**CSC390 (Computer Org. & Arch.)**

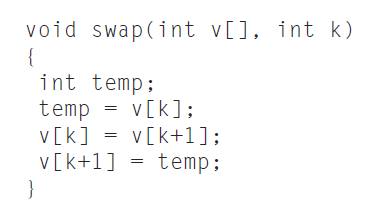
**Exam#2- Spring 2018**

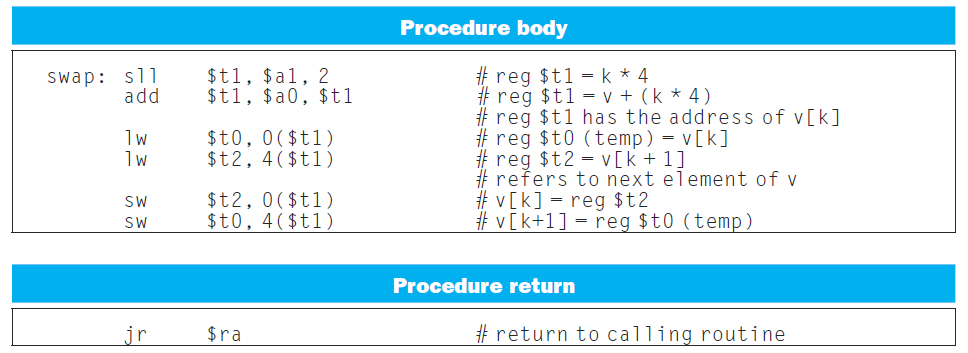


**Name:**

**Student ID:**

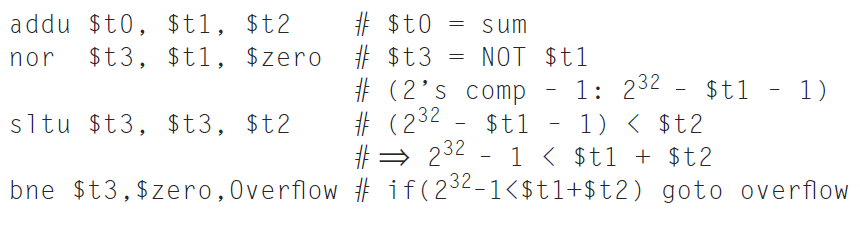
(There are four questions. All Questions have equal weights)**#Q1.** Consider the following C procedure and its corresponding MIPS assembly code which simply swap two adjacent memory locations contents. Note that the given assembly code works only for integer data type. The parameters, v and k, will be found in registers $a0 and $a1. The only other variable temp is associated with register $t0. [10 pts]





What changes you would make in the above MIPS Assembly code so that it will work for **single precision floating point** data type? Write down the modified code. Include Syscall function to display the results.

**Q2.** The following sequence of MIPS codes can detect overflow in unsigned addition of two unsigned numbers. Registers $t1 and $t2 contain the signed numbers.

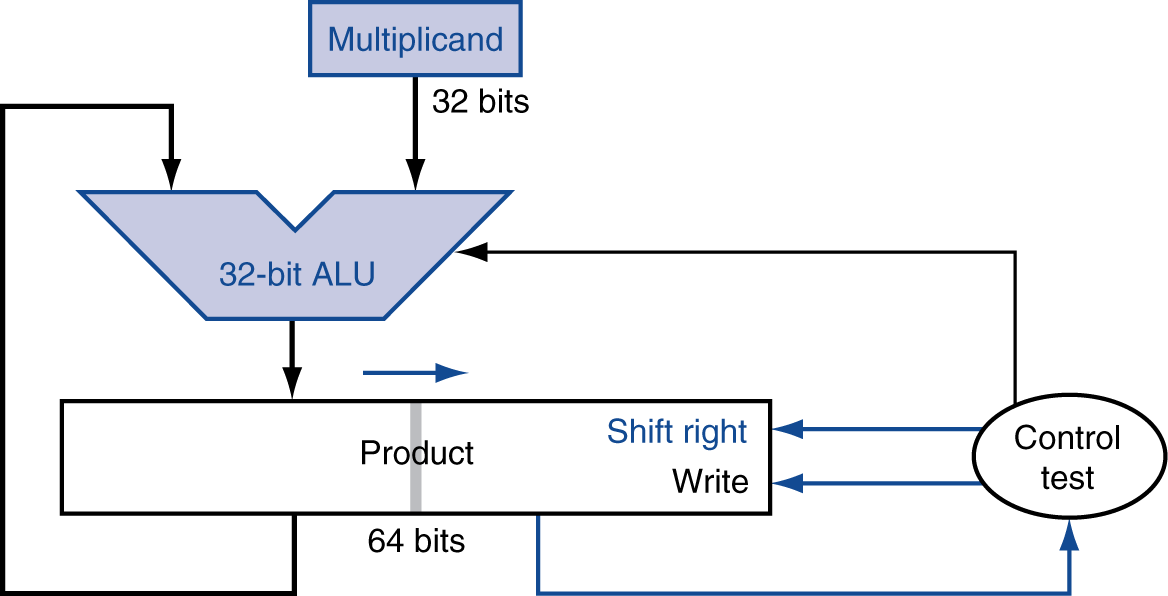


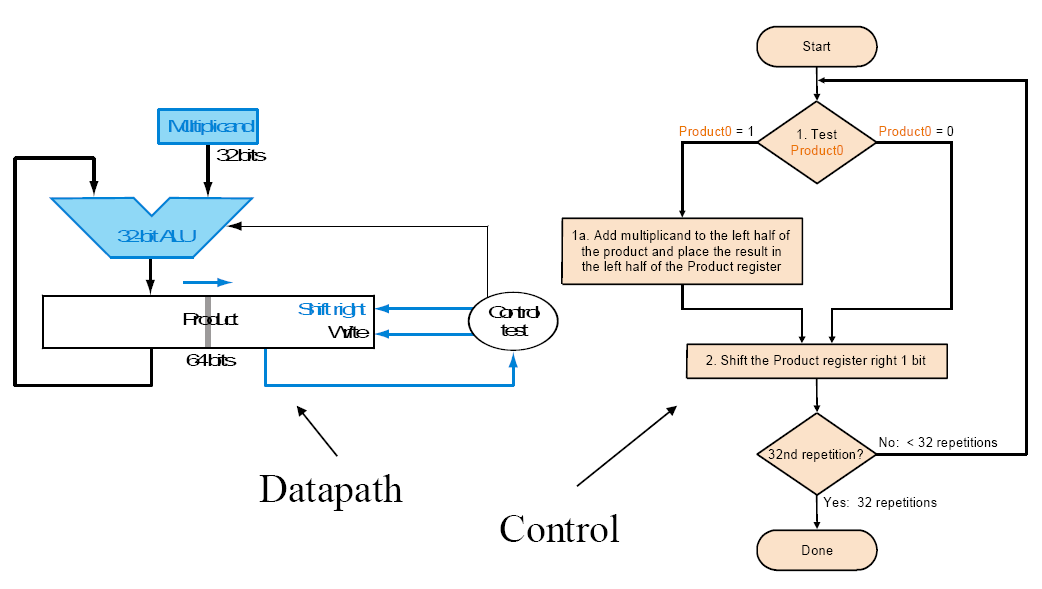
Consider, the code is working on two four-bit unsigned numbers stored in $t1 and $t2. For the following two pairs of $t1 and $t2, find if the code will detect any overflow in the addition operation. **Explain your results with the values of registers $t0 and $t3** as the program executes the sequence of the above MIPS assembly code (i.e. show the values of $t0 and $t3 for every line of the codes and justify your results).   
 [10 points]

1. $t1 = 0101; b) $t1=0110;

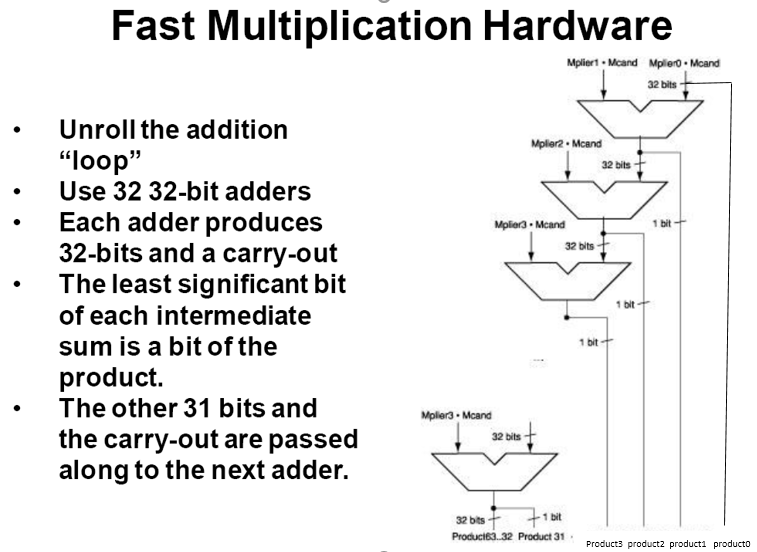
$t2 = 0010; $t2=0100;

**Q3. (a)** The following Circuit performs the multiplication of two 32-bit unsigned binary numbers and produces a 64-bit result. Figure 1 shows the three steps multiplication algorithm that the circuit performs to produce the result. Suppose, you are assigned to design a 4-bit multiplier circuit which will perform the multiplication of the two 4-bit unsigned numbers A=(1101)2 and B=(1011)2 and produce an 8-bit result. **Modify hardware and the flow chart of the algorithm**. Tell me **how many addition and shifting operations** would be **required** to perform the above multiplication. [10 pts]





**Q3. (b)** The circuit in part(a) required several processing steps to perform the multiplication. To improve the execution speed of the above multiplier circuits, a Fast Multiplication Hardware, as shown in figure 3(b), is designed using (n-1) adder circuits, which perform the multiplication of two n-bit numbers and produces the 2n-bit result. Suppose, you are assigned to design a 4-bit Fast Multiplication Hardware circuit which will perform the multiplication of the two 4-bit unsigned numbers A=(1101)2 and B=(1011)2 and produce a 8-bit result. **Draw the necessary hardware** and show how the result is produced in the multiplication process. [10 pts]



**31**

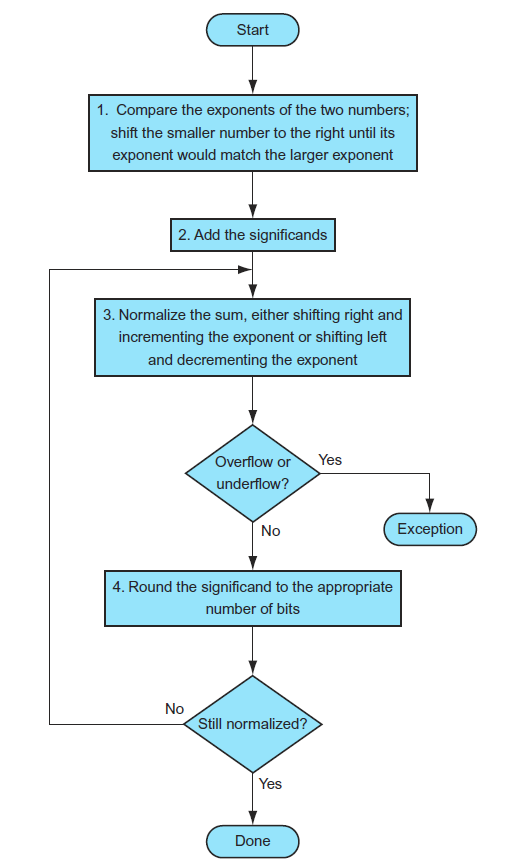
Figure (b)

**Q4. (a)** IEEE 754 binary representation of floating point numbers is widely used in today’s computer systems. Consider the decimal number (-0.09375)10, represent it in a single precision IEEE 754 format. Show all the steps of your calculation and the final results in Hex format. [8 pts]

**Q4. (b)**  Let’s consider a 11-bit binary floating-point number system, in which -like IEEE 754 number system- 1 sign bit, 5 Exponent bits and 5 fraction bits are used to represent a floating-point number. Two such floating point numbers, as shown on figure 4(b), are added following the flow chart of Figure 4. The circuit that implements the flow chart is shown in figure 4(b).

Note that the Exponent = actual exponent + Bias, where Bias = 2n-1 – 1; n = number of bits in Exponent and like the IEEE 754 system, the decimal equivalent of the binary floating point number is represented by: (-1)Sx(1+Fraction)x2(Exponent – Bias) (**ref:** Section 3.5 of your Text)

1. Without doing any calculation tell me which floating point number has the smaller value and then calculate the corresponding decimal value. [7 pts]
2. Write a short note on the operations of the controller circuit (i.e. oval shape control box) of Figure 4(b). Specifically discuss its role in the different steps of the floating-point addition process. [8 pts]

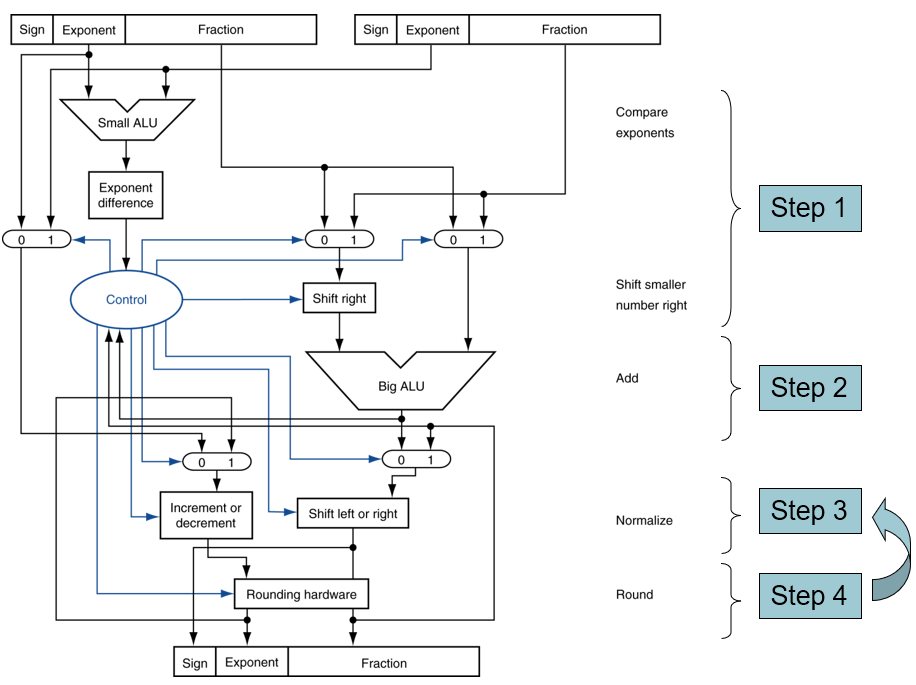


**Figure 4**

**10-bit Binary floating point numbers:**

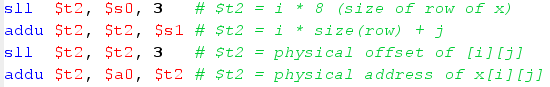
**0 1 0 1 1 0 1 1 0 1 0**

**0 0 1 1 0 1 1 1 1 0 1**



**Figure 4(b)**

**Q5.** Consider, X[i][j] is a **double precision type** matrix of size 8x8. The following code converts the 2D indices (i.e. [i][j]) of the matrix to the corresponding linear indexes of the memory where the matrix data is stored. Note that the registers **$S0** and **$S1** correspond to the **i** and **j** indexes, respectively. Also, register **$a0** contains the base address (i.e. starting memory address) of X.



1. Given, **i = 2, j = 3**, and the base address of X, i.e. **$a0** contains 20010, calculate the corresponding physical address of the memory location. [5 pts]
2. What changes would you make in the above codes so that the modified code would work for a matrix X[i][j] of size **16x16** and each element is of **single precision floating point** type? [12 pts]