COMPITINO #2 di CALCOLATORI ELETTRONICI 1 del 03-12-03 **MATRICOLA**_____

(versione riveduta da errori di stampa) **COGNOME**_____

NOME_____

1) Un calcolatore con processore R3000 avente frequenza di clock pari a 2GHz esegue il seguente programma:

```
        .text
        addi  $12,$0,2
        addi  $10,$0,288
loop:   beq   $10,$0,fine
        lw    $2,100($10)
        add   $2,$2,$12
        sw    $2,200($10)
        j     loop
        addi  $10,$10,-4
fine:   halt
```

Si assuma che il processore R3000 ammetta la scrittura e la lettura di un registro nello stesso ciclo di clock, che sia possibile sfruttare il cosiddetto "delay-slot" generato dalle istruzioni di salto, che sia possibile decidere il salto nello stadio di decodifica.
Si calcoli lo speed-up (con almeno 4 cifre di precisione) relativo all'esecuzione con i circuiti di propagazione (forwarding) della pipeline abilitati rispetto al caso in cui siano disabilitati.

Riepilogo del significato delle istruzioni

Instruction	Example	Meaning	Comments
add	`add $1,$2,$3`	$1 = $2 + $3	3 operands; exception possible
add immediate	`addi $1,$2,100`	$1 = $2 + 100	+ constant ; exception possible
load word	`lw $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
store word	`sw $1,100($2)`	Memory[$2+100] = $1	Data from register to memory
branch on equal	`beq $1,$2,100`	if ($1 == $2) go to PC+4+100	Equal test; PC relative
set on less than	`slt $1,$2,$3`	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; 2`s complement
no operation	`nop`	execute but do nothing	
jump	`j 10000`	go to 10000	Jump to target address
halt the execution	`halt`	stop	halt the execution

2) Si consideri una cache di dimensione 64B e a 2 vie. La dimensione del blocco e' 32 byte, il tempo di accesso alla cache e' 5 ns e la penalita' in caso di miss e' pari a 60 ns, la politica di rimpiazzamento e' LRU. Il processore effettua i seguenti accessi in cache, ad indirizzi al byte:
21, 77, 66, 146, 82, 15, 130, 64, 192, 209, 225, 241, 113, 97, 273, 66, 257, 129, 64, 161, 240, 299.
Per la sequenza data, ricavare il tempo medio di accesso alla cache, riportare i tag contenuti in cache al termine e la lista dei blocchi (ovvero il loro indirizzo) via via eliminati durante il rimpiazzamento.

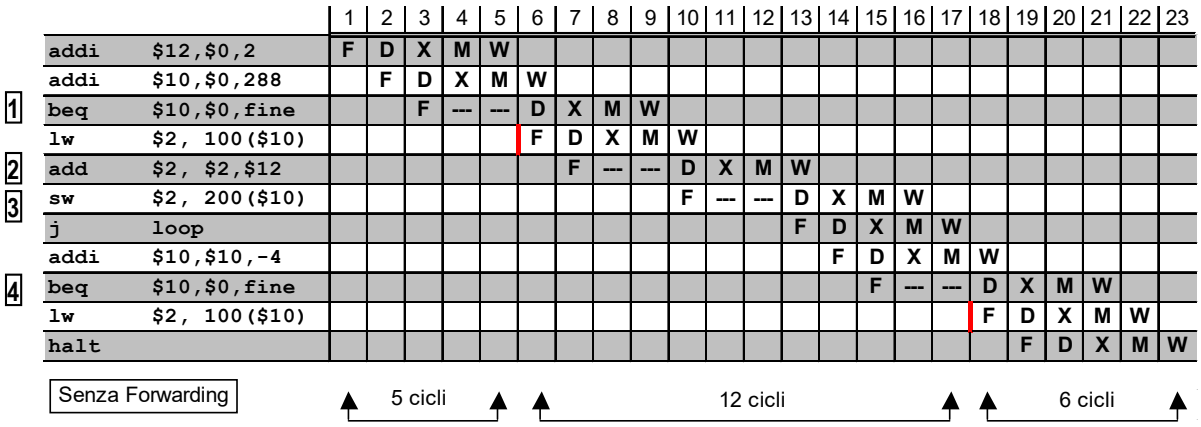
COMPITINO #2 di CALCOLATORI ELETTRONICI 1 del 03-12-03 **SOLUZIONE**

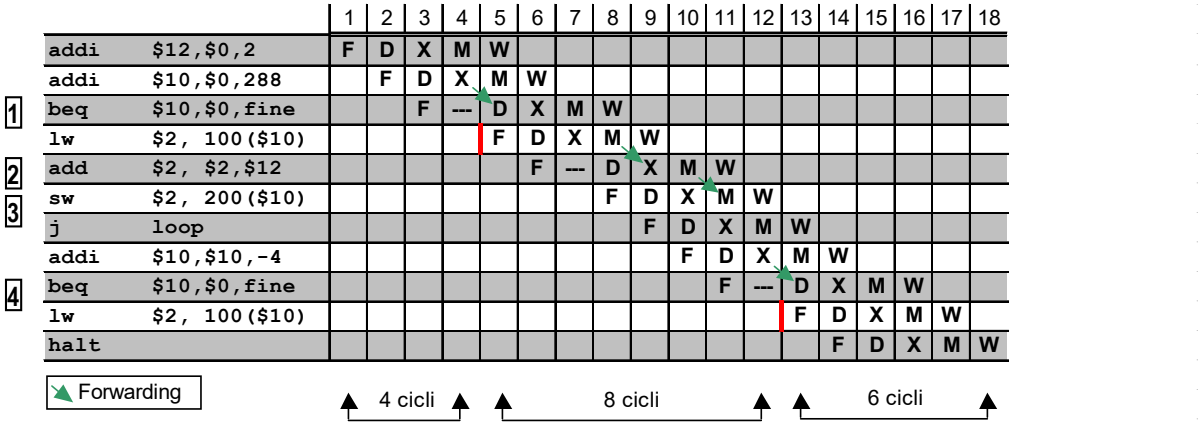
1) Una volta disegnato il diagramma di funzionamento della pipeline (F=Fetch, D=Decode, X=Execute, M=data Memory access, W=Write-back), osservando in particolare che, a causa del delay slot dopo `beq` e `j`, vengono comunque eseguite le istruzioni immediatamente successive alle istruzioni di salto e diramazione, e rispettando gli stalli dovuti alle dipendenze fra i registri (evidenziati in figura con le etichette 1, 2, 3, 4), si ricava che i cicli necessari per eseguire il programma trascurando gli stalli dovuti agli accessi in memoria sono:

$C_{NO\text{-}F}(I) = 5+I*12+6 = 11+I*12$

Il fattore N e' dovuto alla ripetizione della porzione centrale del codice che si trova all'interno del ciclo etichettato "loop". Nel nostro caso I=72, in quanto il numero di ripetizioni e' condizionato dal valore N=288 caricato dalla seconda istruzione del programma. Tale valore viene decrementato di 4 ad ogni ciclo, quindi il numero di iterazioni e' pari a 288/4=72.

$C_{NO\text{-}F}(72) = 875$





Nel caso in cui il forwarding sia abilitato e' possibile risparmiare qualche ciclo, rispettivamente nei casi:

1) e' possibile propagare il risultato della somma ("addi") allo stadio D dove si effettua il confronto fra i due operandi per la decisione sulla diramazione ("beq");
2) e' possibile propagare il risultato della lettura in memoria ("lw") allo stadio X per effettuare la somma ("add");
3) e' possibile propagare il risultato della somma ("add") allo stadio M in cui si fa accesso alla memoria dati per effettuare una operazione di scrittura ("sw");
4) e' possibile propagare il risultato della somma ("addi") allo stadio D (come nel caso 1); notare che questa dipendenza funzionale e' generata dal fatto che e' attivo il delay-slot della istruzione di salto ("j loop").

Risulta quindi (con forwarding abilitato):

$C_F(I) = 4+I*8+6 = 10+I*8$

ovvero:

$C_F(72) = 586$

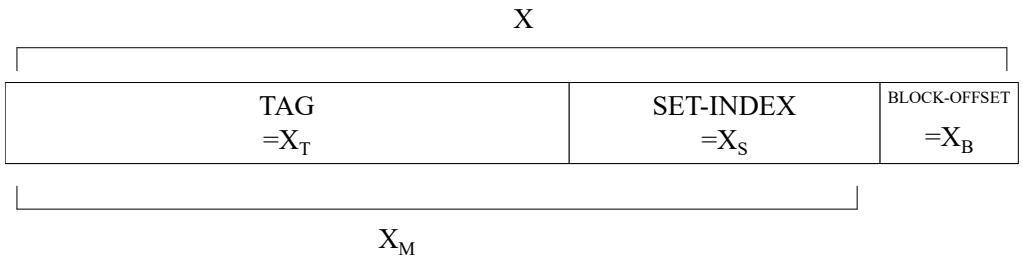
Lo speed-up risulta infine (dato che la frequenza di clock non cambia nei due casi):

$S = T_{CPU\text{-}NO\text{-}F}/T_{CPU\text{-}F} = (C_{CPU\text{-}NO\text{-}F}*f_C) / (C_{CPU\text{-}NO\text{-}F}*f_C) = C_{CPU\text{-}NO\text{-}F} / C_{CPU\text{-}NO\text{-}F} = 875/586 = 1.493$

Al variare di N si osserva inoltre che lo speed-up ottenuto cambia leggeremente, ad es.:

N	I	C_{NO-F}	C_F	S
288	72	875	586	1.493
144	36	443	298	1.487
72	18	227	154	1.474
36	9	119	82	1.451

2) Sia X il generico riferimento, A=associativita'=2, B=dimensione del blocco=32, C=capacita' della cache=64.



Si ricava S=C/B/A=# di set della cache=64/32/2=1, X_M=X/B, X_S=X_M%S, X_T=X_M/S:

	X	XM	XT	XS	XB	HIT	BLOCCO USCENTE	BIT LRU	BIT VALIDITA'	TAG
===	21	0	0	0	21	0		LRU=1,0	V=1,0	0,-
===	77	2	2	0	13	0		LRU=0,1	V=1,1	0,2
===	66	2	2	0	2	1		LRU=0,1	V=1,1	0,2
===	146	4	4	0	18	0	(out: xb=0 xt=0 xs=0)	LRU=1,0	V=1,1	4,2
===	82	2	2	0	18	1		LRU=0,1	V=1,1	4,2
===	15	0	0	0	15	0	(out: xb=4 xt=4 xs=0)	LRU=1,0	V=1,1	0,2
===	130	4	4	0	2	0	(out: xb=2 xt=2 xs=0)	LRU=0,1	V=1,1	0,4
===	64	2	2	0	0	0	(out: xb=0 xt=0 xs=0)	LRU=1,0	V=1,1	2,4
===	192	6	6	0	0	0	(out: xb=4 xt=4 xs=0)	LRU=0,1	V=1,1	2,6
===	209	6	6	0	17	1		LRU=0,1	V=1,1	2,6
===	225	7	7	0	1	0	(out: xb=2 xt=2 xs=0)	LRU=1,0	V=1,1	7,6
===	241	7	7	0	17	1		LRU=1,0	V=1,1	7,6
===	113	3	3	0	17	0	(out: xb=6 xt=6 xs=0)	LRU=0,1	V=1,1	7,3
===	97	3	3	0	1	1		LRU=0,1	V=1,1	7,3
===	273	8	8	0	17	0	(out: xb=7 xt=7 xs=0)	LRU=1,0	V=1,1	8,3
===	66	2	2	0	2	0	(out: xb=3 xt=3 xs=0)	LRU=0,1	V=1,1	8,2
===	257	8	8	0	1	1		LRU=1,0	V=1,1	8,2
===	129	4	4	0	1	0	(out: xb=2 xt=2 xs=0)	LRU=0,1	V=1,1	8,4
===	64	2	2	0	0	0	(out: xb=8 xt=8 xs=0)	LRU=1,0	V=1,1	2,4
===	161	5	5	0	1	0	(out: xb=4 xt=4 xs=0)	LRU=0,1	V=1,1	2,5
===	240	7	7	0	16	0	(out: xb=2 xt=2 xs=0)	LRU=1,0	V=1,1	7,5
===	299	9	9	0	11	0	(out: xb=5 xt=5 xs=0)	LRU=0,1	V=1,1	7,9

Lista blocchi uscenti

Situazione finale della cache

Si ricava quindi che il tempo medio di accesso alla cache e' pari a:
AMAT = $t_{hit}+t_{penalty}*m = t_{hit}+t_{penalty}* (N_{miss}/N_{ref})$ =5+60*16/22 ≅ 48.63 ns

Puo' essere interessante osservare cosa accade per la stessa traccia in ingresso, al variare delle caratteristiche della cache:

COMPITINO #2 di CALCOLATORI ELETTRONICI 1 del 03-12-03 **SOLUZIONE**

Per A=4, B=16 --> AMAT ≅ 59.54 e inoltre:

	X	XM	XT	XS	XB	HIT	BLOCCO USCENTE				BIT LRU	BIT VALID.	TAG
===	21	1	1	0	5	0					LRU=3,0,0,0	V=1,0,0,0	1,-,-,-
===	77	4	4	0	13	0					LRU=2,3,0,0	V=1,1,0,0	1,4,-,-
===	66	4	4	0	2	1					LRU=2,3,0,0	V=1,1,0,0	1,4,-,-
===	146	9	9	0	2	0					LRU=1,2,3,0	V=1,1,1,0	1,4,9,-
===	82	5	5	0	2	0					LRU=0,1,2,3	V=1,1,1,1	1,4,9,5
===	15	0	0	0	15	0	(out:	xb=1	xt=1	xs=0)	LRU=3,0,1,2	V=1,1,1,1	0,4,9,5
===	130	8	8	0	2	0	(out:	xb=4	xt=4	xs=0)	LRU=2,3,0,1	V=1,1,1,1	0,8,9,5
===	64	4	4	0	0	0	(out:	xb=9	xt=9	xs=0)	LRU=1,2,3,0	V=1,1,1,1	0,8,4,5
===	192	12	12	0	0	0	(out:	xb=5	xt=5	xs=0)	LRU=0,1,2,3	V=1,1,1,1	0,8,4,12
===	209	13	13	0	1	0	(out:	xb=0	xt=0	xs=0)	LRU=3,0,1,2	V=1,1,1,1	13,8,4,12
===	225	14	14	0	1	0	(out:	xb=8	xt=8	xs=0)	LRU=2,3,0,1	V=1,1,1,1	13,14,4,12
===	241	15	15	0	1	0	(out:	xb=4	xt=4	xs=0)	LRU=1,2,3,0	V=1,1,1,1	13,14,15,12
===	113	7	7	0	1	0	(out:	xb=12	xt=12	xs=0)	LRU=0,1,2,3	V=1,1,1,1	13,14,15,7
===	97	6	6	0	1	0	(out:	xb=13	xt=13	xs=0)	LRU=3,0,1,2	V=1,1,1,1	6,14,15,7
===	273	17	17	0	1	0	(out:	xb=14	xt=14	xs=0)	LRU=2,3,0,1	V=1,1,1,1	6,17,15,7
===	66	4	4	0	2	0	(out:	xb=15	xt=15	xs=0)	LRU=1,2,3,0	V=1,1,1,1	6,17,4,7
===	257	16	16	0	1	0	(out:	xb=7	xt=7	xs=0)	LRU=0,1,2,3	V=1,1,1,1	6,17,4,16
===	129	8	8	0	1	0	(out:	xb=6	xt=6	xs=0)	LRU=3,0,1,2	V=1,1,1,1	8,17,4,16
===	64	4	4	0	0	1					LRU=2,0,3,1	V=1,1,1,1	8,17,4,16
===	161	10	10	0	1	0	(out:	xb=17	xt=17	xs=0)	LRU=1,3,2,0	V=1,1,1,1	8,10,4,16
===	240	15	15	0	0	0	(out:	xb=16	xt=16	xs=0)	LRU=0,2,1,3	V=1,1,1,1	8,10,4,15
===	299	18	18	0	11	0	(out:	xb=8	xt=8	xs=0)	LRU=3,1,0,2	V=1,1,1,1	18,10,4,15

Per A=4, B=8 --> AMAT ≅ 59.54 e inoltre:

	X	XM	XT	XS	XB	HIT	BLOCCO USCENTE				BIT LRU	BIT VALID	TAG (SET0)	BIT LRU	BIT VALID	TAG (SET1)
===	21	2	1	0	5	0					LRU=3,0,0,0	V=1,0,0,0	1,-,-,-	LRU=0,0,0,0	V=0,0,0,0	-,-,-,-
===	77	9	4	1	13	0					LRU=3,0,0,0	V=1,0,0,0	1,-,-,-	LRU=3,0,0,0	V=1,0,0,0	4,-,-,-
===	66	8	4	0	2	0					LRU=2,3,0,0	V=1,1,0,0	1,4,-,-	LRU=3,0,0,0	V=1,0,0,0	4,-,-,-
===	146	18	9	0	2	0					LRU=1,2,3,0	V=1,1,1,0	1,4,9,-	LRU=3,0,0,0	V=1,0,0,0	4,-,-,-
===	82	10	5	0	2	0					LRU=0,1,2,3	V=1,1,1,1	1,4,9,5	LRU=3,0,0,0	V=1,0,0,0	4,-,-,-
===	15	1	0	1	15	0					LRU=0,1,2,3	V=1,1,1,1	1,4,9,5	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	130	16	8	0	2	0	(out:	xb=2	xt=1	xs=0)	LRU=3,0,1,2	V=1,1,1,1	8,4,9,5	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	64	8	4	0	0	1					LRU=2,3,0,1	V=1,1,1,1	8,4,9,5	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	192	24	12	0	0	0	(out:	xb=18	xt=9	xs=0)	LRU=1,2,3,0	V=1,1,1,1	8,4,12,5	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	209	26	13	0	1	0	(out:	xb=10	xt=5	xs=0)	LRU=0,1,2,3	V=1,1,1,1	8,4,12,13	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	225	28	14	0	1	0	(out:	xb=16	xt=8	xs=0)	LRU=3,0,1,2	V=1,1,1,1	14,4,12,13	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	241	30	15	0	1	0	(out:	xb=8	xt=4	xs=0)	LRU=2,3,0,1	V=1,1,1,1	14,15,12,13	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	113	14	7	0	1	0	(out:	xb=24	xt=12	xs=0)	LRU=1,2,3,0	V=1,1,1,1	14,15,7,13	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	97	12	6	0	1	0	(out:	xb=26	xt=13	xs=0)	LRU=0,1,2,3	V=1,1,1,1	14,15,7,6	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	273	34	17	0	1	0	(out:	xb=28	xt=14	xs=0)	LRU=3,0,1,2	V=1,1,1,1	17,15,7,6	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	66	8	4	0	2	0	(out:	xb=30	xt=15	xs=0)	LRU=2,3,0,1	V=1,1,1,1	17,4,7,6	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	257	32	16	0	1	0	(out:	xb=14	xt=7	xs=0)	LRU=1,2,3,0	V=1,1,1,1	17,4,16,6	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	129	16	8	0	1	0	(out:	xb=12	xt=6	xs=0)	LRU=0,1,2,3	V=1,1,1,1	17,4,16,8	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	64	8	4	0	0	1					LRU=0,3,1,2	V=1,1,1,1	17,4,16,8	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	161	20	10	0	1	0	(out:	xb=34	xt=17	xs=0)	LRU=3,2,0,1	V=1,1,1,1	10,4,16,8	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	240	30	15	0	0	0	(out:	xb=32	xt=16	xs=0)	LRU=2,1,3,0	V=1,1,1,1	10,4,15,8	LRU=2,3,0,0	V=1,1,0,0	4,0,-,-
===	299	37	18	1	11	0					LRU=2,1,3,0	V=1,1,1,1	10,4,15,8	LRU=1,2,3,0	V=1,1,1,0	4,0,18,-

Per A=1, B=32 --> AMAT ≅ 54.09 e inoltre:

	X	XM	XT	XS	XB	HIT	BLOCCO USCENTE				BIT LRU	BIT VALID	TAG (SET0)	BIT LRU	BIT VALID	TAG (SET1)
===	21	0	0	0	21	0					LRU=0	V=1	0	LRU=0	V=0	-
===	77	2	1	0	13	0	(out:	xb=0	xt=0	xs=0)	LRU=0	V=1	1	LRU=0	V=0	-
===	66	2	1	0	2	1					LRU=0	V=1	1	LRU=0	V=0	-
===	146	4	2	0	18	0	(out:	xb=2	xt=1	xs=0)	LRU=0	V=1	2	LRU=0	V=0	-
===	82	2	1	0	18	0	(out:	xb=4	xt=2	xs=0)	LRU=0	V=1	1	LRU=0	V=0	-
===	15	0	0	0	15	0	(out:	xb=2	xt=1	xs=0)	LRU=0	V=1	0	LRU=0	V=0	-
===	130	4	2	0	2	0	(out:	xb=0	xt=0	xs=0)	LRU=0	V=1	2	LRU=0	V=0	-
===	64	2	1	0	0	0	(out:	xb=4	xt=2	xs=0)	LRU=0	V=1	1	LRU=0	V=0	-
===	192	6	3	0	0	0	(out:	xb=2	xt=1	xs=0)	LRU=0	V=1	3	LRU=0	V=0	-
===	209	6	3	0	17	1					LRU=0	V=1	3	LRU=0	V=0	-
===	225	7	3	1	33	0					LRU=0	V=1	3	LRU=0	V=1	3
===	241	7	3	1	49	1					LRU=0	V=1	3	LRU=0	V=1	3
===	113	3	1	1	49	0	(out:	xb=7	xt=3	xs=1)	LRU=0	V=1	3	LRU=0	V=1	1
===	97	3	1	1	33	1					LRU=0	V=1	3	LRU=0	V=1	1
===	273	8	4	0	17	0	(out:	xb=6	xt=3	xs=0)	LRU=0	V=1	4	LRU=0	V=1	1
===	66	2	1	0	2	0	(out:	xb=8	xt=4	xs=0)	LRU=0	V=1	1	LRU=0	V=1	1
===	257	8	4	0	1	0	(out:	xb=2	xt=1	xs=0)	LRU=0	V=1	4	LRU=0	V=1	1
===	129	4	2	0	1	0	(out:	xb=8	xt=4	xs=0)	LRU=0	V=1	2	LRU=0	V=1	1
===	64	2	1	0	0	0	(out:	xb=4	xt=2	xs=0)	LRU=0	V=1	1	LRU=0	V=1	1
===	161	5	2	1	33	0	(out:	xb=3	xt=1	xs=1)	LRU=0	V=1	1	LRU=0	V=1	2
===	240	7	3	1	48	0	(out:	xb=5	xt=2	xs=1)	LRU=0	V=1	1	LRU=0	V=1	3
===	299	9	4	1	43	0	(out:	xb=7	xt=3	xs=1)	LRU=0	V=1	1	LRU=0	V=1	4