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

**HW2**

**Due Date: Feb 6, 2017**

**Ningyuan Zhang**

**Section A**

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | M1 | M2 | M3 |
| P1 | 1 | 100 | 200 |
| P2 | 100 | 1 | 500 |

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.

**[10 points]**

M1: GM = (1\*100)^0.5 = 10

M2: GM = (100\*1)^0.5 = 10

M3: GM = (200\*500)^0.5 = 316.23

GM ratios (M1) = ((200/1)\*(500/100))^0.5 = 31.62

GM ratios (M2) = ((200/100)\*(500/1))^0.5 = 31.62

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]**

Proof: The first statement shows us the calculation of the value of 2’s complement of X, and the second statement shows us the way to convert a number to 2’s complement of it. They have the same result:

Given X = 101

1st way : (-1 x 2¬2 ) + (0 x 21 ) + (1 x 20) = -4 +1 = -3

2nd way: 3 = 011

1’s complement : 100

2’s complement: 101 (just add 1)

1. Compute the sign extension into 16-bits of +20 and -123 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]**

+20 = 0000 0000 0001 0100

-123 = 1111 1111 1000 0101

0001 0100 = -0\*2^7 + 0\*2^6 + 0\*2^5 + 1\*2^4 + 0\*2^3 + 1\*2^2 + 0\*2^1 + 0\*2^0 = 22

1000 0101 = -1\*2^7 + 0\*2^6 + 0\*2^5 + 0\*2^4 + 0\*2^3 + 1\*2^2 + 0\*2^1 + 1\*2^0 = -123

1. Convert *X= (Y+Z) – (U+V);* into MIPS assembly. Assume that the memory addresses of *X, Y, Z, U, V* are 16,20, 24, 36, 48 offset from *R12*. Allocate *X, Y, Z, U, V* to registers *R10, R11, R13, R14, R15*. Convert this assembly code to machine level code. **[10 points]**

lw $R10, 16( $R12 ) 1000 1101 1000 1010 0000 0000 0001 0000

lw $R11, 20( $R12 ) 1000 1101 1000 1011 0000 0000 0001 0100

lw $R13, 24( $R12 ) 1000 1101 1000 1101 0000 0000 0001 1000

lw $R14, 36( $R12 ) 1000 1101 1000 1110 0000 0000 0010 0100

lw $R15, 48( $R12 ) 1000 1101 1000 1111 0000 0000 0011 0000

# Y+ Z

add $R6 , $R11, $R13 0000 0001 0110 1101 0011 0000 0010 0000

# U + V

add $R7, $R14, $R15 0000 0001 1100 1111 0011 1000 0010 0000

# X

sub $R10, $R6, $R7 0000 0000 1100 0111 0101 0000 0010 0010

# store X

sw $R10, 0($R12) 1010 1101 1000 1010 0000 0000 0000 0000

1. Write MIPS assembly equivalent of **if**(*X > Y) X = X+X* **else** *X = X+Y*;. Assume memory address offsets 8, 12 from *R10* for *X, Y*. Use registers *R11, R12* for *X, Y*. **[10 points]**

lw $R11, 8($R10) # load X to register R11

lw $R12, 12($R10) # load Y to register R21

slt $R3, $R12, $R11 # set $R3 to 1 if Y <X

beq $R3, $zero, else # go to else if $R3 = 0 ( same as Y <X is false)

add $R11, $R11, $R12 # X = X+Y

j exit

else: sub $R11, $R11, $R12

exit: sw $R11, 0($R10)