Lab: 12

Department of Computer Science

Iqra University Islamabad

Computer Organization and Assembly Language

Maqsood Ahmed

ID: 38186

5.4.3 Lab Work: Invalid Input

What happens when entering a string longer than 20 characters?
When you enter a string longer than 20 characters, the program may behave unpredictably or crash if it does not handle buffer overflow properly. This is because the buffer allocated for the string might not be large enough to accommodate the excess characters, leading to overwriting of adjacent memory.

What happens when entering an invalid hexadecimal number?
Entering an invalid hexadecimal number (one with characters outside 0-9 and A-F) typically causes the program to either reject the input, display an error, or interpret it incorrectly. If the program does not have proper validation, it might cause undefined behavior.

What happens when entering an invalid decimal number?
Entering an invalid decimal number (characters outside 0-9) usually results in an error message or rejection of the input. If the program lacks proper validation, it might crash or behave unpredictably.

5.5 Generating Random Numbers and Delaying Program Execution

Changes to Generate the Same Random Sequence Every Time
To generate the same random sequence every time the program is executed, you need to seed the random number generator with a fixed value instead of using the current time. This involves modifying the `Randomize` procedure to use a constant seed value.

Review Questions

1. Which procedure in the Irvine link library displays "Press [Enter] to continue …"?
- The procedure is `WaitMsg`.

2. Which procedure writes an integer in unsigned decimal format to standard output?
- The procedure is `WriteDec`.

3. Which procedure generates a random integer within a selected range?
- The procedure is `RandomRange`.

4. Which procedure places the cursor at a specific console window location?
- The procedure is `Gotoxy`.

5. What are the required input parameters for the `ReadString` Procedure?
- The parameters are the address of the buffer to store the string and the maximum number of characters to read.

6. Locate and examine the Irvine.inc file. What type of statements are inside this file?

- The `Irvine.inc` file contains declarations of procedures and macros, constants, and data structures used in the Irvine32 library.

PROGRAMMING EXERCISES

Exercise 1: Input and Redisplay Ten Signed 32-bit Integers

```
TITLE Input and Display Ten Integers (input_disp.asm)

INCLUDE Irvine32.inc

.data
array DWORD 10 DUP(0)
prompt BYTE "Enter a signed 32-bit integer: ", 0
msg BYTE "You entered: ", 0

.code
main PROC
    mov ecx, 10              ; loop counter for 10 integers
    mov esi, OFFSET array    ; point to start of array

input_loop:
    mov edx, OFFSET prompt
    call WriteString
    call ReadInt             ; read integer from user
    mov [esi], eax           ; store integer in array
    add esi, 4               ; move to next array element
    loop input_loop

    mov ecx, 10              ; loop counter for display
    mov esi, OFFSET array    ; point to start of array

display_loop:
    mov edx, OFFSET msg
    call WriteString
    mov eax, [esi]
    call WriteInt
    call Crlf
    add esi, 4               ; move to next array element
    loop display_loop

    exit
main ENDP
END main
```

Exercise 2: Display String in Four Colors

```
TITLE Display String in Four Colors (colors.asm)

INCLUDE Irvine32.inc

.data
str1 BYTE "Assembly", 0
str2 BYTE "Language", 0
str3 BYTE "is", 0
str4 BYTE "COOL", 0

.code
main PROC
    ; Display "Assembly" in color 1
    mov edx, OFFSET str1
    mov ecx, 1              ; color code
    call SetTextColor
    call WriteString
    call Crlf

    ; Display "Language" in color 2
    mov edx, OFFSET str2
    mov ecx, 2              ; color code
    call SetTextColor
    call WriteString
    call Crlf

    ; Display "is" in color 3
    mov edx, OFFSET str3
    mov ecx, 3              ; color code
    call SetTextColor
    call WriteString
    call Crlf

    ; Display "COOL" in color 4
    mov edx, OFFSET str4
    mov ecx, 4              ; color code
    call SetTextColor
    call WriteString
    call Crlf

    exit
main ENDP
END main
```

Exercise 3: Clear Screen, Input Two Integers, Display Sum

```
TITLE Clear Screen and Sum Two Integers (sum_ints.asm)

INCLUDE Irvine32.inc

.data
prompt1 BYTE "Enter first integer: ", 0
prompt2 BYTE "Enter second integer: ", 0
resultMsg BYTE "The sum is: ", 0
num1 DWORD ?
num2 DWORD ?
sum DWORD ?

.code
main PROC
    call Clrscr

    ; Input first integer
    mov edx, OFFSET prompt1
    call WriteString
    call ReadInt
    mov num1, eax

    ; Input second integer
    mov edx, OFFSET prompt2
    call WriteString
    call ReadInt
    mov num2, eax

    ; Calculate sum
    mov eax, num1
    add eax, num2
    mov sum, eax

    ; Display sum
    mov edx, OFFSET resultMsg
    call WriteString
    mov eax, sum
    call WriteInt

    exit
main ENDP
END main
```

Exercise 4: Generate and Display 50 Random Integers Between -20 and +20

```
TITLE 50 Random Integers Between -20 and 20 (random50.asm)

INCLUDE Irvine32.inc

.data
msg BYTE "Random integer: ", 0

.code
main PROC
    call Randomize          ; seed the random number generator

    mov ecx, 50             ; loop counter for 50 integers

random_loop:
    mov edx, OFFSET msg
    call WriteString

    mov eax, 41             ; set range (20 - (-20) + 1)
    call RandomRange        ; generate random number between 0 and 40
    sub eax, 20             ; shift to range -20 to 20
    call WriteInt
    call Crlf

    loop random_loop

    exit
main ENDP
END main
```

Exercise 5: Generate and Display Twenty Random Strings

```
TITLE 20 Random Strings (random_strings.asm)

INCLUDE Irvine32.inc

.data
string BYTE 11 DUP(0)
msg BYTE "Random string: ", 0

.code
main PROC
    call Randomize          ; seed the random number generator

    mov ecx, 20             ; loop counter for 20 strings

string_loop:
    mov edx, OFFSET msg
    call WriteString
```

```
    ; Generate 10 random characters
    mov esi, OFFSET string
    mov ebx, 10             ; 10 characters

char_loop:
    mov eax, 26             ; 26 letters in alphabet
    call RandomRange
    add eax, 'A'            ; convert to ASCII letter
    mov [esi], al
    inc esi
    dec ebx
    jnz char_loop

    mov byte ptr [esi], 0   ; null-terminate the string

    mov edx, OFFSET string
    call WriteString
    call Crlf

    loop string_loop

    exit
main ENDP
END main
```

Exercise 6: Display '*' at 100 Random Screen Locations

```
TITLE Display '*' at 100 Random Locations (random_star.asm)

INCLUDE Irvine32.inc

.data
star BYTE "*", 0

.code
main PROC
    call Randomize          ; seed the random number generator

    mov ecx, 100            ; loop counter for 100 locations

star_loop:
    mov eax, 80             ; screen width
    call RandomRange        ; random x coordinate
    mov ebx, eax

    mov eax, 25             ; screen height
    call RandomRange        ; random y coordinate
    mov ecx, eax
```

```
    mov eax, ecx
    shl eax, 8
    add eax, ebx
    mov cx, eax
    call Gotoxy

    mov edx, OFFSET star
    call WriteChar

    mov eax, 100            ; delay in milliseconds
    call Delay

    loop star_loop

    exit
main ENDP
END main
```