```
;Addition of two numbers

include 'emu8086.inc'
mov ax,12
mov bx,6
add ax,bx
```

```
;Compare and Jump instruction

include 'emu8086.inc'
MOV Ax,16
mov bx,16
cmp ax,bx
    je l1
cmp ax,bx
    jne l2

l1:
    print 'Equal'
    jmp exit
l2:
    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 var1,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,var1
            je f3
      print "it is not amstrong"
      jmp exit
f3:
      print "it is amstrong"
exit:

ret
sum   dw 0
var   dw 0
var1  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
41
42
43
44
```

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

```
01
02
03  ; N number of Fibanoci Series
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  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
```

```
;LCM of two numbers

include 'emu8086.inc'
org 100h
printn "enter first number:"
call scan_num
mov ax,cx
mov n1,ax
printn "enter second number:"
call scan_num
mov bx, cx
mov cx,1
cmp ax,bx
jc swap
lcm:
    div bx
    cmp dx,0
    je last
    inc cx
    mov ax,n1
    mul cx
    mov n2,ax
    jmp lcm
swap:
    xchg ax,bx
    mov n1,ax
    jmp lcm

last:
    mov ax,n2
    call print_num
ret
n1 dw 0
n2 dw 0
define_print_num
define_print_num_uns
define_scan_num
```

```
;Checking whether given number Perfect number or NOT

include 'emu8086.inc'
print 'enter anu number :'
call scan_num

mov ax,cx
mov cx,1
mov bx,0
mov n1,ax
perfect_num_calc:
            xor dx,dx
            mov ax,n1
            cmp cx,n1
                je exit
            div cx
            cmp dx,0
                je addition
            inc cx
            jmp perfect_num_calc
addition:
          add bx,cx
          inc cx
          jmp perfect_num_calc


exit:
     cmp bx,n1
          je perfect
     mov ax,n1
     call print_num
     print 'It is not Perfect'
     ret

perfect:
        mov ax,n1
        call print_num
        print 'It is Perfect Number'
        ret

ret
n1 dw 0
define_print_num
define_print_num_uns
define_scan_num
```

```
;Perfect Number

include 'emu8086.inc'
printn "enter any number:"
call scan_num
mov ax,cx
cmp ax,1
    je exit1
mov n1,ax
mov cx,1
mov bx,0

factors:
    xor dx,dx
    div cx
    cmp dx,0
    je addition
prog:
    inc cx
    mov ax,n1
    cmp cx,ax
    je exit
    jmp factors
addition:
      add bx,cx
      jmp prog
exit:
    cmp bx,n1
    je output
    call print_num
    printn " is not perfect number"
    ret;
output:
    call print_num
    printn " is perfect number"

exit1:
     print '1 is not perfect number'
ret
n1 dw 0
define_print_num
define_print_num_uns
define_scan_num
```

```
;reverse of a number


include 'emu8086.inc'
print 'enter a number:'
call scan_num
mov [0100h],cx
mov bx,10
mov ax,0
reverse:
      mul bx
      mov [01000h],ax
      mov ax,cx
      div bx
      mov cx,ax
      add [01000h],dx
      mov ax,[01000h]
      cmp cx,0
            jne reverse
      cmp cx,0
            je reverse1
reverse1:
      print 'The reverse number:'
      call print_num


ret
define_print_num
define_print_num_uns
define_scan_num
```

```
;Pallindrom or NOT|

include 'emu8086.inc'
print ' enter any number :'
call scan_num
mov ax,cx
mov n3,ax
mov n1,ax
mov cx,10
mov bx,0

reverse:
      xor dx,dx
      cmp ax,0
          je exit
      div cx
      mov n2,dx
      mov n1,ax
      mov ax,bx
      mul cx
      mov bx,ax
      add bx,n2
      mov ax,n1
      jmp reverse

exit:
      cmp bx,n3
          je pollindrom
      mov ax,n3
      call print_num
      print 'Is not Pallindrom'
      ret

pollindrom:
          mov ax,bx
          call print_num
          print ' It is Pallindrom'

ret
n1 dw 0
n2 dw 0
n3 dw 0
define_print_num
define_print_num_uns
define_scan_num
```

```
;Prime or NOT

include 'emu8086.inc'
print ' enter any number:'
call scan_num
cmp cx,2
    je is_prime
mov ax,cx
mov bx,2
prime_calculation:
                xor dx,dx
                div bx
                cmp dx,0
                  je not_prime
                inc bx
                mov ax,cx
                cmp bx,cx
                  je is_prime
                jmp prime_calculation


is_prime:
          printn ' '
          printn 'it is prime'
          jmp exit

not_prime:
          printn ' '
          print 'it is not prime'
          jmp exit
exit:

ret
define_print_num
define_print_num_uns
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
05  include 'emu8086.inc'
06
07  lea si,array
08  printt:
09        mov al,[si]
10        print ' '
11        call print_num
12        inc si
13        cmp [si],0
14              jne printt
15        cmp [si],0
16              je exit
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
02  ;Printing Array Elements
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
```

```
;Minimum number in given Array

include 'emu8086.inc'

lea si,array
mov al,[si]
inc si
mov bl,[si]
printt:
    cmp al,bl
        jb al_below
    cmp al,bl
        jnb bl_below

label2:
    inc si
    cmp [si],0
        je minn
    mov al,min
    mov bl,[si]
    jmp printt


al_below:
    mov min,al
    jmp label2


bl_below:
    mov min,bl
    jmp label2


minn:
    mov al,min
    call print_num

ret
array   db  45,78,60,10,20,100,200,0
min     db  0

define_print_num
define_print_num_uns
define_scan_num
```

```
;Maximum number in given Array

include 'emu8086.inc'

lea si,array
mov al,[si]
inc si
mov bl,[si]
printt:
      cmp al,bl
            ja al_above
      cmp al,bl
            jna bl_above

label2:
      inc si
      cmp [si],0
            je maxx
      mov al,max
      mov bl,[si]
      jmp printt


al_above:
      mov max,al
      jmp label2


bl_above:
      mov max,bl
      jmp label2


maxx:
      mov al,max
      print 'The maximum number:'
      call print_num

ret
array   db   45,78,60,10,20,100,200,0
max     db   0

define_print_num
define_print_num_uns
define_scan_num
```

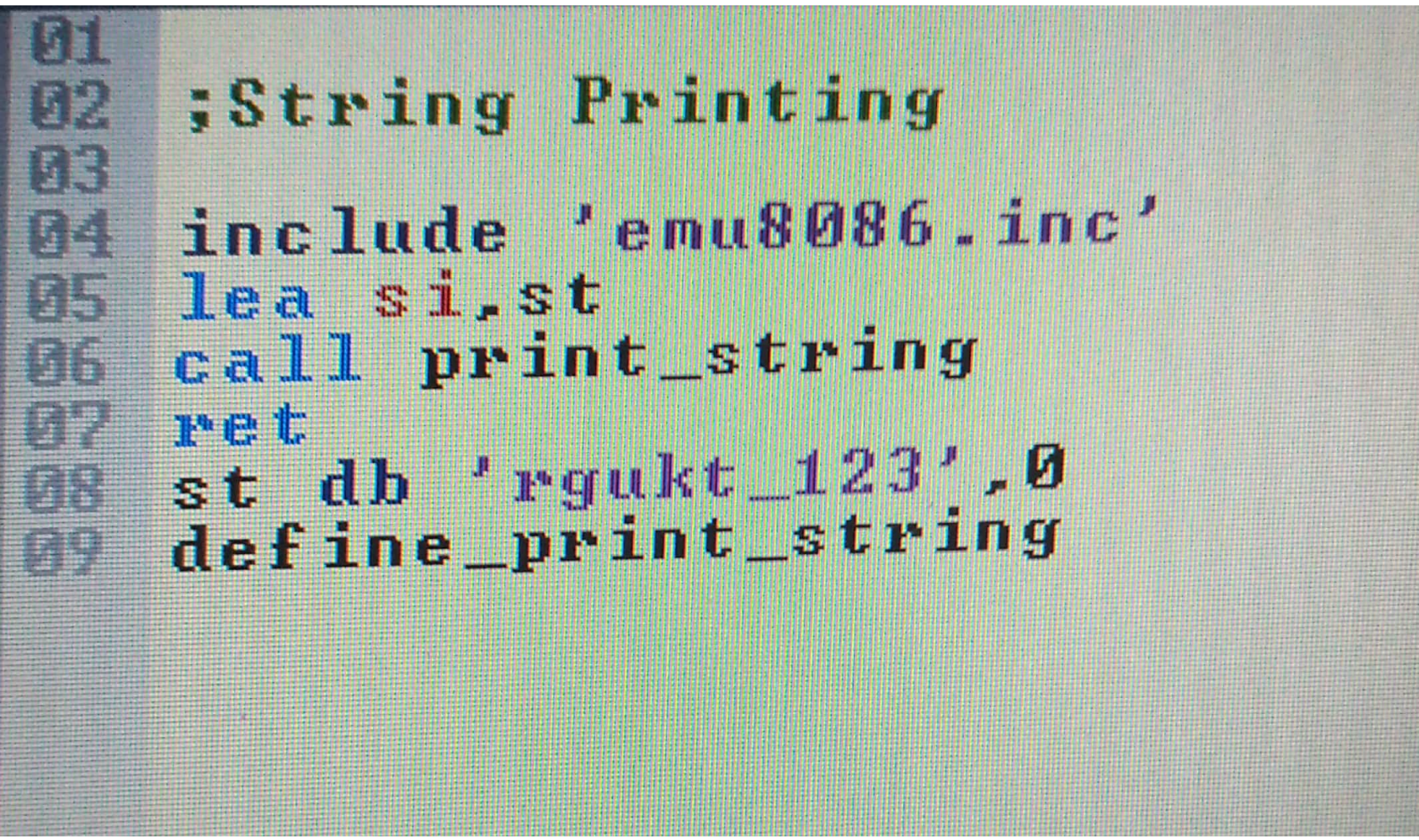
```
01
02  ;Array Sorting
03
04  include 'emu8086.inc'
05  mov si,-1
06  l1:
07        inc si
08        cmp si,9
09            je exit
10        mov bx,si
11        inc bx
12        mov di,bx
13        jmp l2
14  l2:
15        cmp di,9
16              je l1
17        mov al,array[si]
18        mov cl,array[di]
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
```

emulator screen (93x34 chars)

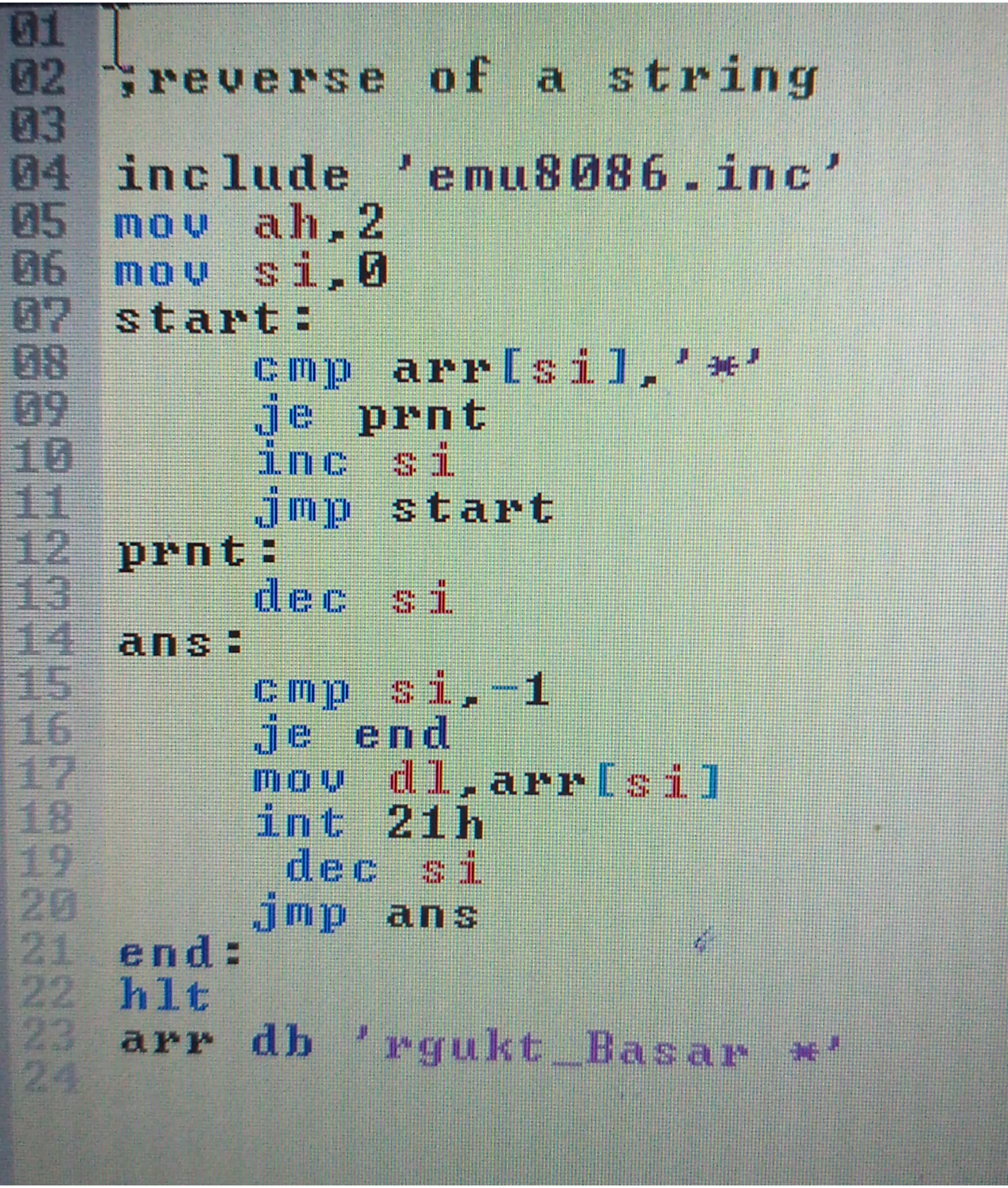
```
enter any number:13
fibanoci series:0  1  1  2  3  5  8  13
```

```
01
02  ;gcd of two numbers
03
04  include 'emu8086.inc'
05  mov cx,20
06  mov bx,30
07  cmp bx,cx
08  jl l1
09  xchg bx,cx
10  l1:
11      mov n1,bx
12  l2:
13      xor dx,dx
14      mov ax,bx
15      div n1
16      cmp dx,0
17      je sec
18  l3:
19      dec n1
20      cmp n1,1
21      je ans
22      jmp l2
23  sec:
24      mov ax,cx
25      div n1
26      cmp dx,0
27      je ans
28      jmp l3
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
```

```
;prime factors
include 'emu8086.inc'
print "enter your number : "
call scan_num
mov ax,cx
mov n1,ax
printn ""
print "prime factors :"
print " 1  "
cmp ax,1
je end
mov cx,2
did:
    xor dx,dx
    div cx
    cmp dx,0
    je lable1
incr:
    cmp cx,n1
    je end
    mov ax,n1
    inc cx
    jmp did
lable1:
    mov ax,cx
    mov bx,2
    mov ax,cx
did1:
    cmp cx,bx
    je printt
    xor dx,dx
    div bx
    cmp dx,0
    je incr
    inc bx
    mov ax,cx
    jmp did1
printt:
    mov ax,cx
    call print_num
     print "  "
    jmp incr
end:
    ret
define_print_num
```



```
;reverse of a string

include 'emu8086.inc'
mov ah,2
mov si,0
start:
      cmp arr[si],'*'
      je prnt
      inc si
      jmp start
prnt:
      dec si
ans:
      cmp si,-1
      je end
      mov dl,arr[si]
      int 21h
       dec si
      jmp ans
end:
hlt
arr db 'rgukt_Basar *'
```

```
14
15  ;negative or positive |
16
17  include 'emu8086.inc'
18  org 100h
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
```

```
;String Pallindrom

include 'emu8086.inc'
mov ah,2
mov si,0
mov di,0
start:
      cmp arr[di],'*'
      je prnt
      inc di
      jmp start
prnt:
      dec di
ans:
      mov al,arr[si]
      mov bl,arr[di]
      cmp si,di
      je palin
      cmp al,bl
      jne end
      inc si
      dec di
      jmp ans
end:
      Print "its not a palindrome"
      ret
palin:
      print "its a palindrome"
arr db 'RguktkugR*'
define_print_num
define_print_num_uns
```

```
;Armstrong number

include 'emu8086.inc'
;print "enter your number : "
;call scan_num
mov cx,153
mov ax,cx
did:
      xor dx,dx
      div n1
      mov n3,ax
      mov n2,dx
      mov ax,dx
      mul n2
      mul n2
      add bx,ax
      mov ax,n3
      cmp ax,0
      jne did
arm:
      cmp bx,cx
      jne notarm
      printn ''
      print "its a armstrong number "
ret
notarm:
      printn ''
      print 'its not a armstrong number '
ret
n1 dw 10
n2 dw 0
n3 dw 0
;define_scan_num
```

```
;GCD of two numbers

include 'emu8086.inc'
printn 'enter first number:'
call scan_num
mov bx,cx
printn 'enter second number:'
call scan_num
printn ' '
cmp bx,cx
jl l1
xchg bx,cx
l1:
    mov n1,bx

l2:
      xor dx,dx
      mov ax,bx
      div n1
      cmp dx,0
      je sec
l3:
    dec n1
    cmp n1,1
    je ans
    jmp l2

sec:
      xor dx,dx
      mov ax,cx
      div n1
      cmp dx,0
      je ans
      jmp l3

ans:
      mov ax,n1
      printn 'gcd of given numbers is :'
      call print_num
ret
n1 dw  0
define_print_num
define_print_num_uns
define_scan_num
```

```
; 2 nd qustion (a)
;print prime numbers below  100
 include 'emu8086.inc'
mov n1,100
mov dx,2
mov bx,2
cmp bx,dx
     je l3

l1:
     mov cx,2
     mov ax,bx
     mov n2,ax
     cmp bx,101
        je exit

start:
       xor dx,dx
       mov ax,n2
       div cx
       cmp dx,0
         je l2
       inc cx
       cmp cx,bx
            je l3
       jmp start

l2:
     inc bx
     jmp l1
l3:
    mov ax,bx
    print ' '
    call print_num
    inc bx
    jmp l1
exit:
     ret
ret
n1   dw   0
n2   dw   0
define_print_num
define_print_num_uns
define_scan_num
```

```
01
02  ; 2 nd qustion (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
```

```
02
03  ; 2 nd qustion (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 l3
11
12  l1:
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 l2
25           inc cx
26           cmp cx,bx
27                 je l3
28           jmp start
29
30  l2:
31       inc bx
32       jmp l1
33  l3:
34      mov ax,bx
35      print ' '
36      call print_num
37      inc bx
38      jmp l1
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 qustion (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 posit-ion :'
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
03  ;String User Input|
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  l2
15              inc si
16              jmp start
17      l2:
18              lea si,str
19              mov ah,2
20              mov bx,0
21              print '   '
22      l3:
23              mov dl,[si]
24              cmp [si],'0'
25                      je exit
26              int 21h
27              inc si
28              inc bx
29              jmp l3
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 """
41
```

```
;print perfect numbers below 500

include 'emu8086.inc'

mov cx,2

loop1:
        mov bx,1
        mov n1,cx
        cmp cx,500
          je exit
loop2:
        xor dx,dx
        mov ax,n1
        div bx
        cmp cx,bx
              je checkPerfect
        cmp dx,0
          je factor
        inc bx
        jmp loop2

checkPerfect:
                cmp bx,var
                    je print
                inc cx
                mov var,0
                jmp loop1
factor:
        add var,bx
        inc bx
        jmp loop2


print:
          mov ax,bx
          print ' '
          call print_num
          inc cx
          mov var,0
          jmp loop1
exit:
    ret
ret
n1      dw    0
var     dw    0
define_print_num
define_print_num_uns
define_scan_num
```

```
;Finding No.of Owels in String

include 'emu8086.inc'
lea si,str
mov cl,0
start:
          mov al,[si]
          cmp al,'$'
               je exit
          cmp al,'a'
               je Owel
          cmp al,'e'
               je Owel
          cmp al,'i'
               je Owel
          cmp al,'o'
               je Owel
          cmp al,'u'
               je Owel
          inc si
          jmp start

Owel:
     inc cl
     inc si
     jmp start

exit:
     mov al,cl
     print 'Number of Owels:'
     call print_num
     ret

ret
str  db 'abcdefghijklmnopqrstuvwxyz$'
define_print_num
define_print_num_uns
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
```