COMPITINO #3 di CALCOLATORI ELETTRONICI del 13-12-01 **MATRICOLA**_____

COGNOME_____

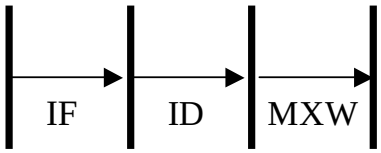
NOME_____

1) Si consideri il seguente programma assembly in cui alla partenza $3 vale $2+396:

```
L1: lw   $1, 0($2)
    addi $1,   $1, 1
    sw   $1, 0($2)
    addi $2, $2, 4
    sub  $4, $3, $2
    bne  $4, $0, L1
```

Si supponga che tale programma vada in esecuzione su un calcolatore con processore R3000 a frequenza di clock pari a 2GHz e che tutti gli accessi in cache generino hit. Si assuma inizialmente che sia possibile la scrittura e la lettura di un registro nello stesso ciclo di clock, che un salto provochi lo svuotamento della pipeline e che siano disabilitati i circuiti per la propagazione (forwarding). Calcolare il tempo di esecuzione in questa situazione.
Successivamente calcolare il tempo di esecuzione supponendo che la rete di propagazione sia attiva e che l'istruzione di salto assuma una predizione fissa di 'non-salto'. Infine, provare a riorganizzare il codice per ridurre al minimo gli stalli e calcolare il nuovo tempo di esecuzione.

2) Si consideri la pipeline:



MXW= 'accesso-Memoria' oppure 'esecuzione (eXecute)' oppure 'scrittura nei registri (Write)'. In questo tipo di pipeline, tutte le dipendenze tra i dati sono tra il registo scritto in MXW dalla istruzione i e quelli letti in ID dalla istruzione i+1: la probablita' di conflitto (hazard) e' 1/p.
Se il risultato di una istruzione venisse scritto in un quarto stadio WB (e MXW diventa MX), invece che essere scritto in MXW, si otterrebbe una diminuzione del ciclo di clock T_C pari a d. In tal caso la probabilita' di una dipendenza tra l'istruzione i e $i+2$ sarebbe pari a p^{-2} e i conflitti fra l'istruzione i e $i+1$ si genererebbero con probabilita' p^{-1}. Si calcoli il limite inferiore su d affinche' la modifica si utile.
Ipotesi semplificative: anche l'istruzione di tipo 'branch' viene risolta nello stadio WB, ogni risultato viene usato una sola volta e non c'e' propagazione (forwarding).

Riepilogo del significato delle istruzioni

Instruction	Example	Meaning	Comments
subtract	`sub $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 not equal	`bne $1,$2,100`	if ($1 != $2) go to PC+4+100	Not equal test; PC relative