Question2. Translate the following code into an assembly language program. Assume that X, Y, Z and i are 32-bit unsigned integers variables and X=1, Y=7, Z=8, and i=0.  Also assume that val2, val3 and val4 are 32-bit signed integer variables and val2=3, val3=2, val4=11. **(use only your .data and .code directives).**

 while ( i < 3){

       if (X <= Y) OR (X < Z){

              val1 = (val4/val2) - val3

              X = X + 2

      }

     else{

            val1 = (val2 \* val3) + val4

            X = X - 1

     }

 i = i + 1

}

Answer: One of the many possible solution.

.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode: DWORD

.data

X DWORD 1

Y DWORD 7

Z DWORD 8

i DWORD 0

val2 DWORD 3

val3 DWORD 2

val4 DWORD 11

.code

main PROC

; while loop condition (i < 3)

beginwhile:

cmp i, 3 ; if (i < 3) is false then i >= 3

jae exitwhile ; jump when the while condition is false

; whileblock

; evaluation (X<=Y)

mov ebx, X ; store one variable in a register

cmp ebx, Y ; if (X<=Y) then don't need to evaluate (X<Z). You can execute the if block

jbe ifblock ; if (X<=Y) go to ifblock

cmp ebx, Z ; otherwise evaluate the (X<Z)

jae elseblock ; if (X<Z) is false, skip the if block

ifblock:

mov eax, val4

mov edx, 0

div val2

sub eax, val3

add X, 2

jmp exit\_ifelse

elseblock:

mov eax, val2

mov edx, 0

mul val3

add eax, val4

sub X, 1

exit\_ifelse:

inc i

jmp beginwhile

exitwhile:

invoke ExitProcess, 0

main ENDP

END main