

```
;Addition of two numbers  
include 'emu8086.inc'  
mov ax,12  
mov bx,6  
add ax,bx
```

```
; Compare and Jump instruction
```

```
include 'emu8086.inc'  
MOU Ax,16  
mov bx,16  
cmp ax,bx  
je 11  
cmp ax,bx  
jne 12
```

```
11:
```

```
    print 'Equal'  
    jmp exit
```

```
12:
```

```
    print 'Not equal'  
    jmp exit
```

```
exit:
```

```
ret
```

# ; Armstrong Number

```
include 'emu8086.inc'
printn "enter any number:"
call scan_num
mov ax,cx
mov var,ax
mov vari,ax

f1:
    mov ax,var
    mov bx,10
    div bx

    mov var,ax
    mov ax,dx
    mov bx,dx
    mul bx
    mul bx
    add sum,ax
    cmp var,0
    je f2
    loop f1
f2:
    mov ax,sum
    cmp ax,vari
    je f3
    print "it is not amstrong"
    jmp exit
f3:
    print "it is amstrong"
exit:

ret
sum dw 0
var dw 0
vari dw 0
define_print_num
define_print_num_uns
define_scan_num
```

```
01  
02  
03 ;Binary Number  
04  
05 include 'emu8086.inc'  
06 print 'enter any number:'  
07 call scan_num  
08 mov ax,cx  
09 mov n1,ax  
10 mov cx,1  
11 mov bx,0  
12 mov n2,2  
13 mov n3,10  
14  
15 Binary:  
16     cmp ax,0  
17     je exit  
18     xor dx,dx  
19     div n2  
20     mov n1,ax  
21     mov ax,dx  
22     mul cx  
23     add bx,ax  
24     mov ax,cx  
25     mul n3  
26     mov cx,ax  
27     mov ax,n1  
28     jmp Binary  
29 exit:  
30     mov ax,bx  
31     call print_num  
32  
33  
34 ret  
35 n1 dw 0  
36 n2 dw 0  
37 n3 dw 0  
38 define_print_num  
39 define_print_num_uns  
40 define_scan_num
```

```
01  
02 ;Factorial of number|  
03  
04 include 'emu8086.inc'  
05 print "enter any number:"  
06 call scan_num  
07 mov ax,cx  
08 mov bx,ax  
09 dec bx  
10 fact:  
11     mul bx  
12     dec bx  
13     cmp bx,1  
14     jne fact  
15 printn ""  
16 call print_num_uns  
17 ret  
18 define_print_num_uns  
19 define_scan_num  
20  
21  
22  
23
```

```
01
02 ; N number of Fibanoci Series
03
04 include 'emu8086.inc'
05 org 100h
06 print 'enter any number:'
07 call scan_num
08 mov n1,cx
09 mov ax,0
10 printn ''
11 printn ''
12 print 'fibonacci series:'
13 call print_num
14 mov ax,1
15 print ', '
16 call print_num
17 mov dx,0
18 mov cx,2
19 fibanoci:
20     mov n2,ax
21     add ax,dx
22     print ', '
23     call print_num
24     mov dx,n2
25     inc cx
26     cmp cx,n1
27         je end
28         jmp fibanoci
29 end:
30
31 ret
32 n1 dw 0
33 n2 dw 0
34 define_print_num
35 define_print_num_uns
36 define_scan_num
```

```
03  
04 ;LCM of two numbers  
05  
06 include 'emu8086.inc'  
07 org 100h  
08 printn "enter first number:"  
09 call scan_num  
10 mov ax,cx  
11 mov n1,ax  
12 printn "enter second number:"  
13 call scan_num  
14 mov bx, cx  
15 mov cx,1  
16 cmp ax,bx  
17 jc swap  
18 lcm:  
19     div bx  
20     cmp dx,0  
21     je last  
22     inc cx  
23     mov ax,n1  
24     mul cx  
25     mov n2,ax  
26     jmp lcm  
27 swap:  
28     xchg ax,bx  
29     mov n1,ax  
30     jmp lcm  
31  
32 last:  
33     mov ax,n2  
34     call print_num  
35 ret  
36 n1 dw 0  
37 n2 dw 0  
38 define_print_num  
39 define_print_num_uns  
40 define_scan_num  
41  
42  
43
```

```
01  
02  
03 ;Checking whether given number Perfect number or NOT  
04  
05 include 'emu8086.inc'  
06 print 'enter any number :'  
07 call scan_num  
08  
09 mov ax,cx|  
10 mov cx,1  
11 mov bx,0  
12 mov n1,ax  
13 perfect_num_calc:  
14     xor dx,dx  
15     mov ax,n1  
16     cmp cx,n1  
17     je exit  
18     div cx  
19     cmp dx,0  
20     je addition  
21     inc cx  
22     jmp perfect_num_calc  
23 addition:  
24     add bx,cx  
25     inc cx  
26     jmp perfect_num_calc  
27  
28  
29 exit:  
30     cmp bx,n1  
31     je perfect  
32     mov ax,n1  
33     call print_num  
34     print 'It is not Perfect'  
35     ret  
36  
37 perfect:  
38     mov ax,n1  
39     call print_num  
40     print 'It is Perfect Number'  
41     ret  
42  
43 ret  
44 n1 dw 0  
45 define_print_num  
46 define_print_num_uns  
47 define_scan_num
```

```
83 ;Perfect Number
84
85 include 'emu8086.inc'
86 printn "enter any number:"
87 call scan_num
88 mov ax,cx
89 cmp ax,1
90     je exit1
91 mov n1,ax
92 mov cx,1
93 mov bx,0
94
95 factors:
96     xor dx,dx
97     div cx
98     cmp dx,0
99     je addition
100
101 prog:
102     inc cx
103     mov ax,n1
104     cmp cx,ax
105     je exit
106     jmp factors
107
108 addition:
109     add bx,cx
110     jmp prog
111
112 exit:
113     cmp bx,n1
114     je output
115     call print_num
116     printn " is not perfect number"
117     ret;
118
119 output:
120     call print_num
121     printn " is perfect number"
122
123 exit1:
124     print '1 is not perfect number'
125 ret
126 n1 dw 0
127 define_print_num
128 define_print_num_uns
129 define_scan_num
```

```
03 ;reverse of a number
04
05
06 include 'emu8086.inc'
07 print 'enter a number:'
08 call scan_num
09 mov [0100h],cx
10 mov bx,10
11 mov ax,0
12 reverse:
13     mul bx
14     mov [01000h],ax
15     mov ax,cx
16     div bx
17     mov cx,ax
18     add [01000h],dx
19     mov ax,[01000h]
20     cmp cx,0
21         jne reverse
22     cmp cx,0
23         je reverse1
24 reverse1:
25     print 'The reverse number:'
26     call print_num
27
28
29
30 ret
31 define_print_num
32 define_print_num_uns
33 define_scan_num
```

```
51 ;Pallindrom or NOT|
52
53
54 include 'emu8086.inc'
55 print ' enter any number :'
56 call scan_num
57 mov ax,cx
58 mov n3,ax
59 mov n1,ax
60 mov cx,10
61 mov bx,0
62
63 reverse:
64     xor dx,dx
65     cmp ax,0
66         je exit
67     div cx
68     mov n2,dx
69     mov n1,ax
70     mov ax,bx
71     mul cx
72     mov bx,ax
73     add bx,n2
74     mov ax,n1
75     jmp reverse
76
77 exit:
78     cmp bx,n3
79         je pallindrom
80     mov ax,n3
81     call print_num
82     print 'Is not Pallindrom'
83     ret
84
85 pallindrom:
86     mov ax,bx
87     call print_num
88     print ' It is Pallindrom'
89
90     ret
91 n1 dw 0
92 n2 dw 0
93 n3 dw 0
94 define_print_num
95 define_print_num_uns
96 define_scan_num
```

```
01 ;Prime or NOT
02
03 include 'emu8086.inc'
04 print ' enter any number:'
05 call scan_num
06 cmp cx,2
07 je is_prime
08 mov ax,cx
09 mov bx,2
10
11 prime_calculation:
12 xor dx,dx
13 div bx
14 cmp dx,0
15 je not_prime
16 inc bx
17 mov ax,cx
18 cmp bx,cx
19 je is_prime
20 jmp prime_calculation
21
22
23 is_prime:
24 printn ''
25 printn 'it is prime'
26 jmp exit
27
28 not_prime:
29 printn ''
30 print 'it is not prime'
31 jmp exit
32 exit:
33
34 ret
35 define_print_num
36 define_print_num_uns
37 define_scan_num
```

```
01  
02  
03 ;Odd or Even  
04  
05 include 'emu8086.inc'  
06 org 100h  
07  
08 print 'enter num'  
09 call scan_num  
10 mov ax,cx  
11 mov bx,2  
12 div bx  
13 cmp dx,0  
14  
15 je label  
16 print 'num is odd'  
17 jmp exit  
18  
19 label:  
20     print 'num is even'  
21 exit:  
22  
23 ret  
24  
25 define_scan_num  
26 define_print_num  
27 define_print_num_uns
```

```
01
02
03 ;Printing Array elements
04 include 'emu8086.inc'
05
06 lea si,array
07 printt:
08     mov al,[si]
09     print ;
10     call print_num
11     inc si
12     cmp [si],0
13     jne printt
14     cmp [si],0
15     je exit
16
17
18 exit:
19
20 ret
21 array db 10,20,30,40,0
22
23 define_print_num
24 define_print_num_uns
25 define_scan_num
```

```
01 ;Printing Array Elements|
02
03
04
05 include 'emu8086.inc'
06
07 mov al,[array+di]
08 cmp al,0
09 je empty
10 printt:
11 mov al,[array+di]
12 print ','
13 call print_num
14 inc di
15 cmp array[di],0
16 jne printt
17 cmp array[di],0
18 je exit
19 empty:
20 print 'No elements'
21 exit:
22
23 ret
24 array db 19,38,4,45,6,55,4,3,0
25
26 define_print_num
27 define_print_num_uns
28 define_scan_num
```

```
03  
04  
05 ; length of array  
06  
07 include 'emu8086.inc'  
08 mov cx,0  
09 lea si,array  
10 printt:  
11     inc cx  
12     inc si  
13     cmp [si],0  
14     jne printt  
15     cmp [si],0  
16     je exit  
17  
18 exit:  
19     mov ax,cx  
20     print 'the length of array:'  
21     call print_num  
22  
23  
24 ret  
25 array db 10,20,30,40,100,0  
26  
27 define_print_num  
28 define_print_num_uns  
29 define_scan_num
```

```
01
02
03 ;Minimum number in given Array
04
05
06 include 'emu8086.inc'
07
08 lea si,array
09 mov al,[si]
10 inc si
11 mov bl,[si]
12 printt:
13     cmp al,bl
14         jb al_below
15     cmp al,bl
16         jnb bl_below |
17
18 label12:
19     inc si
20     cmp [si],0
21         je minn
22     mov al,min
23     mov bl,[si]
24     jmp printt
25
26
27 al_below:
28     mov min,al
29     jmp label12
30
31
32 bl_below:
33     mov min,bl
34     jmp label12
35
36
37 minn:
38     mov al,min
39     call print_num
40
41 ret
42 array db 45,78,60,10,20,100,200,0
43 min db 0
44
45 define_print_num
46 define_print_num_uds
47 define_scan_num
```

```
01
02
03 ;Maximum number in given Array
04
05
06 include 'emu8086.inc'
07
08 lea si,array
09 mov al,[si]
10 inc si
11 mov bl,[si]
12 printt:
13     cmp al,bl
14     ja al_above
15     cmp al,bl
16     jna bl_above
17
18 labe12:
19     inc si
20     cmp [si],0
21     je maxx
22     mov al,max
23     mov bl,[si]
24     jmp printt
25
26
27 al_above:
28     mov max,al
29     jmp labe12
30
31
32 bl_above:
33     mov max,bl
34     jmp labe12
35
36
37 maxx:
38     mov al,max
39     print 'The maximum number:'
40     call print_num
41
42 ret
43 array db 45,78,60,10,20,100,200,0
44 max db 0
45
46 define_print_num
47 define_print_num_uns
48 define_scan_num
```

```

01 ;Array Sorting
02
03 include 'emu8086.inc'
04 mov si,-1
05
06 l1:
07     inc si
08     cmp si,9
09         je exit
10     mov bx,si
11     inc bx
12     mov di,bx
13     jmp l1
14 l2:
15     cmp di,9
16         je l1
17     mov al,array[si]
18     mov cl,array[di]
19     cmp al,cl
20         ja swap
21     inc di
22     jmp l2
23
24 swap:
25     mov array[si],cl
26     mov array[di],al
27     inc di
28     jmp l2
29
30 exit:
31
32     mov si,-1
33 printt:
34     inc si
35     cmp si,9
36         je exit1
37     mov al,array[si]
38     print ','
39     call print_num
40     jmp printt
41
42 exit1:
43
44 ret
45 array db 5,4,1,3,2,10,30,14,18
46
47 define_print_num
48 define_print_num_uns
49 define_scan_num

```

```
01  
02  
03 ; print fibanoci series up to given number  
04  
05 include 'emu8086.inc'  
06 org 100h  
07 print 'enter any number:'  
08 call scan_num  
09 mov n1,cx  
10 mov ax,0  
11 printn ''  
12 print 'fibanoci series:'  
13 call print_num  
14 mov ax,1  
15 print ','  
16 call print_num  
17 mov dx,0  
18 fibanoci:  
19     mov n2,ax  
20     add ax,dx  
21     print ','  
22     call print_num  
23     mov dx,n2  
24     cmp ax,cx  
25         je end  
26         jmp fibanoci  
27 end:  
28  
29 ret  
30 n1 dw 0  
31 n2 dw 0  
32 define_print_num  
33 define_print_num_uns  
34 define_scan_num
```

50 emulator screen (93x34 chars)

```
enter any number:13  
fibanoci series:0 1 1 2 3 5 8 13
```

```
01 ;gcd of two numbers|
02
03 include 'emu8086.inc'
04 mov cx,20
05 mov bx,30
06 cmp bx,cx
07 jl 11
08 xchg bx,cx
09
10 11:
11     mov n1,bx
12 12:
13     xor dx,dx
14     mov ax,bx
15     div n1
16     cmp dx,0
17     je sec
18 13:
19     dec n1
20     cmp n1,1
21     je ans
22     jmp 12
23 sec:
24     mov ax,cx
25     div n1
26     cmp dx,0
27     je ans
28     jmp 13
29 ans:
30     mov ax,n1
31     print "gcd of given numbers is : "
32     call print_num
33     ret
34 n1 dw 0
35 define_print_num
36 define_print_num_uns
```

```
01  
02 ;String Printing  
03  
04 include 'emu8086.inc'  
05 lea si,st  
06 call print_string  
07 ret  
08 st db 'rgukt_123',0  
09 define_print_string
```

```
01 ;prime factors|
02 include 'emu8086.inc'
03 print "enter your number : "
04 call scan_num
05 mov ax,cx
06 mov n1,ax
07 printn ""
08 print "prime factors :"
09 print " 1 "
10 cmp ax,1
11 je end
12 mov cx,2
13 did:
14     xor dx,dx
15     div cx
16     cmp dx,0
17     je lable1
18 incr:
19     cmp cx,n1
20     je end
21     mov ax,n1
22     inc cx
23     jmp did
24 lable1:
25     mov ax,cx
26     mov bx,2
27     mov ax,cx
28 did1:
29     cmp cx,bx
30     je printt
31     xor dx,dx
32     div bx
33     cmp dx,0
34     je incr
35     inc bx
36     mov ax,cx
37     jmp did1
38 printt:
39     mov ax,cx
40     call print_num
41     print " 0"
42     jmp incr
43 end:
44     ret
45 define_print_num
```

```
Q1 }
Q2 ;reverse of a string
Q3
Q4 include 'emu8086.inc'
Q5 mov ah,2
Q6 mov si,0
Q7 start:
Q8     cmp arr[si],'*'
Q9     je print
10     inc si
11     jmp start
12 print:
13     dec si
14 ans:
15     cmp si,-1
16     je end
17     mov dl,arr[si]
18     int 21h
19     dec si
20     jmp ans
21 end:
22 hlt
23 arr db 'rgukt_Basar *'
24
```

```
13  
14 ;negative or positive |  
15  
16 include 'emu8086.inc'  
17 org 100h  
18  
19 print "enter your number :"  
20 call scan_num  
21 shl cx,1  
22 jc less  
23 printn "  
24 print "its a positive number "  
25 ret  
26 less:  
27     printn "  
28     print "its negative number "  
29 ret  
30 define_print_num  
31 define_print_num_uns  
32 define_scan_num  
33
```

```
01 ;String Pallindrom
02
03 include 'emu8086.inc'
04 mov ah,2
05 mov si,0
06 mov di,0
07
08 start:
09     cmp arr[di],'*'
10    je prnt
11    inc di
12    jmp start
13 prnt:
14     dec di
15 ans:
16     mov al,arr[si]
17     mov bl,arr[di]
18     cmp si,di
19     je palin
20     cmp al,bl
21     jne end
22     inc si
23     dec di
24     jmp ans
25 end:
26     Print "its not,a palindrome"
27     ret
28 palin:
29     print "its a palindrome"
30 arr db 'RguktkgR*'
31 define_print_num
32 define_print_num_uns
```

```
01  
02 ;Armstrong number  
03  
04 include 'emu8086.inc'  
05 ;print "enter your number : "  
06 ;call scan_num  
07 mov cx,153  
08 mov ax,cx  
09 did:  
10 xor dx,dx  
11 div n1  
12 mov n3,ax  
13 mov n2,dx  
14 mov ax,dx  
15 mul n2  
16 mul n2  
17 add bx,ax  
18 mov ax,n3  
19 cmp ax,0  
20 jne did  
21 arm:  
22 cmp bx,cx  
23 jne notarm  
24 printn ''  
25 print "its a armstrong number "  
26 ret  
27 notarm:  
28 printn ''  
29 print 'its not a armstrong number '  
30 ret  
31 n1 dw 10  
32 n2 dw 0  
33 n3 dw 0  
34 ;define_scan_num
```

```
81
82 ;GCD of two numbers
83
84 include 'emu8086.inc'
85 printn 'enter first number:'
86 call scan_num
87 mov bx,cx
88 printn 'enter second number:'
89 call scan_num
90 printn ','
91 cmp bx,cx
92 jl l1
93 xchg bx,cx
94 l1:
95     mov n1,bx
96
97 l2:
98     xor dx,dx
99     mov ax,bx
100    div n1
101    cmp dx,0
102    je sec
103 l3:
104    dec n1
105    cmp n1,1
106    je ans
107    jmp l2
108
109 sec:
110     xor dx,dx
111     mov ax,cx
112     div n1
113     cmp dx,0
114     je ans
115     jmp l3
116
117 ans:
118     mov ax,n1
119     printn 'gcd of given numbers is :'
120     call print_num
121 ret
122 n1 dw 0
123 define_print_num
124 define_print_num_uns
125 define_scan_num
```

```
81 ; 2 nd question (a)
82 ;print prime numbers below 100
83 include 'emu8086.inc'
84 mov n1,100
85 mov dx,2
86 mov bx,2
87 cmp bx,dx
88 je 13
89
90 11:
91     mov cx,2
92     mov ax,bx
93     mov n2,ax
94     cmp bx,101
95     je exit
96
97 start:
98     xor dx,dx
99     mov ax,n2
100    div cx
101    cmp dx,0
102    je 12
103    inc cx
104    cmp cx,bx
105    je 13
106    jmp start
107
108 12:
109     inc bx
110     jmp 11
111 13:
112     mov ax,bx,
113     print
114     call print_num
115     inc bx
116     jmp 11
117 exit:
118     ret
119 ret
120 n1 dw 0
121 n2 dw 0
122 define_print_num
123 define_print_num_uns
124 define_scan_num
```

```
01 ; 2 nd question (a)
02 ; Find element in Array
03 include 'emu8086.inc'
04 mov si,0
05 print 'enter search element:'
06 call scan_num
07 mov bl,cl
08 start:
09     cmp si,10
10         je exit
11         mov al,array[si]
12         cmp al,bl
13         je found
14         inc si
15         jmp start
16
17
18 found:
19     inc si
20     mov ax,si
21     gotoxy 0,2
22     print 'Element found at position :'
23     call print_num
24     ret
25
26 exit:
27     gotoxy 0,2
28     print 'Element not found'
29     ret
30
31 ret
32 array db 7,4,93,74,98,3,8,54,78
33
34 define_print_num
35 define_print_num_uns
36 define_scan_num
```

```
02  
03 ; 2 nd question (a)  
04 ;print prime numbers below 100  
05 include 'emu8086.inc'  
06 mov n1,100  
07 mov dx,2  
08 mov bx,2  
09 cmp bx,dx  
10 je 13  
11  
12 11:  
13     mov cx,2  
14     mov ax,bx  
15     mov n2,ax  
16     cmp bx,101  
17     je exit  
18  
19 start:  
20     xor dx,dx  
21     mov ax,n2  
22     div cx  
23     cmp dx,0  
24     je 12  
25     inc cx  
26     cmp cx,bx  
27     je 13  
28     jmp start  
29  
30 12:  
31     inc bx  
32     jmp 11  
33 13:  
34     mov ax,bx,  
35     print  
36     call print_num  
37     inc bx  
38     jmp 11  
39 exit:  
40     ret  
41 ret  
42 n1 dw 0  
43 n2 dw 0  
44 define_print_num  
45 define_print_num_uns  
46 define_scan_num
```

```
01
02 ; 2 nd question (a)
03 ; Find element in Array
04 include 'emu8086.inc'
05 mov si,0
06 print 'enter search element:'
07 call scan_num
08 mov bl,cl
09 start:
10     cmp si,10
11     je exit
12     mov al, array[si]
13     cmp al,bl
14     je found
15     inc si
16     jmp start
17
18 found:
19     inc si
20     mov ax,si
21     gotoxy 0,2
22     print 'Element found at position :'
23     call print_num
24     ret
25
26 exit:
27     gotoxy 0,2
28     print 'Element not found'
29     ret
30
31 ret
32 array db 7,4,93,74,98,3,8,54,78
33
34 define_print_num
35 define_print_num_uns
36 define_scan_num
```

```
01  
02 ;String User Input|  
03  
04  
05  
06 include 'emu8086.inc'  
07 lea si,str  
08 mov ah,1  
09  
10 start:  
11     int 21h  
12     mov [si],al  
13     cmp [si],'0'  
14     je 12  
15     inc si  
16     jmp start  
17 12:  
18     lea si,str  
19     mov ah,2  
20     mov bx,0  
21     print ,  
22 13:  
23     mov dl,[si]  
24     cmp [si],'0'  
25     je exit  
26     int 21h  
27     inc si  
28     inc bx  
29     jmp 13  
30  
31  
32 exit:  
33     hlt  
34  
35  
36 ret  
37 define_print_num  
38 define_print_num_uns  
39 define_scan_num  
40 str db ""
```

```

02
03 ;print| perfect numbers below 500
04
05 include 'emu8086.inc'
06
07 mov cx,2
08
09 loop1:
10     mov bx,1
11     mov n1,cx
12     cmp cx,500
13     je exit
14 loop2:
15     xor dx,dx
16     mov ax,n1
17     div bx
18     cmp cx,bx
19     je checkPerfect
20     cmp dx,0
21     je factor
22     inc bx
23     jmp loop2
24
25 checkPerfect:
26     cmp bx,var
27     je print
28     inc cx
29     mov var,0
30     jmp loop1
31 factor:
32     add var,bx
33     inc bx
34     jmp loop2
35
36
37 print:
38     mov ax,bx
39     print ;
40     call print_num
41     inc cx
42     mov var,0
43     jmp loop1
44 exit:
45     ret
46 ret
47 n1    dw 0
48 var   dw 0
49 define_print_num
50 define_print_num_uns
51 define_scan_num

```

```
01
02
03
04 ;FInding No.of Owels in String
05
06 include 'emu8086.inc'
07 lea si,str
08 mov cl,0
09 start:
10     mov al,[si]
11     cmp al,'$'
12     je exit
13     cmp al,'a'
14     je Owe1
15     cmp al,'e'
16     je Owe1
17     cmp al,'i'
18     je Owe1
19     cmp al,'o'
20     je Owe1
21     cmp al,'u'
22     je Owe1
23     inc si
24     jmp start
25
26 Owe1:
27     inc cl
28     inc si
29     jmp start
30
31 exit:
32     mov al,cl
33     print 'Number of Owels:'
34     call print_num
35     ret
36
37 ret
38 str db 'abcdefghijklmnopqrstuvwxyz$'
39 define_print_num
40 define_print_num_uns
41 define_scan_num
```

```
03  
04  
05 ; length of array  
06  
07 include 'emu8086.inc'  
08 mov cx, 0  
09 lea si, array  
10 printt:  
11     inc cx  
12     inc si  
13     cmp [si], 0  
14     jne printt  
15     cmp [si], 0  
16     je exit  
17  
18 exit:  
19     mov ax, cx  
20     print 'the length of array:'  
21     call print_num  
22  
23  
24 ret  
25 array db 10, 20, 30, 40, 100, 0  
26  
27 define_print_num  
28 define_print_num_uns  
29 define_scan_num
```