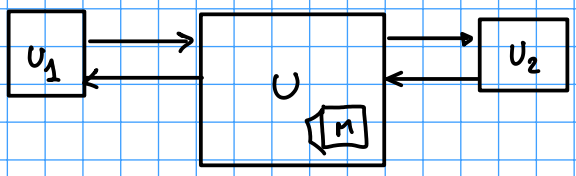


ESERCIZIO 1

Interfaccia verso U_1 :
- IND_1 (ingresso 10 bit)
- VAL_1 (ingresso 32 bit)
- RDY_1 ingresso } segnalatori a funzione di livello
- ACK_1 uscita }

verso U_2 :
- IND_2 (ingresso 10 bit) — $OP = 0$ (leggi IND)
- OP_2 (ingresso 1 bit) — $OP = 1$ (azzera M)
- RDY_2 ingresso } segnalatori a funzione di livello
- ACK_2 uscita }
- VAL_2 (uscita 32 bit)

$\emptyset$. // inizializzazione della memoria
$1 \to I,\ \emptyset \to M[\emptyset],\ 1$

1. $(I_0 = \emptyset)$ $\emptyset \to M[I],\ I+1 \to I,\ 1.$
   $(= 1)$ nop, 2

2. $(RDY_1, RDY_2, OP = \emptyset\emptyset -)$ nop, 2 // nessuna richiesta
   $(= 10 -)$ $VAL_1 \to M[IND_1]$, reset RDY_2, set ACK_1, 2 // op da U_1
   $(= -10)$ $M[IND_2] \to VAL_2$, reset RDY_1, set ACK_2, 2 // op Read da U_2
   $(= -11)$ $1 \to I,\ \emptyset \to M[\emptyset],\ 3$ // op ZERO da U_2

3. $(I_0 = \emptyset)$ $\emptyset \to M[I],\ I+1 \to I,\ 3$
   $(= 1)$ reset RDY_2, set ACK_2, 2

osservazione: lo 1 e lo 3 sono simili
⇒ distinguiamo il caso con
un di contatore settato in $\emptyset$.

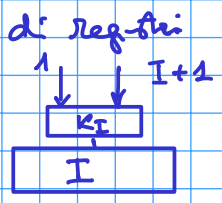
$\emptyset$. $1 \to I,\ \emptyset \to M[\emptyset],\ 1 \to C,\ 1$

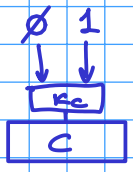
1. $(I_0, C = \emptyset -)$ $\emptyset \to M[I],\ I+1 \to I,\ 1.$
   $(= 11)$ $\emptyset \to C,\ 2$
   $(= 10)$ reset RDY_2, set ACK_2, 2

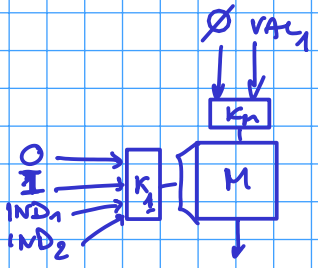
2. $(RDY_1, RDY_2, OP = \emptyset\emptyset -)$ nop, 2 // nessuna richiesta
   $(= 10 -)$ $VAL_1 \to M[IND_1]$, reset RDY_2, set ACK_1, 2 // op da U_1
   $(= -10)$ $M[IND_2] \to VAL_2$, reset RDY_1, set ACK_2, 2 // op Read da
   $(= -11)$ $1 \to I,\ \emptyset \to M[\emptyset],\ 1$ // op ZERO da U_2

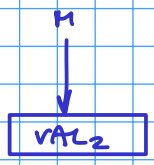
Le 1 e le 2 potrebbero essere "unite notando che nel caso $(=11)$ della 1 si potrebbe direttamente cominciare a considerare le richieste di operazione da U_1 o U_2.

Classi di registri

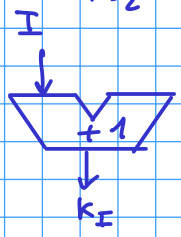








Risorse di calcolo



+ tutti gli indicatori e transizione di livello

Variabili di condizionamento

I_0 C OP RDY_1 RDY_2 (tutti registri $T_{WPO} = \emptyset$)

Registro di stato della PC

2 bit (3 μ-istruzioni)

Variabili di controllo

α_{K_I} α_{K_C} α_{K_M} α_{K_1}

β_I β_C β_M β_{VAL_2} β_{RDY_1} β_{RDY_2} β_{ACK_1} β_{ACK_2}

PC

livello AND (5 V.C. + 2 bit stato = 7 ingressi $\Rightarrow 1 t_p$)

livello OR (8 fasci $\Rightarrow 1 t_p$)

$T_{\sigma PC} = T_{WPC} = 2 t_p$

$T_{WPO} = \emptyset\ t_p$

$T_{\sigma PC} = \max \{ t_a + t_k$, ($M[IND_2] \rightarrow VAL_2$)

$t_{alu} + t_k \} =$ ($I+1 \rightarrow I$)

$t_{alu} + t_k$

$= t_k + \max \{ t_a, t_{alu} \}$

(non specificato ma presumibilmente: $t_a > t_{alu}$ (++))

Dunque

$$\tau = T_{WPO} + \max \begin{Bmatrix} T_{OPC}, \\ T_{WPC} + T_{OPO} \end{Bmatrix} + \delta =$$

$$= \emptyset + \max \begin{Bmatrix} 2t_p, \\ 2t_p + t_k + t_a \end{Bmatrix} + t_p =$$

$2t_p$ per questo numero di ingressi

$$= 5t_p + t_a$$

Tempo medio di elaborazione (non considero inizializzazione, perché avviene solo la prima volta con un costo pari a $(1+1k)\tau$)

$$75\% = 3/4 \qquad 20\% = 1/5 \qquad 5\% = 1/20$$

$$T = \frac{3}{4}(\tau) + \frac{1}{5}(\tau) + \frac{1}{20}(1+1k)\tau =$$

$$= \frac{15\tau + 4\tau + 1 + 1k}{20}\tau = \left(\frac{1k+20}{20}\right)\tau = \frac{1k}{20}\tau + \tau =$$

$$\simeq 51\tau + \tau = 52\tau$$

$$B = \frac{1}{T} = \frac{1}{52\tau} = \frac{1}{52(5t_p + t_a)}$$

ESERCIZIO 2

Il programma legge sequenzialmente il vettore z[] e scrive sequenzialmente il vettore x[]. Il vettore y[] è accedute in modo random.

Il codice è accedute di norma sequenzialmente. In questo caso avremo una decina di istruzioni x il codice del for + le istruzioni + la funzione F

Dunque

	Località	riuso
codice	✓	✓
x	✓	
y		✓
z	✓	

Il working set sarà dunque fatto dal codice, da 1 pagina di x e 1 di z e da tutto y.

fault : pochi fault per il codice (uno x il for e 4 x la F)

1K/16 per z

0 per x (solo scrittura)

128/16 per y

La differenza fra i due sistemi riguardano solo la cache LM che non pone particolari problemi né di capacità né di indirizzamento (è facile verificare che la compilazione del codice del For seguito immediatamente dalla compilazione del codice di F genera accessi a indirizzi distinti).

Dunque in entrambi i casi il degrado delle prestazioni sarà pari a

$$\#fault \times T_{transf}$$

$$(64 + 0 + 8 + 5) \times 2 \cdot 16 \cdot \tau = 77 \cdot 2 \cdot 16 \cdot \tau$$

Per valutare correttamente il numero dei fault relativi al codice, compiliamo.

```
0          CLEAR  Ri
1   loop:  LOAD   Rbasez, Ri, Rzi
2          MOD    Rzi, #128, Riy
3          LOAD   Rbasey, Riy, Ryi
4          CALL   RF, Rret
5          ADD    Rris, Ryi, Rxi
6          STORE  Rbasex, Ri, Rxi
7          INC    Ri
8          IFL    Ri, RN, loop
9          END
```

R_{zi} registro per parametro in ingresso di F

R_{ris} registro x parametro in uscita di F

Formato MOD

(range $[0, 2^{12}-1]$ oppure $[-2^{11}, 2^{11}-1]$)

9 istruzioni + 60 della F in totale

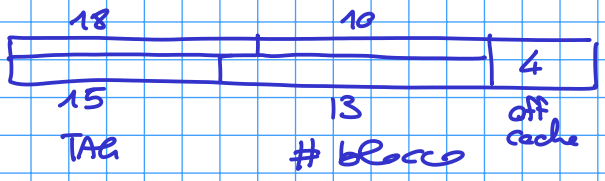
dunque sufficiente a contenere $2^7 = 128$

69 istruzioni

possono essere contenute nello stesso pagina (VM, normalmente da 4K)

x cache set assoc e indirizzamento diretto ($\sigma = 16$)

IF



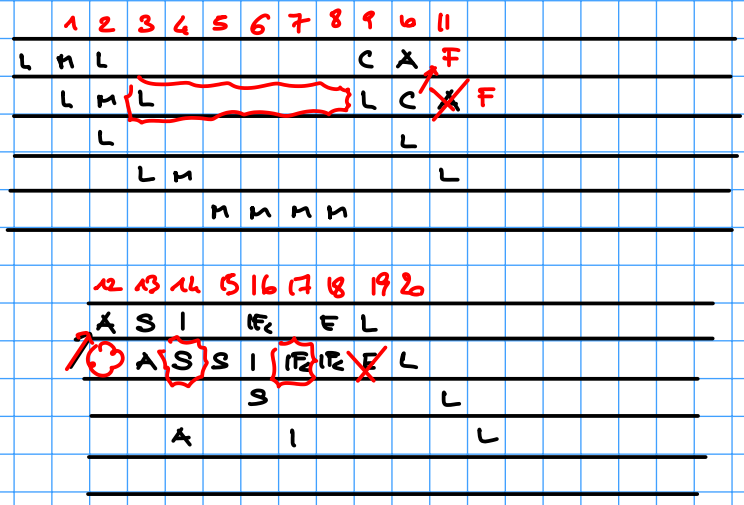
assoc. su insieme 1K insieme diretta 8K insieme

69 istruzioni ⇒ offset di pagina non nullo x 7 bit — Max

2^3 blocchi consecutivi x codice ⟵ 3 (# blocco) + 4 (off cache)

Se volessimo valutare anche il degrado delle prestazioni legato al processore pipeline, considriamo le dip nel codice:

```
loop:  LOAD   R_basez, R_i, R_zi
       MOD    R_zi, #128, R_iy      ] EU-IU
       LOAD   R_basey, R_iy, R_yi   ]
       CALL   R_F, R_ret
       ADD    R_ris, R_yi, R_xi     ] EU-IU
       STORE  R_basex, R_i, R_xi    ]
       INC    R_i                   ] EU-IU
       IF<    R_i, R_N, loop        ]
       END
```



$$T_S = \frac{19+60t}{8+60}$$

$\Rightarrow$ degrado delle prestazioni sulla sola parte del codice "nota" assumendo che il codice della funzione sia ottimizzato

pari a $\frac{10t}{8+60t}$