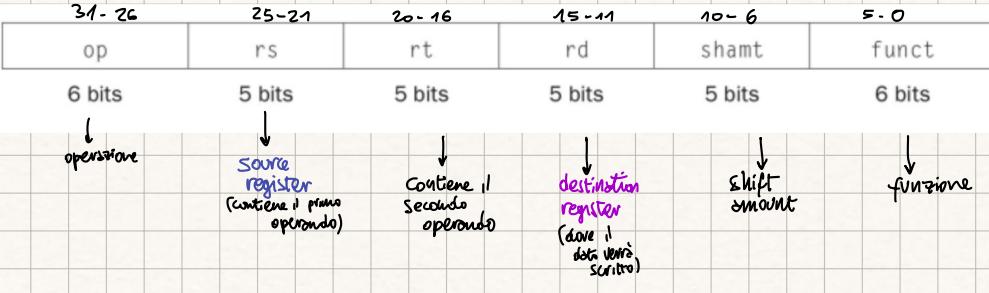


# CHEAP SWEET ARCH

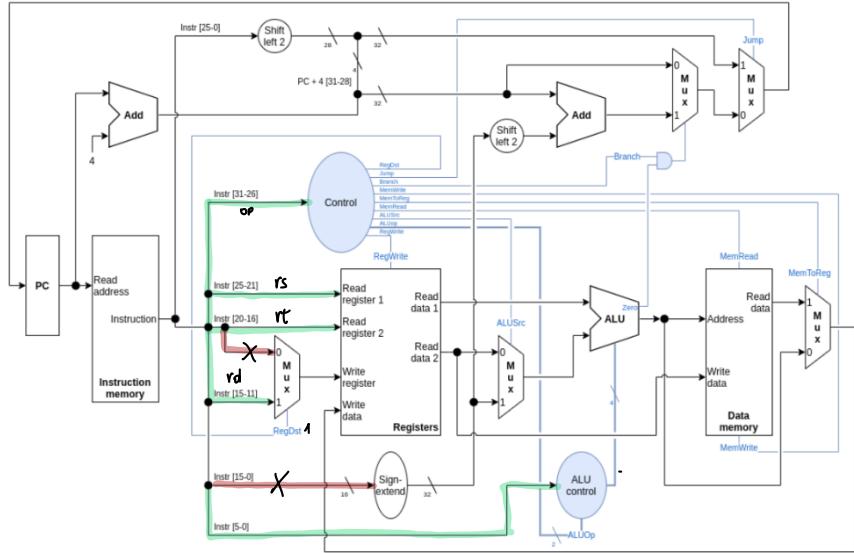
MODIFICA CPU:

IMPORTANTE!! il formato delle istruzioni: (disponibile durante il compito)

**TIPO R** aritmetiche

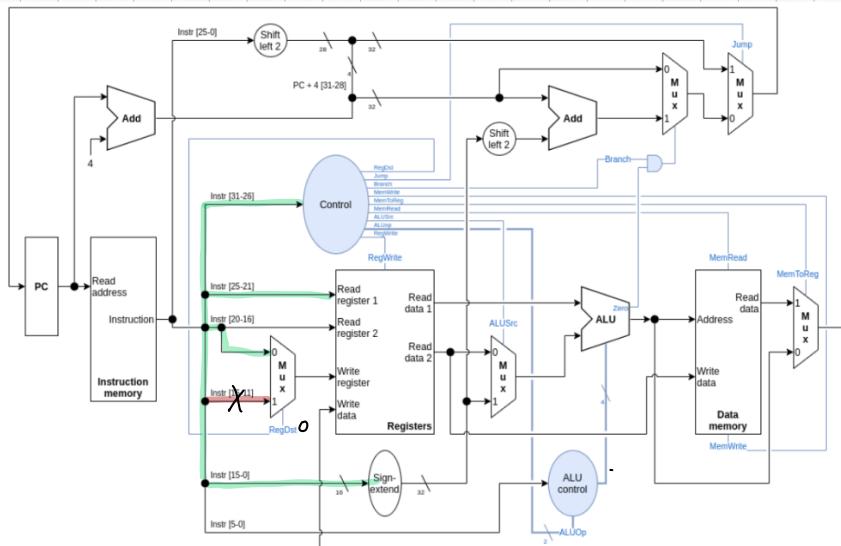


- NON C'È ACCESSO ALLA MEMORIA
- operazioni aritmetico-logiche (es add)



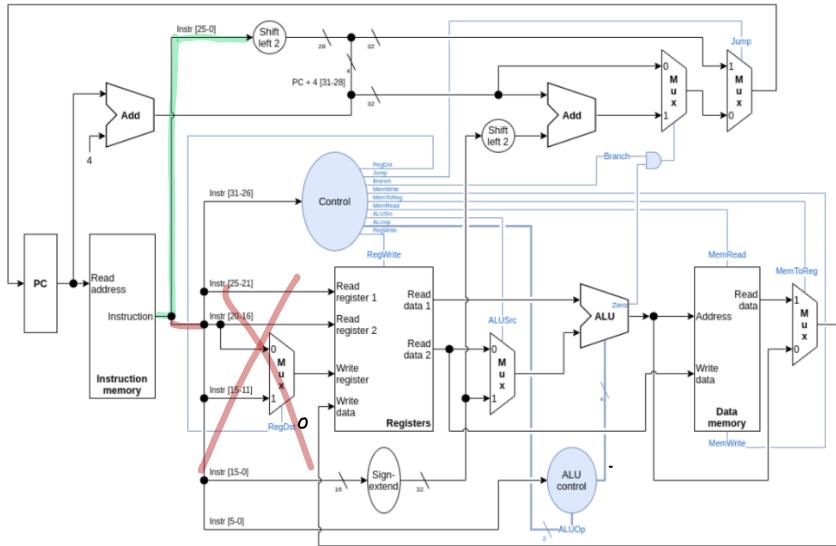
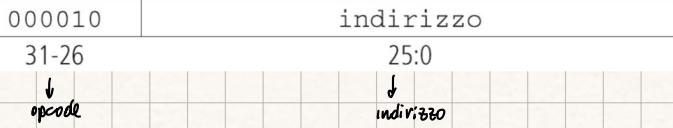
- sll / srl
- jr \$rs
- add / sub / and
- slt

**TIPO I** immediate + load



- load / store
- salti condizionati

## TIPO J salti



PIPOLINEA: • data hazard - regstr.      • control hazard - salti

① Se l'hazard è alla riga prima, DUE STALLI  
se è due righe prima, UNO STALLO ] no FW

② NOTE FORWARDING

IF ID EX MM WB

IF ID EX MM WB

• LOAD WORD, LOAD BYTE → IF ID EX MM WB

• LOAD IMMEDIATE, LOAD ADDRESS → IF ID EX MM WB

serve in ex pronto in mem

pronto in mem

• OPERAZIONI ARITMETICHE → IF ID EX MM WB

serve in id pronto in ex

note:

• 2 ISTR TIPO R (o L/LA) → 0 stalli  
IF ID EX MM WB  
IF ID EX MM WB

• 1 ISTR LW/LB.. + 1 ISTR TIPO R → 1 stallo  
IF ID EX MM WB  
IF ID EX MM WB

add \$t1, \$a0, \$a1  
lb \$s1, (\$t1) #

IF ID EX MM WB  
IF ID EX MM WB

• ARITM /L/LA + BEQ → 1 stallo  
R.e!

addi \$s0, \$t1, 5  
beq \$s0, \$s2, ciao  
Satto in ID

IF ID EX MM WB  
IF ID EX MM WB  
serve in ex  
stall  
→ IF ID EX MM WB

• LB/LH/LW + BEQ → 2 stalli

lw \$s0, (\$t1)  
beq \$s0, \$s2, ciao

IF ID EX MM WB  
IF ID EX MM WB

IF ID EX MM WB  
IF ID EX MM WB  
→ IF ID EX MM WB

100 = 4  
1000 = 8  
(tutte le potenze di 2 ≥ 4 sono div × 4)  
tipicamente nego l'or così = 1 divisibile e = 0 non)

- divisibile per 4 → prendi ultimi 2 bit e fai l'or  
(xx10 / xx01 / xx11 Non sono div 4, invece X00 si)
- controlla se le cose si possono fare su ALU/SLL già presenti  
(riutilizzare operandi/non aggiungere troppo roba)
- per "indirizzi" un array, si somma l'indirizzo base all'indice  
(indice ≠ 4 se è un vettore di word)

RICORDA: • se scrivi due volte sulla stessa registro NON È HAZARD

• se hai messo uno stall, conta come un'altra istruzione  
add \$t0, \$s1, \$s1  
→ istruzione 2 str. in mezzo!  
add \$s0, \$t0, \$t0 no stalli

## ESERCIZIO 4.11 (trickiness)

ESEMPIO: devo trovare lo stato della pipeline al 130 clock

- ① (se senza fw) riscrivo i numeretti intorno alle istruzioni che mi servono (con stalli)

12	lb	\$s1,	<u>(\$t1)</u>	# \$s1
→ 14	and	\$s2,	<u>\$s1,</u>	1 # bi
15	sle	\$t0,	<u>\$s2,</u>	\$0 # \$
→ 17	bne	\$t0,	<u>\$zero,</u>	sum
18	sll	\$s1,	<u>\$s1,</u>	1 # A7

i numeretti rappresentano il ciclo in cui le istruzioni FINISCONO (cioè fase WB) quindi 13 è qui → in WB c'è BOLU

dopodiché, si scende in istruzioni ma si "sale" in fasi  
OVVERO

VB balls, MEM and, EX slc, ID balls, IF bne

12 WB	MM	lb	\$s1,	(\$t1) # \$s1
→ 14 EX		and	\$s2,	\$s1, 1 # bi
15 ID	IF	sle	\$t0,	\$s2, \$0 # \$
→ 17		bne	\$t0,	\$zero, sum
18		sll	\$s1,	\$s1, 1 # A1

## CACHE

Sia data una CPU con processore a 24GHz e 16 CPI (Clock per Instruction) che adoperi indirizzi da 32 bit e memoria strutturata su due livelli di cache (L1, L2),

il cui setup è come segue:

L1 è una cache set-associativa a 8 vie con 2 set e blocchi da 16 word; adopera una politica di rimpiazzo LRU. Ricordiamo che consideriamo il set come

L'insieme del blocco in cache con tag e bit di validità, mentre la linea è il gruppo di set con il medesimo indice.

L2 è una cache direct-mapped con 8 linee e blocchi da 128 word.

- ① trovare blocco, indice, tag degli address

**DIRECT MAPPED** 1 vif - #set = #linee

## SET-ASSOCIATIVE

## CALCOLARE MISS E HIT

① se la combinazione blocco-index-tag non è mai stata inserita → COLD START

② se vedi una combinazione già inserito → guarda i precedenti CON STESSO indice safer controllare con disegno cache se prima c'è una Miss stessa tag → la scatti

se entro  $X$  c'è lo stesso blocco  $\rightarrow$  HIT  
con  $x \leftarrow$  numero di vie

some species? ⇒ calcium is dangerous for some species

CACHE FULLY ASSOCIATIVE  $\rightarrow$  set : var

fully associative correspondence = 4-1															
	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M	M	M	M	M	M	HIT	M	M	HIT	M	M	M	

Id	cold	cold	...	cold	cold	...	cap	cap	c
non c'c'	4	3		2	2	1 ↘			
non c'c'	4	3		3	2	1 ↘			

⑥ contenere entro dati primi → se il dato c'è → CONFLICT  
 ↴  
 "SALTANDO"       $x_i$  dimensione  
 ↓                  fully associative      ↴ se non c'è → CAPACITY  
 i DUPLICANTI!

- 2) Calcolare le dimensioni in bit (compresi i bit di controllo ed assumendo che ne basti uno per la LRU) delle due cache: (a) L1 e (b) L2.  
 3) Assumendo che gli accessi in memoria impiegano 400 ns, che gli hit nella cache L1 impiegano 10 ns e gli hit nella cache L2 impiegano 20 ns, calcolare  
 (a) il tempo totale per la sequenza di accessi, (b) il tempo medio per la sequenza di accessi, e (c) quante istruzioni vengono svolte nel tempo medio calcolato.  
 4) Calcolare il word offset del sesto indirizzo (8680) per la cache L2 spiegando i calcoli effettuati.  
 5) Supponendo che gli indirizzi nella tabella siano virtuali e la memoria virtuale consti di 512 pagine di 4KiB ciascuna, indicarne i numeri di pagina virtuale. Si assume che la cache sia a monte della memoria virtuale.

Address	2000	2124	8082	200	8512	8680	9048	264	216	264	320	2024
Page#												

### CALCOLO DELLE DIMENSIONI

per la dimensione mi servono ④ dimensioni blocco in byte e bit (word + 4 \* 8)

② l'offset di word →  $\log_2$  dim blocco in byte

③ l'index →  $\log_2$  n° set / linee per direct mapped

④ il TAG → 32-offset - index

⑤ FORMULA FINALE:

$$(n^{\circ} \text{ vie} \cdot n^{\circ} \text{ set}) \left( \begin{array}{l} \text{c'è sempre} \\ \text{non nelle} \\ \text{/ direct} \end{array} \right) (\text{VALIDITY BIT} + \text{LRU} + \text{dim blocco in bit} + \text{tag})$$

### CALCOLO DEL TEMPO

① ACCESSI IN MEMORIA = MISS 2a CACHE (gli altri sono self-explanatory)

② TEMPO MEDIO = tempo totale / n° colonne cache (n° dati)

③ QUANTE ISTRUZIONI =  $\frac{\text{tempo medio}}{\text{CPI}} \cdot \text{GHz}$

### WORD OFFSET DI UN'ISTRUZIONE

$$\frac{(\text{address \% dim blocco in byte})}{4}$$

### MEMORIA VIRTUALE

→ Calcolare il word offset del sesto indirizzo (8680) per la cache L2 spiegando i calcoli effettuati.

$$6 \cdot 1024 = 6096$$

5) Supponendo che gli indirizzi nella tabella siano virtuali e la memoria virtuale consti di 512 pagine di 4KiB ciascuna, indicarne i numeri di pagina virtuale. Si assume che la cache sia a monte della memoria virtuale.

Address	8076	8080	4096	4100	2000	2040	2076	2096	8080	8084	4104	4196	1408
Page#	1	1	1	1	0	0	0	0	1	1	1	1	1

$$\frac{\text{address}}{1024 \cdot n^{\circ} \text{ KiB}}$$