Laboratory Manual for
Computer Organization and Assembly Language

Department of Computer Science
National University of Computer and Engineering Sciences
Chiniot-Faisalabad Campus

Outline

- TYPE, LENGTHOF, SIZEOF Operators
- Input and Output Operations of Assembly Language
- Irvine Library Functions
- Working with character strings

1 Operators

1.1 TYPE Operator

The TYPE operator returns the size, in bytes, of a single element of a variable. For example, the TYPE of a byte equals 1, the TYPE of a word equals 2, the TYPE of a doubleword is 4, and the TYPE of a quadword is 8.

```
; Author: Abuzar Ghafari
; Program Name:   exp9
; Program Description: TYPE Operator example
; Date

INCLUDE Irvine32.inc

.data
   bvar byte ?       ; 1 byte variable
   wvar word ?       ; 2 byte variable
   dvar dword ?      ; 4 byte variable
   qvar qword ?      ; 8 byte variable
.code
main PROC

   mov eax, TYPE bvar      ; eax = 1, size of bvar in bytes
   mov ebx, TYPE wvar      ; ebx = 2, size of wvar in bytes
   mov ecx, TYPE dvar      ; ecx = 4, size of dvar in bytes
   mov edx, TYPE qvar      ; eax = 8, size of qvar in bytes

   call WaitMsg         ; Press any key to continue...

exit
main ENDP
END main
```

1.2 LENGTHOF Operator

The LENGTHOF operator counts the number of elements in an array, defined by the values appearing on the same line as its label.
When nested DUP operators are used in an array definition, LENGTHOF returns the product of the two counters.

```
; Author: Abuzar Ghafari
; Program Name: exp10
; Program Description: LENGTHOF Operator Examples
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   byte1 byte 10, 20, 30
   array1 word 30 dup (?), 0, 0
   array2 word 5 dup(3 dup(?))
   array3 dword 1, 2, 3, 4
   digitStr byte "123456789",0

   array4 byte 10, 20, 30, 40, 50
          byte 60, 70, 80, 90, 100

   array5 byte 10, 20, 30, 40, 50,
               60, 70, 80, 90, 100
.code
main PROC

   mov eax, LENGTHOF byte1       ; eax = 3
   mov eax, LENGTHOF array1      ; eax = 32
   mov eax, LENGTHOF array2      ; eax = 15
   mov eax, LENGTHOF array3      ; eax = 4
   mov eax, LENGTHOF digitStr    ; eax = 10
   mov eax, LENGTHOF array4      ; eax = 5
   mov eax, LENGTHOF array5      ; eax = 10

   call WaitMsg          ; Press any key to continue...

exit
main ENDP
END main
```

1.3 SIZEOF Operator

The SIZEOF operator returns a value that is equivalent to multiplying LENGTHOF by TYPE. In the following example, intArray has TYPE = 2 and LENGTHOF = 32. Therefore, SIZEOF intArray equals 64.

```
1  ; Author: Abuzar Ghafari
2  ; Program Name: exp11
3  ; Program Description: SIZEOF Operator Examples
4  ; Date: 10/18/2020
5
6  INCLUDE Irvine32.inc
7
8  .data
9     intArray word 32 dup (0)
10    var1 dword 01
11    var2 qword 02
12 .code
13 main PROC
14
15    mov eax, SIZEOF intArray      ; eax = 2 * 32 = 64
16    mov eax, SIZEOF var1        ; eax = 4 * 1 = 4
17    mov eax, SIZEOF var2        ; eax = 8 * 1 = 8
18
19    call WaitMsg          ; Press any key to continue...
20
21 exit
22 main ENDP
23 END main
```

2 Input and Output Operations

2.1 WriteDec Procedure

The **WriteDec** procedure writes a 32-bit unsigned integer to the console window in decimal format with no leading zeros. Pass the integer in EAX.

```
; Author: Abuzar Ghafari
; Program Name: exp1
; Program Description: WriteDec Procedure Examples
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   x byte 12
.code
main PROC
   mov eax, 295       ; eax = 295
   call WriteDec      ; call WriteDec procedure

   call crlf          ; call crlf procedure to endline

   mov eax, 0         ; eax = 0
   mov al, x          ; eax = 12
   call WriteDec      ; call WriteDec procedure

   CALL crlf          ; call crlf procedure to endline
   call WaitMsg       ; Press any key to continue...
exit
main ENDP
END main
```

2.2 WriteChar Procedure

The **WriteChar** procedure writes a single character to the console window. Pass the character (or its ASCII code) in AL.

```
; Author: Abuzar Ghafari
; Program Name: exp2
; Program Description: WriteChar Procedure Examples
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   a byte 'a'
.code
main PROC

   mov al, 'A'        ; al = 'A'
   call WriteChar     ; call WriteChar procedure
   call Crlf          ; call Crlf procedure to endline

   mov al, 62H        ; al = 62H, 62H = 'b'
   call WriteChar     ; call WriteChar procedure
   call Crlf          ; call Crlf procedure to endline

   mov al, a          ; al = a
   call WriteChar     ; call WriteChar procedure
   call Crlf          ; call Crlf procedure to endline

   call WaitMsg       ; Press any key to continue...

exit
main ENDP
END main
```

2.3 WriteString Procedure

The **WriteString** procedure writes a null-terminated string to the console window. Pass the string's offset in EDX.

```
; Author: Abuzar Ghafari
; Program Name: exp3
; Program Description: WriteString Procedure Example
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   prompt byte "Enter your name: ", 0
.code
main PROC

   mov edx, offset prompt      ; edx = offset of prompt string
   call WriteString          ; call WriteString procedure
   call Crlf                 ; call Crlf procedure to endline

   call WaitMsg              ; Press any key to continue...

exit
main ENDP
END main
```

2.4 ReadChar Procedure

The **ReadChar** procedure reads a single character from the keyboard and returns the character in the AL register. The character is not echoed in the console window.

```
; Author: Abuzar Ghafari
; Program Name: exp4
; Program Description: ReadChar Procedure Example
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   x byte ?
.code
main PROC

   call ReadChar      ; call ReadChar procedure
   mov x, al          ; move char input to x variable

   call WaitMsg       ; Press any key to continue...

exit
main ENDP
END main
```

2.5 ReadDec Procedure

The **ReadDec** procedure reads a 32-bit unsigned decimal integer from the keyboard and returns the value in EAX. Leading spaces are ignored. The return value is calculated from all valid digits found until a nondigit character is encountered. For example, if the user enters 123ABC, the value returned in EAX is 123.

```
1  ; Author: Abuzar Ghafari
2  ; Program Name: exp5
3  ; Program Description: ReadDec Procedure Example
4  ; Date: 10/18/2020
5
6  INCLUDE Irvine32.inc
7
8  .data
9     intVal dword ?
10 .code
11 main PROC
12
13    call ReadDec       ; call ReadDec procedure
14    mov intVal, eax       ; intVal = eax, eax contains the input value
15
16    call WaitMsg       ; Press any key to continue...
17
18 exit
19 main ENDP
20 END main
```

2.6 ReadInt Procedure

The **ReadInt** procedure reads a 32-bit signed integer from the keyboard and returns the value in EAX. The user can type an optional leading plus or minus sign, and the rest of the number may only consist of digits. ReadInt sets the Overflow flag and display an error message if the value entered cannot be represented as a 32-bit signed integer (range: -2,147,483,648 to +2,147,483,647). The return value is calculated from all valid digits found until a nondigit character is encountered. For example, if the user enters +123ABC, the value returned is +123.

```
1  ; Author: Abuzar Ghafari
2  ; Program Name: exp6
3  ; Program Description: ReadInt Procedure Example
4  ; Date: 10/18/2020
5
6  INCLUDE Irvine32.inc
7
8  .data
9     intVal sdword ?
10 .code
11 main PROC
12
13    call ReadInt       ; call ReadInt procedure
14    mov intVal, eax       ; intVal = eax, eax contains the input value
15
16    call WaitMsg       ; Press any key to continue...
17
18 exit
19 main ENDP
20 END main
```

2.7 ReadString Procedure

The **ReadString** procedure reads a string from the keyboard, stopping when the user presses the Enter key. Pass the offset of a buffer in EDX and set ECX to the maximum number of characters the user can enter, plus 1 (to save space for the terminating null byte). The procedure returns the count of the number of characters typed by the user in EAX.

```
1  ; Author: Abuzar Ghafari
2  ; Program Name: exp7
3  ; Program Description: ReadString Procedure Example
4  ; Date: 10/18/2020
5
6  INCLUDE Irvine32.inc
7
8  .data
9     buffer byte 21 dup (0)      ; input buffer
10    byteCount dword ?        ; holds counter
11 .code
12 main PROC
13
14    mov edx, offset buffer      ; point to the buffer
15    mov ecx, sizeof buffer      ; specify max characters
16    call ReadString             ; input the string
17    mov byteCount, eax          ; number of characters
18
19    call WaitMsg             ; Press any key to continue...
20
21 exit
22 main ENDP
23 END main
```

3 Irvine Library Functions

Reading Assignment: Read the 5.3 The Book's Link Library section of the book Assembly Language For x86 Processor sixth edition by Kip R. Irvine.

4 Working with character strings

```
; Author: Abuzar Ghafari
; Program Name: exp8
; Program Description: Replacing the 5th index of charStr
; with the char '_'
; Date: 10/18/2020

INCLUDE Irvine32.inc

.data
   charStr byte "Hello World!", 0
.code
main PROC

   mov ebx, offset charStr    ; ebx points to charStr
   add ebx, 5                 ; add index to ebx
   mov al, 5FH                ; al contains char '_' to replace
   mov [ebx], al              ; write '_' in 5th index of charStr

   call WaitMsg               ; Press any key to continue...

exit
main ENDP
END main
```