OS PROJECT REPORT

LC-3 Lightweight Virtual Machine (Trap/Interrupt Handler thru system calls)



**Group Members:**

21K-3421 (Sana Khalid)

21K-4507 (Virkha Kumari)

21K-3409 (Fizza Rashid)

# 

# 

# Pre-Requisites:

* Ubuntu Terminal
* 128 KB memory space

# Objectives:

* Create a fully functional LC-3 virtual machine that can run any LC3 assembly programs.
* Create the trap/interrupt handler of Input/output behavior on terminal thru system calls.
* Run the Virtual Machine as the child process independently, so that in future multiple lc-3 vm programs can be supported by the single program.
* Build/compile the LC-3 VM on Ubuntu terminal that allows users to load and run their desired lc-3 programs, view the output, and interact with the machine.

# INTRODUCTION:

A VM or Virtual Machine is a program that acts like a computer. It simulates a CPU along with a few other hardware components, allowing it to perform arithmetic, read and write to memory, and interact with I/O devices, just like a physical computer. Most importantly, it can understand a machine language which you can use to program it.

Our Light-weight VM will stimulate a fictional computer called LC-3 for game like Rogue/2048 to be run/executed thru our program. It has a simplified instruction set compared to x86, but contains all the main ideas used in modern CPUs. It basically uses RISC instruction set with 10 opcodes.

# 

# BACKGROUND:

The LC-3 (Little Computer 3) is a simple computer architecture designed for educational purposes. It was developed by Yale N. Patt and Sanjay J. Patel in the late 1990s, and it is used in many computer architecture courses as a teaching tool for understanding the basic concepts of computer organization and design.

The LC-3 architecture is a simplified version of the von Neumann architecture, which is the basis for most modern computer systems.

The LC-3 instruction set consists of 16-bit instructions that can perform simple operations such as loading and storing data, arithmetic and logical operations, and branching, which we tried to include in our project as well through case structure implementations.

# PLATFORM AND LANGUAGES:

Little Computer 3, or LC-3, is a type of computer educational programming language, an assembly language, which is a type of low-level programming language.

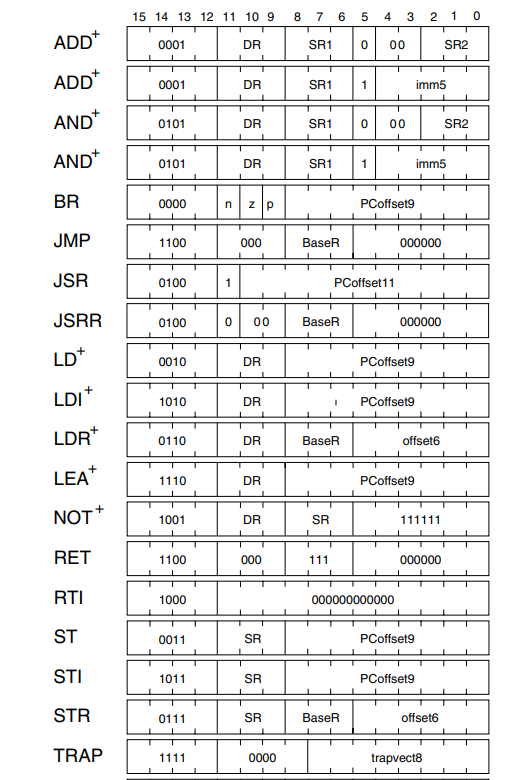
We have used the platform -> Ubuntu gedit to write our program

Language used is ‘C’.

# METHODOLOGY:

For LC-3 architecture, we needed 65336 locations with each taking 16 bits so total of 128KB memory was provided by our program to any LC-3 assembly program to be executed. We have declared 8 registers to perform ALU or other operations+1 program counter+1 condition flag+1 register counter -> each register taking 16 bits so total of 22 bytes were reserved for register allocation, while rest used for instruction execution later. We have also declared three condition flags: positive, negative, zero to indicate the condition of the last instruction executed to implement conditional jumps/branching in code.

Total of 10 opcodes (rti and res are useless in our program) are supported by our program with additional of trap handling which has been included as op-code as well as it is also part of instruction which needs to be executed. As each instruction is allotted 16 bits so 12 to 15 inclusive bits are for op-code and 0 to 11 are for parameters. This all instruction set has been implemented as below:



In that way, we have managed the memory of our program.

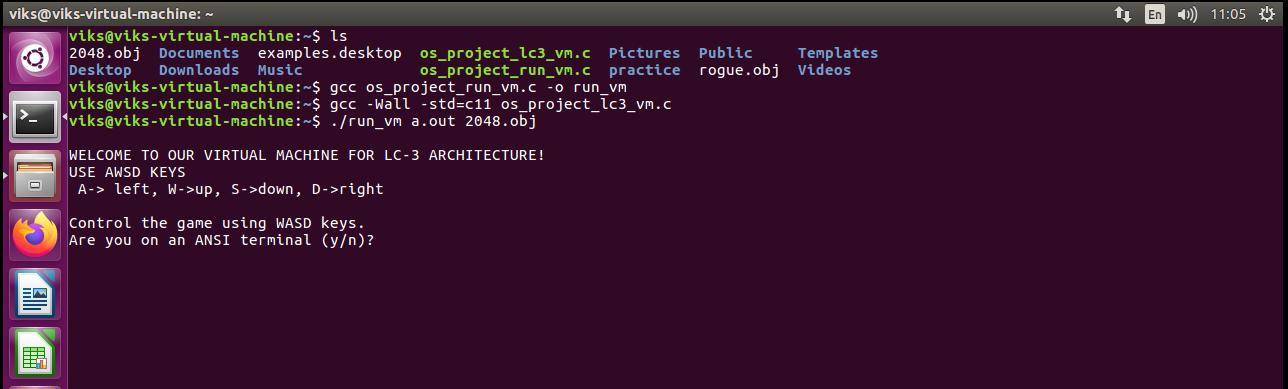
Now let’s look at the trap handling provided by our program. Firstly, we have disabled interrupts, to make sure that no preemption occurs. Furthermore, we have allocated addresses to the keyboard data/status and whenever, the address where data is stored==keyboard status/data then we take an input of character (repeat for string) thru system call ‘getc ()’. Also, 6 types of interrupts/traps are handled thru allocating addresses and if the instruction includes trap, then accordingly that trap handler is executed thru case structure used.

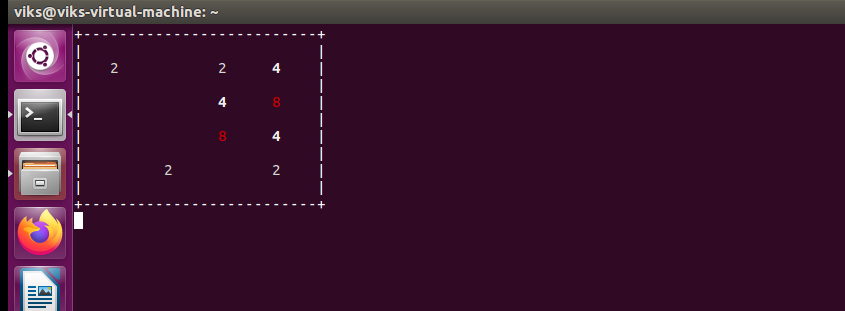
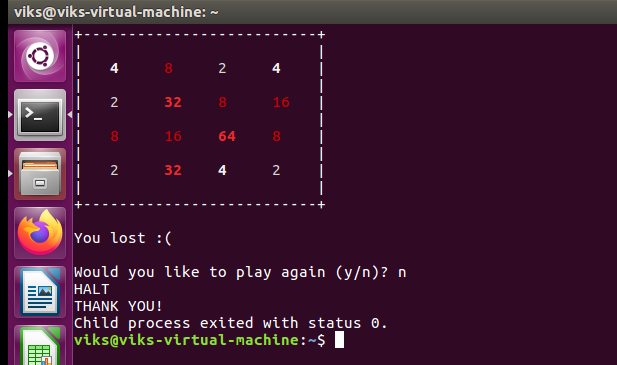
Additionally, we have read the image file of the .obj file of lc-3 architecture thru system calls, so we also have allowed the OS to intervene in the execution, as OS services are used throughout our program.

Lastly, other c program has been used to call and execute the LC-3 VM as the child process with the .obj sent as well, to indicate which lc-3 program has to be executed by the VM. We have implemented execution of the VM as the child process, so multiple VMs/programs can be executed independently and supported by the single program ‘os\_project\_run\_vm.c’.

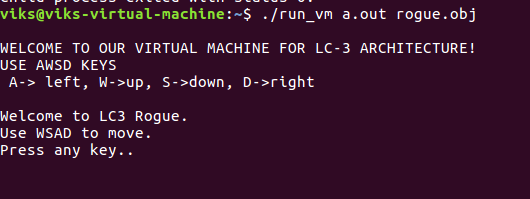
# RESULTS/OUTPUTS:

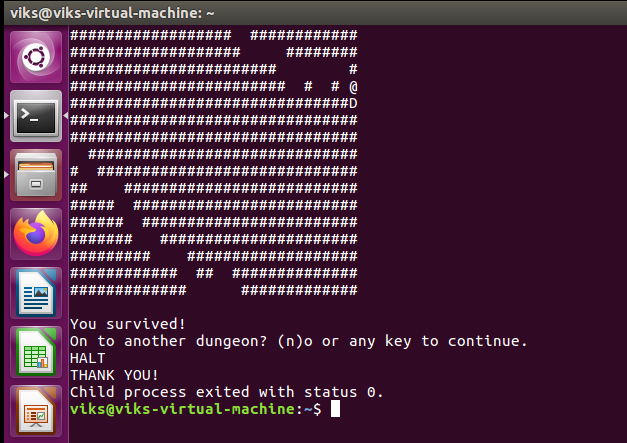
* **2048 game**



* **Rogue game**





# CONCLUSION:

In conclusion, we have tried to implement OS concepts to build lightweight virtual machine, but there are due to some improvements which can be improvised in future e.g., as we discussed multiple childs -> multiple VMs. In brief, we have managed memory, handled traps, execution of LC-3 computer, and I/O behavior of terminal to build fully-functional LC-3 Lightweight Virtual Machine.