

LAB 07

EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

**CONDITIONAL PROCESSING**



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

**[Conditional Processing] [Fall 2022 - COAL LAB]**

**LAB: 07**

**Lab Session 07: CONDITIONAL PROCESSING**

**Objectives:**

* Boolean Instructions
* Set Operations
* CMP Instruction
* Conditional Jumps

**Boolean Instructions**

## AND

Boolean AND operation between a source operand and destination operand.

**Syntax:** *AND reg, reg AND reg, mem AND reg, imm AND mem, reg AND mem, imm*

## OR

Boolean OR operation between a source operand and destination operand.

**Syntax:** *OR reg, reg OR reg, mem OR reg, imm OR mem, reg OR mem, imm*

## XOR

Boolean XOR operation between a source operand and destination operand.

**Syntax:** *XOR reg, reg XOR reg, mem XOR reg, imm XOR mem, reg XOR mem, imm*

## NOT

Boolean NOT operation on a destination operand.

**Syntax:** *NOT reg NOT mem*



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

**Page 1 of 9**

**[Fall 2022 - COAL LAB] [Conditional Processing]**

**LAB: 07**

## TEST

Similar to AND operation, except that instead of affecting any operands it sets the FLAGS appropriately.

**Syntax:** *TEST reg, reg TEST reg, mem TEST reg, imm TEST mem, reg TEST mem, imm*

#### Example 01:

Include Irvine32.inc

.code main proc

|  |  |  |
| --- | --- | --- |
| mov | al, 10101110b | ; Clear only bit 3 |
| and | al, 11110110b | ; AL = 10100110 |
| mov | al, 11100011b | ; set bit 2 |
| or | al, 00000100b | ; AL = 11100111 |
| mov | al, 10110101b | ; 5 bits means odd parity |
| xor | al, 0 | ; PF = 0 (PO) |
| mov | al, 10100101b | ; 4 bits means even parity |
| xor | al, 0 | ; PF = 1 (PE) |
| mov not | al, 11110000b  al | ; AL = 00001111b |
| mov test | al, 00100101b al, 00001001b | ; ZF = 0 |
| mov test call | al, 00100101b al, 00001000b  DumpRegs | ; ZF = 1 |

exit

main ENDP END main



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

**Page 2 of 8**

**[Conditional Processing] [Fall 2022 - COAL LAB]**

**LAB: 07**

# Set Operations (using Boolean instructions)

## Set Complement

The complement of a set can be achieved through NOT instruction.

## Set Intersection

The intersection of two sets can be achieved through AND instruction.

## Set Union

The union of two sets can be achieved through OR instruction.

#### Example 02:

Include Irvine32.inc

.data

A DWORD 10000000000000000000000000000111b B DWORD 10000001010100000000011101100011b

msg1 BYTE "A intersection B is: ", 0 msg2 BYTE "A union B is: ", 0

msg3 BYTE "Complement of A is: ", 0

.code main proc

mov eax,A

and eax, B ; A intersection B mov edx, OFFSET msg1

call WriteString

mov ebx, TYPE DWORD call WriteBinB

call Crlf

mov eax, A

or eax, B ; A union B mov edx, OFFSET msg2

call WriteString

mov ebx, TYPE DWORD call WriteBinB

call Crlf

mov eax, A

not eax ; A complement mov edx, OFFSET msg3

call WriteString

mov ebx, TYPE DWORD call WriteBinB

call DumpRegs exit

main ENDP

END main 

**Page 3 of 9**

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

**[Fall 2022 - COAL LAB] [Conditional Processing]**

**LAB: 07**

# CMP instruction

CMP (compare) instruction performs an implied subtraction of a source operand from a destination operand for comparison.

For unsigned operands:

|  |  |  |
| --- | --- | --- |
| * Destination < source | ZF = 0 | CF = 1 |
| * Destination > source | ZF = 0 | CF = 0 |
| * Destination = source | ZF = 1 | CF = 0 |

For signed operands:

* + - Destination < source SF ! = OF
    - Destination > source SF = OF
    - Destination = source ZF = 1

#### Example 03:

Include Irvine32.inc

.code main proc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mov | ax, 5 |  | | |
| cmp | ax, 10 | ; ZF = 0 | and | CF = 1 |
| mov | ax, 1000 |  |  |  |
| cmp | ax, 1000 | ; ZF = 1 | and | CF =0 |
| mov | si, 106 |  |  |  |
| cmp | si, 0 | ; ZF = 0 | and | CF = 0 |

call DumpRegs exit

main ENDP END main



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

**Page 4 of 8**

**[Conditional Processing] [Fall 2022 - COAL LAB]**

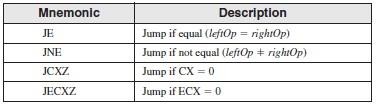
**LAB: 07**

# Conditional Jumps

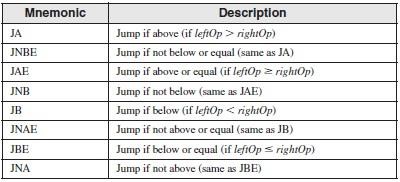
## Jumps based on Flag values



* + **Jumps based on Equality**



## Jumps based on unsigned comparisons





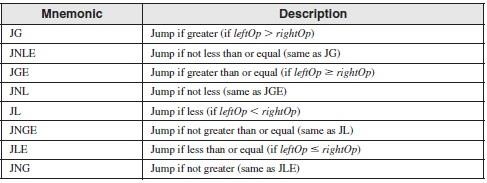
Instructor: Zakir Hussain **National University of Computer & Emerging Sciences, Karachi**

**Page 5 of 9**

**[Fall 2022 - COAL LAB] [Conditional Processing]**

**LAB: 07**

## Jumps based on signed comparisons



#### Example 04:

Include Irvine32.inc

.data

var1 DWORD 250

var2 DWORD 125

larger DWORD ?

.code main proc

mov eax, var1 mov larger, eax mov ebx, var2 cmp eax, ebx

jae L1

mov larger, ebx L1: call DumpRegs exit

main ENDP END main

#### Example 05:

Include Irvine32.inc

.data

var1 DWORD 50

var2 DWORD 25

var3 DWORD 103

msg BYTE "The smallest integer is: ", 0

.code main proc

moveax, var1

cmp eax, var2  jbe L1

**Page 6 of 8**

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

**[Conditional Processing] [Fall 2022 - COAL LAB]**

**LAB: 07**

mov eax, var2 L1:

cmp eax, var3 jbe L2

mov eax, var3 L2:

mov edx, OFFSET msg call WriteString

call WriteDec call DumpRegs exit

main ENDP END main

#### Example 06:

Include Irvine32.inc

.data

char BYTE ?

.code main proc L1:

|  |  |  |
| --- | --- | --- |
| mov | eax, 10 | ; create 10ms delay |
| call | Delay |  |
| call | ReadKey | ; reads a key input |
| jz | L1 | ; repeat if no key is pressed |
| mov | char, al | ; saves the character |

call DumpRegs exit

main ENDP END main



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

**Page 7 of 9**

**[Fall 2022 - COAL LAB] [Conditional Processing]**

**LAB: 07**

**Lab Task(s):**

### Translate the following pseudo-code to Assembly Language:

#### var = 5

**if ( var<ecx ) AND (ecx>=edx) then**

#### x = 0

**else**

**x = 1**

1. Use cmp and jumps to find the first non-zero value in the given array:

## intArr SWORD 0, 0, 0, 0, 1, 20, 35, -12, 66, 4, 0

### Write a program that takes four input integers from the user. Then compare and display a message whether these integers are equal or not.

1. Write a program for sequential search. Take an input from the user and find if it occurs in the following array:

## arr WORD 10, 4, 7, 14, 299, 156, 3, 19, 29, 300, 20

### Translate the following pseudo-code to Assembly Language:

#### Swap\_Count = 0

**for all elements of list**

#### if list[i] > list[i+1]

**swap(list[i], list[i+1]) Swap\_Count = Swap\_Count + 1**

#### end if

**end for**

#### Print Swap\_Count



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

**Page 8 of 8**