**CPRE 381- Intro to Computer Organization & Implementation**

**HW2**

**Due Date: Sept 20, 2017**

1. Consider the following execution times in seconds for Programs P1 & P2 on Machines M1, M2, & M3.

|  |  |  |  |
| --- | --- | --- | --- |
|  | M1 | M2 | M3 |
| P1 | 10 | 100 | 300 |
| P2 | 100 | 10 | 400 |

What are the geometric mean execution times for M1, M2, and M3 over these programs? Now show geometric mean ratios (like SPECratio) for M1 and M2 with M3 as the reference machine.

M1: 10 \* 100 = sqrt(1000) = 31.62

M2: 100 \* 10 = sqrt(1000) = 31.62

M3: 300 \* 400 = sqrt(120000) = 346.41

M1-P1: 300/10 = 30

M1-P2: 400/100 = 4

M2-P1: 300/100 = 3

M2-P2: 400/10 = 40

M1: 30 \* 4 = sqrt(120) = 11 approx.

M2: 3 \* 40 = sqrt(120) = 11 approx.

**[10 points]**

1. Your textbook says that 2’s complement signed representation for *X=xn-1x n-2 …x2 x1 x0* is *- xn-12n-1 + x n-22n-2+…+ x222 + x121+ x0* *20*as opposed to the unsigned representation *xn-12n-1 + x n-22n-2+…+ x222 + x121+ x0* *20*. On the other hand, it is also stated that to get 2’s complement of a number *X*, you can take its 1’s complement and then add 1. Justify how the two statements are equivalent.

**[10 points]**

The signed representation can be simplified to -*xn-12n-1+* The summation of the first of the equation (xi2i) which is the same getting 2’s complement because -*xn-12n-1* is the 1’s complement of the summation, and the summation adds 1. So therefore they are equivalent.

1. Compute the sign extension into 16-bits of +19 and -122 represented in 2’s complement in 8-bits. Prove that when an 8-bit representation is sign-extended into 16 bits by replicating the sign bit 8 times in the more significant end, you get the same value both for a negative and non-negative *X* using *X*=*- xn-12n-1 + x n-22n-2+…+ x222 + x121+ x0* *20*. **[10 points]**

+19: 0001 0011 binary version

0000 0000 0001 0011 16-bit extended

- 0(2^15) + 0(2^14) + 0(2^13) … +1(2^4) + 0(2^3) + 0(2^2) + 1(2^1) + 1(2^0) = -19

- -0(2^15) +-0(2^14)…+-1(2^4)+-0(2^3)+-0(2^2)+-1(2^1) + -1(2^0) = -19

-122: 1000 0110

1111 1111 1000 0110 16-bit extended version

-1(2^15) + 1(2^14) + … + 0(2^6) … + 1(2^2) + 1(2^1) + 0(2^0) = -122

--1(2^15) + -1(2^14) + … + 0(2^6) … + -1(2^2) + -1(2^1) + -0(2^0) = -122

1. Convert *X= (Y+Z) – (U-V);* into MIPS assembly. Assume that the memory addresses of *X, Y, Z, U, V* are 8, 12, 16, 20, 24 offset from *R14*. Allocate *X, Y, Z, U, V* to registers *R6, R7, R8, R9, R10*. Convert this assembly code to machine level code. **[10 points]**

lw $a2, 8($R14) #R6 = X

lw $a3 12($R14) #R7 = Y

lw $t0, 16($R14) #R8 = Z

lw $t1, 20($R14) #R9 = U

lw $t2, 24($R14) #R10 = V

add $t3, $a3, R8 // temp = Y+Z

sub $t4, $t1, $t2 // temp = U-V

sub $a2, $t3, $t4 // X = temp – temp1

1. Write MIPS assembly equivalent of **if**(*X > Y) X = X+Y* **else** *X = X+X*;. Assume memory address offsets 12, 16 from *R12* for *X, Y*. Use registers *R13, R14* for *X, Y*. **[10 points]**

lw $t5, 12($t4) # R13 = x

lw $t6, 16($t4) # R14 = y

slt $t0, $t6, $t5 // if y < x t0 =1 otherwise t0 = 0

beq $t0, 0, Else #if t0 is not 1 branch to else

add $t5, $t5, $t6 # x = x+y

j Exit # end

else:

add $t5, $t5, $t5 # else x = x+x

exit: