VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DIGITAL ASSIGNMENT-1 REPORT

REGISTER NO.	21BEC1851
NAME	Rahul Karthik S
SCHOOL/PROGRAM	SENSE/B. Tech (ECE)
SEMESTER/SLOT	Winter 2022-23 / A1+TA1
COURSE CODE / NAME	BECE204L – Microprocessors and Microcontrollers
DATE OF ANNOUNCEMENT	15/01/2023
LAST DATE FOR SUBMISSION	05/02/2023 (Sunday 11:59 PM)

Q. No.	TITLE	MARKS
1	Arithmetic Expression	
2	Odd/Even, Code Conversion	
3	Average, Comparison	
4	Positive/Negative Number Separation	
5	Search and Replace a Byte	
	Total Marks	

COURSE HANDLER'S NAME	Dr. V. PRAKASH
COURSE HANDLER'S SIGNATURE	

1. SOLVE THE GIVEN EXPRESSION

Question:

Write an 8086 assembly language program to implement following expression and store the final result at memory location starting from 2000H.

$$Y = (a + b)\ (a2 - ab + b2)$$

Consider last 4 digits your register number and assume "a" as first two digits, "b" as last two digits. (For example if Reg. no is 21BEC1073 assume "a" as 10H and "b" as 73H).

Algorithm:

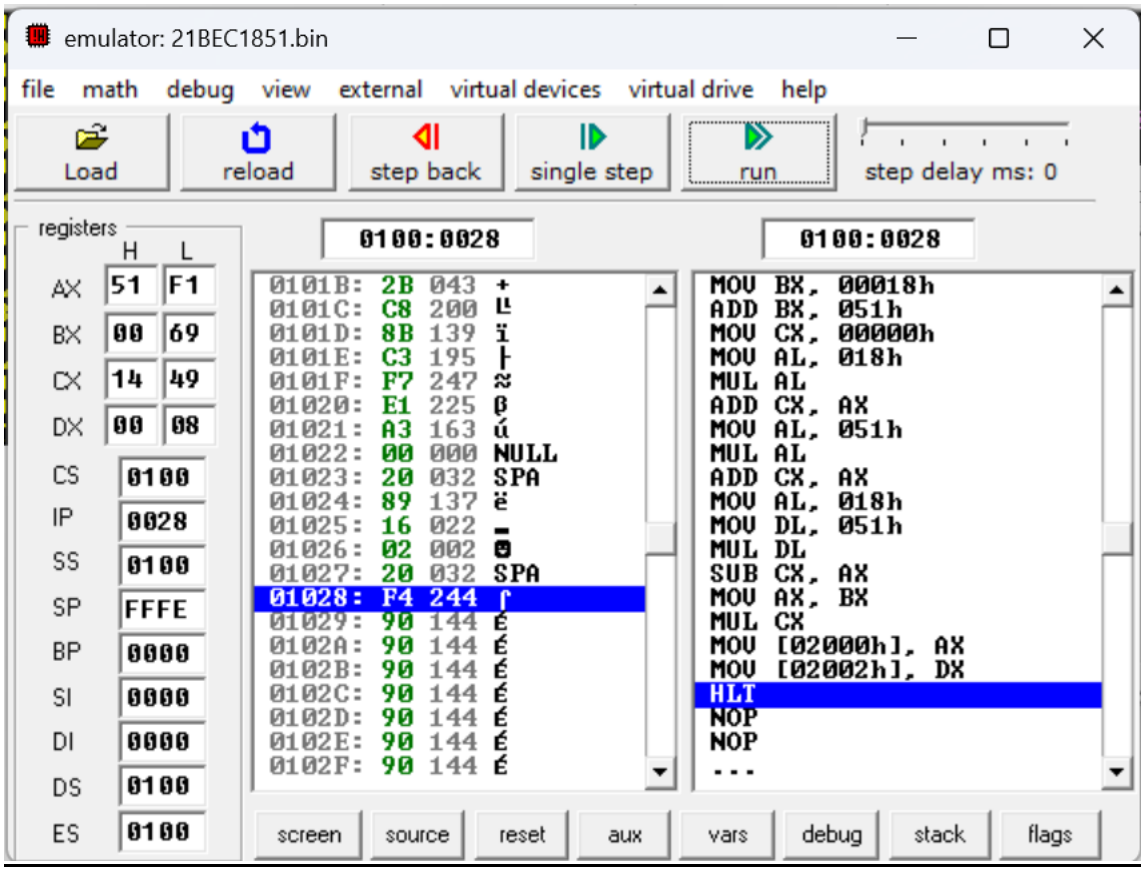
- First the 'a' and 'b' values are being added and stored in BX Register.
- Then, CX register is being initialized with zero.
- Later, 'a' is squared and then added with the CX register, 'a' and 'b' are multiplied and subtracted from CX register and 'b' is squared and added with the CX register.
- Then finally the two values are being multiplied and stored in the 2000H memory location.

Program:

```
MOV BX, 18H         ; STORING 18H VALUE IN BX REGISTER
ADD BX, 51H         ; ADDING 51H VALUE WITH VALUE STORED IN BX REGISTER
MOV CX, 0000H       ; INITIALIZING THE VALUE OF CX REGISTER
MOV AL, 18H         ; STORING 18H VALUE IN AL REGISTER
MUL AL              ; SQUARING THE VALUE IN AL REGISTER AND STORING IT IN AL REGISTER
ADD CX, AX          ; STORING THE SQUARED VALUE IN AX REGISTER IN CX
MOV AL, 51H         ; STORING 51H VALUE IN AL REGISTER
MUL AL              ; SQUARING THE VALUE IN AL REGISTER AND STORING IT IN AL REGISTER
ADD CX, AX          ; ADDING THE SQUARED VALUE OF 51H AND STORING IN CX
MOV AL, 18H         ; STORING 18H VALUE IN AL REGISTER
MOV DL, 51H         ; STORING 51H VALUE IN DL REGISTER
MUL DL              ; MULTIPLYING VALUES IN AL AND DL AND STORING IT IN AL REGISTER
SUB CX, AX          ; SUBTRACTING THE MULTIPLIED VALUE OF 18H AND 51H AND STORING IN CX
MOV AX, BX          ; MOVING VALUE OF BL INTO AL FOR MULTIPLICATION
MUL CX              ; MULTIPLYING TO GET THE FINAL ANSWER
MOV [2000H], AX     ; MOVING AX VALUE INTO 2000H MEMORY LOCATION
MOV [2002H], DX     ; MOVING DX VALUE INTO 2002H MEMORY LOCATION
HLT
```

Screenshots:

(a) Program



(b)Output

Inference:

- In this program, 18H and 51H are the inputs and the outputs are stored in 2000H, 2001H and 2002H memory locations.

2. ODD OR EVEN

Question:

Write an 8086 assembly language program to check the given number "WXYZ" is ODD or EVEN and perform for the following,

- If it is ODD, assume the given number is a Hexadecimal and convert into BCD
- If it is EVEN, assume the given number is Hexadecimal and convert it into ASCII

The converted BCD output must be stored in memory location from 2020H onwards and ASCII output must be stored in memory location from 2040h onwards.

Algorithm:

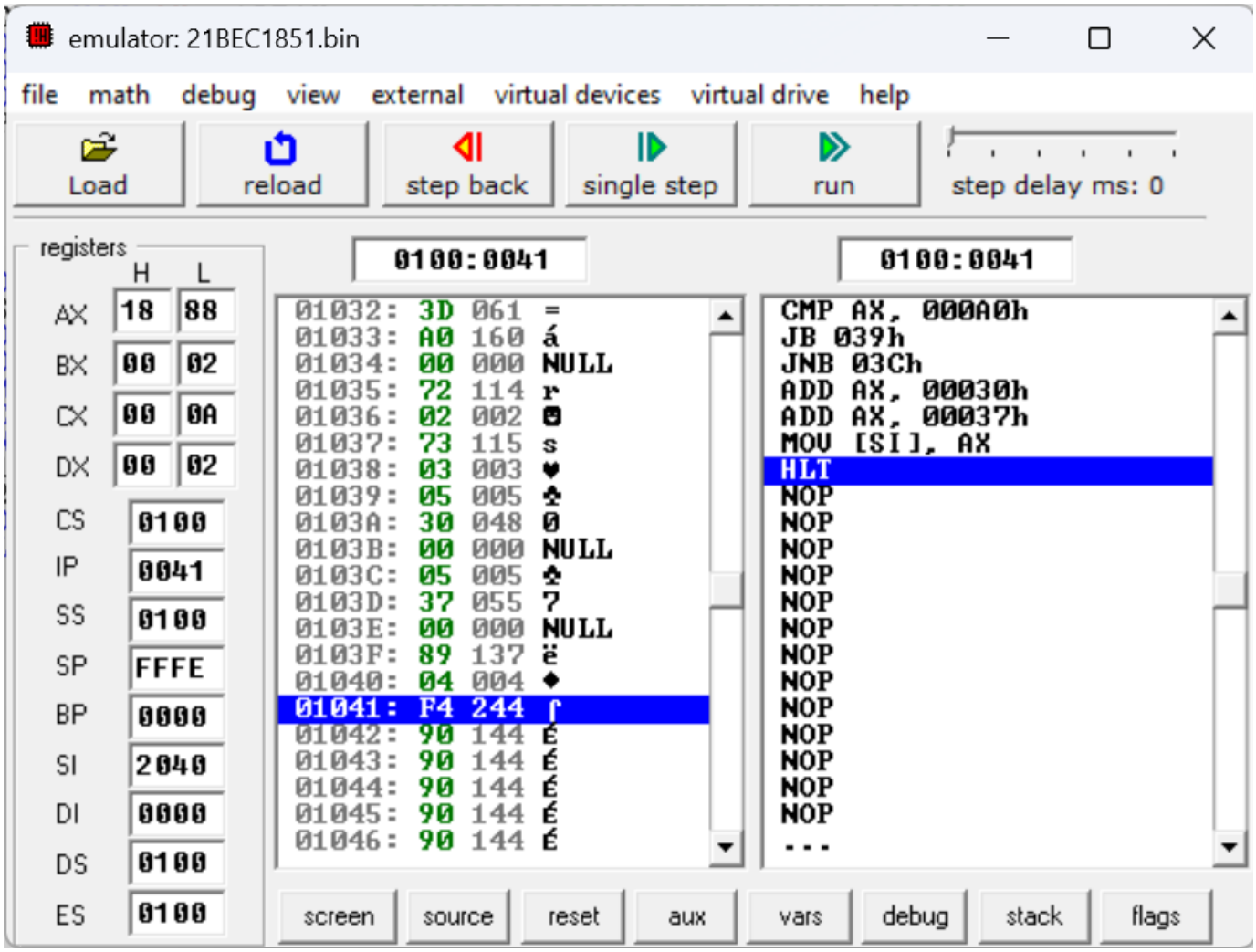
- Move the value to AX register, divide it by 2 and if the CMP command triggers 0 flag it is even otherwise odd.
- To convert to BCD, initialize value again, initialize array pointer, divide the CX and compare it with AX and jump if it is greater.

Program:

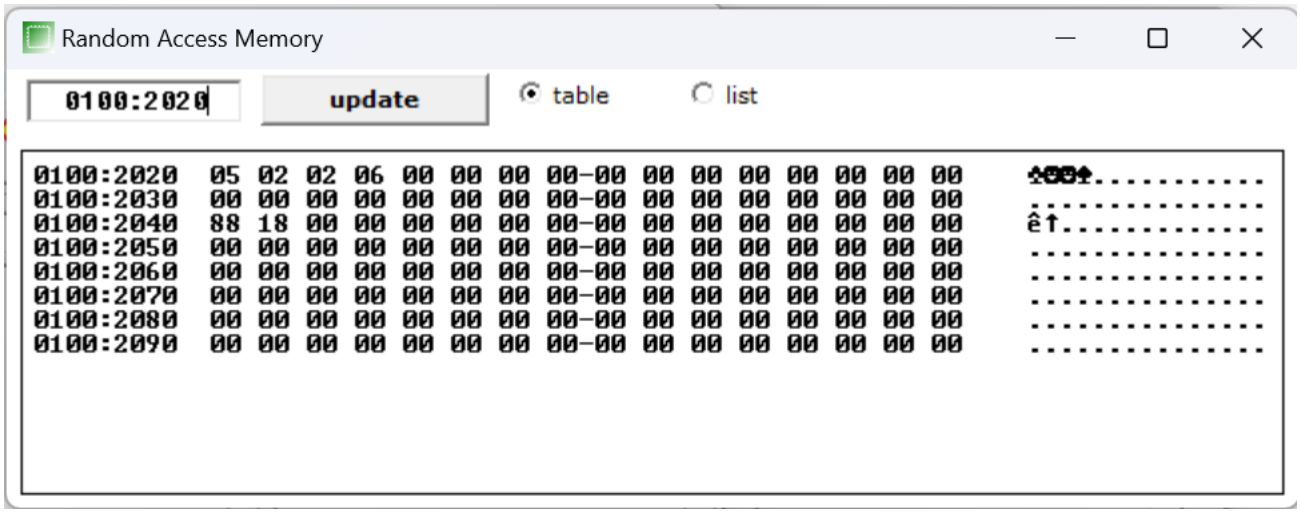
```
MOV AX, 1851H      ; MOVING 1851H VALUE TO AX REGISTER
MOV BX, 0002H      ; MOVING 0002H VALUE TO BX REGISTER
DIV BX             ; DIVIDING 1851H BY 0002H
CMP DX, 0          ; FINDING WHETHER THE NUMBER IS ODD OR EVEN
JNZ BCD            ; GO TO BCD LABEL IF NUMBER IS ODD
JZ ASCII           ; GO TO ASCII LABEL IF NUMBER IS EVEN
BCD: MOV AX, 1851H  ; INITIALIZING THE VALUE AGAIN
MOV SI, 2020H      ; INITIALIZING SI VALUE
MOV [SI],0000H     ; INITIALIZING THE 2020H MEMORY LOCATION AS 0000H
MOV CX, 0AH        ; INITIALIZE THE COUNT VALUE AS 0AH
UP: MOV DX, 00H    ; INITIALIZE DX WITH 00H
   DIV CX          ; DIVIDE AX BY CX
   MOV [SI],DL     ; MOVE DL VALUE TO THE MEMORY LOCATION IN SI REGISTER
   INC SI          ; INCREMENT THE ARRAY POINTER
   CMP AX, CX      ; COMPARE AX WITH CX TO FIND WHICH GREATER VALUE IS
   JAE UP          ; JUMP IF VALUE IS GREATER THAN OR
MOV [SI], AX       ; MOVE AX VALUE TO MEMORY LOCATION VALUE IN SI REGISTER
ASCII: MOV AX, 1851H; RE-INITIALIZE THE VALUE
MOV SI, 2040H      ; INITIALIZE THE SI VALUE
MOV [SI], 0000H    ; INITIALIZE MEMORY VALUE WITH 0H
CMP AX, 0A0H       ; COMPARING AX VALUE WITH 0AH
JC L1              ; IF ITS LESSER THEN GO TO L1 LABEL
JNC L2             ; IF ITS GREATER THAN GO TO L2 LABEL
L1: ADD AX, 30H    ; ADD AX WITH 30H VALUE
L2: ADD AX, 37H    ; ADD AX WITH 37H VALUE
MOV [SI], AX       ; MOV AX VALUE TO MEMORY LOCATION
HLT
```

Screenshots:

(a) Program



(b) Output



Inference:

- The input is in AX register and the output for BCD is stored in 2020H and the output for ASCII is stored in 2040H.

3. AVERAGE OF NUMBERS IN AN ARRAY

Question:

During monsoon months, the weather monitoring station recorded the data (8-Bit) of daily Rainfall (R) for past 10 days' memory location from 0100: 2000H onwards as given in the table-1 below. Write an 8086 assembly language program to compute the Average rainfall (A) of the 10 days' data and perform the following:

- If R = A, then increment BL to indicate number of days with average rainfall
- If R > A, then increment CL to indicate number of days with above average rainfall
- If R< A, then increment DL to indicate number of days with below average rainfall

Algorithm:

- We can find the average by storing the N value, finding the sum value and divide it by N.
- Compare array value and average using CMP and if the array value is greater than average, then use JG, if array value less than average means JL and if both are equal means use JE

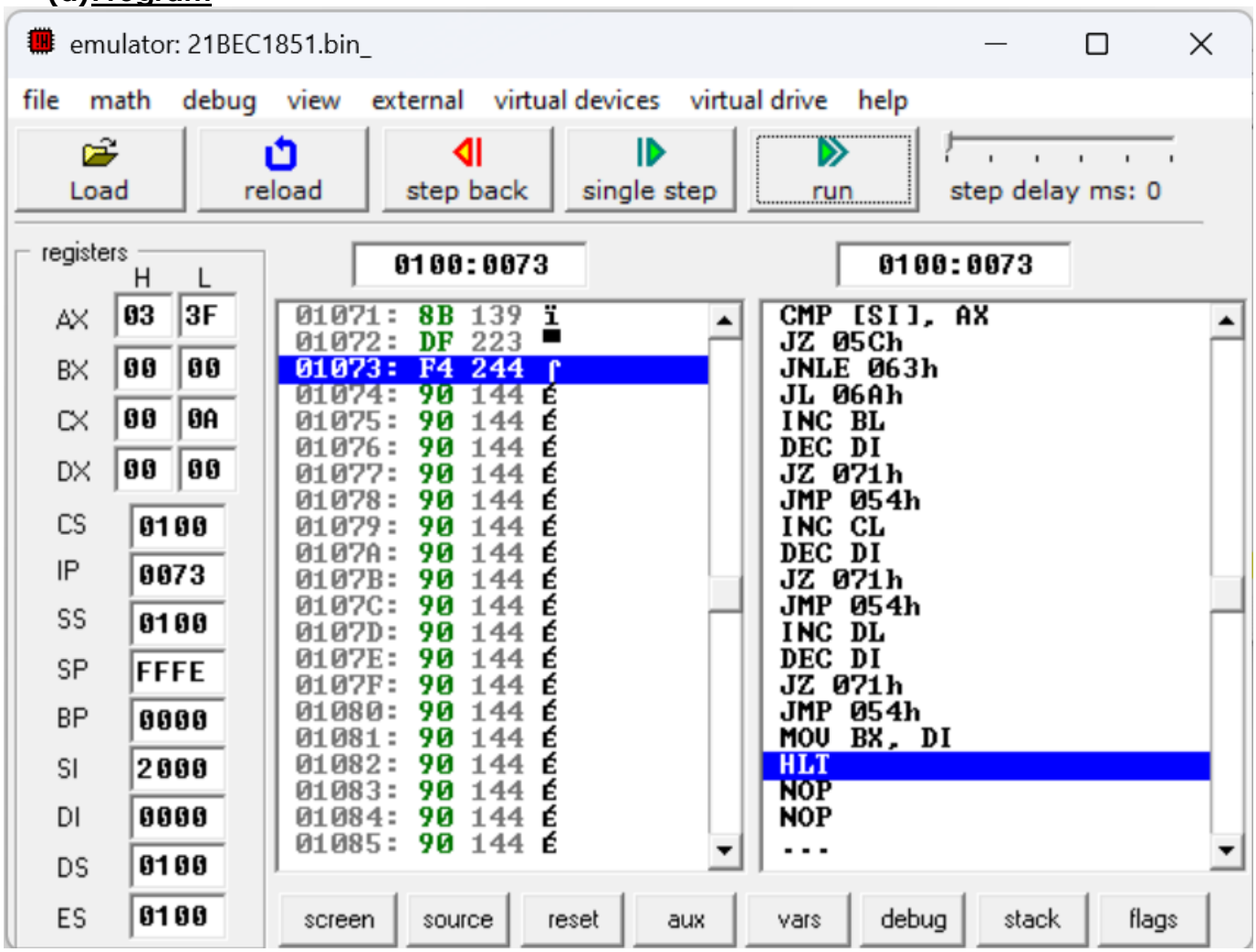
Program:

```
MOV SI, 2000H
MOV AX, 0000H
MOV [2000H], 18H
MOV [2001H], 51H
MOV [2002H], 45H
MOV [2003H], 1AH
MOV [2004H], 78H
MOV [2005H], 22H
MOV [2006H], 54H
MOV [2007H], 2CH
MOV [2008H], 36H
MOV [2009H], 61H
MOV CL, 0AH
MOV BL, 0AH
BACK: ADD AL,[SI]
ADC AH,00H
INC SI
DEC CL
JNZ BACK
DIV BL
MOV DI, 0AH
MOV SI, 2000H
MOV BL, 00H
MOV CL, 00H
MOV DL, 00H
LP1: CMP [SI],AX
JE L11
JG L12
JL L13
L11: INC BL
    DEC DI
    JZ OVER
    JMP LP1
L12: INC CL
```

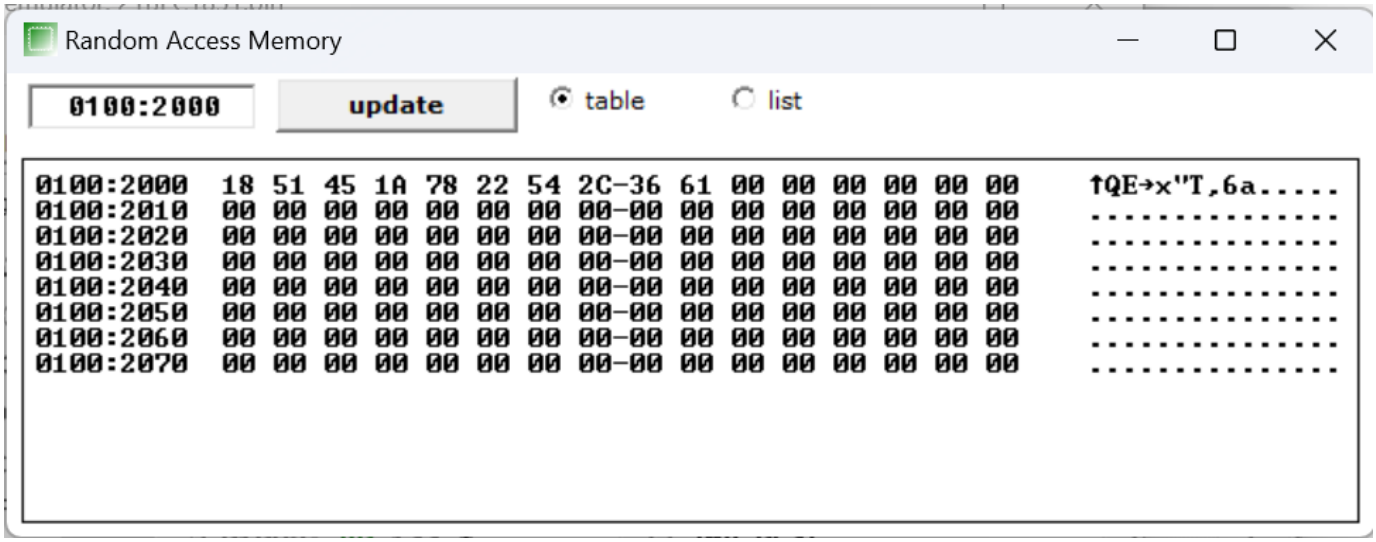
```
    DEC DI
    JZ OVER
    JMP LP1
L13: INC DL
    DEC DI
    JZ OVER
    JMP LP1

OVER: MOV BX, DI
HLT
```

Screenshots:

(a) Program



(b) Output



Inference:

- The inputs are given in the code and the output is stored in BL, CL, DL registers respectively.

4. POSITIVE AND NEGATIVE NUMBERS IN AN ARRAY

Question:

Write an 8086 assembly language program to separate the positive and negative numbers from array of 10 numbers present in the memory location from 2000:1000H as given in the Table-2 below.

- Separate all positive numbers from array and store it in the memory location starting from 2000:1020H and all negative numbers at memory location starting from 2000:1040H.
- Also, store the positive count in register BX and negative count in register DX.

Algorithm:

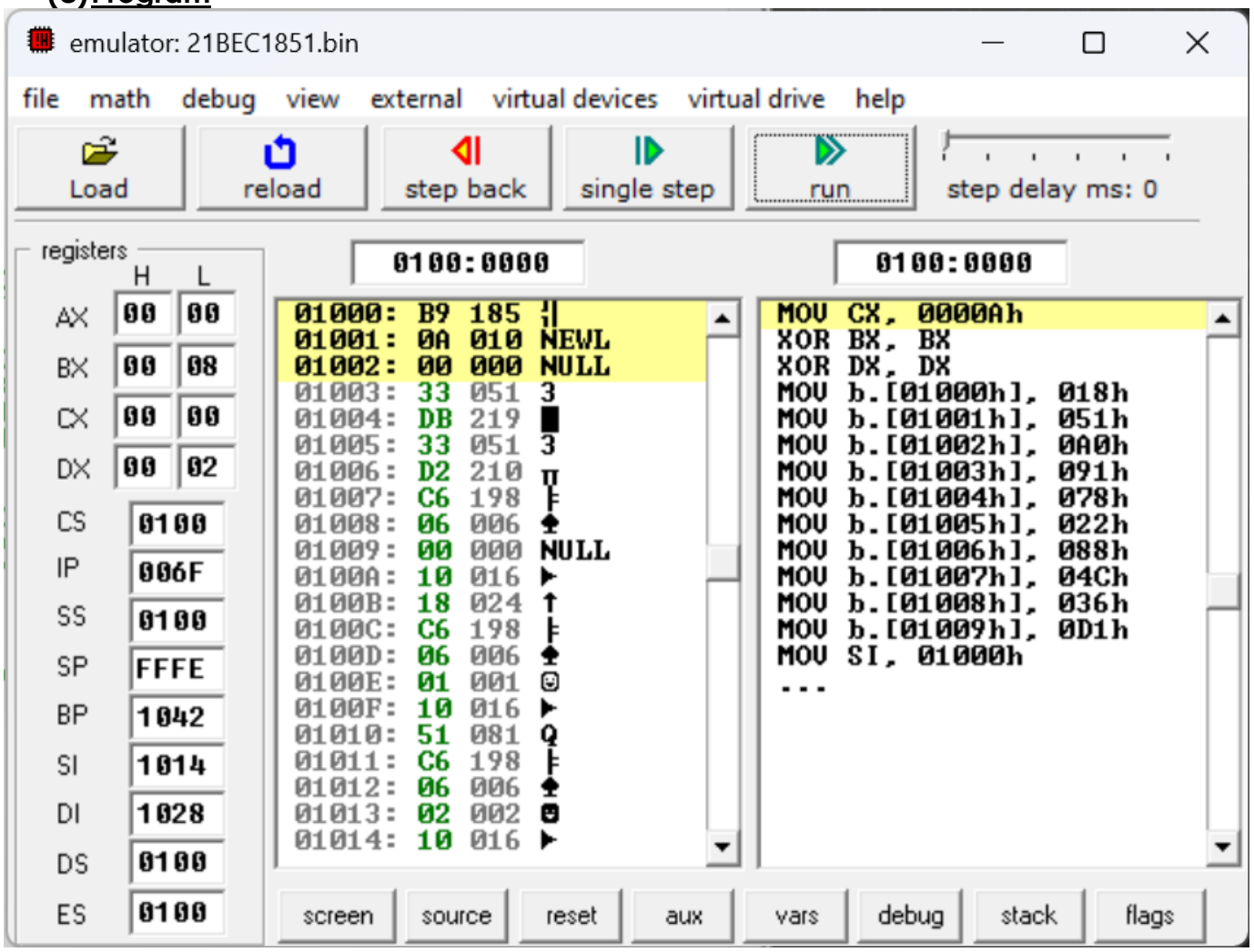
- The register values are cleared and the values are stored
- Jump if the value in array is positive, and storing it in 1020H memory location
- Jump if the value in array is negative, and storing it in 1040H memory location

Program:

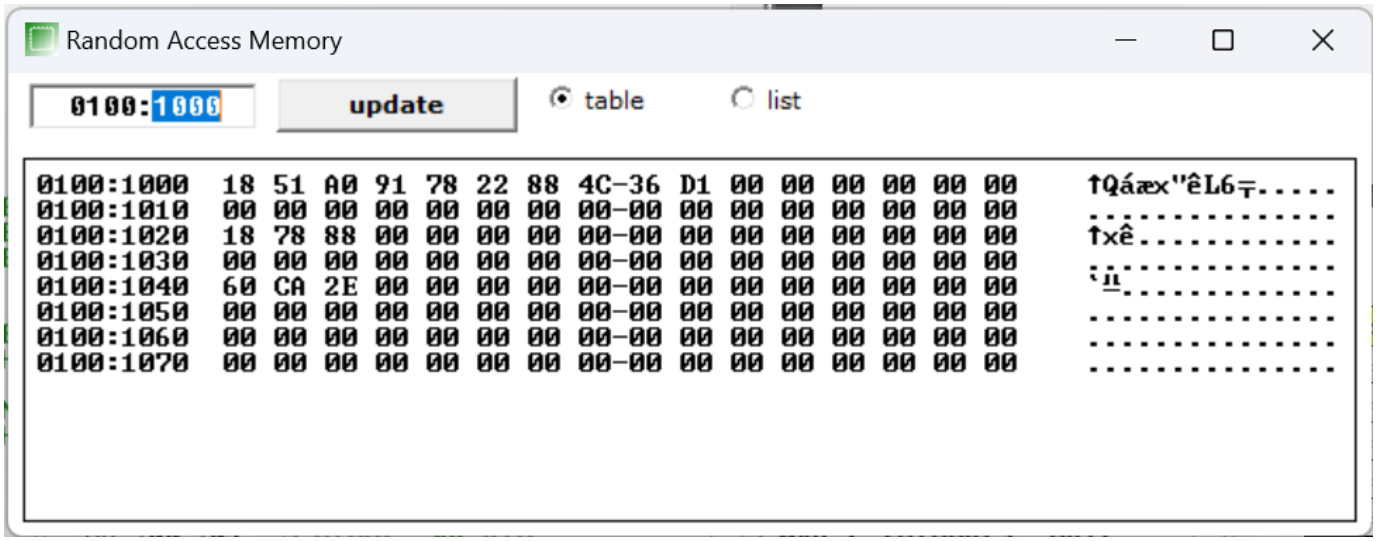
```
MOV CX, 10 ; CX IS THE LOOP COUNTER, 10 NUMBERS IN THE ARRAY
XOR BX, BX ; CLEAR THE BX REGISTER
XOR DX, DX ; CLEAR THE DX REGISTER
MOV [1000H], 18H
MOV [1001H], 51H
MOV [1002H], 0A0H
MOV [1003H], 91H
MOV [1004H], 78H
MOV [1005H], 22H
MOV [1006H], 88H
MOV [1007H], 4CH
MOV [1008H], 36H
MOV [1009H], 0D1H
MOV SI, 1000H ; SI IS THE SOURCE POINTER, POINTING TO THE FIRST ELEMENT IN THE ARRAY
MOV DI, 1020H ; DI IS THE DESTINATION POINTER FOR POSITIVE NUMBERS
MOV BP, 1040H ; BP IS THE DESTINATION POINTER FOR NEGATIVE NUMBERS

L1: ; START OF THE LOOP
  MOV AX, [SI] ; LOAD THE CURRENT ELEMENT FROM THE ARRAY INTO AX
  CMP AX, 0 ; CHECK IF THE ELEMENT IS POSITIVE OR NEGATIVE
  JGE POS ; IF POSITIVE, JUMP TO POS
  NEG AX ; IF NEGATIVE, MAKE IT POSITIVE
  MOV [BP], AX ; STORE THE POSITIVE VALUE IN THE NEGATIVE NUMBERS ARRAY
  INC BP ; INCREMENT THE BP POINTER
  INC DX ; INCREMENT THE NEGATIVE COUNT
  JMP NEXT ; JUMP TO THE NEXT ELEMENT

POS: ; START OF THE POSITIVE CASE
  MOV [DI], AX ; STORE THE POSITIVE VALUE IN THE POSITIVE NUMBERS ARRAY
  INC DI ; INCREMENT THE DI POINTER
  INC BX ; INCREMENT THE POSITIVE COUNT

NEXT: ; START OF THE NEXT ELEMENT
  ADD SI, 2 ; INCREMENT THE SOURCE POINTER BY 2
  LOOP L1 ; REPEAT THE LOOP FOR THE NEXT ELEMENT
```

Screenshots:

(c) Program



(d) Output



Inference:

- The input values are stored in 1000H memory location, the positive values are stored in 1020H memory location and negative numbers are stored in 1040H memory location.

5. FIND AND REPLACE NUMBERS IN AN ARRAY

Question:

Write an 8086 assembly language program to search a given byte "PQ" in an array of 10 numbers and, if this value is found replace it with the value "RS" and exit. Assume the array is present in the current Data Segment memory location from 2000:1000H as given in the Table-3 below. Implement the given logic using following methods separately and verify its output.

(a) String Instructions – SCASB, REPNE

(b) Without string instruction – CMP, LOOP

Algorithm:

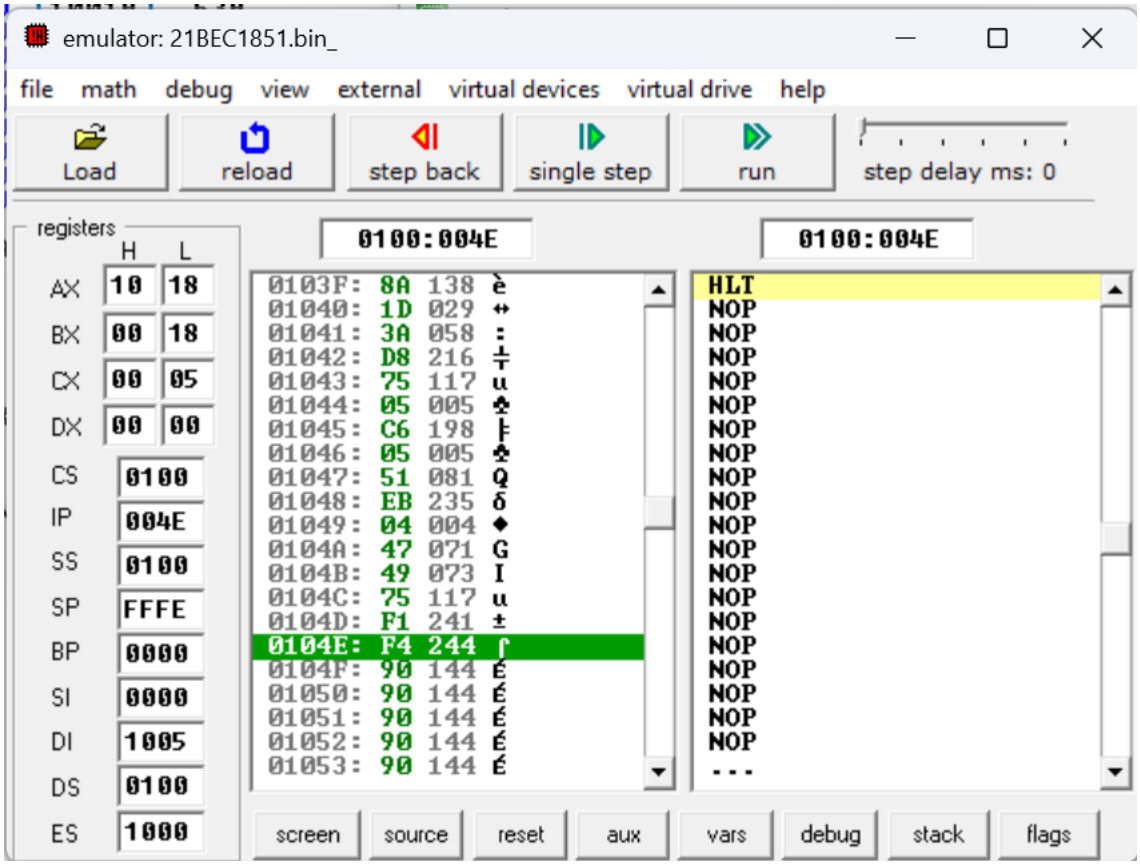
- The values are given and the array is compared with 18H. If it found 18H in the array then it replaces using 51H.

Program:

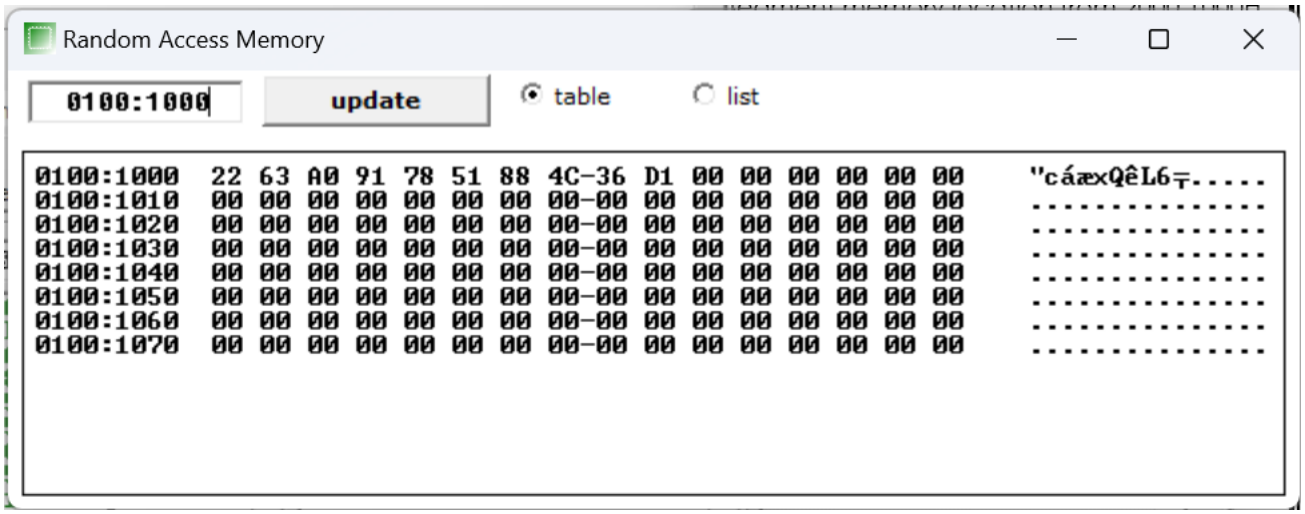
```
MOV AX, 1000H  ; Move memory location of array to AX
MOV ES, AX     ; Move the address of array to ES (Data Segment)
MOV DI, 1000H  ; Move the starting address of the array to DI
MOV CX, 10     ; Load CX with the count of numbers in the array
MOV AL, 18H    ; Move the value to search for 18H to AL
MOV [1000H], 22H
MOV [1001H], 63H
MOV [1002H], 0A0H
MOV [1003H], 91H
MOV [1004H], 78H
MOV [1005H], 18H
MOV [1006H], 88H
MOV [1007H], 4CH
MOV [1008H], 36H
MOV [1009H], 0D1H

SEARCH:
   MOV BL, [DI] ; Move the current value at DI to BL
   CMP BL, AL   ; Compare BL with AL
   JNE NEXT     ; If not equal, jump to NEXT
   MOV [DI],51H ; If equal, replace the value with RS
   JMP DONE     ; Exit the program

NEXT:
   INC DI       ; Increment DI to point to the next value in the array
   DEC CX       ; Decrement CX to keep track of the remaining values to search
   JNZ SEARCH   ; If CX is not zero, jump back to SEARCH
DONE: HLT
```

Screenshots:

(a) Program



(b) Output



Inference:

- The value 18H is being found and replaced with 51H.