**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]**

The two statements are equivalent because the first statement shows us how to calculate the value of 2’s complement of X, while the second statement shows us how to convert a number to 2’s complement of it. Both way gives us 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

add 1 => 2’s complement: 101

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]**

# allocate X, Y, Z, U, V

# to R1, R2, R3, R4, R5

lw $R1, 0( $R18 ) 1000 1110 0100 0001 0000 0000 0000 0000

lw $R2, 4( $R18 ) 1000 1110 0100 0010 0000 0000 0000 0100

lw $R3, 8( $R18 ) 1000 1110 0100 0011 0000 0000 0000 1000

lw $R4, 12( $R18 ) 1000 1110 0100 0100 0000 0000 0000 1100

lw $R5, 16( $R18 ) 1000 1110 0100 0101 0000 0000 0001 0000

# Y+ Z //0000 00ss ssst tttt dddd d000 0010 0000

add $R6 , $R2, $R3 0000 0000 0100 0011 0011 0000 0010 0000

# U + V

add $R7, $R4, $R5 0000 0000 1000 0101 0011 1000 0010 0000

# X

sub $R1, $R6, $R7 0000 0000 1100 0111 0000 1000 0010 0000

# store X #1010 11ss ssst tttt iiii iiii iiii iiii

sw $R1, 0($R18) 1010 1110 0100 0001 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 $R1, 0($R18) # load X to register R1

lw $R2, 4($R18) # load Y to register R2

slt $R3, $R2, $R1 # set $R3 to 1 if Y <X

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

add $R1, $R1, $R2 # X = X+Y

j exit

else: sub $R1, $R1, $R2

exit: sw $R1, 0($R18)