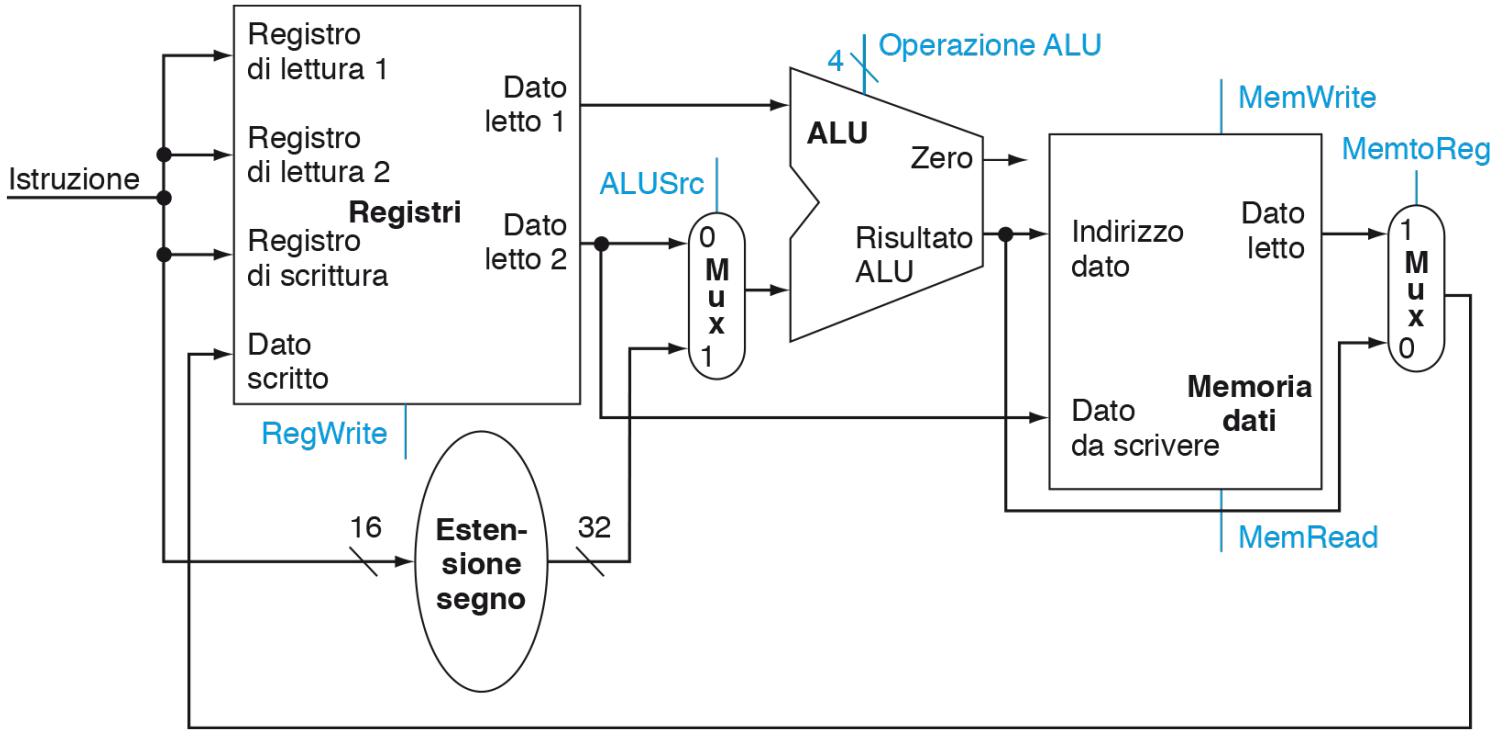
Funzionamento (III)

- **EXECUTE**
  - Calcolo del risultato dell'operazione
  - Calcolo indirizzo di salto



Funzionamento (IV)

- **MEMORY**
  - Caricamento di eventuali dati da memoria
- **WRITE BACK**
  - Scrittura nel Register File



Architettura a singolo ciclo

- Segnali di controllo
- Controllore ALU

Architettura a singolo ciclo – Segnali di controllo

- **I segnali di controllo sono i punti di controllo dell'architettura MIPS**
  - L'unità di controllo genera questi segnali in base all'istruzione da eseguire
  - Il controllore ALU è un sottosistema dell'unità di controllo

Nome del segnale	Effetto quando non asserito	Effetto quando asserito
RegDst	Il numero del registro di scrittura proviene dal campo rt (bit 20-16)	Il numero del registro di scrittura proviene dal campo rd (bit 15-11)
RegWrite	Nulla	Il dato viene scritto nel register file nel registro individuato dal numero del registro di scrittura
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita del register file (Dato letto 2)	Il secondo operando della ALU proviene dall'estensione del segno dei 16 bit meno significativi dell'istruzione
PCSrc	Nel PC viene scritta l'uscita del sommatore che calcola il valore di PC + 4	Nel PC viene scritta l'uscita del sommatore che calcola l'indirizzo di salto
MemRead	Nulla	Il dato della memoria nella posizione puntata dall'indirizzo viene inviato in uscita sulla linea «dato letto»
MemWrite	Nulla	Il contenuto della memoria nella posizione puntata dall'indirizzo viene sostituito con il dato presente sulla linea «dato scritto»
MemtoReg	Il dato inviato al register file per la scrittura proviene dalla ALU	Il dato inviato al register file per la scrittura proviene dalla Memoria Dati

Architettura a singolo ciclo – Segnali di controllo (II)

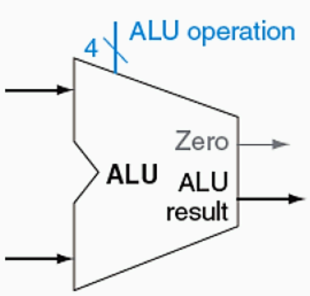
- **Altra vista:**

Operation	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg
add	1	1	0	010	0	0	0
sub	1	1	0	110	0	0	0
and	1	1	0	000	0	0	0
or	1	1	0	001	0	0	0
slt	1	1	0	111	0	0	0
lw	0	1	1	010	0	1	1
sw	X	0	1	010	1	0	X
beq	X	0	0	110	0	0	X

Architettura a singolo ciclo – Controllore ALU

- **Il controllore ALU decide come deve operare la ALU in base all'istruzione da eseguire**

Linee di controllo della ALU	Operazione
0000	AND
0001	OR
0010	somma
0110	sottrazione
0111	set less than
1100	NOR



- **Il segnale ALUOp, insieme al campo funzione determina l'operazione da eseguire**
  - LW/SW → ALUOp = 00
  - BEQ → ALUOp = 01
  - Istruzione R → ALUOp = 10

Calcolo datapath

- **Nell' ipotesi single cicle, calcoliamo tempo totale della propagazione dei dati x la istruzione:**

  lw $t0, 12($sp)

  Si noti:

- **Fetch...**
- **Accesso al register file ($SP)**
- **Somma/Sottrazione**
- **Accesso alla memoria**
- **Accesso al register file ($SP)**

Calcolo datapath

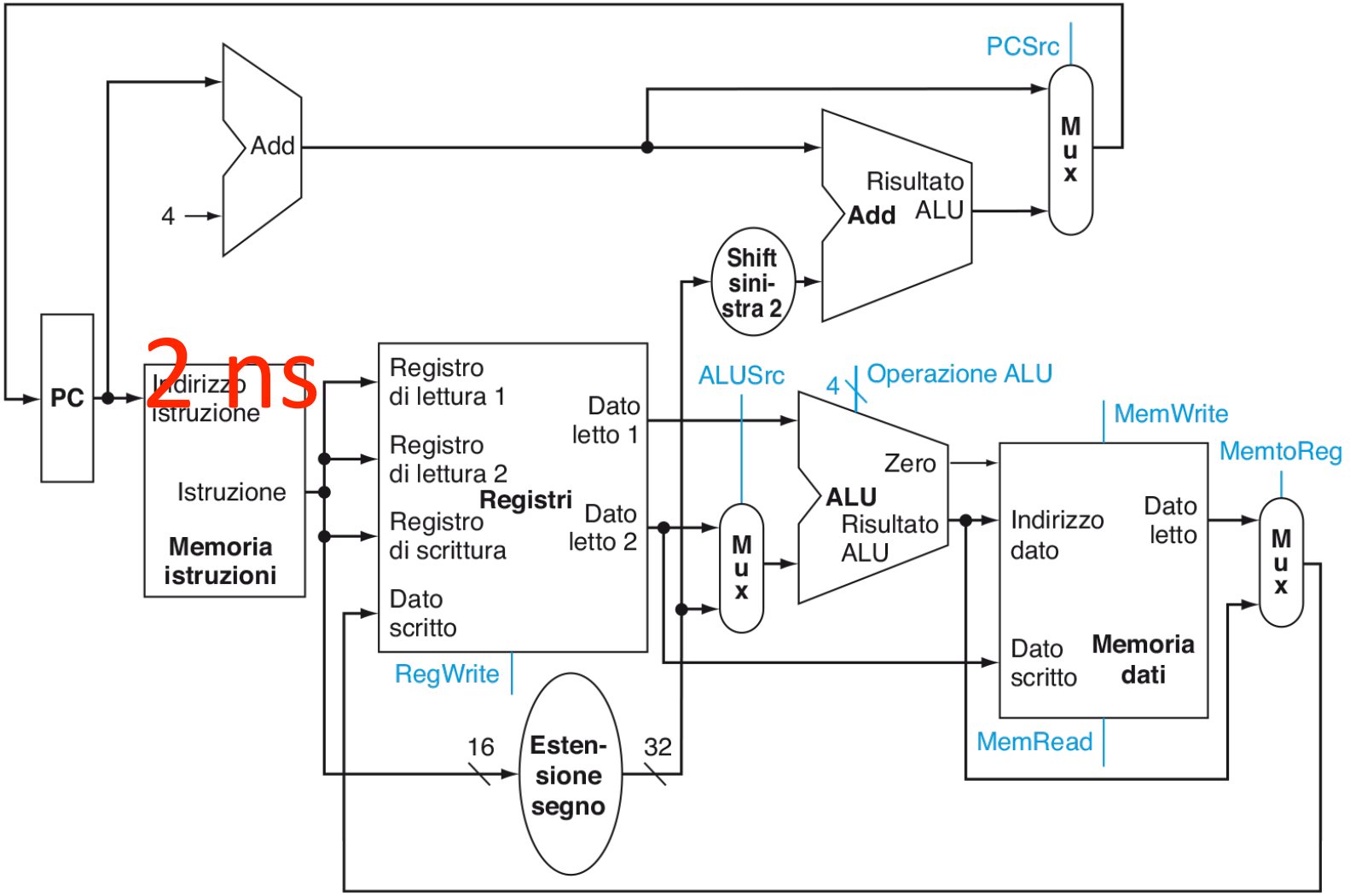
lw $t0, 12($sp)

Siano dati:

- reading the **instruction** memory 2ns
- reading the **data** memory 2ns
- reading **registers** 1ns
- **ALU** timings 2ns
- writing in **registers** 1ns

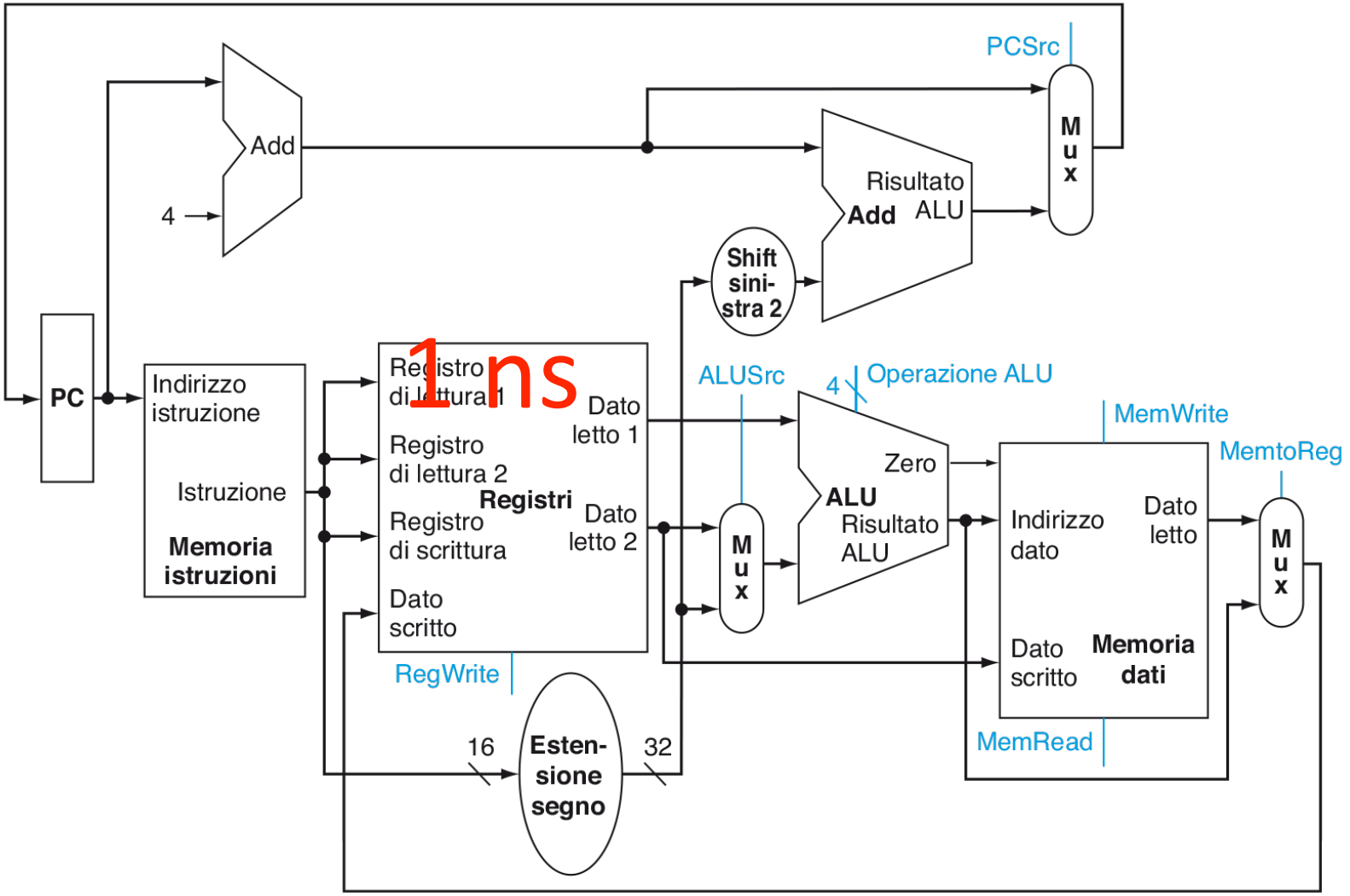
Calcolo datapath: lw $t0, 12($sp)

reading the **instruction** memory (Fetch)



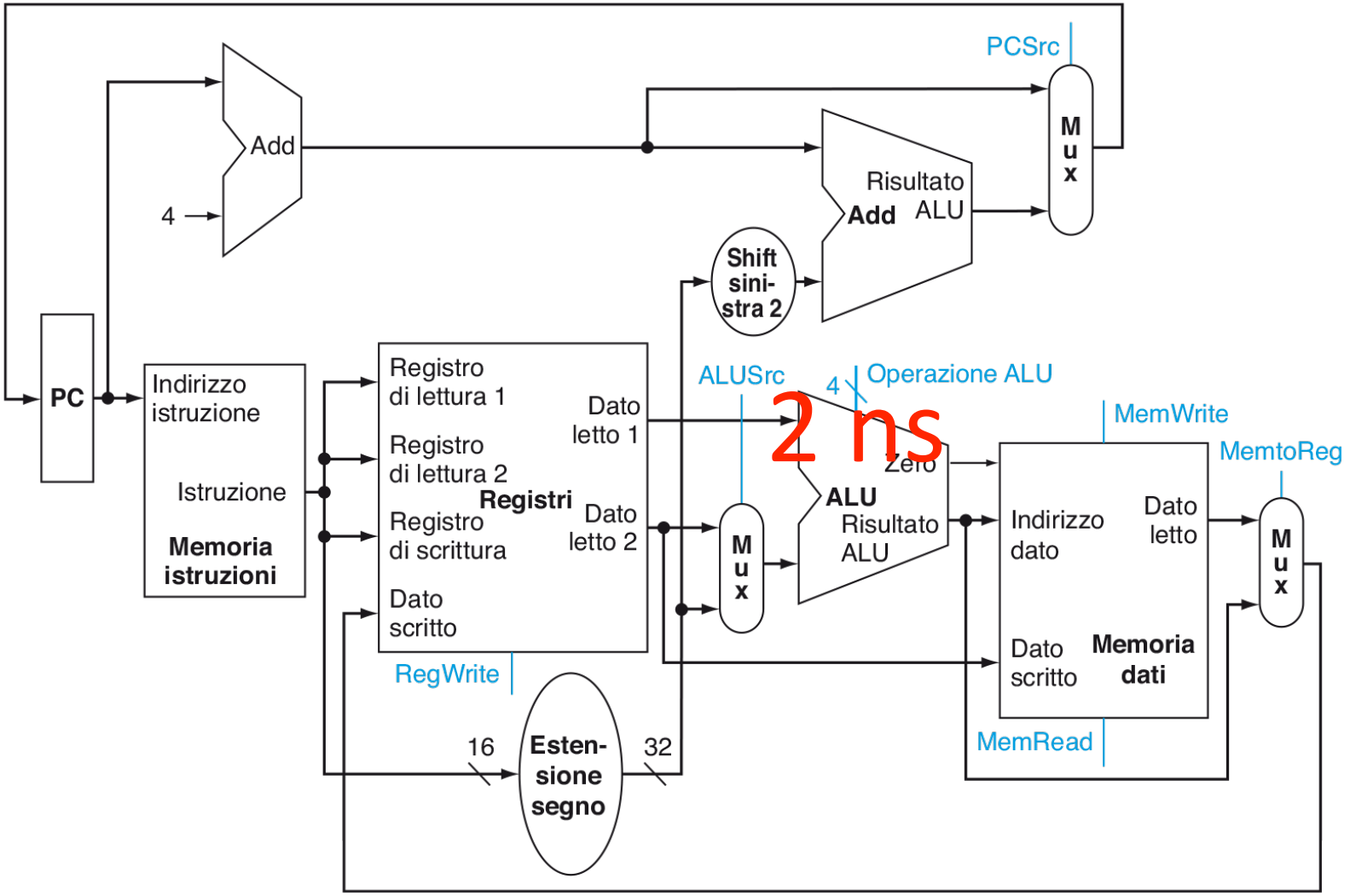
Calcolo datapath: lw $t0, 12($sp)

reading $SP



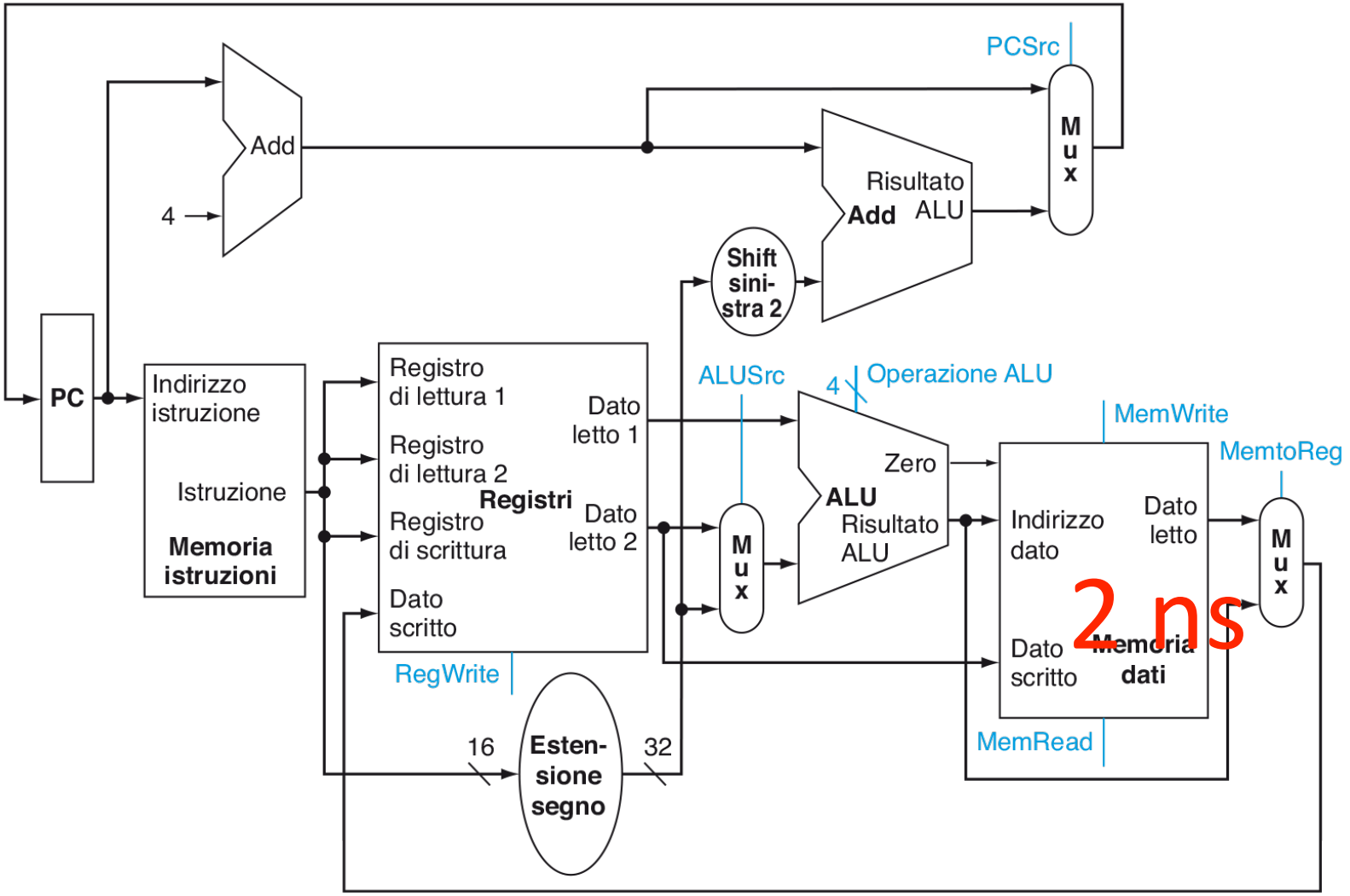
Calcolo datapath: lw $t0, 12($sp)

Add 12 to SP:



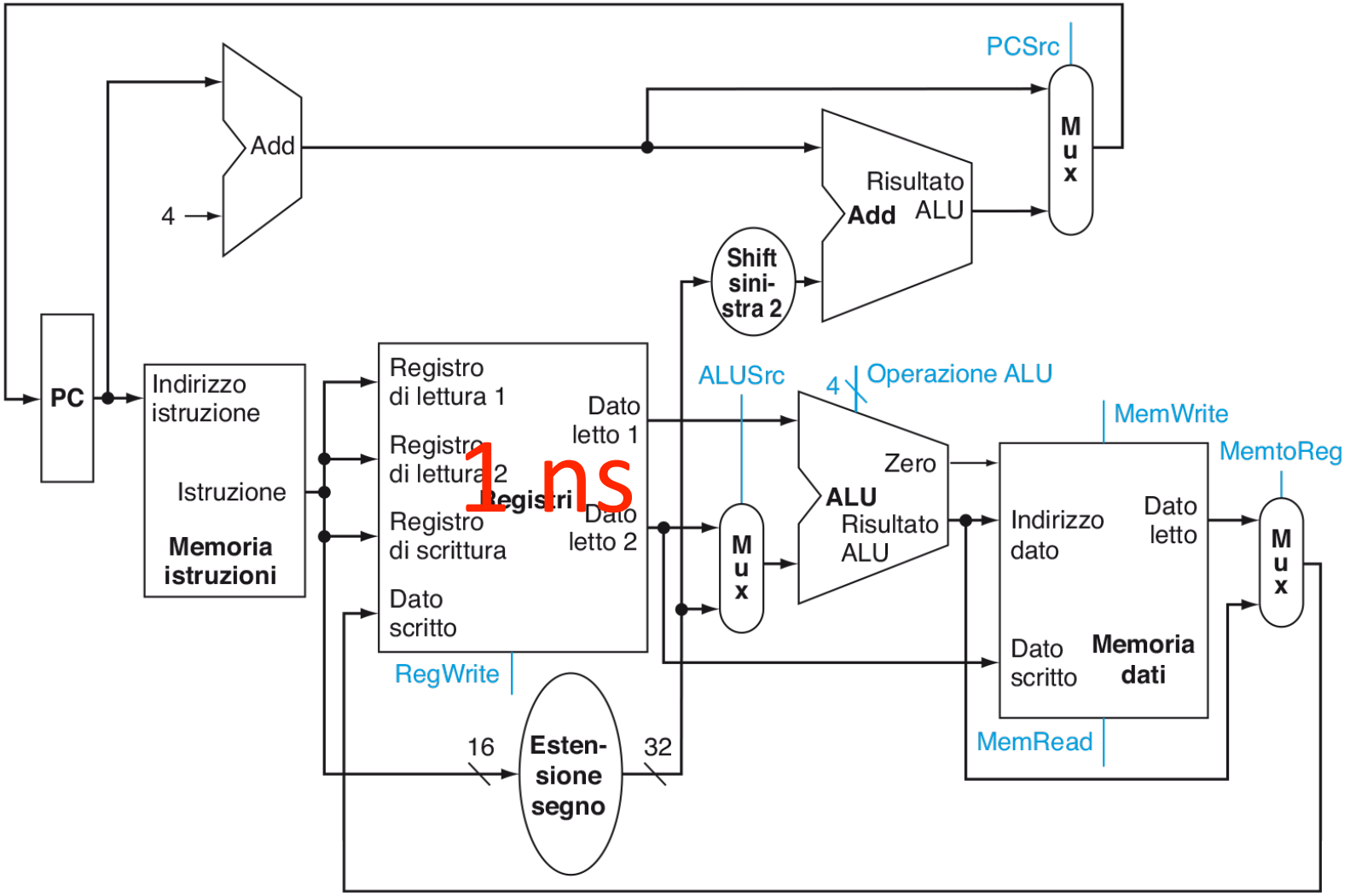
Calcolo datapath: lw $t0, 12($sp)

reading the **data** from memory



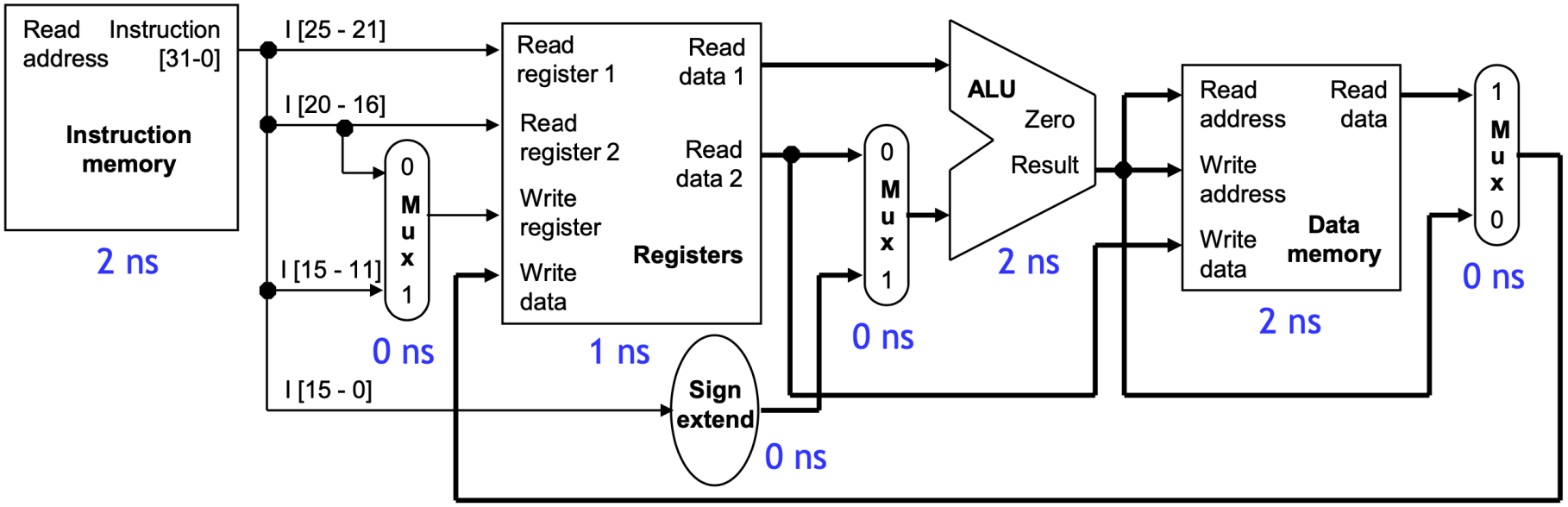
Calcolo datapath: lw $t0, 12($sp)

Writing in register ($t0)



Calcolo datapath: lw $t0, 12($sp)

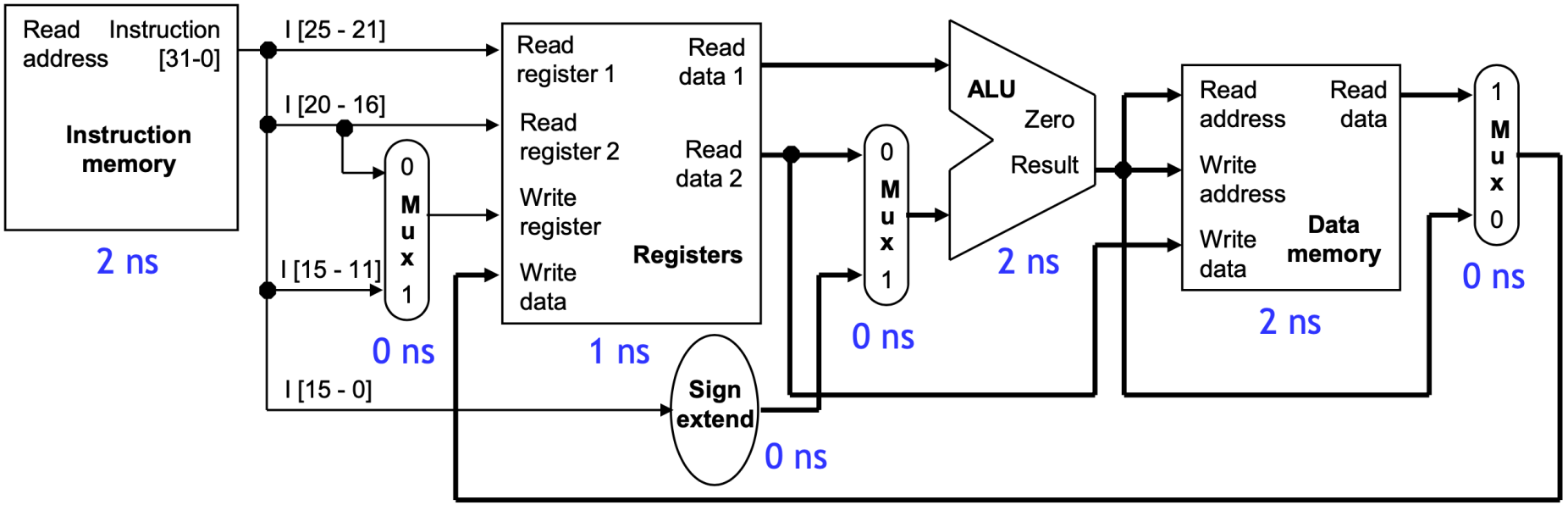
Tot 8 ns



2 1+1 2 2

Si notino i commenti x elem in logica combinatoria (mux)

Calcolo datapath: add $t0, $t1, $t2



2 1+1 2 (no memory access)

read the instruction memory (fetch) = 2

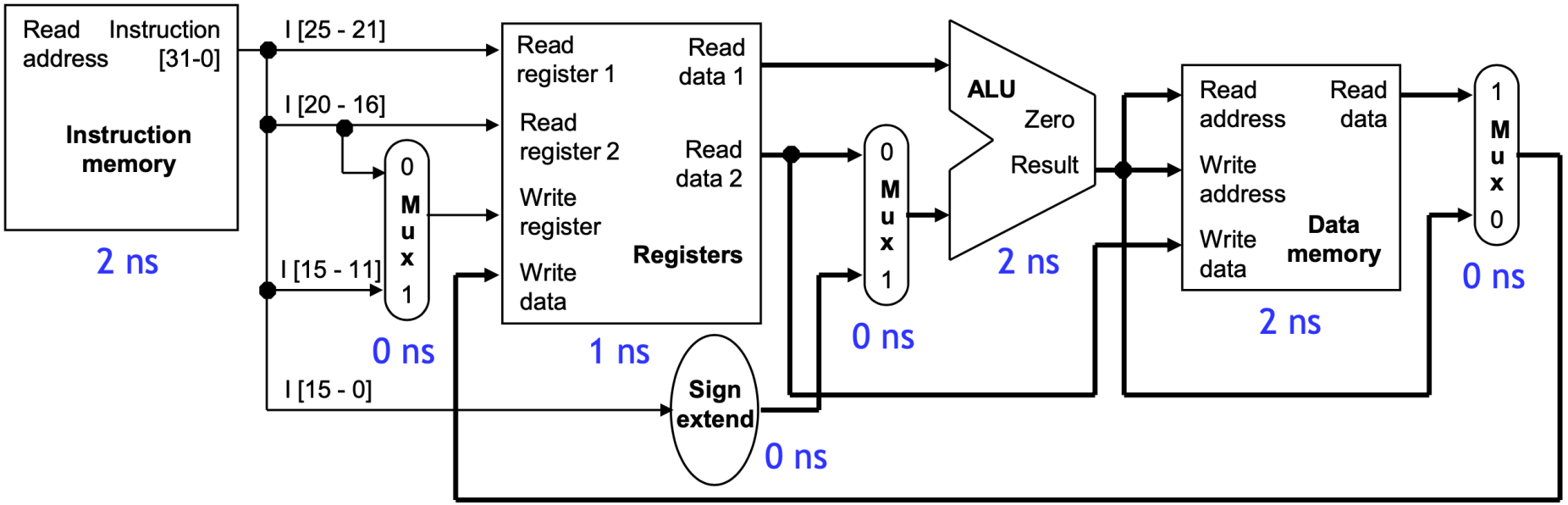
read registers $t1 and $t2 = 1 (don't sum!)

add $t1 + $t2 = 2

save $t0 in register file = 1

Calcolo datapath: add $t0, $t1, $t2

Tot 6 ns



2 1+1 2 (no memory access)

read the instruction memory (fetch) = 2

read registers $t1 and $t2 = 1 (don't sum!)

add $t1 + $t2 = 2

save $t0 in refuser file = 1

Esercizio

- **Si suppone che i blocchi logici richiesti per implementare l'unità di elaborazione di un processore abbiano le latenze riportate nella tabella seguente**

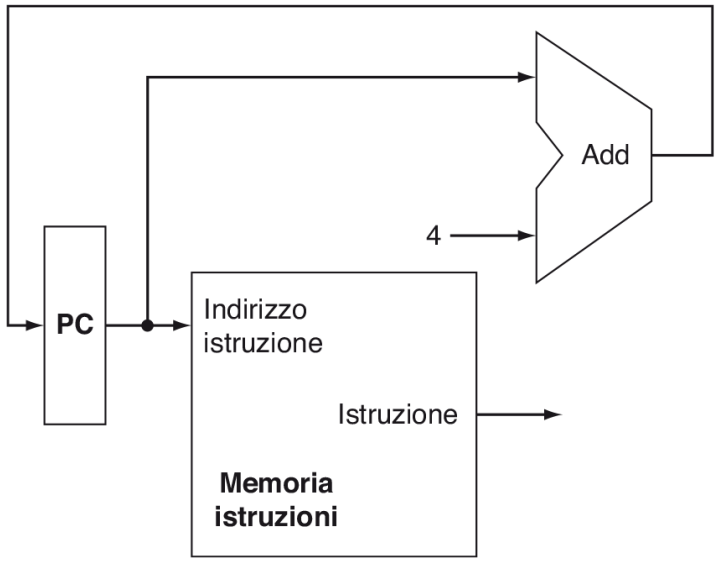
Mem-I	Add	Mux	ALU	Registri	Mem-D	Estensione segno	Shift Sx 2
200 ps	70 ps	20 ps	90 ps	90 ps	250 ps	15 ps	10 ps

1. Se richiedessimo al processore solo di prelevare le istruzioni una dopo l'altra, quale sarebbe il periodo del clock?

2. Si consideri un'unità di elaborazione completa che debba eseguire solamente istruzioni di un unico tipo: salti incondizionati relativi al PC. Quale sarebbe il cammino critico di questa unità di elaborazione?

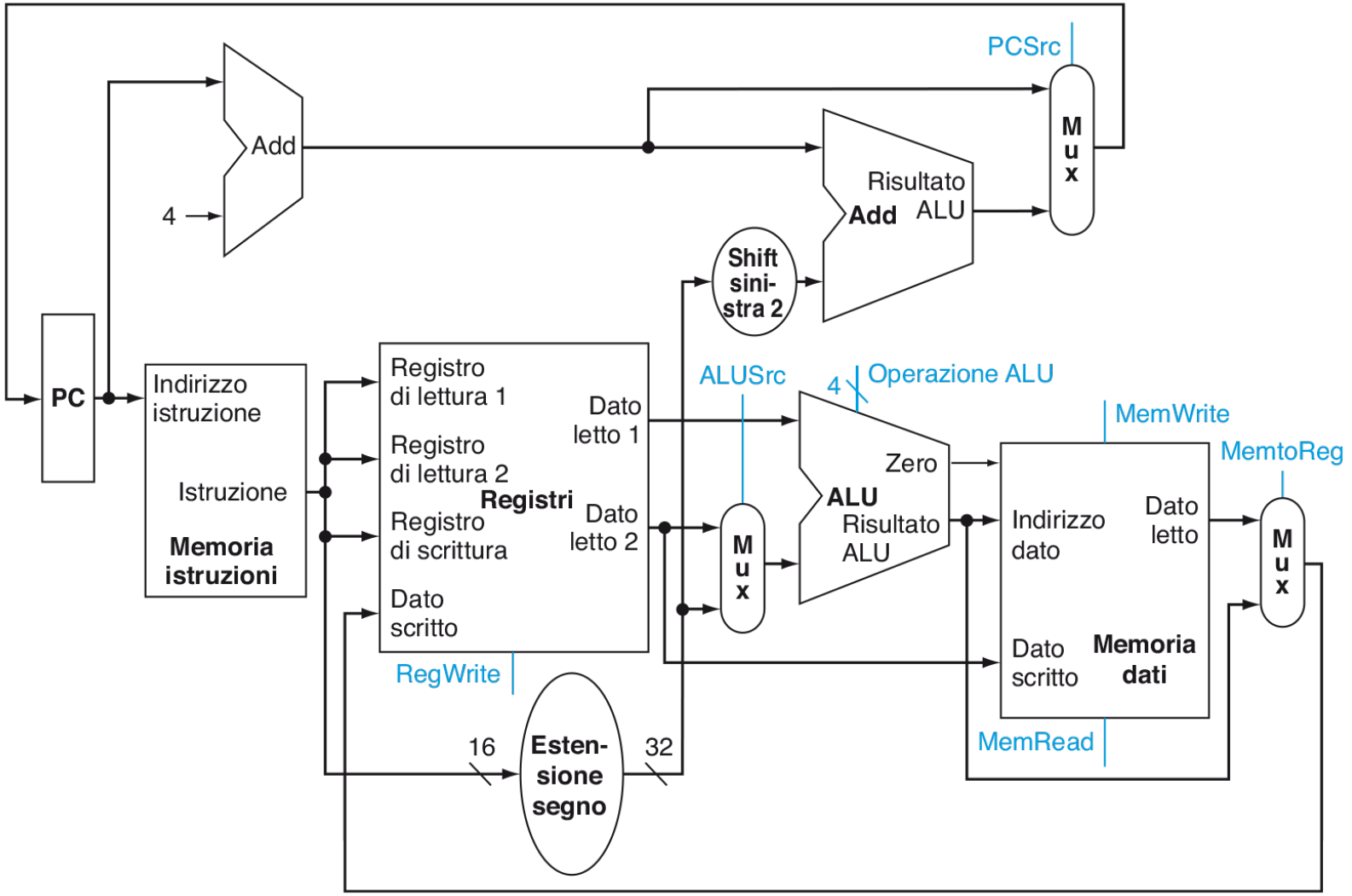
Esercizio

- **Riferimento per punto 1**



Esercizio

- **Riferimento per punto 2-3**



Esercizio

- **I restanti tre problemi di questo esercizio riguardano l'elemento del cammino di elaborazione «Shift sx 2».**

1. Quali tipi di istruzioni richiedono questa risorsa?

2. Per quali tipi di istruzioni (se ce ne sono) questa risorsa si trova sul loro cammino critico?

3. Supponendo che si vogliano supportare solamente le istruzioni add e branch incondizionato, discutere come la variazione della latenza di questa risorsa influenzi il periodo del clock del processore. Si supponga che la latenza delle altre risorse non cambi.

Esercizio 2

- **Per i problemi di questo esercizio si supponga che la percentuale delle singole istruzioni sia la seguente:**

add	addi	not	beq	lw	sw
20 %	20 %	0 %	25 %	25 %	10 %

1. In quale percentuale del numero totale di cicli di clock viene utilizzata la memoria dati?

2. In quale percentuale del numero totale di cicli di clock viene richiesto il circuito di estensione del segno? Che cosa fa questo circuito durante l'esecuzione delle istruzioni che non utilizzano l'estensione del segno?

How to...

- https://www.eg.bucknell.edu/~csci320/mips_web/
-