ELEMENTS OF COMPUTING SYSTEMS [19AIE101] S1 B. TECH CSE (AIE)

A Project Report

*Submitted b*y

GROUP 8 AIE B BATCH

**ROLL NUMBER NAME**

AM.EN.U4AIE21109 AKSHAY KRISHNAN T

AM.EN.U4AIE21119 ASWIN US

AM.EN.U4AIE21148 NIRANJAN PRASANTH  
AM.EN.U4AIE21147 NAVNEETH SURESH

AM.EN.U4AIE21144 MUHAMMED SHAJAHAN



AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM AMRITAPURI 690 525

# Question 2:

# 

**When user presses C on the keyboard, the above pattern should appear. when no key is pressed Screen should be blank**

**PSEUDO CODE:**

KEYBOARD:

address = SCREEN + 14

rowc = 256

vertc = 14

total = 8192

if (KDB == 67) goto X1

goto WHITESET

RESTORE:

rowc = 256

vertc = vertc + 1

address = vertc + SCREEN

if (15 - vertc == 0) goto X2

if (16 - vertc == 0) goto X3

if (17 - vertc == 0) goto X4

X1:

RAM[address] = -1

rowc = rowc - 1

if (rowc == 0) goto RESTORE

address = address + 32

goto X1

X2:

RAM[address] = -1

rowc = rowc - 1

if (rowc == 0) goto RESTORE

address = address + 32

goto X2

X3:

RAM[address] = -1

rowc = rowc - 1

if (rowc == 0) goto RESTORE

address = address + 32

goto X3

X4:

RAM[address] = -1

rowc = rowc - 1

if (rowc == 0) goto RESTORE

address = address + 32

goto X4

ACROSS:

address = SCREEN + 3072

rowc = 64

vertc = 0

goto REPEAT

REPEAT:

RAM[address] = -1

rowc = rowc - 1

if (rowc == 0) goto RESTORE2

address = address + 32

goto REPEAT

RESTORE2:

vertc = vertc + 1

if (vertc - 32 == 0) goto KEYBOARD

address = vertc + SCREEN + 3072

rowc = 64

goto REPEAT

WHITESET:

address = SCREEN

goto WHITEPART

WHITEPART:

RAM[address] = 0

address = address + 1

total = total - 1

if (total > 0) goto WHITEPART

goto KEYBOARD

# HACK ASSEMBLY CODE:

# (KEYBOARD)

# @SCREEN

# D = A

# @14

# D = D + A

# @address

# M = D

# @256

# D = A

# @rowc

# M = D

# @14

# D = A

# @vertc

# M = D

# @8192

# D = A

# @total

# M = D

# @67

# D=A

# @KBD

# D = D - M

# @X1

# D; JEQ

# @WHITESET

# 0; JMP

# (RESTORE)

# @256

# D = A

# @rowc

# M = D

# @vertc

# M = M + 1

# D = M

# @SCREEN

# D = D + A

# @address

# M = D

# @15

# D = A

# @vertc

# D = D - M

# @X2

# D; JEQ

# @16

# D = A

# @vertc

# D = D - M

# @X3

# D; JEQ

# @17

# D = A

# @vertc

# D = D - M

# @X4

# D; JEQ

# (X1)

# @address

# A = M

# M = -1

# @rowc

# M = M - 1

# D = M

# @RESTORE

# D; JEQ

# @32

# D = A

# @address

# M = M + D

# @X1

# 0; JMP

# (X2)

# @address

# A = M

# M = -1

# @rowc

# M = M - 1

# D = M

# @RESTORE

# D; JEQ

# @32

# D = A

# @address

# M = M + D

# @X2

# 0; JMP

# (X3)

# @address

# A = M

# M = -1

# @rowc

# M = M - 1

# D = M

# @RESTORE

# D; JEQ

# @32

# D = A

# @address

# M = M + D

# @X3

# 0; JMP

# (X4)

# @address

# A = M

# M = -1

# @rowc

# M = M - 1

# D = M

# @ACROSS

# D; JEQ

# @32

# D = A

# @address

# M = M + D

# @X4

# 0; JMP

# (ACROSS)

# @SCREEN

# D = A

# @3072

# D = D + A

# @address

# M = D

# @64

# D = A

# @rowc

# M = D

# @0

# D = A

# @vertc

# M = D

# @REPEAT

# 0; JMP

# (REPEAT)

# @address

# A = M

# M = -1

# @rowc

# MD = M - 1

# @RESTORE2

# D; JEQ

# @32

# D = A

# @address

# M = M + D

# @REPEAT

# 0; JMP

# (RESTORE2)

# @vertc

# M = M + 1

# @32

# D = A

# @vertc

# D = D - M

# @KEYBOARD

# D; JEQ

# @vertc

# D = M

# @SCREEN

# D = D + A

# @3072

# D = D + A

# @address

# M = D

# @64

# D = A

# @rowc

# M = D

# @REPEAT

# 0; JMP

# (WHITESET)

# @SCREEN

# D = A

# @address

# M = D

# @WHITEPART

# 0; JMP

# (WHITEPART)

# @address

# A = M

# M = 0

# @address

# M = M + 1

# @total

# MD = M - 1

# @WHITEPART

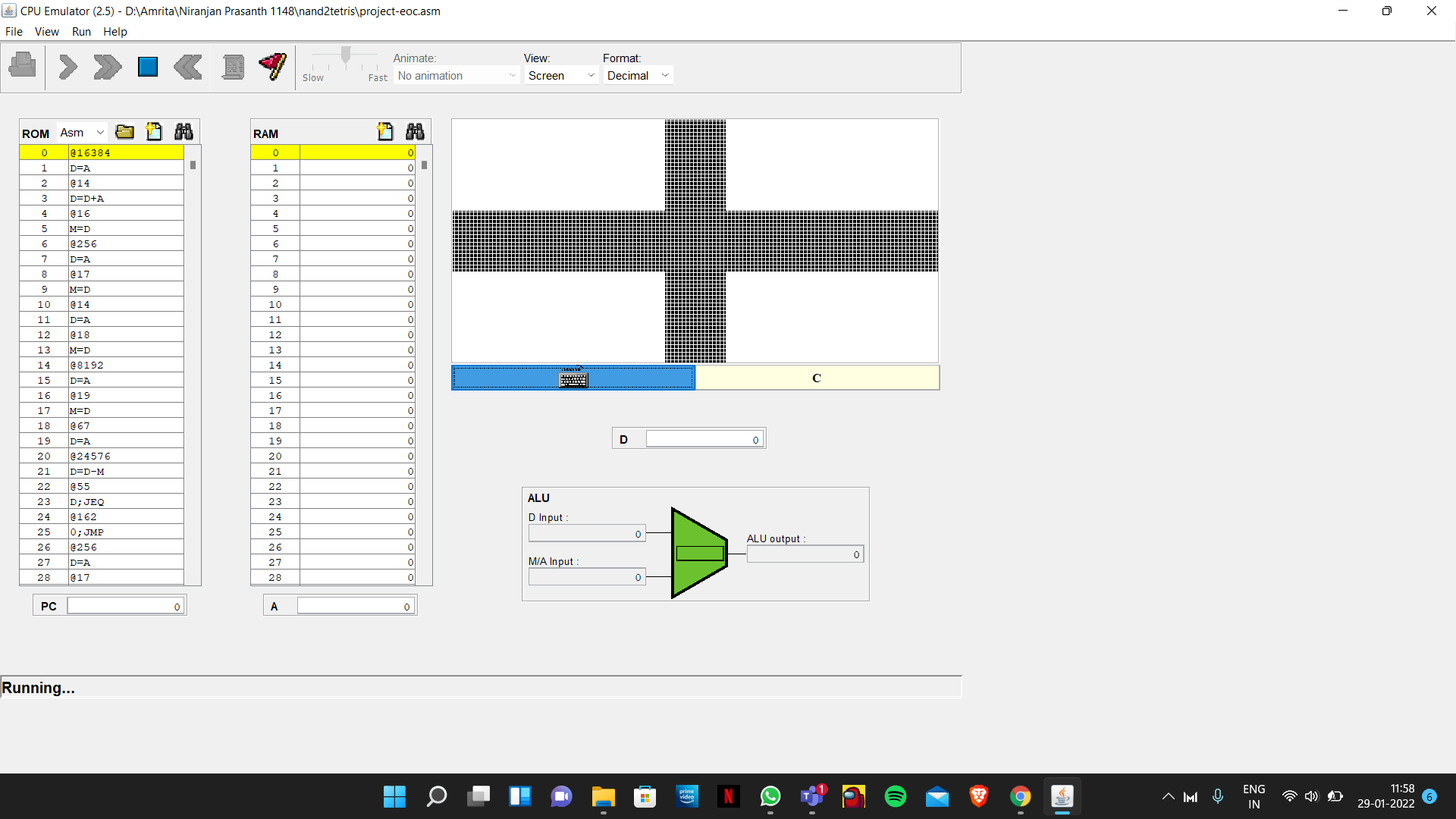
# D; JGT

# @KEYBOARD

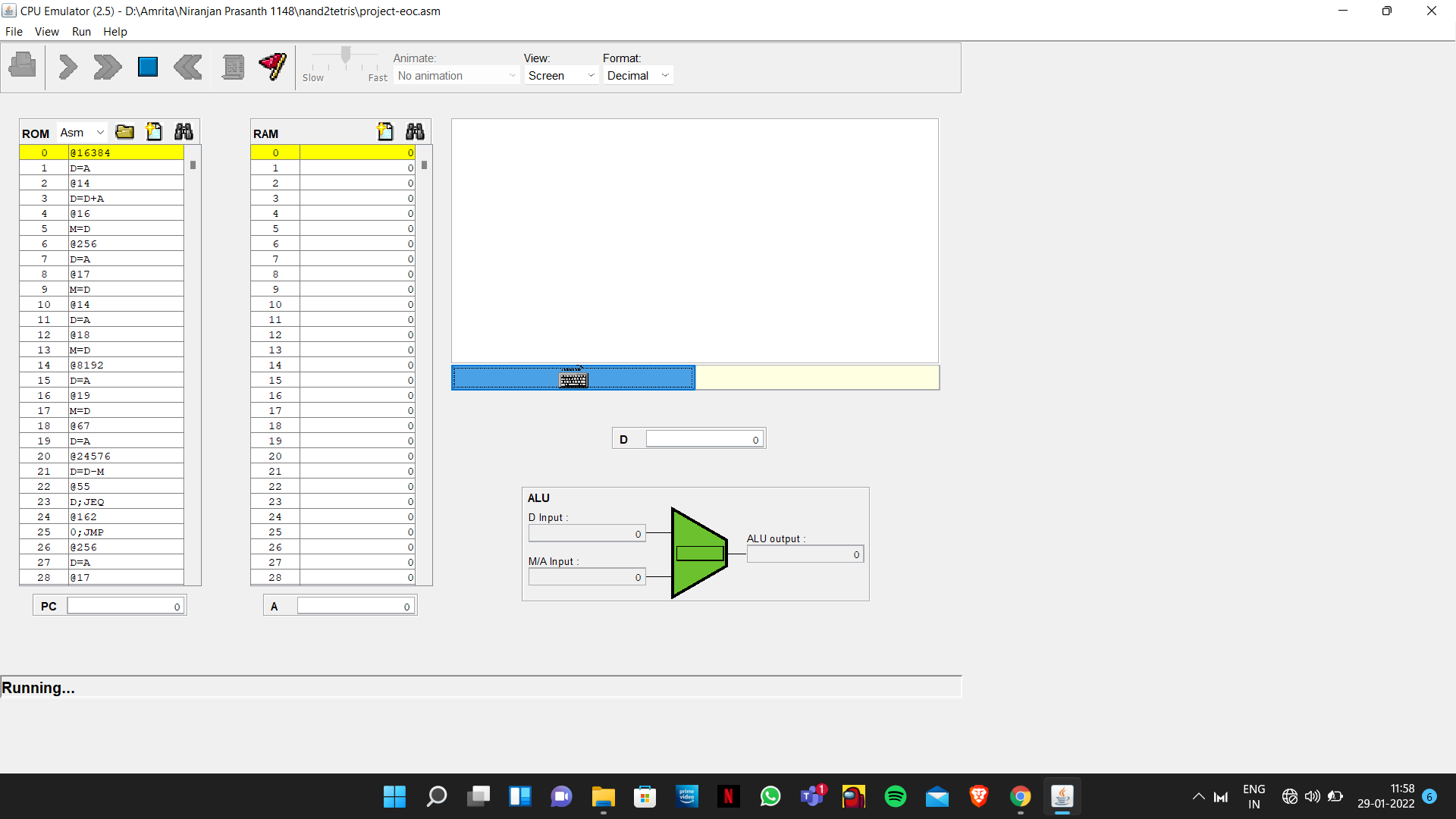
# 0; JMP

# OUTPUT:

BY PRESSING THE KEY ‘C’



WITHOUT PRESSING THE KEY ‘C’



# INSIGHTS ;

# 

# Learnt how to turn the pixel on and off.

# Learnt about screen and keyboard mapping by manipulating the bits.

# We learnt on executing the syntax of hack assembly language programmes.

# Came across the challenges of the low level coding and the huge difficulties to understand the syntax

* Labels, variables, and the two fundamental types of instructions, A and C, were all comprehended into the project.

# How programs are terminated

# Understanding of the implementation of registers in our code helped to create a better understanding of concept

* learnt various functions and applications of A and C instructions.
* We also learned how to run the .ASM code and get the appropriate output using the CPU Emulator and also to create a pseudo code.

# STUDENT INVOLVEMENT:

# At first, all of us learnt about hack assembly code and how to manipulate screen using that. Then we made a group in teams and did most of the works being in the group call(even though we divided on whom to focus more on).

AKSHAY KRISHNAN T(21109)– worked on pseudo code and documentation of project

ASWIN US(21119) – worked on pseudo code and documentation of project

NIRANJAN PRASANTH(21148) - worked on hack code and pseudocode  
NAVNEETH SURESH(21147) – worked on hack code and pseudocode

MUHAMMED SHAJAHAN(21144) - worked on hack code pseudocode