***AIDS MICROPROCESSOR LAB S21 BATCH (2023-24)***

***Experiment 4(a) Title: Assembly language programming to sort numbers in ascending order using software tool TASM 1.4***

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

***Date of Performance: 04/03/2024***

***Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 04/03/2024***

***Assembly language code***

*data\_seg segment*

*ary db 14h, 12h, 21h, 17h, 06h, 01h, 32h, 2h, 22h, 11h*

*data\_seg ends*

*code\_seg segment*

*assume cs:code\_seg, ds:data\_seg #initialise segment registers*

*start:*

*mov ax,data\_seg #initialise data segment register*

*mov ds,ax;*

*mov ch, 09h #initialise the counter for outer loop*

*o\_loop: mov cl, 09h #initialise the counter for the inner loop*

*mov si, offset ary #initialise SI to point to the first element of the array*

*i\_loop: mov al, [si] #fetch element pointed by SI for comparison inc si #update SI*

*cmp al,[si] #compare element pointed by SI to current maximum jbe swap #if element is found <= current max, skip the swap xchg al,[si] #swap if new maximum is found*

*mov [si-1],al*

*swap: dec cl #decrement the counter for inner loop*

*jnz i\_loop;*

*dec ch #decrement counter for outer loop jnz o\_loop;*

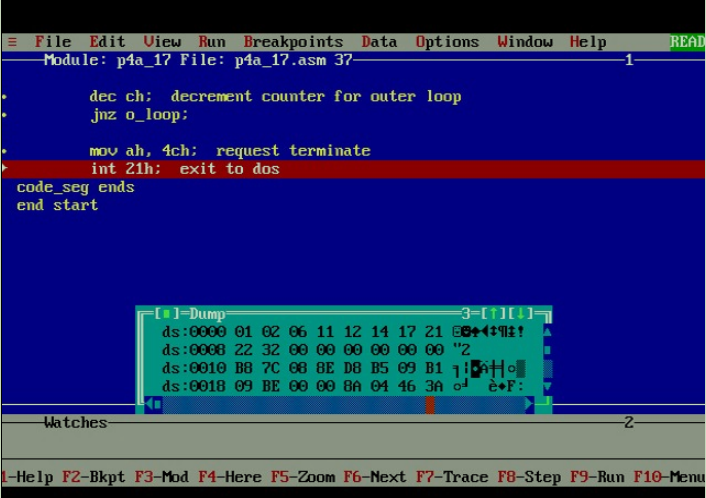
*mov ah, 4ch #request terminate*

*int 21h #exit to dos*

*code\_seg ends*

*end start*

*Result :*

**

***Experiment 4(b) Title: Assembly language programming to sort numbers in descending order using software tool TASM 1.4***

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

***Date of Performance: 04/03/2024***

***Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 04/03/2024***

***Assembly language code***

*data\_seg segment*

*ary db 14h, 12h, 21h, 17h, 06h, 01h, 32h, 2h, 22h, 11h*

*data\_seg ends*

*code\_seg segment*

*assume cs:code\_seg, ds:data\_seg #initialise segment registers*

*start: mov ax,data\_seg #initialise data segment register*

*mov ds,ax;*

*mov ch, 09h #initialise the counter for outer loop*

*o\_loop: mov cl, 09h #initialise the counter for the inner loop*

*mov si, offset ary #initialise SI to point to the first element of the array*

*i\_loop: mov al, [si] #fetch element pointed by SI for comparison*

*inc si #update SI*

*cmp al,[si] #compare element pointed by SI to current maximum*

*jae swap #if element is found <= current max, skip the swap*

*xchg al,[si] #swap if new maximum is found*

*mov [si-1],al*

*swap: dec cl #decrement the counter for inner loop*

*jnz i\_loop*

*dec ch #decrement counter for outer loop*

*jnz o\_loop*

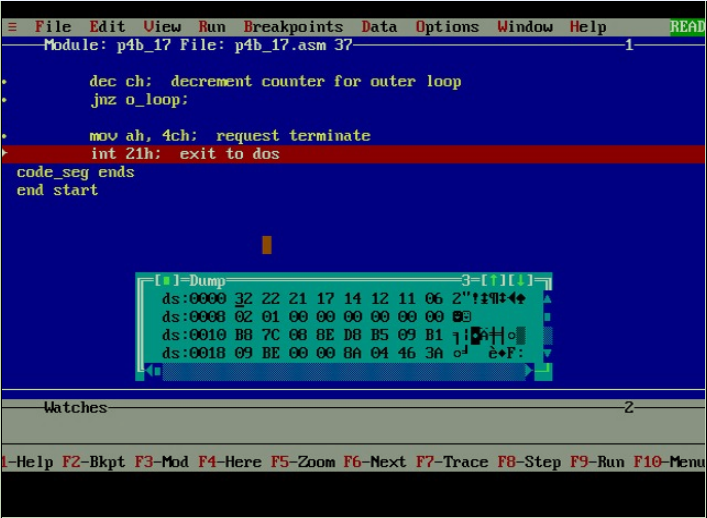
*mov ah, 4ch #request terminate*

*int 21h #exit to dos*

*code\_seg ends*

*end start*

*Result:*

**

*CONCLUSION: LO 2, LO 3 mapped.*

***---------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-----------------------------------***