1)Add 16

```
.model small
 .data
 a dw 1234H
 b dw 0100H
 .code
      mov      ax, @data      ; Initialize data section
      mov      ds, ax
      mov      ax, a          ; Load number1 in ax
      mov      bx, b          ; Load number2 in bx
      add      ax, bx         ; add numbers. Result in ax
      mov     ch, 04h          ; Count of digits to be displayed
      mov      cl, 04h        ; Count to roll by 4 bits
      mov      bx, ax         ; Result in reg bh
 l2:    rol      bx, cl      ; roll bl so that msb comes to lsb
      mov      dl, bl         ; load dl with data to be displayed
      and      dl, 0fH        ; get only lsb
      cmp      dl, 09         ; check if digit is 0-9 or letter A-F
      jbe     l4
      add      dl, 07         ; if letter add 37H else only add 30H
 l4:    add      dl, 30H
      mov      ah, 02         ; Function 2 under INT 21H (Display character)
      int      21H
      dec      ch             ; Decrement Count
      jnz      l2
      mov      ah, 4cH        ; Terminate Program
      int      21H
end
```

2)Count 0’s and 1’s

```
.model small
.data
n1 db 31h
zeros db 1 dup(0)
ones db 1 dup(0)
.code
Start:
mov ax,@data
mov ds,ax
mov cl,08h
mov ah,00h
mov al,n1
mov dx,0000h
up: rcl al,01H
```

```
JC next
inc dl
jmp down
next: inc dh
down: loop up
mov zeros, dh
mov ones,dl
int 03H
end Start
```

3)Even odd

```
.model small
.data
array db 12h, 23h, 26h, 63h, 25h, 36h, 2fh, 33h, 10h, 35h
.code
start: MOV ax,@data
MOV ds,ax
MOV cl,10
MOV SI,2000h
MOV DI,2008h
LEA BP,array
back: MOV AL,DS:[BP]
MOV BL,AL
AND AL,01H
JZ next
MOV [DI],bl
INC DI
JMP skip
next: MOV [SI],bl
INC SI
skip: INC BP
DEC CL
JNZ back
int 03H
end start
```

4)Factorial

```
.model small
.data
num dw 05h
.code
MOV ax, @data ; initialize the data segment
MOV ds, ax
MOV ax, 01 ; initialize ax = 1
MOV bx, num ; load the number in cx
CALL fact ; call procedure
MOV di, ax ; store lsb of result in di
MOV bp, 2 ; initialize count for no of times display is called
MOV bx, dx ; store msb of result in reg bx
MOV bx, di ; store lsb of result in bx
DEC bp ; decrement bp
MOV ah,4ch
Int 21h
Fact proc near ;function for finding the factorial
CMP bx,01 ;if bx=1
JZ l11 ;if yes ax=1
l12: MUL bx ;find factorial
DEC bx ; decrement bx
CMP bx,01 ;multiply bx=1
JNE l12
RET
l11:MOV ax,01 ;initialize ax=1
RET ;return to called program
fact ENDP ;end procedure
END ;end program
```

5)Move to other memory location

```
.model small
.code
start:
MOV SI,2000h
MOV DI,4000h
MOV CL,05h
LOOP1: MOV bl,[SI]
MOV [DI],bl
INC SI
INC DI
DEC CL
JNZ LOOP1
int 03h
end start
```

6)Palindrome

```
.MODEL SMALL
.STACK 100H
.DATA ; The string to be printed
STRING DB 'madam', '$'
STRING1 DB 'String is palindrome', '$'
```

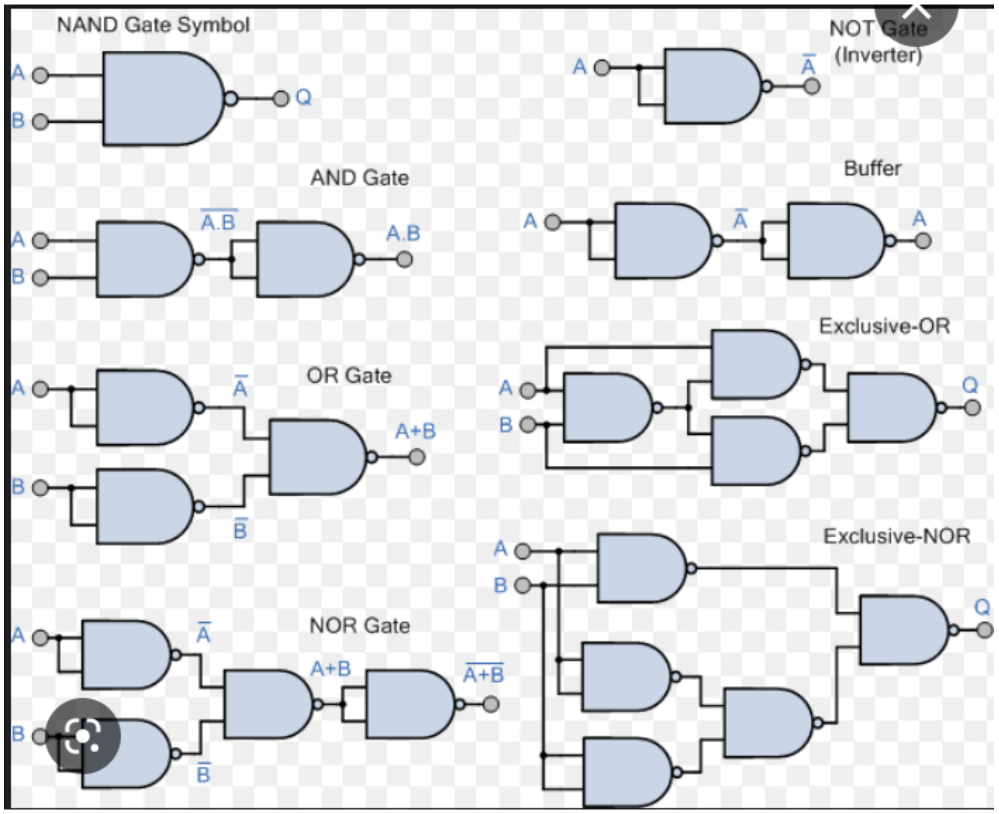
```
STRING2 DB 'String is not palindrome', '$'
.CODE
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX
; check if the string is;
;palindrome or not
CALL Palindrome
;interrupt to exit
MOV AH, 4CH
INT 21H
MAIN ENDP
Palindrome PROC
; load the starting address
; of the string
MOV SI, OFFSET STRING
; traverse to the end of;
;the string
LOOP1 :
MOV AX, [SI]
CMP AL, '$'
JE LABEL1
INC SI
JMP LOOP1
;load the starting address;
;of the string
LABEL1 :
MOV DI, OFFSET STRING
DEC SI
; check if the string is palindrome;
;or not
LOOP2 :
CMP SI, DI
JL OUTPUT1
MOV AX, [SI]
MOV BX, [DI]
CMP AL, BL
JNE OUTPUT2
DEC SI
INC DI
JMP LOOP2
OUTPUT1:
;load address of the string
LEA DX,STRING1
```

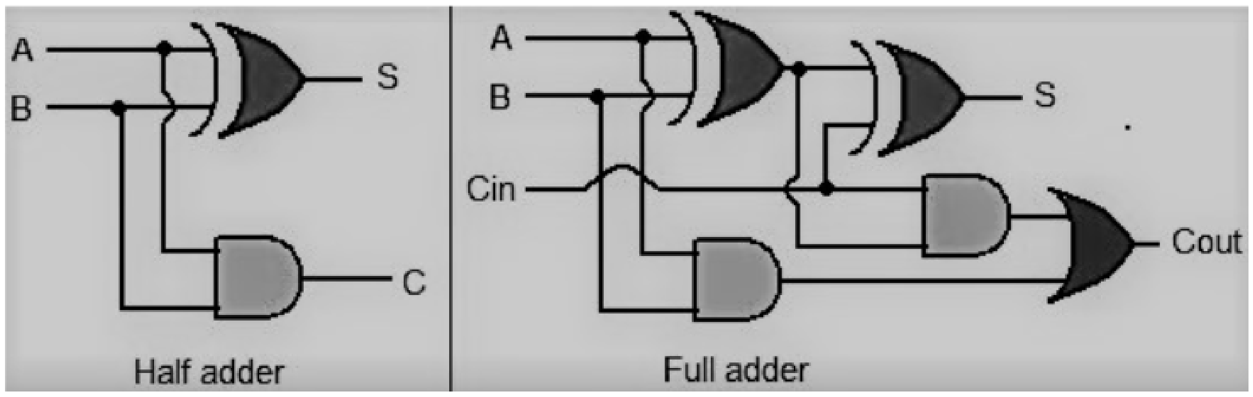
```
; output the string;
;loaded in dx
MOV AH, 09H
INT 21H
RET
OUTPUT2:
;load address of the string
LEA DX,STRING2
; output the string
; loaded in dx
MOV AH,09H
INT 21H
RET
Palindrome ENDP
END MAIN
```

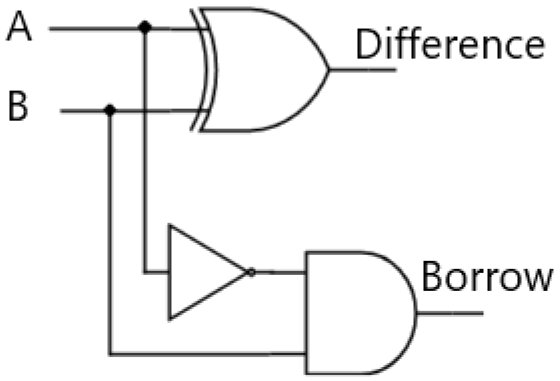
7)Pack unpack

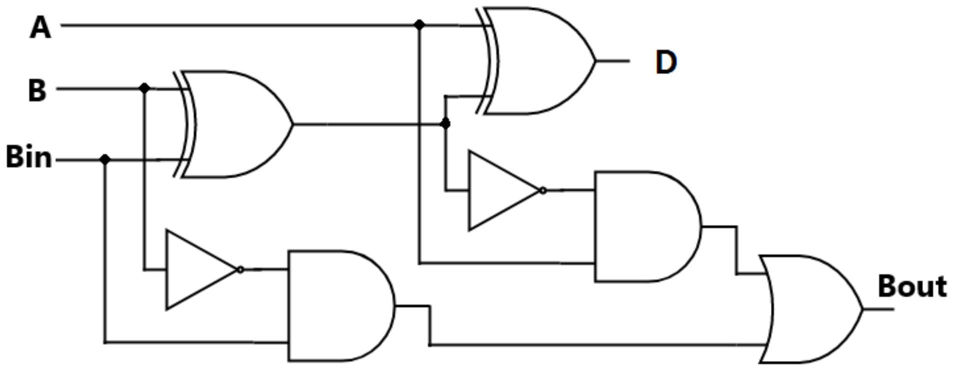
```
.model small
.data
a db 96H
.code
mov ax,@data
mov ds,ax
mov al,a
and al,0f0h
mov cl,04h
rcr al,cl
mov bh,al
call disp
mov al,a
and al,0fh
mov bh,al
call disp
mov ah,4cH
int 21H
disp proc near
mov ch,02h
mov cl,04h
l2: rol bh,cl
mov dl,bh
and dl, 0fH
cmp dl,09
jbe l4
add dl,07
```

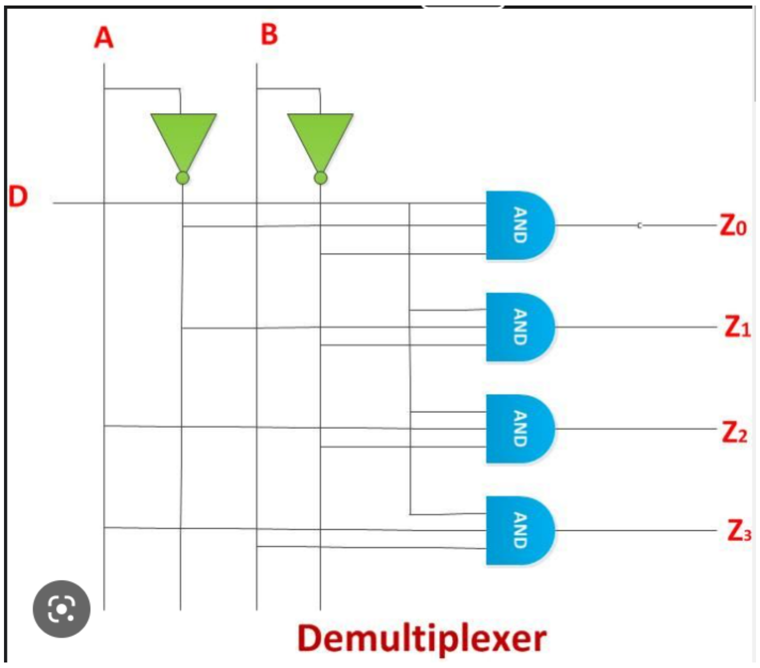
```
l4: add dl,30H
mov ah,02
int 21H
dec ch
jnz l2
mov ah,02h
mov dl,''
int 21h
endp
ret
end
```





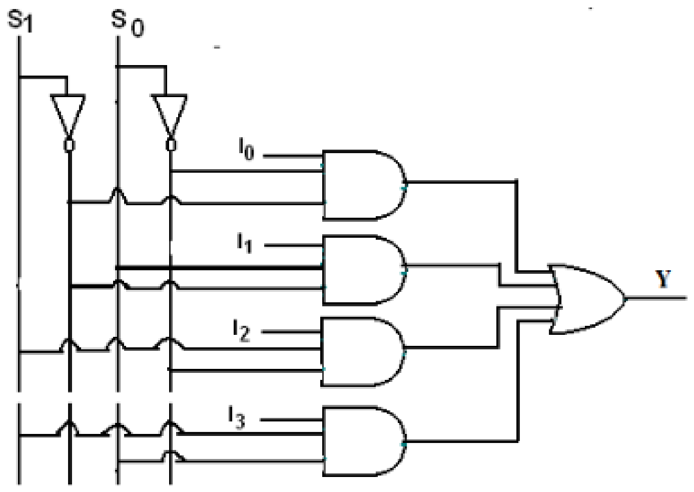

A
B
Difference
Borrow

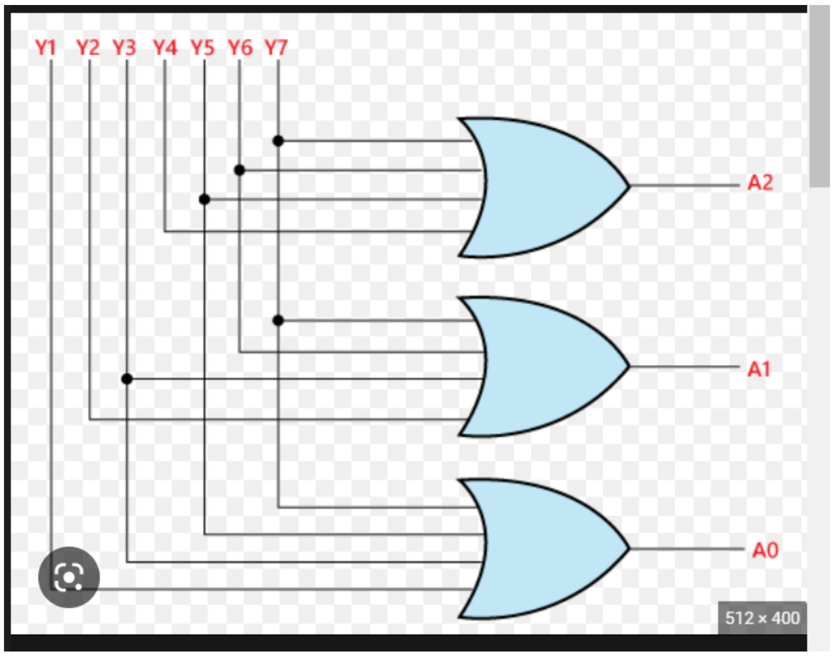

A
B
Bin
D
Bout


A
B
D
AND
AND
AND
AND
Z0
Z1
Z2
Z3
Demultiplexer

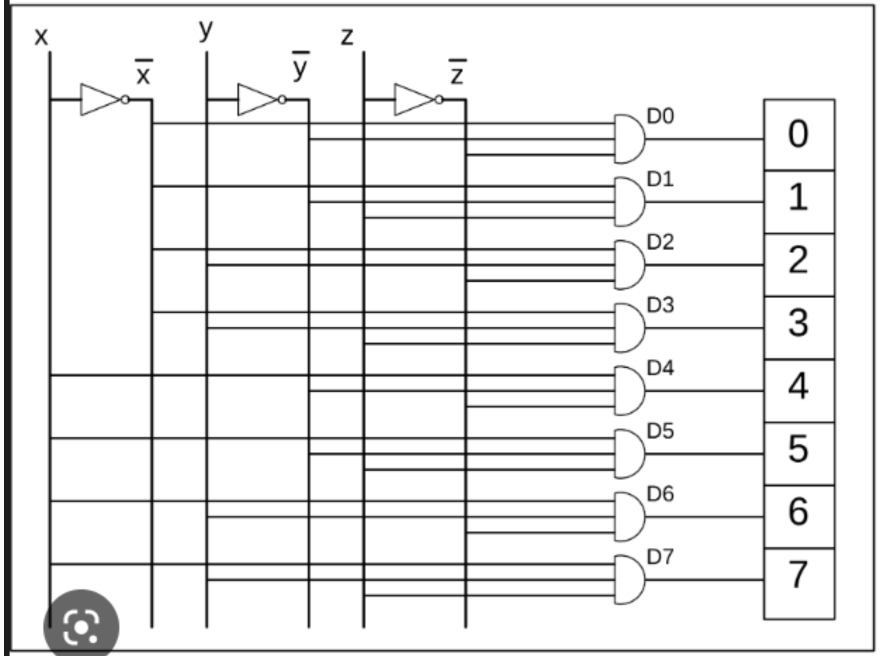
Input	S1	S0	Y
I_0	0	0	I_0
I_1	0	1	I_1
I_2	1	0	I_2
I_3	1	1	I_3

$$Y = S_1 S_0 I_3 + S_1 \overline{S_0} I_2 + \overline{S_1} S_0 I_1 + \overline{S_1}\,\overline{S_0} I_0$$





3 x 8 line Decoder Logic Diagram



electroniclinic.com