Course: CDA-4205L Professor: Yan Zhang PhD

Assignment: Lab 5 Due: 11/11/21 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 a 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

(3pt) 1. Assume that we have a cache with 4 blocks:

– Sequence of block addresses 0,6,4,8,2,0,2, and 0.

(1 pts) 1.a Assume that we have a Direct-Mapped Cache. Fill out the table below and determine what the miss rate of the sequence of block addresses?

First, let's determine to which cache block each block address maps:

Block Address	Cache Block
0	(0 %4)=0
2	(2%4)=2
4	(4%4)=0
6	(6%4)=2
8	(8%4)=0

Missrate=6/8=75%

Address of Memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	Miss	Memory[0]			
6	Miss	Memory[0]		Memory[6]	
4	Miss	Memory[4]		Memory[6]	
8	Miss	Memory[8]		Memory[6]	
2	miss	Memory[8]		Memory[2]	
0	miss	Memory[0]		Memory[2]	
2	hit	Memory[0]		Memory[2]	
0	hit	Memory[0]		Memory[2]	

(1 pts) 1.b Assume that we have a 2-Way Set Associative Cache. Fill out the table below and determine what the miss rate of the sequence of block addresses?

First, let's determine to which cache block each block address maps:

Block Address	Cache Set
0	(0%2)=0
2	(2%2)=0
4	(4%2)=0
6	(6%2)=0
8	(8%2)=0

Missrate=6/8=75%

Address of Memory	Hit	Contents of cache blocks after reference			
		Set0	Set 0	Set1	Set 1
block accessed	or miss	0	1	2	3
0	miss	Memry[0]			
6	miss	Memory [0]	Memory [6]		
4	Miss	Memory [4]	Memory [6]		
8	Miss	Memery[4]	Memeory[8]		
2	miss	Memory [2]	Memory [8]		
0	miss	Memory [2]	Memory [0]		
2	hit	Memory [2]	Memory [0]		
0	hit	Memory [2]	Memory [0]		

(1 pts) 1.c Assume that we have a Fully Associative Cache. Fill out the table below and determine what the miss rate of the sequence of block addresses?

The fully associative cache has four cache blocks (in a single set); any memory block can be stored in any cache block.

Miss rate=6/8=75%

Address of Memory block accessed	Hit or miss	Contents of cache blocks after reference 0	1	2	3
0	Miss	Memory [0]			
6	miss	Memory [0]	Memory [6]		
4	Miss	Memory [0]	Memory [6]	Memory[4]	
8	miss	Memory [0]	Memory [6]	Memory[4]	Memory [8]
2	Miss	Memory [2]	Memory [6]	Memory[4]	Memory [8]
0	Miss	Memory [2]	Memory [0]	Memory[4]	Memory [8]
2	hit	Memory [2]	Memory [0]	Memory[4]	Memory[8]
0	hit	Memory [2]	Memory [2]	Memory[4]	Memory [8]

Virtual Memory

(2pts) 2. Suppose we have 16-bit virtual addressed system with a page size of 256 bytes.

(1pts) 2.a How many bits will be used of page offset? How many bits are used for page indexing? Draw what the address will look like?

Offsetbits= log(256)=8
Page indexig=16-8=8

VA:
Bits[0-7]=offsetbits
Bits[8-15]=Page index

(1pts) 2.b What would be the virtual page number and page offset for the address 9000? Make sure you supply the full 16-bit address in binary.

VA: 0010 0011 0010 1000

Virtual page number: 0010 0011 In decimal 35
Page Offset: 0010 1000 In decimal 40

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 into two different arrays. One will be placed in order starting at address 1032. The second will be in reverse order and will start at $1032 + 8 * n$. Please read the instructions for the problem very carefully.

Problem 3

(5pts) 3. You will need to write a program that will create two array of the first *n* integers of the Fibonacci Sequence. The first array will need to start at memory address 1032 and fill up sequentially after that. You will need to place the sequence in order. So that the first element will be stored a memory address 1032, then next will be stored at 1040 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. The other array will need to start at memory address $1032 + 8 * n$ and fill up after that. You will need to place the sequence in reverse order. So that the x_n element will be stored a memory address $1032 + 8 * n$, $x_{n-1}$and will be stored at $1032 + 8 * (n + 1)$ and so on.

Next, you will then need to iterate through both arrays in conical order (i.e $a[0], a[1], ..., a[n-1]$. And check to see if the i^{th} element in both arrays are equal to each other. If they are equal, then you need to increment a counter. After you have iterated through both arrays you need to store the counter at memory address 1024.

Code for Problem 3

This code version jumps by 8 and stores the array overlapping each other- all memory is based on address 1032, sum of the counter is at 1024:

```
addi x27, x0,1024
add x28,x0,x0
sw x28,0(x27)
addi x27,x27,8
addi x29, x0,1
sw x29,0(x27)
addi x5, x0, 15 # n element to stored- can be changed
addi x18, x5,0
addi x18, x18,-1
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

addi x4, x0,1028
addi x25, x0,8
mult x6, x25,x5
add x6, x6,x4
sw x29,0(x6)
addi x6, x6,-8

addi x22, x0,1028
addi x23, x0,1032
addi x16, x0, 0
addi x15, x0,1024
for:
```

```
beq x5,x0,compare
add x31,x28,x29
sw x31,8(x27)
sw x31,0(x6)
addi x6, x6,-8
addi x27,x27,8
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

compare:

beq x18,x0, exit
addi x18,x18,-1
lw x12, 0(x22)
addi x22, x22, 8
lw x13, 0(x23)
addi x23,x23,8
beq x12,x13, counter

beq x0,x0,compare

counter:
addi x16, x16,1
sw x16,0(x15)
beq x0,x0,compare


exit:
nop
```

This version stores them consecutively first array start at 1032 and then array ends then the other one starts where the last array finished, base address 1032, sum at 1024:

```
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, 15# n
addi x25,x5,0

add x6, x5,x0
addi x7,x0, 8
mult x6, x7,x6
addi x6, x6,1024
sw x29,0(x6)
addi x6, x6,-4
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
addi x21,x0,0
addi x22, x0,1024
addi x16,x16,0
addi x19,x0, 1032

for:
beq x5,x0,compare
add x31,x28,x29
sw x31,4(x27)
sw x31,0(x6)
addi x6,x6,-4
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

compare:

beq x25,x0,exit

lw x18,0(x27)
lw x20,0(x19)
addi x27,x27,4
addi x19,x19,4
addi x25,x25,-1
beq x20,x18,counter
beq x0,x0,compare


counter:
addi x16,x16,1
sw x16,0(x22)
```

```
beq x0,x0,compare




exit:
nop
```