

Trovare il codice assembly MIPS corrispondente dei seguenti micro-benchmark (**utilizzando solo e unicamente istruzioni dalla tabella sottostante**), rispettando le convenzioni di utilizzazione dei registri dell'assembly MIPS (riportate in calce, per riferimento).

```
<BENCHMARK-1:>
<M1,M2,MR sono matrici di 'int'>

for (j = 0; j < 100; ++j) {
  for (k = 0; k < 100; ++k) {
    t = 0;
    for (i = 0; i < 100; ++i) {
      t += M1[j][i]*M2[i][k];
    }
    MR[j][k] = t;
  }
}

<BENCHMARK-2:>
<la funzione e' invocata con fibo(10)>

int fibo(int n)
{
  if (n <= 2) return (1);
  else return (fibo(n-1)+(fibo(n-2)));
}
```

Successivamente, calcolare il tempo di esecuzione di ciascuno dei due benchmark ipotizzando che tali benchmark vengano eseguiti:

- su un processore con frequenza di clock pari a 4 GHz, assumendo i seguenti valori per il CPI di ciascuna categoria di istruzioni: aritmetico-logiche-salti 1, branch 3, load-store 10.
- su un processore con frequenza di clock pari a 1 GHz, assumendo i seguenti valori per il CPI di ciascuna categoria di istruzioni: aritmetico-logiche-salti 1, branch 3, load-store 5.

Infine, si calcolino le medie aritmetica, geometrica e armonica dei tempi di esecuzione sulla macchina A e B, indicando quale delle due macchine (A o B) si ritiene migliore in base ai risultati ottenuti e motivando brevemente la scelta.

### 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 address	la \$1,var	\$1 = &var	Load variable address
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 Num.	Usage
\$zero	0	The constant value 0
\$\$0-\$\$7	16-23	Saved
\$\$0-\$\$9	8-15,24-25	Temporaries
\$\$a0-\$\$a3	4-7	Arguments

Name	Register Num.	Usage
\$\$v0-\$\$v1	2-3	Results
\$\$fp, \$\$sp	30,29	Frame pointer, stack pointer
\$\$ra, \$\$gp	31,28	return address, global pointer
\$\$k0-\$\$k1	26,27	Kernel usage

Name	Usage
\$\$f0, \$\$f1, ..., \$\$f31	Single precision floating point registers
\$\$f0, \$\$f2, ..., \$\$f30	Double precision floating point registers