AIDS MICROPROCESSOR LAB S21 BATCH (2023-24)

Experiment 3(a) Title: Assembly language programming to find minimum number from 10 8-bit hexadecimal numbers (GIVEN ARRAY) using software tool TASM 1.4

Name of student: Meet Raut ***Class Roll Number: 2201084***

Date of Performance: 26/02/2024

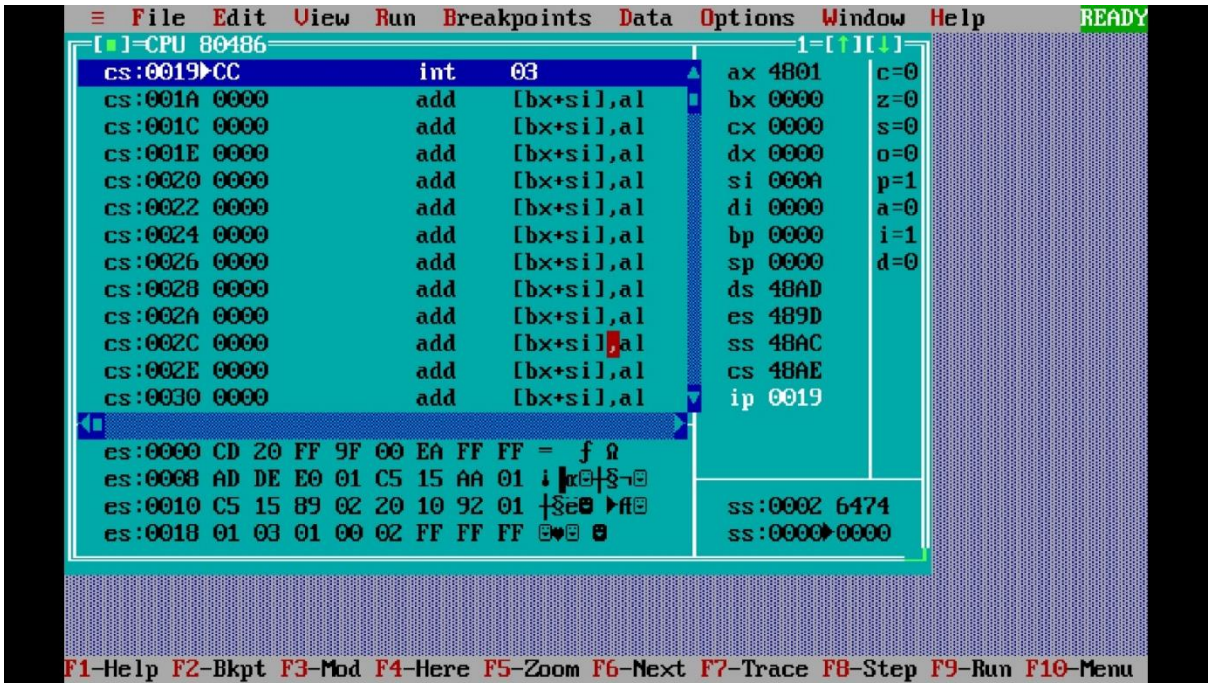
Batch: S2-1 ***Timing: 3:00-5:00*** ***Date of Submission: 26/02/2024***

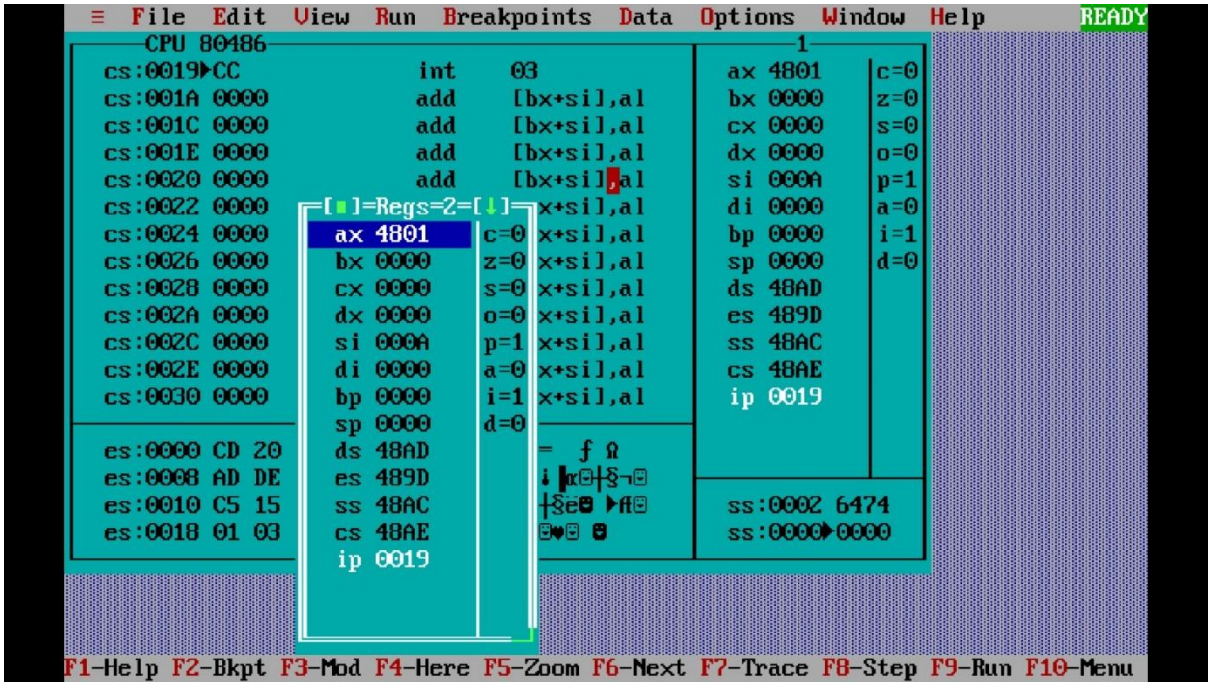
Assembly language code

```
data segment                    # Data segment

array db 04h,02h,03h,0Ch,05h,   # Declares an array named array with ten
07h,08h,0Ah,0Bh,01h               8-bit hexadecimal numbers.

smaller db 00h          # Declares a variable smaller with an initial value of 00h.

data ends               # Assemble directives


code segment

assume cs:code, ds:data     # Assemble directives defining cs and ds
start:                      # start of code segment

mov ax,data                 # moving data to ax register- initialization process

mov ds,ax                   # moving ax to dx register -- initialization process

mov si,offset array         # Uses si register to point to beginning of the array.

mov cl,0Ah                  #Initializes a counter (cx) with the value 0Ah

mov al,[si]                 # moves the value at the array location to al.

inc si                      # Increments the si register
```

```
back: cmp al,[si]       # Enters a loop (back) that compares the value in al with
.                         the value at the memory location pointed to by si
      jnc skip          # If the value in al is not less than the value in the array .
.                         (jnc skip), it skips to the label skip

skip: mov al,[si]       # moves the value at the array location to al.
      inc si            # Increments the si register
      loop back         # repeats the loop until cx becomes zero.

mov smaller,al          # Stores the final value in al into the smaller variable.

int 03h                     # breakpoint interrupt

code ends               # Assembler directives to end code

end start               # Assembler directives to end start
```

Result:





Experiment 3(b) Title: Assembly language programming to find maximum number from 10 8-bit hexadecimal numbers (GIVEN ARRAY) using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 26/02/2024

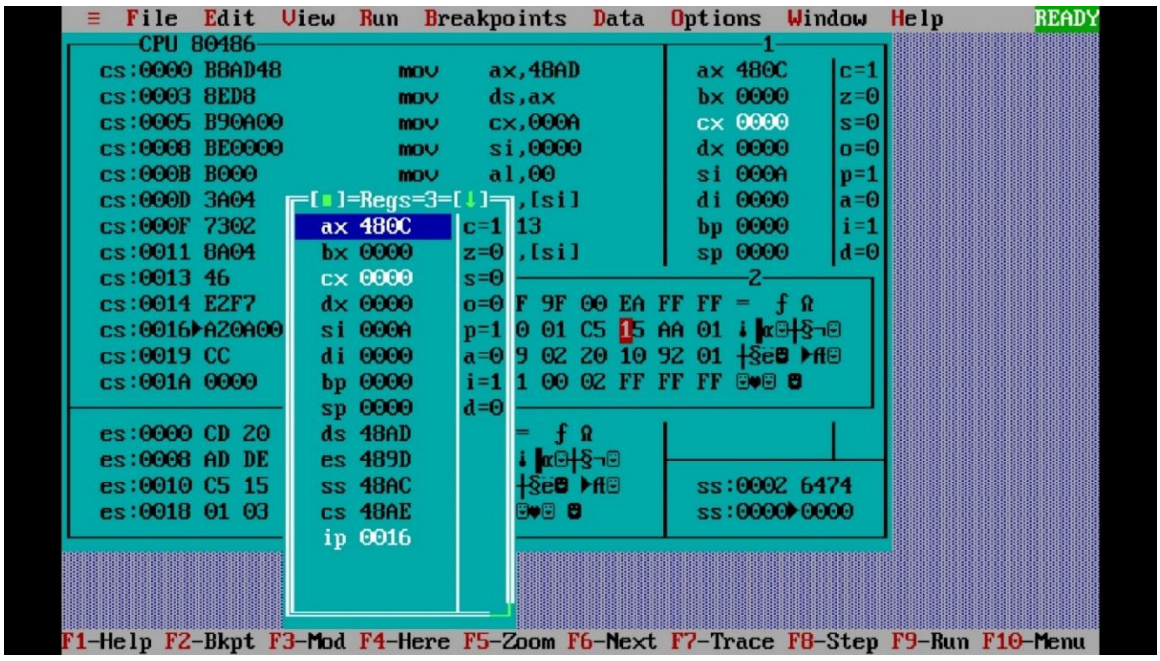
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 26/02/2024

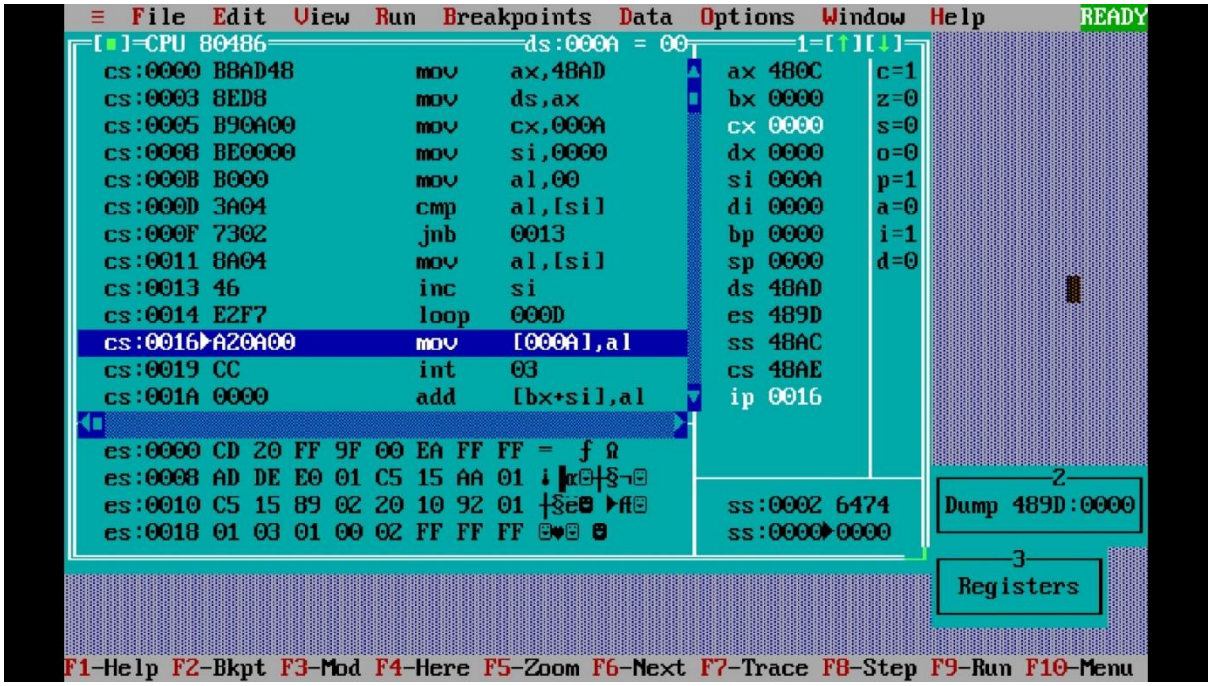
Assembly language code

```
data segment                    # Data segment

array db 01h,02h,03h,04h,05h,   # Declares an array named array with ten
07h,08h,0Ah,0Bh,0Ch                8-bit hexadecimal numbers.

larger db 00h                   # Declares a variable larger with an initial value of 00h.

data ends                       # Assemble directives

code segment
```

```
assume cs:code, ds:data      # Assemble directives defining cs and ds
start:                       # start of code segment

mov ax,data                  # moving data to ax register- initialization process
mov ds,ax                    # moving ax to dx register -- initialization process
mov cx,0Ah                   #Initializes a counter (cx) with the value 0Ah
mov si,offset array          # Uses si register to point to beginning of the array.
mov al,00h                   # Initializes the al register with the value 00h.
back: cmp al,[si]            # Enters a loop (back) that compares the value in al with
.                              the value at the memory location pointed to by si
      jnc skip               # If the value in al is not less than the value in the array .
.                              (jnc skip), it skips to the label skip
      mov al,[si]            # moves the value at the array location to al.


skip: inc si                 # Increments the si register
      loop back              # repeats the loop until cx becomes zero.

mov larger,al                # Stores the final value in al into the larger variable.
int 03h                      # breakpoint interrupt

code ends                    # Assembler directives to end code
end start                    # Assembler directives to end start
```

Result:





CONCLUSION: ***LO 2, LO 3 mapped.***
