.model small

.stack 100h

.data

giris\_msg db 'Bir sayi giriniz (1-255): $'

sonuc\_toplam db 13,10, 'Asal Sayilarin Toplami: $'

sonuc\_adet db 13,10, 'Asal Sayi Adedi : $'

sayi\_str db 4 dup(0)

toplam\_str db 6 dup(0), '$'

adet\_str db 6 dup(0), '$'

girilen\_sayi dw ?

asal\_toplam dw 0

asal\_adet dw 0

.code

start:

mov ax, @data

mov ds, ax

*; Kullanıcıdan sayı iste*

mov ah, 09h

mov dx, offset giris\_msg

int 21h

*; Sayı girişi ayarı*

mov byte ptr [sayi\_str], 3 *; max karakter uzunluğu*

lea dx, sayi\_str

mov ah, 0Ah

int 21h

*; ASCII -> sayı çevirme*

xor ax, ax

xor bx, bx

lea si, sayi\_str + 2

mov cl, byte ptr [sayi\_str+1]

convert\_loop:

mov bl, byte ptr [si]

sub bl, '0'

mov dx, ax

mov ax, 10

mul dx

add ax, bx

inc si

loop convert\_loop

mov girilen\_sayi, ax

*; Asal kontrol döngüsü*

mov cx, 2

next\_number:

mov bx, cx

call is\_prime

cmp ax, 1

jne not\_prime

*; Asal ise topla, say*

inc asal\_adet

add asal\_toplam, cx

not\_prime:

inc cx

mov ax, girilen\_sayi

cmp cx, ax

jbe next\_number

*; Toplamı stringe çevir*

mov ax, asal\_toplam

lea di, toplam\_str

call int\_to\_str

*; Adedi stringe çevir*

mov ax, asal\_adet

lea di, adet\_str

call int\_to\_str

*; Sonuçları yazdır*

mov ah, 09h

mov dx, offset sonuc\_toplam

int 21h

mov ah, 09h

mov dx, offset toplam\_str

int 21h

mov ah, 09h

mov dx, offset sonuc\_adet

int 21h

mov ah, 09h

mov dx, offset adet\_str

int 21h

*; Programdan çık*

mov ah, 4Ch

int 21h

*; ----------------------------------------*

*; Asal kontrol fonksiyonu*

*; Giriş: BX = sayı, Çıkış: AX = 1 (asal), 0 (asal değil)*

is\_prime:

push cx

push dx

push si

mov ax, bx

cmp ax, 2

je prime\_yes

mov cx, 2

check\_loop:

mov dx, 0

div cx

cmp dx, 0

je prime\_no

inc cx

mov ax, bx

cmp cx, ax

jl check\_loop

prime\_yes:

mov ax, 1

jmp is\_end

prime\_no:

xor ax, ax

is\_end:

pop si

pop dx

pop cx

ret

*; ----------------------------------------*

*; AX -> DI'ye decimal string dönüşümü*

int\_to\_str:

push ax

push bx

push cx

push dx

xor cx, cx

mov bx, 10

reverse\_loop:

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax

jnz reverse\_loop

print\_loop:

pop dx

mov byte ptr [di], dl

inc di

loop print\_loop

mov byte ptr [di], '$'

pop dx

pop cx

pop bx

pop ax

ret

end start