Bölüm 8: Yığın

Mikroişlemciler

Yığın (Stack)

- Yığın, geçici verileri saklayan bellek alanıdır.
- CALL komutu tarafından kullanılır ve prosedürün dönüş adresini saklar.
- RET komutu, bu değeri yığından alır ve belirtilen adrese döner.
- INT komutu bir kesmeyi çağırdığında benzer bir işlem gerçekleşir;
  - Durum yazmacı, kod kesimi ve bağıl konum değeri yığına saklanır
  - IRET komutu, kesme çağrısından dönmek için kullanılır.
- PUSH: 16 bit değeri yığına koyar.
- POP: 16 bit değeri yığından alır.

Örnek Kod Parçası

```
ORG     100h

MOV     AX, 42          ; AX register'ına 42 değerini ata.
PUSH    AX              ; AX değerini yığına koy.
POP     BX              ; Yığından değeri BX yazmacına al.
RET                     ; İşletim sistemine dön.

END
```

PUSH Komutu

- PUSH REG
- PUSH SREG
- PUSH memory
- PUSH immediate

- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, CS.
- memory: [BX], [BX+SI+7], 16 bit variable, gibi ..
- immediate: 5, -24, 3Fh, 10001101b, gibi ..

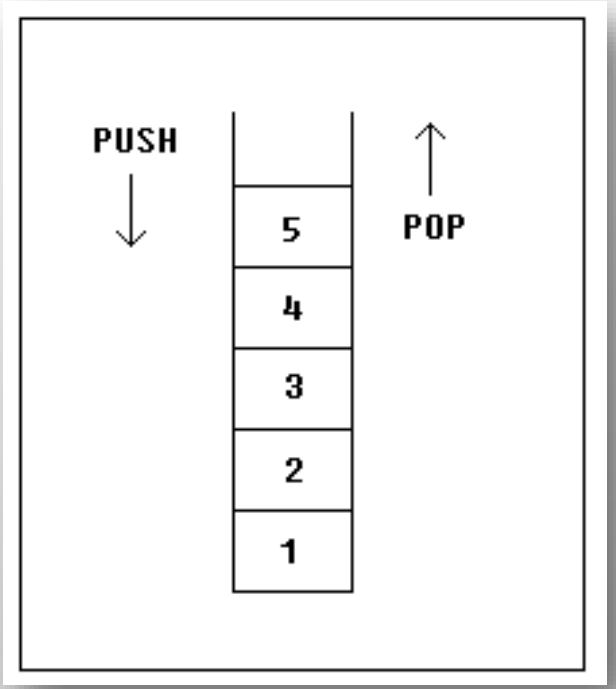
POP Komutu

- POP REG
- POP SREG
- POP memory

- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, (CS hariç).
- memory: [BX], [BX+SI+7], 16 bit variable, gibi ..

LIFO Algoritması

- LIFO, "Last In First Out"un kısaltmasıdır.
- Yığına eklenen son öğe, yığından çıkan ilk öğe olur.
- Yığına koyulan değerler sırasıyla (1, 2, 3, 4, 5) olsun,
  - Yığından ilk alınan değer 5 olur,
  - Ardından sırasıyla 4, 3, 2 ve en son 1 alınır.



Örnek Kod Parçası

```
MOV     AX, 1           ; AX yazmacına 1 değerini ata.
PUSH    AX              ; 1 değerini yığına koy.
MOV     AX, 2           ; AX yazmacına 2 değerini ata.
PUSH    AX              ; 2 değerini yığına koy.
MOV     AX, 3           ; AX yazmacına 3 değerini ata.
PUSH    AX              ; 3 değerini yığına koy.
POP     BX              ; Yığından BX yazmacına al. (3)
POP     BX              ; Yığından BX yazmacına al. (2)
POP     BX              ; Yığından BX yazmacına al. (1)
```

Yığın Güvenliği

- PUSH ve POP komutları,
  - programın çalışma süresince geçici verileri saklar.
- Yığının bütünlüğü,
  - eşit sayıda PUSH ve POP işlemi ile korunmalıdır.
- Eğer eşit sayıda PUSH ve POP yapılmazsa,
  - yığın bozulabilir,
  - işletim sistemine geri dönüş yapılamayabilir.
- RET komutu, yığında bir dönüş adresi bekler (genellikle 0000h).

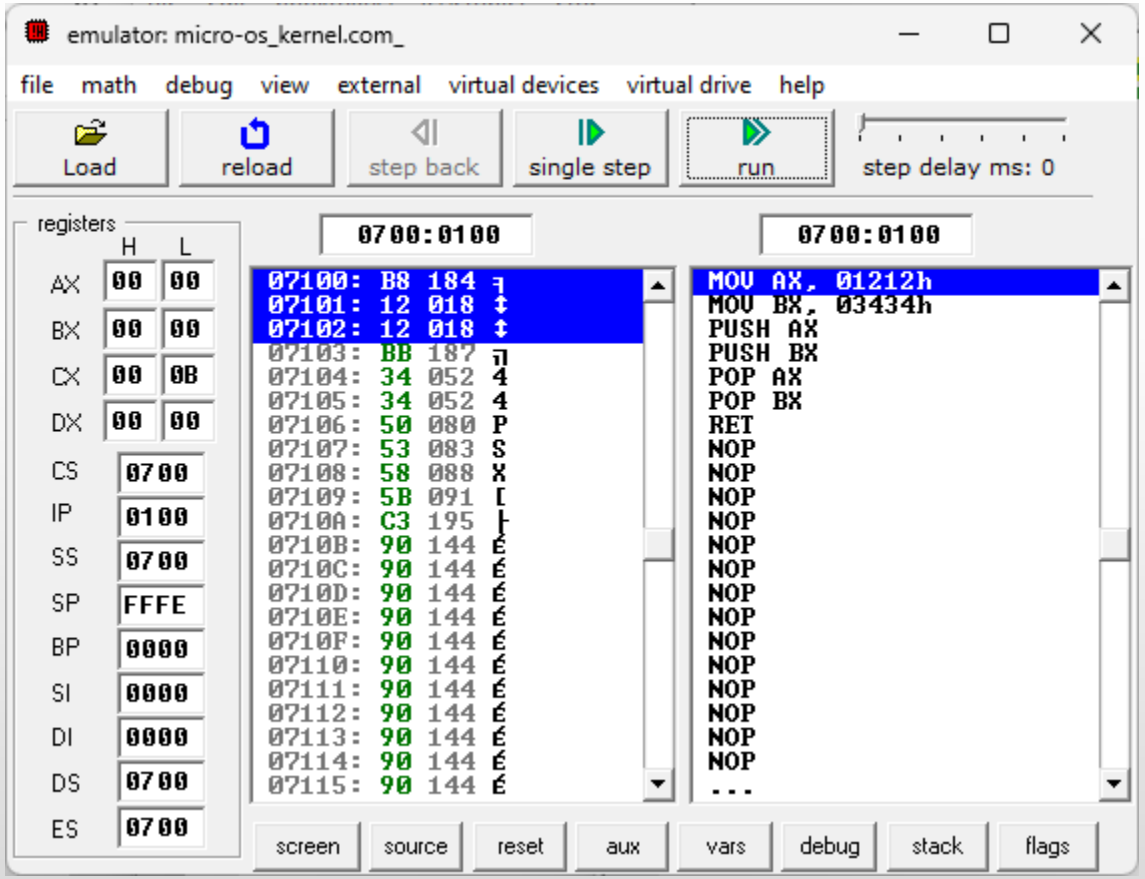
Örnek Kod Parçası

```
ORG     100h
MOV     AX, 1234h
PUSH    AX          ; store value of AX in stack.
MOV     AX, 5678h   ; modify the AX value.
POP     AX          ; restore the original value of AX.
RET
END
```

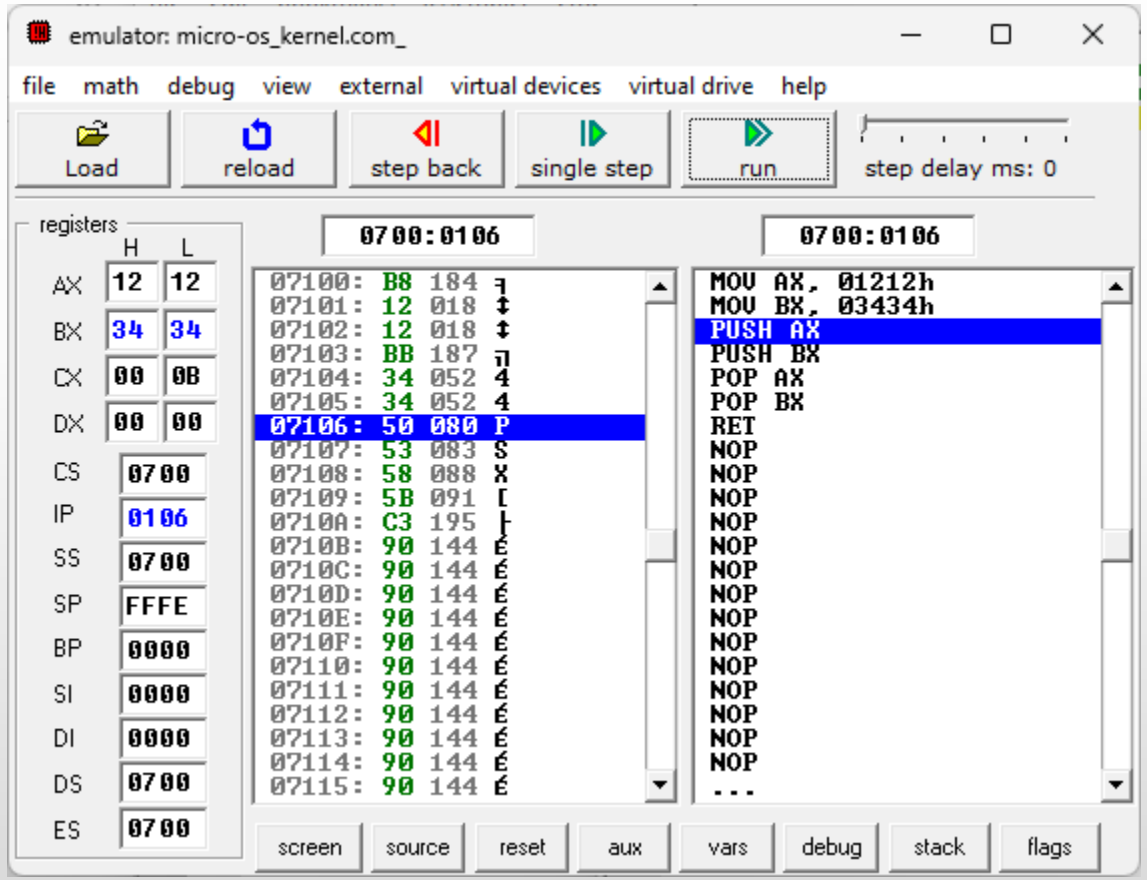
Örnek Kod Parçası

```
ORG     100h
MOV     AX, 1212h       ; store 1212h in AX.
MOV     BX, 3434h       ; store 3434h in BX
PUSH    AX              ; store value of AX in stack.
PUSH    BX              ; store value of BX in stack.
POP     AX              ; set AX to original value of BX.
POP     BX              ; set BX to original value of AX.
RET
END
```

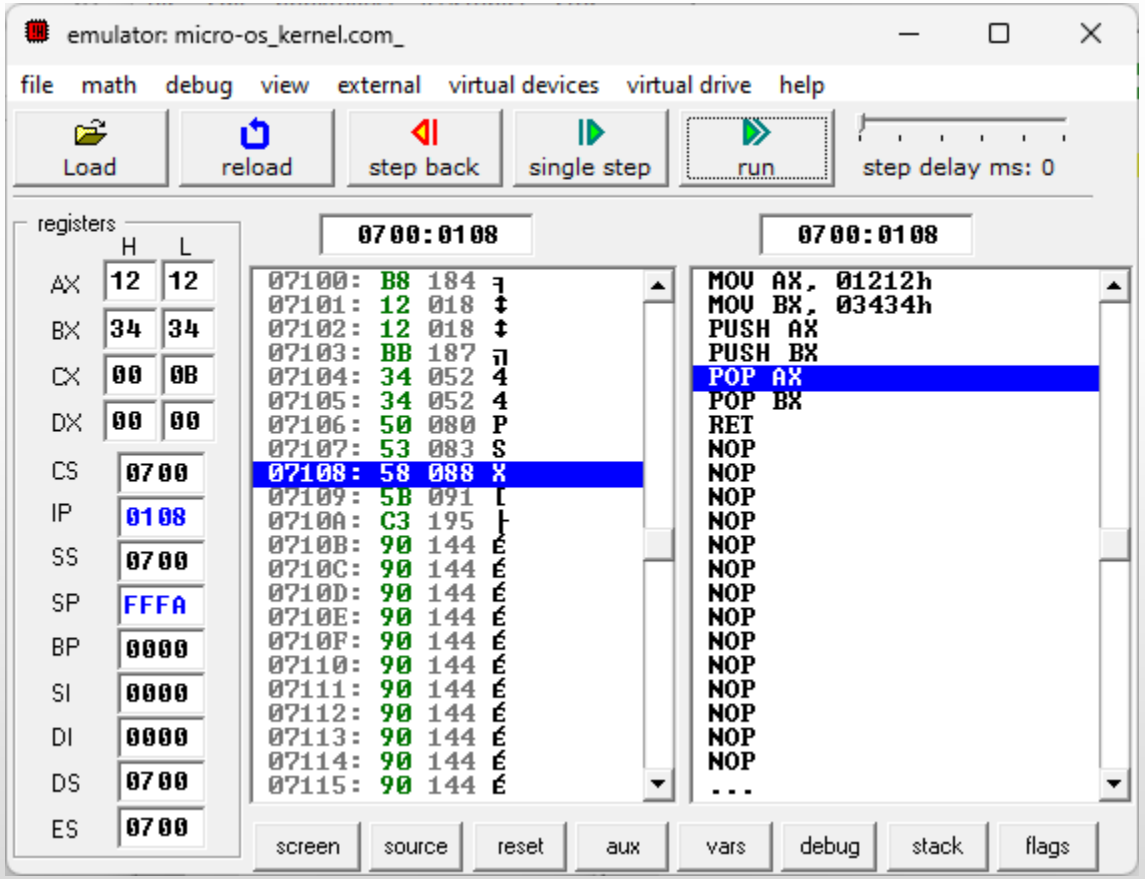
Örnek Kod Parçası



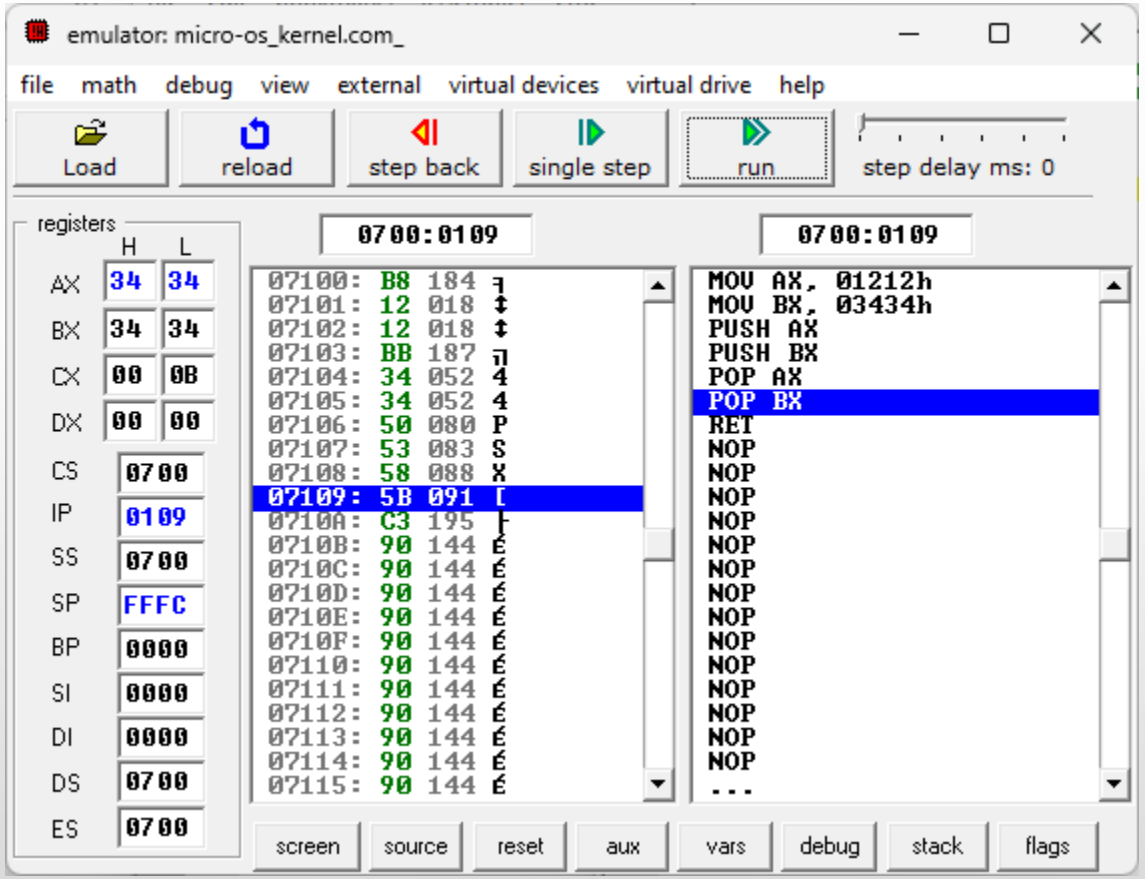
Örnek Kod Parçası



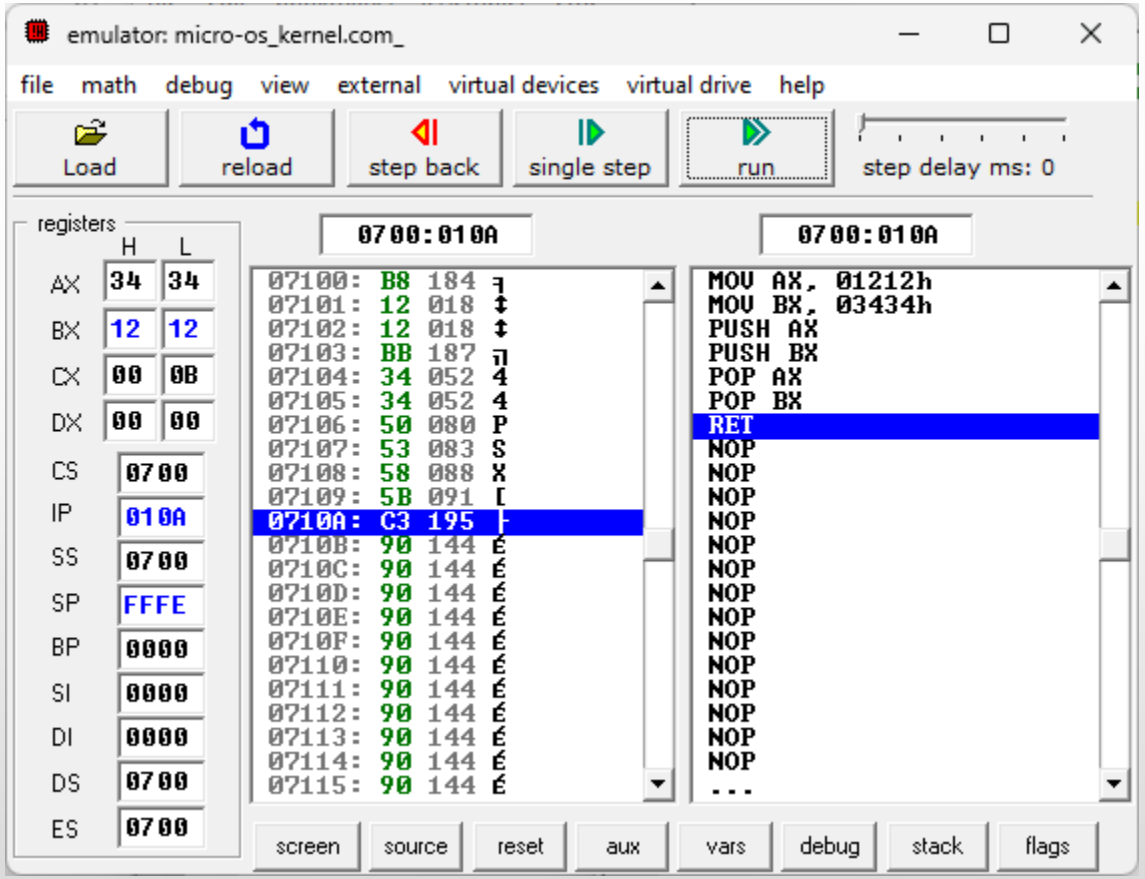
Örnek Kod Parçası



Örnek Kod Parçası



Örnek Kod Parçası



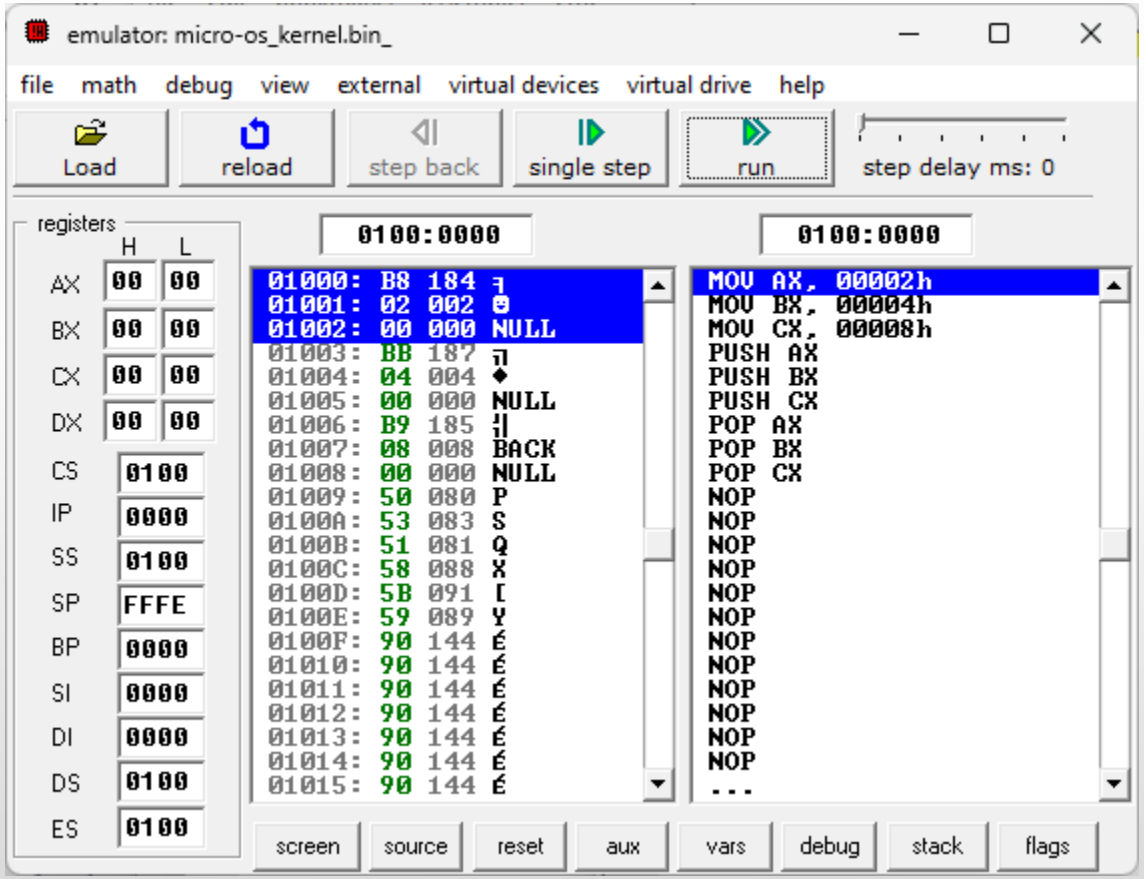
Yığın Belleği ve İşleyişi

- Yığın belleği,
  - SS (Yığın Kesimi) ve SP (Yığın İşaretçisi) yazmaçlarını kullanır.
- İşletim sistemi genellikle bu yazmaçların başlangıç değerlerini belirler.
- PUSH kaynak:
  - SP yazmacından 2 çıkarılır.
  - Kaynak değeri SS:SP adresine yazılır.
- POP hedef:
  - SS:SP adresindeki değer hedefe yazılır.
  - SP yazmacına 2 eklenir.

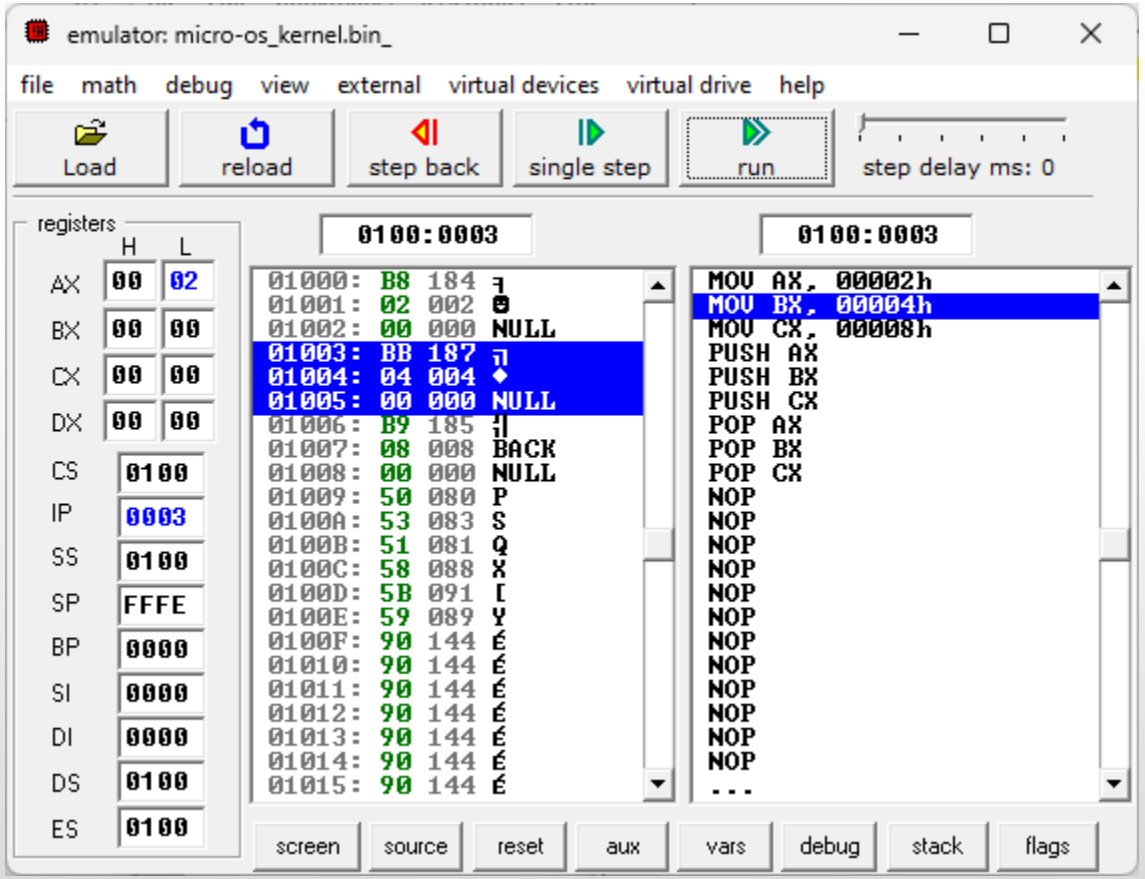
Yığın Kullanımı

```
mov ax, 2
mov bx, 4
mov cx, 8
push ax
push bx
push cx
pop  ax
pop  bx
pop  cx
```

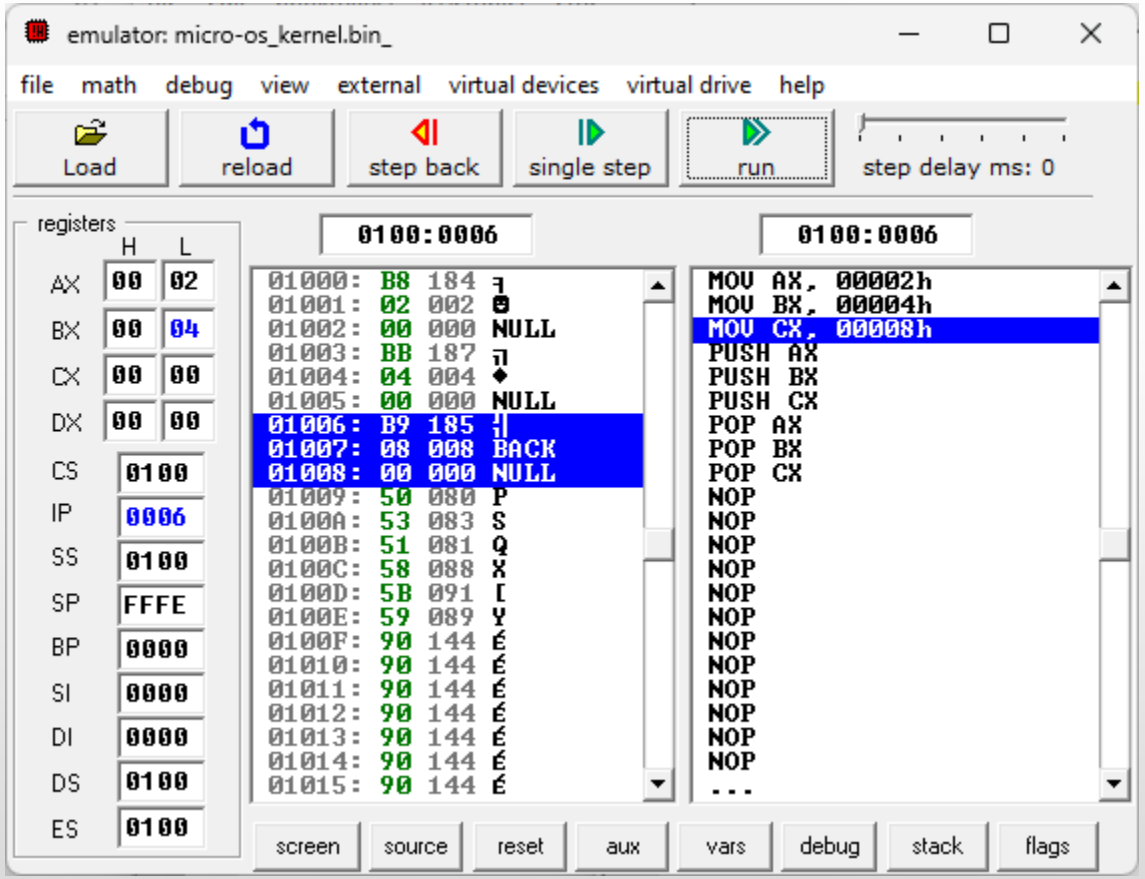
Yığın Kullanımı



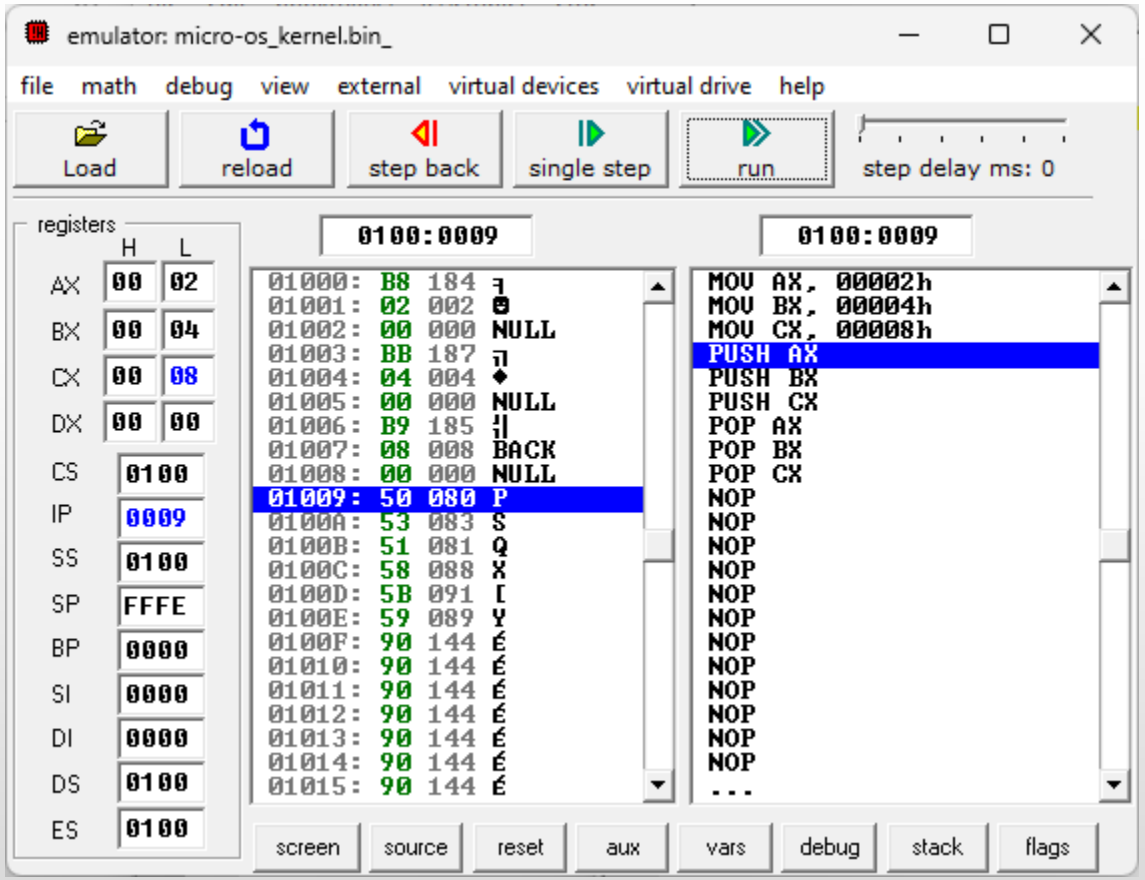
Yığın Kullanımı



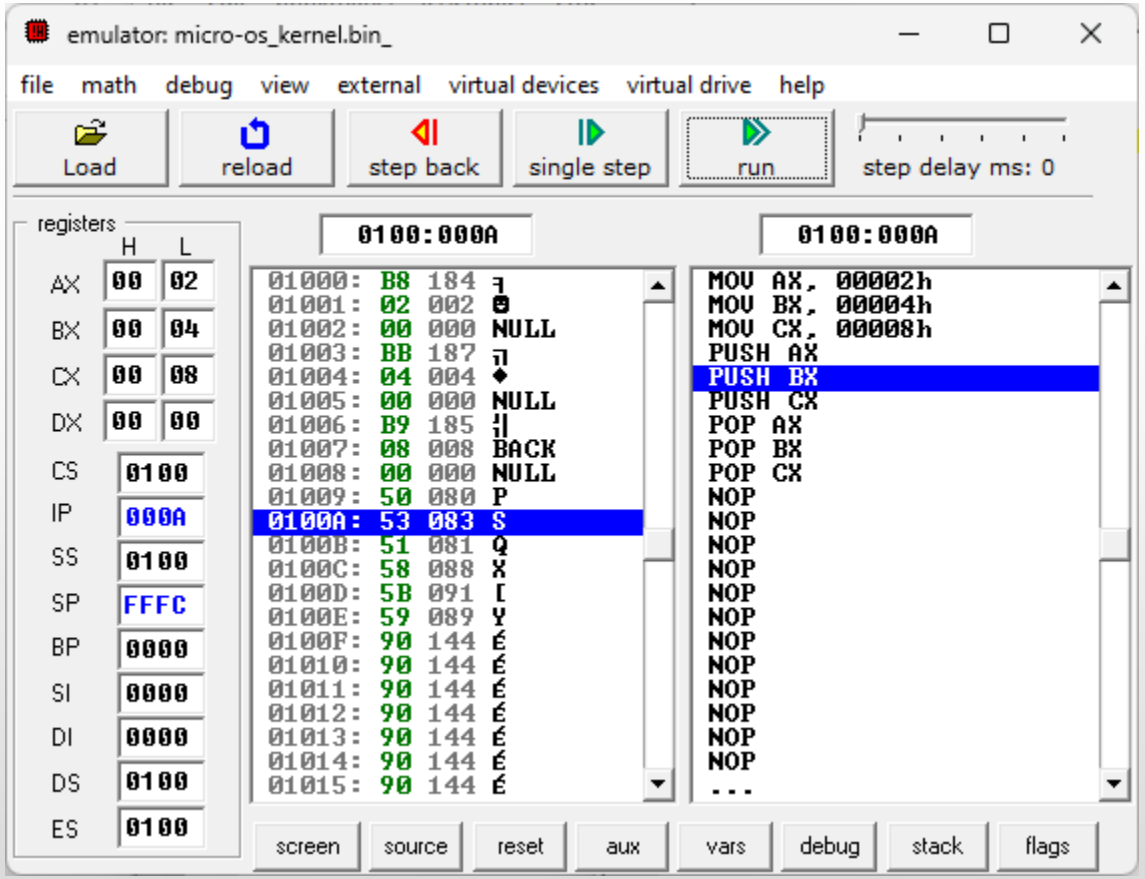
Yığın Kullanımı



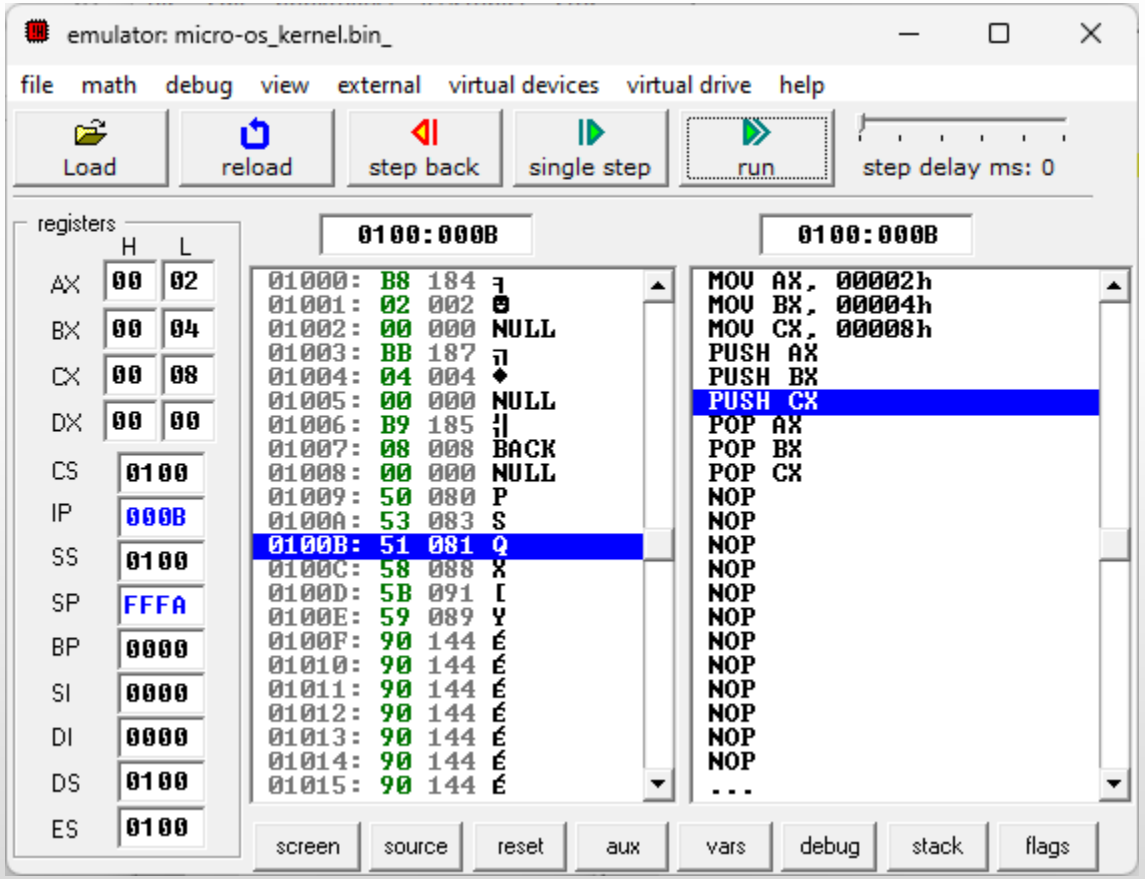
Yığın Kullanımı



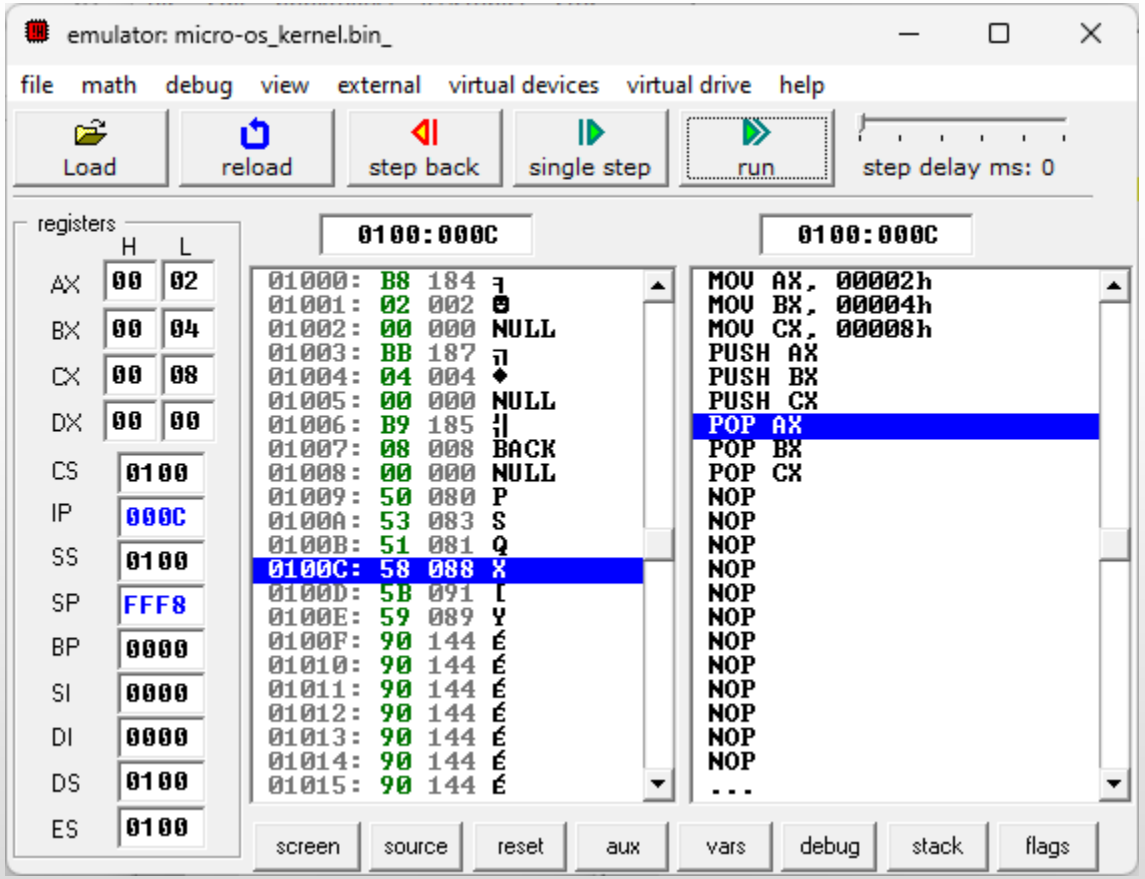
Yığın Kullanımı



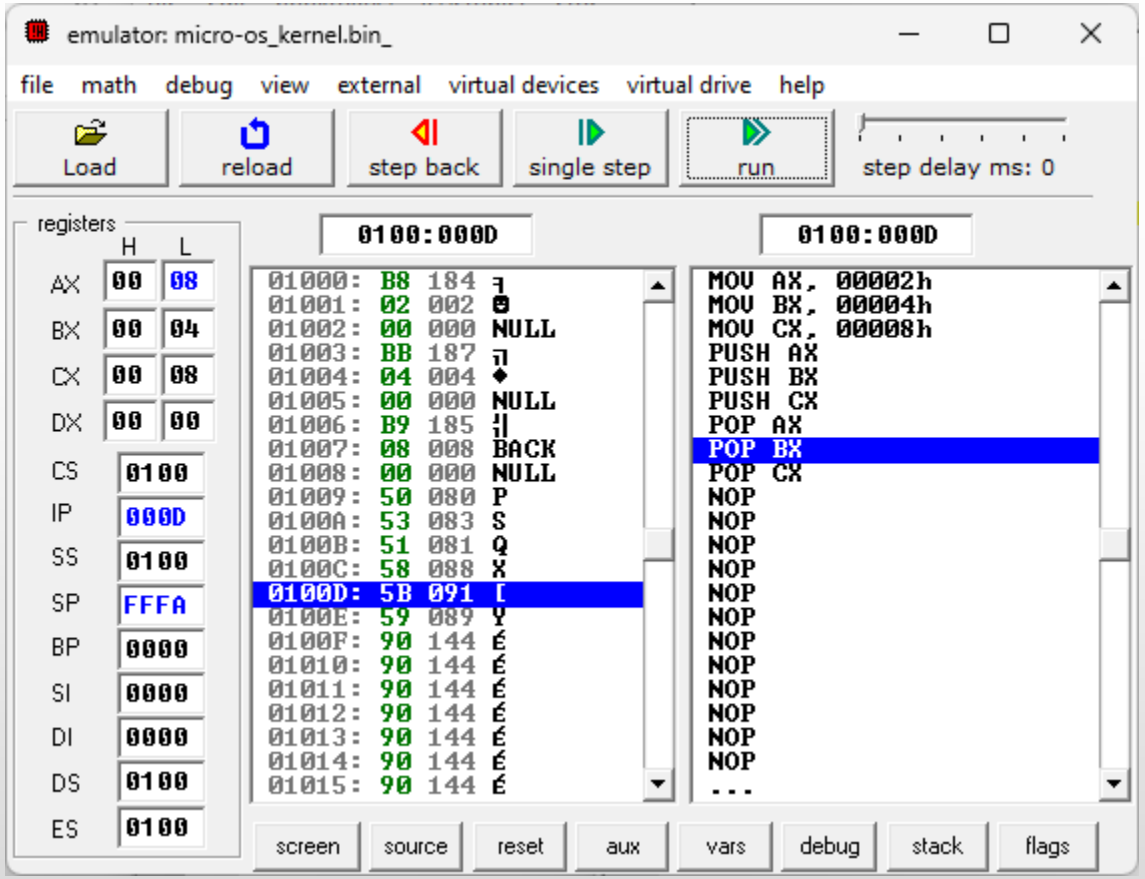
Yığın Kullanımı



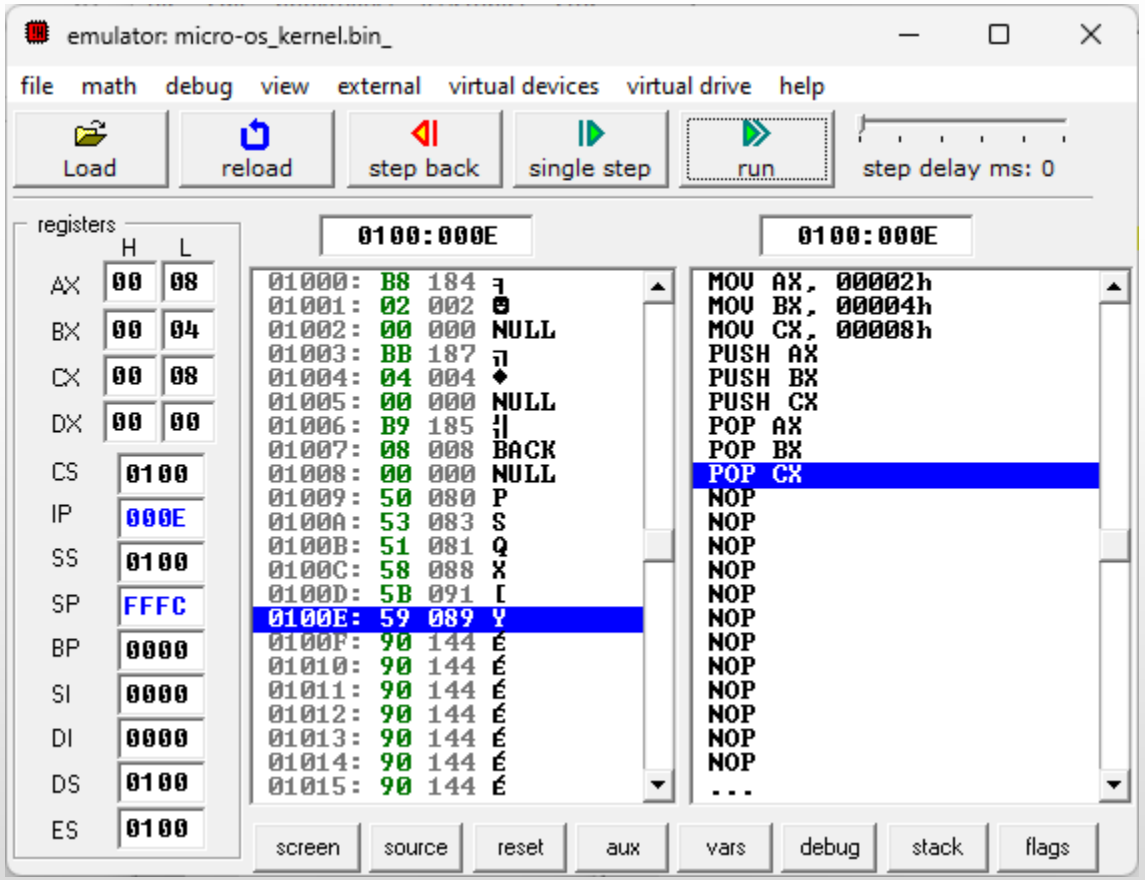
Yığın Kullanımı



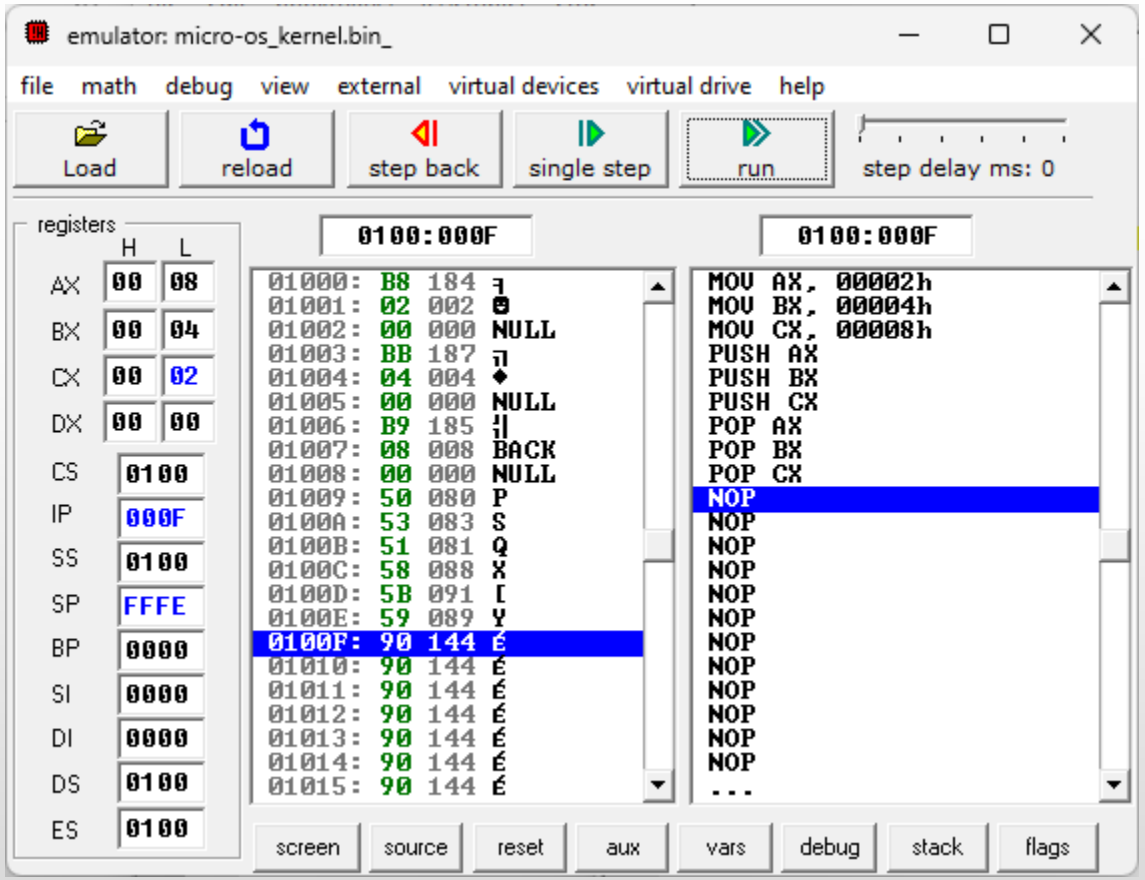
Yığın Kullanımı



Yığın Kullanımı



Yığın Kullanımı



SON