**ECEN 323 – Winter 2020**

Lab #4: RISC-V Simulator

Johnson, Ryan

Section 01

Preliminary

Summarize the difference between binary machine language, assembly language, and high-level source files like C.

High level languages like C can be more easily understood and shared by people.

Assembly language breaks down the individual instructions needed to perform the complex high-level instructions.

Binary machine language is the part that is actually understood by machines because they are just binary, 1’s and 0’s in a format that it will carry out the instructions.

Contrast the difference between a compiler, assembler, and linker

The compiler translates programming language into assembly language.

The assembler translates assembly language into machine code.

A linker combines objects and the needed library code into something executable.

What is the difference between the data segment and the text segment of in an assembly language program?

Data segment is the part of the object file that contains the global or static variables.

Text segment is the portion of the object file that contains executable instructions.

What register is used as the stack pointer?

x2

What is an ebreak instruction in the sample program?

It returns control to a debugging environment.

How would you define a constant named 'BLUE' and assign it to the value 0x00F in assembly language?

.eqv BLUE 0x00F

Describe how to implement the psuedo-instruction li t1, 0 using a native instruction.

Addi x6, x0, x0

Exercise #1

What is the memory address of the start of the text segment for the simulator? (GUI: Settings: Memory Configurations)

x00400000

What is the memory address of the start of the static data in the data segment for the simulator (GUI: Settings: Memory Configurations)

x10000000

What is the memory address of the start of the dynamic data (heap) in the data segment for the simulator? (GUI:Settings: Memory Configurations)

x10040000

What is the memory location of the stack segment for the simulator? (GUI: Settings: Memory Configurations)

x7ffffffc

Example 1

How many extended (pseudo) instructions are there in this example?

9

What does the last instruction do?

It loops indefinitely. Just used in a small debugging like this one.

Why does “lw a0,input” use the auipc command when assembled?

It uses the auipc command to add the offset of the input to the pc, so that the input can be accessed in the load.

What are “.globl,.data,.text,and .word” and what do they mean?

.glbl allows the function to be referenced from another file.

.data marks the base address for the data segment

.text marks the base address for the text segment

.word stores the value given after it as a 32 bit word

What instruction does “j do\_fact” translate to? why does it use x0?

jal x0,0xfffffff8

Because it won’t ever return, 0 is an easy thing to access and so it sets x0 to the return address and then jumps to the address of do\_fact.

What is the cycle count when the simulator gets to the first “j exit\_loop”? (step through the program)

x0000001a

What is at the “output” address at the end of the program?

0x10010004

Example 2

What is the printout when the program is executed?

5!= 120

What is the cycle count when the program finishes?

x0000002f

What system calls are used?

ebreak and ecall are used for the print\_int, print\_str, print\_int calls.

In what register does the jal command store the return address?

It is stored in x1 also know as ra.

Change the input value, what is the lowest positive input that returns a negative output? why?

17

This is due to overflow.

Example 3

What is the cycle count when the program finishes?

0x00000080

How many new lines of instructions were added?

24

What is the lowest value of the stack pointer (sp) during the program?

0x7fffefe0

What registers are used for the arguments to subroutines and the return values?

a0 and a1 are the arguments and a0 is the return value.

What is the difference between sX and aX registers in the calling convention?

a# is caller saved while the s# is callee saved.

Exercise #2

Determine the address (location in memory) of the “main” and “fact\_func” procedures within the .text segment

main 0x00400000

fact\_func 0xfffffff0

What is the address where the ascii string “! = ” is stored?

0x10010028

What is the value of the stack pointer before the program executes?

0x7fffeffc

How many instructions were executed to compute the factorial of 10?

152

What is the smallest value of the stack pointer during the execution when finding the factorial of 16?

0x7fffef74

Exercise #3

**Iterative**

How many iterations of the loop will be executed for fib\_iterative(5)?

4

Provide a copy of your iterative Fibonacci sequence in your lab report.

fibinnoci:

beqz,a0,done # if a0 = 0 return 0

li a1,1

beq a0,a1,done # If a0 = 1 return 1

li a1,1 # fib\_1 = 1;

li a2,0 # fib\_2 = 0;

li a4,2

for:

add a3,a1,a2

addi a2,a1,0

addi a1,a3,0

addi a4,a4,1

ble a4,a0,for

addi a0,a3,0

done:

ret

Report cycle count of your implementation for fib\_iterative(10).

0x00000042

**Recursive**

Provide an ordered list of all recursive calls to the “fib\_recursive” function that are made when calling fib\_iterative(5).

Fib\_iterative (5)

Fib\_iterative (4)

Fib\_iterative (3)

Fib\_iterative (2)

Fib\_iterative(1)

Fib\_iterative(0)

Fib\_iterative(1)

Fib\_iterative(2)

Fib\_iterative(1)

Fib\_iterative(0)

Fib\_iterative (3)

Fib\_iterative (2)

Fib\_iterative(1)

Fib\_iterative(0)

Fib\_iterative(1)

Provide a copy of your recursive Fibonacci sequence in your lab report.

fibinnoci:

addi sp, sp, -4

sw ra, 0(sp)

jal fibinnoci\_rec

lw ra, 0(sp)

addi sp, sp, 4

ret

fibinnoci\_rec:

addi sp, sp, -12

sw s0, 0(sp)

sw s1, 4(sp)

sw ra, 8(sp) # Load return address to the stack

beqz a0, done # if a0 = 0 return 0

li s1, 1

beq a0, s1, done # If a0 = 1 return 1

addi s0, a0, -2

addi a0, a0, -1

jal fibinnoci\_rec

mv s1, a0

mv a0, s0

jal fibinnoci\_rec

add a0,s1,a0

done:

lw s0, 0(sp)

lw s1, 4(sp)

lw ra, 8(sp) # Restore return address

addi sp, sp, 12

ret

What are the advantages and disadvantages of implementing this function recursively?

It takes longer (more cycles) to find the Fibonacci number recursively. This recursive method may be easier to understand, but overall, I think that the iterative approach would be the better option.

Report cycle count of your implementation for fib\_recursive(10).

0x00000a83

How many hours did you work on the lab?

4

Please provide any suggestions for improving this lab in the future:

None. Really insightful and encouraging to see that I can actually do this.