CSc 252: Computer Organization

HW 6

due at 5pm, two days **before** the test
Turn in through GradeScope

Policy Reminders

- You must turn in a single PDF to GradeScope. Other file formats will not be accepted. (Sorry, this is necessary for the sanity of the TAs!)
- You are allowed to work with other students on this homework, as we will not be grading it for correctness. However, **each student must turn in their own copy of the homework.**
- **Show your work for all problems.** While we won't be grading for correctness, you will not receive full credit unless you show your work.

  After all, showing your work is required on the test - and homeworks are intended to help you practice for the test!

Required Problems:

6.1(e), 6.2(e)

Allowable Instructions

When writing MIPS assembly, the only instructions that you are allowed to use (so far) are:

- add, addi, sub, addu, addiu, subu
- beq, bne, j, jal, jr
- slt, slti
- and, andi, or, ori, nor, nori, xor, xori
- sll, srl, sra
- lw, lh, lb, sw, sh, sb
- la
- syscall

While MIPS has many other useful instructions (and the assembler recognizes many pseudo-instructions), do not use them! We want you to learn the **fundamentals** of how assembly language works - you can use fancy tricks after this class is over.

Problem 6.1 – Pipelining

6.1(a) - Turn in this one

```
addi $t0, $sp, 16
addi $t1, $zero, 4
addi $t3, $zero, 0
lw   $t4, 0($t0)
add  $t2, $t2, $t4
I1:
addi $t0, $t0, 4
addi $t3, $t3, 1
addi $t5, $a0, 0
slt  $t7, $t1, $t3
beq  $t7, $zero, I1
addi $t0, $sp, 16
addi $t1, $zero, 4
addi $t3, $zero, 0
```

Fill out the following table to draw the pipeline diagram for the above MIPS program on a pipelined machine that forwards when a data hazard occurs. Do not forget to draw the arrows

	Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1																	
2																	
3																	
4																	
5																	
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	

•••

6.1(b)

For a given program:

- 25% of instructions are loads from memory, and 20% of these loads cause hazards
- 15% of instructions are conditional jumps
- 55% are mispredicted
- 4% of instructions are function calls

Compute the CPI for this program.

6.2(b) - Turn in this one

For a given program:
40% of the branches are taken
75% of the values loaded are immediately used

Instruction type	Percentage of instruction mix
Alu operations (add, sub, . . .)	**45 %**
Load from memory operations	**10 %**
Store from memory operations	**10 %**
Conditional branches	**25 %**
jal instructions	**5 %**
jr instructions	**5 %**

Compute the CPI for this program.

Problem 6.2 - Encoding Floating Point

Show the IEEE754 binary representation for the floating point number in:

- single precision
- double precision

6.2(a)

10.0_{ten}

6.2(b)

1024.5_{ten}

6.2(c)

$-42\frac{5}{16}_{ten}$

6.2(d)

$7\frac{189}{256}_{ten}$

6.2(e) - Turn in this one

3952_{ten}

HINT: With large even numbers, try dividing by 2 (and then again, and then again) to make the number easier to convert to binary.

EXAMPLES

Example: Problem 6.2(a) - 10.0_{ten}

The first part of the conversion is common; both single and double precision use the same values.

Number is positive, so $s = 0$.

$10_{ten} = 1010_{bin} = 1.010_{bin} \cdot 2^3$

Mantissa: 1.0100 0000 ...

Fraction: 0100 0000 ...

Single Precision

Exponent = 3; with bias: $3 + 127 = 130_{ten} = 10000010_{bin}$

s eee eeee e fff ffff ffff ffff ffff ffff
0 100 0001 0 010 0000 0000 0000 0000 0000

Double Precision

Exponent = 3; with bias: $3 + 1023 = 1026_{ten} = 10000000010_{bin}$

s eee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff
0 100 0000 0010 0100 0000 ...

Example: Problem 6.2(b) - 1024.5_{ten}

The first part of the conversion is common; both single and double precision use the same values.

Number is positive, so $s = 0$.

$1024.5 = 1024 + .5 = 2^{10} + 2^{-1}$
$1024.5_{ten} = 10000000000.1_{bin} = 1.00000000001_{bin} \cdot 2^{10}$

Mantissa: 1.00 0000 0000 1…

Fraction: 0000 0000 0010 ...

Single Precision

Exponent = 10; with bias: $10 + 127 = 137_{ten} = 10001001_{bin}$

s eee eeee e fff ffff ffff ffff ffff ffff
0 100 0100 1 000 0000 0001 0000 0000 0000

Double Precision

Exponent = 10; with bias: $10 + 1023 = 1033_{ten} = 10000001001_{bin}$

s eee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff
0 100 0000 1001 0000 0000 0010 0000 ...

Example: Problem 6.2(c) - -42.3125_{ten}

The first part of the conversion is common; both single and double precision use the same values.

Number is negative, so $s = 1$.

$42.3125 = 32 + 8 + 2 + .25 + .0625 = 2^5 + 2^3 + 2^1 + 2^{-2} + 2^{-4}$
$42.3125_{ten} = 101010.0101_{bin} = 1.010100101_{bin} \cdot 2^5$

Mantissa: 1.010100101...

Fraction: 0101 0010 1...

Single Precision

Exponent = 5; with bias: $5 + 127 = 132_{ten} = 10000100_{bin}$

s eee eeee e fff ffff ffff ffff ffff ffff
1 100 0010 0 010 1001 0100 0000 0000 0000

Double Precision

Exponent = 5; with bias: $5 + 1023 = 1028_{ten} = 10000000100_{bin}$

s eee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff
1 100 0000 0100 0101 0010 1000 0000 ...

Example: Problem 6.2(d) - $7\frac{189}{256_{ten}}$

The first part of the conversion is common; both single and double precision use the same values.

Number is positive, so $s = 0$.

$189 = 128 + 32 + 16 + 8 + 4 + 1$

$$7\frac{189}{256}_{ten} = 4 + 2 + 1 + \frac{128 + 32 + 16 + 8 + 4 + 1}{2^8}$$

$$7\frac{189}{256}_{ten} = 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-8}$$

$123.8125_{ten} = 0111.10111101_{bin} = 1.1110111101_{bin} \cdot 2^2$

Mantissa: 1.1110111101...

Fraction: 1110 1111 01...

Single Precision

Exponent = 2; with bias: $2 + 127 = 129_{ten} = 10000001_{bin}$

s eee eeee e fff ffff ffff ffff ffff ffff
0 100 0000 1 111 0111 1010 0000 0000 0000

Double Precision

Exponent = 2; with bias: $2 + 1023 = 1025_{ten} = 10000000001_{bin}$

s eee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff
0 100 0000 0001 1110 1111 0100 0000 ...