Sample Code for Q1:

Q1 Option 1:(Considering signed numbers)

```
INCLUDE Irvine32.inc
.data
var1 sword ?
var2 sword ?
var DWORD 5
x word ?
str1 byte        "Enter a number in the range from -32678 to +32677",0Dh, 0Ah,0
str2 byte      "Enter 2nd number in the range from -32678 to +32677",0Dh, 0Ah,0
.code
main PROC
mov edx, OFFSET str1
call WriteString  ; display str1
call ReadInt      ; Input first number from user
mov var1, ax      ; the result of readstring is moved from eax to var1
mov edx, OFFSET str2
call WriteString  ; display str2
call ReadInt      ; Input 2nd number from user
mov var2,ax       ; the result of readstring is moved from eax to var2
movsx ecx, var1   ; Move the first number to ecx
movsx edx, var2   ; Move the 2nd number to edx
mov x,0           ; we have initialied x with 0
cmp var, ecx
JNL VALUE         ; if var not less than ecx then jump to label VALUE
cmp ecx, edx
JNGE VALUE        ; if ecx not greater than or equal to edx then jump to label VALUE
JMP QUIT          ; this line will execute if both conditions are true
VALUE:
mov x,1
QUIT:
mov esi,OFFSET x
mov ecx, LENGTHOF x
mov ebx, TYPE x
call DumpMem
movzx edx, x
call DumpRegs
exit
main ENDP
END main
```

Q1 Option 2: (Considering signed numbers)

```
INCLUDE Irvine32.inc
.data
var1 sword ?
var2 sword ?
var DWORD 5
x word ?
str1 byte "Enter a number in the range from -32678 to +32677",0Dh, 0Ah,0
str2 byte "Enter 2nd number in the range from -32678 to +32677",0Dh, 0Ah,0
.code
main PROC
mov edx, OFFSET str1
call WriteString ; display str1
```

```
call ReadInt     ; Input first number from user
mov var1, ax     ; the result of readstring is moved from eax to var1
mov edx, OFFSET str2
call WriteString ; display str2
call ReadInt     ; Input 2nd number from user
mov var2,ax     ;the result of readstring is moved from eax to var2
movsx ecx, var1 ; Move the first number to ecx
movsx edx, var2 ; Move the 2nd number to edx
cmp var, ecx
JNL VALUE       ; if var not less than ecx then jump to label VALUE
cmp ecx, edx
JNGE VALUE      ; if ecx not greater than or equal to edx then jump to label VALUE
mov x,0         ; this line will execute if both conditions are true
JMP QUIT      ; this line makes sure that you skip the label value if conditions are true
VALUE:
mov x,1
QUIT:
mov esi,OFFSET x
mov ecx, LENGTHOF x
mov ebx, TYPE x
call DumpMem
movzx edx, x
call DumpRegs
exit
main ENDP

END main
```

Q1 Option 3: (Considering unsigned numbers)

```
INCLUDE Irvine32.inc
.data
var1 byte ?
var2 byte ?
var DWORD 5
x byte ?
str1 byte "Enter a number in the range from 0 to 255",0Dh, 0Ah,0
str2 byte "Enter 2nd number in the range from 0 to 255",0Dh, 0Ah,0
.code
main PROC
mov edx, OFFSET str1
call WriteString   ; display str1
call ReadDec       ; Input first number from user
mov var1, al       ; the result of readstring is moved from eax to var1
mov edx, OFFSET str2
call WriteString   ; display str2
call ReadDec       ; Input 2nd number from user
mov var2,al        ;the result of readstring is moved from eax to var2
movzx ecx, var1    ; Move the first number to ecx
movzx edx, var2    ; Move the 2nd number to edx
cmp var, ecx
JNB VALUE          ; if var not less than ecx then jump to label VALUE
cmp ecx, edx
JNAE VALUE         ;if ecx not greater than or equal to edx then jump to label VALUE
mov x,0            ; this line will execute if both conditions are true
JMP QUIT     ; this line makes sure that you skip the label value if conditions are true
VALUE:
mov x,1
```

```
QUIT:
mov esi,OFFSET x
mov ecx, LENGTHOF x
mov ebx, TYPE x
call DumpMem
movzx edx, x
call DumpRegs
exit
main ENDP

END main
```

Q1 Option 4: (Considering unsigned numbers)

```
INCLUDE Irvine32.inc
.data
var1 byte ?
var2 byte ?
var DWORD 5
x byte ?
str1 byte "Enter a number in the range from 0 to 255",0Dh, 0Ah,0
str2 byte "Enter 2nd number in the range from 0 to 255",0Dh, 0Ah,0
.code
main PROC
mov edx, OFFSET str1
call WriteString   ; display str1
call ReadDec       ; Input first number from user
mov var1, al       ; the result of readstring is moved from eax to var1
mov edx, OFFSET str2
call WriteString   ; display str2
call ReadDec       ; Input 2nd number from user
mov var2,al        ;the result of readstring is moved from eax to var2
movzx ecx, var1    ; Move the first number to ecx
movzx edx, var2    ; Move the 2nd number to edx
mov x,0            ; x initialized to 0
cmp var, ecx
JNB VALUE          ; if var not less than ecx then jump to label VALUE
cmp ecx, edx
JNAE VALUE         ;if ecx not greater than or equal to edx then jump to label VALUE
JMP QUIT           ;this line will execute if conditions are true and skip label VALUE
VALUE:
mov x,1
QUIT:
mov esi,OFFSET x
mov ecx, LENGTHOF x
mov ebx, TYPE x
call DumpMem
movzx edx, x ; value moved to edx for display
call DumpRegs
exit
main ENDP

END main
```

Question 2 option 1:

Important points about this code :

In this code we have used indirect operands, the registers used for indexed and indirect addressing should be 32-bit,

```
; Scan an array for the first nonzero value.
INCLUDE Irvine32.inc
.data
intArray SWORD 0,0,0,0,1,20,35,-12,66,4,0
noneMsg BYTE "A non-zero value was not found",0
.code
main PROC
 mov ebx,OFFSET intArray     ; point to the array
 mov ecx,LENGTHOF intArray   ; loop counter
L1:
cmp  WORD ptr [ebx],0        ; compare value to zero, don't forget to use ptr operator as
;we have used indirect operand and comparing it with a constant so you need use ptr
 jnz found                   ; found a value
 add ebx,2                   ; point to next
 loop L1                     ; continue the loop
jmp notFound                 ; none found
found:                       ; display the value
 movsx eax, word PTR [ebx] ; sign-extend into EAX. In this case eax is 32-bit whereas the
;value pointed by residter is of 16-bit so use ptr
 call WriteInt               ; Will display the output we saved in eax
 jmp quit
notFound:                    ; display "not found" message
 mov edx,OFFSET noneMsg
 call WriteString
quit:
 call Crlf
 exit
main ENDP
END main
```

Question 2 option 2:

In this code we have used indexed operands with scaled factors, the registers used for indexed and indirect addressing should be 32-bit,

```
; Scan an array for the first nonzero value.
INCLUDE Irvine32.inc
.data
intArray SWORD 0,0,0,0,1,20,35,-12,66,4,0
noneMsg BYTE "A non-zero value was not found",0
.code
main PROC
 mov esi,0
 mov ecx,LENGTHOF intArray           ; loop counter
L1:
cmp  intArray[esi*type IntArray], 0  ; compare value to zero
 jnz found                           ; found a value
 inc esi                             ; point to next
 loop L1                             ; continue the loop
jmp notFound                         ; none found
found:                               ; display the value
 movsx eax, intArray[esi*type intArray] ; sign-extend into EAX
 call WriteInt
 jmp quit
```

```
notFound:                                   ; display "not found" message
 mov edx,OFFSET noneMsg
 call WriteString
quit:
 call Crlf
 exit
main ENDP
END main
```