Chapter THREE

Arithmetic and Logic Instructions

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall

2.5: DATA TYPES AND DATA DEFINITION

- Figure 2-7 shows the memory dump of the data section.
  - It is essential to understand the way operands are stored in memory.

```
-D 1066:0 100
1066:0000 19 89 12 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
1066:0010 32 35 39 31 00 00 00 00-00 00 00 00 00 00 00 00 2591............
1066:0020 4D 79 20 6E 61 6D 65 20-69 73 20 4A 6F 65 00 00 My name is Joe..
1066:0030 FF FF FF FF FF FF 00 00-FF FF FF FF FF FF 00 00 ................
1066:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
1066:0060 63 63 63 63 63 63 63 63-63 63 00 00 00 00 00 00 cccccccccc......
1066:0070 BA 03 54 09 3F 25 00 00-09 00 02 00 07 00 0C 00 :.T.?%..........
1066:0080 20 00 05 00 4F 48 00 00-00 00 00 00 00 00 00 00  ...OH...........
1066:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
1066:00A0 FF 03 00 00 5C 96 08 00-F2 57 2A 5C 23 00 00 00 ....\...rW*\#...
1066:00B0 89 47 03 00 FD FF 00 00-00 00 00 00 00 00 00 00 B#E......IH.....
1066:00C0 C2 23 45 00 00 00 00 00-49 48 00 00 00 00 00 00 ................
1066:00D0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
1066:00E0 29 98 56 43 79 86 00 00-00 00 00 00 00 00 00 00 9.VCy6..........
```

OBJECTIVES

this chapter enables the student to:

- Demonstrate how 8-bit and 16-bit unsigned numbers are added in the x86.
- Convert data to any of the forms:
  - ASCII,packed BCD,unpacked BCD.
- Explain the effect of unsigned arithmetic instructions on the flags.
- Code the following Assembly language unsigned arithmetic instructions:
  - Addition instructions: **ADD** and **ADC**.
  - Subtraction instructions **SUB** and **SBB**.
  - Multiplication and division instructions **MUL** and **DIV**.

3.1: UNSIGNED ADDITION AND SUBTRACTION
addition of unsigned numbers

- The form of the ADD instruction is:

```
ADD destination,source ;destination = destination + source
```

- ADD and ADC are used to add two operands.
  - The destination operand can be a register or in memory.
  - The source operand can be a register, in memory, or immediate.
    - Memory-to-memory operations are never allowed in x86 Assembly language.
  - The instruction could change ZF, SF, AF, CF, or PF bits of the flag register.

3.1: UNSIGNED ADDITION AND SUBTRACTION
addition of unsigned numbers

Example 3-1

Show how the flag register is affected by

```
MOV   AL,0F5H
ADD   AL,0BH
```

Solution:

```
      F5H          1111 0101
+     0BH      +   0000 1011
     100H          0000 0000
```

After the addition, the AL register (destination) contains 00 and the flags are as follows:
CF = 1, since there is a carry out from D7
SF = 0, the status of D7 of the result
PF = 1, the number of 1s is zero (zero is an even number)
AF = 1, there is a carry from D3 to D4
ZF = 1, the result of the action is zero (for the 8 bits)

3.2: UNSIGNED MULTIPLICATION & DIVISION
multiplication of unsigned numbers

- In multiplying two numbers in the x86 processor, use of registers AX, AL, AH, and DX is necessary.
  - The function assumes the use of those registers.
- Three multiplication cases:
  - byte times byte; word times word; byte times word.

Table 3-1: Unsigned Multiplication Summary

Multiplication	Operand 1	Operand 2	Result
byte × byte	AL	register or memory	AX
word × word	AX	register or memory	DX AX
word × byte	AL = byte, AH = 0	register or memory	DX AX

3.2: UNSIGNED MULTIPLICATION & DIVISION
division of unsigned numbers

- **byte/byte** - the numerator must be in the AL register and AH must be set to zero.
  - The denominator cannot be immediate but can be in a register or memory, supported by the addressing modes.
    - After the DIV instruction is performed, the quotient is in AL and the remainder is in AH.

Table 3-2: Unsigned Division Summary

Division	Numerator	Denominator	Quotient	Rem.
byte/byte	AL = byte, AH = 0	register or memory	AL[1]	AH
word/word	AX = word, DX = 0	register or memory	AX[2]	DX
word/byte	AX = word	register or memory	AL[1]	AH
doubleword/word	DXAX = doubleword	register or memory	AX[2]	DX

Notes: 1. Divide error interrupt if AL > FFH. 2. Divide error interrupt if AX > FFFFH.

OBJECTIVES (cont)

this chapter enables the student to:

- Code BCD arithmetic instructions:
  - **DAA** and **DAS**.
- Code the Assembly language logic instructions:
  - **AND**, **OR**, and **XOR**.
  - Logical shift instructions **SHR** and **SHL**.
  - The compare instruction **CMP**.
- Code bitwise rotation instructions
  - **ROR**, **ROL**, **RCR**, and **RCL**.
- Demonstrate an ability to use all of the above instructions in Assembly language programs.
- Perform bitwise manipulation using the C language.

3.3: LOGIC INSTRUCTIONS
SHIFT RIGHT

- SHR - logical shift right.
  - Operand is shifted right bit by bit.
    - For every shift the LSB (least significant bit) will go to the carry flag. (CF)
    - The MSB (most significant bit) is filled with 0.

Example 3-9

Show the result of SHR in the following:

```
MOV   AL,9AH
MOV   CL,3    ;set number of times to shift
SHR   AL,CL
```

Solution:

```
 9AH =      10011010
            01001101     CF = 0 (shifted once)
            00100110     CF = 1 (shifted twice)
            00010011     CF = 0 (shifted three times)
```

After shifting right three times, AL = 13H and CF = 0.

3.3: LOGIC INSTRUCTIONS
SHIFT LEFT

- SHL - Logical shift left, the reverse of SHR.
  - After every shift, the LSB is filled with 0.
    - MSB goes to CF.
  - All rules are the same as for SHR.

Example 3-11

Show the effects of SHL in the following:

```
MOV    DH,6
MOV    CL,4
SHL    DH,CL
```

Solution:

	00000110	
CF=0	00001100	(shifted left once)
CF=0	00011000	
CF=0	00110000	
CF=0	01100000	(shifted four times)

After the four shifts left, the DH register has 60H and CF = 0.

3-11 can also be coded as:

```
MOV    DH,6
SHL    DH,1
SHL    DH,1
SHL    DH,1
SHL    DH,1
```

3.3: LOGIC INSTRUCTIONS
COMPARE of unsigned numbers

- CMP destination,source
  - Compares two operands & changes flags according to the result of the comparison, leaving the operand unchanged.
    - Destination operand can be in a register or in memory.
    - Source operand can be in a register, in memory, or immediate.
- CF, AF, SF, PF, ZF, and OF flags reflect the result.
  - Only CF and ZF are used.

Table 3-3: Flag Settings for Compare Instruction

Compare operands	CF	ZF
destination > source	0	0
destination = source	0	1
destination < source	1	0

3.5: ROTATE INSTRUCTIONS

- If the operand is to be rotated once, the 1 is coded.
  - If it is to be rotated more than once, register CL is used to hold the number of times it is to be rotated.