ELEMENTS OF COMPUTING SYSTEMS

PROJECT REPORT

SUBMITTED BY-

S HARI SANKAR (AM.EN.U4AIE21056)

R RAKESH (AM.EN.U4AIE21052)

K SUBHASH (AM.EN.U4AIE21036)

SAI MANASA SOUMYA (AM.EN.U4AIE21057)

PULLAKHANDAM CHARISHMA AKSHAYA (AM.EN.U4AIE21051)

WORKING OF THE PROJECT

❖ First the user inputs 6 random numbers from RAM[0] to RAM[5].

0	7
1	3
2	54
3	6
4	1
5	58

❖ Two variables l and i are created at the starting of the program and the values are initialized to 0 and 30 respectively. l is used for iterating from 0 to 5 whereas i is used for iterating from 30

16	0
17	30

❖ Any odd number when performed logical AND with 1 will produce an output of 1 and any even number when performed logical AND with 1 will produce an output of 0.

❖ Now the emulator takes up values from RAM[0] to RAM[5] and if a value is found out to be odd, it is stored from RAM[30] onwards.

30	7
31	3
32	1

❖ Now when the emulator reaches the 6th register it will jump to the sorting instructions. i value is changed to 30

16	6
17	30

❖ The emulator considers values inside RAM[i] and RAM[i+1]. If RAM[i] is greater then RAM[i+1] the value gets swapped. Upon reaching 0, the emulator checks if the values are sorted, if not it will repeat the sort cycle.

Output after 1st cycle

30	3
31	1
32	7

Output after completion

30	1
31	3
32	7

❖ If all the values are sorted it will jump to ending infinite loop where the variables l and i are set to 0

16	0
17	0

<u>PSEUDO CODE</u>

//TWO VARIABLES i AND l. i IS USED FOR ITERATION FROM 30 . l IS USED FOR ITERATION FROM 0 TO 5 AND FOR STORING VALUES.

i=30

l=0

go to SAS

//CHECKING IF THE NUMBER IS ODD OR EVEN

(SAS)

D=ram(l)

D=D AND 1

```
 IF D==0
  JUMP TO LOOP1
 IF D==1
  JUMP TO LOOP
//IF EVEN THEN GO TO NEXT REGISTER
(LOOP1)
 D=l
 IF D-6 == 0:
   GOTO SORT
 l=l+1
JUMP TO SAS
//IF ODD STORE VALUE IN lTH REGISTER TO THE iTH REGISTER
(LOOP)
 A=l
 D=RAM[l]
 A=i
 RAM[i]=D
 i=i+1
 l=l+1
 JUMP TO SAS
 //SORTING BEGINS
(SORT)
  i=30
 JUMP TO SORT1
//SORTING IF RAM[i]>RAM[i+1]
(SORT1)
```

```
  D=RAM[i]
  if D==0
     jump to check
  A=i+1
  if M==0
      jump to check
  if D-M>0
    JUMP TO SWAP
  if D-M<0
    jump to cont
//SWAPPING VALUES BETWEEN REGISTERS
(SWAP)
  D=RAM[i]
  l=RAM[i+1]
  A=i+1
  M=D
  A=i
  M=l
  i=i+1
  jump to SORT1
 //CONTINUING TO THE NEXT REGISTER IF RAM[i]<RAM[i+1]
(CONT)
  i=i+1
  jump to SORT1
//CHECKING IF THE ARRAY IS SORTED
(CHECK)
```

```
  i=30
  GO TO CHECK2
(CHECK2)
  D=RAM[i]
  A=i+1
  if D==0:
    GO TO ENDING
  if M==0:
    GO TO ENDING
  if D>M
    GO TO SORT
  i=i+1
  GO TO CHECK2
//ENDING THE PROGRAM WITH INFINITE LOOP
(ENDING)
 i=0
 l=0
 GO TO ENDING
```

ASM CODE

```
@0
D=A
@l
M=D
```

```
@30
D=A
@i
M=D
@SAS
(SAS)
   @l
   A=M
   D=M
   @1
   D=D&A
   @LOOP1
   D;JEQ
   @LOOP
   D-1;JEQ
(LOOP)
  @l
  A=M
  D=M
  @i
  A=M
  M=D
  @i
  M=M+1
  @l
  M=M+1
```

```
  @SAS
  0;JEQ
(LOOP1)
  @l
  D=M
  @6
  D=D-A
  @SORT
  D;JEQ
  @l
  M=M+1
  @SAS
  0;JEQ
(SORT)
 @30
 D=A
 @i
 M=D
 @SORT1
 0;JMP
(SORT1)
 @i
 A=M
 D=M
 @CHECK
 D;JEQ
```

```
 @i
 A=M+1
 D=M
 @CHECK
 D;JEQ
 @i
 A=M
 D=M
 A=A+1
 D=D-M
 @SWAP
 D;JGT
 @i
 M=M+1
 @SORT1
 0;JMP
(SWAP)
  @i
  D=M+1
  @l
  A=D
  D=M
  @l
  M=D
  @l
  M=D
```

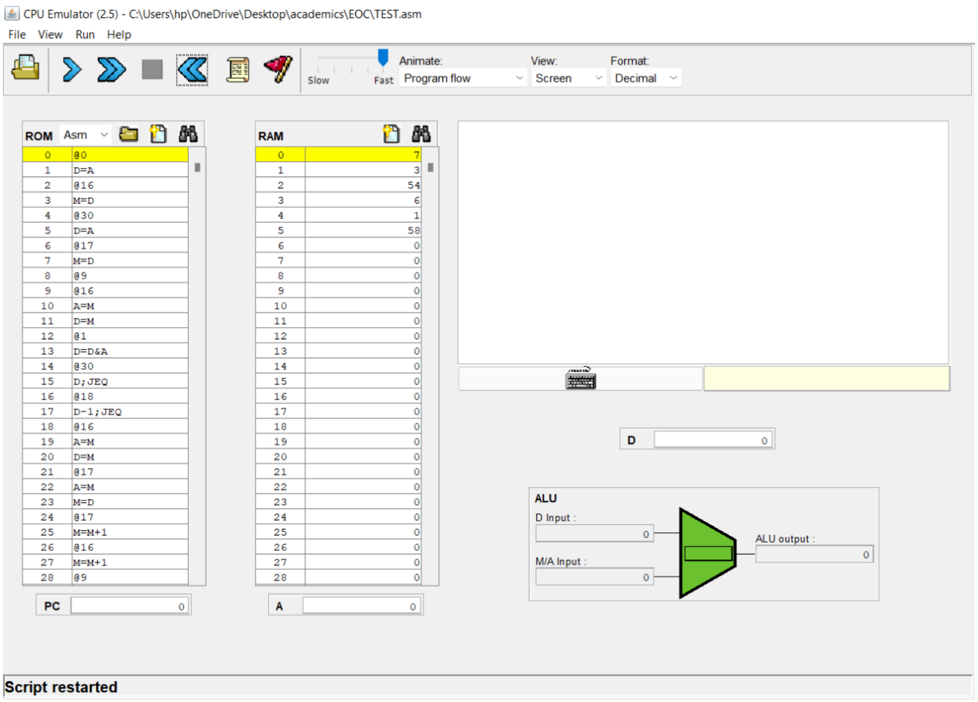
```
 @i
 A=M
 D=M
 @i
 A=M+1
 M=D
 @l
 D=M
 @i
 A=M
 M=D
 @i
 M=M+1
 @SORT1
 0;JMP
(CHECK)
 @30
 D=A
 @i
 M=D
 @CHECK2
 0;JMP
(CHECK2)
 @i
 A=M
 D=M
```

```
 @ENDING
 D;JEQ
 @i
 A=M+1
 D=M
 @ENDING
 D;JEQ
 @i
 A=M
 D=M
 A=A+1
 D=D-M
 @SORT
 D;JGT
 @i
 M=M+1
 @CHECK2
 0;JMP
(ENDING)
@i
M=0
@l
M=0
@ENDING
0;JMP
```

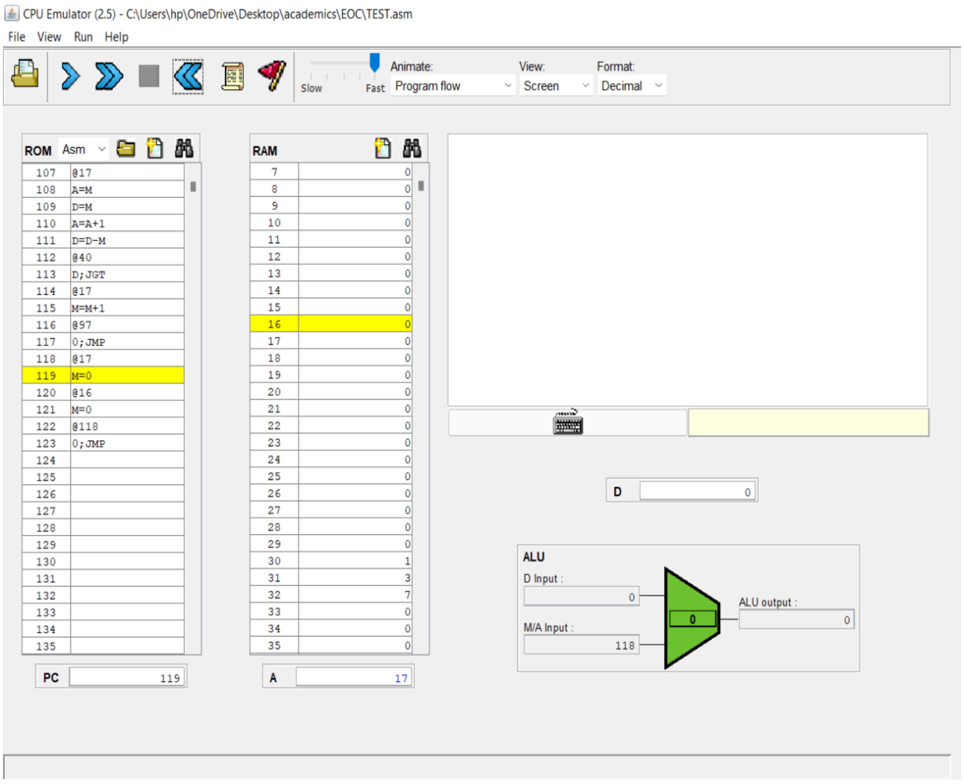
OUTPUT

RAM[0] = 7, RAM[1] = 3, RAM[2] = 54, RAM[3] = 6, RAM[4] = 1, RAM[5] = 58

INPUT SNAPSHOT



OUTPUT SNAPSHOT



INSIGHTS LEARNED DURING IMPLENTATION

- The assembly language does not have operators like division, modulo etc , but it is achievable through logical operations like AND NOT etc.
- Usage of labels while coding in assembly language is a good practise as it helps in the flexibility of the code as well as helps to understand the code.
- It is a very good practise to write pseudo code before writing the asm file as it helps to reduce the chance of getting errors by a large margin.
- Identifying logical errors in machine language is a hard and is an almost impossible task. So it is always a good practise to test the code manually by hand before running the code on the emulator.
- Another advantage that we learned from assembly language is that since it directly works with the hardware we can store specific inputs in the registers according to our choice
- It also gives us a good picture of how the computer performs the complex tasks such as sorting etc using the ALU

****