

Bölüm 8: Makrolar

Mikroişlemciler

Makrolar



- Prosedürlere benzer.
- Derleme aşamasında gerçek komutlar ile değiştirilen geçici yapılardır.
- Kod içerisinde kullanılmayan makrolar görmezden gelinir.
- Makrolar, prosedürlerden farklı olarak,
 - kullanıldıkları kodun üzerinde tanımlanmalıdır.





Örnek Kod Parçası



```
MyMacro MACRO p1, p2, p3

MOV AX, p1

MOV BX, p2

MOV CX, p3
```

MOV AX, 00001h
MOV BX, 00002h
MOV CX, 00003h
MOV AX, 00004h
MOV BX, 00005h
MOV CX, DX

```
ORG 100h
```

ENDM

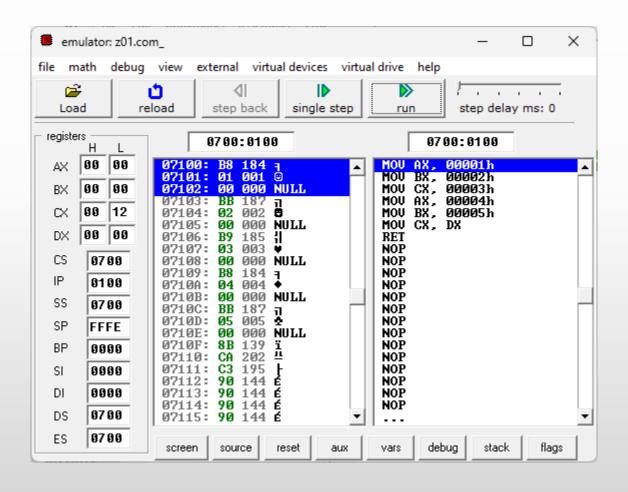
MyMacro 1, 2, 3
MyMacro 4, 5, DX
RET

; Makro kullanımı

; Başka bir örnek

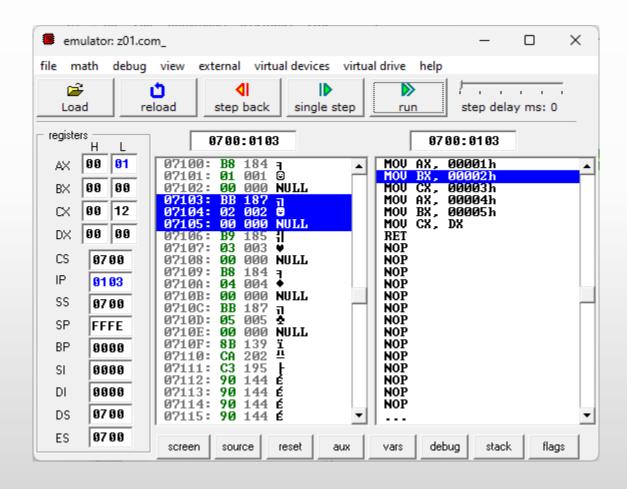






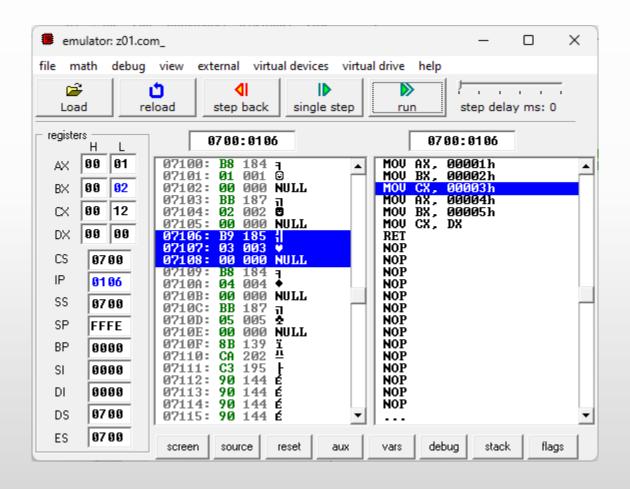






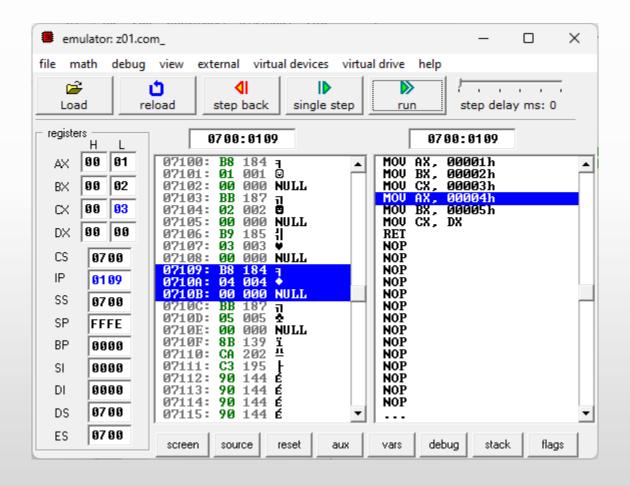






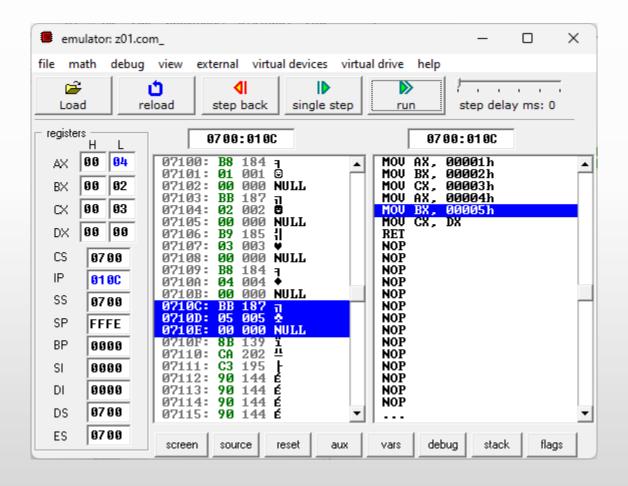






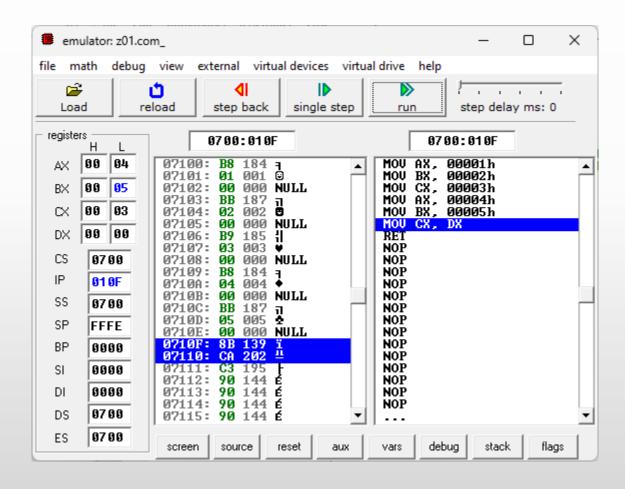






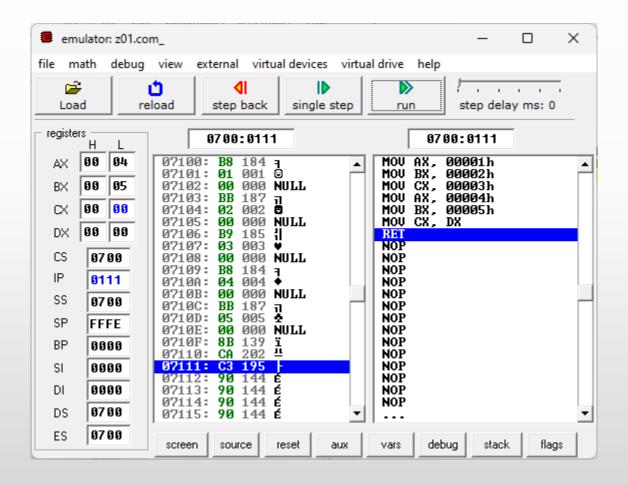












Prosedür Özellikler



- Prosedür CALL komutu ile çağrılır.
- Prosedür bir bellek adresinde bulunur.
- Çağrıldığında CPU kontrolü bu bölgeye aktarır.
- RET komutu ile kontrol, programın ana kısmına geri döner.
- Dönüş adresini saklamak için yığın kullanılır.
- Aynı prosedür 100 kez çağrılırsa dosyanın boyutunu çok az arttırır.
 - CALL komutu 3 byte yer kaplar,
- Prosedürlere parametre geçmek için yığın veya yazmaçlar kullanılır.
- Prosedürü sonlandırmak için prosedür adı ve ENDP direktifi yazılır.

Makro Özellikler



- Makro sadece adı yazılarak kullanılır.
- Makrolar program kodunda doğrudan genişletilir.
- Aynı makro 100 kez kullanılırsa, derleyici makroyu 100 kez genişletir.
 - Yürütülebilir dosya boyutu giderek büyür.
- Makroya parametre geçmek için makro adının ardından değerler yazılır.
- Makro sonlandırmak için ENDM direktifi yeterlidir.
- Macro tanımında etiketler varsa,
 - birden fazla kez kullanıldığında Duplicate declaration hatası alınabilir.
 - değişken, etiket, prosedür adları için LOCAL direktifi kullanılır.





```
MyMacro2 MACRO
  LOCAL label1, label2
  CMP AX, 2
  JE label1
  CMP AX, 3
  JE label2
  label1:
    INC AX
  label2:
   ADD AX, 2
ENDM
```

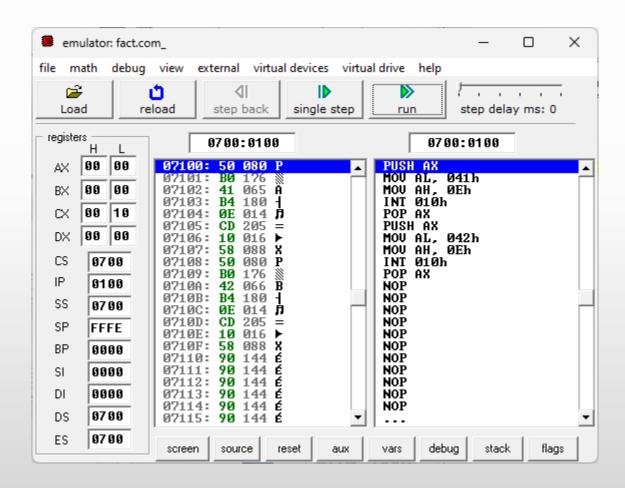




```
; this macro prints a char in AL and advances the current cursor position
PUTC MACRO char
    push ax
    mov al, char
    mov ah, 0eh
    int 10h
    pop ax
ENDM
ORG 100h
START:
    PUTC 'A'
    PUTC 'B'
```

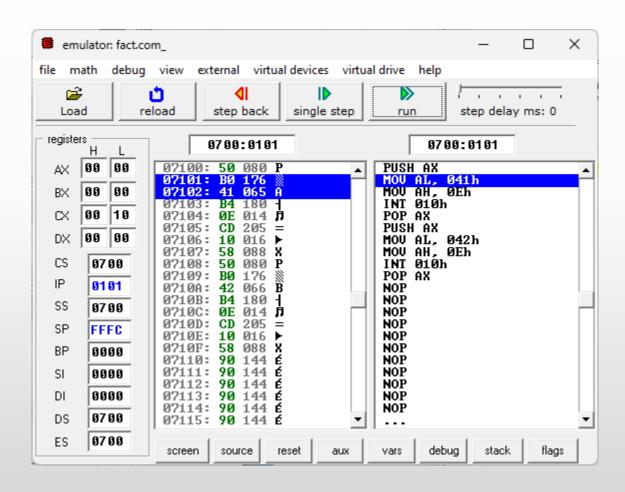






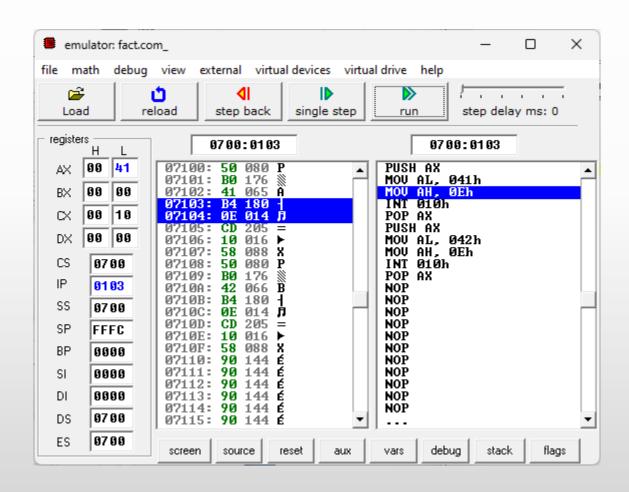






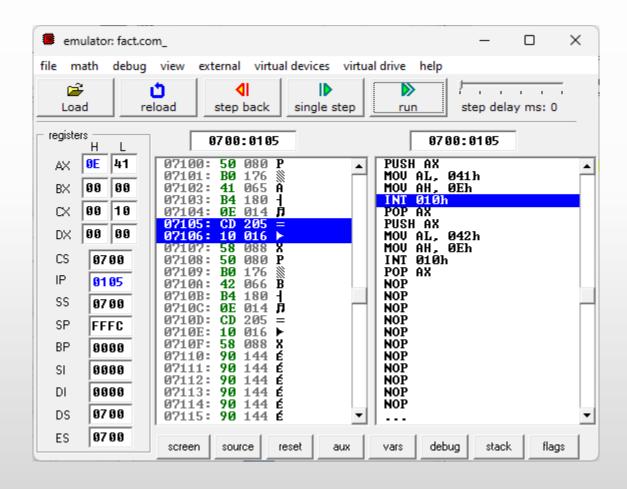






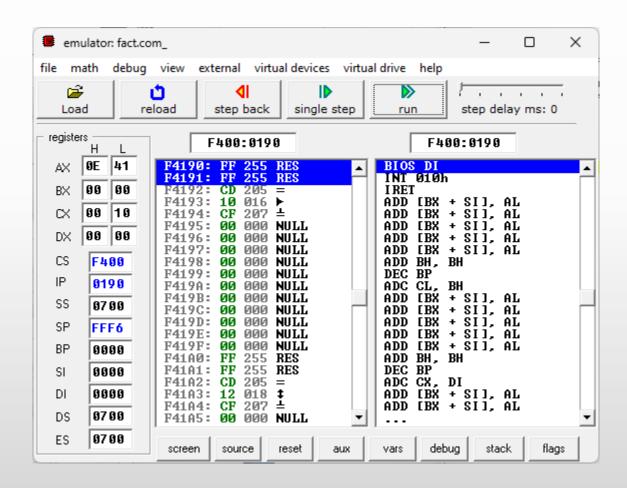








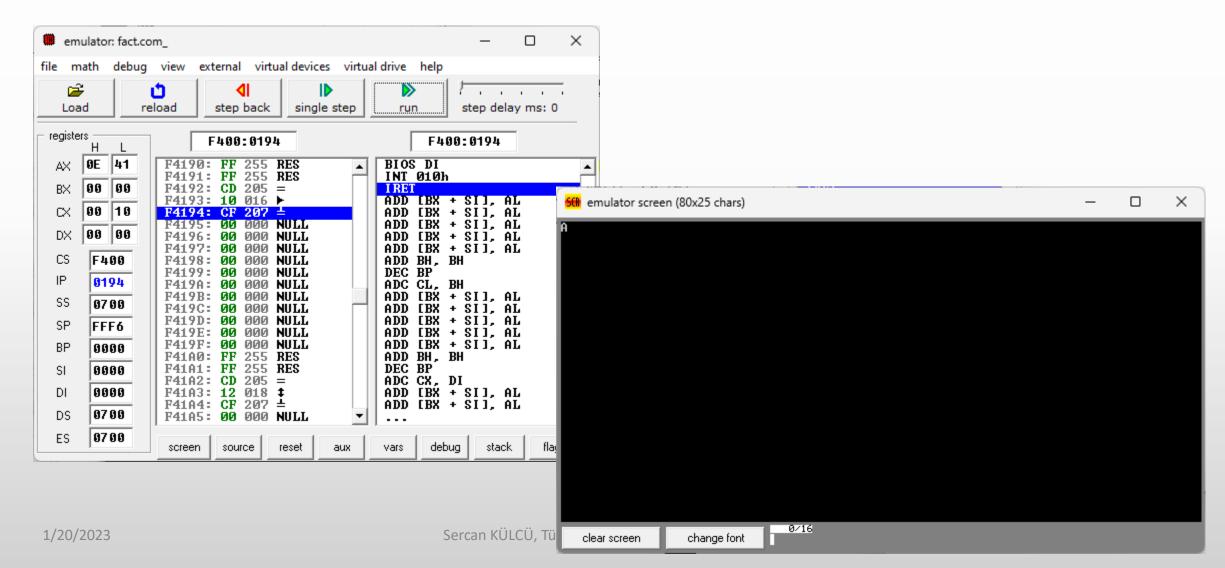






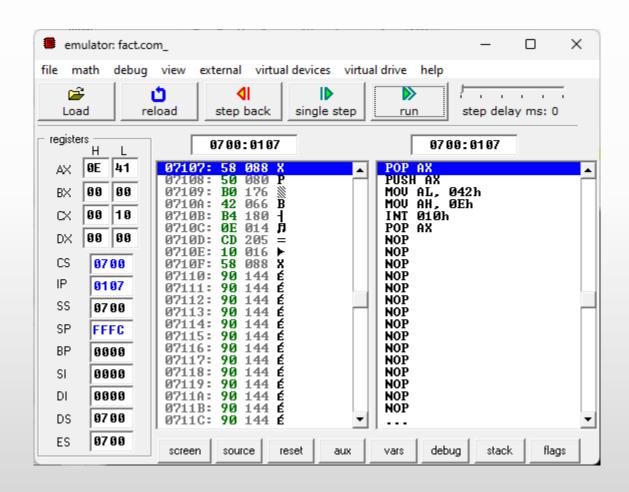
Ekrana Karakter Yazdırma







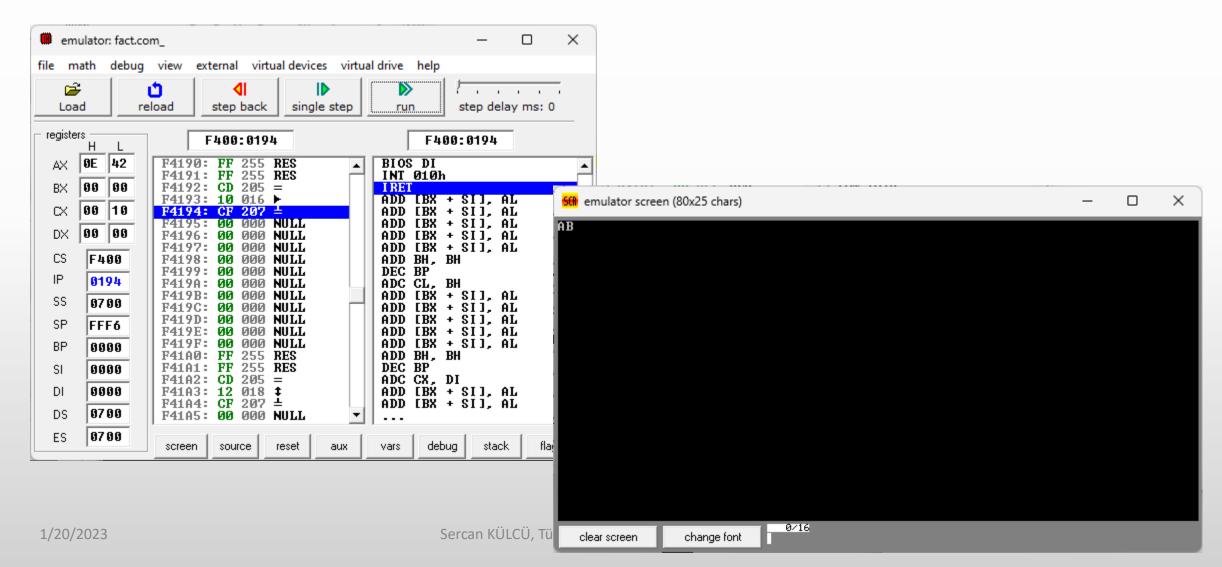






Ekrana Karakter Yazdırma





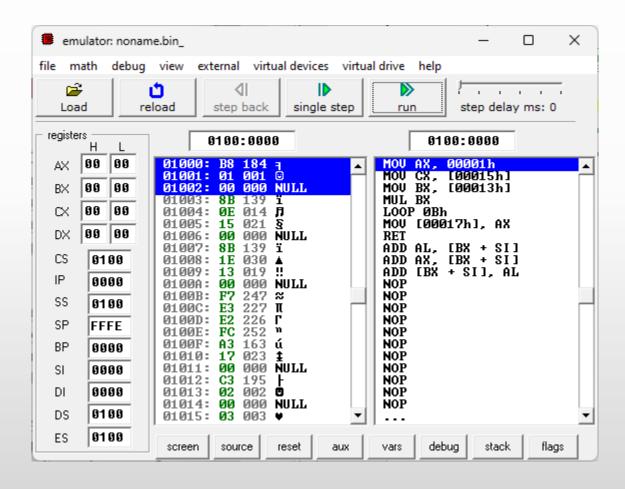




```
POWER macro b, e, r
  mov ax, 1
                    ; geçici sonucu 1 yap
                    ; üs değerini cx yazmacına yükle
  mov cx, e
  mov bx, b
                    ; taban değerini bx yazmacına yükle
  powerloop:
    mul bx
                    ; taban ile geçici sonucu çarp
    loop powerloop ; üs kere döngüyü tekrarla
  mov r, ax ; sonucu result değişkenine yaz
endm
start:
  POWER base, exponent, result ; power makrosunu çağır
ret
                         ; taban değeri
base
          dw
                         ; üs değeri
exponent dw
result
          dw
                         ; sonuç
```

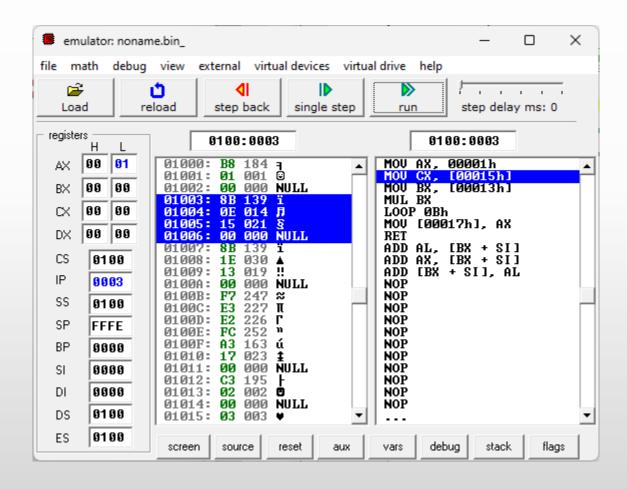






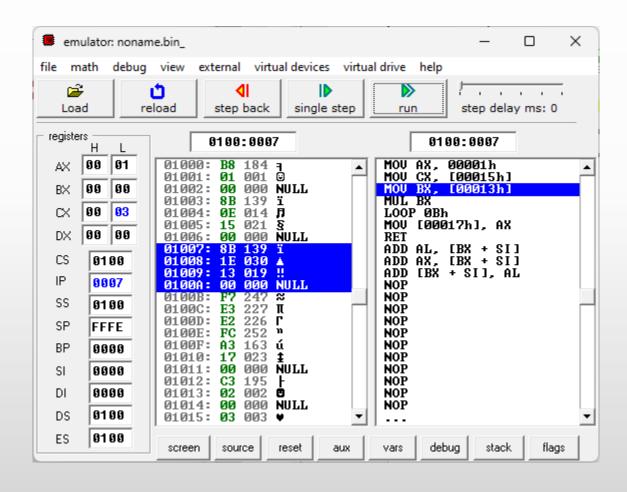






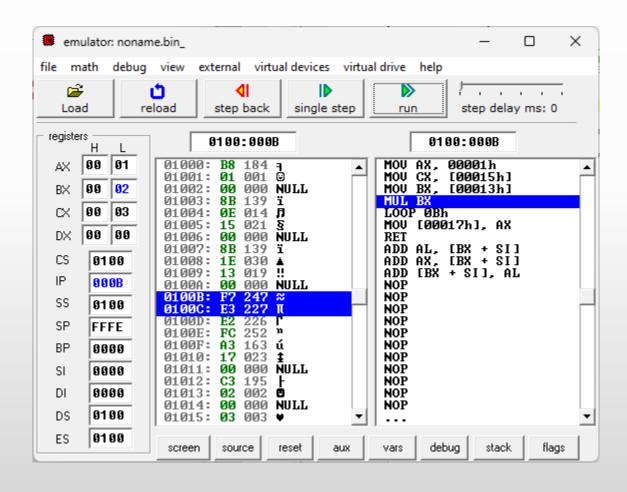






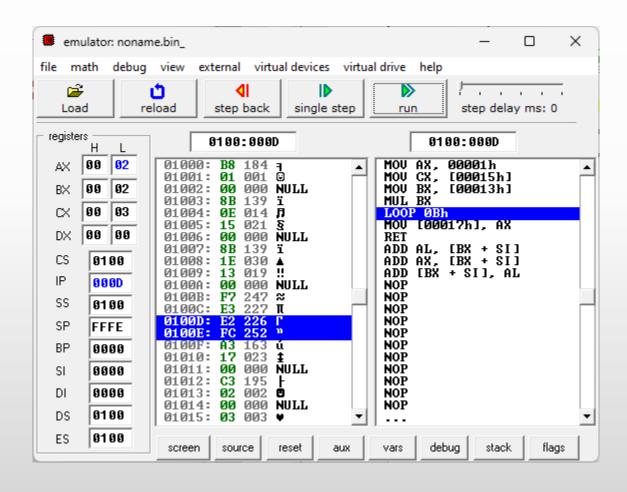






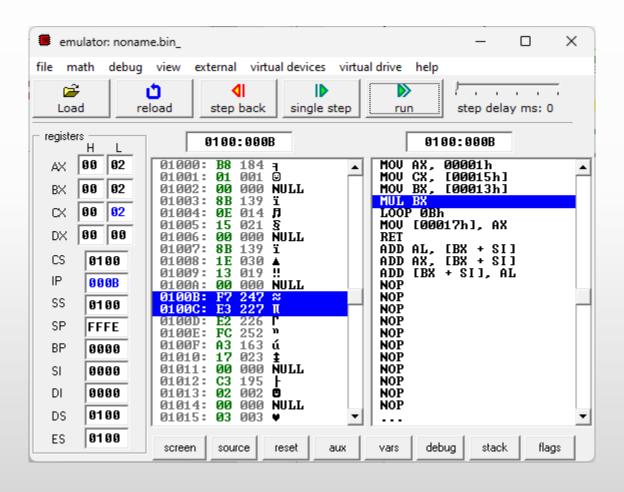






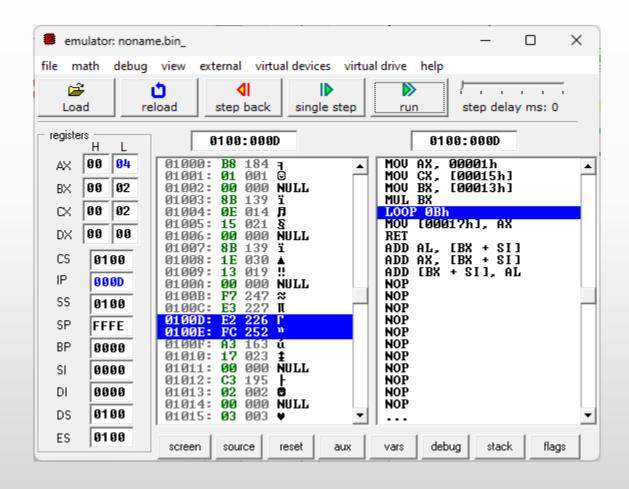






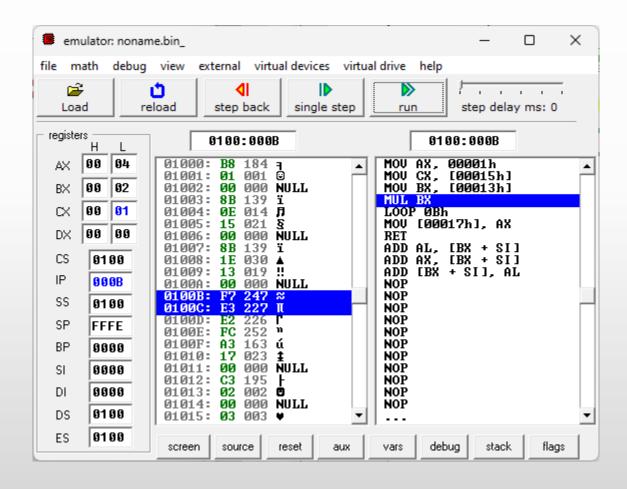






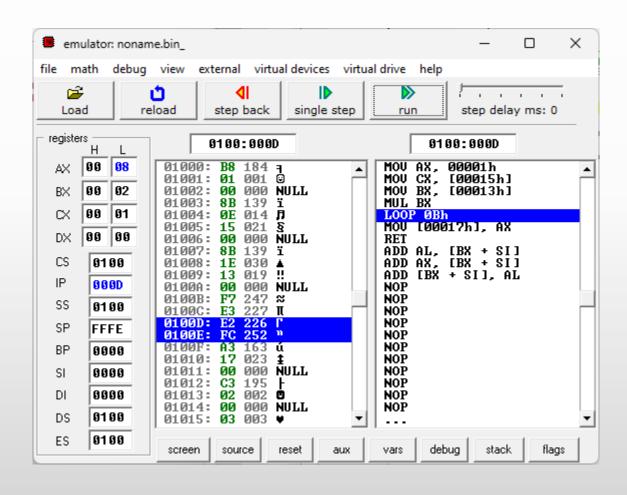






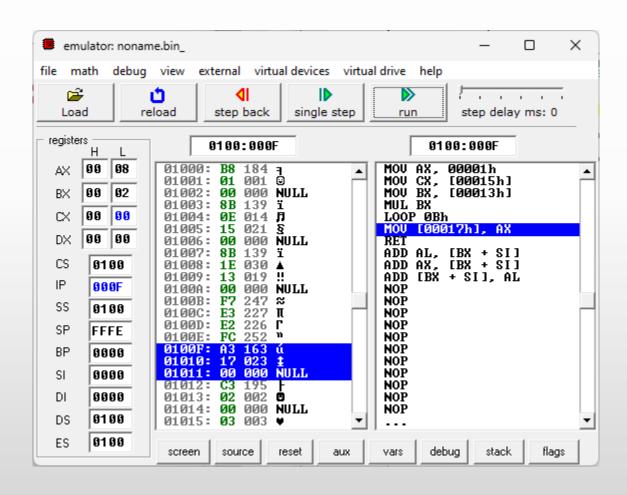






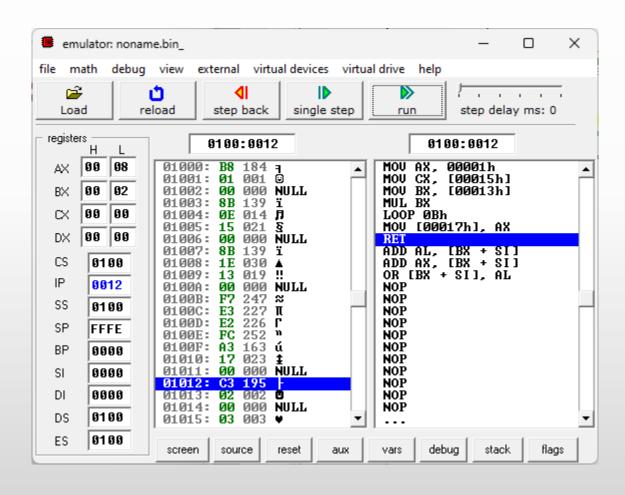














SON