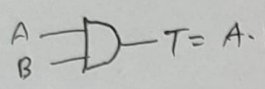
CSE 331/243/23.03.2024/

Basic Logic Instructions

- For all logic instruction, flag bits changed and carry and overflow flags became 0.
- all operation done by bit wise. And bit indexed from left side, start with 0.

AND



⇒ logical multiplication

⇒ in the truth table of AND, most of the output are zero. So, we can say that AND works as a clear operation.

- used to clear a part of a number, which is known as mask.
- in 8086, AND operation takes about 1 microsecond.
- used any mode except memory to memory and segment register.
- ASCII number can be converted to BCD using this

mask ⇒ 0000 1111

AND AX, BX ← mask

⇒ AX = AX · BX

AX = XXXX XXXX

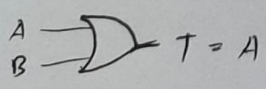
BX = 0000 1111

⇒

AX = 0000 XXXX

C = a = 0 = 0 (flags)

OR



⇒ logical addition and inclusive OR

⇒ as maximum output is 1, it is known as set operation.

⇒ Restriction : segment register addressing

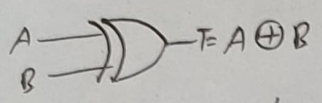
OR AX, BX ← mask

⇒ AX = AX + BX

| AX = XXXX XXXX
| BX = 0000 1111
| ⇒
| AX = XXXX 1111

C = 0 = a = 0

XOR



⇒ exclusive - OR

↳ $\bar{A}B + A\bar{B}$

⇒ same input = 0

different input = 1

- work for toggle, inverse or 1st compliment.
- sometime called a comparator

⇒ Restriction: segment register addressing

- common use is to clear a register to zero

XOR AX, BX ← mask

⇒ AX = AX ⊕ BX

| AX = XXXX XXXX
| BX = 0000 1111
| ⇒
| AX = XXXX $\bar{X}\bar{X}\bar{X}\bar{X}$

TEST

- works as same as CMP
- performs AND operation
- used conditional jump ⇒ JZ or JNZ
- second operand, source operand is ~~imme~~ ~~ram~~ immediate data.

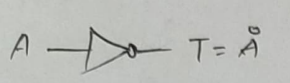
TEST AX, 05

change Z according to (AX · 05)

AX = XXXX XXXX
0000 0101
⇒
0000 0X0X

Z will change according to these X.

NOT



⇒ logical inversion

- invent all bits of the given operand.
- only one operand
- Restriction: Segment RA

NOT AX

⇒ AX = $\bar{AX}$

AX = xxxx xxxx

⇒

AX = $\bar{x}\bar{x}\bar{x}\bar{x}$ $\bar{x}\bar{x}\bar{x}\bar{x}$

NEG

⇒ arithmetic inversion
or
sign inversion.

⇒ convert positive number to negative
and negative to positive

- Do the 2's compliment operation.
- only one operand
- Restriction : segment RA

NEG AX

⇒ AX = $\bar{AX}$ + 1

AX = 0000 0101 ⇒ +5

⇒

AX = 1111 1011 ⇒ -5

Slide Page - 52

Bit Scan Instruction: only available in 80386 and onward

⇒ BSF ⇒ Bit Scan Forward
- Left to Right

⇒ BSR ⇒ Bit Scan Reverse
- Right to left

⇒ for both Z flag is set by 1.

BSF EBX, EAX
(EBX: Destination, EAX: where to scan)

⇒
EBX = index of 1st 1 encountered in EAX from left to Right scan

EAX = 60000000 H

0110 0000 0000

→ Scan → 1st 1 encountered

index = 30 (from Right)

↳ EBX = 0000001E

BSR EBX, EAX
(EBX: Destination, EAX: where to scan)

EAX = 00000005 H

0000 ..... 0000 0101

Scan ← 1st 1 encountered at index 0

↳ EBX = 00000000 H

Shift and Rotate

⇒ performs bit wise operation

⇒ used to control I/O Devices

⇒ two operand.

Destination operand ⇒ the content we want to shift or rotate

Source operand ⇒ immediate number or CL
↳ can't use other registers
- it will count the shift or rotate bit.

Shift:

Right Shift ⟶ 0010 ⇒ 2 ⇒ 2^1

0100 ⇒ 4 ⇒ 2^2

2^3 ← 8 ← 1000 ⟵ Left shift

arithmetic shift

⇒ 2^{n+1} ⟵ Left ⟵ 2^n ⟶ Right ⟶ 2^{n-1}

↙ multiplication operation by 2

↘ Divisional operation by 2

⇒ Shift

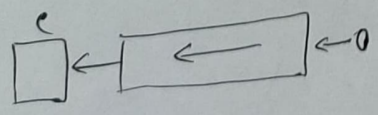
- Logical Shift
  - Right ⇒ SHR
  - Left ⇒ SHL
- Arithmetic Shift
  - Right ⇒ SAR
  - Left ⇒ SAL

AX, 1 → shift bit count

↓ Destination

According to the diagram, slide- 60

SHL AX, 1



AX = 1010 1100 0011 0110 ←

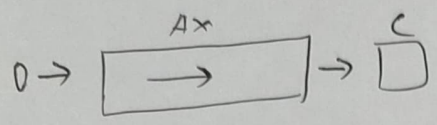
C = 0

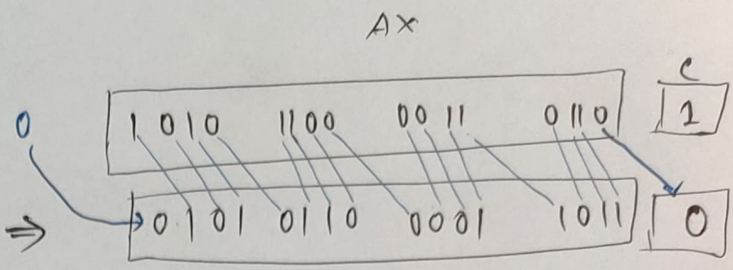
⇒

C = 1

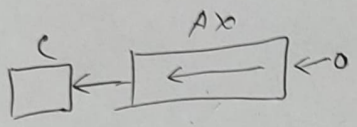
AX = 0101 1000 0110 1100

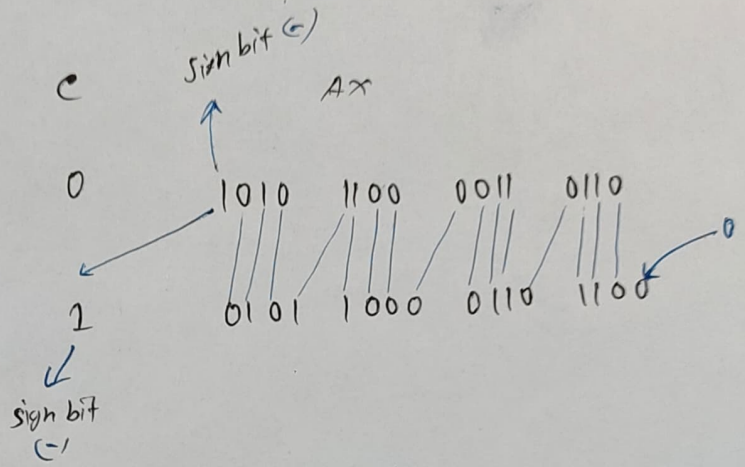
⊛ SHR AX, 1



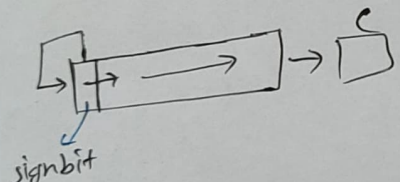


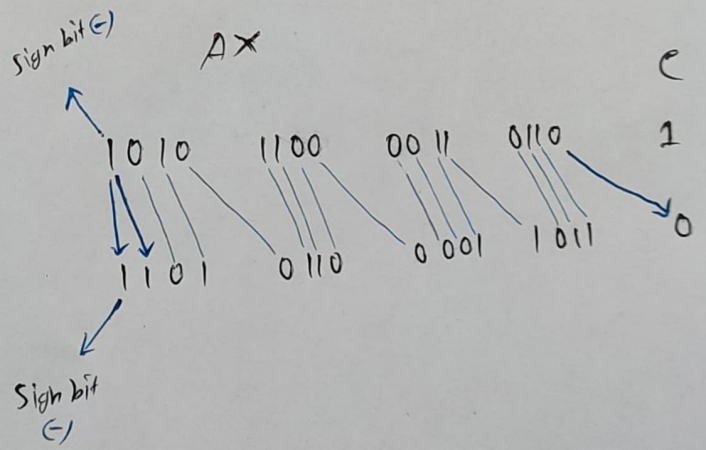
⊛ SAL AX, 1





⊛ SAR AX, 1

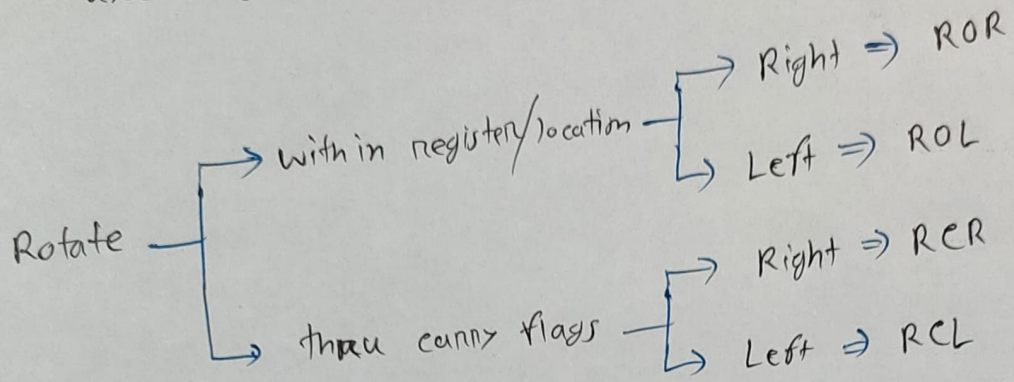




Slide - 62

Rotate

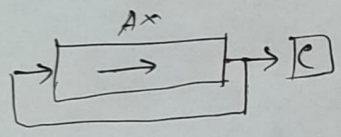
- used to shift wide numbers to left or right.

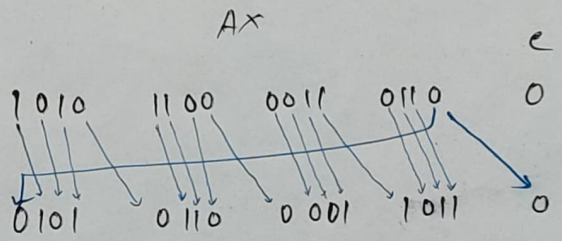


Slide - Page - 66

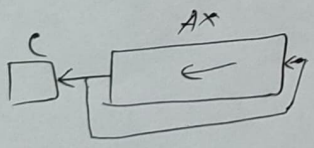
according to diagram, slide - 64

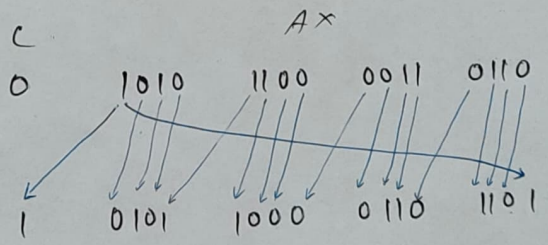
ROR AX, 1



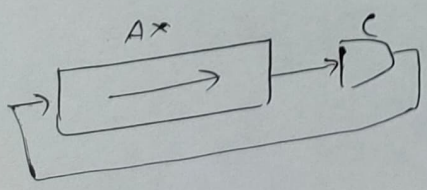


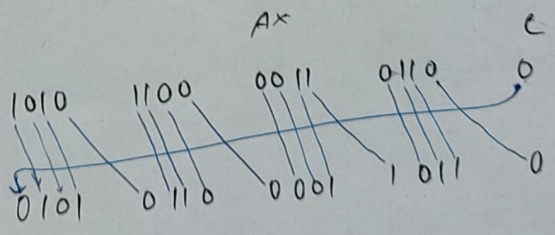
ROL AX, 1



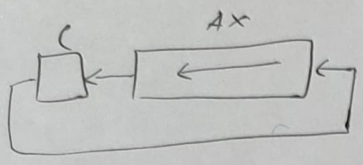


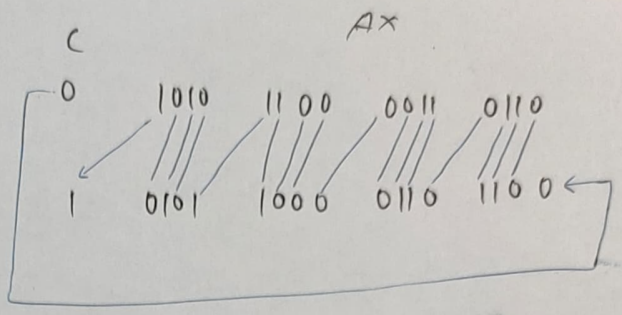
RCR AX, 1





⊛ RCL AX, 1





most clean visualization

⊛ Special case for rotate:

⇒ if Register is 16 bit and you are asked to rotate it 16 bit either Right or left (without carry) then the output will be same as before.

ROL SI, 16
⇒ SI = SI

ROR SI, 16
⇒ SI = SI

Lets say you are asked to rotate 20 bit, when register is 16 bit.
Then, find out remainder.
(if Rotate count > Register bit)

16) 20 (1
16
4

that means,
ROL SI, 20 = ROL SI, 4
ROR SI, 20 = ROR SI, 4