Logic, Shift and Rotate Instructions

Logic Instructions

- We can change individual bits in the computer by using logic operations
- The binary values of **0 and 1** are treated as **false and true** respectively
- When a **logic operation** is applied to 8 or 16 bit operands, the result is obtained by applying the logic operation at **each bit position**
- Logic instructions are: AND, OR, XOR and NOT

Mask

- One use of AND, OR and XOR is to **selectively modify** the bits in the destination
- To do is we construct a **source bit pattern** known as **mask**
- The mask bits are chosen so that the corresponding destination bits are modified in the desired way

Mask

- The **AND** instruction
  - Used to **clear** specific destination bits while **preserving** the others.
    - A **0 mask bit clears** the corresponding destination bit
    - A **1 mask bit preserves** the corresponding destination bit
- The **OR** instruction
  - Used to **set** specific destination bits while **preserving** the others.
    - A **1 mask bit sets** the corresponding destination bit
    - A **0 mask bit preserves** the corresponding destination bit
- The **XOR** instruction
  - Used to **complement** specific destination bits while **preserving** the others.
    - A **1 mask bit complements** the corresponding destination bit
    - A **0 mask bit preserves** the corresponding destination bit

The **AND, OR and XOR** Instructions

- Syntax
  - AND destination, source
  - OR destination, source
  - XOR destination, source

- The **result** of the operation is **stored in the destination**
- **Destination** must be a **register or memory location**
- **Source** may be a **constant, register or memory location**
- **Memory to memory** operations are **not allowed**
- **Has effects on flags**

The **NOT** Instruction

- Syntax
  - NOT destination
- Perform the one's complement on the destination
- The result of the operation is stored in the destination
- Destination must be a register or memory location
- **There is no effect on the status flags**
- Complement the bits of a register or memory location

The **TEST** Instruction

- Syntax
  - TEST destination, source

- The **TEST instruction** performs an **AND operation** of the destination with the source but **does not change the destination contents.**

- The **purpose** of the TEST instruction is to **set the status flags.**

Shift Instructions

- The shift instructions shift the bits in the destination operand by one or more positions either to the left or right.
- For a shift instruction, the bits shifted out are lost.
- Syntax
  - opcode destination, 1
  - opcode destination, CL
- In both cases, destination is an 8 or 16-bit register or memory location.
- For intel's more advanced processors, a shift instruction also allows the use of an 8-bit constant.

The SHL Instruction

- The SH**L**(**shift left**) instruction shifts the bits in the destination to the left.
- Syntax
  - SHL destination, 1 ;for a single shift
- **A 0 is shifted into the rightmost bit position and the MSB is shifted into CF.**
- Effect on flags
  - SF, PF, ZF reflect the result
  - CF= last bit shifted out
  - OF= 1 if result changes sign on last shift
- The SHL instruction on a binary number doubles the value

The SAL Instruction

- The opcode SA**L** (**shift arithmetic left**) is often used in instances where numeric multiplication is intended.
- SAL instructions generate the same machine code as SHL instruction.
- Negative numbers can also be multiplied by powers of 2 by left shifts
- For example, if AX is FFFFh (-1), then shifting three times will yield AX= FFF8h (-8).

Overflow

- When we treat left shifts as multiplication, overflow may occur.
- For a single left shift, CF and OF accurately indicate unsigned and signed overflow, respectively.
- **But the overflow flags are not reliable indicators for a multiple left shift.**
- This is because a multiple shift is really a series of single shifts, and CF, OF only reflect the result of the last shift.

The SHR Instruction

- The instruction SH**R** (**shift right**) performs right shifts on the destination operand.
- Syntax
  - SHR destination, 1
  - SHR destination, CL
- A **0 is shifted into the MSB position**, and the **rightmost bit is shifted into CF.**
- Effect on flags
  - SF, PF, ZF reflect the result
  - CF= last bit shifted out
  - OF= 1 if result changes sign on last shift
- **The SHR instruction on a binary number halves the value if it is an even number. For odd numbers, a right shift halves it and rounds down to the nearest integer.**

The SAR Instruction

- The SAR Instruction (shift arithmetic right) operates like SHR
- The MSB retains its original value.
- Syntax
  - SAR destination, 1
  - SAR destination, CL
- Effect on flags
  - SF, PF, ZF reflect the result
  - CF= last bit shifted out
  - OF= 1 if result changes sign on last shift

Rotate Instructions

- The **rotate instructions** rotate the bits in the destination operand by **one or more positions** either to the **left or right.**
- For a rotate instruction, bits shifted out from one end of the operand are put back into the other end.
- Syntax
  - opcode destination, 1
  - opcode destination, CL
- In both cases, destination is an 8 or 16-bit register or memory location.
- For intel's more advanced processors, a rotate instruction also allows the use of an 8-bit constant.

The ROL Instruction

- The instruction RO**L** (**rotate left**) shifts bits to the left.
- The MSB is shifted into the rightmost bit.
- The CF also gets the bit shifted out of the MSB.
- Destination bits forming a circle, with the least significant bit following the MSB in the circle.
- Syntax
  - ROL destination, 1
  - ROL destination, CL

- **In ROL, CF reflects the bit that is rotated out. This can be used to inspect the bits in a byte or word without changing the contents.**

The ROR Instruction

- The instruction **ROR** (**rotate right**) shifts bits to the right.
- The **rightmost bit** is shifted into the **MSB** and **also into the CF.**
- Syntax
  - ROR destination, 1
  - ROR destination, CL

- **In ROR, CF reflects the bit that is rotated out. This can be used to inspect the bits in a byte or word without changing the contents.**

Example

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

```
XOR    AX, AX        ;AX counts bits
MOV    CX, 16        ;loop counter
TOP:
ROL    BX,1          ;CF=bit rotated out
JNC    NEXT          ;0 bit
INC    AX            ;1 bit, increment total
NEXT:
LOOP   TOP           ;loop until done
```

The RCL Instruction

- The instruction **RCL** (**Rotate through Carry Left**) shifts the bits of the destination to the left.
- The MSB is shifted into the CF, and the previous value of CF is shifted into the rightmost bit.
- RCL works like ROL, except that CF is part of the circle of bits being rotated.
- Syntax
  - RCL destination, 1
  - RCL destination, CL
- Effect on the flags
  - SF, PF, ZF reflect the result
  - CF = last bit shifted out
  - OF = 1 if result changes sign in the last rotation

The RCR Instruction

- The instruction **RCR** (**Rotate through Carry Right**) shifts the bits of the destination to the right.
- The **LSB** is shifted into the **CF**, and the **previous value of CF** is shifted into the **leftmost bit**.
- **RCR works like ROR, except that CF is part of the circle of bits being rotated**.
- Syntax
  - RCR destination, 1
  - RCR destination, CL
- Effect on the flags
  - SF, PF, ZF reflect the result
  - CF = last bit shifted out
  - OF = 1 if result changes sign in the last rotation