

Logical Instructions:

Logical operations:

AND -> ANL A, #10101111B

OR -> ORL A, #11111001B

XOR -> XRL A, #10000000B

Complement -> CPL A

Clear -> CLR A

ROTATE -> RL A, RLC A, RR A, RRC A

SWAP

Clear bit with ANL instruction:

; Let clear a given bit

; suppose we want to clear bit0

; and bit7 of accumulator

MASK: EQU 01111110B

MOV A,#10101011B

ANL A,#MASK; A = ?

; We must have '0' at bit position

; being to be cleared!

SET bit with ORL instruction:

; Let set a given bit

; suppose we want to set bit

; 0,1,2,3 of accumulator

MASK: EQU 00001111B

MOV A,#10110010B

ORL A,#MASK; A = ?

; We must have '1' at bit position

; being to be set!

Print BCD to screen:

PrintBCD:

PUSH ACC

ANL A,#0F0H ; A = 30H

SWAP A ; A = 03H

ADD A,#'0'

CALL COUT

POP ACC

Print BCD to screen:

```
POP ACC  
ANL A,#0FH  
ADD A,#'0'  
CALL COUT  
RET
```

```
MAIN: MOV A,#39H  
CALL PrintBCD
```

Print HEX to screen:

```
; suppose A = 1AH  
; first we must split A into  
; 01 and 0A  
; Then we must convert 01 to '1'  
; and 0A to 'A'  
; Then send '1' and 'A' to screen  
; with CALL cout
```

Print HEX to screen:

```
; the first job must do  
; beforehand is to convert 4-bit  
; into ASCII code, says  
; 0-9 to 30H - 39H  
; a-f to 41H - 46H  
; suppose we will use addition to  
; help convert them.
```

Print HEX to screen:

```
; for value 0-9, different = ?  
; 30H - 0 = 30H  
; 31H - 1 = 31H  
; ...  
; 39H - 9 = 39H  
; So we can simply add it with  
; 30H, right?
```

Print HEX to screen:

```
; for value a-f, different = ?  
; 41H - 0aH = ?  
; 42H - 0bH = ?  
; ...  
; 46H - 0fH = ?  
; So we can simply add it with  
; XX, right?
```

Print HEX to screen:

```
; subroutine convert 4-bit to  
; ASCII code  
; entry: A = 0-F  
; exit: A  
convertHEX: MOV B,A ; save A  
             SUBB A,#0AH  
             JNC  AtoF ; >=  
             ADD  A,#30H ; <  
             RET
```

Print HEX to screen:

```
AtoF:      MOV A,B ; restore A
           ADD A,#37H
           RET
```

```
; let's test above subroutine
; with single stepping!
```

```
Mian:      MOV A,#1
           CALL convertHEX
; change accumulator A and see..
```

Print HEX to screen:

```
; now ready to print?  
; not yet..  
; we need more subroutine to make  
; life more easy.
```

```
printHEX: PUSH ACC  
          ANL A,#0F0H  
          SWAP A  
          CALL convertHEX
```

Print HEX to screen:

```
CALL COUT  
POP ACC  
ANL A,#0FH  
CALL convertHEX  
CALL COUT  
RET
```

```
MAIN: MOV A,#0EFH  
CALL printHEX
```