EECE7352 - COMPUTER ARCHITECTURE: HOMEWORK 2

JULIAN GUTIERREZ[1]

CONTENTS

LIST OF FIGURES

LIST OF TABLES

[1] *Department of Electrical and Computer Engineering, Northeastern University, Boston, United States*

1 PART A

1.1 Write a RISC-V assembly program for multiplication

Description: Write a RISC-V assembly program to compute the product of two integer values. The two values should be initialized in main(). The main() function should call the product(int x, int y) function, passing the two integer values as arguments. The product function should return the product of the two numbers.

1.1.1 *Develop both a non-recursive and recursive implementation of your assembly program. Submit your assembly code on Blackboard through Turnitin.*

1.1.2 *What is the largest product that can be computed in your program?*

1.1.3 *Discuss how you implemented integer multiplication, since it is not directly supported on the simulator. Discuss an alternative implementation for multiplication. Which implementation would you expect to perform better, and why?*

1.2 Write a recursive RISC-V assembly program to compute the factorial

Description: Write a recursive RISC-V assembly program to compute the factorial of a value that is initialized in main (). We provide a recursive factorial program in the c program example on Blackboard.

1.2.1 *Submit your assembly code on Blackboard through Turnitin.*

1.3 What is the largest integer value for factorial

Description: What is the largest integer value that you can compute the factorial in your program on the RV32I ISA? Explain why.

1.4 Solution

This is where you write your answers. Adding a figure as an example. You can reference it using this: Figure 1.

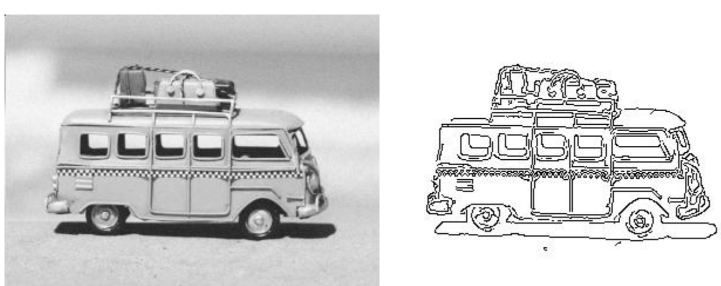

Figure 1: Original image (A) and post-processed image (B).

You should add a copy of your code in the homework report as well:

```
import numpy as np

def incmatrix(genl1, genl2):
    m = len(genl1)
```

```
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Algorithm 1: Python example

2 PART B

2.1 Profile Quick-sort

Description: For this problem you will use the qsort.c (quicksort) program provided, and you need to produce a dynamic instruction mix table (similar to Figure A.29 in your textbook) to characterize the execution of the quicksort program. You can perform this study on any architecture of your choice. There are a number of approaches you can take to produce this data. Please make sure to explain how you produced the data in your table and provide details of the tools that you used.

2.1.1 ***You could instrument the code to capture the execution frequency of each basic block, and then, using an assembly listing of the program, provide instruction counts (this is slightly imprecise, but very acceptable for this assignment).***

2.1.2 ***You could find a tracing program that can capture an instruction trace. You would then have to write a program to count individual instructions (challenging, but not impossible).***

2.1.3 ***You could find a tool out on the Internet that provides this capability already for you. While this sounds easy, it may be a bit of work to learn the particular tool you have chosen to use.***

2.2 Solution

This is where you write your answers. Adding a table as an example. You can reference it using this: Table 1.

Table 1: Your first table.

Value 1	**Value 2**	**Value 3**
α	β	γ
1	1110.1	a
2	10.1	b
3	23.113231	c

Adding a reference. The references are all in bib format. You can obtain these references from google scholar. Makes it a lot easier to add the references you need to cite on the homework. Just search for the paper you were reading on google scholar, and click on cite and search for the bibtex format. Add this reference (text) into the bib/sample.bib file and then cite the reference name inside this code. An example: Using the book [1] we answer the questions by providing Table 1. Tables can be generated with: https://www.tablesgenerator.com/.

3 PART C

3.1 Floating point benchmarks

Description: For this part of the assignment, write two different benchmark programs on your own that contain significant floating-point content. Compile the programs on X86 and generate an assembly listing of the benchmarks. Then identify 4 different floating-point instructions used in each program (a total of 8) and explain both the operands used by each instruction and the operation performed on the operands by the instruction.

3.2 Solution

4 PART D

4.1 Appendix K

Description: For this problem you will need to read through Appendix K in your text, covering a number of instructions sets, and then answer the following questions:

4.1.1 *Name 2 CISC instruction set architectures and 2 RISC instruction set architectures.*

4.1.2 *Describe 3 characteristics of the DEC Alpha instruction set.*

4.1.3 *Discuss the differences/similarities between MIPS and PowerPC in terms of how they handle conditional branches.*

4.1.4 *Given an example of how register windows work on the SPARC ISA.*

4.1.5 *In your opinion, which generation of the Intel x86 architecture was the most significant advancement from the previous generation of the ISA.*

4.2 Solution

5 PART E

5.1 Amdahl

Description: Read the Amdahl, Blaauw and Brooks 1964 paper on the IBM 360 Architecture. Given the timeframe of the paper, what do you find the most impressive feature of the architecture as described by the authors? Justify why you feel this is such a great feature. Also, discuss the representation of the various data types supported on this important ISA, and contrast it with the RISC-V.

5.2 Solution

6 PART F

6.1 Chapter 1 Problems (Extra credit)

Description: Complete problems 1.7, 1.8, 1.10, and 1.12 (only a and b) from the text.

6.2 Solution

REFERENCES

[1] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.