

PROBLEMI CON PRESTAZIONI

• # istr. al secondo = $\frac{\text{freq}}{\text{CPI}}$

Si considerino tre diversi processori P1, P2 e P3 che eseguono lo stesso insieme di istruzioni.

- P1 ha una frequenza di clock di 3 GHz e un CPI (Cicli Per Istruzione) di 1,5.
- P2 ha una frequenza di clock di 2,5 GHz e un CPI di 1,0.
- P3 ha una frequenza di clock di 4,0 GHz e un CPI di 2,2.

- Quale processore ha le prestazioni migliori espresse in numero di istruzioni al secondo?

$$\frac{3}{\frac{3}{2}} = 3 \cdot \frac{2}{3} = \frac{6}{3} = 2 \quad \text{P1}$$

$$\frac{2.5}{1} = 2.5 \quad \text{P2} \checkmark$$

$$\frac{4.0}{2.2} = 2.2 \quad \text{P3}$$

• # di cicli di clock più alto dato tempo

$$V_s = \frac{\# \text{ clock}}{\text{freq}} \Rightarrow \underline{V_s \cdot \text{freq} = \# \text{ clock}}$$

- Determinare il processore (scelto fra i 3 riportati) che necessita del numero di cicli di clock più alto, supponendo che tutti eseguano un programma in 10 secondi.

$$P_1 \Rightarrow 10 \cdot 3 = 30 \quad P_2 = 10 \cdot 2.5 = 25$$

$$P_3 \Rightarrow 10 \cdot 4 = 40$$

• freq. di clock

$$V_s = \frac{\# \text{ clock}}{\text{freq}} \Rightarrow \text{freq} = \frac{\# \text{ clock}}{V_s}$$

- Si vorrebbe ridurre il tempo di esecuzione del 30% (da 10 a 7 secondi), ma per raggiungere questo obiettivo il CPI aumenterebbe del 20%. Quale sarebbe la frequenza di clock che consentirebbe questa riduzione del tempo di esecuzione per il processore P1? [Risposta espressa in GHz]

Al momento vale che per P1

$$10s = \frac{x \cdot 1.s}{3 \cdot 10^9} \quad \text{dove } x = \# \text{ istruzioni}$$

Mi ricavo x

$$10s \cdot 3 \cdot 10^9 = x \cdot 1.s \Rightarrow x = \frac{10s \cdot 3 \cdot 10^9}{1.s} = \frac{30}{\frac{1}{10}} = \frac{300}{1} = 300 \cdot 10^9$$

Sa che il CPI aumenta del 20%

$$\begin{aligned} \text{CPI} &= 1.s + (1.s \cdot 0.20) \\ &= \frac{3}{2} + \left(\frac{3}{2} \cdot \frac{20}{100} \right) = \frac{3}{2} + \frac{30}{100} = \frac{15+3}{10} \\ &= \frac{18}{10} = \frac{9}{5} \Leftarrow \text{CPI aggiornato} \end{aligned}$$

Ri-scrivo le formule aggiornate

$$T_S = \frac{20 \cdot 10^9 \cdot \frac{1}{5}}{\text{freq}} \Rightarrow \text{freq} \cdot T_S = 20 \cdot 10^9 \cdot \frac{1}{5}$$

$$\Rightarrow \text{freq} = \frac{20 \cdot \frac{1}{5}}{\frac{4}{7}} = \frac{36}{7} \approx 5.14 \cdot 10^9 \xrightarrow{\text{GHz}}$$

Si vorrebbe ridurre il tempo di esecuzione del 30% (da 10 a 7 secondi), ma per raggiungere questo obiettivo il CPI aumenterebbe la frequenza di clock che consentirebbe questa riduzione del tempo di esecuzione per il processore P1? [Risposta espr

Risposta errata
La risposta corretta è : 5.14
Punteggio ottenuto 0,00 su 1,00

REFERENCE UNITÀ DI MISURA

Nome	fattore
Kibi	2^{10}
Mebi	2^{20}
Gibi	2^{30}
Tebi	2^{40}
Petebi	2^{50}
Exabbi	2^{60}
Zettabi	2^{70}
Yottabi	2^{80}

Aumentano di $2^{10} \downarrow$

Nome	fattore
Kilo	10^3
Mega	10^6
Giga	10^9
Tera	10^{12}
Peta	10^{15}
Exa	10^{18}
Zetta	10^{21}
Yotta	10^{24}

Aumentano di $10^3 \downarrow$

CODIFICA

COMPLEMENTO A 2

Parti da D_x → S_x copice tutti gli 0 fino al primo 1 incluso.

Quelli dopo invertiti.

ES : 0 1 1 0 (6)

→ 1 0 1 0 (-6)

- tutti 1 => -1 1 1 1

- tutti 1 + 1 in C₂ => 0 no overflow

- se non specificato => Sempre C₂

Segno

→ 0 : positivo

→ 1 : negativo

Supponiamo che i registri x5 e x6 contengano i numeri 0xFFFFFFFFFFFFFF e 0x0000000000000001 rispettivamente. Determinare quale sarà il contenuto di x30 dopo l'esecuzione di questa istruzione assembler:

add x30, x5, x6

$$\begin{array}{r} x_5 = \frac{1}{|1\ 1\ 1\ldots\ 1\ 1\ 1}} \\ x_6 = \frac{0\ 0\ 0\ldots\ 0\ 0\ 1}{\hline 1\ 0\ 0\ 0\ 0\ 0\ 0} = 0 \end{array} \quad \begin{array}{l} = -1 \\ \text{A 000} \\ = 1 \\ \checkmark \end{array}$$

Troncato

anche guardando riporti

$$(n+1, n), (n-1, n-2) \Rightarrow (1, 1) = (1, 1) \quad \text{no overflow}$$

COMPLEMENTO A 1

Invertire tutti i bit

ES : 0 1 1 0 (6)

1 0 0 1 (-6)

Segno

→ 0 : positivo

→ 1 : negativo

MODULO + SEGNO

Il 1° bit fa da segno

ES : 0 1 1 0 (6)

1 1 1 0 (-6)

Segno

→ 0 : positivo

→ 1 : negativo

VARIABILI NEL SEGMENTO DATI

- Segnarsi con attenzione le variabili
- Attenzione alla dimensione di load/store

Si assume che un programma RISC-V abbia le seguenti variabili definite nel Segmento Dati:

x: .word 4
y: .word 6
z: .word 5

Scrivere il numero (in rappresentazione decimale) equivalente alla parola che si troverà nella variabile z dopo l'esecuzione del seguente frammento di codice:

la t0, x
lw t0, 0(t0)
la t1, y
lw t1, 0(t1)
or t0, t1, t0
la t2, z
sw t0, 0(t2)

t0

t1

t2

&x

(4) 010000

&y

(6) 0110

(6) 0110

&z

$z = t_0 = 0110 = 6$

la t0, x

lw t0, 0(t0)

la t1, y

lw t1, 0(t1)

or t0, t1, t0

la t2, z

sw t0, 0(t2)

SIMULAZIONE RAM

- Scrivere i risultati in dimensione esatta ovvero quella riportata negli es
- La RAM è in Byte
- Le Load leggono nella RAM dall'alto verso il Basso
- Le Store scrivono Dall'alto verso il Basso Da sec
verso
Dec → Bit

Si considera che la memoria principale contenga i valori riportati nella figura, dove il numero a sinistra della cella rappresenta l'indirizzo della cella.

Memoria RAM

7	0x2B
6	0x2A
5	0x1F
4	0x1E
3	0x1D
2	0x1C
1	0x1B
0	0x1A

• Se il contenuto memorizzato dal registro $x4$ fosse `0xAABBCCDDEEFF0011` e se quello del registro $x5$ fosse `0x0000000000000000` quale sarebbe il valore (8 byte in esadecimale) del registro $x4$ se il processore eseguisse l'istruzione `lw x4, 0(x5)`? 0x1A1B1C1D ✗

• Se il contenuto memorizzato dal registro $x4$ fosse `0xDDCCBBAAABBCDD` e se quello del registro $x5$ fosse `0x0000000000000000` quale sarebbe il valore del byte all'indirizzo 2 se il processore eseguisse l'istruzione `sw x4, 4(x5)`? 0xBB ✗

1) $x_s = 0$ Indirizzo 0 come limite inferiore

$lw \Rightarrow$ word = 4 spazi

$$\hookrightarrow [0+4, 0] = [3, 0]$$

$\hookrightarrow -1$

$$x_u = 0x \ 1D \ 1C \ 1B \ 1A$$

3 2 1 0

2) $x_s = 0$ Indirizzo di fine +4

$x_u = 0x \ \underline{D} \ D C C B B A A \dots D D$ ogni blocco 2 digit

$sw \Rightarrow$ word 4 Blochi (spazi [4, 7])

7 : $0x DD$

6 : $0x CC$

5 : $0x BB$

4 : $0x AA$

z era 1C
e riceve 1

ESECUZIONE FLUSSO RISC-V

- Eseguire passo-passo le istruzioni tenendo conto del PC

Si supponga che il registro PC abbia il valore **0x0000000000400004** (in esadecimale), e che l'istruzione corrente sia la prima istruzione riportata nel frammento di codice RISC-V di seguito.

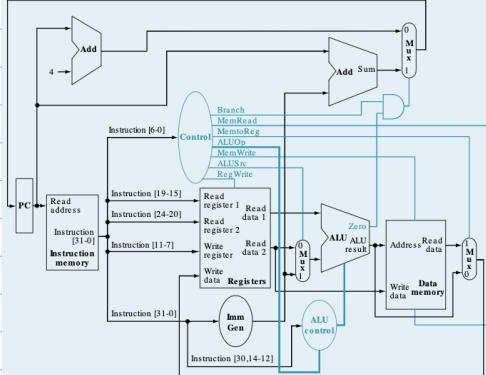
Determinare l'indirizzo che sarà presente nel PC dopo l'esecuzione dell'istruzione **bit t0, zero, cont1**

```

addi t0, zero, -4
addi t1, zero, 4
xor t0, t0, t1
bit t0, zero, cont1
addi t0, zero, 0
jal zero, cont0
cont1:
    addi t0, zero, 1
cont0:
    nop

```

Come riferimento, il PC viene aggiornato dalla seguente rete nel RISC-V:



$$PC : 0x \dots 4000004$$

+4

$$PC : 0x \dots 4000008$$

+4

$$PC : 0x \dots 400000C$$

$$\begin{array}{ll}
 A = 10 & F = 15 \\
 B = 11 & \\
 C = 12 & \\
 D = 13 & \\
 E = 14 &
 \end{array}$$

+4

$$PC : 0x \dots 4010 \overbrace{\quad}^{16}$$

+12

$$PC : 0x \dots 401C \text{ blt } \text{OK} \rightarrow \text{salto in avanti}$$

$$16 + 12 = 28 \text{ di } 32 \text{ str. } \cdot 4 \text{ bit}$$

$$\begin{array}{r}
 S \ 4 \ 1 \ 1 \ 1 \ 0 \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
 \end{array}$$

$$\begin{array}{r}
 1 \ 1 \ 1 \\
 1 \ 1
 \end{array}$$

$$PC : 0x \dots 401C \leftarrow$$

$$\text{addi } t_0, \text{zero}, -4$$

$$t_0 = -4$$

$$\text{addi } t_1, \text{zero}, 4$$

$$t_1 = 4$$

$$\text{xor } t_0, t_0, t_1$$

$$\begin{array}{r}
 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
 \end{array} \text{ (t}_1\text{)}$$

$$\begin{array}{r}
 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
 \end{array} \text{ (t}_0\text{)}$$

$$= 10000000$$

\hookrightarrow Inizia per 1 \Rightarrow <0

blt t0, zero, cont1

t0 < 0 ✓ \Rightarrow cont1

blt t0, zero, cont1

t0 < 0 ✓ \Rightarrow cont1

$$= 12 \text{ } \boxed{(C)}$$

\hookrightarrow Non contiene label

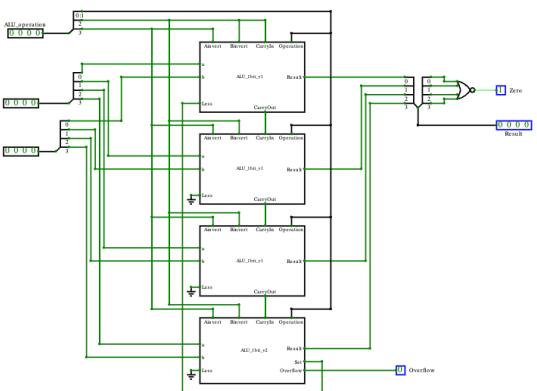
DETERMINARE USCITE ALU

- ALU OPS:

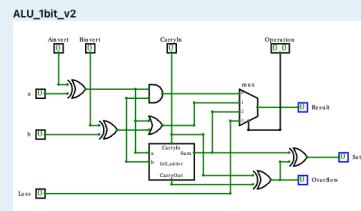
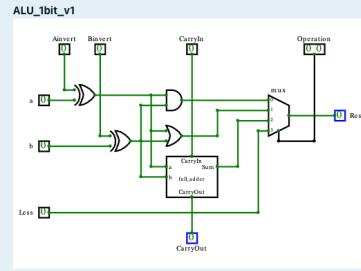
- 0000 \Rightarrow AND
- 0001 \Rightarrow OR
- 0010 \Rightarrow SUM

- 0110 \Rightarrow SUB
- 0111 \Rightarrow LESS
- 1100 \Rightarrow NOR

Si consideri una ALU a 4 bit come realizzata in laboratorio.



Questa ALU a 4 bit è composta da 3 ALU a 1 bit di "tipo 1" e 1 ALU a 1 bit di "tipo 2", che sono realizzate come nelle figure seguenti:



Determinare le uscite dall'ALU.

- (0.50 punti) Result = ✗
- (0.25 punti) Zero = ✗
- (0.25 punti) Overflow = ✗

Scrivere il valore **Result** in uscita dall'ALU come stringa di 4 bit, es. 1010.

Assumendo che siano portati in ingresso i seguenti valori binari:

- a = 0000
- b = 0001
- ALU_operation = 0010

- 2 opzioni

- ↳ mettersi a seguire il circuito
- ↳ troppo poco tempo
- ↳ fare i conti al volo seguendo il codice dell'ALU

$$A = 0000$$

$$B = 0001$$

$$\text{ALU OP: } 0010 \text{ (somma)}$$

$$\text{RESULT : } 0000 +$$

$$\begin{array}{r} 0001 \\ \hline 0001 = 1 \end{array}$$

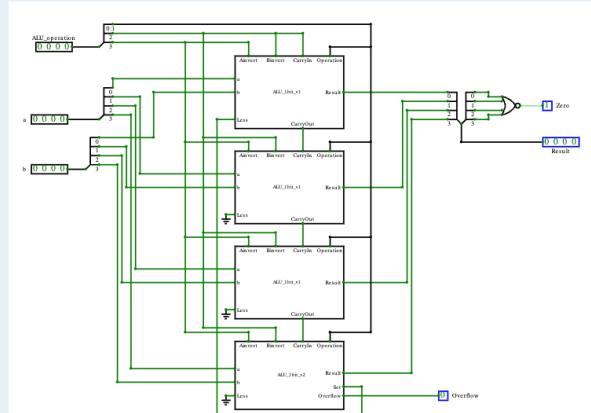
$$\text{Zero} = 0$$

$$\text{Overflow} = 0$$

DETERMINARE USCITE ALU (CONTINUAZIONE)

- ESEMPIO CON LESS (Metodo 1)

Si consideri una ALU a 4 bit come realizzata in laboratorio.



Questa ALU a 4 bit è composta da 3 ALU a 1 bit di "tipo 1" e 1 ALU a 1 bit di "tipo 2", che sono realizzate come nelle figure seguenti:

Assumendo che siano portati in ingresso i seguenti valori binari:

- $a = 1111$
- $b = 0111$
- $ALU_operation = 0111$

ALU OP: 0111
(LESS)

Il less determina $a < b$ facendo

$$a - b < 0$$

- Nego b per farci la sottrazione

$$-b = 1001$$

- eseguo $a + (-b)$

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \ 1 \\ 1 \ 1 \ 1 \ 1 \ 1 \\ + \\ \hline 1 \ 0 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} 1 \ 1 \ 0 \ 0 \ 0 \\ \hline \end{array}$$

- Mostro il risultato

$$\begin{array}{r} 0 \ 0 \ 0 \ 1 \\ \hline \end{array}$$

Result

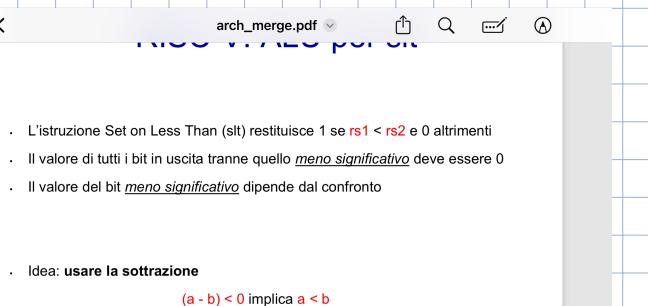
Dal risultato salvo solo il bit di segno (quello più a sinistra)

Il bit di prima diventa il bit meno significativo (0x1)

Overflow e Zero dipendono da $a + (-b)$

$$= 0$$

$$= 0$$



RISC-V: ALU per slt

- L'ingresso Less dell'ALU per i bit da 1 a 63 deve essere uguale a 0
- L'ingresso Less dell'ALU per il bit 0 (il meno significativo) deve essere uguale al bit di segno del risultato della differenza $a - b$
 - ovvero il bit della somma dell'ALU del bit 63
- Per questo l'ALU per il bit 63 ha un'uscita Set per il risultato dell'addizionatore

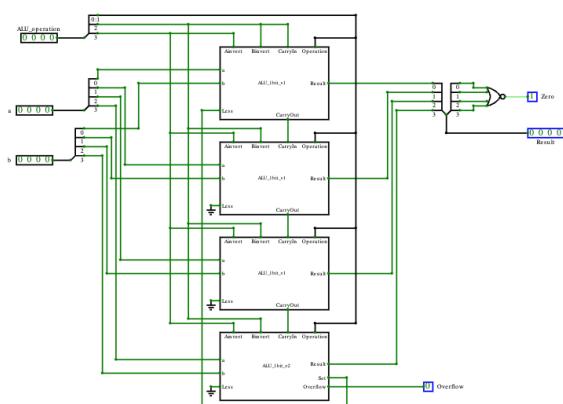


DETERMINARE USCITE ALU (CONTINUAZIONE)

- ESEMPIO CON LESS

Metodo 2 (controllo)

Si consideri una ALU a 4 bit come realizzata in laboratorio.



Questa ALU a 4 bit è composta da 3 ALU a 1 bit di "tipo 1" e 1 ALU a 1 bit di "tipo 2", che sono realizzate come nelle figure seguenti:

Assumendo che siano portati in ingresso i seguenti valori binari:

- $a = 1111$
- $b = 0111$
- $\text{ALU_operation} = 0111$

ALU OP: 0111
(LESS)

So che il less ritorna $\begin{cases} 1 & \Rightarrow a < b \\ 0 & \Rightarrow \text{altrimenti} \end{cases}$

Converti in Dec e faccio il confronto

$$\begin{aligned} a &= 1 \ 1 \ 1 \ 1 && \Rightarrow \text{tutti } 1 \text{ in } c_2 = -1 \\ b &= 0 \ 1 \ 1 \ 1 && \Rightarrow \text{Inizia con } 0 \\ &= 4 + 2 + 1 && \text{Quindi deve essere} \\ &= 7 && \underline{\text{positivo}} \end{aligned}$$

Quindi il less dovrà tornare 1

$$\text{Result} = 0 \ 0 \ 0 \underline{1}$$

Per Overflow e Zero guarda l'altro metodo

CACHE: MAPPATURA DIRETTA

- Disegnare la struttura e simulare a mano
- Se si trova un valore con stesso modulo e tag diverso \Rightarrow MISS

Di seguito viene riportata una sequenza di nove richieste di dati a una memoria cache a mappatura diretta di otto blocchi, inizialmente vuota. Gli indirizzi delle richieste sono espressi in binario a 5 bit.

Numero della richiesta - Indirizzo in binario:

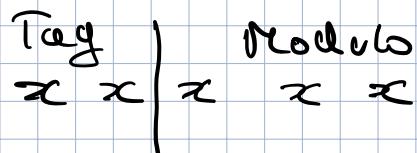
1. 01000
2. 10001
3. 01010
4. 00011
5. 00000
6. 00001
7. 01010
8. 00011
9. 01000

[0.5 punti] Completare la tabella seguente riportando lo stato della cache **dopo le nove richieste**, indicando per ogni blocco il tag del blocco memorizzato, o se il blocco è ancora non valido.

Indice: 000	Tag: 01	✓
Indice: 001	Tag: 00	✓
Indice: 010	Tag: 01	✓
Indice: 011	Tag: 00	✓
Indice: 100	Tag: Non valido	✓
Indice: 101	Tag: Non valido	✓
Indice: 110	Tag: Non valido	✓
Indice: 111	Tag: Non valido	✓

[0.5 punti] Quanti sono stati i hit e i miss nella cache?

TAB	MODULO
0+1 01	01
0+1 10	00
0+1 01	01
0+1 00	00
	0 0 0
	0 0 1
	0 1 0
	0 1 1
	1 0 0
	1 0 1
	1 1 0
	1 1 1



HIT 1 2

MISS:

1 2 3 4 5 6 7