**HW#3 (CSC390)**

Due: 02/12/2016 by 5:00PM

**#Q1.**

Write a MIPS assembly language program that will perform the following C code operations:

for (i = 0; i < 8; i ++) {

C[i] = A[i + 1] - A[i] \* B[i + 2]

}

Consider the arrays A and B are initialized with the following two arrays, respectively

[10, 12, 14, 16, 18, 11, 13, 15, 17, 19] and [11, 12, 13, 14, 15, 16, 18, 20, 22, 24].

Store the results in an array of consecutive memory locations C.

**#Q2.**

Write a MIPS assembly language program that calls a procedure, Add\_Sub\_Mul, which accept four parameters (g,h,i,j) and returns,

f = (g+h) if i >j; f= (g-h) if i < j; and f = g\*h if i == j; the equivalent C function is shown below:

int Add\_Sum\_Mul (int g, int h, int i, int j) {

int f;

if (i > j) {

f=(g+h);}

else if (i < j) {

f=(g-h);}

else { f=0;}

}

Consider the variables g, h, i, j and f are initialized with some initial values in the data segment. Use $s0 as f in the function and also use $s0 to store the base address of f in the memory location. Clearly comment on the every instruction you use in your program. Specially, clearly show and describe the stack operation. Remember, resisters ($a0-$a2) are used for passing arguments in to the function and $v resisters are used to store the results in the function.

**#Q3.**

Convert the C function below to MIPS assembly language. Also write a MIPS assembly code to call the function with some initial value of n and store the result in a suitable memory location, labeled as result. Make sure that your assembly language code could be called from a standard C program (that is to say, make sure you follow the MIPS calling conventions).

unsigned int sum(unsigned int n)

{

if (n == 0) return 0;

else return n + sum(n-1);

}

This machine has no delay slots. The stack grows downward (toward lower memory addresses). The following registers are used in the calling convention:

