**ELEMENTS OF COMPUTING SYSTEMS**

**PROJECT REPORT**

**Group 7**

Advaith PR (AM.EN.U4AIE21106)

Mahadev Sunil (AM.EN.U4AIE21141)

Nandana Ajoy (AM.EN.U4AIE21145)

S Anand (AM.EN.U4AIE21155)

Shreya Pulluri (AM.EN.U4AIE21159)

**

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITAPURI 690 525

January 2022

**Project Question**:

Given 2 numbers in RAM[0] and RAM[1], perform RAM[0]/RAM[1] store the quotient in RAM[2] and the remainder in RAM[3].

* **PSEUDO CODE**

RAM[2] = 0 //set RAM[2] to 0

RAM[3] = 0 //set RAM[3] to 0

set D = RAM[0]

if (D=0) go to END

else

RAM[store] = D = RAM[0]

LOOP //Loop begins

if (RAM[0] - RAM[1]<0) go to REMAINDER

else if (RAM[0] - RAM[1] = 0) increment RAM[2] by 1 for getting the quotient and go to

EVENLY

go to LOOP (LOOP continues)

REMAINDER //When RAM[0] is not completely divisible by RAM[1]

D = D + RAM[1]

RAM[3] = D //assigning the value of remainder to RAM[3]

EVENLY //When RAM[0] is completely divisible by RAM[1]

D = RAM[store]

RAM[0] = D //restoring the value of RAM[0]

END

go to END

* **HACK ASSEMBLY CODE**

// Divides R0 by R1 and stores the quotient in R2 and remainder in R3

// (R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.)

@R2

M = 0

@R3

M = 0

@R0

D = M

@END

D, JEQ

@store

M = D // store to restore

(LOOP)

@R1

D = D - M

@REMAINDER

D, JLT

@R2

M = M + 1

@EVENLY

D, JEQ

@LOOP

0, JMP

(REMAINDER)

@R1

D = D + M

@R3

M = D

(EVENLY)

@store

D = M

@R0

M = D

(END)

@END

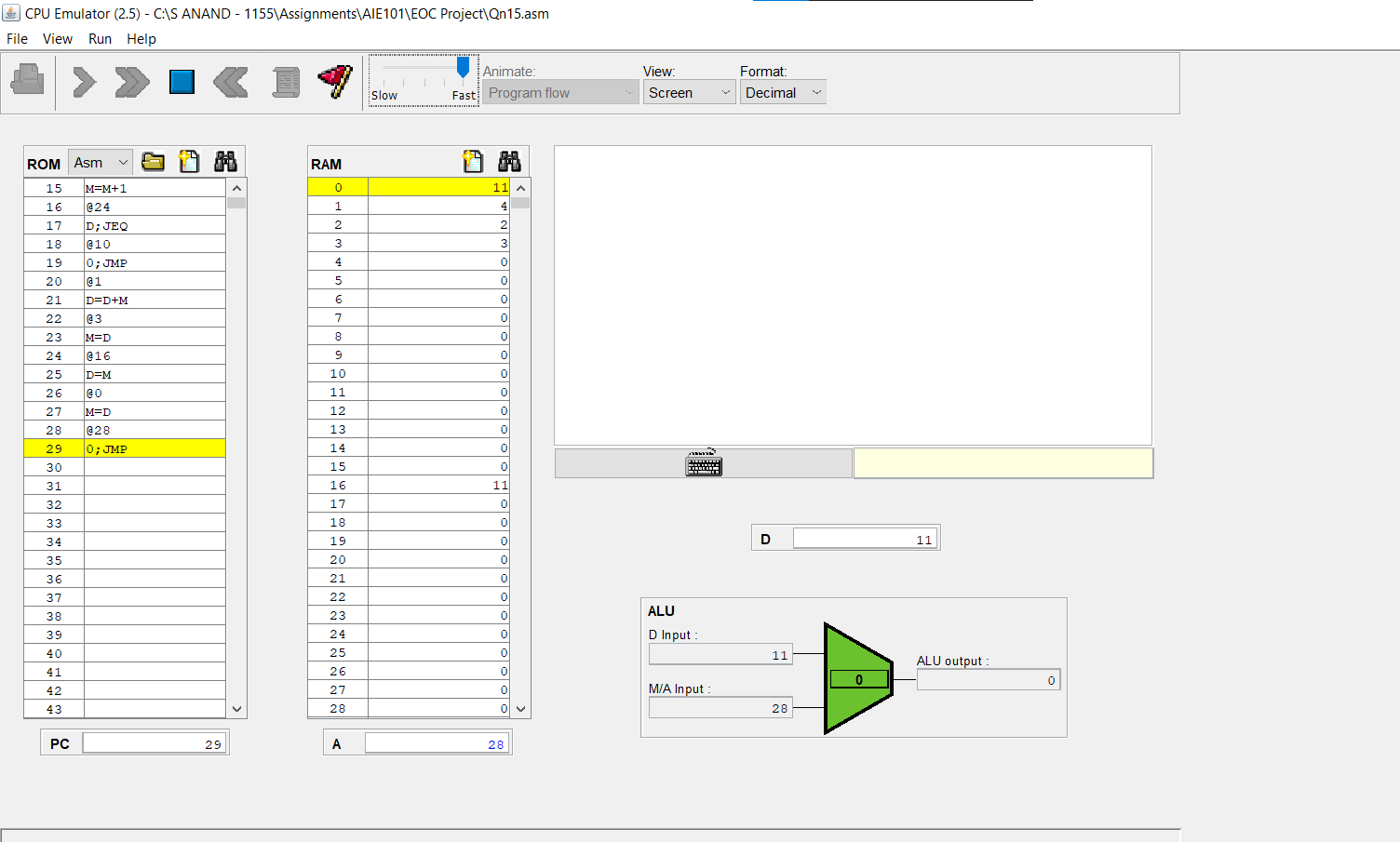
0, JMP

* OUTPUT FROM CPU EMULATOR

Inputs given:

RAM[0] = 11

RAM[1] = 4



Output formulated by CPUEmulator:

Quotient = RAM[2] = 2

Remainder = RAM[3] = 3

* INSIGHTS ABOUT HACK ASSEMBLY CODE WHILE IMPLEMENTING THE PROJECT
* To find out the solution to the given question, we had to have a good understanding about the two types of instructions used in Hack Machine Language, namely A and C instructions. The jump statements in C instructions were used widely in the code.
* For understanding the working of the code, we had to have a thorough insight on the working of the RAM and its registers.
* Implementation of certain features like Loops made the construction of the code in Hack Machine Language a little bit easier.
* We also learned the working of the software CPU Emulator in which we simulated the outputs of the above-mentioned question.
* CONTRIBUTION

The group was split into two: 3 members (Anand, Advaith and Shreya) for making the Hack Machine Language code and 2 members (Mahadev and Nandana) for making the pseudocode.