**Department of Electrical and Computer Engineering**

Author: Zainab Hussein

Title: MIPS Assembly Programming

Date: 3-27-2017

1. Time Spent: 6 Hours

2. Commented Fibonacci code

# File: Task1\_Fibonacci\_Numbers.s

# Author: Zainab Hussein

# Created on March 28, 2017, 8:30 AM

.global main # define main as global label

.set noreorder # don't let the assembler reorder instructions

Main: addi $a0, $0, 6 # n = 8 (find 8th fib number )

addi $s0, $0, 1 # prev = fib(-1) = 1

addi $s1, $0, 0 # curr = fib(0) = 0

loop: beq $s2,$a0,done # if the current and nth order to stop # the loop are equal, go to done

add $s1,$s0,$s1 # adding the previous value to the current value

# to get the fibonacci sequence

sub $s0,$s1,$s0 # update previous value to the last current

addi $v0,$s1,0 # save the current value into $v0

addi $s2,$s2,1 # increment counter by 1

j loop # return back to begin the loop

add $0,$0,$0 # branch delay slot (nop)

done: j done # infinite loop

add $0, $0, $0 # branch delay slot (nop)

3. Floating-point addition code

# File: Task1\_Fibonacci\_Numbers.s

# Author: Zainab Hussein

# Created on March 28, 2017, 9:40 AM

# The numbers below are loaded into memory (the Data Segment)

# before your program runs. You can use a lw instruction to

# load these numbers into a register for use by your code.

.data

atest1: .word 0x3F800000 # 1.0

btest1: .word 0x3F800000 # 1.0

# add more test vectors here

atest2: .word 0x40000000 # 2.0

btest2: .word 0x3F800000 # 1.0

atest3: .word 0x40000000 # 2.0

btest3: .word 0x40600000 # 3.5

atest4: .word 0x3F0103B0 # 0.50390625

btest4: .word 0x477FB000 # 65535.6875

fmask: .word 0x007FFFFF # mask for masking the fraction bits

emask: .word 0x7F800000 # mask for masking the exponent

ibit: .word 0x00800000 # mask for the implicit leading one

obit: .word 0x01000000 # mask for the overflow bit

.text

.global main # define main as a global label

.set noreorder # don't let the assembler reorder instructions

# Test the floating point add

main:

lw $a0, atest1 # first operand

lw $a1, btest1 # second operand

jal flpadd # do the addition, look for result in $v0

nop # branch delay slot (nop)

lw $a0, atest2 # first operand

lw $a1, btest2 # second operand

jal flpadd # do the addition, look for result in $v0

add $0, $0, $0 # branch delay slot (nop)

lw $a0, atest3 # first operand

lw $a1, btest3 # second operand

jal flpadd # do the addition, look for result in $v0

add $0, $0, $0 # branch delay slot (nop)

lw $a0, atest4 # first operand

lw $a1, btest4 # second operand

jal flpadd # do the addition, look for result in $v0

add $0, $0, $0 # branch delay slot (nop)

infiniteloop:

j infiniteloop # wait forever

nop # branch delay slot (nop)

flpadd:

# extract exponent

# bits 30-23 = exponent (8 bits)

lw $t0, emask # loading emask into temporary register t0

and $t1, $a0, $t0 # exponent bits of first number

srl $t1, $t1, 23 # shift the first number right 23 bits

and $t2, $a1, $t0 # exponent bits of second number

srl $t2, $t2, 23 # shift the second number right 23 bits

# extract fraction

# bits 22-0 = fraction (23 bits)

lw $t0, fmask # loading fmask into temporary register t0

and $t3, $a0, $t0 # fractional bits of the first number

and $t4, $a1, $t0 # fractional bits of the second number

# prepend leading one

# bit 31 = sign (1 bit)

lw $t0, ibit # loading ibit into temporary register t0

or $t5, $t0, $t3 # append leading 1 at the first number

or $t6, $t0, $t4 # append leading 1 at the second number

sltu $t0, $t1, $t2 # sets t0 to 1 if t2's exponents are < t1's

bne $t0, $0, subtract2 # compares t0 to 0, if it equals 0, subtract2

add $0, $0, $0 # branch delay slot (nop)

subtract1:

# compare exponents t1>t2 and shift smaller mantissa

sub $t7, $t1, $t2 # subtract the smaller register($t2) from the

# larger one($t1)

add $t2, $t1, $0 # set the exponent to the larger of the two

srl $t6, $t6, $t7 # shift the smaller register($t6) by the

# difference of exponents

j sum # jump to the sum label

add $0, $0, $0 # branch delay slot (nop)

subtract2:

# compare exponents t2=>t1 and shift smaller mantissa

sub $t7, $t2, $t1 # subtract the value in t1 from t2

add $t1, $t2, $0 # set the exponent to the larger of the two

srl $t5, $t5, $t7 # shift the exponent by the difference of the two

add $0, $0, $0 # branch delay slot (nop)

sum:

# add mantissas

add $t7, $t5, $t6 # add the two mantissas

lw $t0, obit # load the obit into the register t0

sltu $t3, $t7, $t0 # compare the summed mantissa with the obit value

bne $t3, $0, combine # if t3 is 1, then the mantissa is less than the

# obit, and we merge without normalizing

add $0, $0, $0 # branch delay slot (nop)

normalize:

# normalize mantissa

srl $t7, $t7, 1 # shift the mantissa to the right by 1

addi $t1, $t1, 1 # add one to the exponent value

combine:

# reassemble into fp format

lw $t0, ibit # load the ibit value into t0

sub $t7, $t7, $t0 # subtract the implicit 1 from the mantissa

sll $t1, $t1, 23 # shift the exponent to the left 23 places

or $v0, $t1, $t7 # combine the exponent and the mantissa values

jr $ra

add $0, $0, $0 # branch delay slot (nop)

4. Result of test cases

|  |  |  |
| --- | --- | --- |
| **Num1** | **Num2** | **Sum** |
| 0x3F800000 | 0x3F800000 | 0x40000000 |
| 0x40000000 | 0x3F800000 | 0x40400000 |
| 0x40000000 | 0x40600000 | 0x40B00000 |
| 0x3F0103B0 | 0x477FB000 | 0x477FB081 |