ICS 233, Term 071

Computer Architecture & Assembly Language

HW# 4

Q.1. Write a MIPS assembly program to perform **signed multiplication** of 32-bit numbers using the algorithm studied in a class. The program should ask the user to inter two integers and then display the result of multiplication. If the result cannot fit in 32-bit then the program should indicate that there is overflow. Test your program using the following numbers:

1. -1 x -1
2. 100 x -2
3. 0 x 10
4. 2147483647 x 2

A sample execution of the program is shown below:

Enter the multiplier: 100
Enter the multiplicand: -2
Result of multiplication = -200

Q.2. Write a MIPS assembly program to perform **signed division** of 32-bit numbers using the algorithm studied in a class. The program should ask the user to inter two integers and then display the result of division displaying both the quotient and remainder. Test your program using the following numbers:

1. +17 ÷ +3
2. +17 ÷ -3
3. -17 ÷ +3
4. -17 ÷ -3

A sample execution of the program is shown below:

Enter the dividend: 17
Enter the divisor: -3
Result of division: Quotient = -5 Remainder = 2

Q.3. What is the decimal value of the following single-precision floating-point numbers?

(i) `0010 0000 0001 1100 0000 0000 0000 0000`

(ii) `1100 1111 1110 1000 0000 0000 0000 0000`

Q.4. Show the IEEE 754 binary representation for: `-24.0625` in …

(i) Single Precision

(ii) Double precision

Q.5. Perform the following floating-point operations rounding the result to the nearest even. Perform the operation assuming both infinite precision and using only guard, round and sticky bits. Compare your solution in both cases.

(i)
```
  0100 0011 1000 0000 0000 0000 0000 0000
- 0100 0001 1000 0000 0000 0000 0000 1100
```

(ii)
```
  0011 1111 1000 0000 0000 0000 0000 0000
- 0011 1111 1000 0100 0000 0000 0000 0000
```

(iii)
```
  0011 1111 1111 1111 1111 1111 1111 1110
+ 0011 0100 0100 0000 0000 0000 0000 0000
```

Q.6. Given that x = `0101 1111 1011 1110 0100 0000 0000 0000`
y = `0011 1111 1111 1000 0000 0000 0000 0000`
z = `1101 1111 1011 1110 0100 0000 0000 0000`
represent single precision IEEE 754 floating-point numbers. Perform the following operations showing all work:

(i) $x + y$

(ii) Result of (**i**) + z

(iii) Why is the result of (**ii**) counterintuitive?