**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: December 3, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

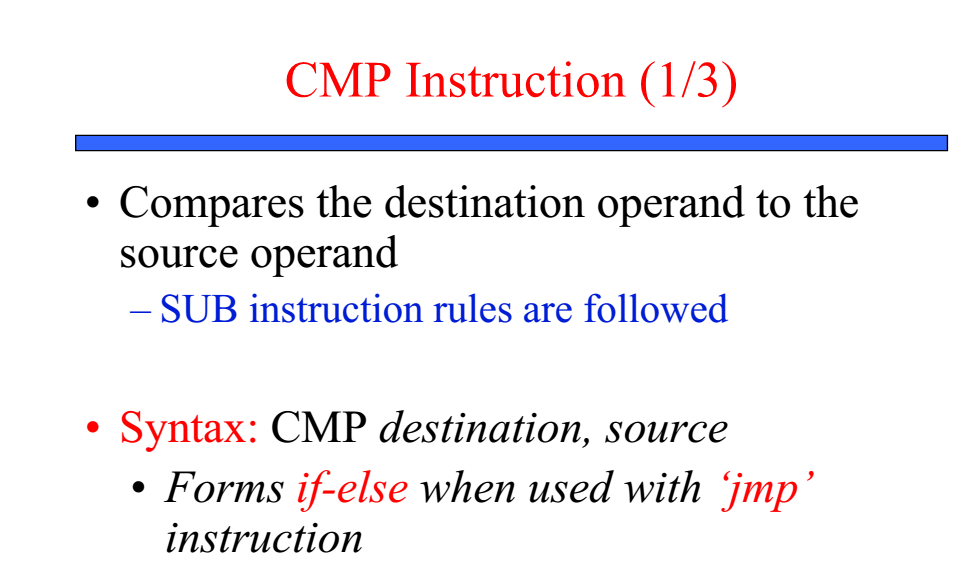
**Computer Organization and**

**Assembly Language (CS235)**

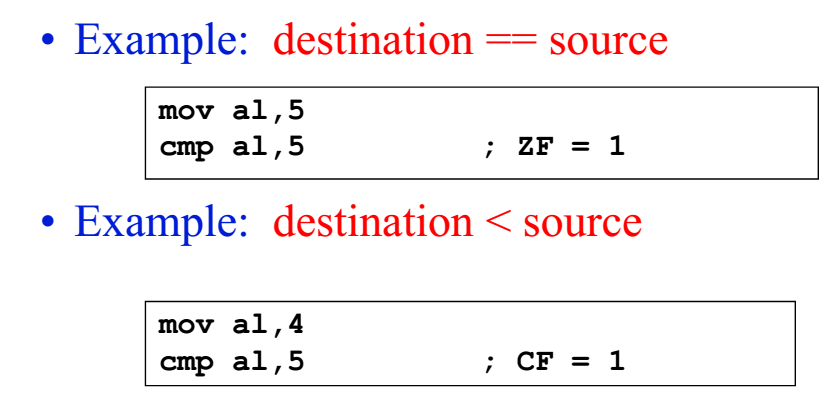
**Lab # 8 Conditional Structure in Assembly Language**

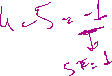
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team-work**  **5 marks** | **Total**  **25 marks** |
| **Fatima Seemab** | **291310** |  |  |  |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |  |  |  |

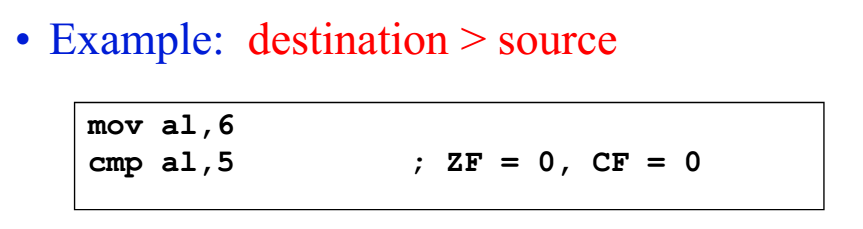
**Objective:** The aim of this lab is to use conditional jumps to implement conditional structures in Assembly language.









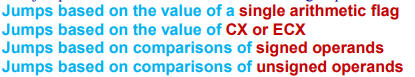




**USE THE CONDITIONAL JUMP**

****

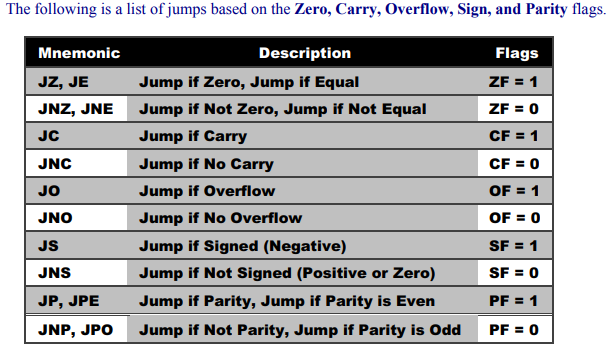
**CONDITIONAL JUMPS CAN BE DIVIDED IN TO FOUR GROUPS**

****



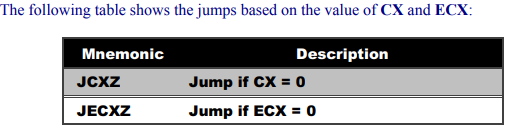
**Jumps Based On Single Arithmetic Flags**

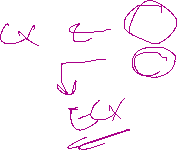


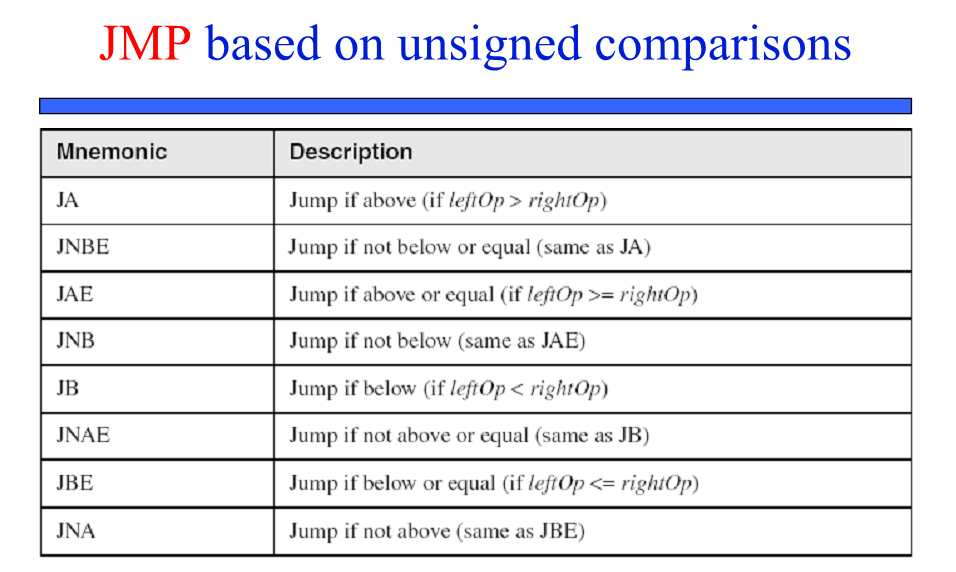
****

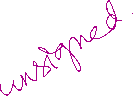


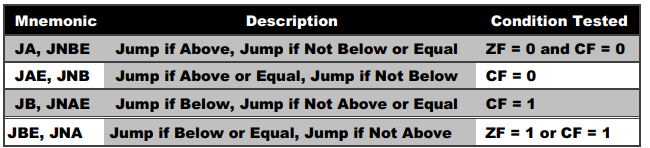
**Jumps Based On ECX**

****



****



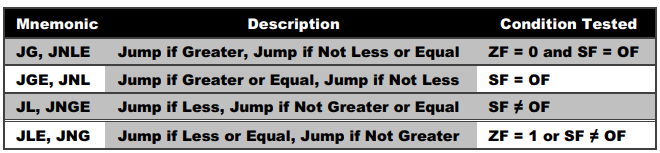
****



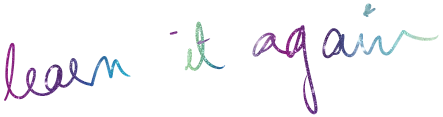
****

**Jumps Based On Comparison of Signed Operand**

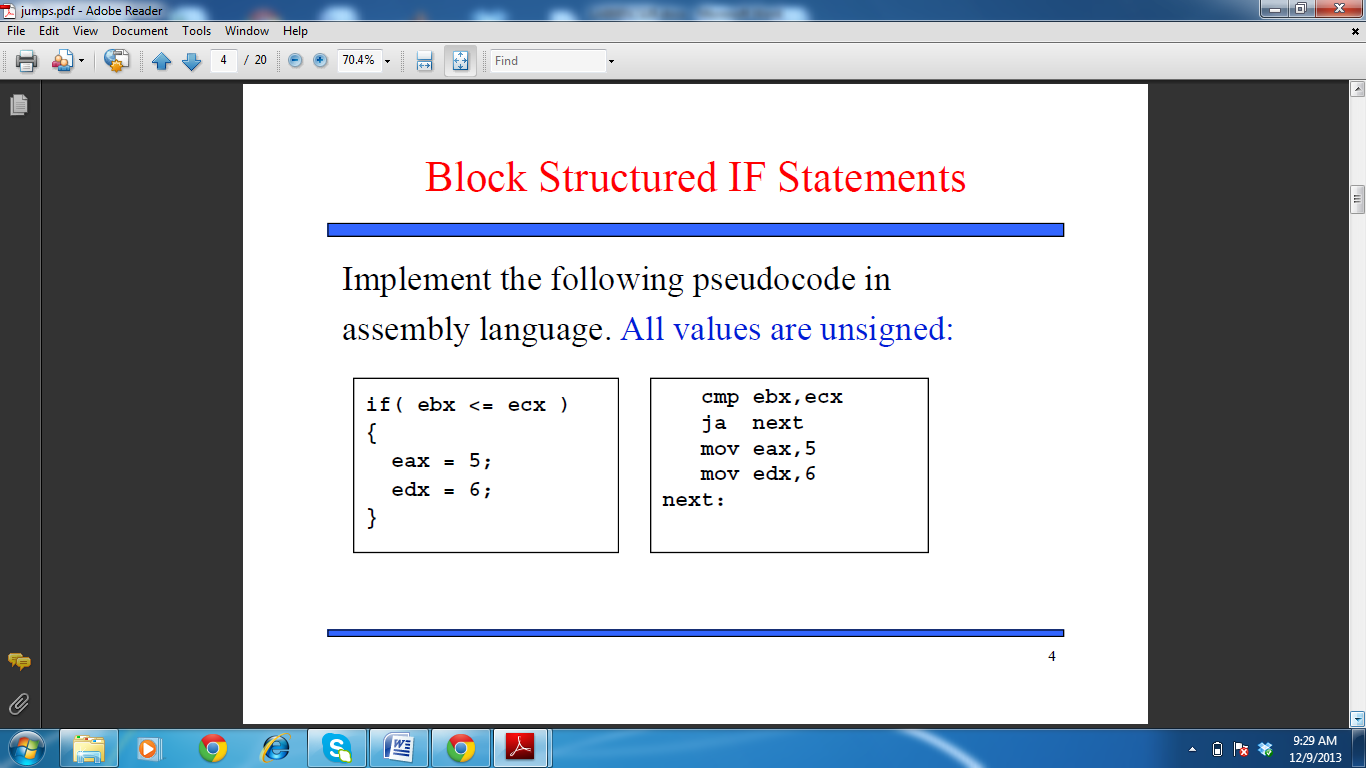


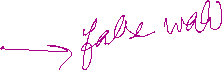
****

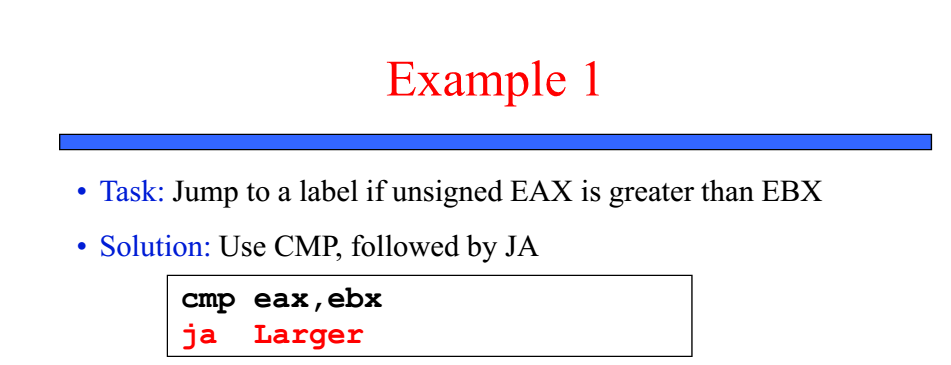
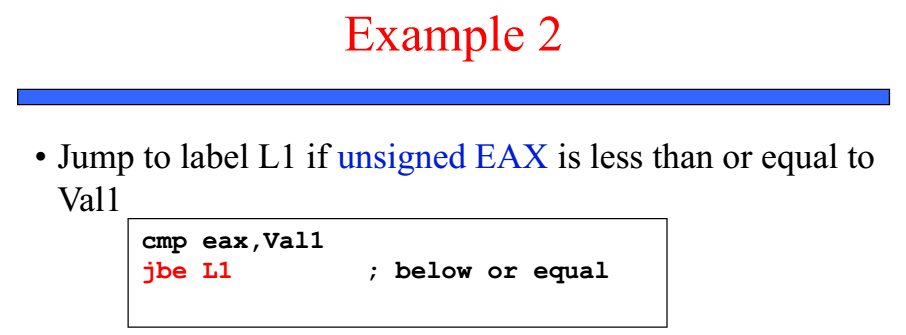


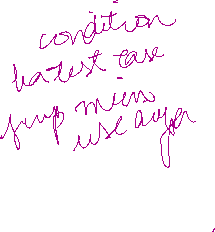
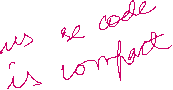
****

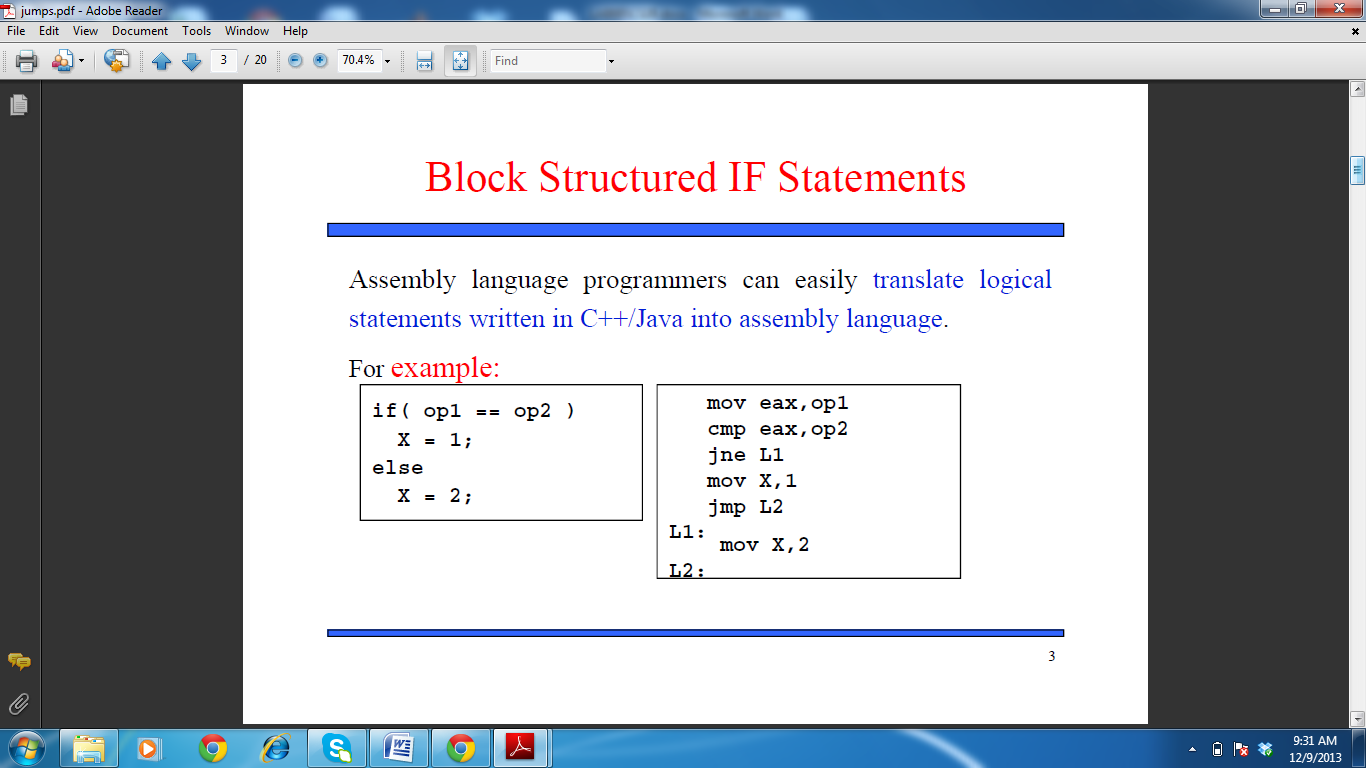


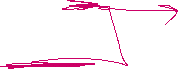
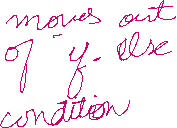
****







****



**Sample code:**

1. We can also use another jump label after jump body which points to the “exit” / end program



;(example)

INCLUDE Irvine32.inc

.data

.code

main proc

mov eax,6

mov ebx, 5

cmp eax, ebx



jb **L1** ; jump to L1 if eax is less than ebx otherwise move to next line



Jmp **progmend**; jump to label progmend; If this jmp is not used then L1 will execute in all conditions



**L1:**



mov eax,10

call dumpregs

Jmp progmend



**progmend:**

exit

main endp

END main

1. OR It totally depends on the program/task requirement that how you use the jmp labels for executing the program.

**LAB TASK 1:**

Compare unsigned AX to BX and copy the smaller of the two into a variable named smaller. Take integer input from the user. Display the variable ‘smaller’ content.

**Hint:** Use call dumpmem, to display variable.



Mov esi, offset var

Mov ecx,lengthof var

Mov ebx, type var

call dumpmem

**Code:**

TITLE smaller unsigned variable

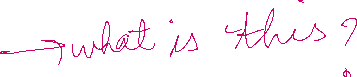
INCLUDE irvine32.inc

.data

smaller WORD ?

sInput1 BYTE "Enter first number: ",0

sInput2 BYTE "Enter second number: ",0



sOutput BYTE "smaller number is ",13,10,0



.code



main proc

mov edx,offset sInput1

call writeString ; prompt user

call readInt ; read integer 1



mov ebx,eax



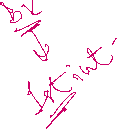
mov edx,offset sInput2

call writeString ; prompt user

call readInt ; read integer 2



 call small ; get smaller number

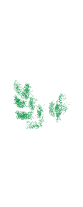
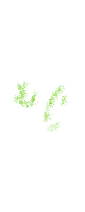
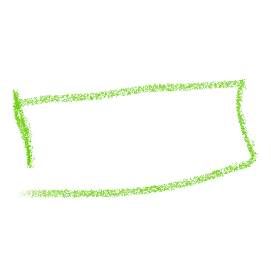


mov edx,offset sOutput

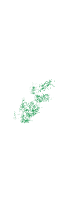
call writeString ; prompt user

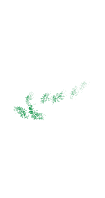
mov esi,offset smaller

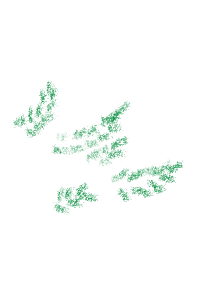
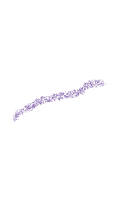
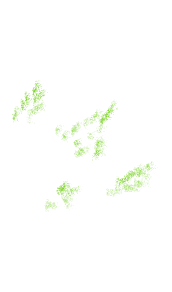
 mov ecx,lengthof smaller

 mov ebx,type smaller

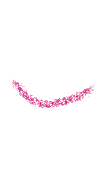
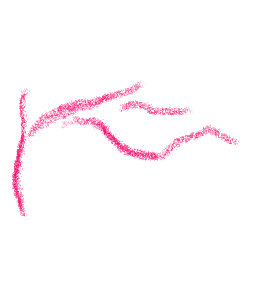
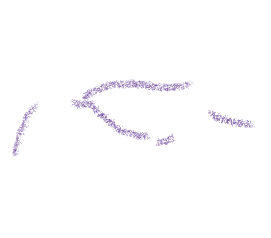
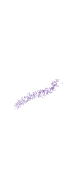
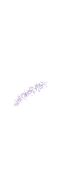
call dumpmem ; output



exit

main endp



small proc uses eax ebx

;-------------------------------------

; recieves two integers in eax and ebx

; returns smaller integer in var 'smaller'

;--------------------------------------

cmp ax,bx

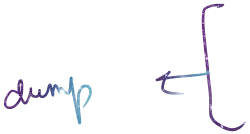


jna L



 mov smaller,bx

jmp endProc

L:

mov smaller,ax

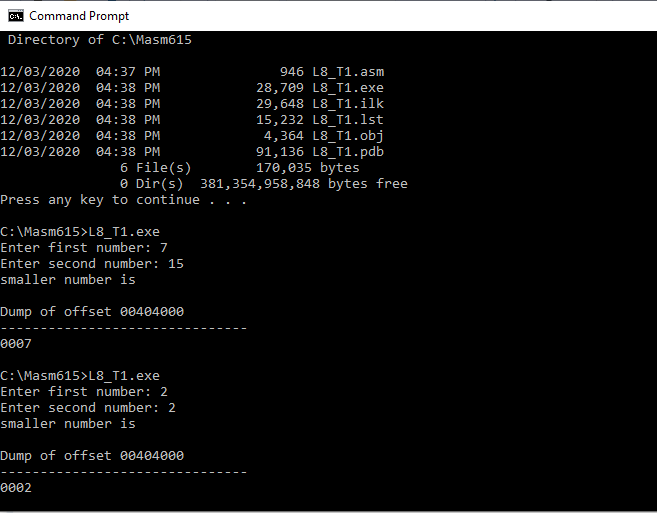
endProc:

ret

small endp

end main

**Output:**





**LAB TASK 2:**

Using the following table as a guide, write an assembly program that asks the user to enter an integer test score between 0 and 100. The program should display the appropriate letter grade:

|  |  |
| --- | --- |
| **Score Range** | **Letter Grade** |
| 90 to 100 | A |
| 80 to 89 | B |
| 70 to 79 | C |
| 60 to 69 | D |
| 0 to 59 | F |

Add the following features to the above program;

1- Use jmp/loop in such a way that multiple test scores can be entered. i.e., program is continuing to ask input (test marks) from the user.



2- When you press the ‘-1’, program should end.

3- Perform range checking on the user's input: display an error message if the test score is less than 0 or greater than 100.



You will display result such that you will give inputs to show all grades (A to F), give out of range input to print error message, give input -1, to exit from program



**Code:**

TITLE grade generator

INCLUDE irvine32.inc

.data

sInput BYTE "Enter score (-1 to exit): ",0

sOutput BYTE "Grade: ",0

sInvalid BYTE "Invalid score.",13,10,0

.code



main proc

Ask:



; prompt for score



mov edx,offset sInput



call writeString

; read score

 call readInt



cmp al,-1 ; if zf = 1



jz quit ; end proc



; calculate grade

call grade

jnz Ask

quit:

exit

main endp

grade proc uses eax



;---------------------------------

; requires score in eax



; returns nothing

;---------------------------------

call validScore ; check validity

jz endProc ; if invalid

cmp al, 59



 ja gradeD

 mov al, 'F'

 jmp quit



gradeD:

cmp al, 69

ja gradeC

mov al, 'D'



jmp quit

gradeC:

cmp al, 79

ja gradeB

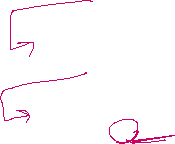
mov al, 'C'

jmp quit

gradeB:

cmp al, 89

ja gradeA



mov al, 'B'

jmp quit

gradeA:

mov al, 'A'

quit:

mov edx,offset sOutput

call writeString



call writechar ; output grade

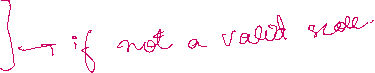


call crlf

endProc:

 OR al,1 ; reset ZF

call crlf



ret

grade endp

validScore proc uses eax



;---------------------------------

; requires score in eax

; returns set zf = 1 if invalid



;---------------------------------

cmp al, 0

jb Invalid ; if < 0

 cmp al,100



jbe quit

Invalid: ; if > 100

mov edx,offset sInvalid

call writeString

 test al,0 ; set ZF

jmp endProc

quit:

; in case score is 0 or 100

 OR al,1 ; reset ZF

endProc:

ret

validScore endp

end main

**Output:**

