January 8, 2009

COMPUTER ENGINEERING DEPARTMENT

ICS 233

COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE

Major Exam II

First Semester (081)

Time: 1:00-3:30 PM

Student Name : _KEY____

Student ID. : ____

Question	Max Points	Score
Q1	20	
Q2	16	
Q3	16	
Q4	16	
Q5	8	
Q6	8	
Q7	16	
Total	100	

Dr. Aiman El-Maleh

[20 Points]

(Q1) Given below a summary of syscall services:

Service	$v0	Arguments / Result
Print Integer	1	$a0 = integer value to print
Read Integer	5	$v0 = integer read
Exit Program	10	

(i) Determine the output produced by the following program given that the program inputs are 7 and 4.

```
        .text
        .globl main
main:
        li $v0, 5
        syscall
        move $t0, $v0
        li $v0, 5
        syscall
        move $a1, $v0
        move $a0, $t0
        jal Proc1
        move $a0, $v0
        li $v0, 1
        syscall
        li $v0, 10
        syscall
Proc1:
        bne $a0, $a1, Skip
        move $v0, $a0
        jr $ra
Skip:
        addi $sp, $sp, -8
        sw $a0, ($sp)
        sw $ra, 4($sp)
        addi $a0, $a0, -1
        jal Proc1
        lw $t0, ($sp)
        lw $ra, 4($sp)
        addi $sp, $sp, 8
        mul $v0, $v0, $t0
        jr $ra
```

The program computes 7x6x5x4=840 and then displays it.

(ii) Determine the output produced by the following program given that the program input is 987.

```
        .text
        .globl main
main:
        li $v0, 5
        syscall
        move $a0, $v0
        jal Proc2
        move $a0, $v0
        li $v0, 1
        syscall
        li $v0, 10
        syscall

Proc2:
        li $t0, 10
        move $t1, $a0
Next:
        xor $t3, $t3, $t3
Again:
        divu $t1, $t0
        mflo $t1
        mfhi $t2
        addu $t3, $t3, $t2
        bnez $t1, Again
        move $t1, $t3
        bge $t1, $t0, Next
        move $v0, $t1
        jr $ra
```

The program adds individual digits of the entered number to get another number and repeats the process until the result is a single digit. Thus, the program will display 6.

[16 Points]

(Q2) Given that **Multiplicand=1010** and **Multiplier=1111**.

(i) Using the **refined unsigned multiplication** hardware, show the **unsigned** multiplication of **Multiplicand** by **Multiplier**. The result of the multiplication should be an 8 bit **unsigned** number in HI and LO registers. Show the steps of your work.

Iteration		**Multiplicand**	**Carry**	**Product = HI,LO**
0	Initialize (LO = Multiplier)	1010		0000 1111
1	LO[0] = 1 => ADD		0	1010 1111
	Shift Product = (HI, LO) right 1 bit	1010		0101 0111
2	LO[0] = 1 => ADD		0	1111 0111
	Shift Product = (HI, LO) right 1 bit	1010		0111 1011
3	LO[0] = 1 => ADD		1	0001 1011
	Shift Product = (HI, LO) right 1 bit	1010		1000 1101
4	LO[0] = 1 => ADD		1	0010 1101
	Shift Product = (HI, LO) right 1 bit			**1001 0110**

(ii) Using the **refined signed multiplication** hardware, show the **signed** multiplication of **Multiplicand** by **Multiplier**. The result of the multiplication should be an 8 bit **signed** number in HI and LO registers. Show the steps of your work.

Iteration		**Multiplicand**	**Sign**	**Product = HI,LO**
0	Initialize (LO = Multiplier)	1010		0000 1111
1	LO[0] = 1 => ADD		1	1010 1111
	Shift Product = (HI, LO) right 1 bit	1010		1101 0111
2	LO[0] = 1 => ADD		1	0111 0111
	Shift Product = (HI, LO) right 1 bit	1010		1011 1011
3	LO[0] = 1 => ADD		1	0101 1011
	Shift Product = (HI, LO) right 1 bit	1010		1010 1101
4	LO[0] = 1 => SUB (ADD 2's comp.)		0	0000 1101
	Shift Product = (HI, LO) right 1 bit			**0000 0110**

[16 Points]

(Q3) Given that **Dividend=1001** and **Divisor=0101**.

(i) Using the **refined unsigned division** hardware, show the **unsigned** division of **Dividend** by **Divisor**. The result of division should be stored in the Remainder and Quotient registers. Show the steps of your work.

Iteration		**Remainder**	**Quotient**	**Divisor**	**Difference**
0	Initialize	0000	1001	0101	
1	1: SLL, Difference	0001	0010	0101	1100
	2: Diff < 0 => Do Nothing				
2	1: SLL, Difference	0010	0100	0101	1101
	2: Diff < 0 => Do Nothing				
3	1: SLL, Difference	0100	1000	0101	1111
	2: Diff < 0 => Do Nothing				
4	1: SLL, Difference	1001	0000	0101	0100
	2: Rem = Diff, set lsb Quotient	**0100**	**0001**		

(ii) Using the **refined unsigned division** hardware, show the **signed** division of **Dividend** by **Divisor**. The result of division should be stored in the Remainder and Quotient registers. Show the steps of your work.

First, we convert the dividend into a positive number by taking its 2's complement. Thus, the dividend becomes 0111. Then, we perform unsigned division as shown below.

Iteration		**Remainder**	**Quotient**	**Divisor**	**Difference**
0	Initialize	0000	0111	0101	
1	1: SLL, Difference	0000	1110	0101	1011
	2: Diff < 0 => Do Nothing				
2	1: SLL, Difference	0001	1100	0101	1100
	2: Diff < 0 => Do Nothing				
3	1: SLL, Difference	0011	1000	0101	1110
	2: Diff < 0 => Do Nothing				
4	1: SLL, Difference	0111	0000	0101	0010
	2: Rem = Diff, set lsb Quotient	**0010**	**0001**		

Since Quotient sign should be negative we take its 2's complement. Thus, Quotient=1111.
Also, since remainder sign is negative, we take its 2's complement. Thus, Remainder=1110.

[16 Points]

(Q4)

(i) What is the decimal value of the following single-precision floating-point number?

`0100 0011 0110 1001 1000 0100 0000 0000.`

$= + (1.1101001100001000...0)_2 * 2^{(134-127)} = + (1.1101001100001000\ldots0)_2 * 2^7$
$= + (11101001.100001000\ldots.0)_2$
$= + 233.515625$

(ii) Show the single-precision floating-point binary representation for: **555.9375**.

$555.9375=(1000101011.1111)_2 = (1.0001010111111)_2 * 2^9$
Exp. = 9 +127=136
Single precision binary representation:

`0100 0100 0000 1010 1111 1100 0000 0000`

(iii) Perform the following floating-point operation rounding the result to the **nearest even**. Perform the operation using **guard**, **round** and **sticky** bits.

```
    1100 1110 0000 0000 0000 0000 0100 0000
+   0101 0010 0000 0000 1000 0000 0000 0000
```

We add three bits for each operand representing G, R, S bits as follows.

```
     1.000 0000 1000 0000 0000 0000 000 x 2^37
-    1.000 0000 0000 0000 0100 0000 000 x 2^29
----------------------------------------------
     1.000 0000 1000 0000 0000 0000 000 x 2^37
-    0.000 0000 1000 0000 0000 0000 010 x 2^37 (align)
----------------------------------------------
    01.000 0000 1000 0000 0000 0000 000 x 2^37
+   11.111 1111 0111 1111 1111 1111 110 x 2^37 (2's complement)
----------------------------------------------
    00.111 1111 1111 1111 1111 1111 110 x 2^37
= +  0.111 1111 1111 1111 1111 1111 110 x 2^37
= +  1.111 1111 1111 1111 1111 1111 100 x 2^36 (normalize)
= + 10.000 0000 0000 0000 0000 0000     x 2^36 (round)
= +  1.000 0000 0000 0000 0000 0000     x 2^37 (renormalize)
```

[8 Points]

(Q5)

(i) Fill the following table by placing a check mark (✓) indicating the impact of each listed factor on the Instruction Count (I-Count), CPI and Cycle time.

	I-Count	CPI	Cycle
Compiler	✓	✓	
Instruction Set Architecture (ISA)	✓	✓	✓
Organization		✓	✓
Technology			✓

(ii) List three problems in using MIPS as a performance metric.

Three problems using MIPS as a performance metric:

1. Does not take into account the capability of instructions. Cannot use MIPS to compare computers with different instruction sets because the instruction count will differ.
2. MIPS varies between programs on the same computer. A computer cannot have a single MIPS rating for all programs.
3. MIPS can vary inversely with performance. A higher MIPS rating does not always mean better performance.

[8 Points]

(Q6) Suppose that a program runs in 150 seconds on a machine, with ALU operations responsible for 40 seconds of this time, multiply operations responsible for 50 seconds of this time and divide operations responsible for 40 seconds of this time. The remaining time is taken by the remaining operations. Suppose that a new implementation of the machine has improved the execution time of the ALU by a factor of 2, the multiplier by a factor of 1.5 and the divider by a factor of 1.6. Determine the new execution time and the speedup of the program based on the new implementation.

Execution time of new implementation = 40/2 + 50/1.5 + 40/1.6 + 20
=20+33.33+25+20=98.33 seconds

Speedup = 150/98.33 = 1.525

[16 Points]

(Q7) Given the following instruction mix of a program on a RISC processor:

Class	CPI	Frequency
ALU	2	40%
Branch	2	25%
Jump	1	15%
Load	4	10%
Store	3	10%

(i) What is the average CPI?

Average CPI=2*0.4 + 2*0.25+1*0.15+4*0.10+3*0.10=2.15

(ii) Assuming that the processor has a clock rate of 2 GHz, determine MIPS.

MIPS= Clock Rate / (CPI*10^6)=2* 10^9/ 2.15*10^6=930.23

(iii) What is the percent of time used by each instruction class?

Class	Percentage of Time
ALU	2*0.4/2.15=0.8/2.15=37.21%
Branch	2*0.25/2.15=0.5/2.15=23.26%
Jump	1*0.15/2.15=0.15/2.15=6.98%
Load	4*0.10/2.15=0.4/2.15=18.60%
Store	3*0.10/2.15=0.3/2.15=13.95%

(iv) How much faster would the program run if load time is reduced to 3 cycles, and two ALU instructions could be executed at once, assuming that the cycle time has increased by 5% and the instruction count has increased by 10%?

New Average CPI=**1***0.4 + 2*0.25+1*0.15+**3***0.10+3*0.10=1.65

Speedup = Old Execution Time / New Execution Time
= IC * 2.15 * Cycle Time / 1.1* IC * 1.65 * 1.05 * Cycle Time
= 2.15 / 1.1*1.65*1.05 = 2.15/1.90575=1.128