Course: CDA-4205L Professor: Yan Zhang PhD

Assignment: Lab 4 Due: 10/27/21 at 11:59 pm

Description and Instructions

For this lab you will be looking at the RISC-V instruction set covered in chapters 2-4 of your textbook.

You will need to complete the following questions and show all work when necessary. You will also have 2 coding portions. For the coding portions you are required to create a lab report outlining any equation and calculations you performed in order to code your solutions. You will also need to answer some questions regarding your solution and provide some screenshots in your report. Feel free to use this document as a foundation for your report or create your own. All solutions need to be typed, No hand written solutions will be accepted.

To submit you will need to upload a single PDF file lab report which will include the answers to the questions below as well as sections outlining your solutions for the coding portion. You will also need to provide a section that includes the code you wrote. Make sure that the code is properly formatted and runs properly in the RISC-V simulation we covered in class. The link is provided below.

RISC-V Simulator

All submissions should be done through Canvas by the due date or will be subjected to the penalties outlined in the syllabus. We encourage collaboration, however, you must submit your own original work must be submitted and cheating will not be tolerated. Your solutions will need to follow strict adherence to the RISC-V coding style. This means that you your solutions should be case sensitive, if commenting make sure you use // to represent the commented section. A new line will be associated with a new line of code and use of indentation is needed to separate label, instructions, and registers (both destination and source).

RISC-V Cache Performance

(2pts) 1. Assume the following:

– Miss rate of an instruction cache is 2%

– Miss rate of data cache is 5%

– A processor has a CPI of 2 cycles without any memory stalls (perfect cache)

– Miss penalty of 120 cycles for all misses

– Frequency of all loads and stores is 40%

(1 pts) 1.a Determine how much faster a processor would run if it had a perfect cache compared to one that had misses (use CPU time for your comparison).

Let's assume that some IC and Clock cycle time
Instruction miss cycles per instruction=0.02*120=2.4cycles per instruction

$$Data\ miss\ cycles = .4 * .05 * 120 = 2.4\ cycles\ per\ instruction$$

$$CPI_{total} = CPI_{perfect!} + Instuction\ miss\ cycles + Data\ miss\ cycles$$

$$CPI_{total} = 2 + 2\ .4 + 2.4 = 6.8\ cycles$$

CPU time for a perfect cache=2*IC*Clockcycletime
CPU time with misses=IC*6.8* Clockcycletime
CPU time with misses/ CPU time for a perfect cache=3.4 faster

(1 pts) 1.b Determine how much of the processing time is spent on memory access stalls (i.e. give me a percentage of total CPI time).

4.8/6.8=70.59%

Computer Architecture Lab

(2pts) 2. Assume the following:

– 5 ns clock cycle

– Miss penalty of 120 cycles

– Miss rate of 1% misses per instruction

– cache access time including hit detection is 3 ns (i.e. hit time)

(2 pts) 1.a Assuming that the read and write miss penalties are the same and ignore other write stalls, what is the average memory access time?

$$AMAT = Hittime + Missrate * Misspenalty = 3 + .01 * 600 = 9\text{ns}$$

Coding Portion

For this section you will be writing two short programs. You will be creating an array like structure in memory that will contain the first n integers of the Fibonacci sequence. If you are unfamiliar this this sequence below is a link to a web page with more information.

Fibonacci sequence

The Fibonacci sequence is a series of numbers where a number is the addition of the last two numbers, starting with 0, and 1.

The Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 . . .

The rule for finding the n^{th} integer in the sequence is given by:

$$X_n = X_{n-1} + X_{n-2}$$

For the next question you will need to write a program to calculate the first n integers of the Fibonacci Sequence and store them in memory in a specific way. The user data starts a memory address 1024 and you will use this as the starting point of your array. Your array will need to be consecutive (i.e. the next spot in memory is the next element in the array. You should also write a general solution so that you can change n to any number you want. Please read the instructions for the problem very carefully as how you create the array will change.

Problem 3

(6pts) 3. You will need to write a program that will create an array of the first n integers of the Fibonacci Sequence. The array will need to start at memory address 1024 and fill up after that. You will need to place the sequence in order. So that the first element will be stored a memory address 1024, then next will be stored at 1032 and so on. Note that you will need to manually store the first two values of the Fibonacci Sequence (i.e. $x_1 = 0$ and $x_2 = 1$) at the correct addresses and then calculate the rest using the rule provided above.

Code for Problem 3

```
addi x27, x0,1024
add x28,x0,x0
sw x28,0(x27)
addi x27,x27,4
addi x29, x0,1
sw x29,0(x27)

addi x5, x0, 10 # n element to stored- can be changed
add a0,x0,x0 #a0will store the result of the whole sequence
beq x5,x0, exit

addi a0,x29,0 #will store the result of the whole sequence
addi x30, x0,2
blt x5,x30,exit

add x31,x0,x0

addi x5,x5,-1

for:
beq x5,x0,exit
add x31,x28,x29
sw x31,4(x27)
addi x27,x27,4
add a0, x31,x0# register with result as before
add x28,x29,x0
add x29,x31,x0
addi x5,x5,-1
beq x0,x0,for

exit:
nop
```