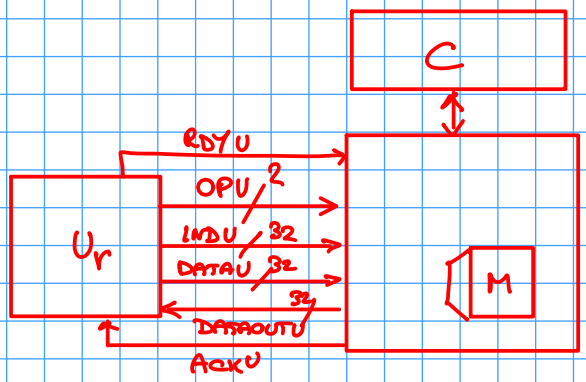
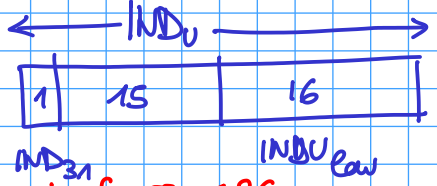
Domanda 1



Schema generale



Microcodice

Ø. (RDYU, $OP_Ø$, OP_1, $INDU_{31}$ Ø - - -) nop, Ø

R(CACHE) (=1, Ø, Ø, Ø) INDU → IND, "read" → OP, set RDYM, 1

W(CACHE) (=1, Ø, 1, Ø) INDU → IND, DATAU → DATAOUT, "write" → OP, set RDYM, 2

R(M) (=1, Ø, Ø, 1) M[$INDU_{low}$] → DATAOUTU, reset RDYU, set ACKU, Ø

W(M) (=1, Ø, 1, 1) DATAU → M[$INDU_{low}$], reset RDYU, set ACKU, Ø

ZERO(M) (=1, 1, -, -) Ø → I, 3

1. (ACK_M, or(ESITO) = 0 -) nop, 1 // lettura dalla cache
   (=10) reset ACK_M, set ACKU, reset RDYU, DATAIN → DATAOUTU, Ø
   (=11) " " " " " , fatt_ecc

2. (ACK_M, OR(ESITO) = 0 -) nop, 2 // scrittura nella cache
   (=10) reset ACK_M, set ACKU, reset RDYU, Ø
   (=11) " fatt_ecc

3. (I_0 = 0) 0 → M[I], I+1 → I, 3 // azzeramento memoria interna
   (=1) reset RDYU, set ACKU, Ø

Valutazione ciclo di clock

PC max 4 var di condizionamento testate x μ-istruzione
2 bit di stato
⇒ 6 input x livello AND

14 frasi
⇒ max 13 "1" per α_i e β_j ⇒ 2 livelli OR =
che si riducono a 1 livello OR codificando gli "0" e negando l'uscita in caso #"1" > 8

Dunque $T_{WPC} = T_{OPC} = 2t_p$

PO $T_{WPO} = 1t_p$ (XOR(ESITO))

T_{OPO} determinato da DATAU → M[INDU$_{lav}$] ← non costa niente: prende le 16 uscite da 1 bit + basse di INDU

$t_k + t_a$

↳ commutatore × indirizzo (INDU o I)

↳ tempo di accesso × scrittura

Quindi la fase + lunga (1.1) richiede

$$\tau = \underbrace{1t_p}_{T_{WPO}} + \max\left\{\underbrace{2t_p}_{T_{OPC}}, \underbrace{2t_p}_{T_{WPC}} + \underbrace{\underbrace{t_k}_{2t_p} + \underbrace{t_a}_{16t_p}}_{T_{OPO}}\right\} + \underbrace{t_p}_{\delta} =$$

$$= (1 + \max\{2, 14\} + 1)\, t_p = \underline{16t_p}$$

Cicli di clock × operazione esterna

OP SU CACHE $\left\{\begin{matrix} R \\ W \end{matrix}\right\}$ $2\tau + \begin{cases} 2\tau & \text{Hit} \\ 100\tau & \text{Miss} \end{cases}$

OP SU M $\left\{\begin{matrix} R \\ W \end{matrix}\right\}$ 1τ

AZZERAMENTO M $1\tau + 64k\,\tau$

Domanda 2

Compilazione in D-RISC standard

			tesec	A[i] pari	dispari
loop:	LOAD	R_{baseA}, R_i, R_{ai}	$2\tau + t_a$	x	x
	AND	$R_{ai}, \#1, R_c$	τ	x	x
	IF=0	R_c, then	2τ	x	x
else:	LOAD	R_{baseB}, R_i, R_{bi}	$2\tau + t_a$		x
	INC	R_{bi}	τ		x
	STORE	R_{baseB}, R_i, R_{bi}	$3\tau + t_a$		x
	GOTO	cont	τ		x
then:	LOAD	R_{baseB}, R_i, R_{bi}	$2\tau + t_a$	x	
	LOAD	R_{baseC}, R_i, R_{ci}	$2\tau + t_a$	x	
	ADD	R_{bi}, R_{ci}, R_{ai}	τ	x	
	STORE	R_{baseA}, R_i, R_{ai}	$3\tau + t_a$	x	
cont:	INC	R_i	τ	x	x
	IF<	R_i, R_n, loop	2τ	x	x
				$16\tau + 4t_a$	$15\tau + 3t_a$

iterazione pari: $16\tau + 4t_a + 9(2\tau + t_a) : T_{ch}$ → $34\tau + 13t_a$

iterazione dispari: $15\tau + 3t_a + 9(2\tau + t_a) : T_{ch}$ → $33\tau + 12t_a$

nel testo non è specificata la percentuale di A[i] pari ⇒ assumo 50%

Tempo di completamento ideale

$$T_{c\,id} = N\left(\frac{1}{2}(34\tau + 13t_a) + \frac{1}{2}(33\tau + 12t_a)\right)$$

fault
- 1 fault × codice (allineato)
- N/5 fault × A
- N/5 fault × B
- N/20 fault × C

} letti ad ogni iterazione

(letto solo in ½ iterazioni)

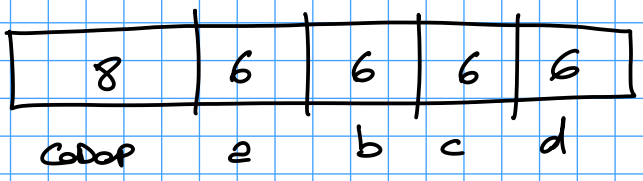
$$T_{fault} = 2(\tau + T_{tr}) + \frac{16}{m}\tau_m + m\tau = \quad (T_{tr} = 10\tau,\ m = 4)$$

$$= 22\tau + 4\tau_m + 4\tau = 26\tau + 4\tau_m$$

$$T_C = T_{c\,id} + \#fault \times T_{fault}$$

$$T_C = N\left(\frac{1}{2}(34\tau + 13t_a) + \frac{1}{2}(33\tau + 12t_a)\right) + \left(1 + \frac{2N}{5} + \frac{N}{20}\right) \times (26\tau + 4\tau_m)$$

MOVIFØ R_a, R_b, R_c, R_d



formato in memoria

t_{exec} (serve μ-codice)

movifØ.Ø. (OR(REG[IR.R_a]), INT = ØØ)

REG[IR.R_b] → REG[IR.R_d], IC+1 → IC, chØ

(=01) " " , batt_int

(=10) REG[IR.R_c] → REG[IR.R_d], IC+1 → IC, chØ

(=11) " " , batt_int

costo ⇒ 1Σ

va osservato che introduciamo un $T_{CICLO} = 2T_p$ mentre nell'interprete di riferimento $T_{CICLO} = Ø$!!

Se volessimo mantenere $T_{CICLO} = Ø$ dovremmo scrivere:

movifØ.Ø. OR(REG[IR.R_a] → COND), movifØ.1

movifØ.1. (COND, INT =) } tutto come prima

che porta il costo t_{exec} movifØ = 2Σ

Compilazione D-RISC estesa:

```
loop:  LOAD   R_baseA, R_i, R_ai
       LOAD   R_baseB, R_i, R_bi
       LOAD   R_baseC, R_i, R_ci
       ADD    R_bi, R_ci, R_bc
       INC    R_bi
       AND    R_ai, #1, R_c
       MOVIFd R_c, R_ac, R_bi, R_ai
       STORE  R_baseA, R_i, R_ai
       INC    R_i
       IF<    R_i, R_N, loop
```

	t_{exec}	t_p
	2	1
	2	1
	2	1
	1	
	1	
	1	
	2	
	3	1
	1	
	2	

t_{cl} = $10(2\tau + t_a)$

$17\tau + 4t_a + 20\tau + 10t_a$

$37\tau + 14t_a$

$$T_{cid} = N(37\tau + 14t_a)$$

(Prima era $T_{cid} > 33\tau + 12t_a$)

Codice ASM + semplice MA:

maggior costo di esecuzione dovuto al fatto che si calcola comunque sia il ramo then che il ramo else ☹