

[Versione vista durante l'esercitazione del 17/12/2015]

Si consideri il seguente programma assembly in cui alla partenza $\$2=0x100$, $\$4=4$, $\$10=10$:

```

    add    $1,    $0,    $0
    add    $5,    $0,    $0
L1:  lw     $6,    0($2)
    add    $1,    $1,    $6
    add    $2,    $2,    $4
    addi   $5,    $5,    1
    bne   $5,    $10,   L1
    div   $1,    $10
    mflo  $3
L2:  lw     $6,   -4($2)
    div   $6,    $3
    mflo  $6
    sw    $6,   -4($2)
    sub   $2,    $2,    $4
    addi  $5,    $5,   -1
    bne   $5,    $0,    L2

```

Si supponga che tale programma vada in esecuzione su un calcolatore con processore MIPS a 32-bit (senza pipeline) con frequenza di clock pari a 2GHz. Le istruzioni aritmetico/logiche/jump richiedono 1 ciclo, i salti (branch) 3 cicli e le operazioni di load/store 1 ciclo oltre ai cicli necessari per accedere alla gerarchia di memoria.

Si calcoli il tempo di esecuzione di questo programma nei seguenti casi:

- nel caso in cui la gerarchia di memoria sia costituita solo dalla memoria principale e questa abbia un tempo di accesso di 50 ns;
- nel caso in cui nella gerarchia di memoria sia interposto uno write-buffer fra processore e memoria principale. Lo write-buffer consente di memorizzare 16 parole e il tempo di accesso allo write-buffer corrisponde a un ciclo;
- nel caso in cui nella gerarchia di memoria sia interposta una split-cache (al posto dello write-buffer) fra processore e memoria principale. La split-cache presenta un tempo di accesso corrispondente a un ciclo. Per la cache istruzioni si assuma un miss rate dell' 1% , mentre la cache dati e' set associative a 2 vie di dimensioni 1KB e con blocco di dimensioni 8 Byte.

Riepilogo del significato delle istruzioni

Instruction	Example	Meaning	Comments
add	add \$1, \$2, \$3	$\$1 = \$2 + \$3$	3 operands; exception possible
subtract	sub \$1, \$2, \$3	$\$1 = \$2 - \$3$	3 operands; exception possible
add immediate	addi \$1, \$2, 100	$\$1 = \$2 + 100$	+ constant; exception possible
division	div \$1, \$2	Hi = $\$1 \% \2 , Lo = $\$1 / \2	Signed division
move from Lo	mflo \$1	$\$1 = Lo$	Create copy of Lo
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
load upper immediate	lui \$1, 100	$\$1 = 100 * 256$	Load constant in upper 16bits
branch on not equal	bne \$1, \$2, 100	if $(\$1 \neq \$2)$ go to PC+4+100	Not equal test; PC relative

Register Usage

Name	Register Number	Usage
\$zero	0	the constant value 0