Computer Organization & Assembly Language

- Shift & Rotate Instructions
- Binary & Hex IO

Shift/Rotate Instructions

- Shift the bits in destination operand by one or more positions either to the left or right.
- Shift: Bit shifted out is lost
- Rotate: Bit shifted out from one end of the destination operand is put back on the other end.
- Syntax:

  OPCODE *destination*, *1* ;single shift/rotate

  OPCODE *destination*, *CL* ;for N positions shift/rotate

Where:

destination can be 8-bit or 16-bit registers or memory variable

Shift Instructions

- SHL Instruction (Left Shift)
- SAL (Shift Arithmetic Left)
- SHR (Right Shift)
- SAR (Shift Arithmetic Right)

The SHL Instruction

- Shifts the bit in destination to the left
- Effects on flags:
  - SF, PF, ZF reflects the result
  - AF is undefined
  - CF = last bit shifted out
  - OF = 1 if result changes sign on last shift
- Example:
  - DH = 8Ah
  - CL = 3
  - Value of DH and CF after executing instruction:

SHL DH, CL

Solution: DH = 50h, CF = 0

Contd..

- Multiplication by left shift
  - Consider digit 235, if each digit is shifted left one position and a 0 is attached at right end, the value will be 2350
  - Same as Multiplying 235 by 10
  - Left shift on a binary number means multiplying the number by 2
  - Example: If AL = 5d, after left shift AL = 10d, after another left shift AL = 20d

The SAL Instruction

- Used when numeric multiplication is intended
- Both SHL and SAL instructions generates same machine code.
- Example: Multiply AX by 8

```
MOV CL, 3
SAL AX, CL
```

The SHR Instruction

- Performs right shift on destination operand.
- A 0 is shifted into MSB and rightmost bit is shifted to CF.
- The effect on flag is same as SHL.
- Example:
  - DH = 8Ah
  - CL = 2
  - After executing instruction: SHR DH, CL:
  - CF = 1 and DH = 22h
  - Erase rightmost two bits and add two 0 bits to the left end
- If an unsigned interpretation is being given, use SHR.

The SAR Instruction

- Operates like SHR, with one difference: the MSB retains its original value.
- If number is even, one right shift is same as divide the number by 2.
- If number is odd, one right shift halves it and rounds down to nearest integer.
- Example:

  BL = 0000 0101b = 5d

  After one right shift:

  BL = 0000 0010b = 2d

- If an signed interpretation is being given, use SAR. (preserves the MSB)

Examples

- Use right shift to divide unsigned number 65143 by 4. Put quotient in AX.
- *Solution*:

```
MOV AX, 65143
MOV CL, 2
SHR AX, CL
```

- If AL contains -15, give the decimal value of AL after SAR AL, 1 is performed.
- *Solution*:

The instruction will divide –15 by 2 and round it down to –8

AL = 1111 0001b

AL = 1111 1000b = – 8

Rotate Instructions

- ROL (Rotate Left)
  - MSB is shifted into the rightmost bit
  - CF also gets the bit shifted out of the MSB
  - Syntax:

    ROL *destination*, *1*

    ROL *destination*, *CL*
- ROR (Rotate Right)
  - The rightmost bit is shifted into MSB and also into CF.
  - Syntax:

    ROR *destination*, *1*

    ROR *destination*, *CL*

Contd..

- RCL (Rotate Carry Left)
  - Shifts the bit of destination to the left
  - The MSB is shifted into CF and the previous value of CF is shifted into the rightmost bit.
  - Syntax:

    RCL *destination, 1*

    RCL *destination, CL*
- RCR (Rotate Carry Right)
  - Works just like RCL except that the bits are rotated to the right.
  - Syntax:

    RCR *destination, 1*

    RCR *destination, CL*

Effects of Rotate Instruction on Flags

- SF, PF, ZF reflects the result
- AF is undefined
- CF = last bit shifted out
- OF = 1 if result changes sign on the last rotation

Example

- Use ROL to count the number of 1 bits in BX, without changing BX. Put answer in AX
- *Solution*:

```
XOR AX, AX
MOV CX, 16
TOP:
        ROL BX, 1
        JNC NEXT
        INC AX
NEXT:
        LOOP TOP
```

Example

- Suppose DH contains 8Ah, CF = 1, and CL contains 3. What are the values of DH and CF after the instruction RCR DH, CL is executed?
- *Solution*:

	CF	DH
Initial value	1	1000 1010
After 1 right rotation	0	1100 0101
After 2 right rotations	1	0110 0010
After 3 right rotations	0	1011 0001 = B1h

An application of reversing bit pattern

- AL = 1101 1100, Required = 0011 1011

```
MOV CX, 8
REVERSE:
   SHL AL, 1
   RCR BL, 1
   LOOP REVERSE
   MOV AL, BL
```

Binary Input

- Algorithm to read a binary number from keyboard and stored it in BX

```
Clear BX
Input a character
While character <> CR Do
        Convert character to binary value
        Left shift BX
        Insert value into LSB of BX
        Input a character
End_While
```

Contd..

- Assembly Language Code for Binary Input:

```
XOR BX, BX
MOV AH, 1
INT 21h
WHILE_:
        CMP AL, 0Dh
        JE END_WHILE
        AND AL, 0Fh      ;convert to binary value
        SHL BX, 1
        OR BL, AL
        INT 21h
        JMP WHILE_
END_WHILE:
```

Binary Output

- Algorithm:

```
FOR 16 TIMES DO
   ROTATE LEFT BX
   IF CF = 1
         THEN
                  OUTPUT '1'
         ELSE
                  OUTPUT '0'
   END_IF
END_FOR
```

Hex Input

- Assumptions:
  - Only uppercase letters
  - Maximum four hex characters
- Algorithm for Hex Input:

```
CLEAR BX
Input Hex Character
WHILE Character <> CR DO
        Convert Character To Binary Value
        Left Shift BX Four Times
        Insert Value Into Lower 4 Bits Of BX
        Input A Character
END_WHILE
```

Hex Output

- Algorithm:

```
For 4 Times Do
    Move BH to DL
    Shift DL 4 times to the right
    IF DL < 10
            THEN
                    Convert to character in '0'….'9'
            ELSE
                    Convert to character in 'A'….'F'
    END_IF
    Output Character
    Rotate BX left 4 times
END_FOR
```