EL213: Computer Org. & Assembly Language Lab

# Lab#10: Integer Arithmetic - II

## Agenda

* Shift and Rotate Instructions
  + SHLD/SHRD Instructions
* Multiplication and Division Instructions
  + MUL Instruction
  + IMUL Instruction
  + DIV Instruction
  + CBW, CWD, CDQ Instructions
    - CBW Instruction
    - CWD Instruction
    - CDQ Instruction
  + IDIV Instruction

### SHLD/SHRD Instructions

The SHLD (shift left double) instruction shifts a destination operand a given number of bits to the left. The bit positions opened up by the shift are filled by the most significant bits of the source operand. The source operand is not affected , but the Sign, Zero, Auxiliary, Parity, and Carry flags are affected:

SHLD destination, source, count

The SHRD (shift right double) instruction shifts a destination operand a given number of bits to the right. The bit positions opened up by the shift are filled by the least significant bits of the source operand:

SHRD destination, 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 either 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

**Examples:**

**SHLD**

. data

wval WORD 9BA6h

. code

mov ax,0AC36h

shld wval,ax,4 ; wval BA6Ah

**SHRD**

mov ax,234Bh

mov dx,7654h

shrd ax,dx,4 ; AX = 4234h

; Display a 32-bit integer in binary.

INCLUDE Irvine32.inc

.data

binValue DWORD 1234ABCDh

buffer BYTE 32 DUP(0) ,0

.code

main PROC

mov eax,binValue

mov ecx,32

mov esi,OFFSET buffer

L1:

shl eax,1

mov BYTE PTR [esi],'0'

jnc L2

mov BYTE PTR [esi],'1'

L2:

inc esi

loop L1

mov edx,OFFSET buffer

call WriteString

call Crlf

exit

main ENDP

END main

## Multiplication and Division Instructions

### MUL Instruction

The following statements perform 8-bit unsigned multiplication (5 \* 10h), producing 50h in AX:

mov al,5h

mov bl,l0h

mul bl ; CF = 0

The Carry flag is clear because AH (the upper half of the product) equals zero.

The following statements perform 16-bit unsigned multiplication (0100h \* 2000h).producing 00200000h in DX:AX:

.data

val1 WORD 2000h

val2 WORD 0100h

.code

mov ax , val1

mul val2 ; CF = 1

The Carry flag is set because DX is not equal to zero.

The following statements perform 32-bit unsigned multiplication (12345h \* 1000h), producing 0000000012345000h in EDX:EAX:

Mov eax,12345h

mov ebx , 1000h

mul ebx ; CF = 0

The Carry flag is clear because EDX equals zero.

### IMUL Instruction

The IMUL (signed multiply ) instruction performs signed integer multiplication. It has the same syntax and uses the same operands as the MUL instruction. The difference is that it preserves the sign of the product.

IMUL sets the Carry and Overflow flags if the high-order product is not a sign extension of the low-order product. (Because the Overflow flag is normally used for signed arithmetic , we will focus on it here. ) The following examples help to illustrate:

**Example 1:** The following instructions perform 8-bit signed multiplication (48 \*4), producing +192 in AX:

Mov al , 48

mov bl , 4

imul bl ; AX = 00C0 h , OF = 1

AH is not a sign extension of AL, so the Overflow flag is set.

**Example 2:** The following instructions perform 8-bit signed multiplication (- 4 \* 4), producing -16 in AX:

mov al, -4

mov bl, 4

imul bl ; AX = FFF0h, OF = 0

AH is a sign extension of AL (the signed result fits within AL), so the Overflow flag is clear.

### DIV Instruction

The following instructions perform 8-bit unsigned division (83h / 2), producing aquotient of 41h and a remainder of 1:

mov ax,0083h ;dividend

mov bl,2 ;divisor

div bl ;AL = 41h, AH = 0lh

### CBW, CWD, CDQ Instructions

#### CBW Instruction

The CBW (convert byte to word) instruction extends the sign bit of AL into the AH register. This preserves the number's sign:

.data

byteVal SBYTE -101 ;9Bh

.code

mov al,byteVal ;al = 9Bh

cbw ;ax = FF9Bh

#### CWD Instruction

The CWD (convert word to doubleword) instruction extends the sign bit of AX into the DX register.

.data

wordVal SWORD -101 ;FF9Bh

.code

mov ax,wordVal ;ax = FF9Bh

cbw ;dx:ax = FFFFFF9Bh

#### CDQ Instruction

The CDQ (convert doubleword to quadword) instruction extends the sign bit of EAX into the EDX register:

. data

dwordVal SWORD -101 ;FFFFFF9Bh

. code

mov eax,wordVal ;Eax = FFFFFF9Bh

cbw ;Edx:Eax = FFFFFFFFFFFFFF9Bh

### IDIV Instruction

Performs signed division

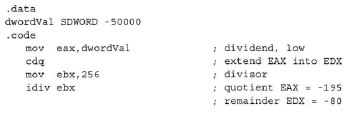
When doing 8-bit division, you must sign-extend the dividend into AH before IDIV executes. (The CBW instruction can be used.) In the next example, we divide -48 by 5. After IDlY executes, the quotient in AL is -9 and the remainder in AH is - 3:



Similarly, l6-bit division requires that AX be sign-extended into DX. In the next example. we divide -5000 by 256:



Similarly, 32-bit division requires that EAX be sign-extended into EDX. The next example divides - 50000 by 256:



# Practice Session

1. Write a program to that tells the user if a given number is even or odd
   * 1. The number should be entered by the user.
     2. Use procedures.
2. Create a array of 10 integers containing numbers from 1 to 10, write a program that should:
   * 1. Be able to find out the prime numbers in that array.
     2. Also the number of prime numbers found should be printed on the screen.
     3. Use proper messages to display output in your program.