National University of Computer and Emerging Sciences



**Laboratory Manual**

*for*

*Computer Organization and Assembly Language*

Course Instructors

Lab Instructor(s)

Section

Semester

**Department of Computer Science**

# COAL Lab 11 Manual

|  |
| --- |
| **Objectives:**     * Bit Manipulation * Shift Rotate instructions * MUL DIV instructions * Problems & Assignments |

**11.1 Shift and Rotate Instructions:**

For a shift instructions, the bits shifted out are lost, for a rotate instruction, bit shifted out from one end of the operand are put back into the other end. For a single shift or rotate of N-positions:

Instruction Destination, count

**Note:** In both cases, destination can be a register or memory location, count can be CL register or 8-bit immediate value.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Syntax** | **Explanation** | **Examples** |
| **SHL/SAL** | SHL Dest, CL SAL Dest, CL |  | **EX 11.1: Multiply EAX by 8** |
| **SAL** **EAX**,3 |
| **SHR** | SHR Dest, CL |  | **EX 11.2: Divide EAX by 8** |
| **MOV** **CL**,3 **SHR** **EAX**,CL |
| **SAR** | SAR Dest, CL |  | **EX 11.3: Divide AX by 8** |
| **MOV** **CL**,3 **SAR** **AX**,CL |
| **ROL** | ROL Dest, CL |  | **EX 11.4: Count 1’s in BX** |
| **XOR** **AX**, **AX MOV** **CX**,16  ONE: **ROL** **BX**,1  **JNC** NEXT  **INC** **AX**  NEXT:**LOOP** ONE |
| **ROR** | ROR Dest, CL |  | SAME AS EX 11.4 |
| **RCL** | RCL Dest, CL |  |  |
| **RCR** | RCL Dest, CL |  |  |

# 11.2 SHLD/SHRD Instructions

**Syntax**

SHLD *dest, source, count*

The following instruction formats apply to both SHLD and SHRD. The destination operand can be a register or memory operand, and the source operand must be a register. The count operand can be the CL register or an 8-bit immediate operand:

SHLD *reg16,reg16,CL/imm8*

SHLD *mem16,reg16,CL/imm8*

SHLD *reg32,reg32,CL/imm8* SHLD *mem32,reg32,CL/imm8*

## 11.3 MUL Instruction

The syntax of these instructions is;

MUL source

**Note:** The source may be 8/16/32 bit register or memory but not a constant.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **MULTIPLICATION** | | **Syntax** | **Source**  **(1st operand or Multiplier)** | **Destination**  **(2nd operand or**  **Multiplicand)** | **Product** | | **Examples** |
| **Higher Byte/word** | **Lower Byte/word** |
| **Byte Form** | **MUL** | MUL source | Reg8 or mem8 | AL | AH | AL | **MOV** **AL**,5  **MOV** **BL**,2  **MUL** **BL**  ;**AX** = 000A ;CF = 0 |
| **Word Form** | Reg16 or mem16 | AX | DX | AX | **MOV** **AX**,50  **MOV** **BX**,20  **MUL** **BX**  ; **DX** = 0000h  ; **AX** = 03E8h ; CF = 0 |
| **Double Word Form** | Reg32 or mem32 | EAX | EDX | EAX | **MOV** **EAX**,5050  **MOV** **EBX**,20  **MUL** **EBX**  ;**EDX**= 00 00  00 00h  ;**EAX**= 00 01  8A88h  ;CF = 0 |

## 11.4 DIV instruction

The syntax of these instructions is;

DIV divisor

**Note:** The source may be 8/16/32 bit register or memory but not a constant.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DIVISION** | | **Syntax** | **Divisor** | **Dividend** | | **Answer** | | **Examples** |
| **Higher Byte/word** | **Lower Byte/word** | **Remainder** | **Quotient** |
| **Byte**  **Form** | **DIV** | DIV divisor | Reg8 or mem8 | AH | AL | AH | AL | **MOV** **AX**,5  **MOV** **BL**,2 **DIV** **BL**  ; **AH** = 1  ; **AL** = 2 |
| **Word Form** | Reg16 or mem16 | DX | AX | DX | AX | **MOV** **DX**,0000  **MOV** **AX**,2004  **MOV** **BX**,100 **DIV** **BX**  ; **AX** = 14h  ; **DX** = 04h |
| **Double**  **Word**  **Form** | Reg32 or mem32 | EDX | EAX | EDX | EAX |  |

**Divide Overflow:**

It is possible that the quotient will be too big to fit in the specified destination (AL or AX). This can happen if the divisor is much smaller that the dividend. When this happens, the program terminates and the system displays the message “Divide Overflow”.

MOV DX,FFFFh MOV AX,FFFBh

MOV BX,0002h

DIV EBX

# Problem(s) / Assignment(s)

|  |  |
| --- | --- |
| **Discussion & Practice** | **Estimated completion time: 1 hr, 30 mins** |

|  |  |
| --- | --- |
| **PROBLEM 11.1: NUMBER SYSTEM CONVERSION**  **Write a program that converts 32-bit hex input into equivalent binary output using procedure CONVERT\_BINARY and display the result.**  **Note: Don’t use any built-in procedure for binary number conversion.**  **Solution:**  include Irvine32.inc  .data  input byte "Input 32 bit hexadecimal number",0  .code  main proc  mov eax,0  mov edx,offset input  call writestring  call readhex  mov ebx,eax  call convert\_binary  exit1:  call crlf  exit  main endp  convert\_binary proc  mov ecx,32  l1:  shl ebx,1  jc return  mov al,'0'  call writechar  loop l1  ret  jmp ex  return:  mov al,'1'  call writechar  jmp l1  ex:  convert\_binary endp  end main | **Estimated completion time:30mins** |

|  |  |
| --- | --- |
| **PROBLEM 11.2: NUMBER SYSTEM CONVERSION**  **Write a program that takes a Binary input (0000 0000-1111 1111) from user using procedure IN\_PROC1. Convert the input into hexadecimal using procedure CONVERT\_HEX. Display the output using procedure OUT\_PROC1.** | **Estimated completion time:30mins** |

|  |  |
| --- | --- |
| **PROBLEM 11.3: NUMBER SYSTEM CONVERSION**  **Write a program that takes a digit input (0-F) from user (use ReadChar) using procedure IN\_PROC, store it in BL. Convert the input into decimal (make a logic using MUL/DIV) using procedure CONVERT\_DECIMAL. Display the output in decimal using procedure OUT\_PROC.**  **Solution:**  include Irvine32.inc  .data  .code  main proc  call IN\_proc  call convrtdecimal  call out\_proc  exit  main endp  IN\_proc proc  call readhex  MOV BL, 10  ret  IN\_proc endp  convrtdecimal proc  DIV BL  MOV DH, AH  MOV DL,AL  ret  convrtdecimal endp  out\_proc PROC  mov eax,0  MOV AL,DL  call writedec  mov eax, 0  MOV AL,DH  call writedec  ret  out\_proc ENDP  end main | **Estimated completion time:30mins** |

|  |  |
| --- | --- |
| **BONUS TASK**: **Write a program that take input of Byte size from the user, store it in BL reverse that input and display it in Hexadecimal.**    **Sample:**  **Entered value: 4B (0100 1011)**  **Reverse Output is:** **D2 (1101 0010)**  Solution:  include Irvine32.inc  .data  input byte "Enter Number",0  .code  main proc  mov eax,0  mov edx,offset input  call writestring  call readhex  mov bl,al  mov ecx,8  l1:  rcr al,1  rcl bl,1  loop l1  mov eax,0  mov al,bl  call writehex  call crlf  exit  main endp  end main | **Estimated completion time: 30mins** |