

Elements Of Computing Systems 1

END - SEMESTER 1- PROJECT



NAME: ADITHYA KRISHNA ROLL NO: AM.EN.U4AIE21005

NAME: ADITHYA S NAIR ROLL NO: AM.EN.U4AIE21006

NAME: ATHUL GIREESH ROLL NO: AM.EN.U4AIE21020

NAME: NAVNEETH KRISHNA ROLL NO: AM.EN.U4AIE21047

NAME: ANOOP BOBY MANUEL ROLL NO: AM.EN.U4AIE21015



# **PSEUDO CODE**

import java.util.\*;

class nTwoT

{

public static void main(String[] args)

{ Scanner sc=new Scanner(System.in);

int a[] = new int[8];

for(int i=0; i<8; i++)

{

a[i]= sc.nextInt();

}

int k, j, temp;

for(k = 0; k < 8; k++)

{

for(j = k+1; j < 8; j++)

{

if(a[j] < a[k])

{

temp = a[k];

a[k] = a[j];

a[j] = temp;

}

}

}

for(int t=0;t<8;t++)

{

System.out.print(a[t]+ " ");

}

}

}

The pseudo code is written in java programming language and it uses the bubble sort algorithm to sort 8 values entered by the user into an array. The algorithm uses two loops, one nested inside the other to execute the program.

# **HACK ASSEMBLY CODE**

//This asm aims at sorting numbers manually entered into the registers 0-7 of the ram and then storing the sorted numbers to the registers 30-37

//We proceed by copying the manually entered numbers in the registers 0-7 to the registers 30-37

//Then we use the ram registers 30-37 to store the sorted numbers too

//RAM 40 acts as a pointer to the initial location of the first element to be copied i.e at RAM 0

//RAM 41 acts as a pointer to the initial location of the first register where the element must be copied to

//RAM 50 stores the array size or the total no. of values that must be sorted

//RAM 45 copies the value in RAM 50 for loop iterations

@0 //initializing the value of the first register for pointer

D=A

@40

M=D

@30 //initializing the destination register as pointer

D=A

@41

M=D

@8 //Putting the array size into the register

D=A

@50

M=D

@50 //copying the array size into another register for loop iterations

D=M

@45

M=D

(loop) //loop label indicating the code start for the copy loop

//@45

//D=M

//transfering elements from 0-7 to 30-37

@40

A=M

D=M

@41

A=M

M=D

//Updating the location values

@40

M=M+1

@41

M=M+1

//condition for exiting or returning to the loop

@45

M=M-1

D=M

@loop //reruns the loop

D;JGT

@init //moves on to the next part of the code

D;JEQ

(init)

//Initialize the start value for sort(register number):A=30

@30

D=A

@20

M=D

// Load arraysize into both parent and child iteration pass

@50

D=M

@10

M=D

@12

M=D

@45

D=M

@CONDITION

D;JEQ

(CONDITION)

// Check parent iteration

@10

D=M

(END)

@END

D; JEQ

//Check child iteration within parent

@12

M=M-1 // M=5-1; M=4;

D=M // D=4 ;

@RELOAD

D; JEQ

//moves first value into register for comparison

@20

A=M //Load array[i]

D=M //D = array[i]

//brings in the next value for comparison

@20

A=M+1 //Load array[i+1]

D=D-M // D = array[i] - array[i+1]

// If array[i] > array[i+1] then SWITCH{MAIN CONDITION FOR SORT}

@SWITCH

D; JGT // Jump to SWITCH

// If array[i] <= array[i+1] then move to next iteration

@20

M=M+1 //Increment child iteration

@CONDITION

0; JMP // Jump to CONDITION again on next iteration

(SWITCH)

//Store array[i]

@20

A=M //load array[i]

D=M //D = array[i]

@11

M=D // Place array[i] for SWITCH

//Store array[i+1]

@20

A=M+1 //load array[i+1]

D=M //store array[i+1] in D register

//Move i+1 into i

@20

A=M //load array[i]

M=D //Place i+1 into i

//Move i into i+1

@11

D=M //Load i into D

@20

A=M+1 //Load array[i+1]

M=D //Place i into i+1

// Increment child iteration and jump to next comparison

@20

M=M+1 //Increment child pass iteration

@CONDITION

0; JMP // Jump to CONDITION again on next iteration

(RELOAD)

// Reset child iteration within pass

@50

D=M

@12

M=D

// Decrement child iteration

@10

M=M-1

// Reset A-register place

@30

D=A

@20

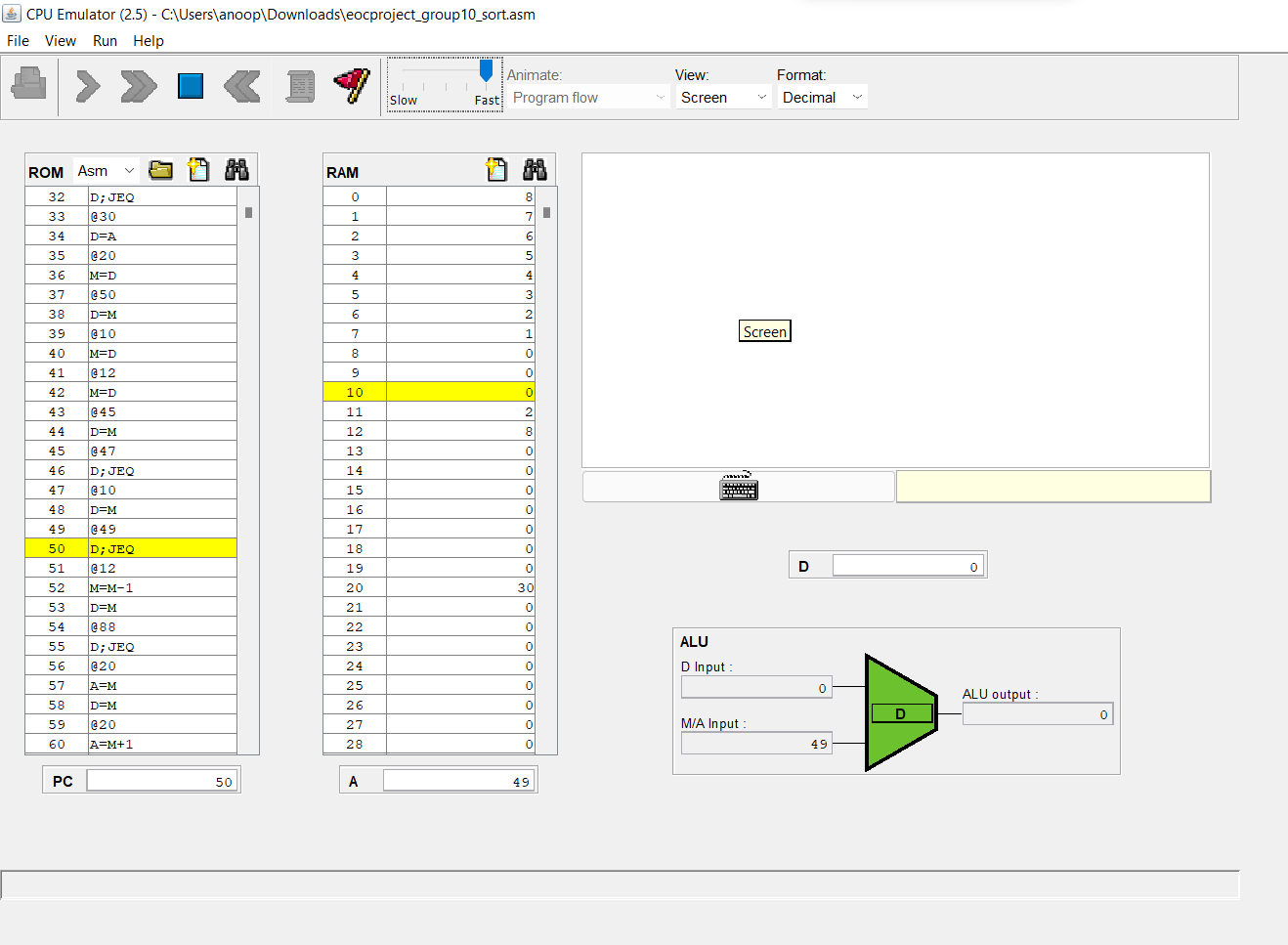
M=D

@CONDITION

0; JMP

# **INPUT**

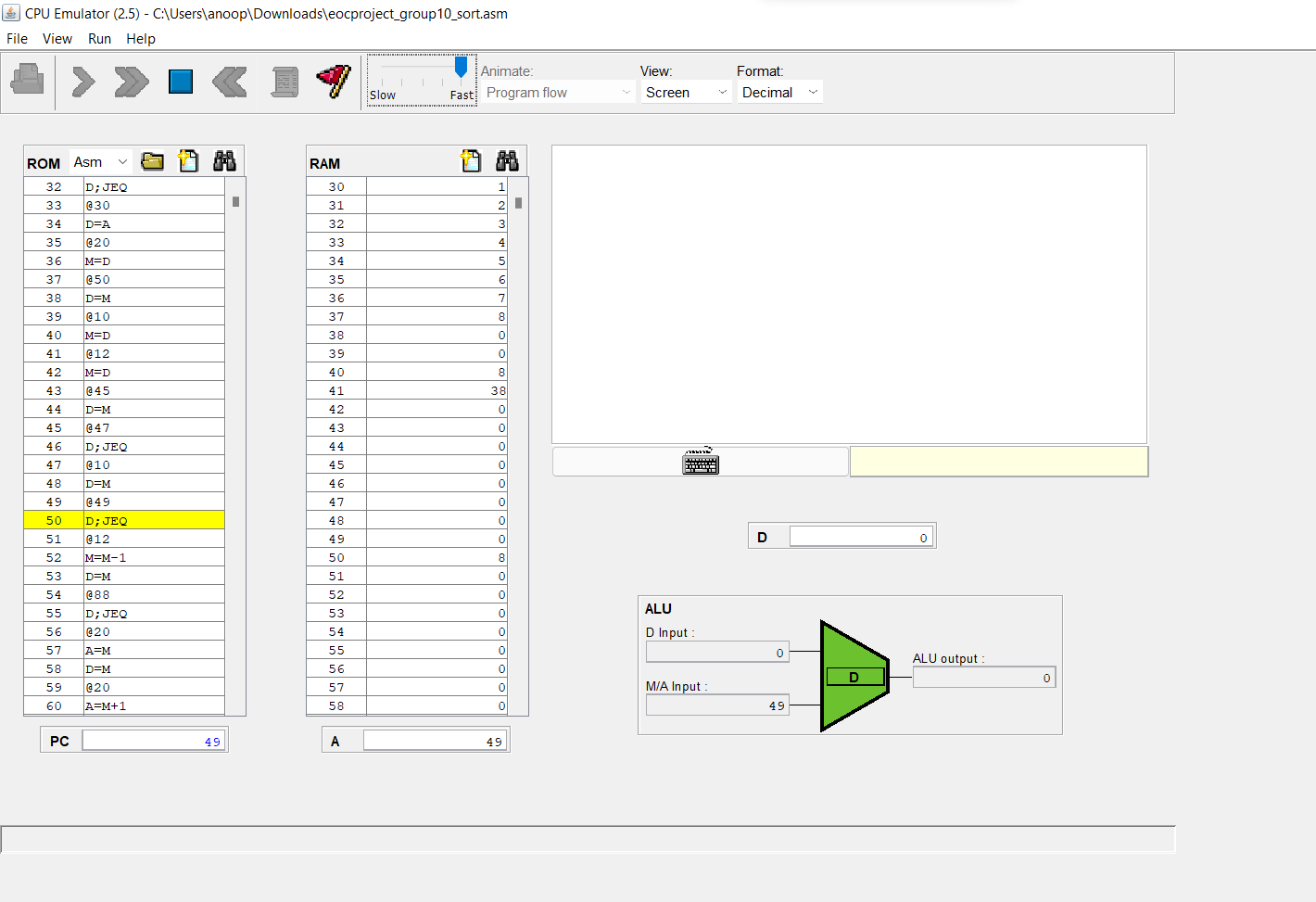
The values to be sorted are placed in registers 0-7 manually.

****

# **OUTPUT**

The sorted values are displayed in the registers 30-37 automatically.

The values are sorted in ascending order i.e lowest to highest.

****

# 

# **INSIGHTS**

By doing this project we have learned many things about the hack assembly code some of which include:-

* Learning how the basic A,D,M registers work.
* How to copy information from registers and transfer them to other registers.
* How to implement and manipulate loops to our favour.
* How to exit a loop to continue the program.
* How to use pointers to point to registers where we extract values.
* How to work on extracted values in a temporary register.
* How to implement nested loops and how to check the child and parent loop conditions.
* Implementing labels to access specific parts of code whenever we want.
* Updating existing values inside a register for it to act as a counter of sorts.
* Using an infinite loop to terminate the program.

**=====================================================**