



Bölüm 7: Akış Kontrol

Mikroişlemciler



Program Akış Kontrolü

- Koşulsuz atlamalar (*jumps*)
- Koşullu atlamalar (*jumps*)
 - Tek bir bayrağı test eden atlama komutları
 - İşaretli sayılar için atlama komutları
 - İşaretsiz sayılar için atlama komutları
- Döngüler (*loops*)



Koşulsuz Atlamalar

- JMP (Jump): program içinde kontrolü başka bir noktaya aktarır.
 - JMP etiket
- Etiket tanımlamak için adı yazılır ve sonuna ":" eklenir.
- Etiket herhangi bir karakter kombinasyonu olabilir,
 - ancak bir sayı ile başlayamaz.
- Etiket, ayrı bir satırda veya başka bir komutun önünde tanımlanabilir.
- JMP, kontrolü hem ileri hem de geri yönlendirebilir.
- Mevcut kod segmenti içinde (65,535 bayt) herhangi bir yere atlayabilir.



Koşulsuz Atlamalar

```
org      100h
mov      ax, 5          ; set ax to 5.
mov      bx, 2          ; set bx to 2.
jmp      calc           ; go to 'calc'.
back:    jmp stop       ; go to 'stop'.
calc:
add      ax, bx         ; add bx to ax.
jmp      back           ; go 'back'.
stop:
ret                          ; return to operating system.
```



Tek Bir Bayrağı Test Eden Atlama Komutları

Instruction	Description	Condition
JZ , JE	Jump if Zero (Equal).	ZF = 1
JC , JB, JNAE	Jump if Carry (Below, Not Above Equal).	CF = 1
JS	Jump if Sign.	SF = 1
JO	Jump if Overflow.	OF = 1
JPE, JP	Jump if Parity Even.	PF = 1
JNZ , JNE	Jump if Not Zero (Not Equal).	ZF = 0
JNC , JNB, JAE	Jump if Not Carry (Not Below, Above Equal).	CF = 0
JNS	Jump if Not Sign.	SF = 0
JNO	Jump if Not Overflow.	OF = 0
JPO, JNP	Jump if Parity Odd (No Parity).	PF = 0



Tek Bir Bayrağı Test Eden Atlama Komutları

- JZ, JE: Sıfıra eşitse atlama yap.
 - Koşul: $ZF = 1$ Zıt Komut: JNZ, JNE
- JC, JB, JNAE: Taşma durumunda atlama yap.
 - Koşul: $CF = 1$ Zıt Komut: JNC, JNB, JAE
- JS: Negatifse atlama yap.
 - Koşul: $SF = 1$ Zıt Komut: JNS
- JO: Taşma durumunda atlama yap.
 - Koşul: $OF = 1$ Zıt Komut: JNO
- JPE, JP: Çiftlik durumunda atlama yap.
 - Koşul: $PF = 1$ Zıt Komut: JPO



Tek Bir Bayrağı Test Eden Atlama Komutları

- Atlama komutları sabit uzunluktadır (iki bayt).
- Bağıl konum (*Offset*) 1 baytta saklanır.
 - -128 bayt geriye veya 127 bayt ileriye atlama yapabilir.
- Değer her zaman işaretli bir sayıdır.
- JE, JZ; JNE, JNZ ile aynı makine koduna derlenir.
- JC, JB, JNAE; JNC, JNB, JAE ile aynı makine koduna derlenir.



Tek Bir Bayrağı Test Eden Atlama Komutları

`jnc a`

`jnb a`

`jae a`

`mov ax, 4`

`a: mov ax, 5`

`ret`



İşaretili Sayılar İçin Atlama Komutları

Instruction	Description	Condition
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0
JG , JNLE	Jump if Greater (>). Jump if Not Less or Equal (not <=).	ZF = 0 and SF = OF
JL , JNGE	Jump if Less (<). Jump if Not Greater or Equal (not >=).	SF <> OF
JGE , JNL	Jump if Greater or Equal (>=). Jump if Not Less (not <).	SF = OF
JLE , JNG	Jump if Less or Equal (<=). Jump if Not Greater (not >).	ZF = 1 or SF <> OF



İşaretili Sayılar İçin Atlama Komutları

- JE, JZ: Eşitse atlama yap.
 - Koşul: $ZF = 1$ Zıt Komut: JNE, JNZ
- JNE, JNZ: Eşit değilse atlama yap.
 - Koşul: $ZF = 0$ Zıt Komut: JE, JZ
- JG, JNLE: Büyüğe atlama yap.
 - Koşul: $ZF = 0$ ve $SF = OF$ Zıt Komut: JNG, JLE
- JL, JNGE: Küçüğe atlama yap.
 - Koşul: $SF \neq OF$ Zıt Komut: JNL, JGE
- JGE, JNL: Büyük veya eşitse atlama yap.
 - Koşul: $SF = OF$ Zıt Komut: JNGE, JL
- JLE, JNG: Küçük veya eşitse atlama yap.
 - Koşul: $ZF = 1$ veya $SF \neq OF$ Zıt Komut: JNLE, JG



İşaretli Sayılar İçin Atlama Komutları

- <> işareti eşit değil anlamına gelir.

```
mov ax, 5
```

```
mov bx, 5
```

```
cmp ax, bx
```

```
je equal_message
```

```
jmp not_equal_message
```

```
equal_message:           ; Eşitse yapılacak işlemler
```

```
jmp end_program
```

```
not_equal_message:       ; Eşit değilse yapılacak işlemler
```

```
end_program:
```



İşaretsiz Sayılar İçin Atlama Komutları

Instruction	Description	Condition
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0
JA , JNBE	Jump if Above (>). Jump if Not Below or Equal (not <=).	CF = 0 and ZF = 0
JB , JNAE, JC	Jump if Below (<). Jump if Not Above or Equal (not >=). Jump if Carry.	CF = 1
JAE , JNB, JNC	Jump if Above or Equal (>=). Jump if Not Below (not <). Jump if Not Carry.	CF = 0
JBE , JNA	Jump if Below or Equal (<=). Jump if Not Above (not >).	CF = 1 or ZF = 1



İşaretsiz Sayılar İçin Atlama Komutları

- JE, JZ: Eşitse atlama yap.
 - Koşul: $ZF = 1$ Zıt Komut: JNE, JNZ
- JNE, JNZ: Eşit değilse atlama yap.
 - Koşul: $ZF = 0$ Zıt Komut: JE, JZ
- JA, JNBE: Büyükse atlama yap.
 - Koşul: $CF = 0$ ve $ZF = 0$ Zıt Komut: JNA, JBE
- JB, JNAE, JC: Küçükse atlama yap.
 - Koşul: $CF = 1$ Zıt Komut: JNB, JAE, JNC
- JAE, JNB, JNC: Büyük veya eşitse atlama yap.
 - Koşul: $CF = 0$ Zıt Komut: JNAE, JB
- JBE, JNA: Küçük veya eşitse atlama yap.
 - Koşul: $CF = 1$ veya $ZF = 1$ Zıt Komut: JNBE, JA



İşaretsiz Sayılar İçin Atlama Komutları

```
mov ax, 5
mov bx, 7
cmp ax, bx
ja  jump_above
jmp not_jump_above
jump_above:      ; ax büyükse yapılacak işlemler
jmp end_program
not_jump_above:  ; ax küçükse veya eşitse yapılacak işlemler
end_program:
```



CMP ve Atlama Komutları

- Sayısal değerleri karşılaştırmak için CMP (compare) komutu kullanılır.
- CMP komutu, SUB (çıkarma) komutunu gerçekleştirir.
- Örnek 1: 5 ve 2'yi karşılaştır,
 - $5 - 2 = 3$
 - Sonuç sıfır değil (Zero Bayrağına 0 atanır).
- Örnek 2: 7 ve 7'yi karşılaştır,
 - $7 - 7 = 0$
 - Sonuç sıfır! (Zero Bayrağına 1 atanır, JZ veya JE atlama yapar).



CMP ve Atlama Komutları

```
include "emu8086.inc"
org      100h
mov      al, 25      ; set al to 25.
mov      bl, 10      ; set bl to 10.
cmp      al, bl      ; compare al - bl.
je       equal       ; jump if al = bl (zf = 1).
putc     'n'         ; if it gets here, then al <> bl,
jmp      stop        ; so print 'n', and jump to stop.
equal:          ; if gets here,
putc     'y'         ; then al = bl, so print 'y'.
stop:
ret                ; gets here no matter what.
```




Döngüler (Loops)

Instruction	Operation And Jump Condition
LOOP	decrease cx, jump to label if cx not zero.
LOOPE	decrease cx, jump to label if cx not zero and equal (zf = 1).
LOOPNE	decrease cx, jump to label if cx not zero and not equal (zf = 0).
LOOPNZ	decrease cx, jump to label if cx not zero and zf = 0.
LOOPZ	decrease cx, jump to label if cx not zero and zf = 1.
JCXZ	jump to label if cx is zero.



Döngüler (Loops)

- Döngüler, bir koşula bağlı olarak bir kod bloğunun tekrarlanmasını sağlar.
- LOOP: CX sıfır olmadığı sürece belirtilen etikete atlama yapar.
- LOOPE, LOOPZ:
 - CX sıfır olmadığı ve ZF = 1 olduğu sürece etikete atlama yapar.
- LOOPNE, LOOPNZ :
 - CX sıfır olmadığı ve ZF = 0 olduğu sürece etikete atlama yapar.
- JCXZ: CX sıfır olduğunda belirtilen etikete atlama yapar.



Döngüler (Loops)

```
include "emu8086.inc"
```

```
org 100h
```

```
mov cx, 5
```

; CX döngü tekrar sayısı 5 ata.

```
dongu:
```

```
    ; Döngü İçeriği
```

```
    loop dongu
```

; CX sıfır değilse dongu etiketine atla

```
jmp dur
```

; Döngü bittiğinde dur etiketine atla

```
dur:
```

```
    ret
```



Döngüler (Loops)

```
1  org 100h
2  mov bx, 0   ;Toplam adım.
3  mov cx, 5
4  k1:
5      add bx, 1
6      mov al, '1'
7      mov ah, 0eh
8      int 10h
9      push cx
10     mov cx, 5
```

```
11  k2:
12      add bx, 1
13      mov al, '2'
14      mov ah, 0eh
15      int 10h
16      loop k2   ; İç döngü.
17      pop cx
18  loop k1       ; Dış döngü.
19  ret
```



Dizi Elemanları Toplamı

```
org 100h
mov cx, 5 ; eleman sayısı
mov al, 0 ; toplam al yazmacında tutulacak
mov bx, 0 ; bx indis olarak kullanılacak
next: add al, vector[bx] ; elemanları topla
inc bx ; sonraki eleman
loop next ; cx=0 olana kadar dön
mov m, al ; sonucu m değişkenine atar
ret
vector db 5, 4, 5, 2, 1
m db 0
```



Dizi Elemanları Toplamı

emulator: calc-sum.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	19
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100

07100:	B9	185	⌵
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	⌵
07104:	00	000	NULL
07105:	BB	187	⌵
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	⌵
07109:	87	135	⌵
0710A:	13	019	⌵
0710B:	01	001	⌵
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	·
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	⌵
07112:	C3	195	⌵
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:0100

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	05
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0108

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	⊖
07109:	87	135	ç
0710A:	13	019	::
0710B:	01	001	⊖
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	·
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	⊖
07112:	C3	195	↓
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:0108

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	05
BX	00	01
CX	00	05
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100

07100:	B9	185	!
07101:	05	005	⬆
07102:	00	000	NULL
07103:	B0	176	⬆
07104:	00	000	NULL
07105:	BB	187	⬆
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	⬆
07109:	87	135	⬆
0710A:	13	019	!!
0710B:	01	001	⬆
0710C:	43	067	C
0710D:	E2	226	⬆
0710E:	F9	249	-
0710F:	A2	162	⬆
07110:	18	024	↑
07111:	01	001	⬆
07112:	C3	195	⬆
07113:	05	005	⬆
07114:	04	004	⬆
07115:	05	005	⬆

0700:0100

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	09
BX	00	01
CX	00	04
DX	00	00
CS	0700	
IP	010C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:010C

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	///
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	0
07109:	87	135	ç
0710A:	13	019	!!
0710B:	01	001	0
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	-
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	0
07112:	C3	195	†
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:010C

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	0E
BX	00	02
CX	00	03
DX	00	00
CS	0700	
IP	010C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:010C

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	0
07109:	87	135	ç
0710A:	13	019	!!
0710B:	01	001	0
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	-
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	0
07112:	C3	195	†
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:010C

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	10
BX	00	03
CX	00	02
DX	00	00
CS	0700	
IP	010C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:010C

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	///
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	0
07109:	87	135	ç
0710A:	13	019	!!
0710B:	01	001	0
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	-
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	0
07112:	C3	195	†
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:010C

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	11
BX	00	04
CX	00	01
DX	00	00
CS	0700	
IP	010C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:010C

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	///
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	0
07109:	87	135	ç
0710A:	13	019	!!
0710B:	01	001	0
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	.
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	0
07112:	C3	195	†
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:010C

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Dizi Elemanları Toplamı

emulator: calc-sum.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	11
BX	00	05
CX	00	00
DX	00	00
CS	0700	
IP	0112	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0112

07100:	B9	185	!
07101:	05	005	♣
07102:	00	000	NULL
07103:	B0	176	
07104:	00	000	NULL
07105:	BB	187	7
07106:	00	000	NULL
07107:	00	000	NULL
07108:	02	002	0
07109:	87	135	ç
0710A:	13	019	!!
0710B:	01	001	0
0710C:	43	067	C
0710D:	E2	226	Γ
0710E:	F9	249	.
0710F:	A2	162	ó
07110:	18	024	↑
07111:	01	001	0
07112:	C3	195	†
07113:	05	005	♣
07114:	04	004	♦
07115:	05	005	♣

0700:0112

```
MOV CX, 00005h
MOV AL, 00h
MOV BX, 00000h
ADD AL, [BX] + 00113h
INC BX
LOOP 0108h
MOV [00118h], AL
RET
ADD AX, 00504h
ADD AL, [BX + DI]
ADC [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Ekrana Hello Yazdırma

```
jmp    start
msg db 'Hello', 0
start:
    mov    si, 0            ; source index'i ayarla
next_char:
    mov    al, msg[si]      ; yazdırılacak karakteri al
    cmp    al, 0            ; 0 ile karşılaştır
    je     stop             ; 0 ise yazmayı durdur
    mov    ah, 0eh          ; karakteri yazdır
    int    10h
    inc    si               ; source index'i güncelle
    jmp    next_char        ; diğer karakteri yazdırmaya geç
stop:  mov    ah, 0          ; klavye tuşuna basılmasını bekle
    int    16h
```



Ekрана Hello Yazdırma

emulator: hello.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	00
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0000	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

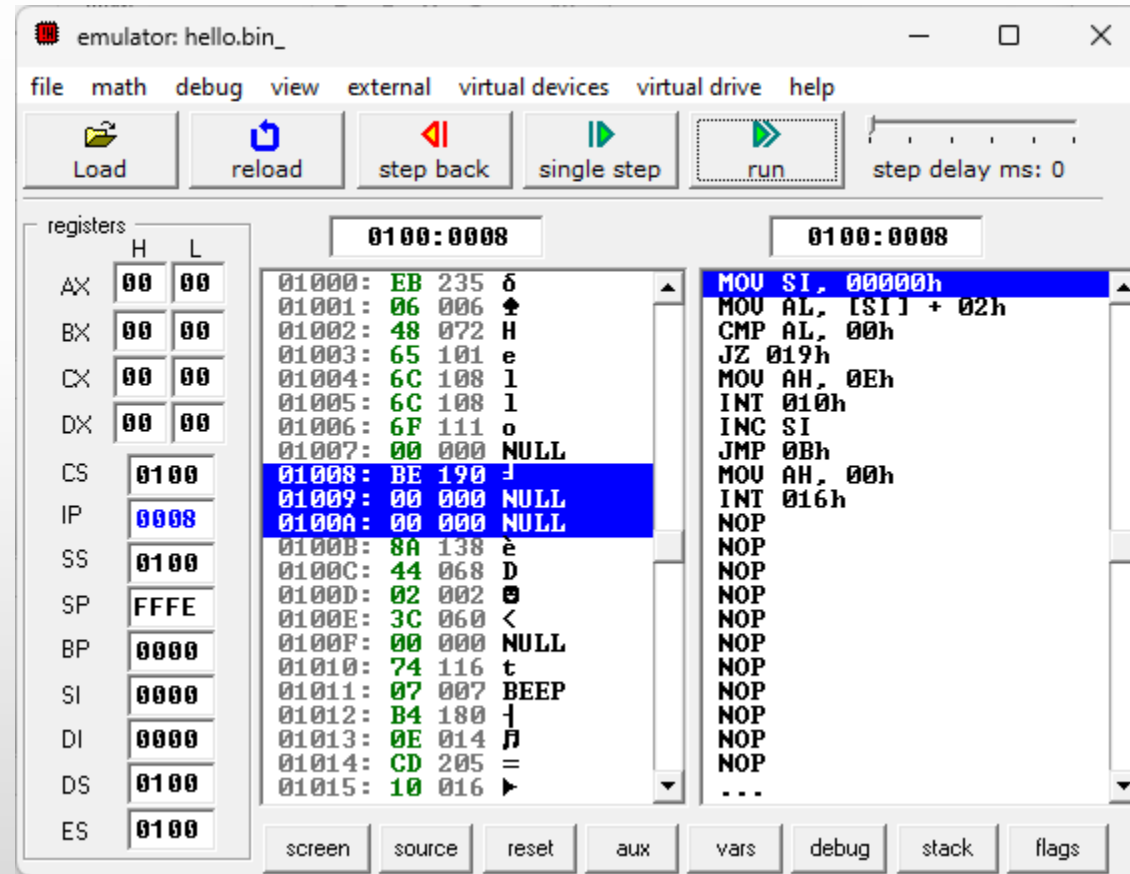
0100:0000 0100:0000

01000: EB 235 6	JMP 08h
01001: 06 006 4	DEC AX
01002: 48 072 H	DB 65h
01003: 65 101 e	INSB
01004: 6C 108 1	INSB
01005: 6C 108 1	OUTSW
01006: 6F 111 o	ADD [BP1 + 00000h], BH
01007: 00 000 NULL	MOV AL, [SI] + 02h
01008: BE 190 d	CMP AL, 00h
01009: 00 000 NULL	JZ 019h
0100A: 00 000 NULL	MOV AH, 0Eh
0100B: 8A 138 è	INT 010h
0100C: 44 068 D	INC SI
0100D: 02 002 0	JMP 0Bh
0100E: 3C 060 <	MOV AH, 00h
0100F: 00 000 NULL	INT 016h
01010: 74 116 t	NOP
01011: 07 007 BEEP	NOP
01012: B4 180 i	NOP
01013: 0E 014 j	NOP
01014: CD 205 =	NOP
01015: 10 016 ►	...

screen source reset aux vars debug stack flags



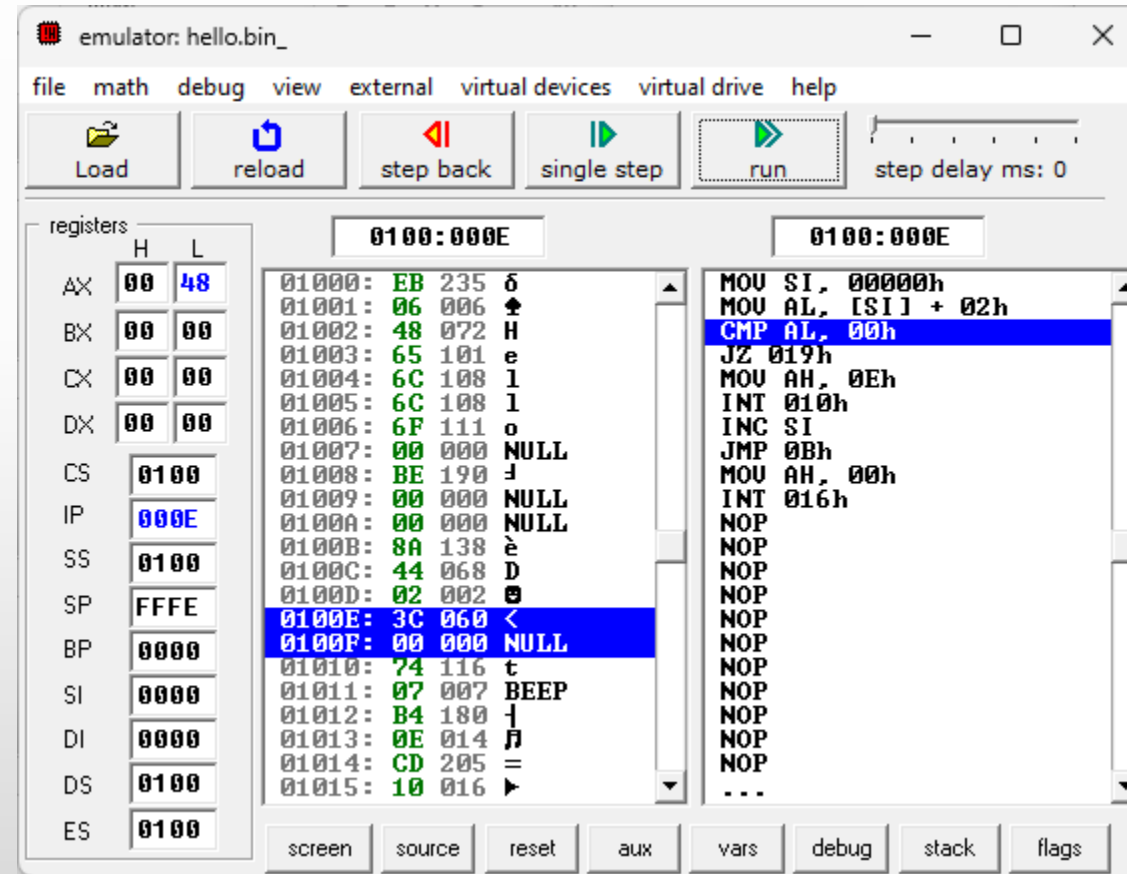
Ekrana Hello Yazdırma







Ekрана Hello Yazdırma





Ekрана Hello Yazdırma

emulator: hello.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	48
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0010	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:0010

01000:	EB	235	6
01001:	06	006	↑
01002:	48	072	H
01003:	65	101	e
01004:	6C	108	l
01005:	6C	108	l
01006:	6F	111	o
01007:	00	000	NULL
01008:	BE	190	d
01009:	00	000	NULL
0100A:	00	000	NULL
0100B:	8A	138	è
0100C:	44	068	D
0100D:	02	002	2
0100E:	3C	060	<
0100F:	00	000	NULL
01010:	74	116	t
01011:	07	007	BEEP
01012:	B4	180	↓
01013:	0E	014	¶
01014:	CD	205	=
01015:	10	016	▶

0100:0010

```
MOV SI, 00000h
MOV AL, [SI] + 02h
CMP AL, 00h
JZ 019h
MOV AH, 0Eh
INT 010h
INC SI
JMP 0Bh
MOV AH, 00h
INT 016h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Ekрана Hello Yazdırma

emulator: hello.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	48
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0012	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:0012

```
01000: EB 235 6
01001: 06 006
01002: 48 072 H
01003: 65 101 e
01004: 6C 108 l
01005: 6C 108 l
01006: 6F 111 o
01007: 00 000 NULL
01008: BE 190 d
01009: 00 000 NULL
0100A: 00 000 NULL
0100B: 8A 138 è
0100C: 44 068 D
0100D: 02 002 2
0100E: 3C 060 <
0100F: 00 000 NULL
01010: 74 116 t
01011: 07 007 BEEP
01012: B4 180 |
01013: 0E 014 n
01014: CD 205 =
01015: 10 016 ►
```

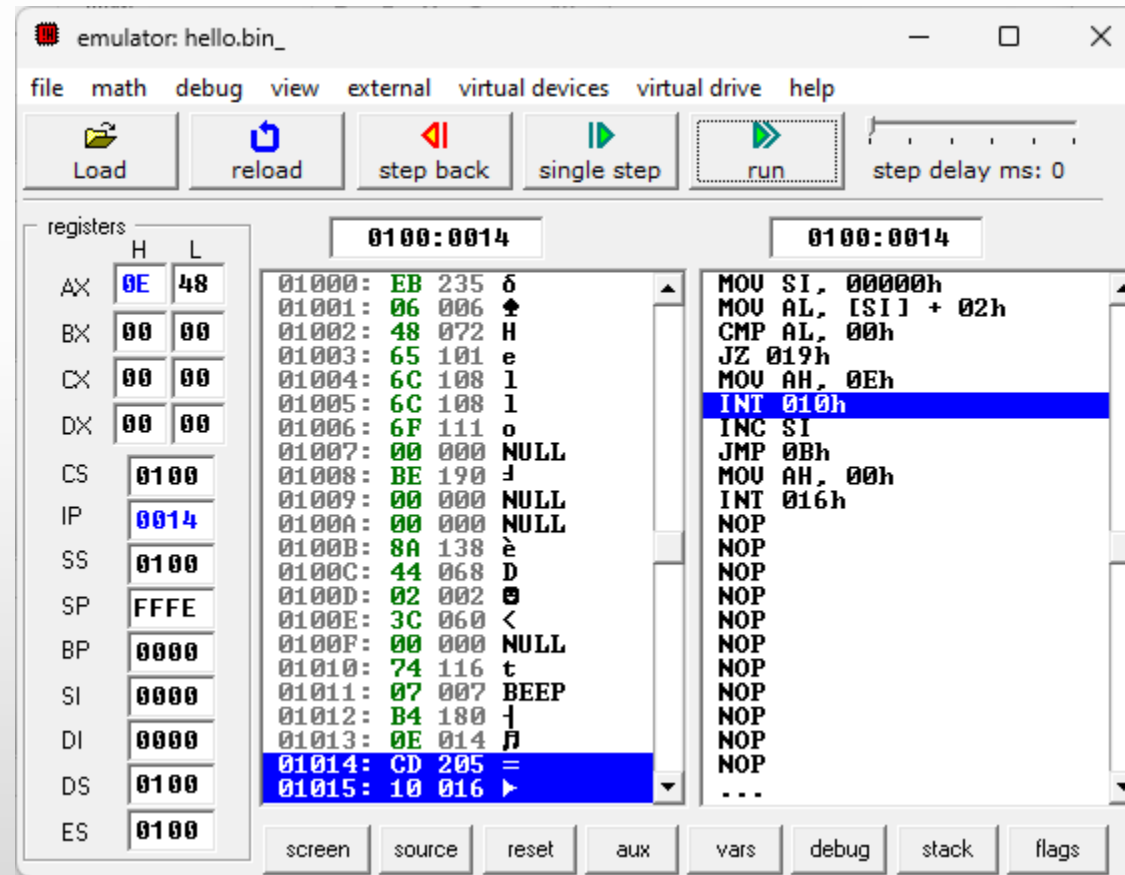
0100:0012

```
MOV SI, 00000h
MOV AL, [SI] + 02h
CMP AL, 00h
JZ 019h
MOV AH, 0Eh
INT 010h
INC SI
JMP 0Bh
MOV AH, 00h
INT 016h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



Ekрана Hello Yazdırma





Ekrana Hello Yazdırma

emulator: hello.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	0E	48
BX	00	00
CX	00	00
DX	00	00
CS	F400	
IP	0190	
SS	0100	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

F400:0190 F400:0190

F4190: FF 255 RES
F4191: FF 255 RES
F4192: CD 205 =
F4193: 10 016 >
F4194: CF 207 ±
F4195: 00 000 NULL
F4196: 00 000 NULL
F4197: 00 000 NULL
F4198: 00 000 NULL
F4199: 00 000 NULL
F419A: 00 000 NULL
F419B: 00 000 NULL
F419C: 00 000 NULL
F419D: 00 000 NULL
F419E: 00 000 NULL
F419F: 00 000 NULL
F41A0: FF 255 RES
F41A1: FF 255 RES
F41A2: CD 205 =
F41A3: 12 018 ‡
F41A4: CF 207 ±
F41A5: 00 000 NULL

BIOS DI
INT 010h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADC CL, BH
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADC CX, DI
ADD [BX + SI], AL
ADD [BX + SI], AL
...

screen source reset aux vars debug stack flag

emulator screen (80x25 chars)

Hello

clear screen change font 0/16





Ekрана Hello Yazdırma

emulator: hello.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	0E	00
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	000E	
SS	0100	
SP	FFFE	
BP	0000	
SI	0005	
DI	0000	
DS	0100	
ES	0100	

0100:000E

0100B: 8A 138	è
0100C: 44 068	D
0100D: 02 002	2
0100E: 3C 060	<
0100F: 00 000	NULL
01010: 74 116	t
01011: 07 007	BEEP
01012: B4 180	
01013: 0E 014	µ
01014: CD 205	=
01015: 10 016	►
01016: 46 070	F
01017: EB 235	δ
01018: F2 242	≥
01019: B4 180	
0101A: 00 000	NULL
0101B: CD 205	=
0101C: 16 022	■
0101D: 90 144	é
0101E: 90 144	é
0101F: 90 144	é
01020: 90 144	é

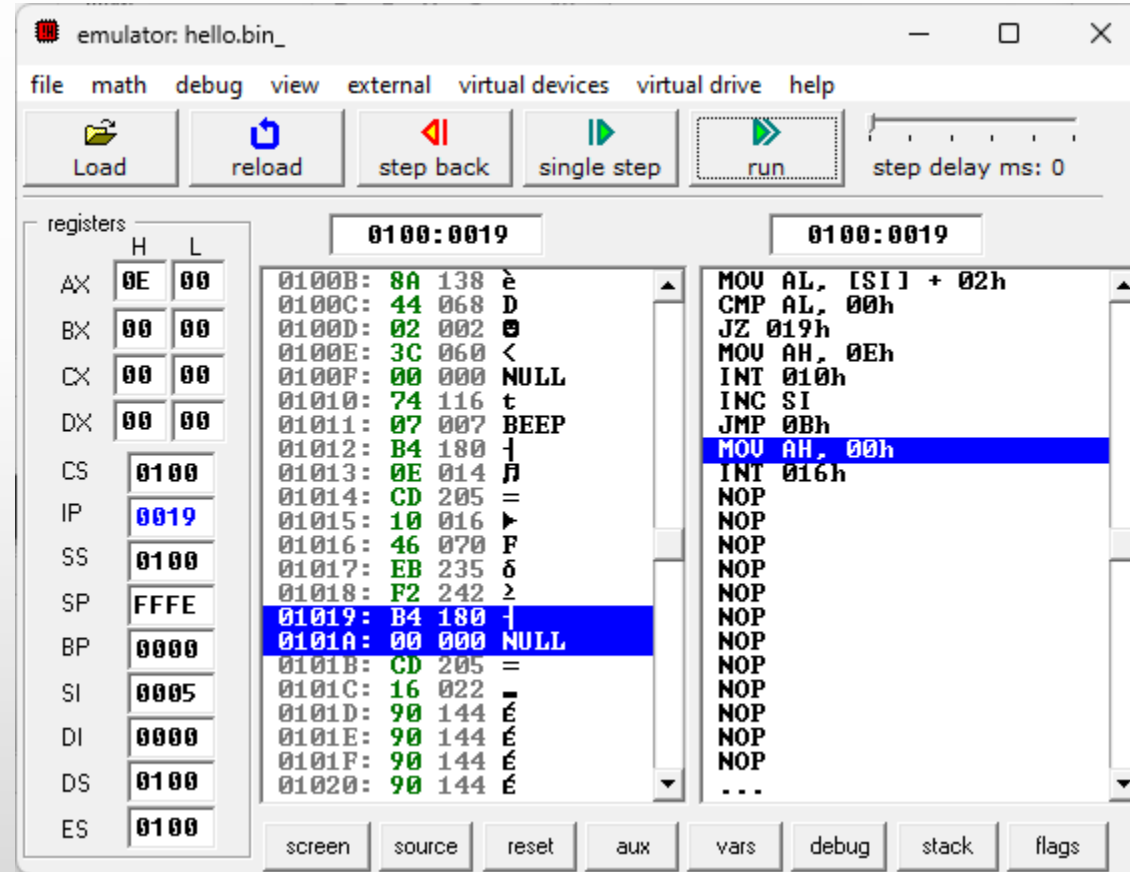
0100:000E

```
MOV AL, [SI] + 02h
CMP AL, 00h
JZ 019h
MOV AH, 0Eh
INT 010h
INC SI
JMP 0Bh
MOV AH, 00h
INT 016h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



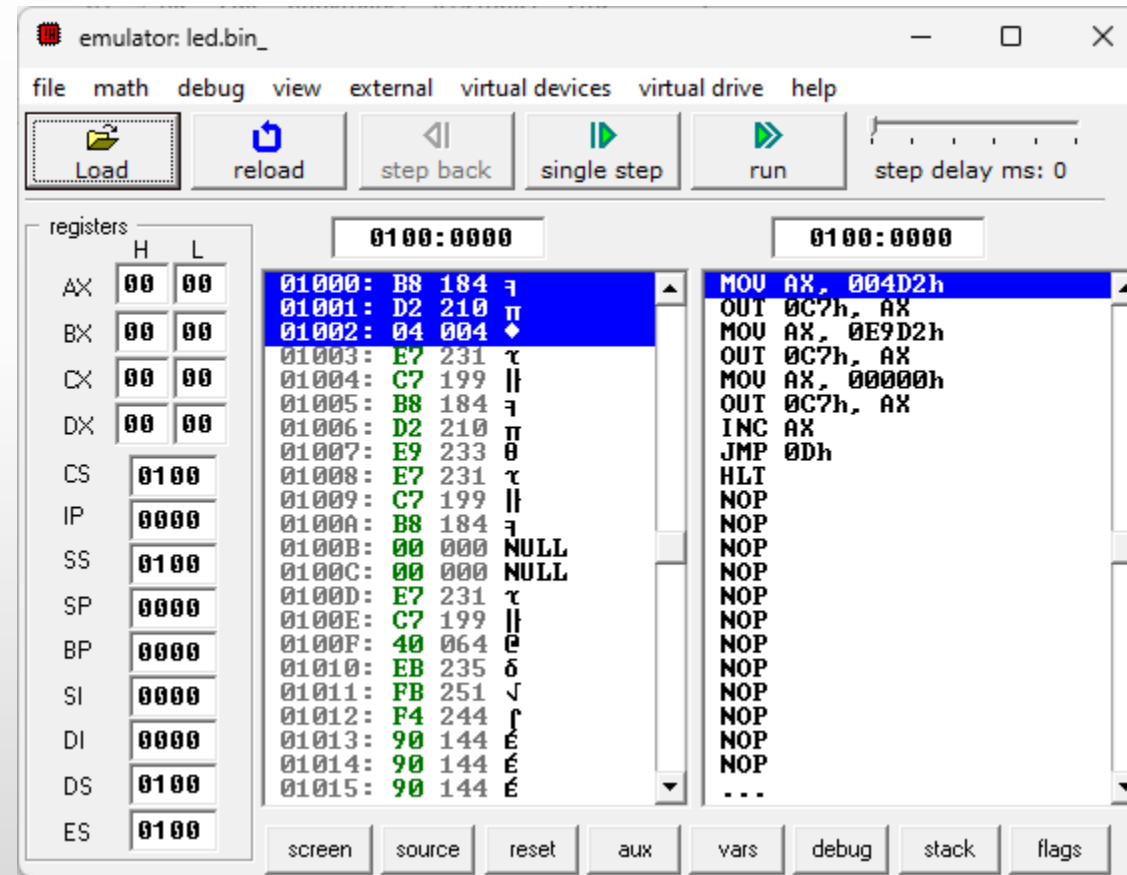
Ekрана Hello Yazdırma





LED Ekran Testi

```
#start=led_display.exe#  
#make_bin#  
mov ax, 1234  
out 199, ax  
mov ax, -5678  
out 199, ax  
mov ax, 0  
x1:  
    out 199, ax  
    inc ax  
jmp x1  
hlt
```





emulator: led.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	04	D2
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0005	
SS	0100	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

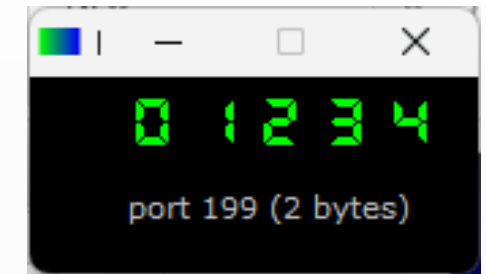
0100:0005

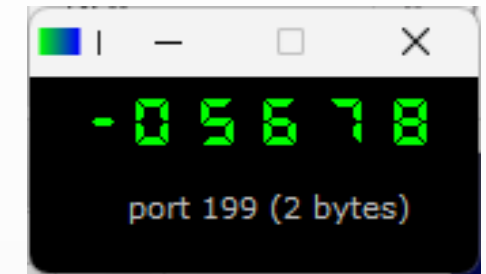
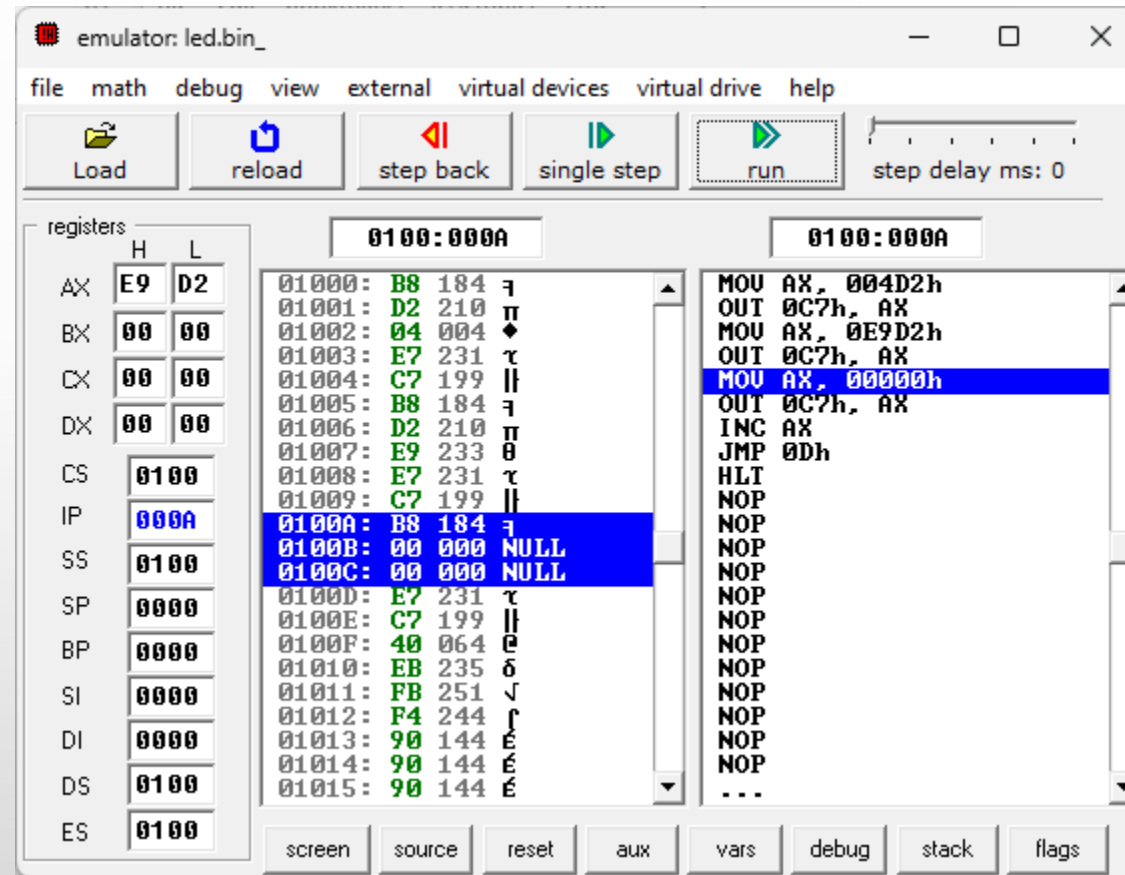
01000:	B8	184	?
01001:	D2	210	π
01002:	04	004	♦
01003:	E7	231	τ
01004:	C7	199	
01005:	B8	184	?
01006:	D2	210	π
01007:	E9	233	0
01008:	E7	231	τ
01009:	C7	199	
0100A:	B8	184	?
0100B:	00	000	NULL
0100C:	00	000	NULL
0100D:	E7	231	τ
0100E:	C7	199	
0100F:	40	064	@
01010:	EB	235	δ
01011:	FB	251	√
01012:	F4	244	ƒ
01013:	90	144	É
01014:	90	144	É
01015:	90	144	É

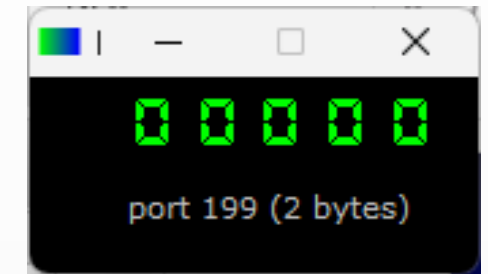
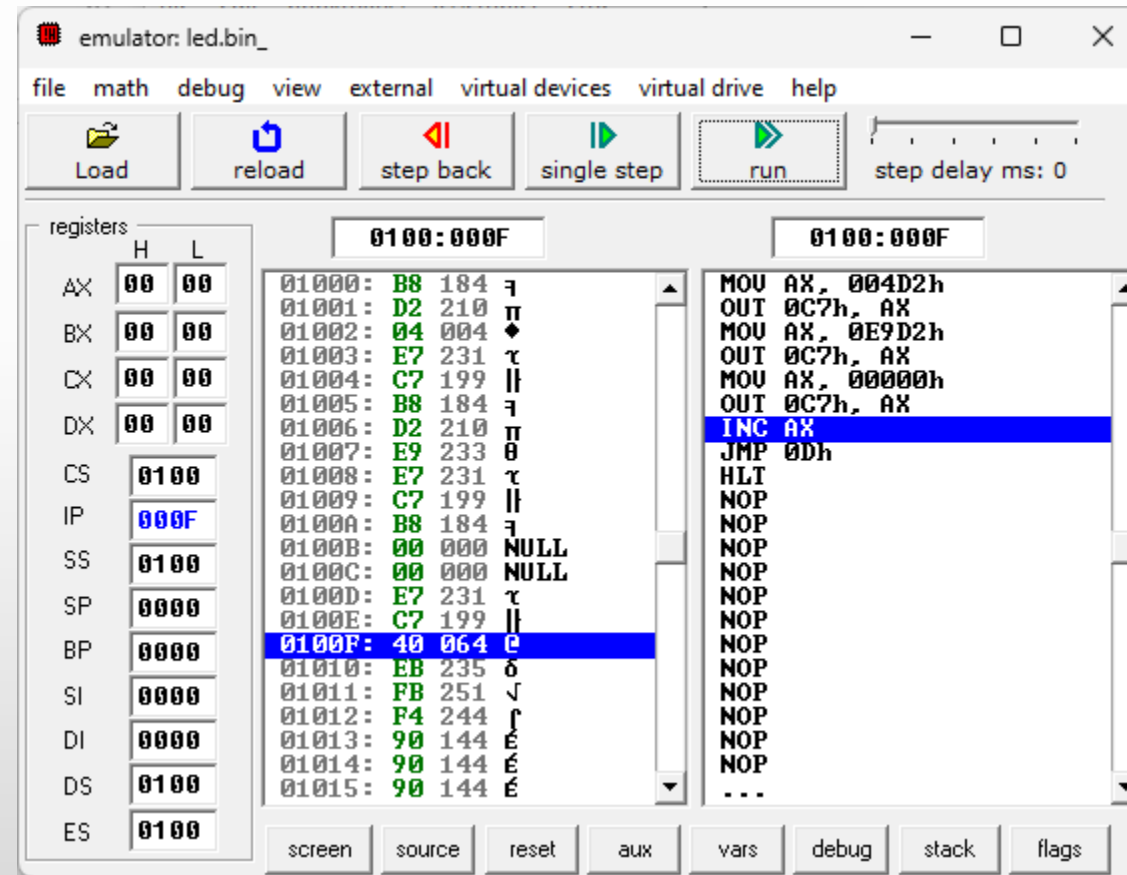
0100:0005

```
MOV AX, 004D2h
OUT 0C7h, AX
MOV AX, 0E9D2h
OUT 0C7h, AX
MOV AX, 00000h
OUT 0C7h, AX
INC AX
JMP 0Dh
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags

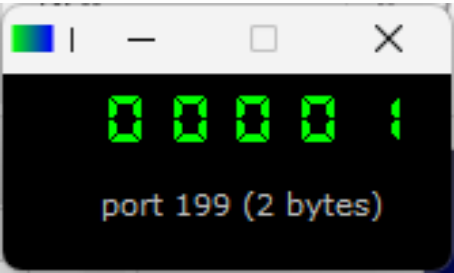














SON