**Roll No: …………………………. Name …………………………………………..**

**National Institute of Technology Calicut**

**Department of Computer Science & Engineering**

**CS2004 – First Mid Term Exam (Winter Semester 2012-‘13)**

**Max. Marks: 20 Time: 1 Hour**

1. A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? [ 1.5 marks]
2. Consider two different implementations, M1 and M2, of the same instruction set. M1 has a clock rate of 6 GHz, and M2 has a clock rate of 3 GHz. There are three classes of instructions (A, B, and C) in the instruction set. The average number of cycles for each instruction class on M1 and M2 is given in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| class | CPI on M1 | CPI on M2 | C1 usage | C2 usage |
| A | 2 | 1 | 40% | 50% |
| B | 3 | 2 | 40% | 25% |
| C | 5 | 2 | 20% | 25% |

The table also contains a summary of average proportion of instruction classes generated by two different compilers. C1 and C2 are compilers. Usage of compilers for appropriate classes is given above. Assume that each compiler uses the same number of instructions for a given program but that the instruction mix is as described in the table.

i) Using C1 on both M1 and M2, how much faster can M1 claim when compared to M2? [2 marks]

ii) Using C2, how much faster can M1 claim when compared to M2? [2 marks]

iii) If you purchase M1, which compiler would you use? If you purchase M2, which compiler would you use? [1 mark]

1. You are designing a system for a real time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary. How much energy do you save if you set the voltage and frequency to be half as much? [1 Mark]

1. Imagine an array A, which can contain a maxium of ‘k’ elements in it. Consider the following **Bound check** condition on the above array.

“Jump to *IndexOutOfBounds* if **i>=k** or if i is negative”

Assume that the variables k &i are assigned to $t2, $a1. Write the MIPS shortcut (in 2 instructions) for the above bound check? [1 Mark]

sltu $t0, $a1, $t2

beq $t0, $Zero, *IndexOutOfBounds*

1. Translate the given C code to MIPS. Assume that the variables f,g,h,i & j are assigned to $S0, $S1, $S2, $S3 & $S4 respectively. Assume the base address of the arrays A & B are in registers $S6 & $S7 respectively. [1.5 Marks]

f = A[ B[g] + 1 ]

add $t0, $s7, $s1

lw $t0, 0($t0)

add $t0, $t0, $s6

lw $s0, 4($t0)

1. Convert the following MIPS code to C! (Assume $s0, $s1 & $t0 corresponds to variable a, b & i) [1.5 Marks]

addi $t0, $ZERO, 0

beq $ZERO, $ZERO, TEST

LOOP: add $s0, $s0, $s1

addi $t0, $t0, 1

TEST: slti $t2, $t0, 10

bne $t2, $ZERO, LOOP

for(i=0; i<10; i++)

a+=b;

1. The given table shows bit patterns expressed in hexadecimal notation. Answer the following questions

|  |  |
| --- | --- |
| A | 0x00af8020 |
| B | 0x08010000 |

If this bit pattern is placed into the Instruction register, what MIPS instruction will be executed? [1.5 Marks]

0000 0000 1010 1111 1000 0000 0010 0000

000000 00101 01111 10000 00000 100000

add $S0, $a1, $t7

0000 1000 0000 0001 0000 0000 0000 0000

000010 000000000100000000000000000

j 10000

1. Write the addressing mode and instruction format of the following instruction
2. addi $a1, $a1, 100 [2 Marks]

Immediate addressing, I type

1. beq $t1,$t5, label

PC relative addressing, I type

1. add $s2 $s1, $t2

Register addressing, R type

1. jal swap

Direct jump addressing, J type

1. Translate the given function(pointer version of swap) into MIPS level!

void swap (int \*p) [1.5 Marks]

{

int temp;

temp = \*p;

\*p = \* (p+1);

\*(p+1) = temp;

}

swap:

lw $t0,0($a0)

lw $t1,4($a0)

sw $t1,0($a0)

sw $t0,4($a0)

jr $ra

1. Translate the following C function code to MIPS! [2 Marks]

int nthfib( int a, int b, int n)

{

if (n==0)

return b;

else

return nthfib( a+b, a, n-1)

}

fi b\_iter:

addi $sp, $sp, –16

sw $ra, 12($sp)

sw $s0, 8($sp)

sw $s1, 4($sp)

sw $s2, 0($sp)

add $s0, $a0, $0

add $s1, $a1, $0

add $s2, $a2, $0

add $v0, $s1, $0,

bne $s2, $0, exit

add $a0, $s0, $s1

add $a1, $s0, $0

add $a2, $s2, –1

jal fi b\_iter

exit:

lw $s2, 0($sp)

lw $s1, 4($sp)

lw $s0, 8($sp)

lw $ra, 12($sp)

addi $sp, $sp, 16

jr $ra

1. Translate the below ARM assembly code to MIPS!

ROR r1, r2, #4 ; r1= r2[3:0] concatenated with r2[31:4]

Assume that ARM registers r1, r2 hold the same values as MIPS registers $S1,$S2 respectively [1.5 Marks]

sll $s1, $s2, 28

srl $s2, $s2, 4

or $s1, $s1, $s2