



EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

LABORATORY MANUAL ⚫ Fall 2021 ⚫ Instructor: Aamir Ali, Aashir Mehboob, Amin Sadiq, Qurat- ul- Ain, M Kariz, Rabia

|  |  |
| --- | --- |
| **LAB 05** |  |
| **COMPUTER ORGANIZATION AND ASSEMBLY LANG(COAL)** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  |  |
|  | STUDENT NAME | ROLL NO | |  | SEC |
|  | | | | | |
| SIGNATURE & DATE | | | | | |
| **MARKS AWARDED:** | | | | | |
|  | | | | | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI** | | | | | |
| Prepared by: | Amin Sadiq | Version: 1.0 |  |  |  |

**[COMP. ORGANIZATION & ASSEMBLY LANG(COAL)] [Fall 2022 - COAL LAB]**



**LAB: 05**

**Lab Session 01: JMP, LOOP & BUILT-IN-PROCEDURE**

# Objectives:

* JMP Instruction
* Loop Instruction
* Built-in-Procedure

**JMP INSTRUCTION**

Jumping is the most direct method of modifying the instruction flow. A *transfer of control*, or *branch*, is a way of altering the order in which statements are executed. There are two basic types of transfers:

* **Unconditional Transfer**
* **Conditional Transfer**

o **UNCONDITIONAL**

The unconditional jump instruction (jmp) unconditionally transfers control to the instruction located at the target address i.e. there is no need to satisfy any condition for the jump to take place. . The general format is:

**JMP *destination***

When the CPU executes an unconditional transfer, the offset of *destination* is moved into the instruction pointer, causing execution to continue at the new location.

**Syntax:**

### L2:

**………………… JMP L1**

### ………………...

**L1:**

### ………............... JMP L2

Instructor: Zakir Hussain **National University of Computer & Emerging**

**Page 1 of 10**

**Sciences, Karachi**

**[Fall 2022 - COAL LAB] [COMP. ORGANIZATION & ASSEMBLY LANG(COAL)]**

**LAB: 05**

### EXAMPLE # 01:

### INCLUDE Irvine32.inc

### .data

### mystring byte "hello",0

### .code

### main PROC

### L1:

### mov edx, offset mystring

### call DumpRegs

### jmp L1

### exit

### main ENDP

### END main

o **CONDITIONAL**

In these types of instructions, the processor must check for the particular condition. If it is true, then only the jump takes place else the normal flow in the execution of the statements is maintained. Syntax is:

**JMP opcode *destination***

o **CMP INSTRUCTION**

The CMP instruction compares two operands. It is generally used in conditional execution. This instruction basically subtracts one operand from the other for comparing whether the operands are equal or not. It does not disturb the destination or source operands. It is used along with the conditional jump instruction for decision making. Syntax is:

### CMP Destination, Source

|  |  |
| --- | --- |
| Some conditional jump instructions treat operands of the CMP (compare) instruction as signed numbers. | |
| **Mnemonic** | **Description** |
| **JE** | Jump if equal |
| **JG/JNLE** | Jump if greater/Jump if not less than or equal |
| **JL/JNGE** | Jump if less/Jump if not greater |
| **JGE/JNL** | Jump if greater or equal/Jump if less |
| **JLE/JNG** | Jump if less or equal/Jump if not greater |
| **JNE** | Jump if not equal |



**National University of Computer & Emerging Sciences, Karachi** Instructor: Zakir Hussain

**Page 2 of 10**

**[COMP. ORGANIZATION & ASSEMBLY LANG(COAL)] [Fall 2022 - COAL LAB]**

**LAB: 05**

|  |
| --- |
| Some conditional jump instructions can also test values of the individual CPU flags: |
|  |

### EXAMPLE # 02:

### INCLUDE Irvine32.inc

### .data

### .code

### main PROC

### mov eax, 1

### L1:

### add eax, 1

### cmp eax, 9

### je endd

### call DumpRegs

### jmp L1

### endd:

### exit

### main ENDP

### END main

Instructor: Zakir Hussain **National University of Computer & Emerging Sciences, Karachi**

**Page 3 of 10**

**[Fall 2022 - COAL LAB] [COMP. ORGANIZATION & ASSEMBLY LANG(COAL)]**

**LAB: 05**

**LOOP INSTRUCTION**

The LOOP instruction, formally known as *Loop According to ECX Counter*, repeats a block of statements a specific number of times. ECX is automatically used as a counter and is decremented each time the loop repeats.

Its syntax is:

**LOOP *destination***

The execution of the LOOP instruction involves two steps: First, it subtracts 1 from ECX. Next, it compares ECX to zero. If ECX is not equal to zero, a jump is taken to the label identified by *destination*. Otherwise, if ECX equals zero, no jump takes place, and control passes to the instruction following the loop.

### EXAMPLE # 01:

*INCLUDE Irvine32.inc*

*.code*

*main PROC mov ax,0 mov ecx,5 L1:*

*Inc ax*

*call dumpregs loop L1*

*exit*

*main ENDP END main*

**EXAMPLE # 02:**

*INCLUDE Irvine32.inc*

*.data*

*intArray WORD 100h, 200h, 300h, 400h, 500h*

*.code*

*main PROC mov esi, 0*

*mov eax, 0*

*mov ecx, LENGTHOF intArray call dumpregs*

*L1:*

*mov ax, intArray[esi] add esi, TYPE intArray call dumpregs*

*loop L1 exit*

*main ENDP END main*



**National University of Computer & Emerging Sciences, Karachi** Instructor: Zakir Hussain

**Page 4 of 10**

**[COMP. ORGANIZATION & ASSEMBLY LANG(COAL)] [Fall 2022 - COAL LAB]**



**LAB: 05**

o **NESTED LOOPS**

When creating a loop inside another loop, special consideration must be given to the outer loop counter in ECX. You can save it in a variable.

**EXAMPLE # 03**

*INCLUDE Irvine32.inc*

*.code*

*main PROC mov eax, 0*

*mov ebx, 0*

*mov ecx, 5 L1:*

*inc eax*

*mov edx, ecx call dumpregs mov ecx, 10 L2:*

*inc ebx*

*call dumpregs loop L2*

*mov ecx, edx loop L1*

*call DumpRegs exit*

*main ENDP*

**PROCEDURE IN IRVINE32 LIBRARY**

## Clrscr

Clears the console window and locates the cursor at the above left corner.

## Crlf

Writes the end of line sequence to the console window.

## WriteBin

Writes an unsigned 32-bit integer to the console window in ASCII binary format.

## WriteChar

Writes a single character to the console window.

## WriteDec

Writes an unsigned 32-bit integer to the console window in decimal format.

## WriteHex

Writes a 32-bit integer to the console window in hexadecimal format.

**Page 5 of 10**

Instructor: Zakir Hussain **National University of Computer & Emerging Sciences, Karachi**

**[Fall 2022 - COAL LAB] [COMP. ORGANIZATION & ASSEMBLY LANG(COAL)]**

**LAB: 05**

## WriteInt

Writes a signed 32-bit integer to the console window in decimal format.

## WriteString (EDX= OFFSET String)

Write a null-terminated string to the console window.

## ReadChar

Waits for single character to be typed at the keyboard and returns that character.

## ReadDec

Reads an unsigned 32-bit integer from the keyboard.

## ReadHex

Reads a 32-bit hexadecimal integers from the keyboard, terminated by the enter key.

## ReadInt

Reads a signed 32-bit integer from the keyboard, terminated by the enter key.

## ReadString (EDX=OFFSET, ECX=SIZEOF)

Reads a string from the keyboard, terminated by the enter key.

## Delay (EAX)

Pauses the program execution for a specified interval (in milliseconds).

## Randomize

Seeds the random number generator with a unique value.

## DumpRegs

Displays the EAX, EBX, ECX, EDX, ESI, EDI, ESP, EIP and EFLAG registers.

## DumpMem (ESI=Starting OFFSET, ECX=LengthOf, EBX=Type)

Writes the block of memory to the console window in hexadecimal.

## getDateTime

Gets the current date and time from system

## GetMaxXY (DX=col, AX=row)

Gets the number of columns and rows in the console window buffer.

## GetTextColor (Background= Upper AL, Foreground= Lower AL)

Returns the active foreground and background text colors in the console window.

## Gotoxy (DH=row , DL=col)

Locates the cursor at a specific row and column in the console window. By default X coordinate range is 0-79,and Y coordinate range is 0-24.

## MsgBox (EDX=OFFSET String, EBX= OFFSET Title)

**Page 6 of 10**

**National University of Computer & Emerging Sciences, Karachi** Instructor: Zakir Hussain

**[COMP. ORGANIZATION & ASSEMBLY LANG(COAL)] [Fall 2022 - COAL LAB]**

**LAB: 05**

Displays a pop-up message box.

## MsgBoxAsk (EDX=OFFSET String, EBX= OFFSET Title)

Displays a yes/no question in a pop-up message box. (EAX=6 YES, EAX=7 NO)

## SetTextColor (EAX= Foreground + (Background\*16))

Sets the foreground and background colors of all subsequent text output to the console.

## WaitMsg

Display a message and wait for the Enter key to be pressed.

**EXAMPLE # 04**

**WriteDec:** The integer to be displayed is passed in EAX **WriteString:** The offset of string to be written is passed in EDX **WriteChar:** The character to be displayed is passed in AL

*INCLUDE Irvine32.inc*

*.data*

*Dash BYTE " - ", 0*

*.code*

*main PROC*

*mov ecx, 255 mov eax,1*

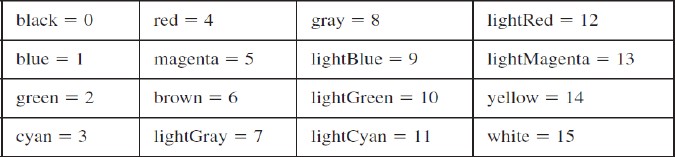
*mov edx, OFFSET Dash L1:*

*exit*

*call WriteDec ; EAX is a counter call WriteString ; EDX points to string call WriteChar ; AL is the character call Crlf*

*inc al ; next character Loop L1*

*main ENDP END main*



Instructor: Zakir Hussain **National University of Computer & Emerging Sciences, Karachi**

**Page 7 of 10**

**[Fall 2022 - COAL LAB] [COMP. ORGANIZATION & ASSEMBLY LANG(COAL)]**

**LAB: 05**

**EXAMPLE # 05**

**DumpMem:** Pass offset of array in ESI, length of array in ECX & type in EBX

**ReadInt:** Reads the signed integer into EAX

**WriteInt:** Signed integer to be written is passed in EAX **WriteHex:** Hex value to be written is passed in EAX **WriteBin:** Binary value to be written is passed in EAX

*INCLUDE Irvine32.inc*

*.data*

*COUNT = 4*

*arrayD SDWORD 12345678h, 1A4B2000h, 3434h, 7AB9h*

*prompt BYTE "Enter a 32-bit signed integer: ", 0*

*.code*

*main PROC*

*; Display an array using DumpMem.*

*mov esi, OFFSET arrayD ; starting OFFSET*

*mov ebx, TYPE arrayD ; doubleword = 4 bytes*

*mov ecx, LENGTHOF arrayD ; number of units in arrayD call DumpMem ; display memory*

*call DumpRegs*

*; Ask the user to input a sequence of signed integers*

*call Crlf ; new line*

*mov ecx, COUNT L1:*

*mov edx, OFFSET prompt call WriteString*

*call ReadInt ; input integer into EAX*

*call Crlf ; new line*

*; Display the integer in decimal, hexadecimal, and binary*

*call WriteInt ; display in signed decimal*

*call Crlf*

*call WriteHex ; display in hexadecimal*

*call Crlf*

*call WriteBin ; display in binary*

*call Crlf call Crlf*

*Loop L1 ; repeat the loop*

*exit*

*main ENDP END main*

**EXAMPLE # 06**

**SetTextColor:** Background & foreground colors are passed to EAX

*. INCLUDE Irvine32.inc*

*.data*

*str1 BYTE "Sample string in color", 0dh, 0ah, 0*

*.code*

*main PROC*

**Page 8 of 10**

**National University of Computer & Emerging Sciences, Karachi** Instructor: Zakir Hussain

**[COMP. ORGANIZATION & ASSEMBLY LANG(COAL)] [Fall 2022 - COAL LAB]**

**LAB: 05**

*mov eax, yellow + (blue\*16) call SetTextColor*

*mov edx, OFFSET str1 call WriteString*

*call DumpRegs*

*exit*

*main ENDP END main*

**EXAMPLE # 07**

*INCLUDE Irvine32.inc*

*.data*

*.code*

*caption BYTE "Dialog Title", 0*

*HelloMsg BYTE "This is a pop-up message box.", 0dh,0ah BYTE "Click OK to continue...", 0*

*main PROC*

*mov ebx, 0 ; no caption*

*mov edx, OFFSET HelloMsg ; contents call MsgBox*

*exit*

*mov ebx, OFFSET caption ; caption*

*mov edx, OFFSET HelloMsg ; contents call MsgBox*

*main ENDP END main*



**EXAMPLE # 08:**

**MsgBoxAsk:** Offset of question string is passed in EDX. Offset of caption is passed in EBX. Selected value is returned in EAX (IDYES equal to 6 or IDNO equal to 7)

INCLUDE Irvine32.inc

.data

caption BYTE "Survey Completed",0

question BYTE "Thank you for completing the survey." BYTE 0dh, 0ah

BYTE "Would you like to receive the results?", 0

.code

main PROC

mov ebx, OFFSET caption mov edx, OFFSET question call MsgBoxAsk

;(check return value in EAX)

exit

main ENDP END main

Instructor: Zakir Hussain **National University of Computer & Emerging Sciences, Karachi**

**Page 9 of 10**

**[Fall 2022 - COAL LAB] [COMP. ORGANIZATION & ASSEMBLY LANG(COAL)]**

**LAB: 05**

# Exercise: Dry Run on Paper First then on IDE

### Task: 1

Write a program that uses a loop to calculate the first ten numbers of Fibonacci sequence.

### Task: 2

Initialize a double word array consisting of elements 8,5,1,2,6. Sort the given array in ascending order using bubble sort.

### Task: 3

Write a program that uses a nested loop to implement following patterns.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1111 | 4321 | 1234 |
| 11 | 111 | 432 | 123 |
| 111 | 11 | 43 | 12 |
| 1111 | 1 | 4 | 1 |

### Task: 4

Write a program to take input data for 5 employees and store it in appropriate variables. The program should ask for Employee ID, Name, Year of Birth & Annual Salary from the user. All variables should be stored in an array whose index represent employee number.

The program should then calculate the annual salary for all employees by adding all the elements in Annual Salary array.

### Task: 5

Initialize an array named Source and use a loop with indexed addressing to copy a string represented as an array of bytes with a null terminator value in an array named as target.

### Task: 6

Use a loop with direct or indirect addressing to reverse the elements of an integer array in place. Do not copy

elements to any other array. Use SIZEOF, TYPE and LENGTHOF operators to make program flexible.



**National University of Computer & Emerging Sciences, Karachi** Instructor: Zakir Hussain

**Page 10 of 10**