ICS 233 - Computer Architecture & Assembly Language

Exam II – Fall 2007

Saturday, December 8, 2007
7:00 pm – 9:00 pm

Computer Engineering Department
College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals

Student Name: SOLUTION

Student ID: ______________________________

Q1	/ 15	Q2	/ 15
Q3	/ 25	Q4	/ 20
Q5	/ 25		
Total	/ 100		

Important Reminder on Academic Honesty

Using unauthorized information or notes on an exam, peeking at others work, or altering graded exams to claim more credit are severe violations of academic honesty. Detected cases will receive a failing grade in the course.

Q1. (10 pts) Using the refined multiplication hardware, show the **unsigned** multiplication of:

Multiplicand = **01101101** by Multiplier = **10110110**

The result of the multiplication should be a 16 bit unsigned number in HI and LO registers. Eight iterations are required. Show your steps.

Iteration	Multiplicand	Carry	HI	LO
0: Initialize	01101101		00000000	10110110
1: Shift right			00000000	01011011
2: LO[0] = 1	ADD	0	01101101	01011011
2: Shift right			00110110	10101101
3: LO[0] = 1	ADD	0	10100011	10101101
3: Shift right			01010001	11010110
4: Shift right			00101000	11101011
5: LO[0] = 1	ADD	0	10010101	11101011
5: Shift right			01001010	11110101
6: LO[0] = 1	ADD	0	10110111	11110101
6: Shift right			01011011	11111010
7: Shift right			00101101	11111101
8: LO[0] = 1	ADD	0	10011010	11111101
8: Shift right			01001101	01111110

Check:

Multiplicand = 01101101_2 = 109

Multiplier = 10110110_2 = 182

Product = 19838 (decimal) = 01001101 01111110 (binary)

b) (5 pts) What is the decimal value of the following floating-point number?

1 10001101 10101000000000000000000 (binary)

Sign = negative

Exponent value = 10001101_2 – Bias = 141 – 127 = 14

Decimal Value = $-1.10101_2 \times 2^{14}$ = -1.65625×2^{14} = -27136

Q2. (10 pts) Using the refined division hardware, show the **unsigned** division of:

Dividend = **11011001** by Divisor = **00001010**

The result of the division should be stored in the Remainder and Quotient registers. Eight iterations are required. Show your steps.

Iteration	Remainder	Quotient	Divisor	Difference
0: Initialize	00000000	11011001	00001010	
1: SLL, Diff	00000001	10110010	00001010	< 0
2: SLL, Diff	00000011	01100100	00001010	< 0
3: SLL, Diff	00000110	11001000	00001010	< 0
4: SLL, Diff	00001101	10010000	00001010	00000011
4: Rem = Diff	00000011	10010001		
5: SLL, Diff	00000111	00100010	00001010	< 0
6: SLL, Diff	00001110	01000100	00001010	00000100
6: Rem = Diff	00000100	01000101		
7: SLL, Diff	00001000	10001010	00001010	< 0
8: SLL, Diff	00010001	00010100	00001010	00000111
8: Rem = Diff	00000111	00010101		

Check:

Dividend = 11011001_2 = 217 (unsigned)

Divisor = 00001010_2 = 10

Quotient = 00010101_2 = 21 and Remainder = 00000111_2 = 7

b) (5 pts) Show the **Double precision** IEEE 754 representation for: **-0.05**

```
0.05 * 2 = 0.1
0.1 * 2 = 0.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6
0.6 * 2 = 1.2
0.2 * 2 = 0.4
```

$0.05 = 0.00001100\overline{11001}_2 = 1.100\overline{11001}_2 \times 2^{-5}$

Exponent = -5 + 1023 = 1018 = 01111111010_2

Double Precision Representation:

1 01111111010

1001100110011001100110011001100110011001100110011010 (rounded)

Q3. Given x = **1 10000101 10110000000000000000001**$_2$
and y = **1 01111111 01000000000000011000000**$_2$
represent single precision floating-point numbers. Perform the following operations showing all the intermediate steps and final result in binary. Round to the nearest even.

a) (12 pts) $x + y$

Exponent Value(x) = 10000101$_2$ – bias = 133 – 127 = 6
Exponent Value(y) = 01111111$_2$ – bias = 127 – 127 = 0

- 1.101 1000 0000 0000 0000 0001$_2$ × 2^6

- 1.010 0000 0000 0000 1100 0000$_2$ × 2^0

- 1.101 1000 0000 0000 0000 0001$_2$ × 2^6

- 0.000 0010 1000 0000 0000 0011 000000$_2$ × 2^6 (shift)

- 1.101 1010 1000 0000 0000 0100 000000$_2$ × 2^6 (add)

- 1.101 1010 1000 0000 0000 0100 × 2^6 (rounded)

Result = 1 10000101 10110101000000000000100

Q3. b) (13 pts) $x \times y$

```
Biased exponent = 10000101₂ + 01111111₂ – 127 = 10000101₂
Result sign = 0 (positive)

                                   1.10110000000000000000001₂
                                 × 1.01000000000000011000000₂
         ___________________________________________________

                              110110000000000000000001
                             110110000000000000000001
                110110000000000000000001
             1.10110000000000000000001
         ___________________________________________________

          10.0001110000000010100010101000000000000011

Normalize and adjust exponent:

1.00001110000000010100010  1  01000000000000011₂

Biased exponent = 10000101₂ + 1 = 10000110₂

Round to nearest even:

Round bit = 1, Sticky bit = 1 (OR of remaining bits)

Rounded Significand = 1.00001110000000010100010₂ + 1

                    = 1.00001110000000010100011₂


Product = 0 10000110 00001110000000010100011₂
```

Q4. (20 pts) A program, being executed on a processor, has the following instructions mix:

Operation	**Frequency**	**Clock cycles per instruction**
ALU	40 %	2
Load	20 %	10
Store	15 %	4
Branches	25 %	3

a) (3 pts) Compute the average clock cycles per instruction

Average CPI_a = 0.4*2 + 0.2*10 + 0.15*4 + 0.25*3 = 4.15

b) (6 pts) Compute the percent of execution time spent by each class of instructions

Operation	**Frequency**	**CPI**	**CPI * Frequency**	**% Execution Time**
ALU	40 %	2	0.8	0.8 / 4.15 = 19.3%
Load	20 %	10	2.0	2.0 / 4.15 = 48.2%
Store	15 %	4	0.6	0.6 / 4.15 = 14.4%
Branches	25 %	3	0.75	0.75 / 4.15 = 18.1%

c) (6 pts) A designer wants to improve the performance. He designs a new execution unit that makes 80% of ALU operations take only **1** cycle to execute. The other 20% of ALU operations will still take **2** cycles to execute. The designer also wants to improve the execution of the memory access instructions. He does it in a way that **95%** of the **load** instructions take only **2** cycles to execute, while the remaining **5%** of the **load** instructions take **10** cycles to execute per **load**. He also improves the store instructions in such a way that each **store** instruction takes **2** cycles to execute.

Compute the new average cycles per instruction

Average CPI_c = 0.8*0.4*1 + 0.2*0.4*2 +
0.2*0.95*2 + 0.2*0.05*10 +
0.15*2 + 0.25*3 = 2.01

d) (2 pts) What is the speedup factor by which the performance has improved in part **c**?

Speedup = 4.15 / 2.01 = 2.06 (I-count & clock are the same)

e) (3 pts) The designer decides to improve the clock speed in such a way to **triple** the overall performance of the original CPU specified in part **a**.

By what factor should the clock rate be improved if the designer uses the design specified in part **c**?

Speedup = (CPI_a / CPI_c) * (Clock $Rate_c$/Clock $Rate_a$)

Speedup = 3 = (4.15/2.01) * (Clock $Rate_c$/Clock $Rate_a$)

Clock should be faster by 3/2.06 = 1.45 (45% faster)

Q5. (25 pts) The following code fragment processes two double-precision floating-point arrays *A* and *B*, and produces an important result in register **`$f0`**. Each array consists of **`10000`** double words. The base addresses of the arrays *A* and *B* are stored in **`$a0`** and **`$a1`** respectively.

```
        ori     $t0, $zero, 10000
        sub.d   $f0, $f0, $f0

loop:   ldc1    $f2, 0($a0)
        ldc1    $f4, 0($a1)
        mul.d   $f6, $f2, $f4
        add.d   $f0, $f0, $f6
        addi    $a0, $a0, 8
        addi    $a1, $a1, 8
        addi    $t0, $t0, -1
        bne     $t0, $zero, loop
```

a) (6 pts) Write the code in a high-level language, and describe what is produced in **`$f0`**.

```
for (i=0; i<10000, i++) sum = sum + A[i] * B[i];
Compute the dot product and return sum in $f0.
```

c) (5 pts) Count the total number of instructions executed by all the iterations (including those executed outside the loop).

```
Instruction Count = 2 + 10000 * 8 = 80002
```

d) (14 pts) Assume that the code is run on a machine with a **2 GHz** clock that requires the following number of cycles for each instruction:

Instruction	Cycles
addi, ori	1
ldc1	3
add.d, sub.d	5
mul.d	6
bne	2

(7 pts) How many cycles does it take to execute the above code?

Clock cycles = 1 (ori) + 5 (sub.d) + 10000 * (2*3 (ldc1) + 6 (mul.d) + 5 (add.d) + 3*1 (addi) + 2 (bne))

= 6 + 10000 * 22 = 220006 cycles

(3 pts) How many second to execute the above code?

Execution time = cycles / clock rate = 220006/2 nsec

= 110003 nsec = 110 usec = 0.11 msec = 0.00011 seconds

(2 pts) What is the average CPI for the above code?

Average CPI = Clock Cycles / Instruction-Count =

Average CPI = 220006 / 80002 = 2.75

(2 pts) What is the MIPS rate for the above code?

MIPS rate = 80002 / 110 usec = 727.3 MIPS