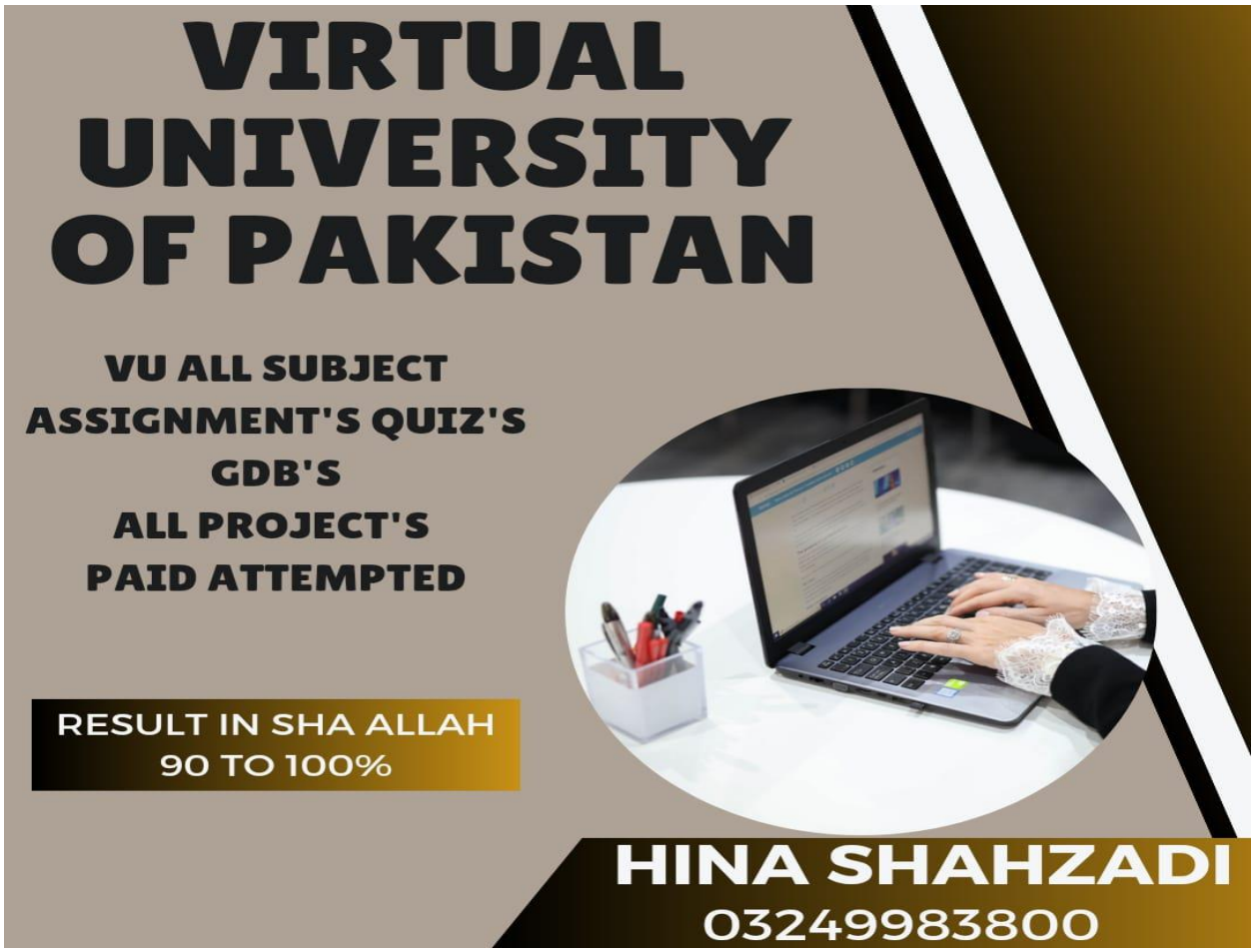
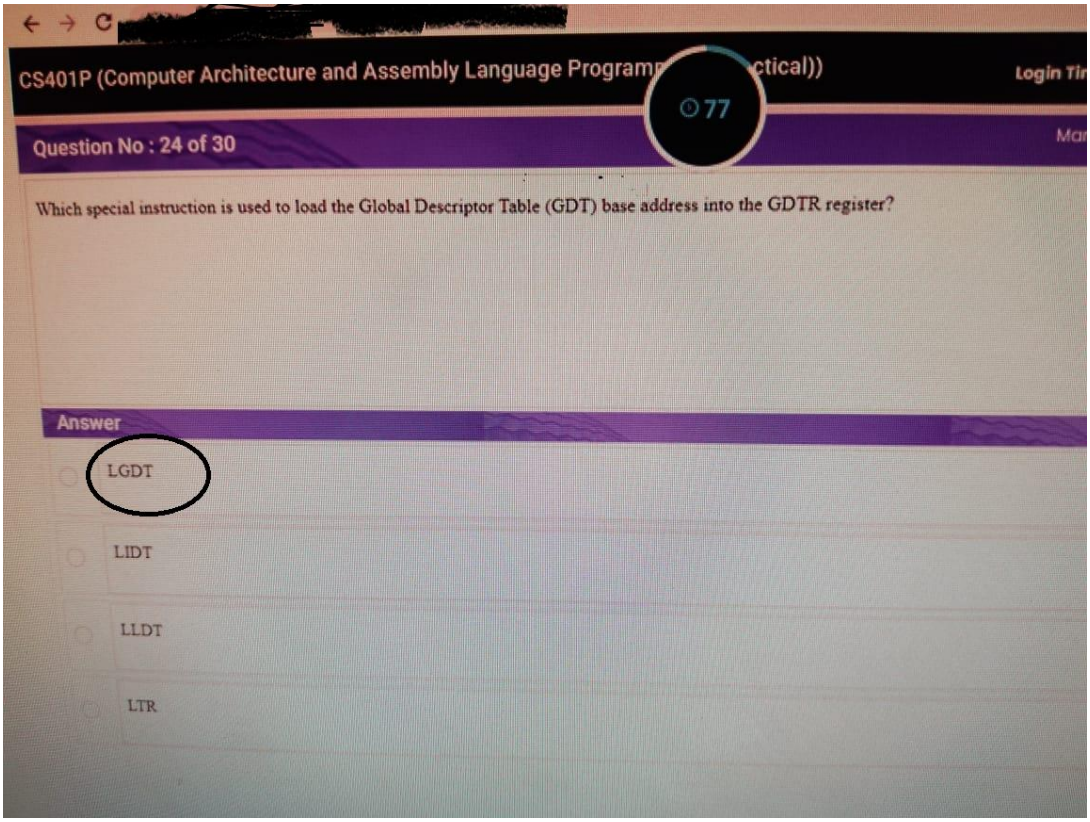
Solved By Hina Shahzadi

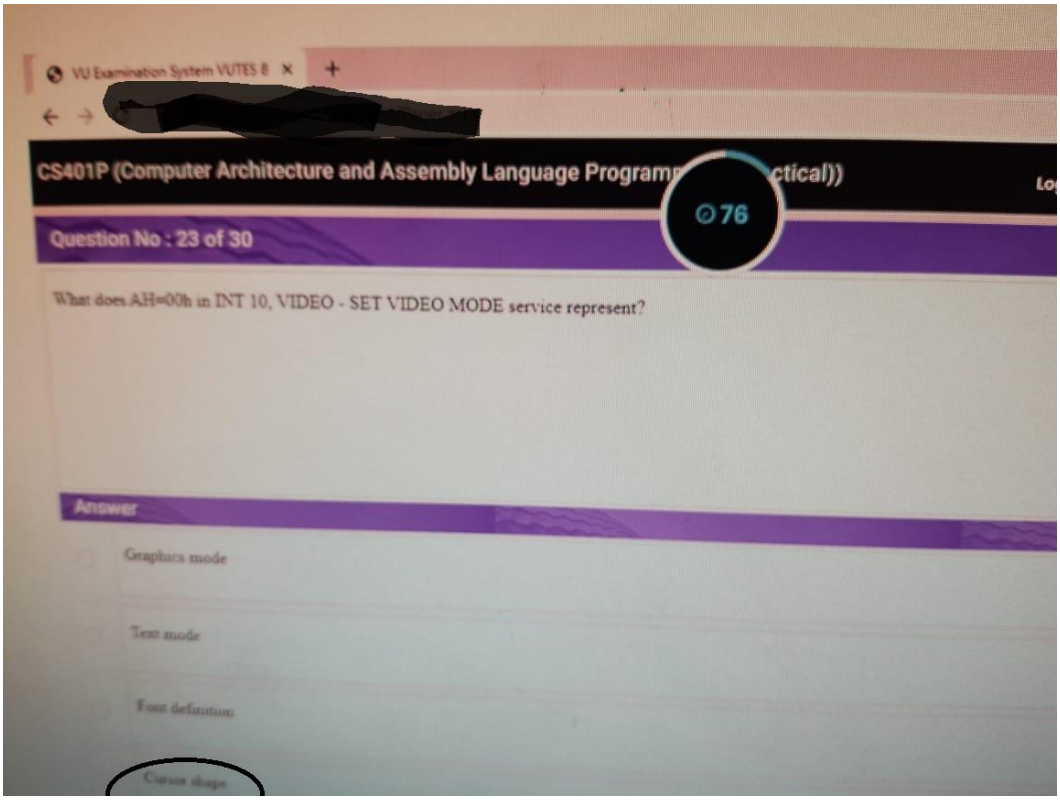
Current Paper CS401P Final Term



Solved By Hina Shahzadi

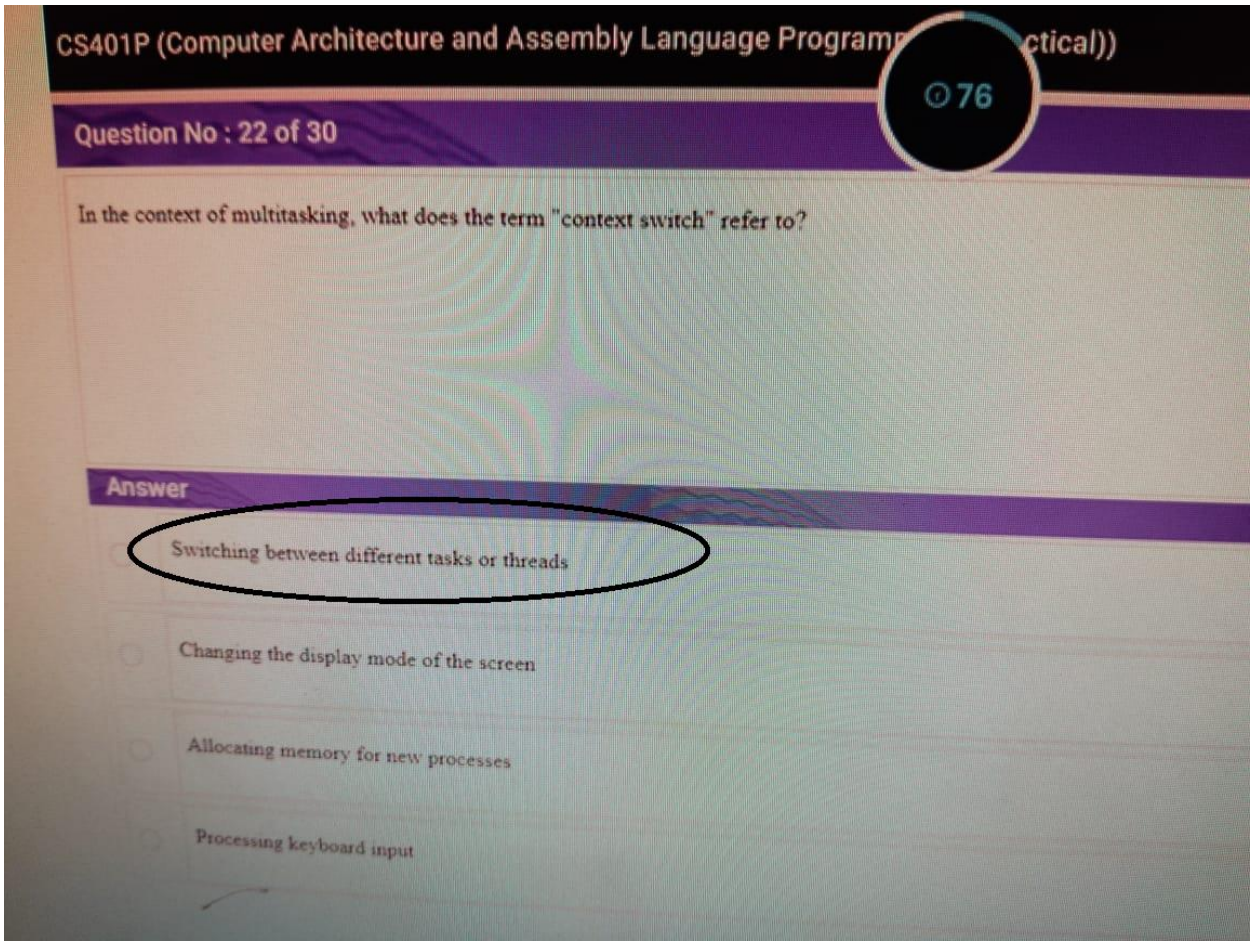


Solved By Hina Shahzadi

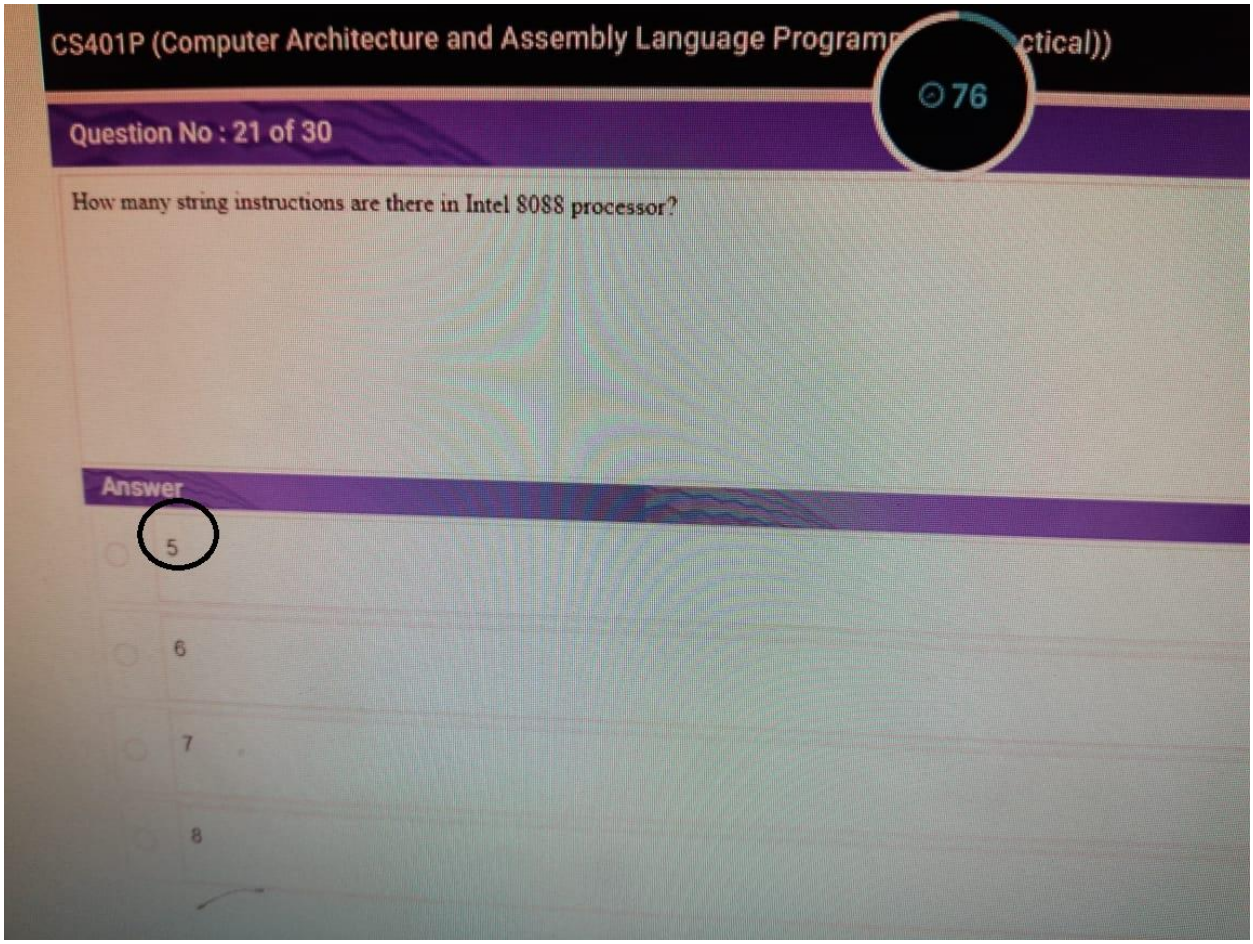


INT 10 - VIDEO - SET TEXT-MODE CURSOR SHAPE

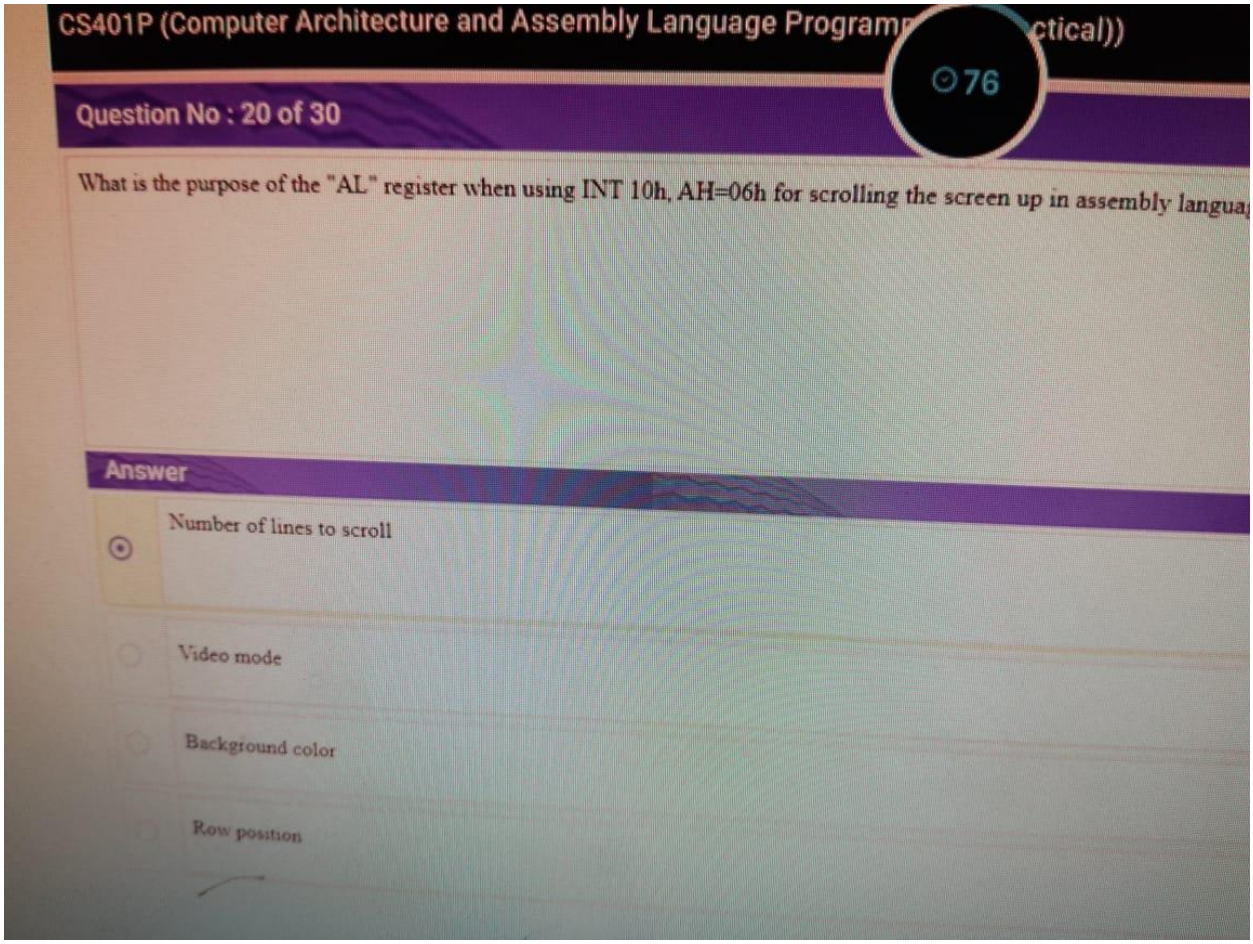
Solved By Hina Shahzadi



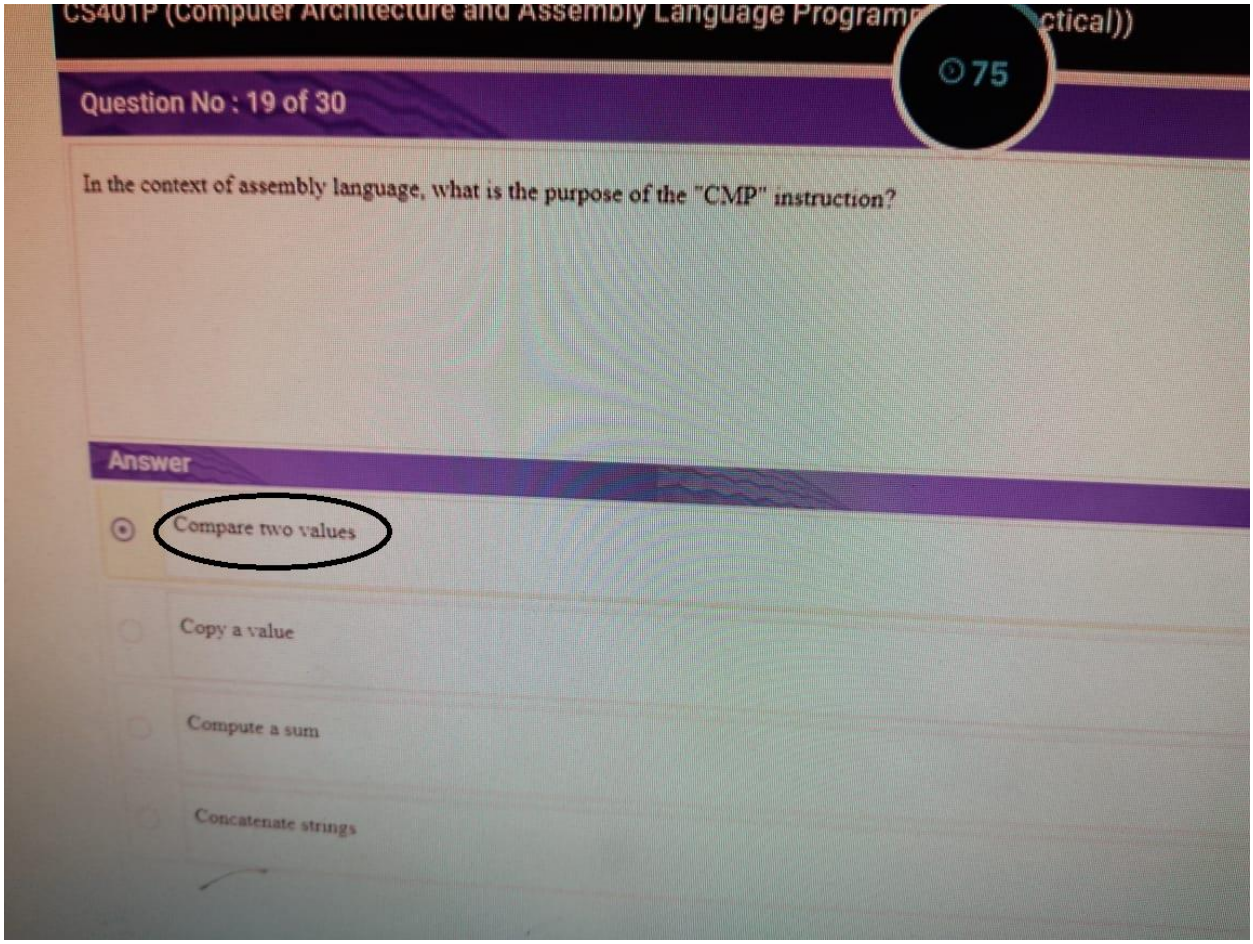
Solved By Hina Shahzadi



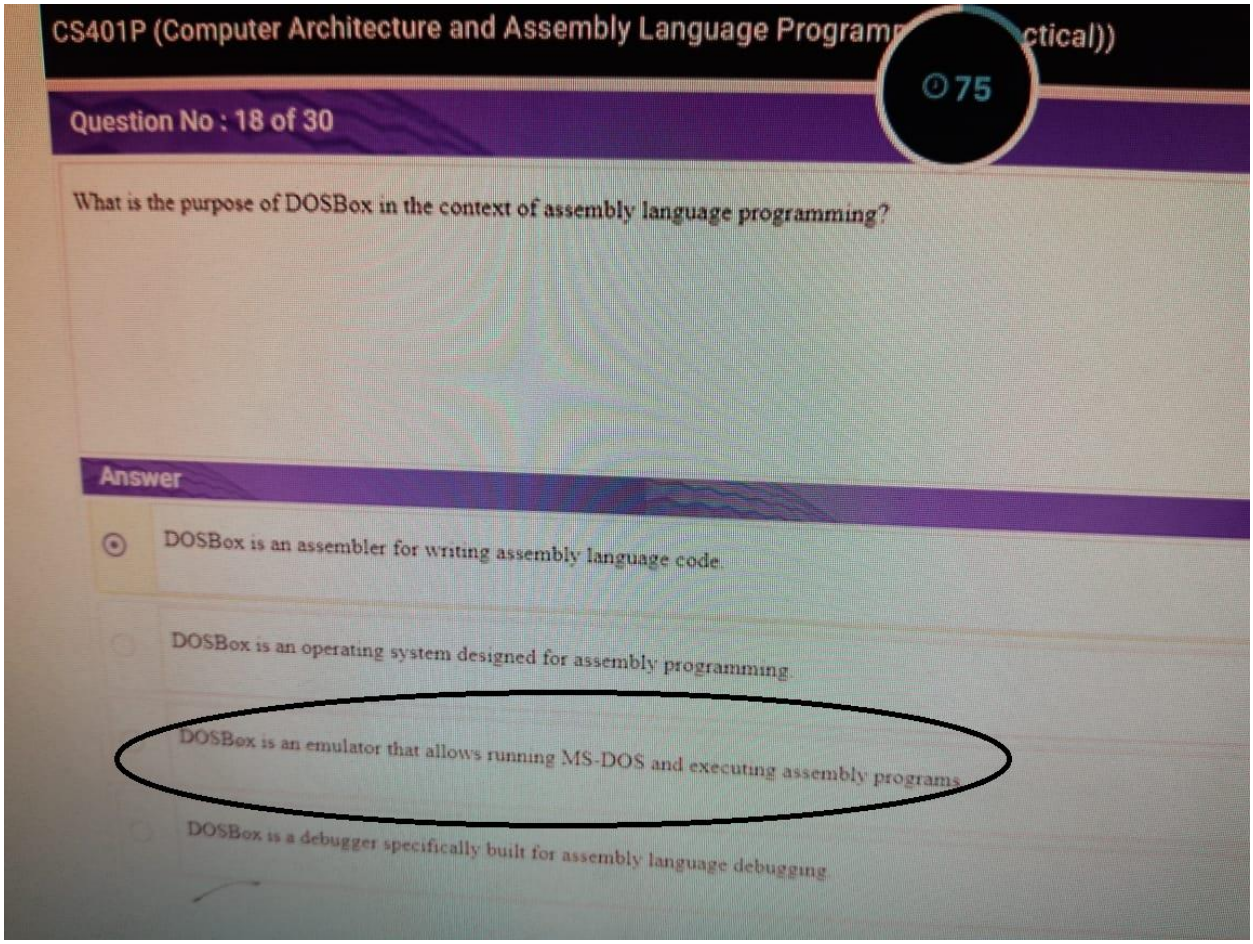
Solved By Hina Shahzadi



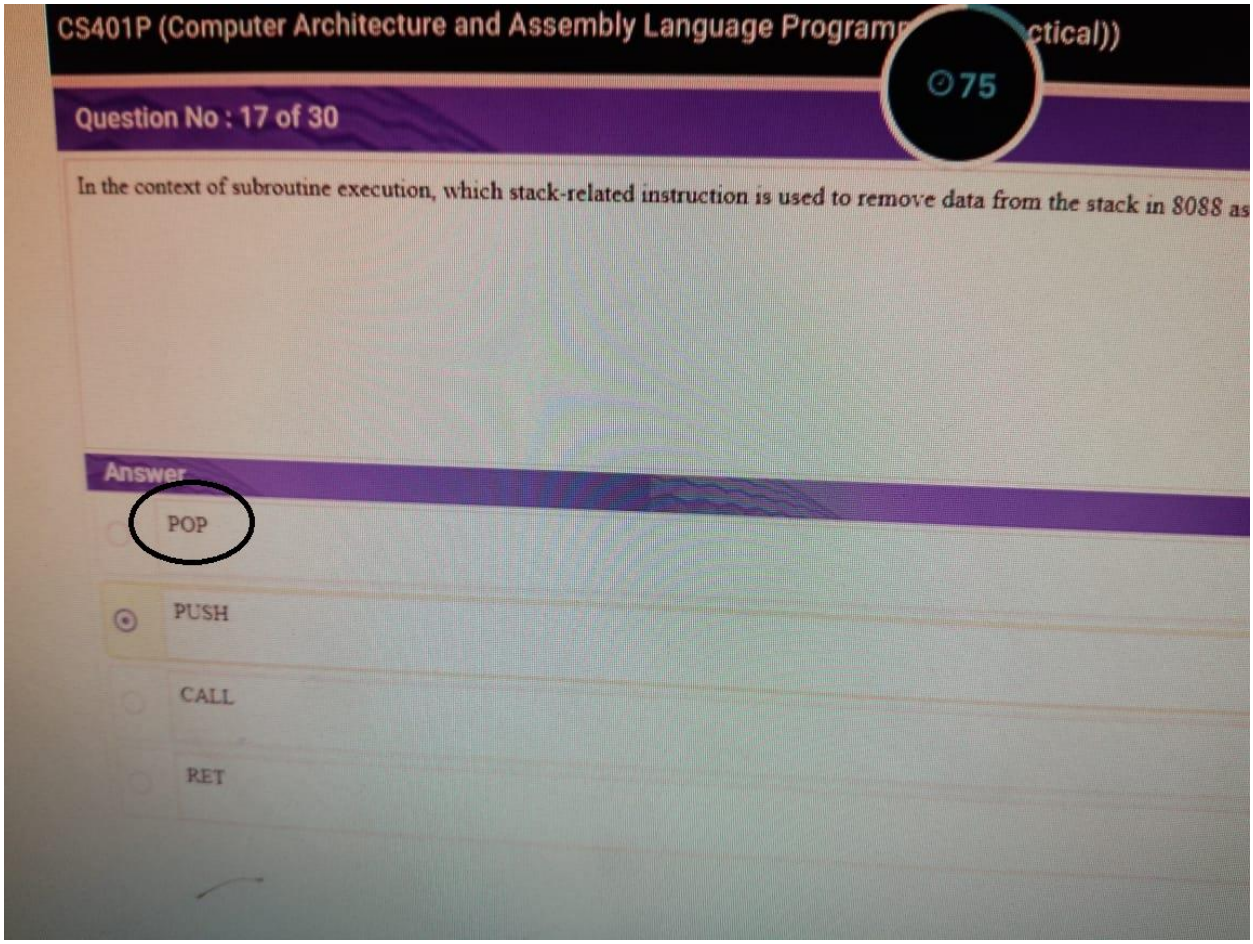
Solved By Hina Shahzadi



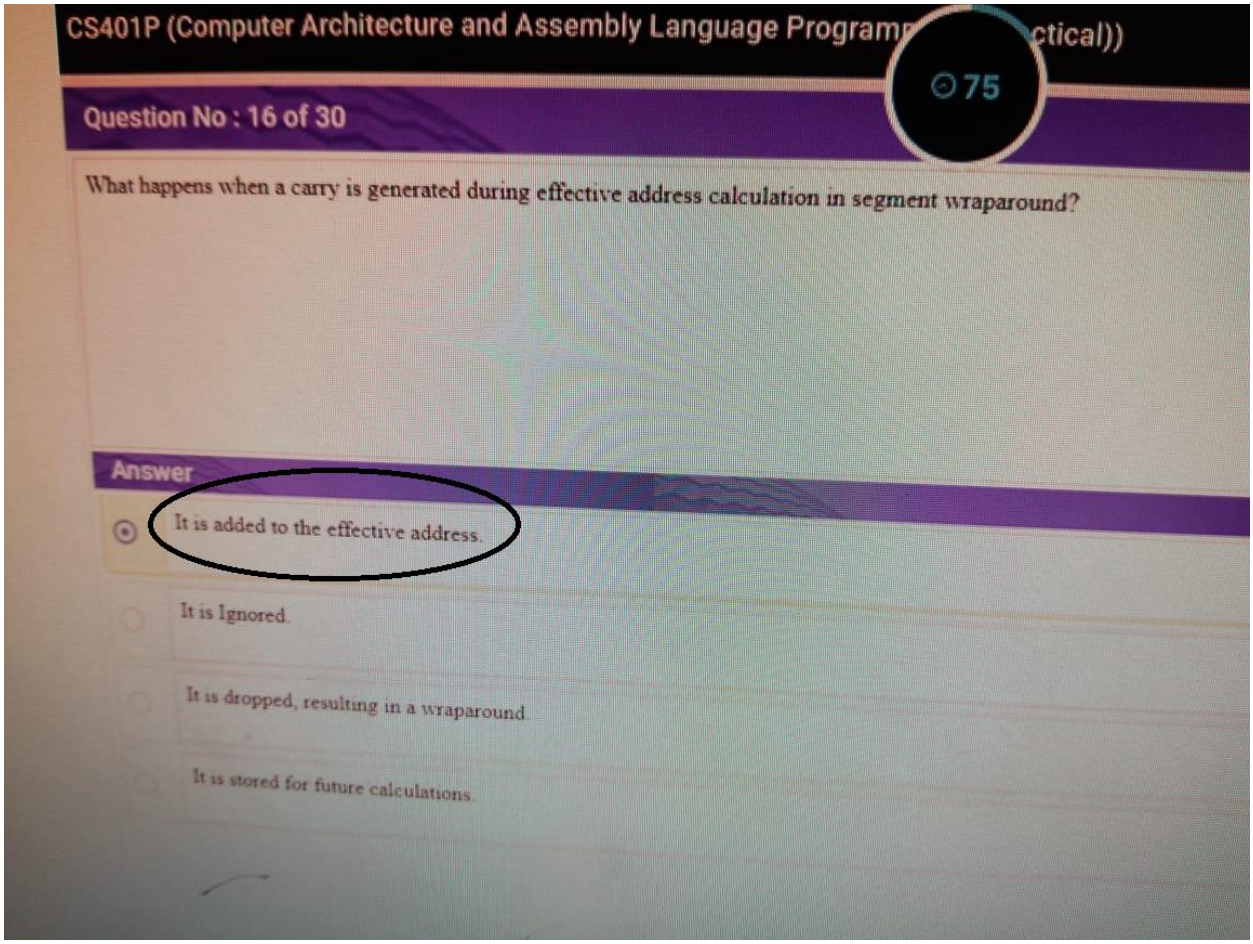
Solved By Hina Shahzadi



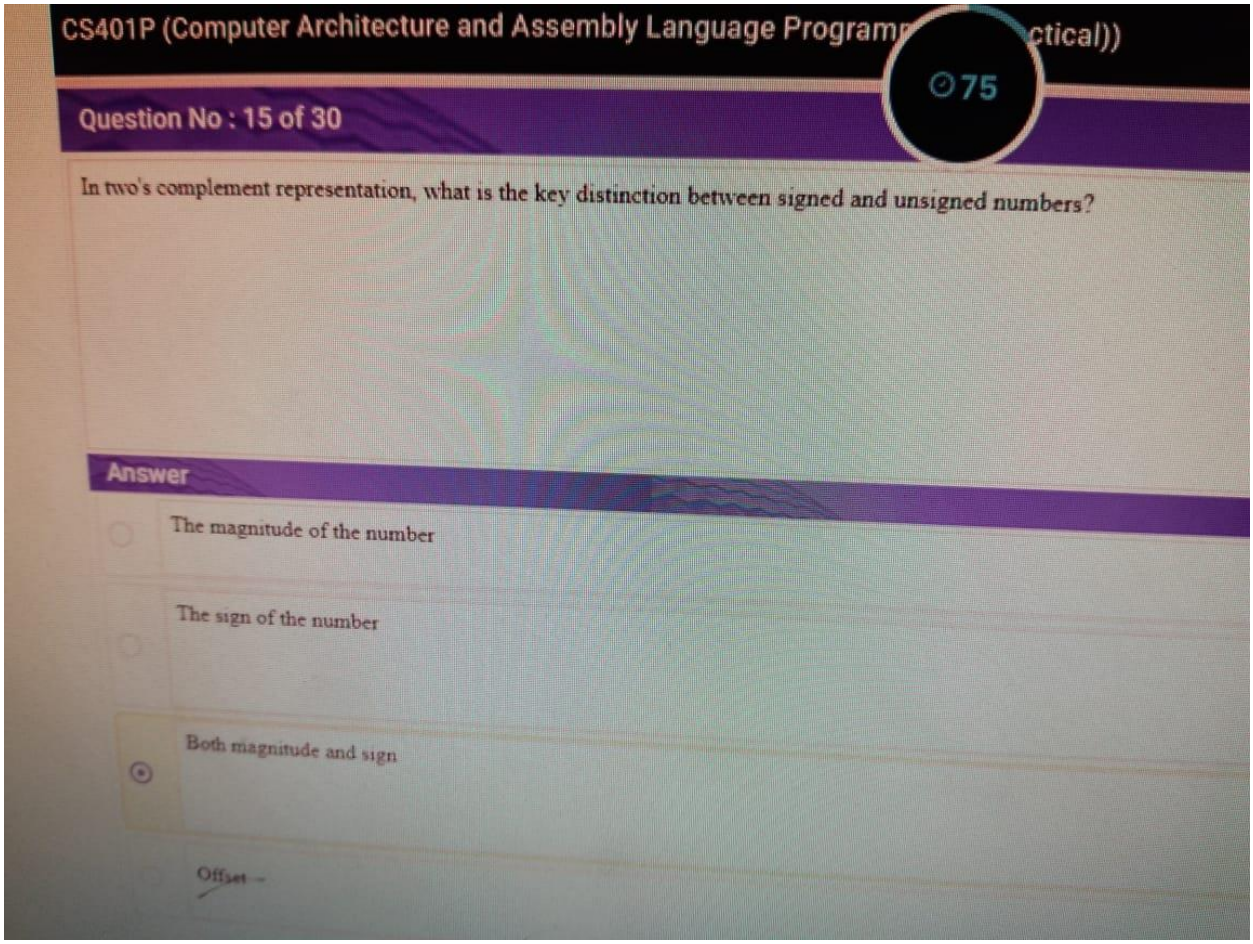
Solved By Hina Shahzadi



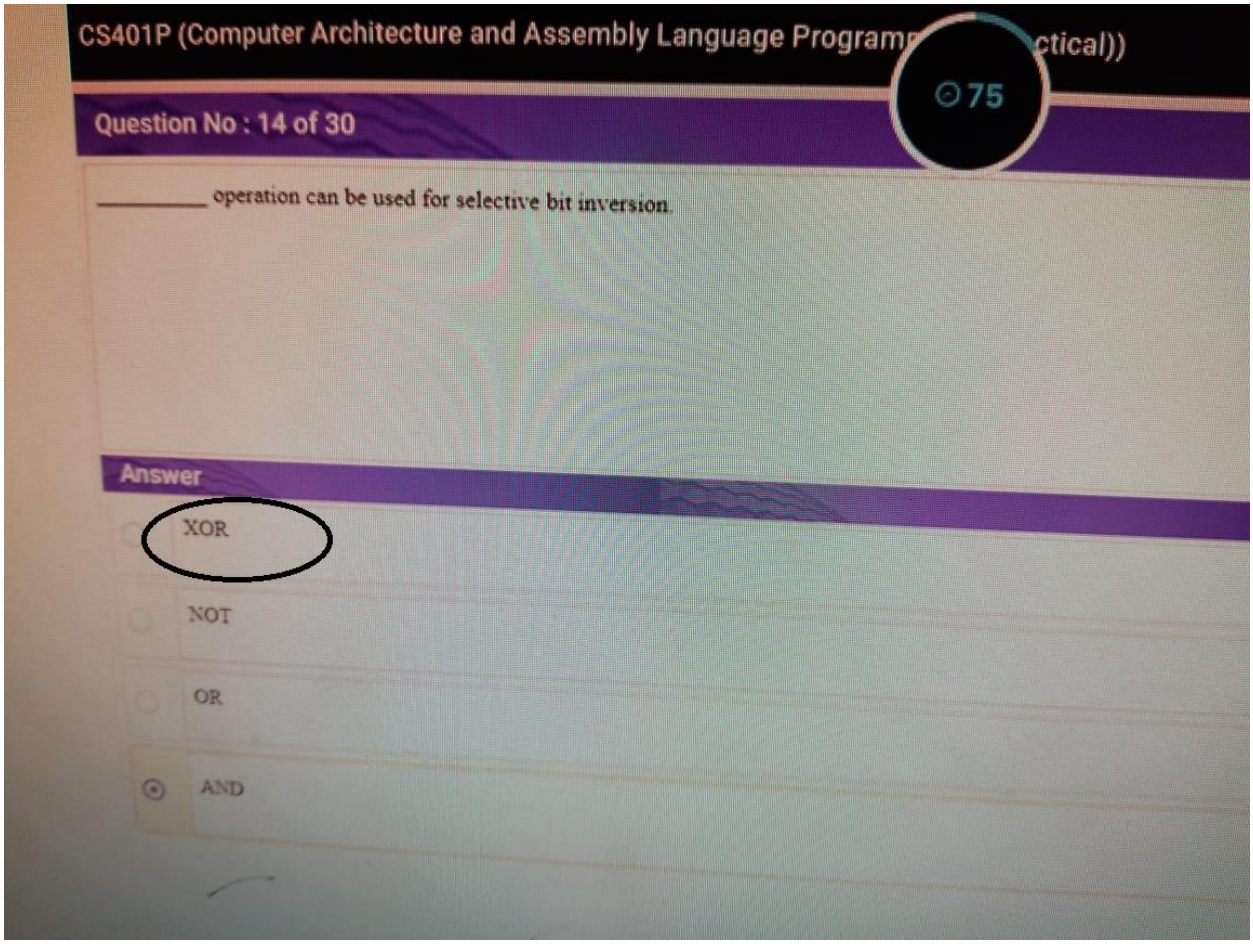
Solved By Hina Shahzadi



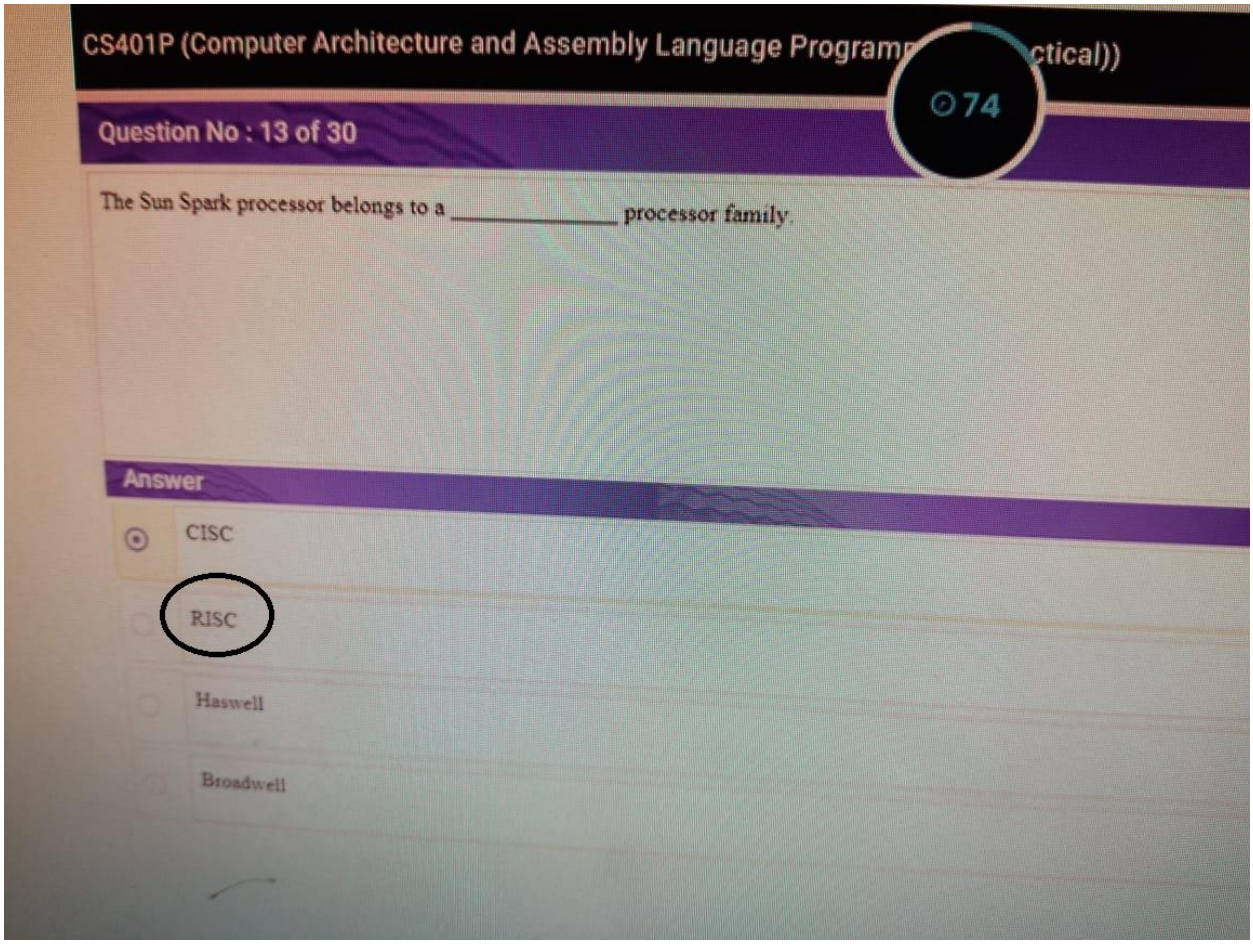
Solved By Hina Shahzadi



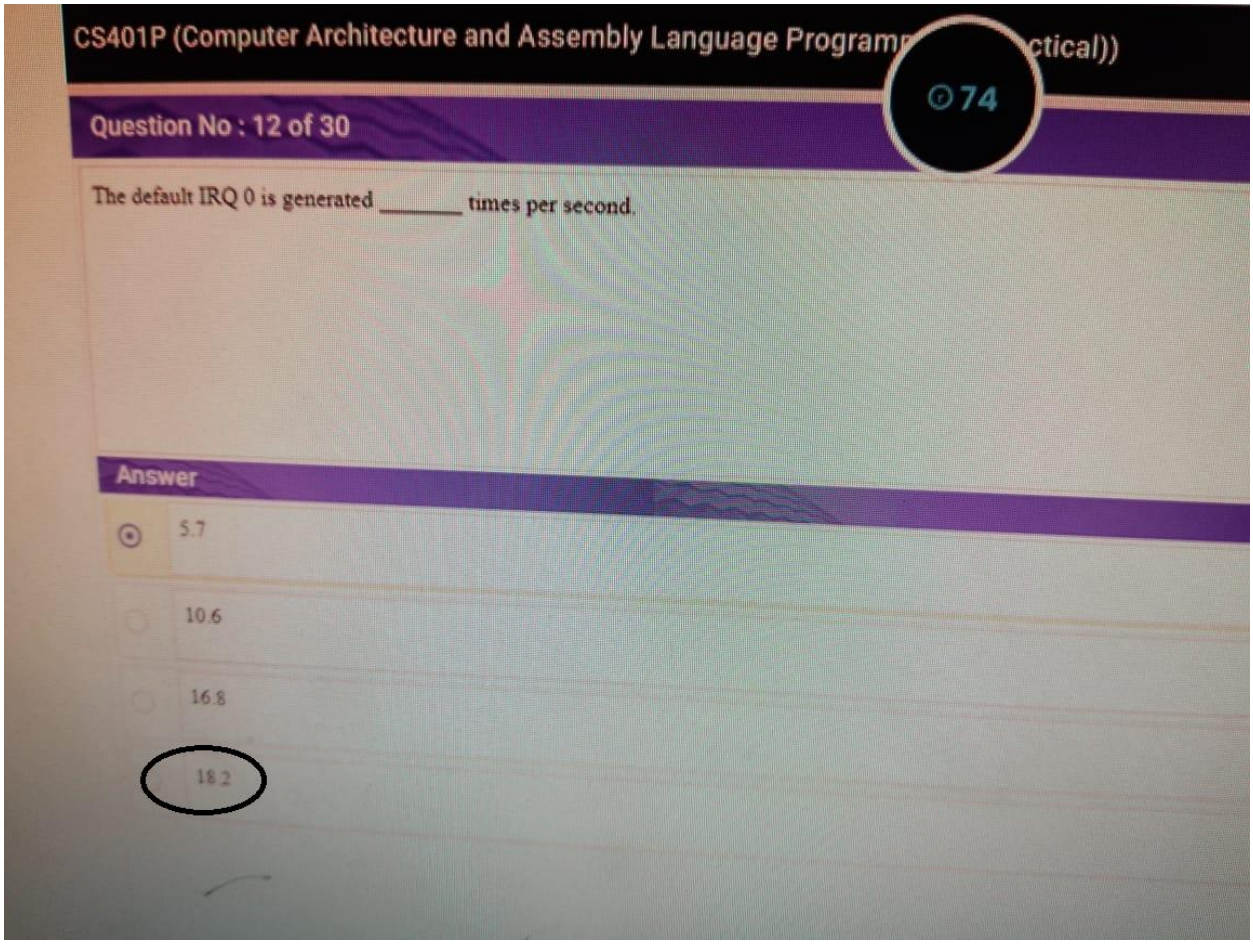
Solved By Hina Shahzadi



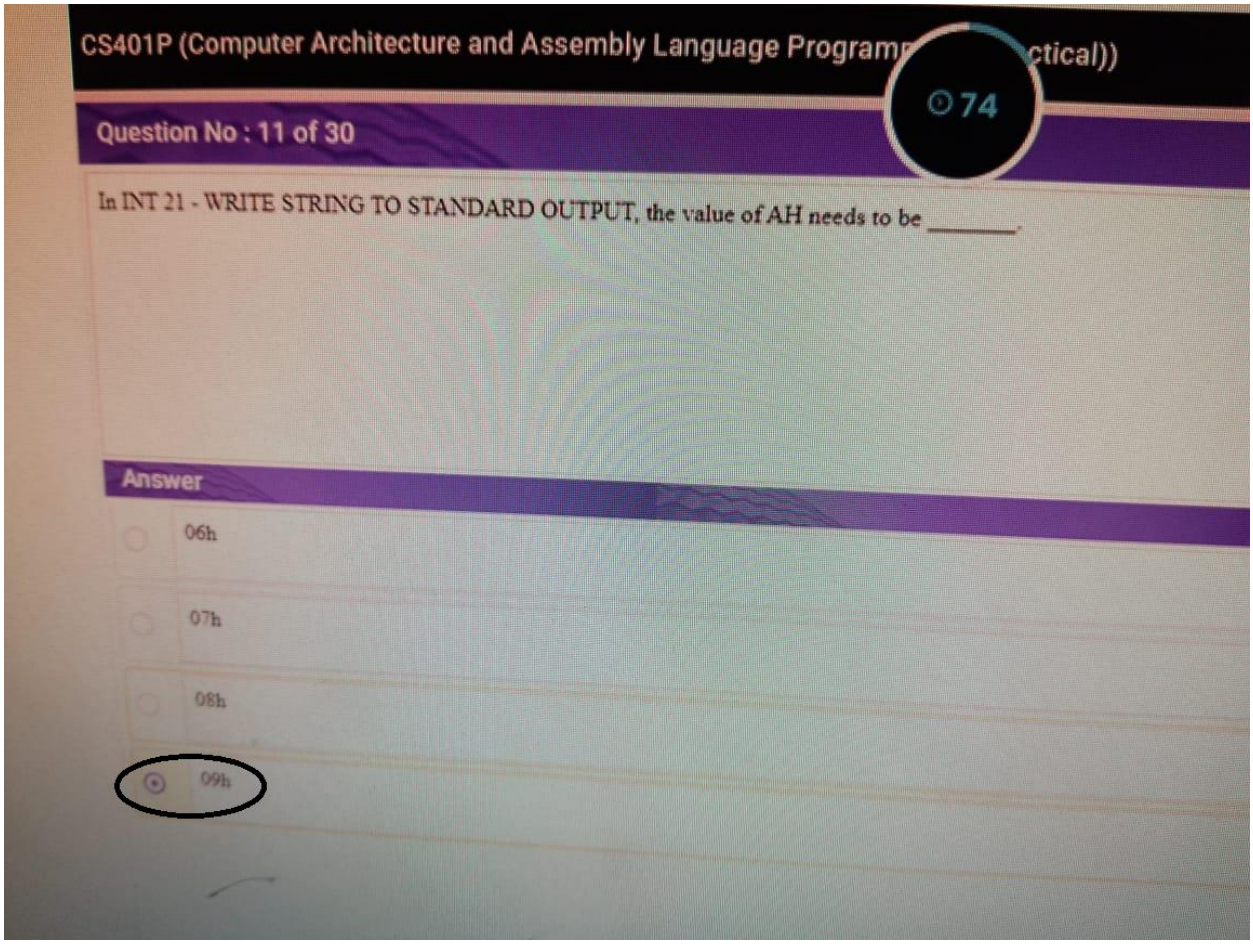
Solved By Hina Shahzadi



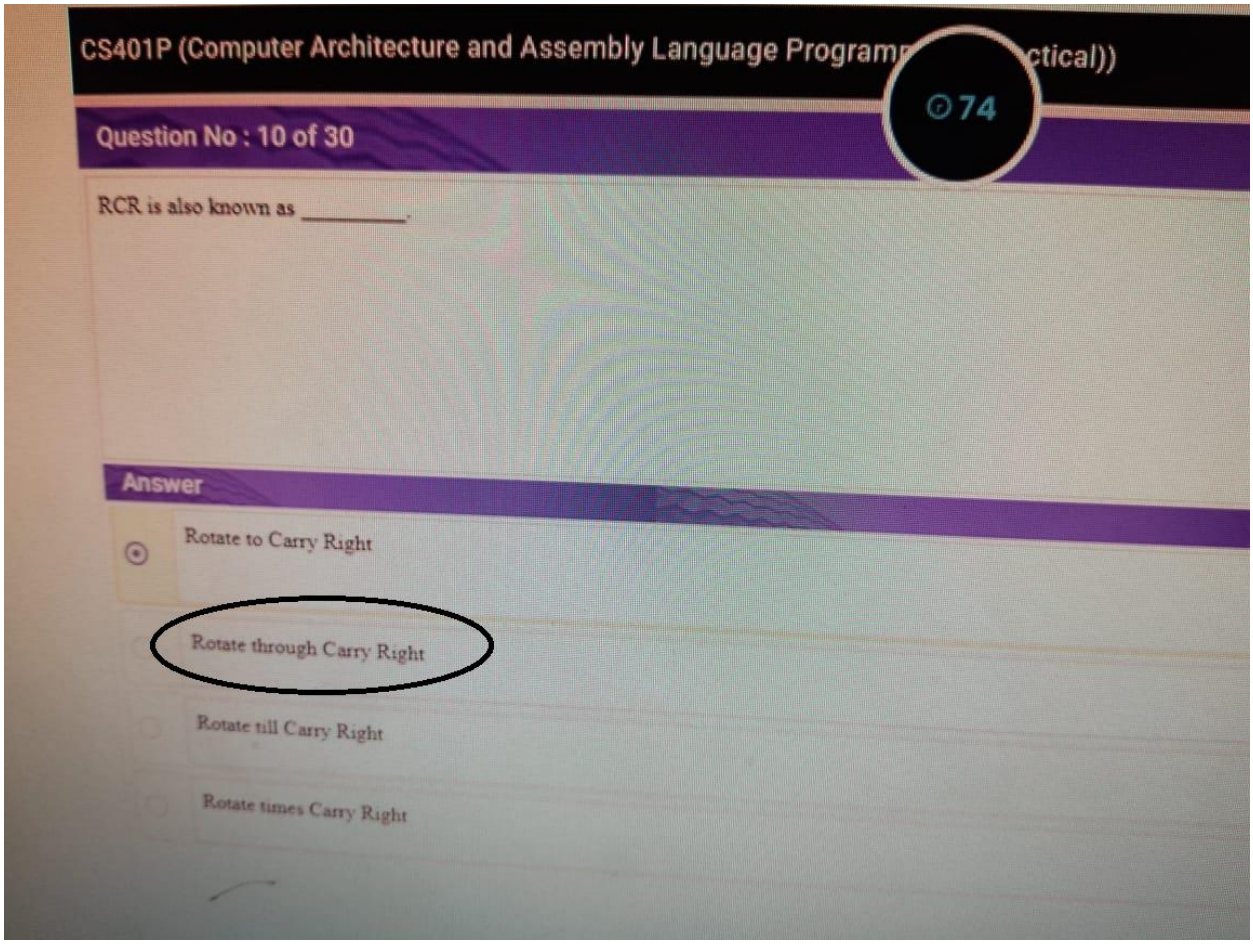
Solved By Hina Shahzadi



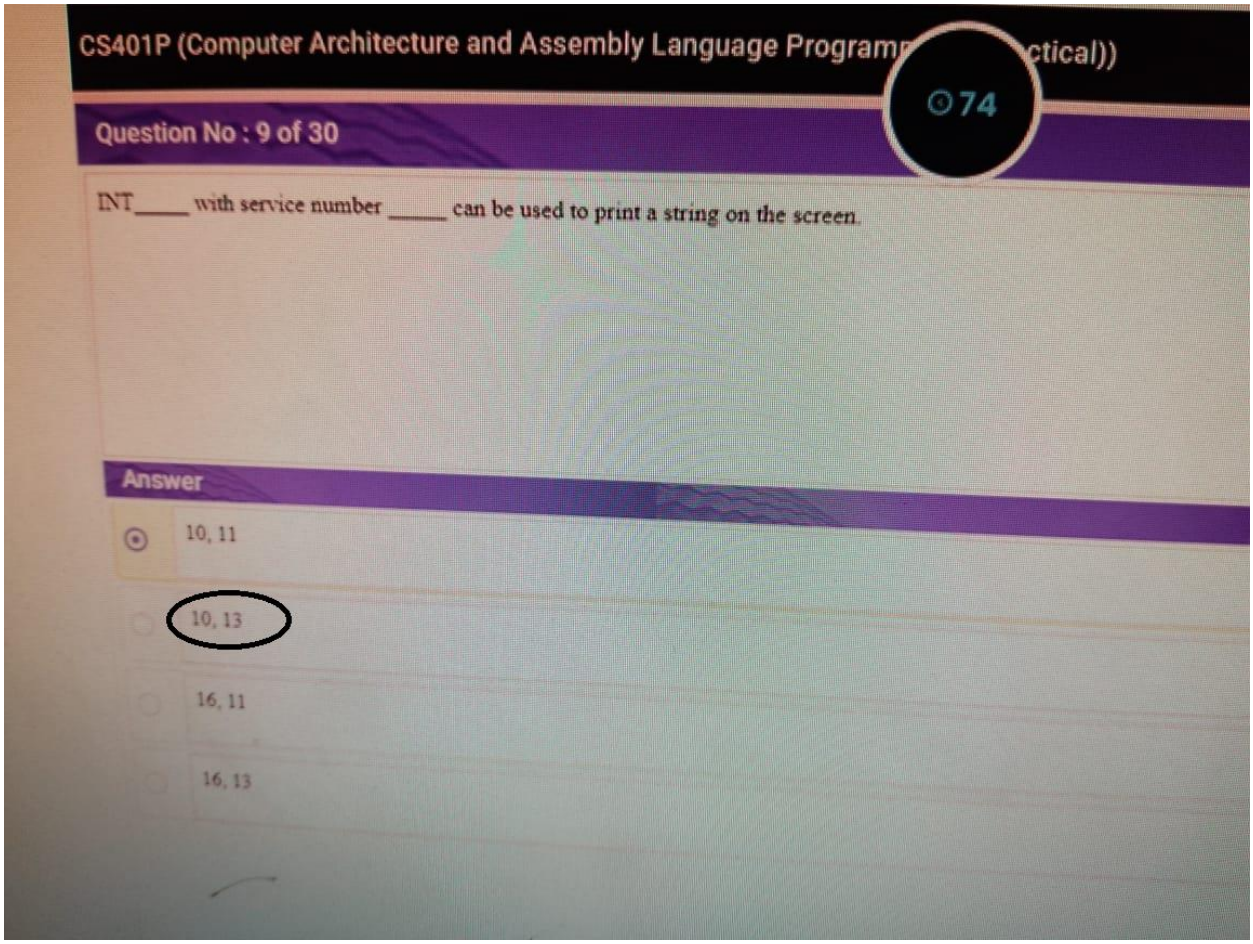
Solved By Hina Shahzadi



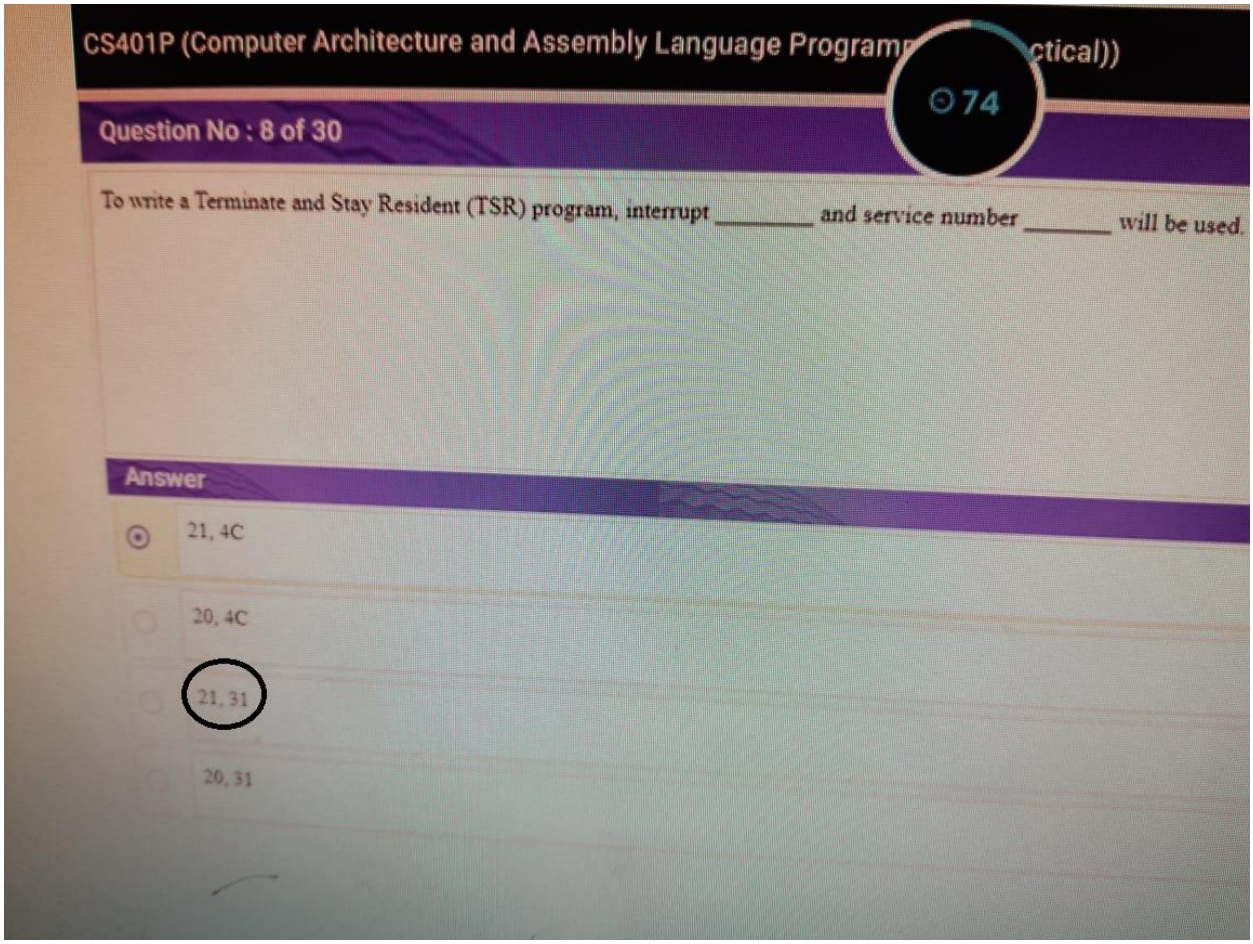
Solved By Hina Shahzadi



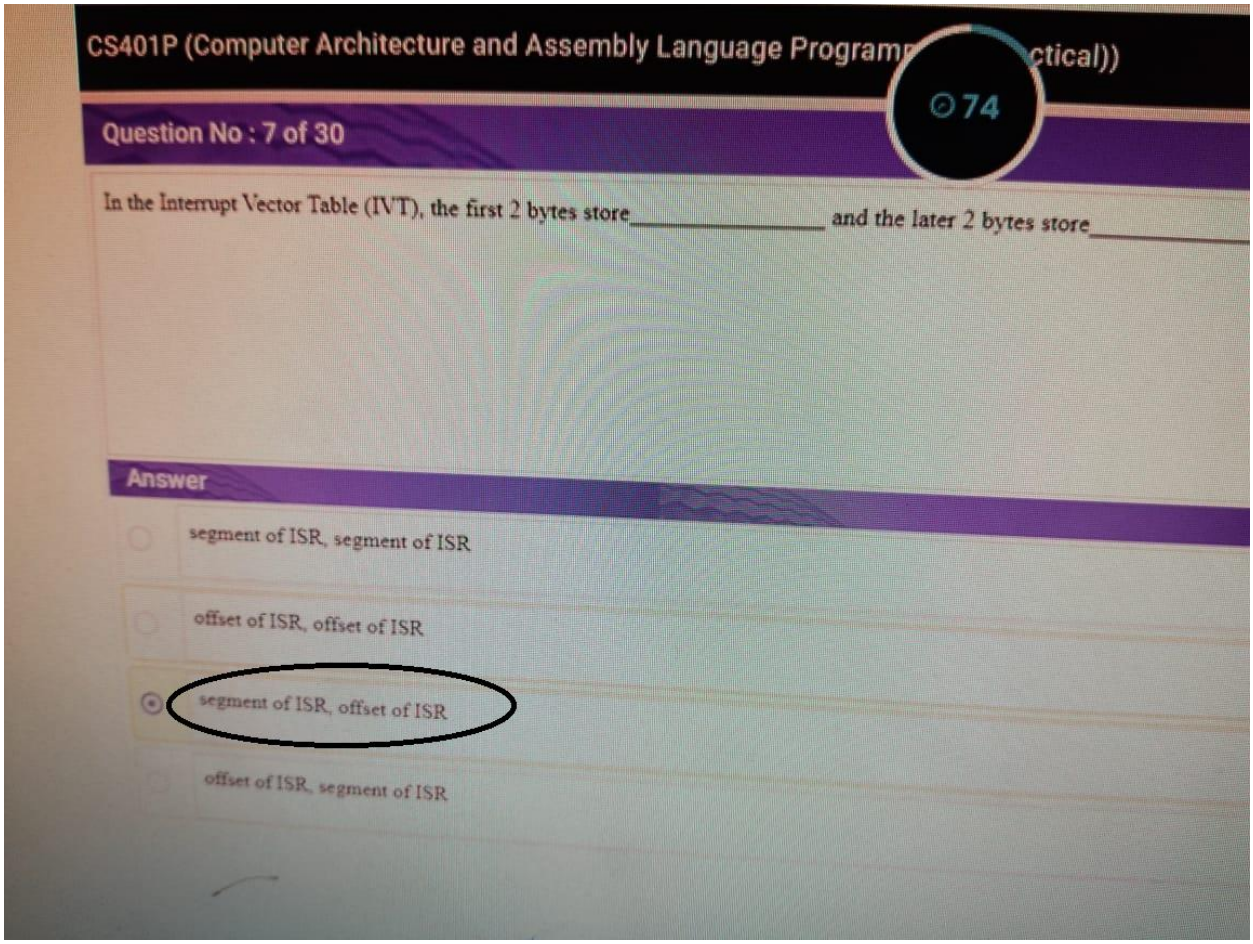
Solved By Hina Shahzadi



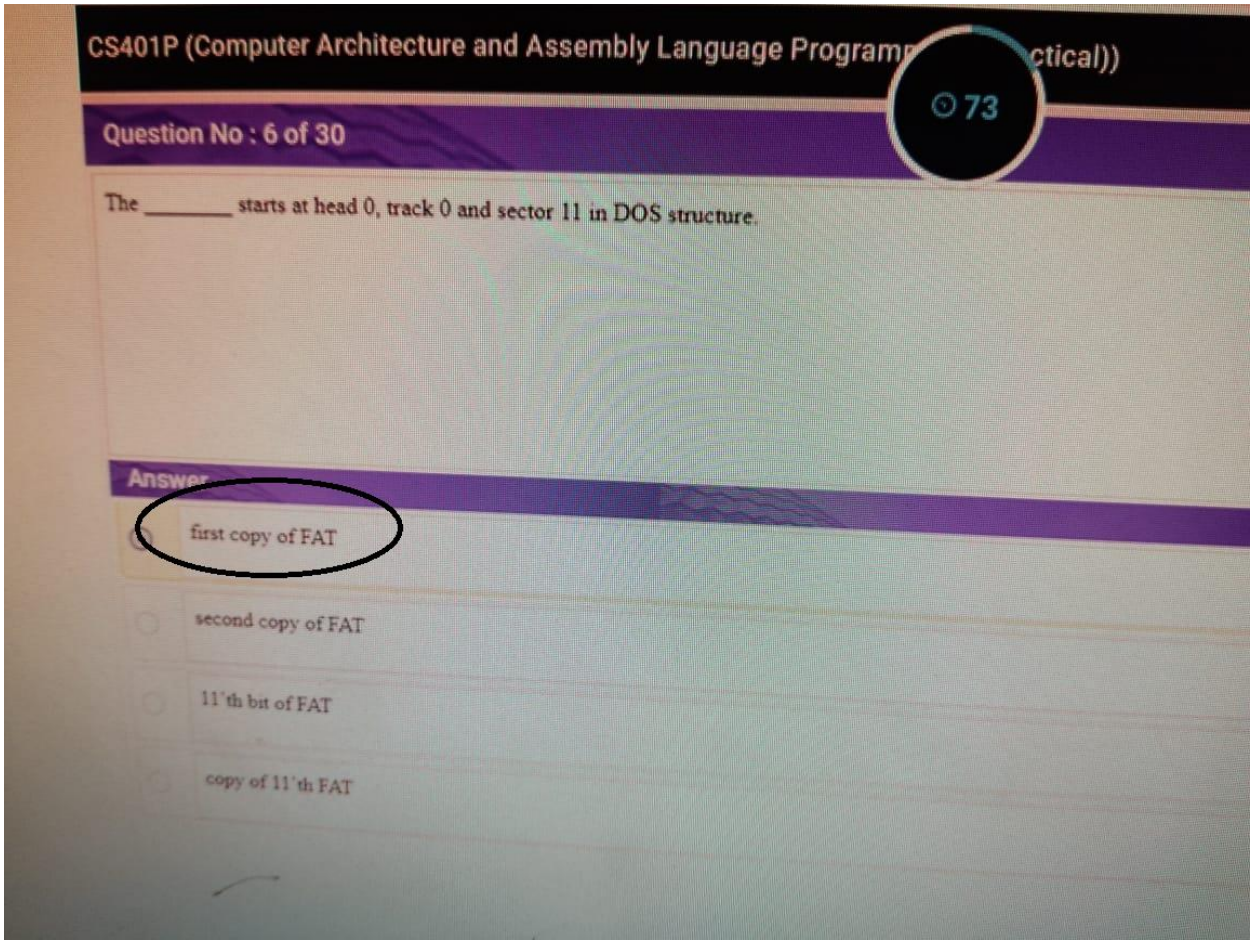
Solved By Hina Shahzadi



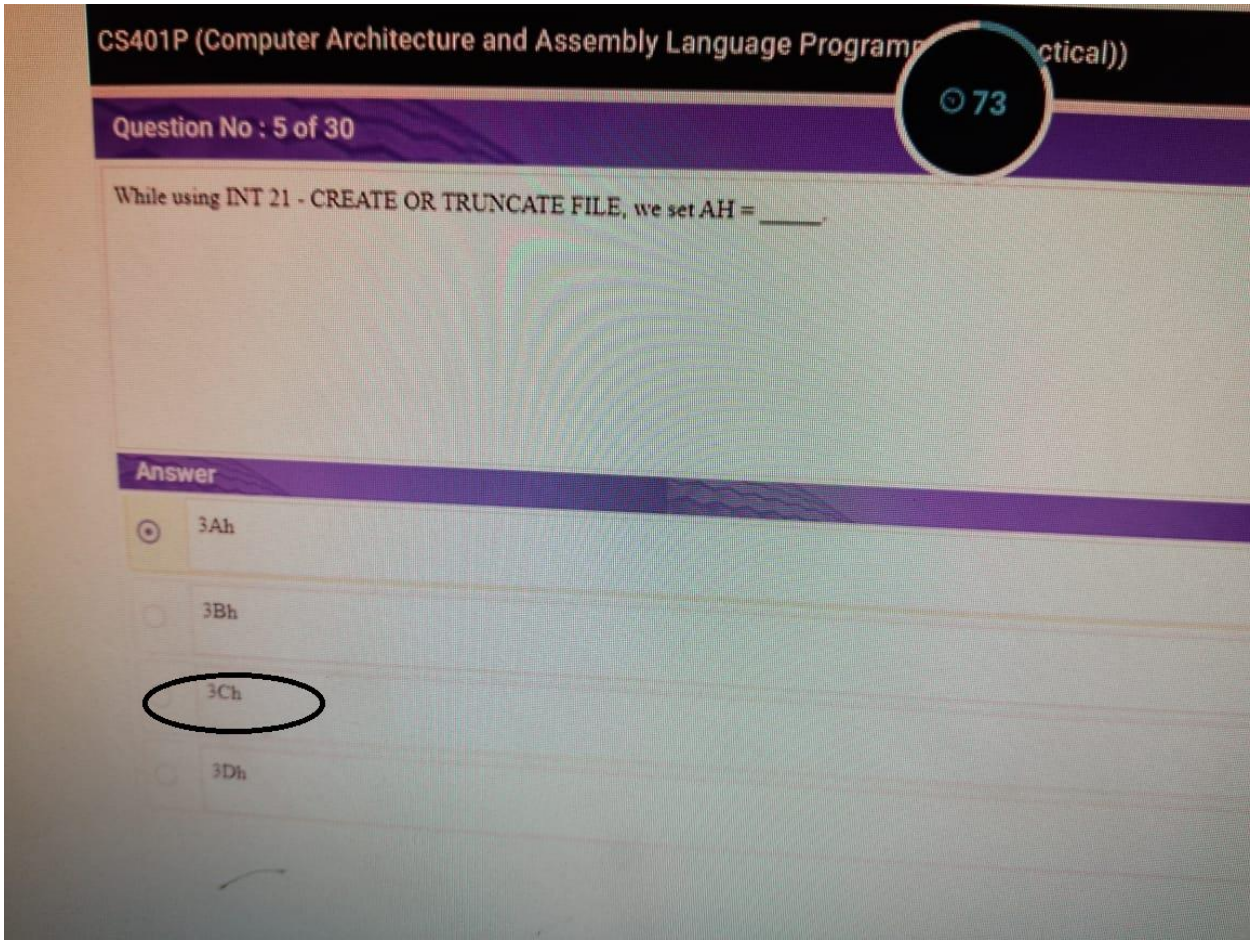
Solved By Hina Shahzadi



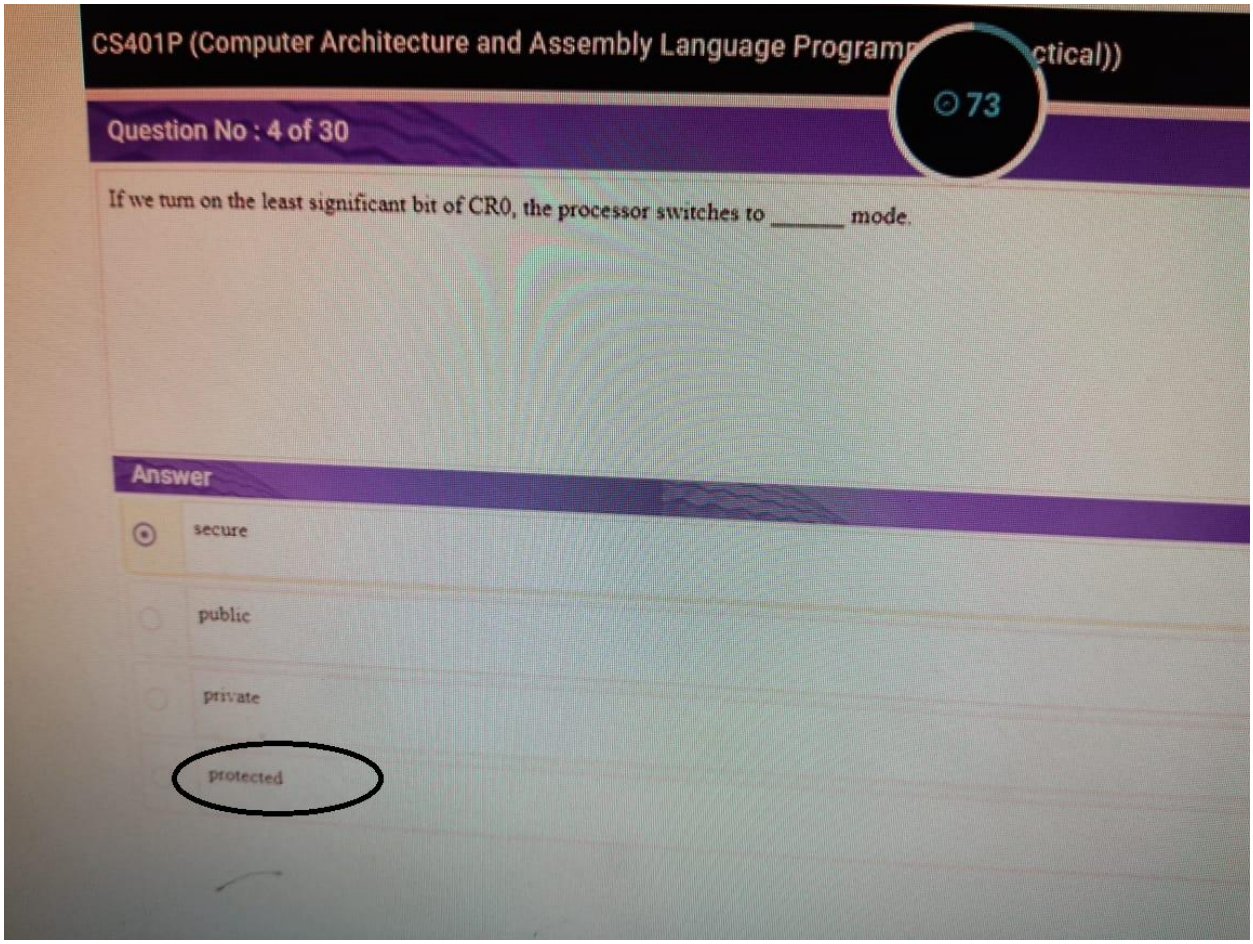
Solved By Hina Shahzadi



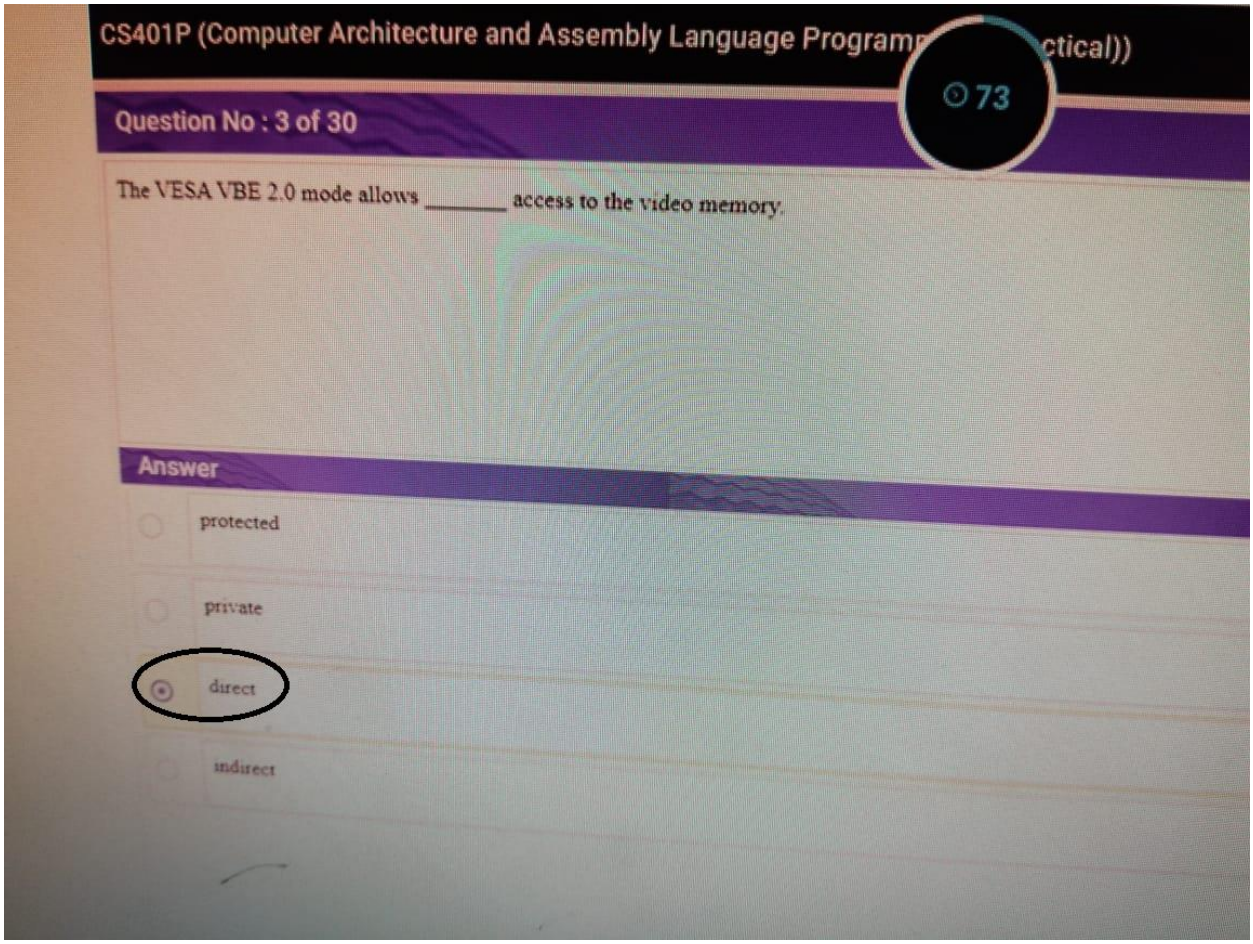
Solved By Hina Shahzadi



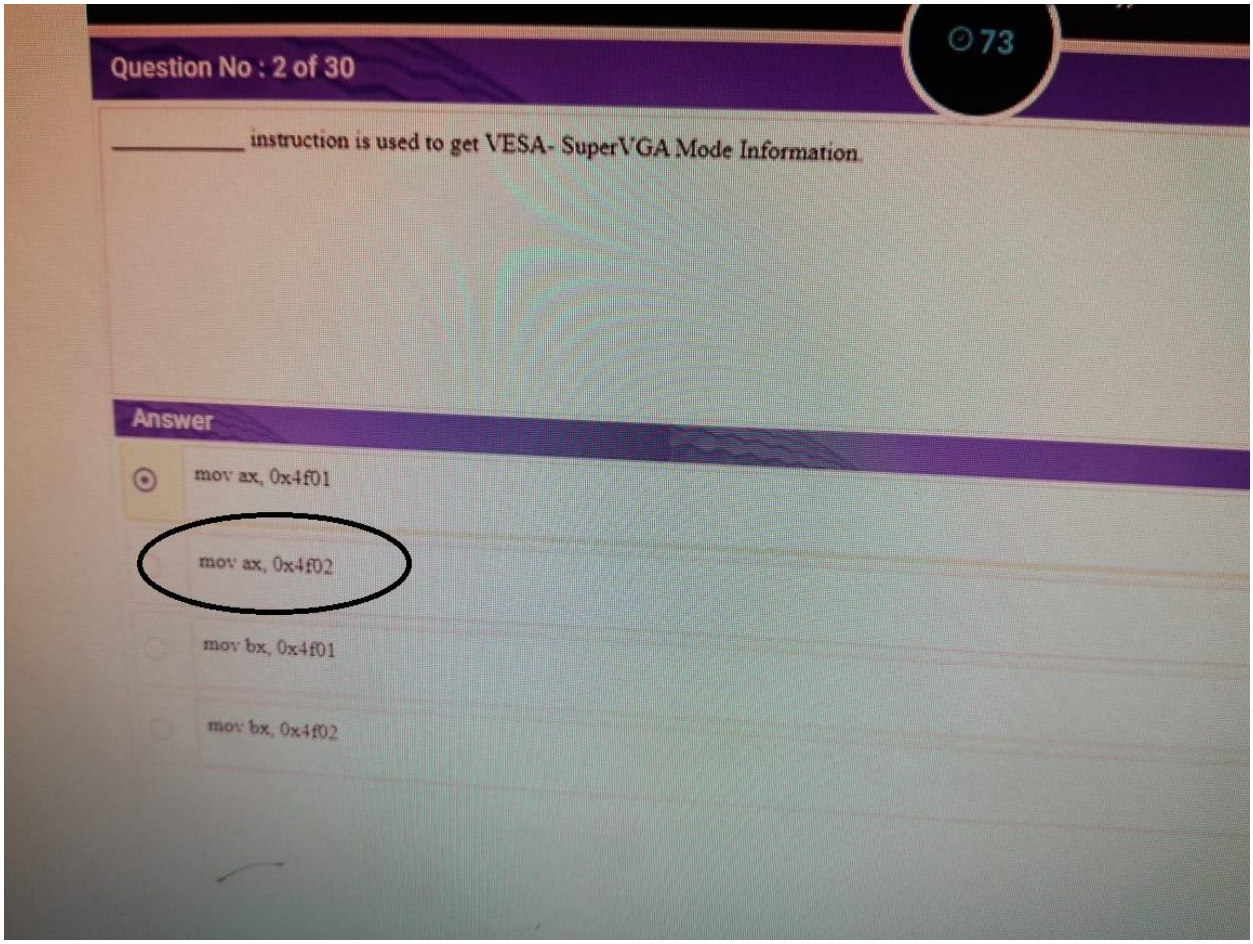
Solved By Hina Shahzadi



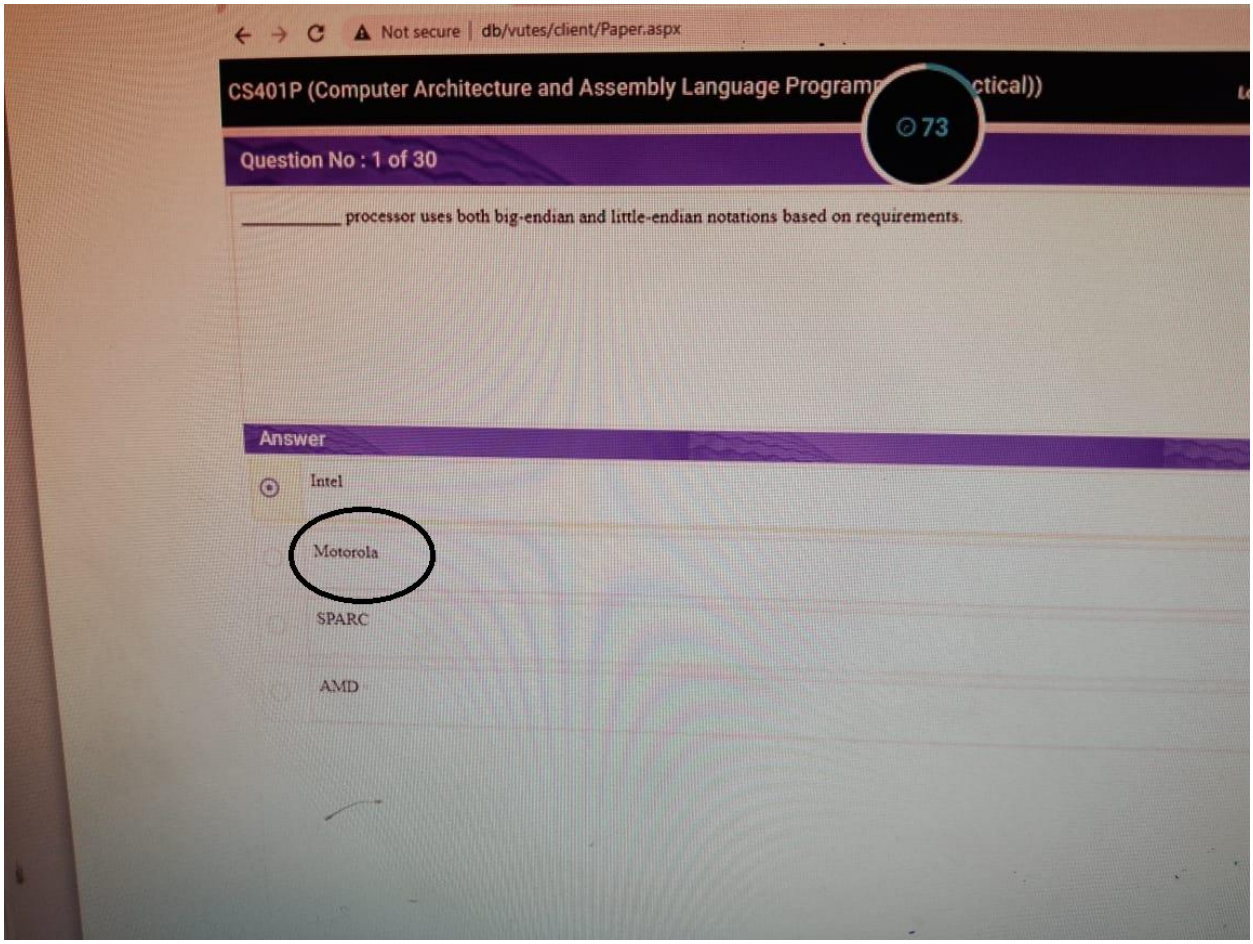
Solved By Hina Shahzadi



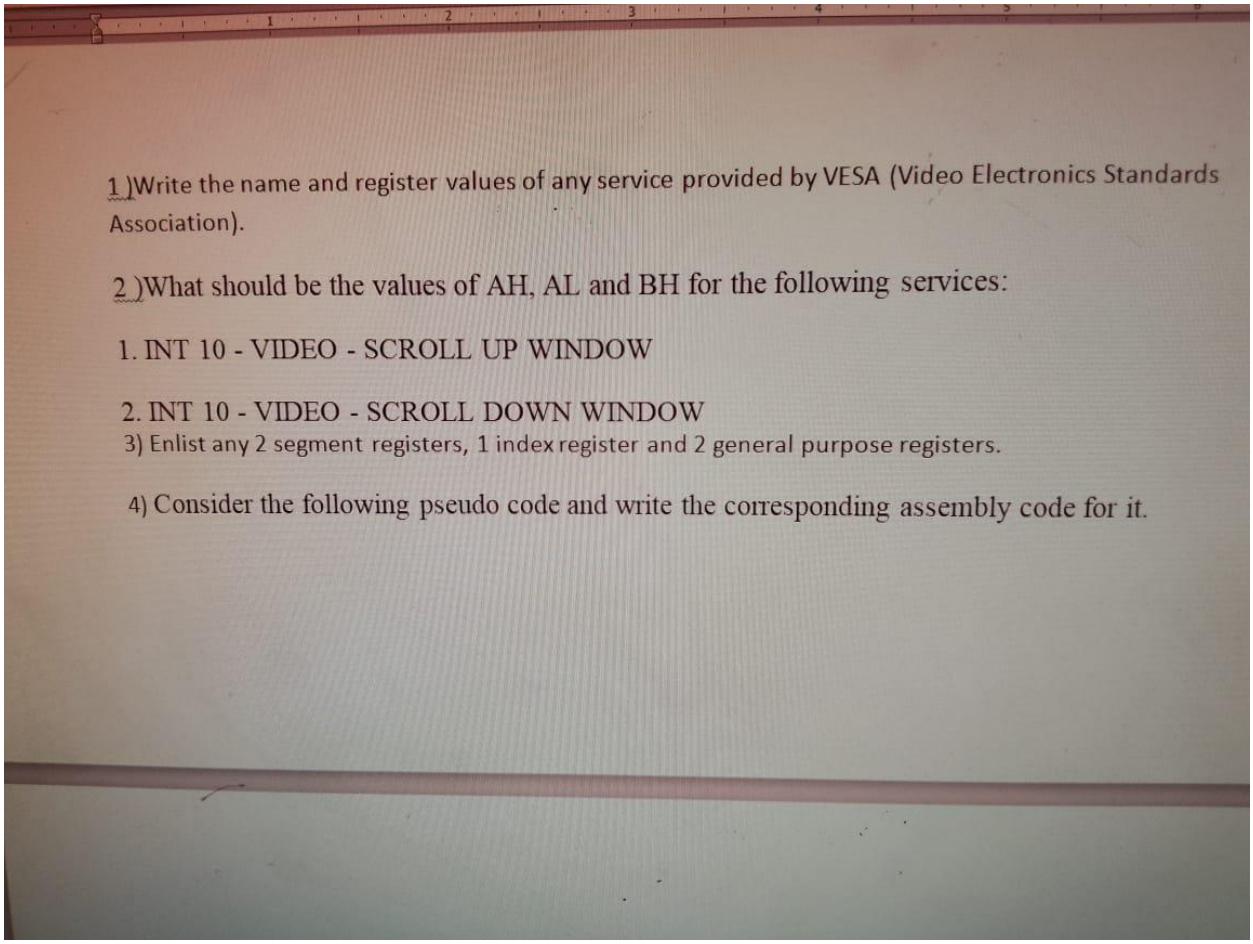
Solved By Hina Shahzadi



Solved By Hina Shahzadi



Solved By Hina Shahzadi



1)Write the name and register values of any service provided by VESA (Video Electronics Standards Association).

2)What should be the values of AH, AL and BH for the following services:

1. INT 10 - VIDEO - SCROLL UP WINDOW

2. INT 10 - VIDEO - SCROLL DOWN WINDOW

3) Enlist any 2 segment registers, 1 index register and 2 general purpose registers.

4) Consider the following pseudo code and write the corresponding assembly code for it.

Q :1

```
INT 10 - VESA - Get  SuperVGA Infromation
AX = 4F00h
ES:DI -> buffer for SuperVGA information
Return:
AL = 4Fh if function supported
AH = status
INT 10 - VESA - Get SuperVGA Mode Information
AX = 4F01h
CX = SuperVGA video mode
ES:DI -> 256-byte buffer for mode information
Return:
AL = 4Fh if function supported
AH = status
ES:DI filled if no error
INT 10 - VESA - Set VESA Video Mode
AX = 4F02h
```

Q 2:

Solved By Hina Shahzadi

INT 10 - VIDEO - SCROLL UP WINDOW
AH = 06h
AL = number of lines by which to scroll up (00h = clear entire window)
BH = attribute used to write blank lines at bottom of window
CH, CL = row, column of window's upper left corner
DH, DL = row, column of window's lower right corner
INT 10 - VIDEO - SCROLL DOWN WINDOW
AH = 07h
AL = number of lines by which to scroll down (00h=clear entire window)
BH = attribute used to write blank lines at top of window
CH, CL = row, column of window's upper left corner
DH, DL = row, column of window's lower right corner

Q 3:

Solved By Hina Shahzadi

Segment Registers:

1. CS (Code Segment)
2. DS (Data Segment)

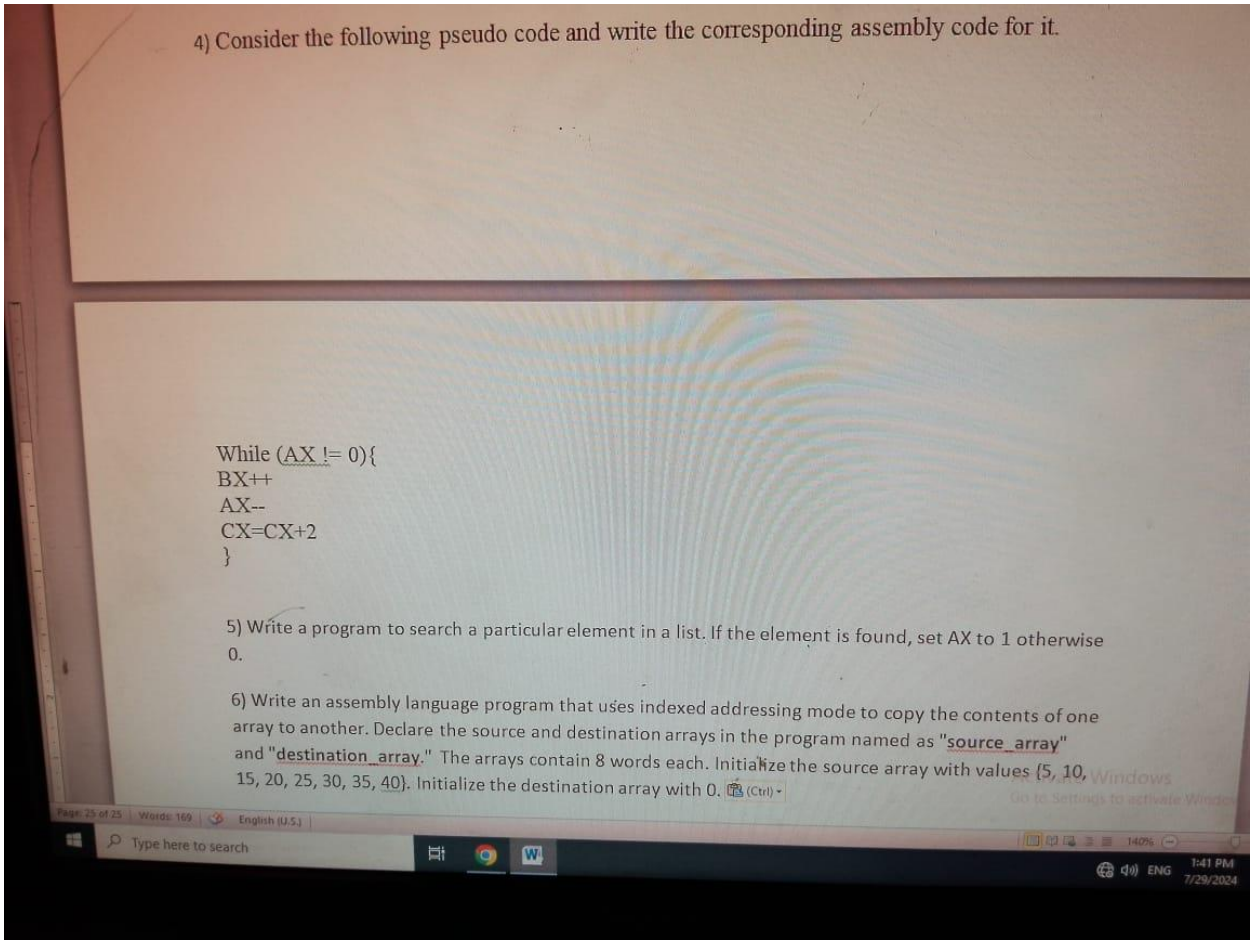
Index Register:

1. SI (Source Index)

General Purpose Registers:

1. AX (Accumulator Register)
2. BX (Base Register)

Solved By Hina Shahzadi



Q 4:

Assume AX, BX, and CX are already initialized

```
start_loop:
    cmp AX, 0        ; Compare AX with 0
    je end_loop      ; If AX is zero, jump to end_loop

    inc BX           ; Increment BX
    dec AX           ; Decrement AX
    add CX, 2        ; Add 2 to CX
```

Solved By Hina Shahzadi

```
    jmp start_loop    ; Jump back to start_loop

end_loop:
    ; Loop ends here
```

Q 5:

```
section .data
list db 10, 20, 30, 40, 50  ; Define the list of elements
list_size equ 5             ; Define the size of the list
element_to_find db 30       ; Define the element to search for

section .bss
found resb 1                ; Define a variable to store the found flag

section .text
global _start

_start:
    mov cx, list_size       ; Set CX to the size of the list
    lea si, [list]          ; Load the address of the list into SI
    mov al, [element_to_find] ; Load the element to find into AL
    xor ax, ax              ; Clear AX, initially assume not found
    xor bx, bx              ; Clear BX, this will be our index

search_loop:
    cmp bx, cx              ; Compare index with list size
    je not_found
```

Q 6:

Solved By Hina Shahzadi

```
section .data
source_array dw 5, 10, 15, 20, 25, 30, 35, 40  ; Define and initialize source array
destination_array dw 0, 0, 0, 0, 0, 0, 0, 0   ; Define and initialize destination array

section .bss
; No uninitialized data required for this program

section .text
global _start

_start:
   mov si, 0              ; Initialize SI to 0 (source index)
   mov di, 0              ; Initialize DI to 0 (destination index)
   mov cx, 8              ; Set CX to 8 (number of elements)

copy_loop:
   mov ax, [source_array + si]  ; Load word from source array into AX
   mov [destination_array + di], ax  ; Store word from AX to destination array

   add si, 2              ; Increment source index by 2 (word size)
   add di, 2              ; Increment destination index by 2 (word size)
   loop copy_loop         ; Decrement CX and repeat until CX is 0

end_program:
   ; Exit the program (for illustration, typically you'd do this via a system call)
   mov ah, 0x4c           ; DOS interrupt for program exit
   int 0x21               ; Call DOS interrupt
```