****

**Computer Organisation And**

**Architecture Lab**

**RV85**

**Semester:** III

**Batch:** B8

**Submitted By:**

Rhythm Gupta – 17103310

Vidhi Jain – 17103323

Tejas Pandey – 17103319

INTRODUCTION

**Objective of the project:**

We are creating a simulator of a processor like JC62 which can perform various functions like loading, storing, adding and many more. The

code will be written in C++. The user can enter the code and the processor will convert it into machine language and perform the operation accordingly.

There will be different registers to which have various functionalities:-

1. Read the data

2. Load the data

3. Store the data

4. Add

5. Subtract

6. Multiply

7. Write the data

8. Divide

9. Complement

10. Increment

11. Decrement

12. Jump

13. Halt

14. Jump on negative

All these operations will have a different function in C++.

There will be an accumulator and ALU where different instructions will get executed.

**Theory:**

A processor, or "microprocessor," is a small chip that resides in computers and other electronic devices. Its basic job is to receive input and provide the appropriate output. While this may seem like a simple task, modern processors can handle trillions of calculations per second.

The central processor of a computer is also known as the CPU, or "central processing unit." This processor handles all the basic system instructions, such as processing mouse and keyboard input and running applications. Most desktop computers contain a CPU developed by either Intel or AMD, both of which use the x86 processor architecture. Mobile devices, such as laptops and tablets may use Intel and AMD CPUs, but can also use specific mobile processors developed by companies like ARM or Apple.

Modern CPUs often include multiple processing cores, which work together to process instructions. While these "cores" are contained in one physical unit, they are actually individual processors. In fact, if you view your computer's performance with a system monitoring utility like Windows Task Manager (Windows) or Activity Monitor (Mac OS X), you will see separate graphs for each processor. Processors that include two cores are called dual-core processors, while those with four cores are called quad-core processors. Some high-end workstations contain multiple CPUs with multiple cores, allowing a single machine to have eight, twelve, or even more processing cores.

Besides the central processing unit, most desktop and laptop computers also include a GPU. This processor is specifically designed for rendering graphics that are output on a monitor. Desktop computers often have a video card that contains the GPU, while mobile devices usually contain a graphics chip that is integrated into the motherboard. By using separate processors for system and graphics processing, computers are able to handle graphic-intensive applications more efficiently.

**COMPUTER ARCHITECTURE SIMULATOR:**

A computer architecture simulator is a program that simulates the execution of computer architecture.

Computer architecture simulators are used for the following purposes:

• Lowering cost by evaluating hardware designs without building physical hardware systems.

• Enabling access to unobtainable hardware.

• Increasing the precision and volume of computer performance data.

• Introducing abilities that are not normally possible on real hardware such as running code backwards when an error is detected or running in faster-than-real time.

**CATEGORIES:**

Computer architecture simulators can be classified into many different categories depending on the context.

• Scope: Microarchitecture simulators model the microprocessor and its components. Full-system simulators also model the processor, memory systems, and I/O devices.

• Detail: Functional simulators, such as instruction set simulators, achieve the same function as modeled components. They can be simulated faster if timing is not considered. Timing simulators are functional simulators that also reproduce timing. Timing simulators can be further categorized into digital cycle-accurate and analog sub-cycle simulators.

• Workload: Trace-driven simulators (also called event-driven simulators) react to pre-recorded streams of instructions with some fixed input. Execution-driven simulators allow dynamic change of instructions to be executed depending on different input data.

**INSTRUCTION SET SIMULATOR:**

An instruction set simulator (ISS) is a simulation model, usually coded in a high-level programming language, which mimics the behavior of a mainframe or microprocessor by "reading" instructions and maintaining internal variables which represent the processor's registers.

OPERATIONS

* LDA

It has operation code 20. The address is asked from the user and the data in that address is loaded in the Accumulator. If the data is negative, the negative flag is set and the data is converted to positive.

* STA

It has operation code 21. The address is asked from the user and the data in the Accumulator is loaded in that address.

* RED

It has operation code 10. The address is asked from the user and then the data is asked too. The data entered by the user in stored at that address.

* WRI

It has operation code 11. The address is asked from the user and the data stored at that address is shown to the user.

* ADD

It has operation code 30. The address is asked from the user and the data stored at that address is added with the data already in the Accumulator. If the result is negative, the negative flag is set.

* SUB

It has operation code 31. The address is asked from the user and the data stored at that address is subtracted with the data already in the Accumulator. If the result is negative, the negative flag is set.

* MUL

It has operation code 33. The address is asked from the user and the data stored at that address is multiplied with the data already in the Accumulator.

* DIV

It has operation code 32. The address is asked from the user and the data stored at that address is divided from the data already in the Accumulator.

* CMA

It has operation code 34. It performs 1’s complement of the data in the Accumulator by subtracting 255.

* INR

It has operation code 35. It increments the data in the Accumulator by 1.

* DCR

It has operation code 36. It decrements the data in the Accumulator by 1.

* JMP

It has operation code 40. The line is asked from the user and the Program Counter is initialized with that line number.

* JPN

It has operation code 41. The line is asked from the user and the Program Counter is initialized with that line number if the negative flag is set.

* HLT

It has operation code 99. This instruction terminates the program.

SCREENSHOTS







