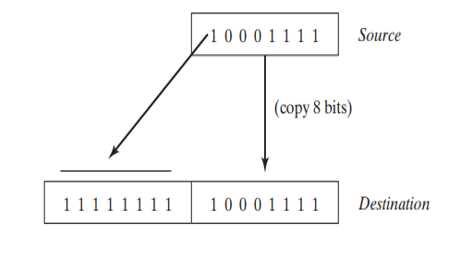
**Section A**

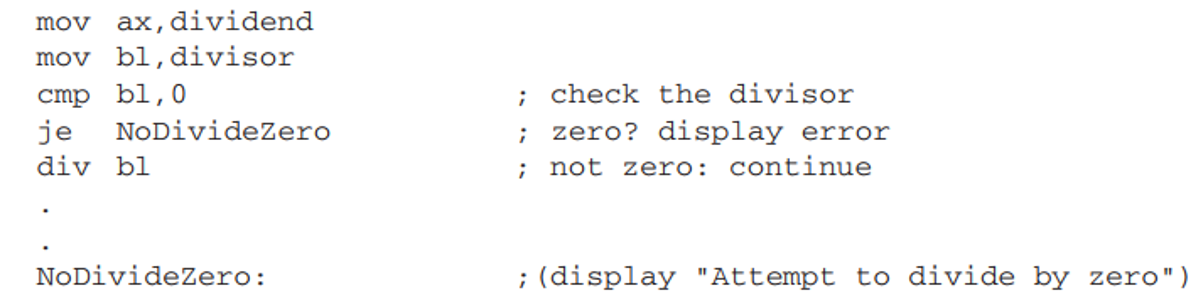
Answer the following **Ten** short questions 10×1=10

* 1. Which instruction has the following effect?



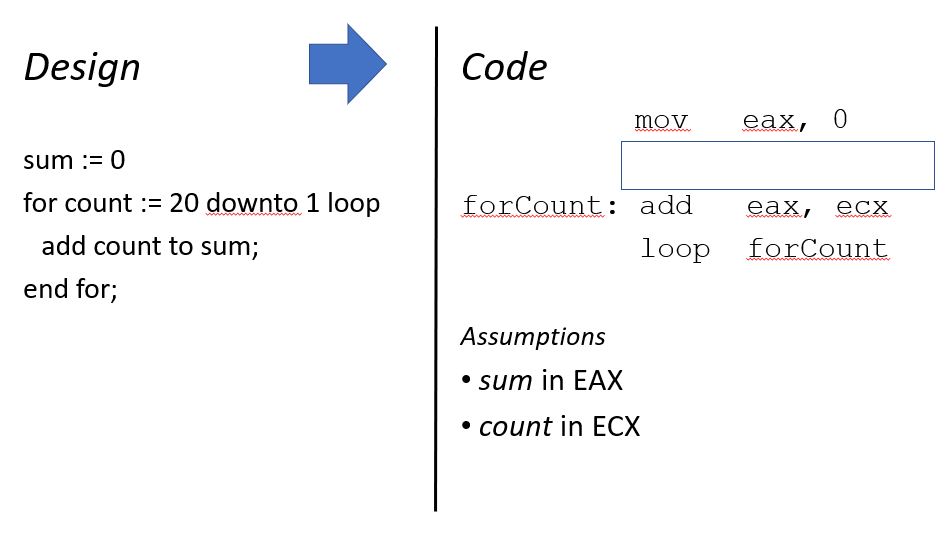
movsx

* 1. Write down the missing statement to prevent “**division by zero**”:



Jz NoDivideZero

* 1. Write down the missing statement of the code section that is marked by a box.

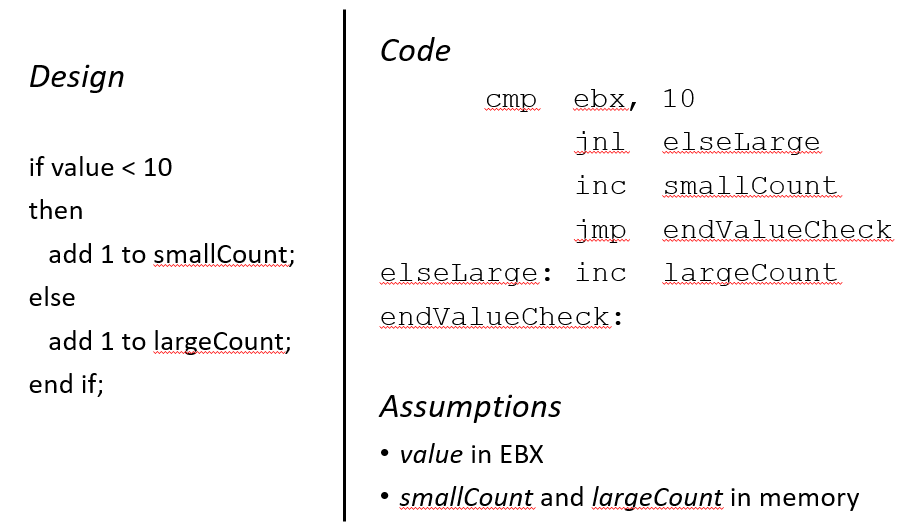


Mov ecx, 20

* 1. Write down a statement that creates an **array** of **five doublewords without initializing** any value.

Array DWORD 5 DUP (?)

* 1. Write down the missing labels in the following two boxes.



endValueCheck

elseLarge

* 1. What value should be multipied with esi?

nbrArr WORD 50 DUP (?)

...

mov eax, 0 ; sum := 0

mov ecx, 50 ; number of elements

mov esi, 0 ; array index

**addElt: add eax, nbrArr[ \*esi] ;**  **TO DO add element**

inc esi ; increment array index

loop addElt ; repeat

2

* 1. Write a statement to calculate ***value* mod 128** where *value* is an unsigned number in AX.

Hints: and ax, 000Fh ;for *value* mod 16

* 1. Write down **a statement** that will guard the code against 4,294,967,296 iterations.

mov eax, 0

mov ecx, 0

forCount: add eax, ecx

jecxz avoid ; **TO DO**

loop forCount

avoid:

* 1. What is the value of ZF and CF after executing all of the following statements?

mov ax, 203

xor ax, 203

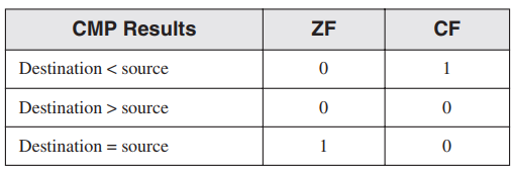
mov bx, 100

cmp ax, bx

|  |  |
| --- | --- |
| ZF | CF |
| 0 | 1 |

; **TO DO**

Hints:



* 1. Write down the missing statement.

myArray BYTE 50 DUP (?)

...

mov eax, 0 ; sum := 0

lea esi , myArray ; **TO DO** (load array address)

mov ecx, 50 ; number of elements

addLabel: add eax, [esi] ; add array element

inc esi ; point at next element

loop addLabel ; repeat

**Section B**

Write down assembly programs to solve the following problems 5×2=10

* 1. Write down a program to find the greatest common divisor (gcd). You will only need to implement the “whileRemainderNotZero” label in the following code.

.DATA ; reserve storage for data

prompt1 BYTE “Enter the first number: “, 0

prompt2 BYTE “Enter the second number: “, 0

dividend WORD ?

remainder WORD ?

gcd WORD ?

result BYTE “The greatest common divisor is “, 0

.CODE ; start of main program code

\_start:

; obtain two numbers greater than zero

whileBadInput1:

inputW prompt1, gcd

cmp gcd, 0

jle whileBadInput1

whileBadInput2:

inputW prompt2, remainder

cmp remainder, 0

jle whileBadInput2

whileRemainderNotZero: **; To Do Section**

**;Formula**

; gcd := num1

; remainder := num2

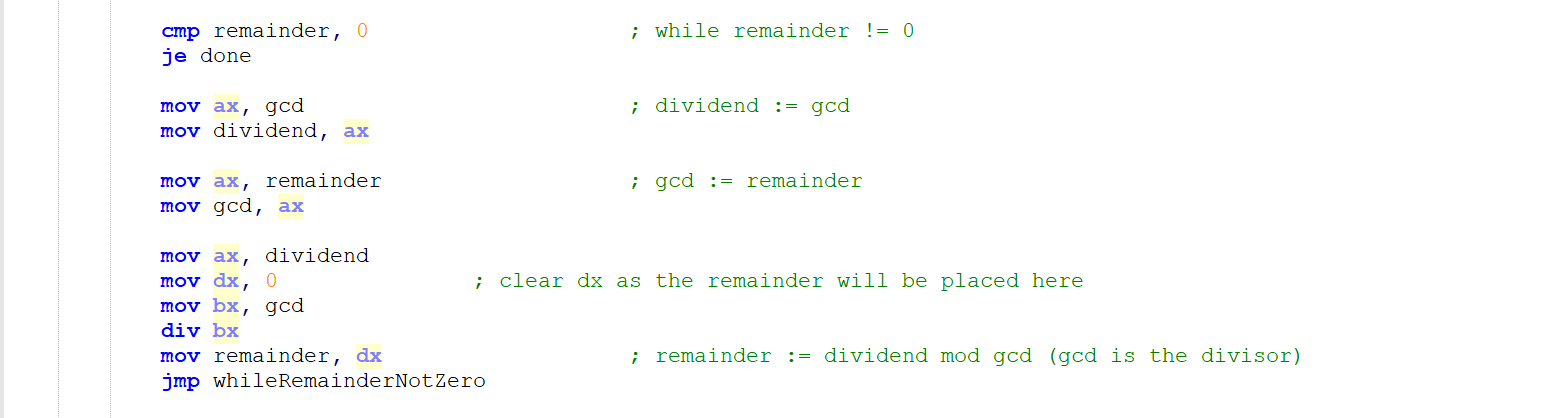
; while (remainder != 0)

; dividend := gcd

; gcd := remainder

; remainder := dividend mod gcd

; end while



done:

output result

output gcd

output carriage

INVOKE ExitProcess, 0 ; exit with return code 0

PUBLIC \_start ; make entry point public

END ; end of source code

* 1. Write the missing assembly statements to evaluate the expression – (x + y – 2\*z + 1).

.DATA

x DWORD 35

y DWORD 47

z DWORD 26

.CODE

main PROC

mov eax, x ; result := x

; result := x + y **TO DO**

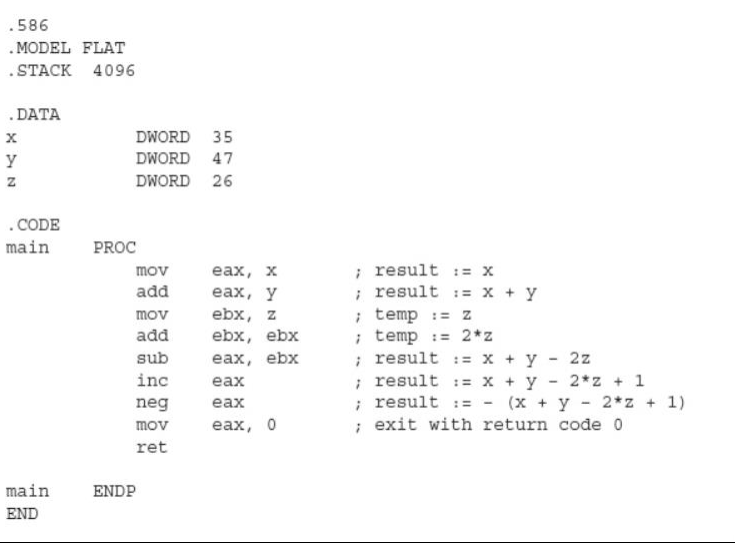
mov ebx, z ; temp := z

; temp := 2\*z **TO DO**

; result := x + y – 2\*z **TO DO**

; result := x + y – 2\*z + 1 **TO DO**

; result := - (x + y – 2\*z + 1) **TO DO**



mov eax, 0 ; exit with return code 0

ret

main ENDP

END

**Appendix: Useful Instructions**

|  |  |
| --- | --- |
| **Instruction** | **Description** |
| movsx | Move with sign extension |
| movzx | Move with zero extension |
| CBW | Convert Byte to Word |
| je | Jump if equal |
| neg | negates a value |
| jecxz | jump if ECX is zero |
| LEA | Load Effective Address |