Programming Challenges

INTERNSHIP PROJECT REPORT

by

SRIVATHSAN NARASIMHAN

Aheesa Digital Innovations
Chennai
June 2024

Go to contents

BONAFIDE CERTIFICATE

This is to certify that this project report entitled programming challenges submitted to Aheesa Digital Innovations, Chennai is a bonafide record of work done by Srivathsan Narasimhan currently in Higher grade Secondary S6 in Glasgow, UK under my supervision from 27/05/2024 to 21/06/2024 in our Karapakkam office.

Swaroop Adusumilli VP - ASIC

Aheesa Digital Innovations, Chennai
21/06/2024

Go to contents

Declaration by Author

This is to declare that this report has been written by me. No part of the report is plagiarised from other sources. All information included from the other sources have been duly acknowledged. I aver that if any part of the report is found to be plagiarised, I shall take full responsibility for it.

SRIVATHSAN NARASIMHAN

Aheesa Digital Innovations, Chennai
21/06/2024

Go to contents

Contents:

Go to contents

Abstract

This report details a series of challenges I was given as part of the internship program and application of my thought process in solving them. The challenges required me to research various concepts in computer science and mathematics in detail and create applications using the concepts as the basis for the design. The challenges addressed include understanding the RISC-V architecture and developing Assembly programs for the RV64, finding efficient algorithms to verify prime numbers, researching and testing to find the most suitable search algorithm for numerical data, implementing a library for complex number operations, and using that library to resize images.

Go to contents

Introduction

I am currently studying High school grade Secondary-S6 in Glasgow, UK and the objective of my internship at Aheesa Digital Innovations, Chennai is to acquire an introduction to the field of Computer Architecture. Each challenge I encountered required in-depth research into fundamental aspects of computer science, enhancing my understanding of how common programming functions operate. These challenges involved rediscovering the procedures for performing basic computing tasks using logic derived from mathematical concepts taught in school.

Go to contents

Methodology

An iterative methodology was employed to address the programming challenges. Initially, comprehensive online research was conducted on the relevant areas of computer science or mathematics underlying each assignment. Following this, a preliminary algorithm was developed and subsequently refined for efficiency. The choice of platform was determined based on the programming language best suited for implementing each challenge, with the selection guided by the availability of necessary libraries and tools.

Go to contents

Learning about RISC V

Detailed study of the RISC-V (Reduced Instruction Set Computers) architecture and in particular its instruction set for the integer, atomic and multiplication extensions was imperative to solve the assigned challenges. The base instruction set covered the movement of data between registers, arithmetic operations and bitwise operations. I was able to understand the architecture by reading the RISC-V User Level Manual and asking ChatGPT clarifications on the terminology specific to the architecture of RISC-V. The RISC-V User Level Manual provided me a deep understanding of the instruction formats used, introduced me to the C-register convention, and how data is handled in registers in RISC-V processors.

To implement programs on RISC-V processors I followed the SHAKTI programmer Manual to understand the syntax of instructions used in assembly programming for RISC-V, as well as how to apply pseudo codes to make my program simpler. Through the course of my study, I gained a solid foundational understanding about the RISC-V architecture and gained an insight into how its minimal instruction set can be used to create complex programs.

Go to contents

Small programs in Assembly

The first challenge was to apply the instructions I learned from the SHAKTI Programmer Manual and implement new solutions, for the small program examples in the manual. These exercises encompassed the creation of conditional loops, switch statements, if-elseif-else constructs, Bubble sort algorithms, and the computation of Fibonacci numbers. Utilising the Venus RISC-V emulator, I wrote and tested assembly code using memory to store the output. Subsequently, I was tasked with replicating these programs on the NEXYS A7 FPGA board, introducing a new layer of complexity. Interfacing with a physical test platform necessitated the installation of Linux on my laptop to interact with the board, requiring me to learn commands to navigate the Linux terminal for code editing, compilation, execution and debugging. A significant hurdle arose when I discovered that while the Venus simulator simulated an RV32 core, while the FPGA board employed an RV64 architecture, resulting in numerous memory-related errors. However, through analysis of hexadecimal register values and reference to the SHAKTI Programmer Manual for error code interpretation, I successfully solved these issues.

Go to contents

Algorithm to find a Prime Number

The next challenge was to create the most efficient algorithm to verify a prime number. This is a very important algorithm as very large prime numbers are used extensively in other algorithms to generate unique values. The algorithm needs to be very efficient and should not take long to verify even very large prime numbers. The initial solution was to check if the number was divisible by all the numbers less than it. But this method is very inefficient due to its repeated division by multiples of other numbers. Thinking farther, most numbers are divisible by 2, 3, 5 and 7 but some like 121 are only divisible by their square root, so to determine if a number is prime we need to check if it is divisible by 2, 3, 5 or 7 and if it is a perfect square. In theory this is the best algorithm but the process to compute a square root is very tedious especially in assembly where we would have to create an algorithm for that as well.

The final method proposed was to divide by odd numbers till the square of the number is greater than the number we are checking. This method is simple, efficient and can easily be implemented into the NEXYS A7 board. This initial challenge highlights the importance of carefully checking each line of code and ensuring that each line has a purpose for the end goal, so that the most efficient solution is used. Reducing calculation time and preserving the limited resources of a processor.

Go to contents

Most suitable Search algorithm

The next challenge was to address the inherent inefficiency in the prime verification program regarding the repeated verification of numbers. Everytime a number is entered to be verified we have to process the entire algorithm. The result of verifications could be stored to address this flaw, but it requires a fast search algorithm to be effective. Traditional search algorithms proved inadequate in achieving consistent search times across all primes. Despite exploring fast algorithms such as jump, binary, and interpolation search, the recurring issue of inconsistent search times persisted. However, by using hashing we can facilitate efficient data storage and retrieval by mapping each data element to a specific index using mathematical functions. Various hashing techniques were evaluated, ranging from rudimentary methods like storing numbers at their respective indices to more sophisticated algorithms like xxHash and MurmurHash. Notably, xxHash 32 emerged as the optimal choice for storing the initial 1000 prime numbers. However Implementation of xxHash 32 necessitated reference to forums and clarifications through ChatGPT, to implement into a C program. So a premade option in C++ was used in the final program, the STL unordered_map provided a streamlined approach to hash table implementation, enabling seamless addition and retrieval of values across different data types.

Go to contents

Library for Complex Number Operations

Initially research was done into performing fundamental arithmetic operations—addition, subtraction, multiplication, and division—with complex numbers. After analysis a pseudocode algorithm was devised for each operation. It was decided to also include the functions to find the conjugate and inverse of complex numbers as they are both involved in the division process. Given the composite nature of complex numbers comprising real and imaginary components, a decision was made to use structure variables accommodating two independent float variables, as the optimal approach to represent complex numbers inside the program. A header file was implemented, housing functions tailored to execute each arithmetic operation. The library was tested using an external C application getting inputs and calling the appropriate function and displaying the output.

Go to contents

Resizing images

The challenge to develop a program to resize images was very daunting, and required extensive research into the composition of image formats in particular jpegs. I was also given the clue to use the complex number library developed previously, this provided a way to represent pixels using the same structure as the complex numbers, allowing me to perform numeric operations. I employed the multiplication function to scale back the new pixel coordinates to where they would be in the source image. This method was incomplete as most scaled back values were not exact pixel coordinates, so pixel scaling algorithms were researched and linear interpolation was employed to decide the colour of intermediate pixels in the new image. To copy colour data from an image we need to store its data in an array, this became a major hurdle as I tried to analyse where colour data is stored in the jpeg format. Following online advice I decided to use the libjpeg library to parse through the header of a jpeg image and extract the necessary information. The 2 part nature of both pixel coordinates and complex numbers allowed pixel coordinates to also be manipulated using the same library, reflecting how properties of mathematical concepts can be used to infer algorithms even in unrelated topics.

Finding Coefficients of Expressions

Go to contents

The next task was to determine the coefficients of a & b, in an equation with the form $y = ax^2 + b$ with using the points (1, 5), (2, 14) and (3, 29). This was initially done on paper using simultaneous equations, but the task was to come up with a general solution. The general solution that was followed also used simultaneous equations but we needed n + 1 expressions, where n is the degree of the polynomial. This created a square matrix of n + 1 terms of x, by n + 1 expressions, this matrix multiplied by a vector of coefficients would give us a vector for y values. To find the coefficients we need to divide the vector containing the y values by the matrix of x values, the resulting vector will contain all the coefficients in increasing order of x powers. But the division of matrices is not an inherent process, we need to find its inverse then multiply. The process for calculating an inverse matrix involves the Gaussian Elimination method, where we perform row calculations on the matrix in order to make the current matrix into the identity matrix, and also copy the same operations on an existing identity matrix. This manipulated identity matrix contains the inverse matrix after all the row operations have been finished. The multiplication between the inverse matrix and a vector was done by taking the dot product between each row in the matrix and the vector. The final program was implemented in C and returns all the coefficients of any polynomial function when given n + 1 points.

Gradient Descent in Excel

Go to contents

A similar task was given but to implement in Excel. And this time the system should be able to get multiple coordinates and still only give a quadratic output. To accomplish this a different algorithm was used, called gradient descent. This method provides us with a very general way to guess the function, for any set of points, to a very high degree of accuracy. This can be done, as first we guess a random value of a, b & c, then try the given x values. Using the calculated y value we find our error rate by using a loss function, in this specific example Mean Squared Error. Which is the mean difference squared between calculated and actual y values. Afterwards, we can partially differentiate a, b & c in the loss function, and calculate the gradient, giving us the direction to move the value of a, b or c towards a value which minimises the loss function. This process of optimising the value of a, b & c is independent and so can be parallelised. The equations used was implemented in an excel sheet as formulas then a, b & c was found by dragging the formula down, until the MSE (Mean Squared Error) was below 1×10^{-6}. Giving very accurate values for the coefficients.

Presentation on UAT

Go to contents

The final task was to use all the information gathered in the other challenges as the research for a presentation that was about UAT, or The Universal Approximation Theorem. UAT is a pillar stone in Neural Networks which it both explains and is the reason for the recent research. As part of the presentation I had to learn what neural networks are, how they operate, what are their applications, how they are trained, and also explain how different types of neural networks operate. During the research for this presentation I was able to use knowledge learnt from previous challenges giving me useful background information. Allowing me to explain clearly when presenting to the team.

Concluding Remarks

Go to contents

My internship project has been a transformative experience. With the support of online resources and valuable guidance from my supervisor and colleagues. I have been able to solve the challenges, improving my skills in developing efficient programs and gained invaluable insights into the intricacies of Computer Architecture design. The lessons learned during this internship have profoundly influenced my problem-solving approach, equipping me with the ability to quickly tackle complex tasks in a methodical manner.

I am sincerely grateful to Mr Swaroop Adusumilli and the entire team for providing me with this enriching opportunity. A special thanks to Ms Guathami for mentoring me throughout this internship, giving guidance on how to solve the challenges presented to me, as well as to Mr Sridharan Mani for organising this internship program, and designing the assignments. This experience has not only expanded my technical knowledge but has also instilled in me a deep appreciation for continuous learning and growth in the dynamic field of computer architecture.

References

Go to contents

The resources I used to solve these challenges was:

1. C programming *by Mr Sridharan Mani*
2. Linux Shell Scripting Tutorial v2.0 *by Vivek Gite*
3. RISC-V ASSEMBLY LANGUAGE Programmer Manual part I *by SHAKTI Development team*
4. The RISC-V Instruction Set Manual Volume I: User-Level ISA *by Andrew Waterman & Krste Asanovic*

Go to contents