



Bölüm 7: Prosedürler

Mikroişlemciler



Prosedürler

- Belirli bir görevi gerçekleştiren kod parçacığı.
- Programın içinden çağrılır.
- Görevi tamamladığında genellikle çağrıldığı noktaya geri döner.
- Programı daha yapılandırılmış (*structured*) ve anlaşılır hale getirir.



Prosedür Tanımlama

```
name PROC
```

```
    ; Prosedür kodu buraya yazılır.
```

```
    RET
```

```
name ENDP
```

name: Prosedürün adıdır. Başta ve sonda aynı olmalıdır.



Derleyici Direktifleri

- RET:
 - İşletim sistemine geri dönmek için kullanılır.
 - Aynı zamanda prosedürden dönmek için de kullanılır.
- PROC ve ENDP:
 - Derleyiciye prosedürün adresini hatırlatır.
 - Gerçek makine koduna çevrilmezler.
- CALL:
 - Bir prosedürü çağırmak için kullanılır.



Örnek Kod Parçası

```
ORG      100h
        CALL    m1
        MOV     AX, 2
RET                                ; return to operating system.
m1 PROC
        MOV     BX, 5
        RET     ; return to caller.
m1 ENDP
END
```





Örnek Kod Parçası

emulator: z01.com_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

| | H | L |
|----|-------|----|
| AX | 00 | 00 |
| BX | 00 | 00 |
| CX | 00 | 0B |
| DX | 00 | 00 |
| CS | 07 00 | |
| IP | 01 07 | |
| SS | 07 00 | |
| SP | FF FC | |
| BP | 00 00 | |
| SI | 00 00 | |
| DI | 00 00 | |
| DS | 07 00 | |
| ES | 07 00 | |

07 00: 01 07

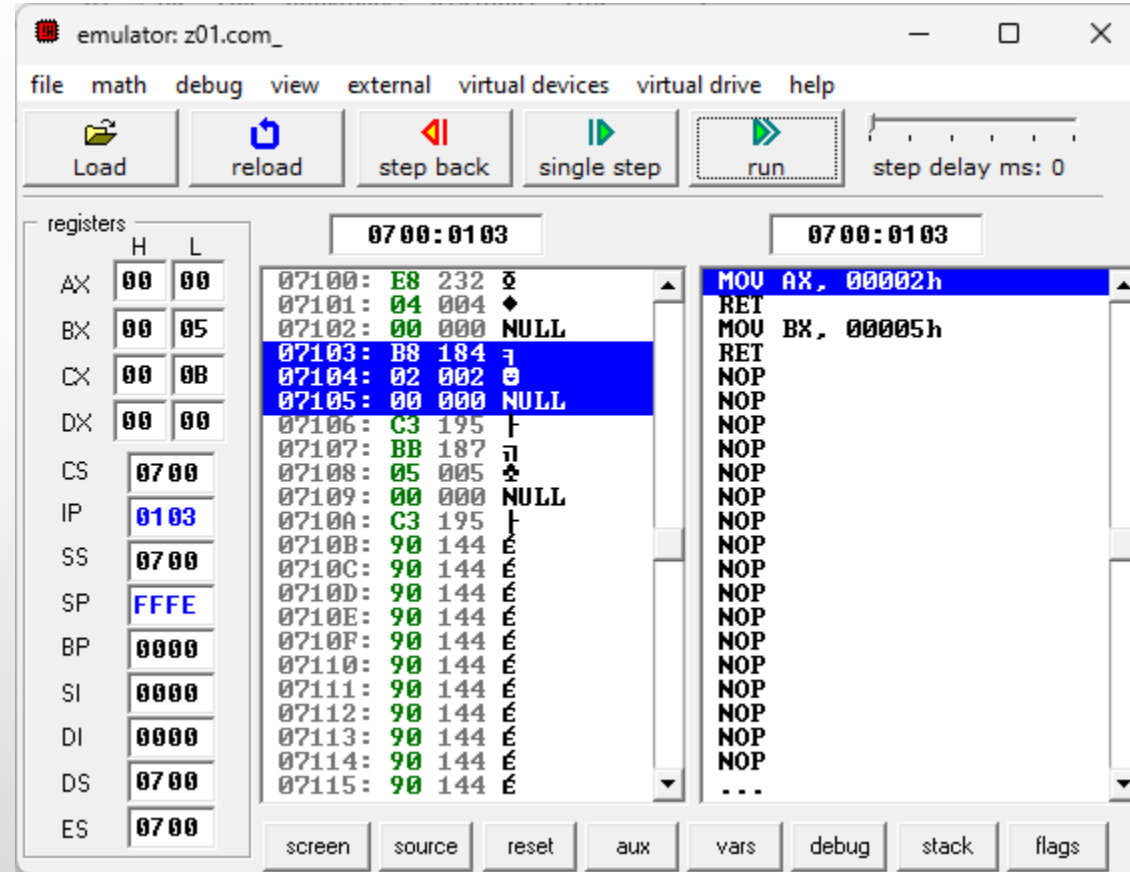
| | | | |
|--------|----|-----|------|
| 07100: | E8 | 232 | × |
| 07101: | 04 | 004 | ♦ |
| 07102: | 00 | 000 | NULL |
| 07103: | B8 | 184 | ? |
| 07104: | 02 | 002 | 0 |
| 07105: | 00 | 000 | NULL |
| 07106: | C3 | 195 | |
| 07107: | BB | 187 | ? |
| 07108: | 05 | 005 | ♠ |
| 07109: | 00 | 000 | NULL |
| 0710A: | C3 | 195 | |
| 0710B: | 90 | 144 | É |
| 0710C: | 90 | 144 | É |
| 0710D: | 90 | 144 | É |
| 0710E: | 90 | 144 | É |
| 0710F: | 90 | 144 | É |
| 07110: | 90 | 144 | É |
| 07111: | 90 | 144 | É |
| 07112: | 90 | 144 | É |
| 07113: | 90 | 144 | É |
| 07114: | 90 | 144 | É |
| 07115: | 90 | 144 | É |

CALL 00107h
MOV AX, 00002h
RET
MOV BX, 00005h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...

screen source reset aux vars debug stack flags

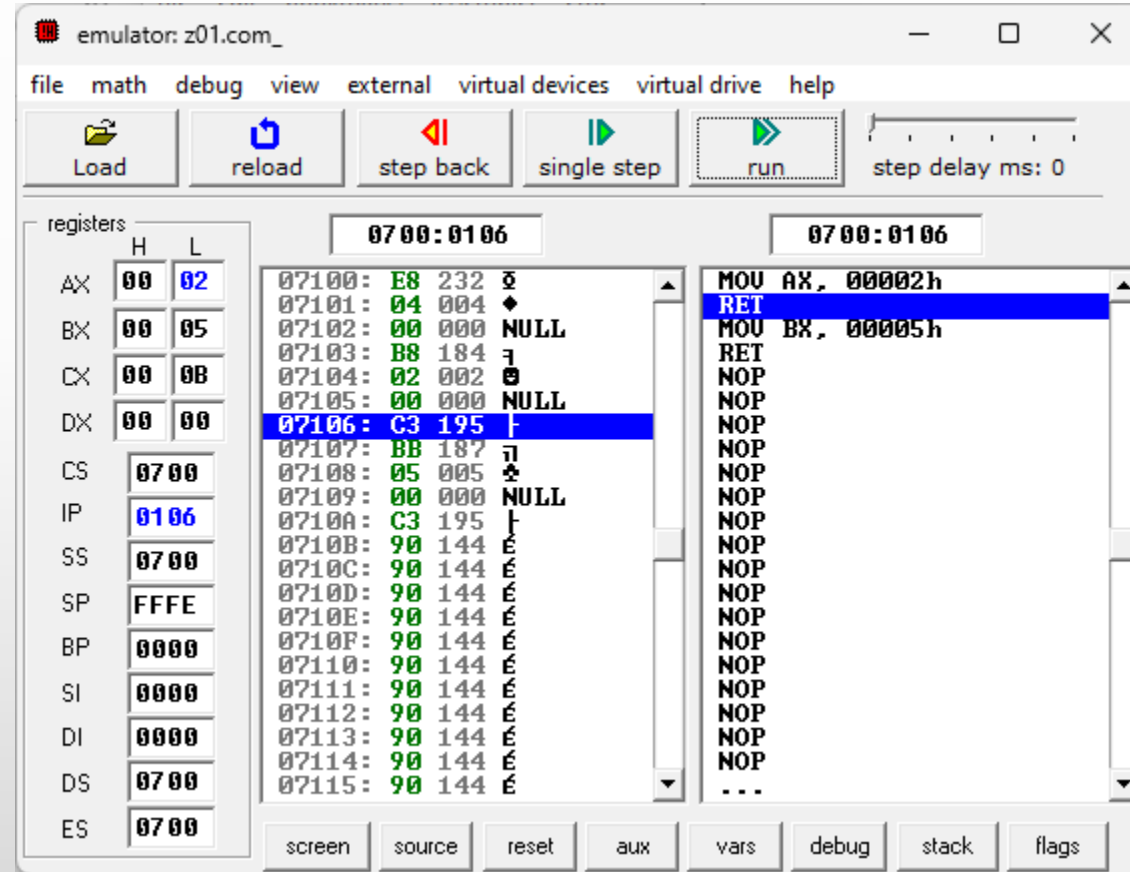


Örnek Kod Parçası





Örnek Kod Parçası





Prosedürlere Parametre Geçirme

- En kolay yolu, yazmaçları kullanmaktır.
- Örneğin, AL ve BL yazmaçları iki parametreyi temsil eder.
- m2 prosedürü,
 - AL ve BL yazmaçlarını kullanarak iki parametre alır,
 - Çarpar ve sonucu AX yazmacına saklar.
- CALL m2 prosedürünü çağırır.
- RET: Çağrıyı yapan yere döner.



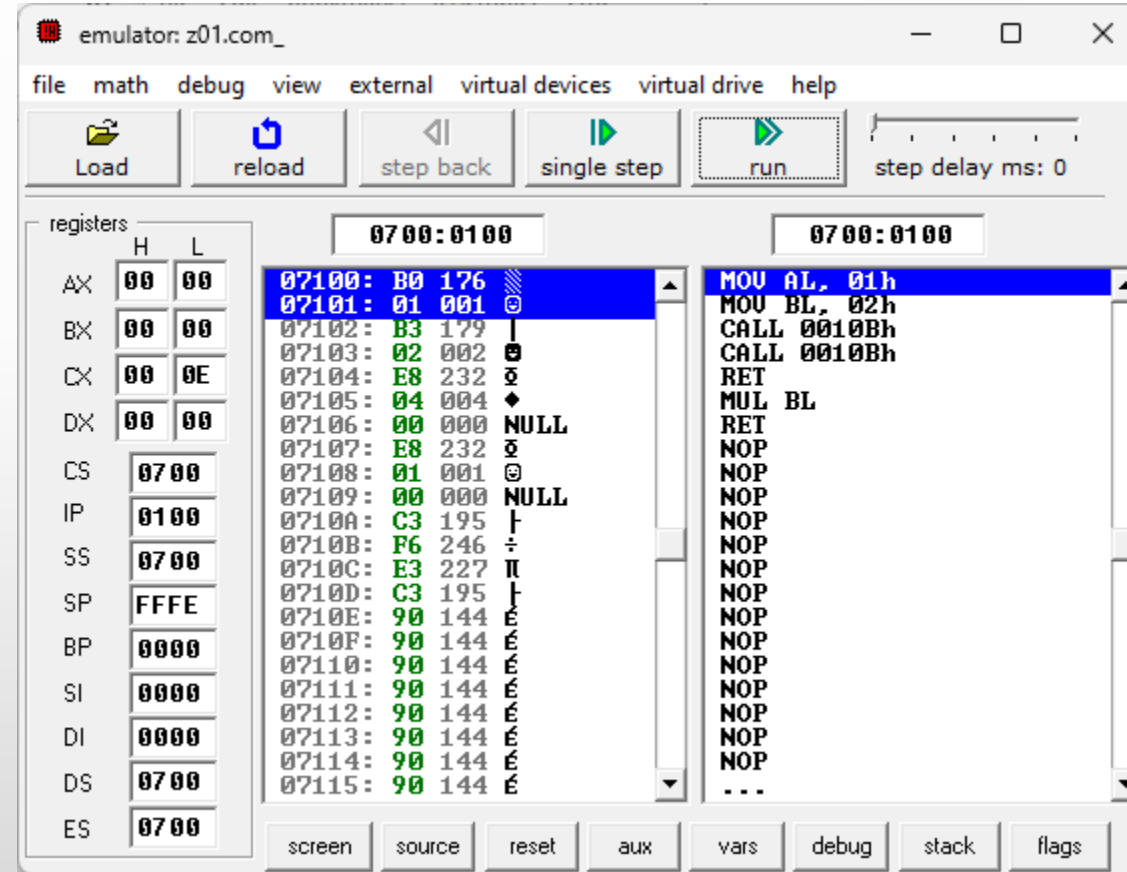
Prosedürlere Parametre Geçirme

```
ORG 100h
    MOV     AL, 1
    MOV     BL, 2
    CALL    m2
    CALL    m2

RET                                ; return to operating system.
m2 PROC
    MUL     BL                    ; AX = AL * BL.
    RET                                ; return to caller.
m2 ENDP
END
```

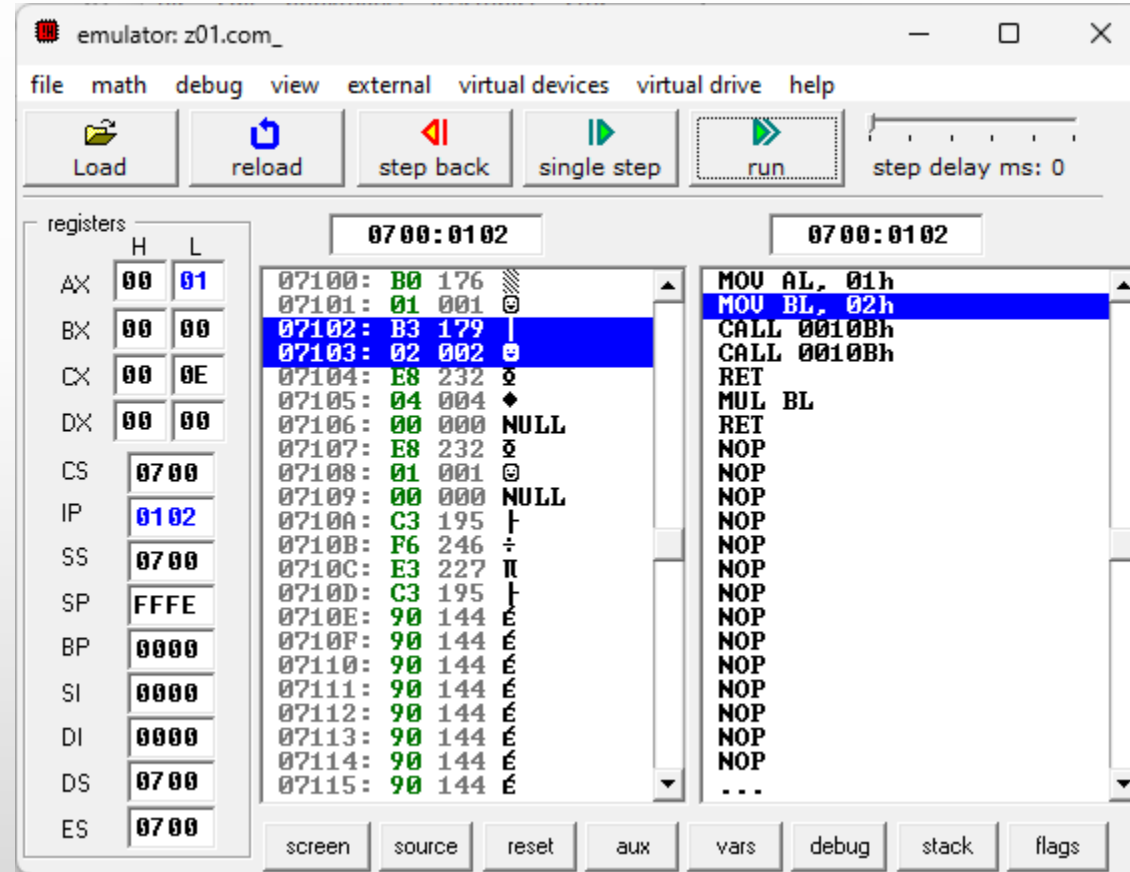


Prosedürlere Parametre Geçirme



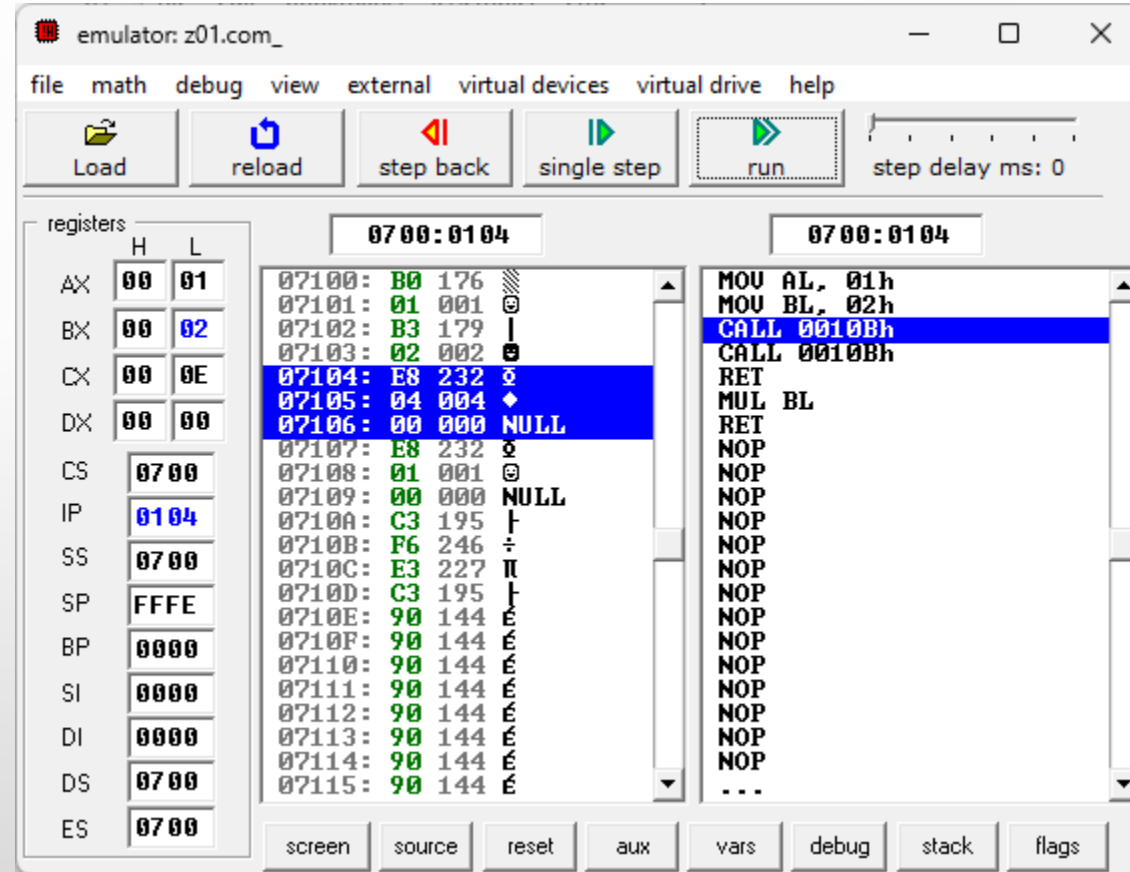


Prosedürlere Parametre Geçirme



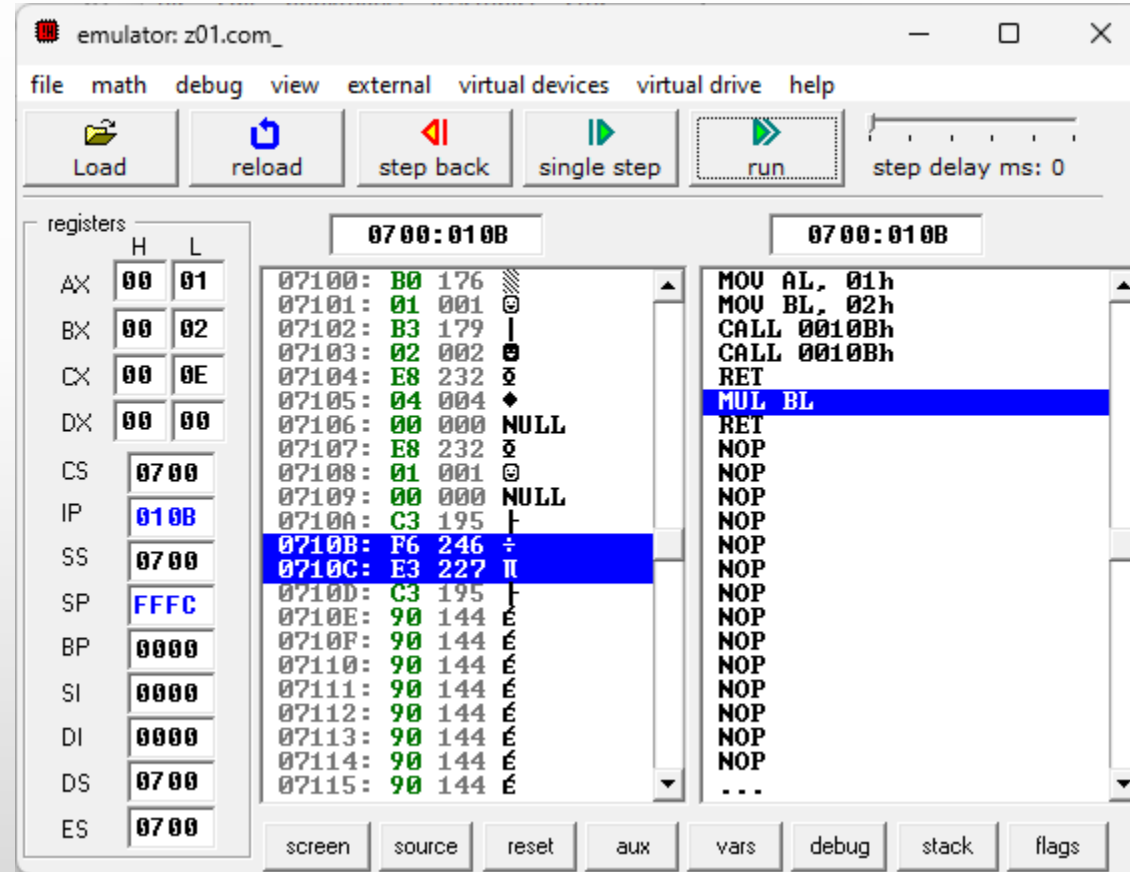


Prosedürlere Parametre Geçirme



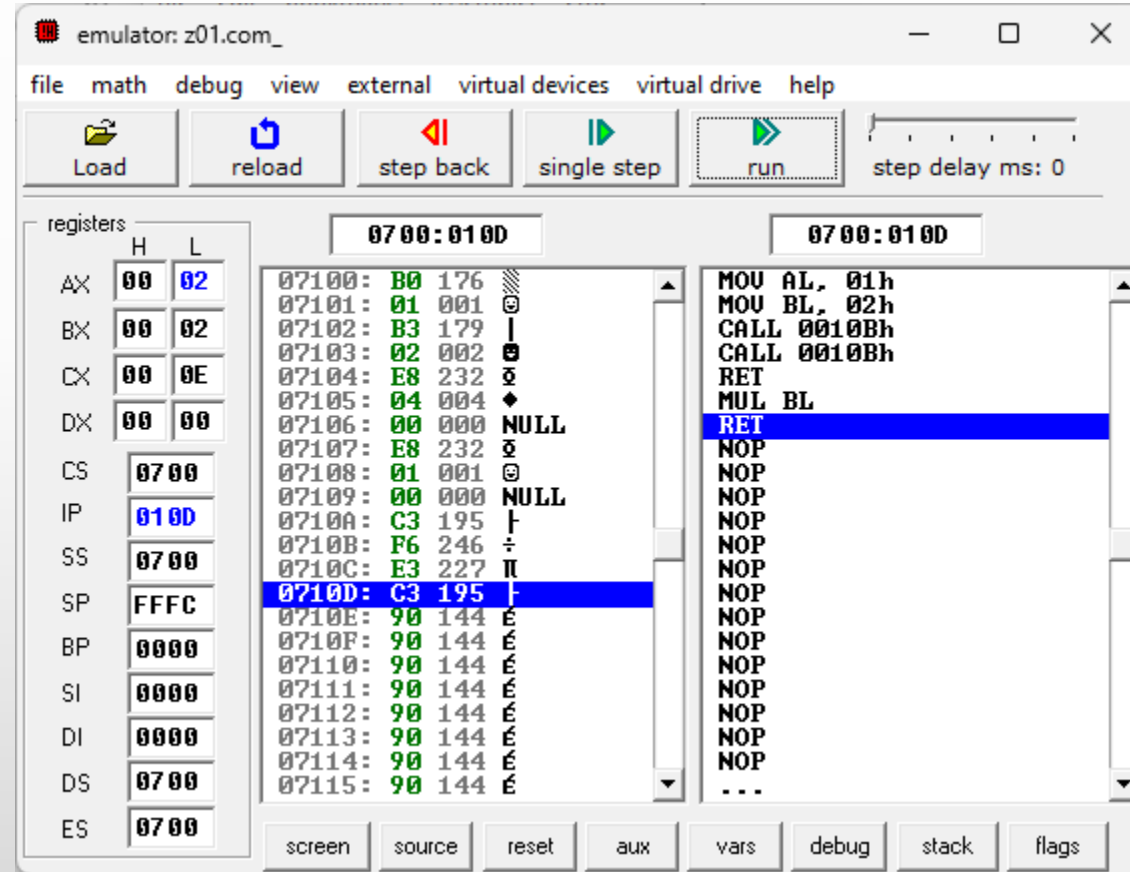


Prosedürlere Parametre Geçirme





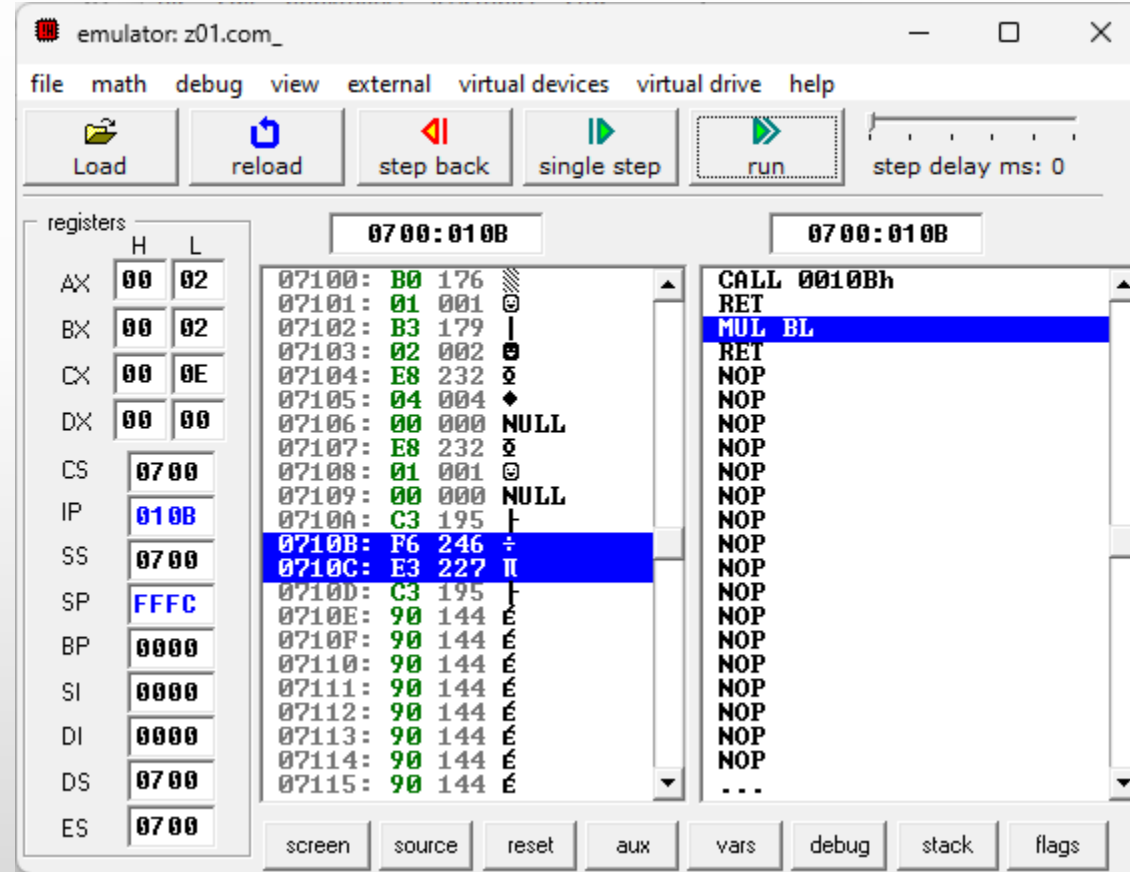
Prosedürlere Parametre Geçirme





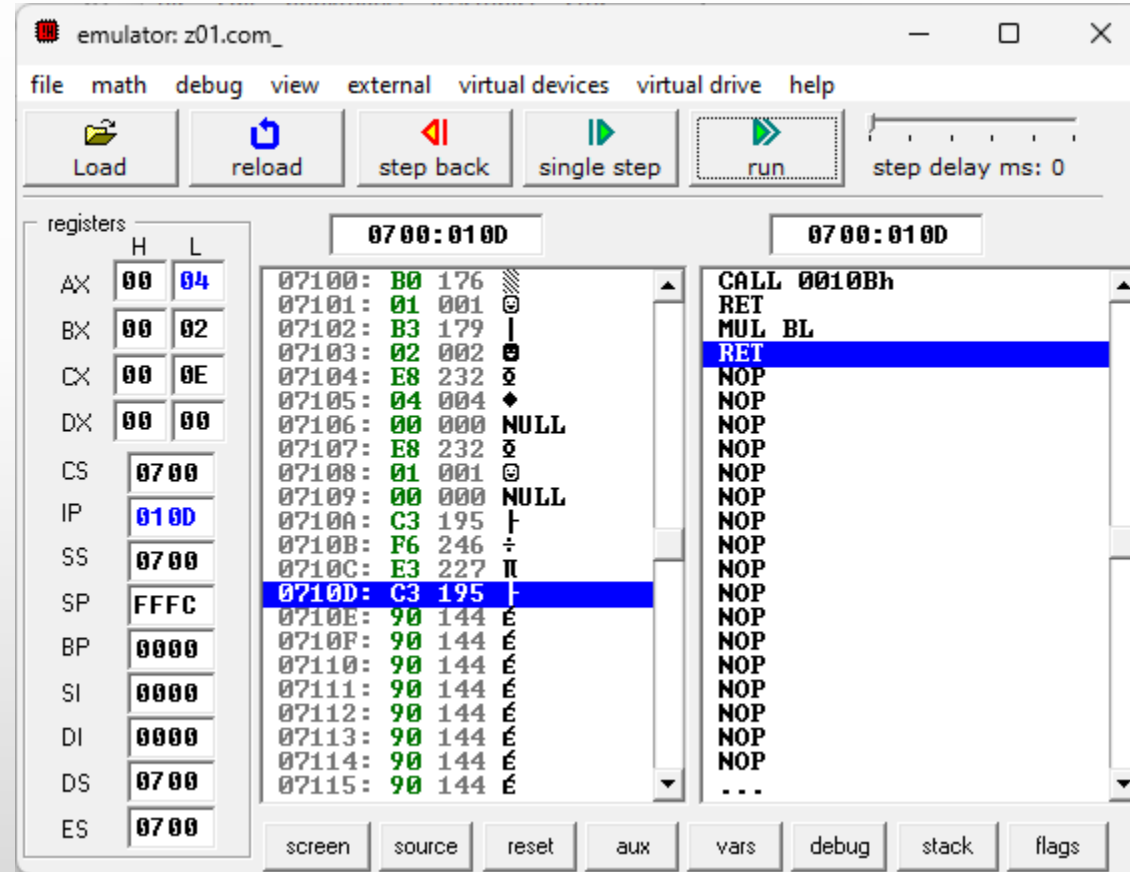


Prosedürlere Parametre Geçirme





Prosedürlere Parametre Geçirme





Merhaba Dünya Mesajı Yazdırma

- Prosedür kullanarak "Merhaba Dünya!" mesajı yazdırma.
- LEA SI, msg:
 - msg adlı dizgenin adresini SI yazmacına yükler.
- CALL print_me:
 - print_me prosedürünü çağırır.
- RET:
 - İşletim sistemine geri döner.
- print_me prosedürü,
 - null ile sona eren bir dizgeyi yazdırır.



Merhaba Dünya Mesajı Yazdırma

```
ORG      100h
LEA      SI, msg          ; load address of msg to SI.
CALL     print_me
RET      ; return to operating system.
print_me PROC
; .....
print_me ENDP
msg      DB  'Hello World!', 0 ; null terminated string.
END
```



Merhaba Dünya Mesajı Yazdırma

```
print_me      PROC
next_char:
    CMP  b.[SI], 0      ; check for zero to stop
    JE   stop          ;
    MOV  AL, [SI]       ; next get ASCII char.
    MOV  AH, 0Eh        ; teletype function number.
    INT  10h            ; using interrupt to print a char in AL.
    ADD  SI, 1          ; advance index of string array.
    JMP  next_char      ; go back, and type another char.
stop:
RET           ; return to caller.
print_me      ENDP
```



İki Değişken Toplama

start:

```
CALL AddNumbers    ; prosedürü çağır
```

```
RET
```

```
AddNumbers PROC
```

```
MOV AX, number1    ; İlk sayıyı AX yazmacına yükle
```

```
ADD AX, number2    ; İkinci sayıyı AX yazmacına ekle
```

```
MOV result, AX     ; Sonucu result değişkenine taşı
```

```
RET                ; Prosedürü bitir
```

```
AddNumbers ENDP
```

```
number1 DW 5       ; İlk sayı 5
```

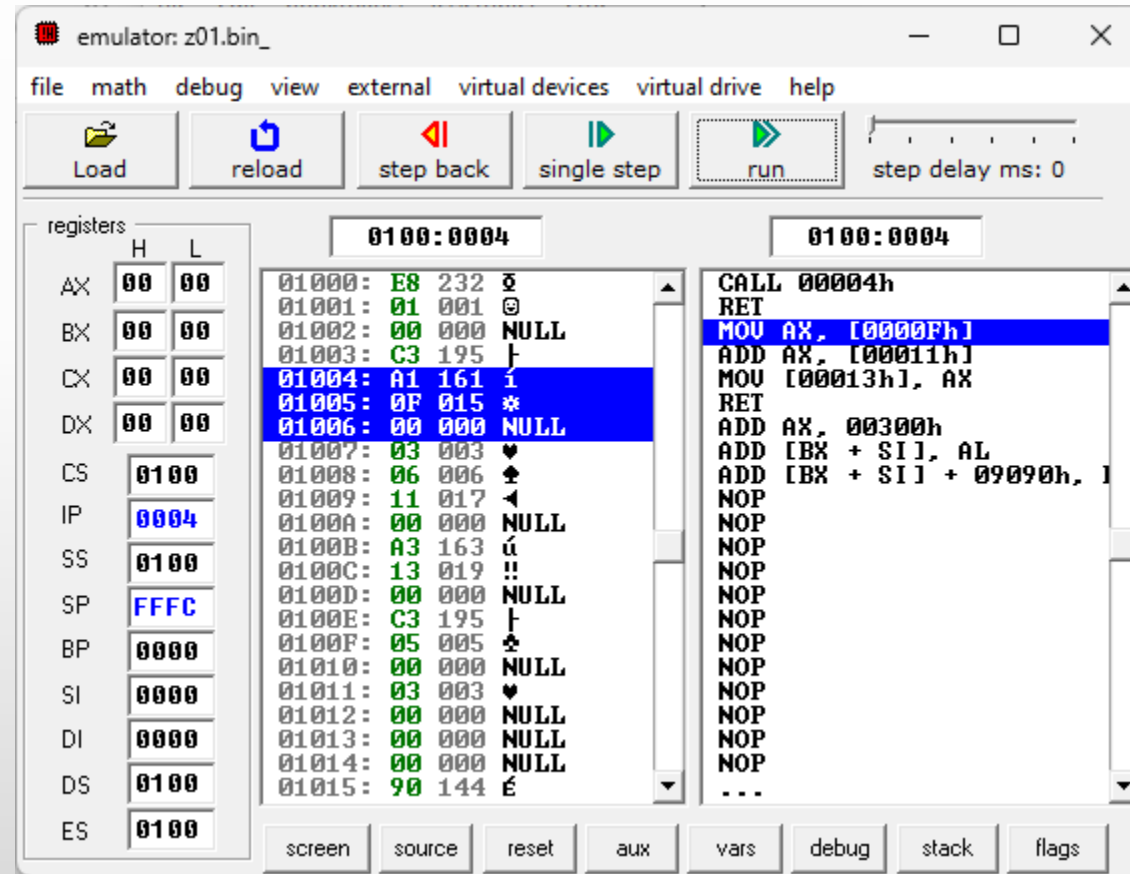
```
number2 DW 3       ; İkinci sayı 3
```

```
result  DW ?       ; Sonucu tutar
```





İki Değişken Toplama





İki Değişken Toplama

emulator: z01.bin_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

| | H | L |
|----|------|----|
| AX | 00 | 05 |
| BX | 00 | 00 |
| CX | 00 | 00 |
| DX | 00 | 00 |
| CS | 0100 | |
| IP | 0007 | |
| SS | 0100 | |
| SP | FFFC | |
| BP | 0000 | |
| SI | 0000 | |
| DI | 0000 | |
| DS | 0100 | |
| ES | 0100 | |

0100:0007

| | | | |
|--------|----|-----|------|
| 01000: | E8 | 232 | õ |
| 01001: | 01 | 001 | 0 |
| 01002: | 00 | 000 | NULL |
| 01003: | C3 | 195 | † |
| 01004: | A1 | 161 | i |
| 01005: | 0F | 015 | * |
| 01006: | 00 | 000 | NULL |
| 01007: | 03 | 003 | ♥ |
| 01008: | 06 | 006 | ♠ |
| 01009: | 11 | 017 | ◀ |
| 0100A: | 00 | 000 | NULL |
| 0100B: | A3 | 163 | ü |
| 0100C: | 13 | 019 | !! |
| 0100D: | 00 | 000 | NULL |
| 0100E: | C3 | 195 | † |
| 0100F: | 05 | 005 | ♠ |
| 01010: | 00 | 000 | NULL |
| 01011: | 03 | 003 | ♥ |
| 01012: | 00 | 000 | NULL |
| 01013: | 00 | 000 | NULL |
| 01014: | 00 | 000 | NULL |
| 01015: | 90 | 144 | É |

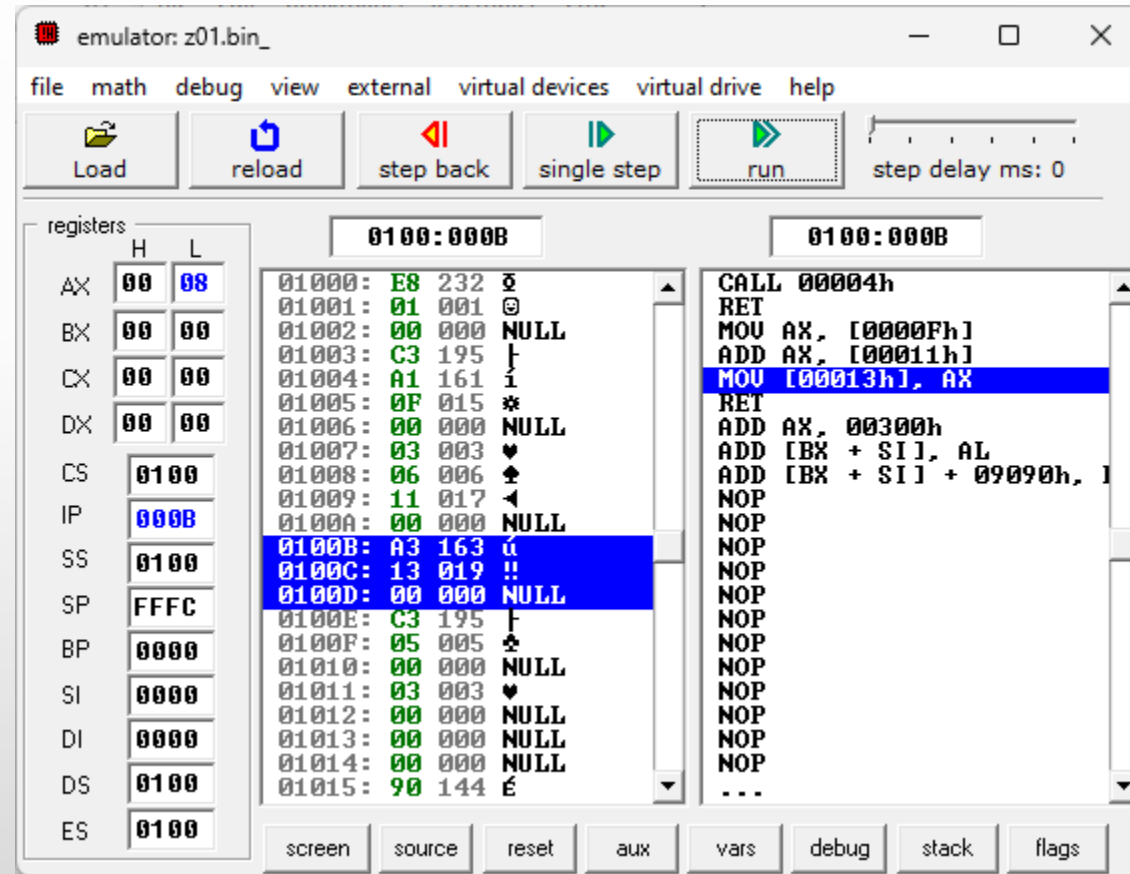
0100:0007

```
CALL 00004h
RET
MOV AX, [0000Fh]
ADD AX, [00011h]
MOV [00013h], AX
RET
ADD AX, 00300h
ADD [BX + SI], AL
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags

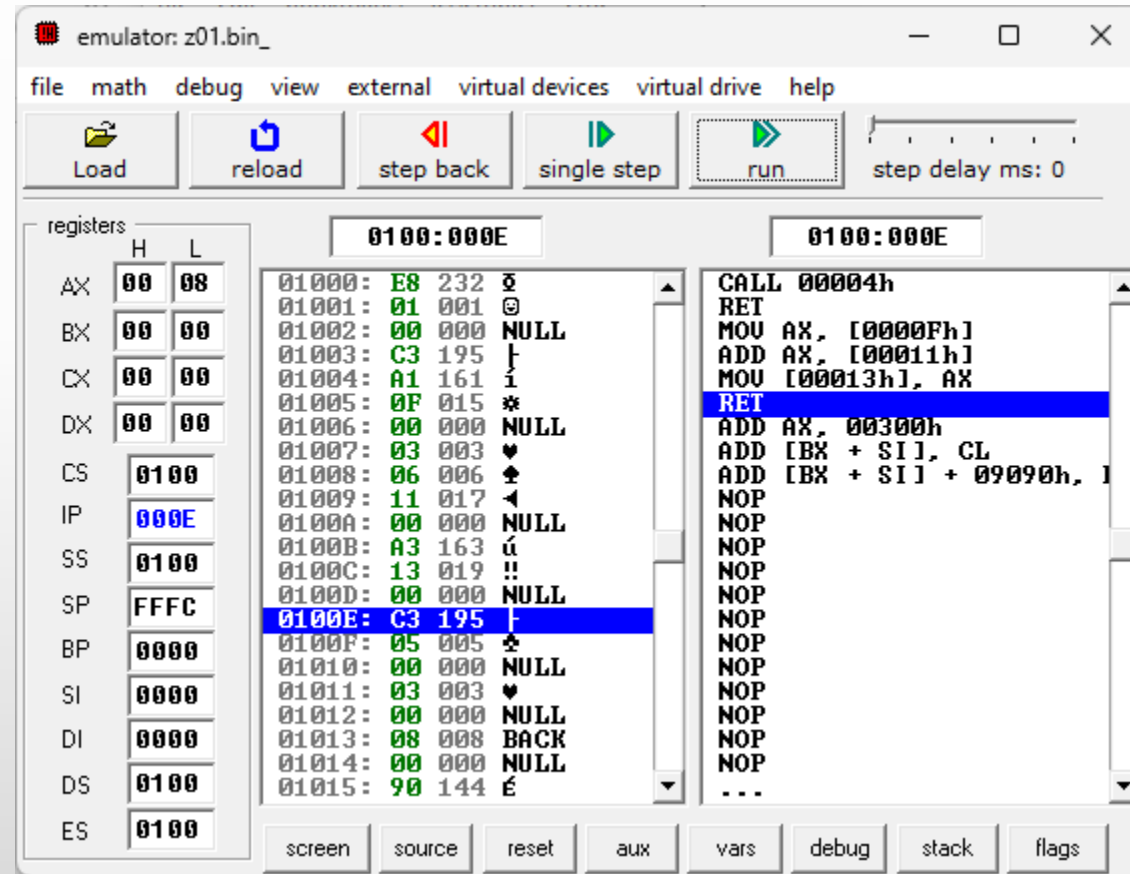


İki Değişken Toplama



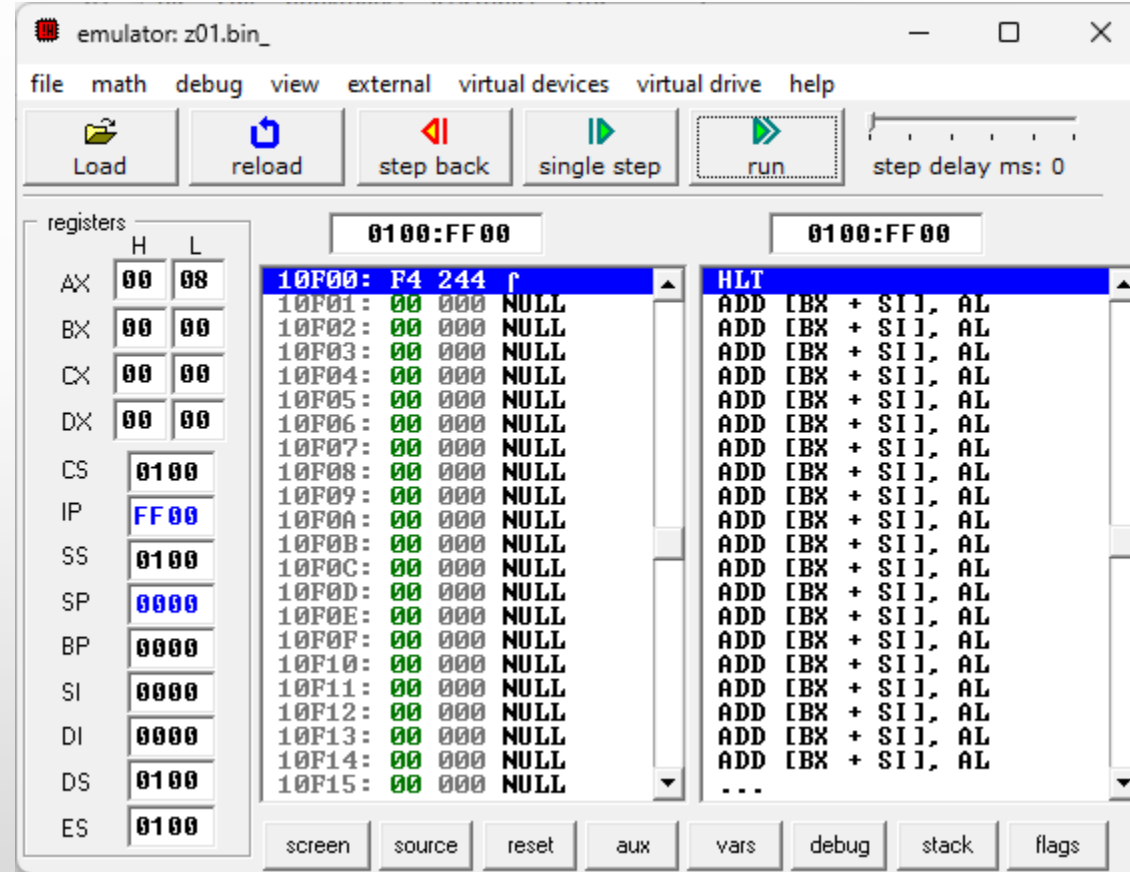


İki Değişken Toplama





İki Değişken Toplama





SON