COMPITO di CALCOLATORI ELETTRONICI del 25-03-02 **MATRICOLA**_____

COGNOME_____

NOME_____

1) Si consideri il seguente programma:

```
int A[4000], B[4000];
int Loop4(int n, int *X, int *Y) {
   int i, Q;
   Q = 0;
   for (i = 1, i < n; ++i) {
      Q = Q + X[i] * Y[i-1];
   }
   return(Q);
}
main()
{
   int n1, n2, k, i, V;
   n1 = 1000; n2 = 100; V = 0;
   for (k = 0; k < n1; ++k) {
      A[k] = 444; B[k] = 444;
   }
   for (i = 0; i < n2; ++i) {
      V = V + Loop4(n1, A, B);
   }
}
```

a) Scrivere il corrispondente codice assembly MIPS utilizzando solo e unicamente istruzioni dalla tabella riportata qua sotto.
b) Calcolare il tempo di esecuzione del codice tradotto al punto precedente su un processore con frequenza di clock pari a 2 GHz, assumendo i seguenti valori per il CPI di ciascuna categoria di istruzioni: aritmetico-logiche 1, branch 3, load-store 5.

MIPS instructions

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
subtract immediate	`subi $1,$2,100`	$1 = $2 - 100	- constant; exception possible
multiplication	`mult $1, $2`	Hi,Lo= $1 x $2	64-bit Signed Product ; result in Hi,Lo
division	`div $1, $2`	Hi= $1 % $2, Lo = $1 / $2	Signed division
move from Hi	`mfhi $1`	$1 = Hi	Create copy of Hi
move from Lo	`mflo $1`	$1 = Lo	Create copy of Lo
and	`and $1,$2,$3`	$1 = $2 & $3	3 register operands; Logical AND
or	`or $1,$2,$3`	$1 = $2 | $3	3 register operands; Logical OR
nor	`nor $1,$2,$3`	$1 = !($2 | $3)	3 register operands; Logical NOR
xor	`xor $1,$2,$3`	$1 = $2 ^ $3	3 register operands; Logical XOR
and immediate	`andi $1,$2,100`	$1 = $2 & 100	Logical AND register, constant
or immediate	`ori $1,$2,100`	$1 = $2 | 100	Logical OR register, constant
xor immediate	`xori $1,$2,100`	$1 = $2 ^ 100	Logical XOR register, constant
shift left logical	`sll $1,$2,10`	$1 = $2 << 10	Shift left by constant
shift right logical	`srl $1,$2,10`	$1 = $2 >> 10	Shift right by constant
load word	`lw $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
load byte	`lb $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
load byte unsigned	`lbu $1,100($2)`	$1 = Memory[$2+100]	Data from mem. to reg.; no sign extension
store word	`sw $1,100($2)`	Memory[$2+100] = $1	Data from register to memory
store byte	`sb $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 equal	`beq $1,$2,100`	if ($1 == $2) go to PC+4+100	Equal test; PC relative branch
branch on not equal	`bne $1,$2,100`	if ($1 != $2) go to PC+4+100	Not 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
set on less than immediate	`slti $1,$2,100`	if ($2 < 100) $1 = 1; else $1 = 0	Compare < constant; 2`s complement
set on less than unsigned	`sltu $1,$2,$3`	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; natural number
set on less than imm. unsigned	`Sltiu $1,$2,100`	if ($2 < 100) $1 = 1; else $1 = 0	Compare constant; natural number
jump	`j 10000`	go to 10000	Jump to target address
jump register	`jr $31`	go to $31	For switch, procedure return
jump and link	`jal 10000`	$31 = PC + 4;go to 10000	For procedure call

Register Usage

Name	Register Number	Usage
$zero	0	the constant value 0
$s0-$s7	16-23	Saved
$t0-$t9	8-15,24-25	Temporaries
$a0-$a3	4-7	Arguments
$v0-$v1	2-3	Results
$fp, $sp, $ra, $gp	30,29,31,28	Frame pointer, stack pointer, return address, global pointer

2) Si consideri una cache 4-way set associative, di dimensione pari a 256 byte e rimpiazzamento LRU che processa la seguente sequenza di accessi a indirizzi di byte, la parola e' pari a 32 bit e i blocchi sono di 16 parole. Calcolare il tempo di medio di accesso alla memoria (AMAT) nell'ipotesi che t_{hit}=8ns e $t_{penalty}$=60ns. Riportare inoltre il contenuto dei campi tag al termine degli accessi.
Sequenza: 21, 77, 66, 146, 82, 15, 130, 64, 192, 209, 225, 241, 113, 97, 273, 66, 257, 129, 64, 161, 240, 299.

3) Si supponga di aver migliorato una macchina in modo da far si' che esegua tutte le operazioni in virgola mobile 6 volte piu' velocemente e si analizzi come si comporta lo speedup quando la modifica viene introdotta. Se prima del miglioramento il tempo di esecuzione di un dato benchmark e' di 10 secondi, quale sara' lo speedup nel caso in cui meta' dei 10 secondi sono spesi per l'esecuzione delle operazioni in virgola mobile?

4) Si descriva, passo-passo, la sequenza di operazioni che si verificano in un calcolatore in conseguenza di un interruzione da una periferica.