



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 ..



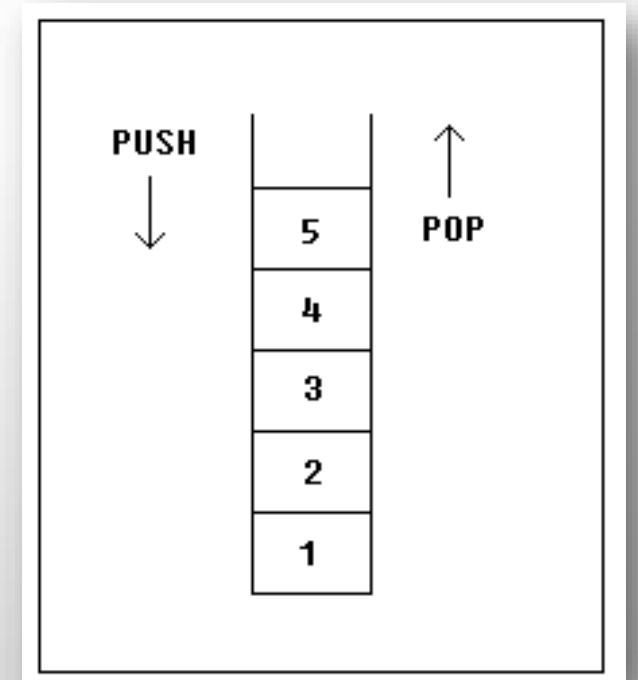
POP Komutu

- 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 öge, yığından çıkan ilk öge 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

emulator: micro-os_kernel.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	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100

07100:	B8	184	↓
07101:	12	018	↓
07102:	12	018	↓
07103:	BB	187	↓
07104:	34	052	4
07105:	34	052	4
07106:	50	080	P
07107:	53	083	S
07108:	58	088	X
07109:	5B	091	I
0710A:	C3	195	↓
0710B:	90	144	E
0710C:	90	144	E
0710D:	90	144	E
0710E:	90	144	E
0710F:	90	144	E
07110:	90	144	E
07111:	90	144	E
07112:	90	144	E
07113:	90	144	E
07114:	90	144	E
07115:	90	144	E

0700:0100

```
MOV AX, 01212h
MOV BX, 03434h
PUSH AX
PUSH BX
POP AX
POP BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Örnek Kod Parçası

emulator: micro-os_kernel.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	12	12
BX	34	34
CX	00	0B
DX	00	00
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0106

07100:	B8	184	3
07101:	12	018	3
07102:	12	018	3
07103:	BB	187	7
07104:	34	052	4
07105:	34	052	4
07106:	50	080	P
07107:	53	083	S
07108:	58	088	X
07109:	5B	091	[
0710A:	C3	195	!
0710B:	90	144	E
0710C:	90	144	E
0710D:	90	144	E
0710E:	90	144	E
0710F:	90	144	E
07110:	90	144	E
07111:	90	144	E
07112:	90	144	E
07113:	90	144	E
07114:	90	144	E
07115:	90	144	E

0700:0106

```
MOV AX, 01212h
MOV BX, 03434h
PUSH AX
PUSH BX
POP AX
POP BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Örnek Kod Parçası

emulator: micro-os_kernel.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	12	12
BX	34	34
CX	00	0B
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0108

07100:	B8	184	3
07101:	12	018	3
07102:	12	018	3
07103:	BB	187	7
07104:	34	052	4
07105:	34	052	4
07106:	50	080	P
07107:	53	083	S
07108:	58	088	X
07109:	5B	091	L
0710A:	C3	195	1
0710B:	90	144	E
0710C:	90	144	E
0710D:	90	144	E
0710E:	90	144	E
0710F:	90	144	E
07110:	90	144	E
07111:	90	144	E
07112:	90	144	E
07113:	90	144	E
07114:	90	144	E
07115:	90	144	E

MOV AX, 01212h
MOV BX, 03434h
PUSH AX
PUSH BX
POP AX
POP BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...

screen source reset aux vars debug stack flags



Örnek Kod Parçası

emulator: micro-os_kernel.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	34	34
BX	34	34
CX	00	0B
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFC	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0109

07100:	B8	184	f
07101:	12	018	+
07102:	12	018	+
07103:	BB	187	n
07104:	34	052	4
07105:	34	052	4
07106:	50	080	P
07107:	53	083	S
07108:	58	088	X
07109:	5B	091	L
0710A:	C3	195	T
0710B:	90	144	E
0710C:	90	144	E
0710D:	90	144	E
0710E:	90	144	E
0710F:	90	144	E
07110:	90	144	E
07111:	90	144	E
07112:	90	144	E
07113:	90	144	E
07114:	90	144	E
07115:	90	144	E

0700:0109

```
MOV AX, 01212h
MOV BX, 03434h
PUSH AX
PUSH BX
POP AX
POP BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Örnek Kod Parçası

emulator: micro-os_kernel.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	34	34
BX	12	12
CX	00	0B
DX	00	00
CS	07 00	
IP	01 0A	
SS	07 00	
SP	FF FE	
BP	00 00	
SI	00 00	
DI	00 00	
DS	07 00	
ES	07 00	

07 00: 01 0A

07100:	B8	184	3
07101:	12	018	4
07102:	12	018	4
07103:	BB	187	7
07104:	34	052	4
07105:	34	052	4
07106:	50	080	P
07107:	53	083	S
07108:	58	088	X
07109:	5B	091	L
0710A:	C3	195	1
0710B:	90	144	E
0710C:	90	144	E
0710D:	90	144	E
0710E:	90	144	E
0710F:	90	144	E
07110:	90	144	E
07111:	90	144	E
07112:	90	144	E
07113:	90	144	E
07114:	90	144	E
07115:	90	144	E

MOV AX, 01212h
MOV BX, 03434h
PUSH AX
PUSH BX
POP AX
POP BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...

screen source reset aux vars debug stack flags



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.

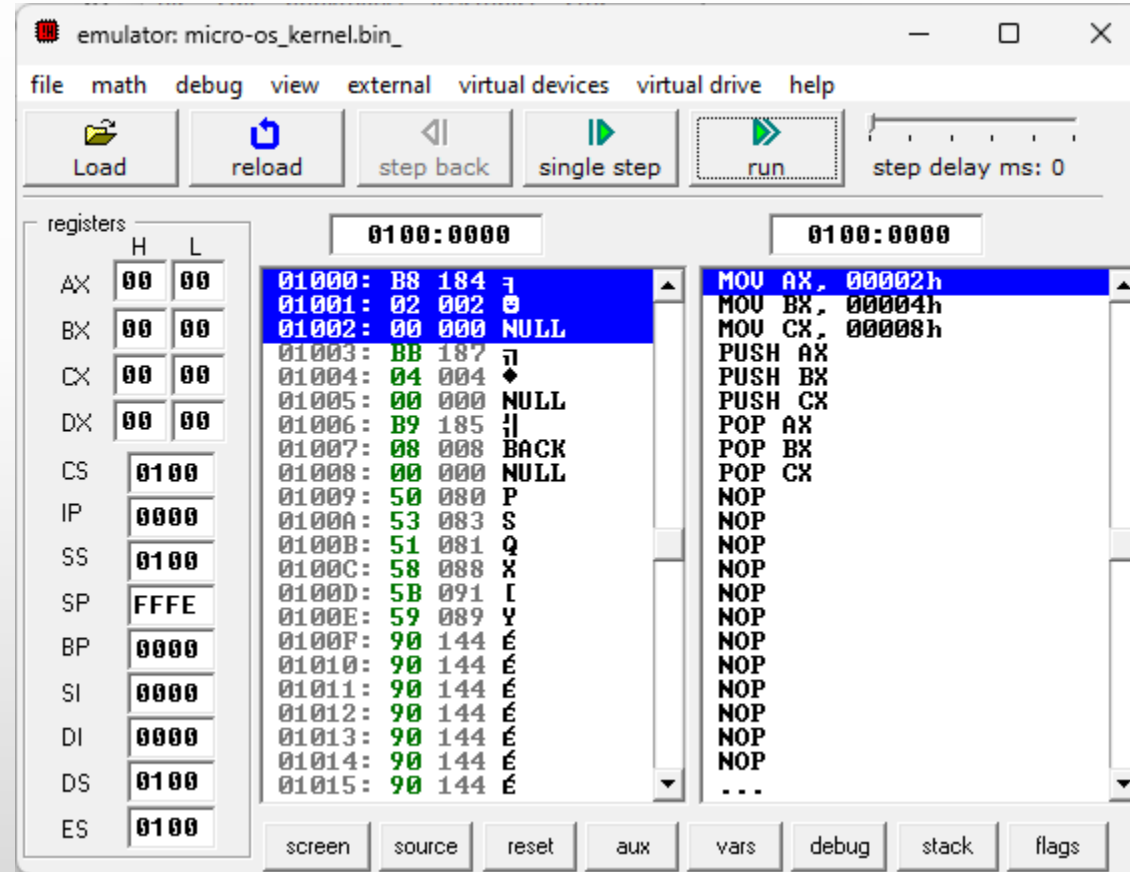


Yığın Kullanımı

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

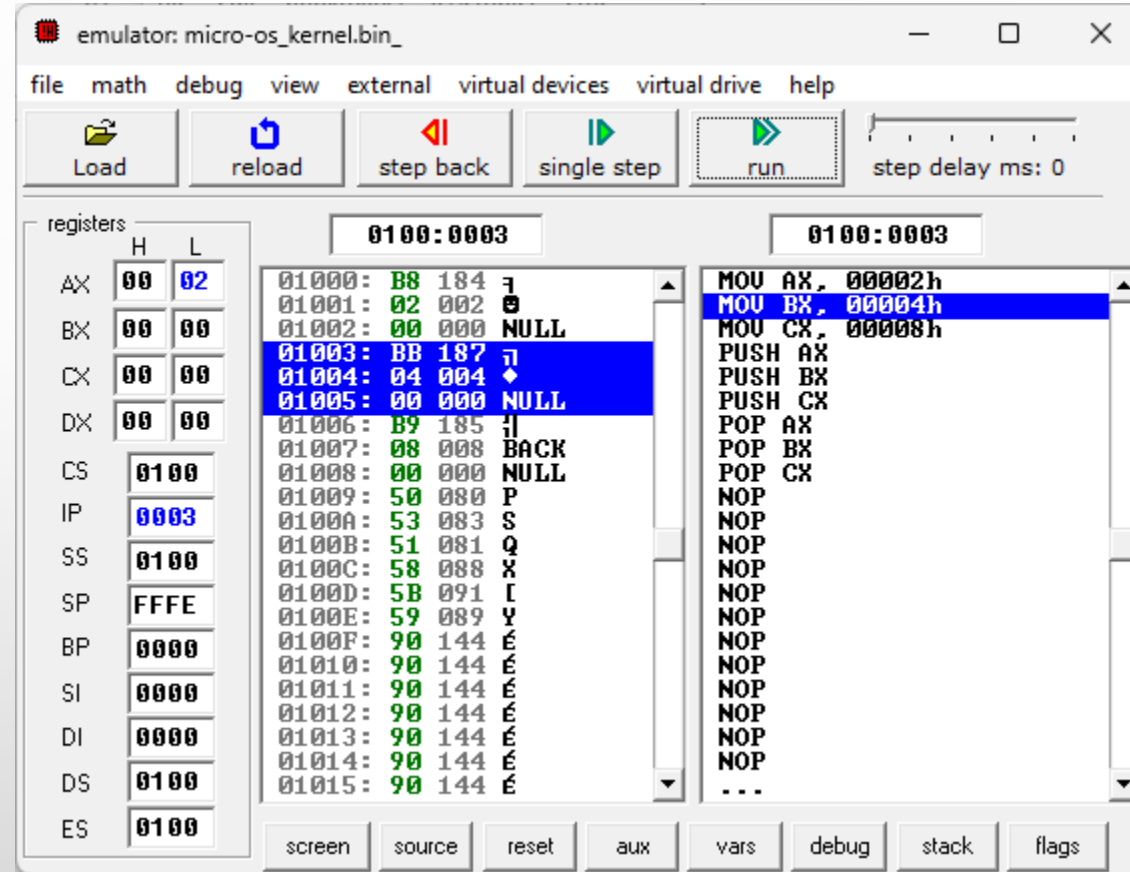


Yığın Kullanımı



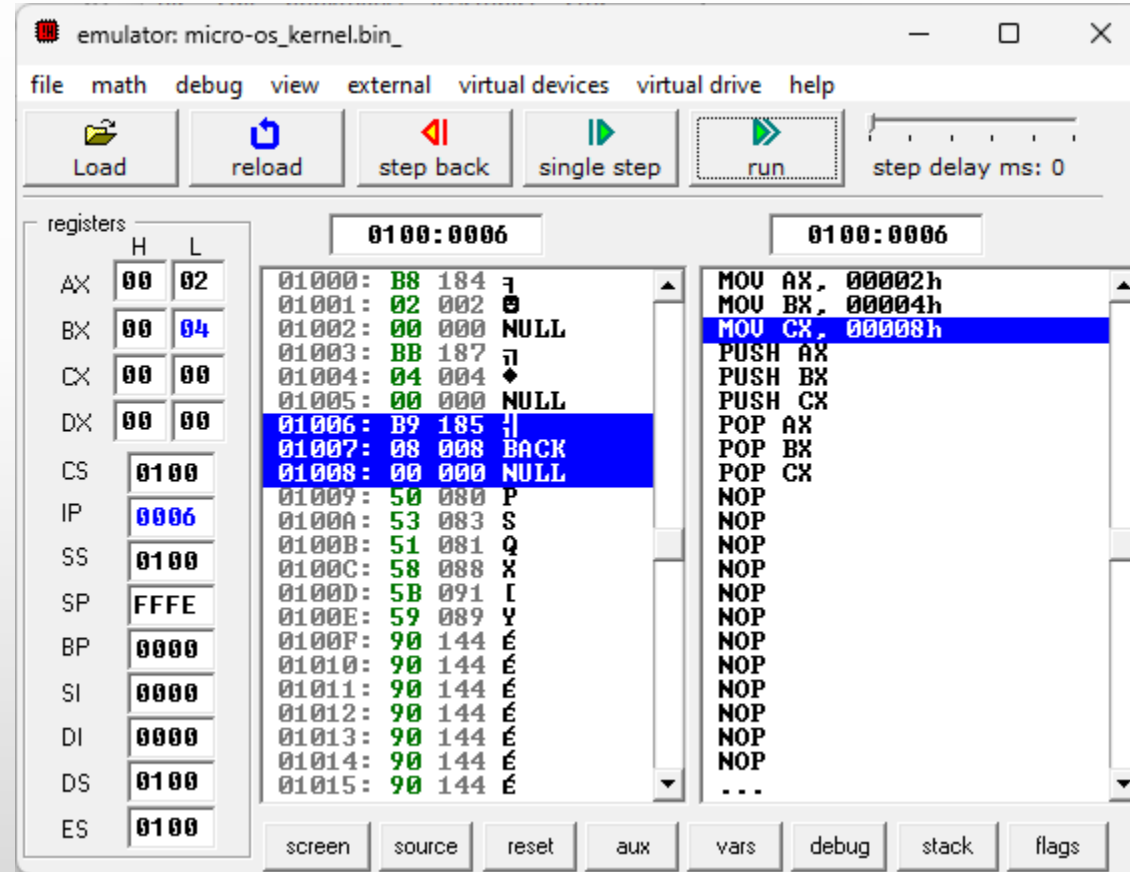


Yığın Kullanımı



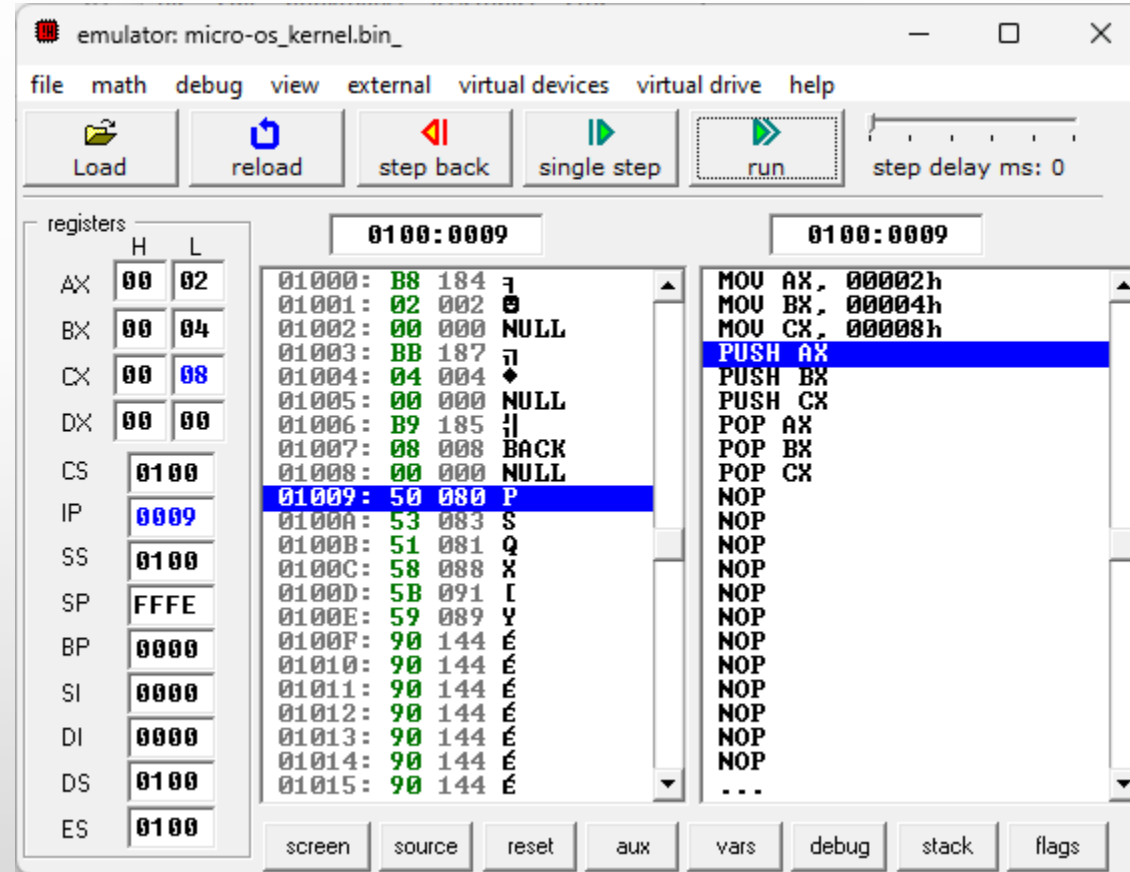


Yığın Kullanımı



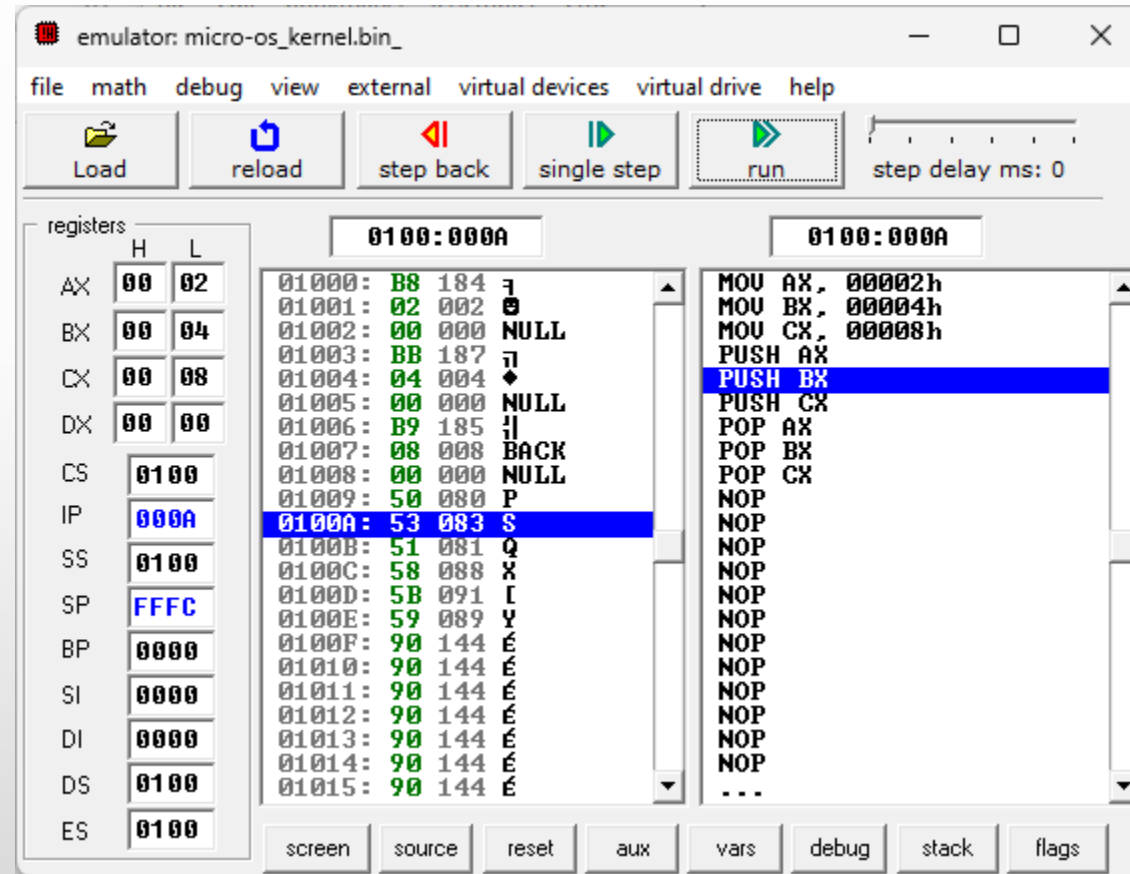


Yığın Kullanımı

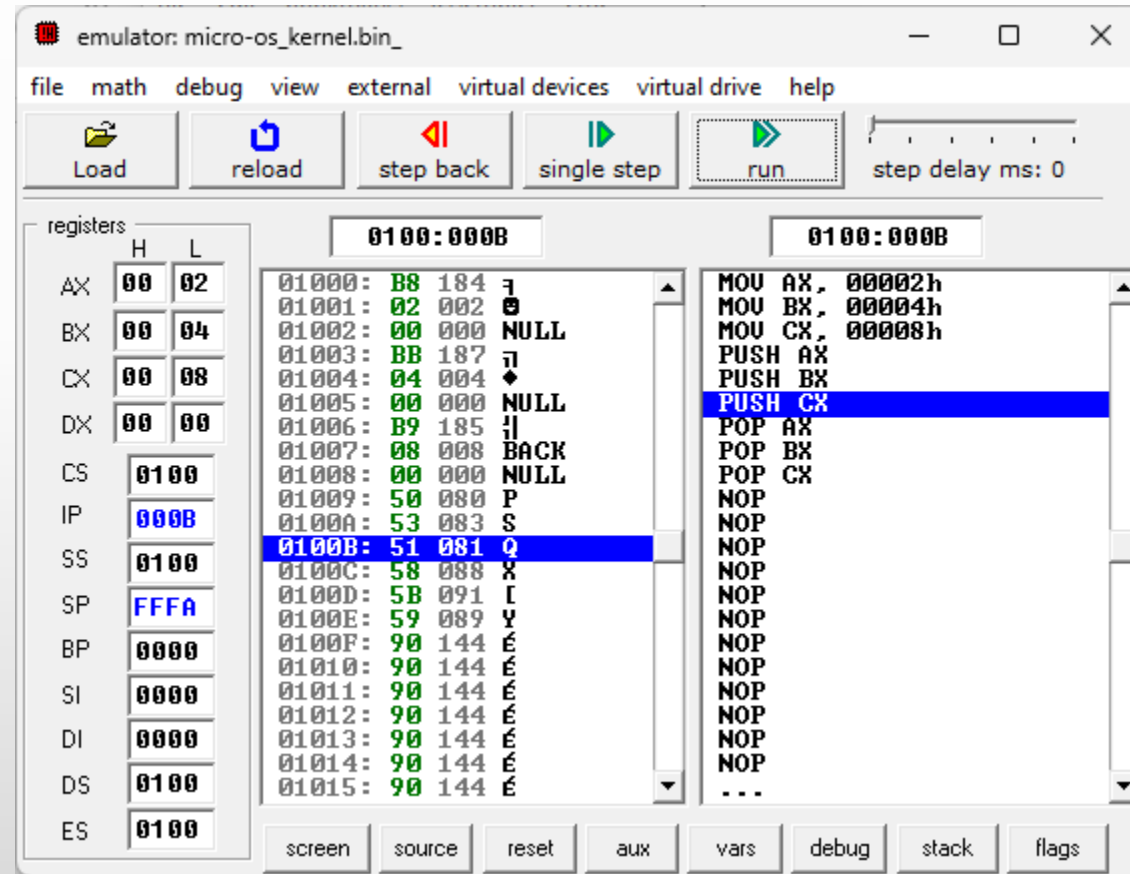




Yığın Kullanımı

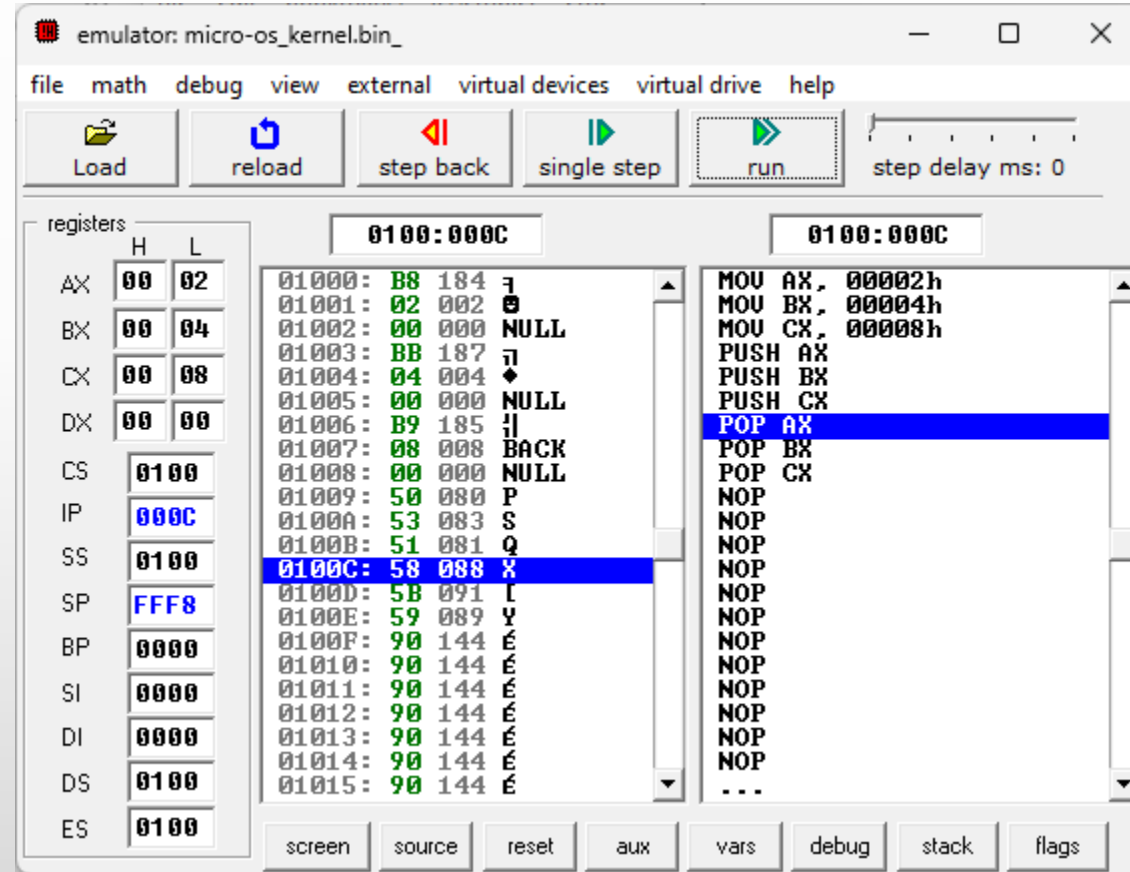


Yığın Kullanımı



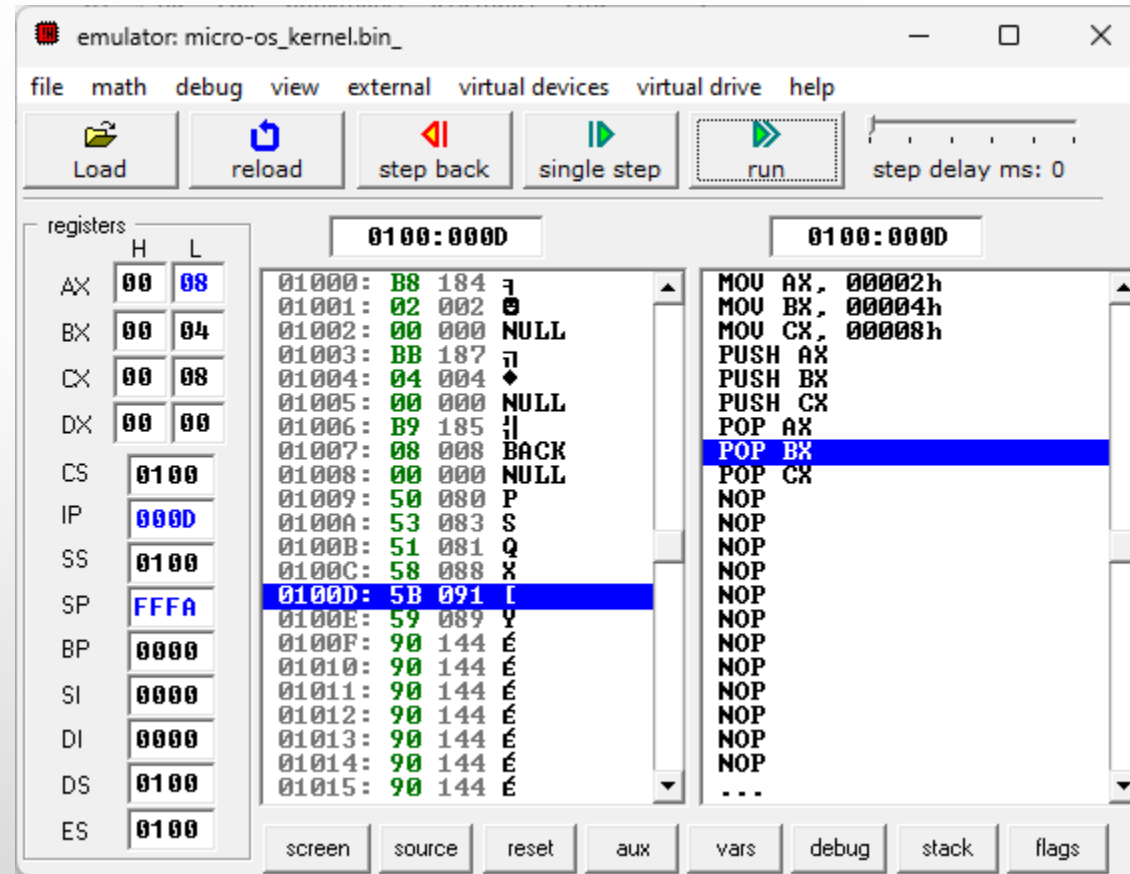


Yığın Kullanımı

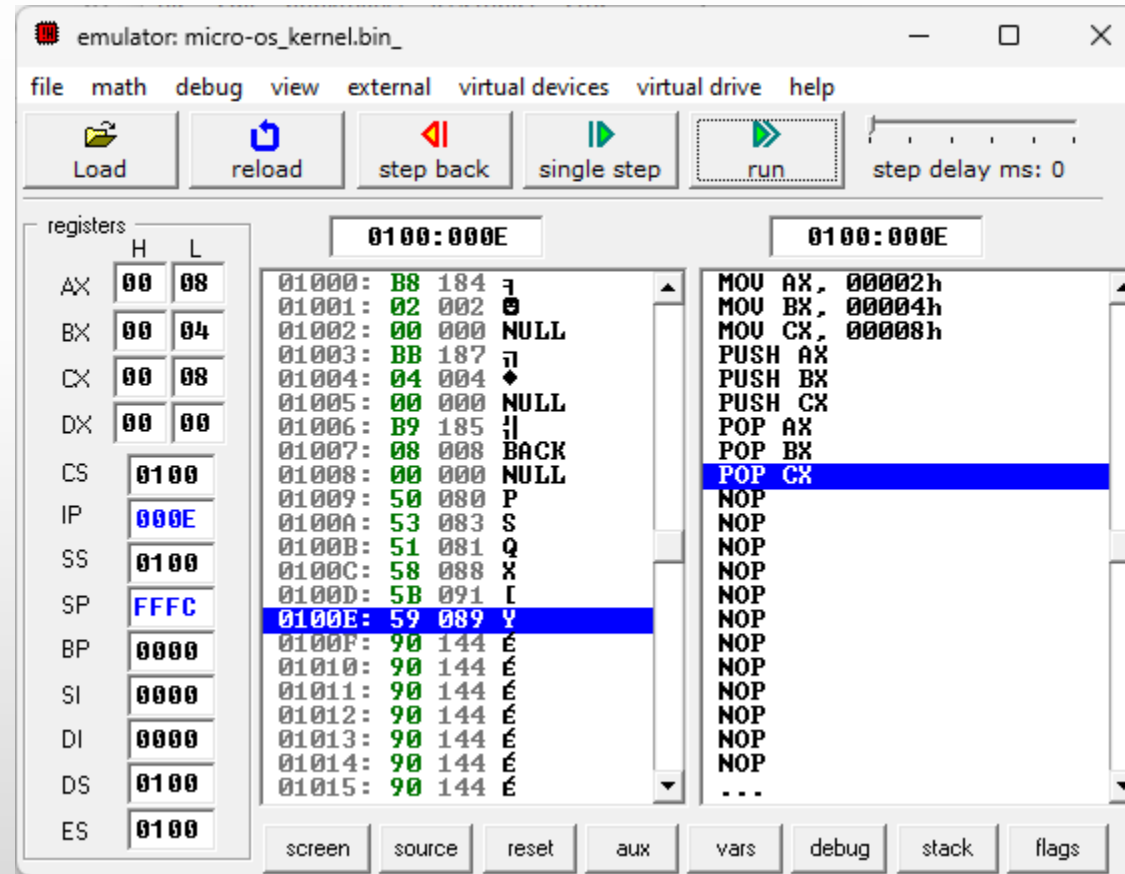




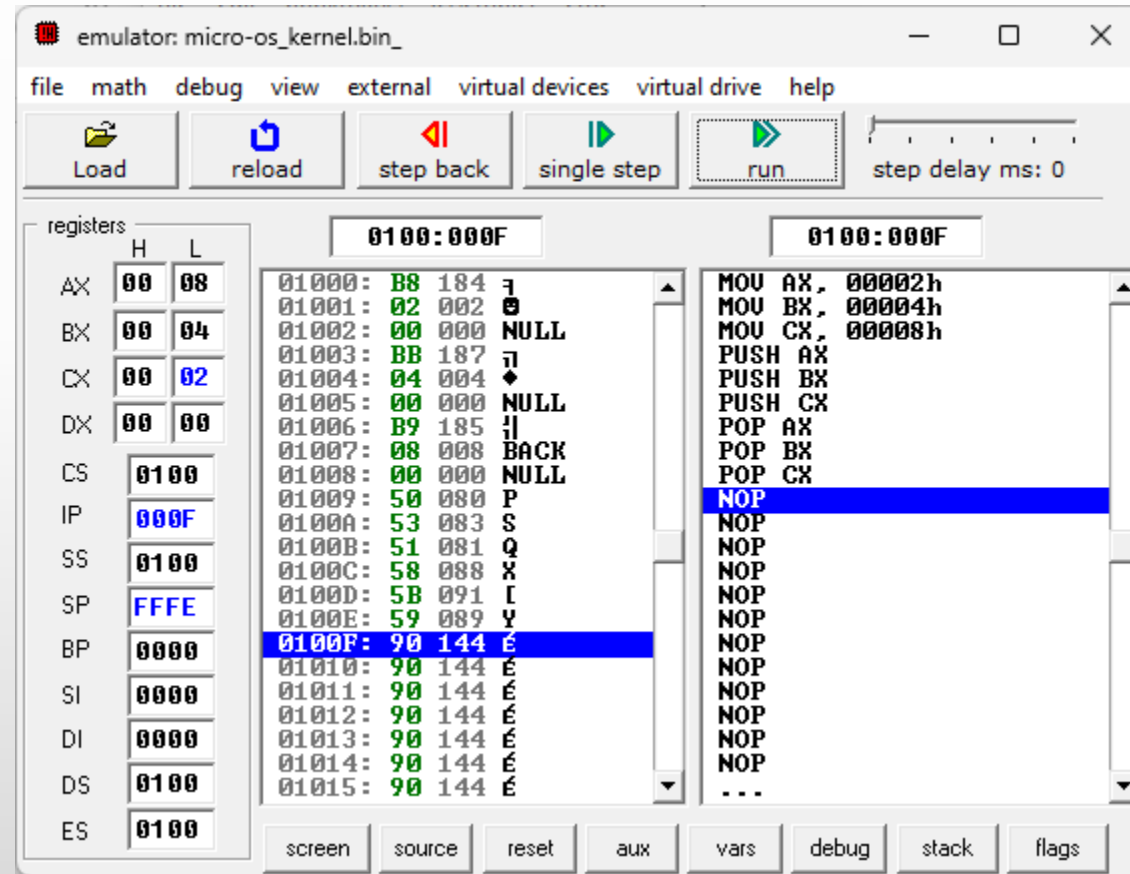
Yığın Kullanımı



Yığın Kullanımı



Yığın Kullanımı





SON