1. *ALP for addition of two 8 bit numbers using indirect addressing mode*

```
data segment
        first db ?
        second db ?
        sum db 1 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov al,first
        mov bl,second
        add al,bl
        mov sum,al
        code ends
        end start
```

2. *ALP for addition of two 16 bit numbers using indirect addressing mode*

```
data segment
        first dw ?
        second dw ?
        sum dw 2 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov ax,first
        mov bx,second
        add ax,bx
        mov sum,ax
        code ends
        end start
```

3. *ALP for subtraction of two 8 bit numbers using indirect addressing mode*

```
data segment
        first db ?
        second db ?
        diff db 1 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov al,first
        mov bl,second
        sub al,bl
        mov diff,al
        code ends
        end start
```

4. ALP for subtraction of two 16 bit numbers using indirect addressing mode

```
data segment
        first dw ?
        second dw ?
        diff dw 2 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov ax,first
        mov bx,second
        sub ax,bx
        mov diff,ax
        code ends
        end start
```

5. ALP for multiplication of two 8 bit numbers using indirect addressing mode

```
data segment
        first db ?
        second db ?
        pro dw 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov al,first
        mov bl,second
        mul bl
        mov pro,ax
        code ends
        end start
```

6. ALP for multiplication of two 16 bit numbers using indirect addressing mode

```
data segment
        first dw ?
        second dw ?
        pro dw 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov ax,first
        mov bx,second
        mul bx
        mov pro,ax
        mov pro+2,dx
        code ends
        end start
```

7. ALP for addition of two 8 bit numbers with carry using indirect addressing mode

```
data segment
        first db ?
        second db ?
        sum db 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov al,first
        mov bl,second
        add al,bl
        mov sum,al
        mov al,00h
        adc al,00h
        mov sum+1,al
        code ends
        end start
```

8. ALP for subtraction of two 16 bit numbers using indirect addressing mode

```
data segment
        first db ?
        second db ?
        diff db 2 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov al,first
        mov bl,second
        sub al,bl
        mov diff,al
        mov al,00h
        sbb al,00h
        mov diff+1,al
        code ends
        end start
```

9. ALP for finding the factorial of an given 8 bit number using indirect addressing mode

```
data segment
        first db ?
        sum dw 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov cl,first
        mov al,01H
l:
        mul cx
        dec cl
        JNZ l
        mov sum,ax
        code ends
        end start
```

10. ALP for finding the 1's complement of an given 8 bit number using indirect addressing mode

```
data segment
        number db ?
        compliment db 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov ah,number
        xor ah,0ffh
        mov compliment, ah
        code ends
        end start
```

11. ALP for finding the 2's complement of an given 8 bit number using indirect addressing mode

```
data segment
        number db ?
        compliment db 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov ah,number
        xor ah,0ffh
        add ah,01h
        mov compliment, ah
        code ends
        end start
```

12. ALP for finding the sum of squares of given 8 bit number using indirect addressing mode

```
data segment
        first dw ?
        sum dw 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov bx,0000
        mov cx,first
        mov si,0010h
        l:
        mov ax,cx
        mul cx
        add bx,ax
        dec cx
        JNZ l
        mov [si],bx
        mov [si+2],dx
        code ends
        end start
```

13. ALP for finding the sum of cubes of given 8 bit number using indirect addressing mode

```
data segment
        first dw ?
        sum dw 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov bx,0000
        mov cx,first
        mov si,0010h
        l:
        mov ax,cx
        mul cx
        mul cx
        add bx,ax
        dec cx
        JNZ l
        mov [si],bx
        mov [si+2],dx
        code ends
        end start
```

14. ALP for finding division of given 8 bit number using indirect addressing mode

```
data segment
        first db ?
        second db ?
        quotient db 1 dup(0)
        remainder db 1 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov al,first
        mov bl,second
        div bl
        mov quotient,al
        mov remainder,ah
        code ends
        end start
```

15. ALP for division of given 16 bit number with 8 bit number using indirect addressing mode

```
data segment
        first dw ?
        second db ?
        quotient db 1 dup(0)
        remainder db 1 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov ax,first
        mov cl,second
        div cl
        mov quotient,al
        mov remainder,ah
        code ends
        end start
```

16. ALP for arithmetic mean of 8-bit number using indirect addressing mode

```
data segment
        first dw ?
        mean dw 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov ax,0000h
        mov cx,first
        mov bx,first
   up:
        add ax,cx
        dec cl
        jnz up
        div bx
        mov mean,ax
        code ends
        end start
```

17. ALP for finding Fibonacci series using direct addressing mode

```
code segment
        assume cs:code
start:
        mov si,0000h
        mov cx,0005h
        mov ax,0001h
        mov bx,0000h
   l:
        add ax,bx
        mov [si],bx
        mov bx,ax
        mov ax,[si]
        inc si
        dec cx
        jnz l
        code ends
        end start
```

18. ALP for swapping of two 16bit numbers using indirect addressing mode

```
data segment
        first dw ?
        second dw ?
        swapp dw 2 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov ax,first
        mov bx,second
        xchg ax,bx
        mov swapp,ax
        mov swapp+2,bx
        code ends
        end start
```

19. ALP for fabonacci series using indirect addressing mode

```
data segment
        first db ?
        data ends
code segment
        assume cs:code,ds:data
start:
        mov si,0001h
        mov cl,first
        mov al,01h
        mov bl,00h
   l:
        add al,bl
        mov [si],bl
        mov bl,al
        mov al,[si]
        inc si
        dec cl
        jnz l
        code ends
        end start
```

20. ALP for finding largest number using indirect addressing mode

```
data segment
        count db ?
        data ends
code segment
        assume cs:code,ds:data
start:
        mov cl,count
        mov si,0001h
        mov al,[si]
        dec cl
    l2: inc si
        cmp al,[si]
        jnb l1
        mov al,[si]
    l1: loop l2
        inc si
        mov [si],al
        code ends
        end start
```

21. ALP for finding largest number using indirect addressing mode

```
data segment
        count db ?
        data ends
code segment
        assume cs:code,ds:data
start:
        mov bx,0000h
        mov dx,0000h
        mov cl,count
        mov si,0001h
    l3: mov al,[si]
        ror al,01h
        jc l1
        inc bl
        jmp l2
    l1: inc dl
    l2: inc si
        dec cl
        jnz l3
        mov [si],bl
        mov [si+1],dl
        code ends
        end start
```

22. ALP for a block of data using indirect addressing mode

```
data segment
        data ends
code segment
        assume cs:code,ds:data
start:
        mov si,0000h
        mov di,0010h
        mov cl,04
    l:  mov al,[si]
        mov [di],al
        inc si
        inc di
        dec cx
        jnz l
        code ends
        end start
```

23. ALP for converting BCD number Hexadecimal using indirect addressing mode

```
data segment
        number db ?
        res dw 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov al,number
        and al,0fh
        mov ah,number
        rol ah,04h
        and ah,0fh
        aad
        mov res,ax
        code ends
        end start
```

24. ALP to arrange the given numbers in an ascending order

```
data segment
     array db 5 dup(0)
     data ends
code segment
     assume cs:code,ds:data
start:
      mov cl,05h
  L1: mov si, 0000h
      mov bl,cl
      dec bl
  L2:  mov al,array[si]
      inc si
      cmp al,array[si]
      jb L3
      xchg al,array[si]
      mov array[si-1],al
  L3:  dec bl
      cmp bl,0000h
      jne L2
      loop L1
      code ends
      end start
```

25. ALP to arrange the given numbers in an descending order

```
data segment
     array db 5 dup(0)
     data ends
code segment
     assume cs:code,ds:data
start:
      mov cl,05h
  L1: mov si,0000h
      mov bl,cl
      dec bl
  L2: mov al,array[si]
      inc si
      cmp al,array[si]
      ja L3
      xchg al,array[si]
      mov array[si-1],al
  L3:  dec bl
      cmp bl,0000h
      jne L2
      loop L1
      code ends
     end start
```

26. ALP to convert Packed BCD numbers to ASCII

```
data segment
        number db ?
        result db 2 dup(0)
        data ends
code segment
        assume cs:code, ds:data
start:
        mov al, number
        rol al,04h
        and al,0fh
        or al,30h
        mov result,al
        mov al,number
        and al,0fh
        or al,30h
        mov result+1,al
        code ends
        end start
```

27. ALP to convert Packed BCD numbers to unpacked BCD numbers

```
data segment
        number db ?
        result db 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov al,number
        rol al,04h
        and al,0fh
        mov result,al
        mov al,number
        and al,0fh
        mov result+1,al
        code ends
        end start
```

27. ALP for counting number of positive and negative numbers

```
data segment
        count db ?
        pos db 1 dup(0)
        neg db 1 dup(0)
        data ends
code segment
        assume cs:code,ds:data
start:
        mov bl,00h
        mov dl,00h
        mov cl,count
        mov si,0001h
        l3:mov al,[si]
        rol al,01h
        jc l1
        inc bl
        jmp l2
        l1:inc dl
        l2: inc si
        dec cl
        jnz l3
        mov pos,bl
        mov neg,dl
        code ends
        end start
```