



Bölüm 4: Değişkenler

Mikroişlemciler



Değişkenler

- Bir değişken, bir bellek konumunu temsil eder.
- Programcı için, *var1* değişkeninin
 - *5A73:235B* adresi yerine kullanılması çok daha kolaydır.
- Derleyici iki tür değişkeni destekler.
 - BYTE: Bir bayt (8 bit) değerini temsil eder.
 - name DB value (*Define Byte*).
 - WORD: Bir sözcük (16 bit) değerini temsil eder.
 - name DW value (*Define Word*).
- counter DB 10h (counter adında bir BYTE değişkeni).
- sum DW 0FFFFh (sum adında bir WORD değişkeni).



Değişkenler

- Değişken adları,
 - harfle başlamalıdır,
 - sonrasında harf veya rakam kombinasyonları içerebilir.
- İsim belirtilmeyen değişkenler,
 - isim belirtilmemiş olabilir ancak bir adresleri vardır.
- Herhangi bir sayısal değer (ikili, onlu, onaltılı) alabilir.
- İlk değer atanmamış değişkenler için "?" sembolü kullanılır.



Örnek Kod

```
ORG 100h
MOV AL, var1    ; AL, var1 değişkeninin değerini alır.
MOV BX, var2    ; BX, var2 değişkeninin değerini alır.
RET             ; Programı sonlandırır.
VAR1 DB 7       ; BYTE türünde var1 değişkeni, değeri 7.
var2 DW 1234h   ; WORD türünde var2 değişkeni, değeri 1234h.
```



Örnek Kod

•

memory (1K) at: 0B56 : 0100 Disassemble from: 0B56 : 0100

Address	Hex	ASCII	Instruction
0100	A0 160	á	MOV AL, [00108h]
0101	08 008		MOV BX, [00109h]
0102	01 001	@	RET
0103	8B 139	i	POP ES
0104	1E 030	▲	XOR AL, 012h
0105	09 009		ADD [BX + SI], AL
0106	01 001	@	ADD [BX + SI], AL
0107	C3 195	†	ADD [BX + SI], AL
0108	07 007	•	ADD [BX + SI], AL
0109	34 052	4	ADD [BX + SI], AL
010A	12 018	†	ADD [BX + SI], AL
010B	00 000		ADD [BX + SI], AL
010C	00 000		ADD [BX + SI], AL
010D	00 000		ADD [BX + SI], AL
010E	00 000		ADD [BX + SI], AL
010F	00 000		ADD [BX + SI], AL
0110	00 000		ADD [BX + SI], AL
0111	00 000		ADD [BX + SI], AL
0112	00 000		ADD [BX + SI], AL

variables



Örnek Kod

- Derleyici,
 - Kaynak kodu bir dizi bayta dönüştürür.
 - Değişken adlarını bellek konumlarıyla değiştirir.
 - Büyük-küçük harfe duyarsızdır ("VAR1" ve "var1" aynı değişken).
- COM dosyaları yüklendiğinde DS, CS yazmacının değerini alır.
- Bağıl konum (*offset*), bellek konumunun başlangıcından olan kaydırmadır.
- VAR1'in bağıl konumu 0108h, tam adresi 0B56:0108'dir.
- var2'nin bağıl konumu 0109h, tam adresi 0B56:0109'dur.
 - WORD türünde olduğu için 2 BYTE kaplar.
 - Düşük bayt düşük adreste saklanır, 34h, 12h'den önce yer alır.



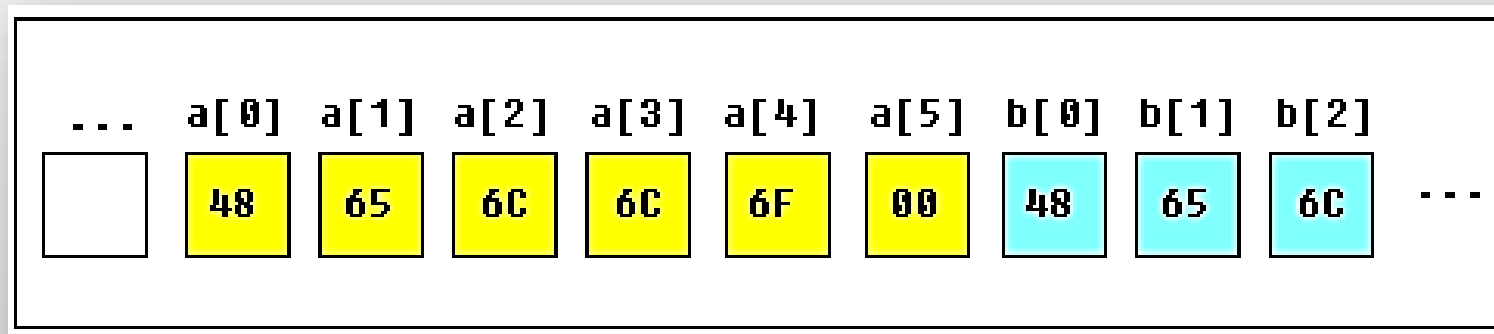
ORG 100h Direktifi

- Derleyiciye yürütülebilir dosyanın yükleneceği adresi söyler.
 - Değişken adları bellek adresleriyle değiştirilirken dikkate alınır.
- COM dosyalarında kullanılır.
- İşletim sistemi, CS'nin ilk 256 baytında programla ilgili bazı verileri tutar.
 - Komut satırı parametreleri gibi bilgiler bu alanda saklanır.
- EXE dosyaları 0000 ofsetinde yüklenir ve
 - Değişkenler için özel bir kesim (*segment*) kullanır.
- Direktifler, gerçek makine koduna dönüştürülmez.



Diziler

- Diziler, değişken zincirleri olarak düşünülebilir.
- Diziler, ardışık bellek konumlarında saklanan veri gruplarıdır.
- Tırnak içindeki metin otomatik olarak bayt dizisine dönüştürür.
- Dizi elemanlarına bellek adresi üzerinden erişilebilir.
- a DB 48h, 65h, 6Ch, 6Ch, 6Fh, 00h
- b DB 'Hello', 0.





Diziler

- Dizinin herhangi bir elemanına,
 - Köşeli parantez kullanarak erişilebilir.
 - Örneğin: `MOV AL, a[3]`
 - Bellek indis yazmaçları `BX`, `SI`, `DI`, `BP` kullanarak erişilebilir.
 - Örneğin: `MOV SI, 3` ve `MOV AL, a[SI]`
- Büyük bir dizi tanımlamak için `DUP` işleci kullanılır.
 - `number DUP (value(s))`
 - `c DB 5 DUP(9) → c DB 9, 9, 9, 9, 9`
 - `d DB 5 DUP(1, 2) → d DB 1, 2, 1, 2, 1, 2, 1, 2, 1, 2`
- `DW` dizi tanımlamak için kullanılamaz.!



Değişkenin Adresini Alma

- LEA ve OFFSET, değişkenin adresini almak için kullanılır.
- LEA, indisli değişkenlerin adresini almak için de kullanılabilir.
- Değişkenin adresi, prosedüre parametre geçirmek için çok kullanışlıdır.



Değişkenin Adresini Alma

```
ORG 100h
MOV     AL, VAR1           ; VAR1 değerini AL yazmacına kopyala.
LEA     BX, VAR1           ; VAR1 adresini BX yazmacına kopyala.
MOV     BYTE PTR [BX], 44h ; VAR1 değerini güncelle.
MOV     AL, VAR1           ; VAR1 değerini AL yazmacına kopyala.
RET
VAR1    DB    22h
END
```



Değişkenin Adresini Alma

```
ORG 100h
```

```
MOV     AL, VAR1           ; VAR1 değerini AL yazmacına kopyala.
```

```
MOV     BX, OFFSET VAR1   ; VAR1 adresini BX yazmacına kopyala.
```

```
MOV     BYTE PTR [BX], 44h ; VAR1 değerini güncelle.
```

```
MOV     AL, VAR1           ; VAR1 değerini AL yazmacına kopyala.
```

```
RET
```

```
VAR1    DB    22h
```

```
END
```



Değişkenin Adresini Alma

- LEA BX, VAR1 ve MOV BX, OFFSET VAR1 aynı makine koduna derlenir:
 - MOV BX, num
 - num, değişkenin bağıl konum değerinin 16 bitlik temsili.
- Sadece BX, SI, DI, BP yazmaçları köşeli parantez içinde kullanılabilir.
- LEA (Load Effective Address), değişkenin adresini yazmaca yükler.
- OFFSET, değişkenin bağıl konum değerini sağlar.



Sabitler (Constants)

- Değişkenlere benzer.
- Ancak program derlendiğinde var olurlar.
- Bir sabitin değeri tanımlandıktan sonra değiştirilemez.
- Sabitler, programın farklı bölümlerinde aynı değer kullanılmasını sağlar.
- Sabit tanımlamak için EQU direktifi kullanılır:
 - name EQU <herhangi bir ifade>.
 - k EQU 5
 - MOV AX, k



Değişkenler Arasında Veri Taşıma

jmp start

value1 DW 10 ; İki byte'lık bir değişken tanımlama

value2 DW 0 ; İkinci bir değişken tanımlama

start:

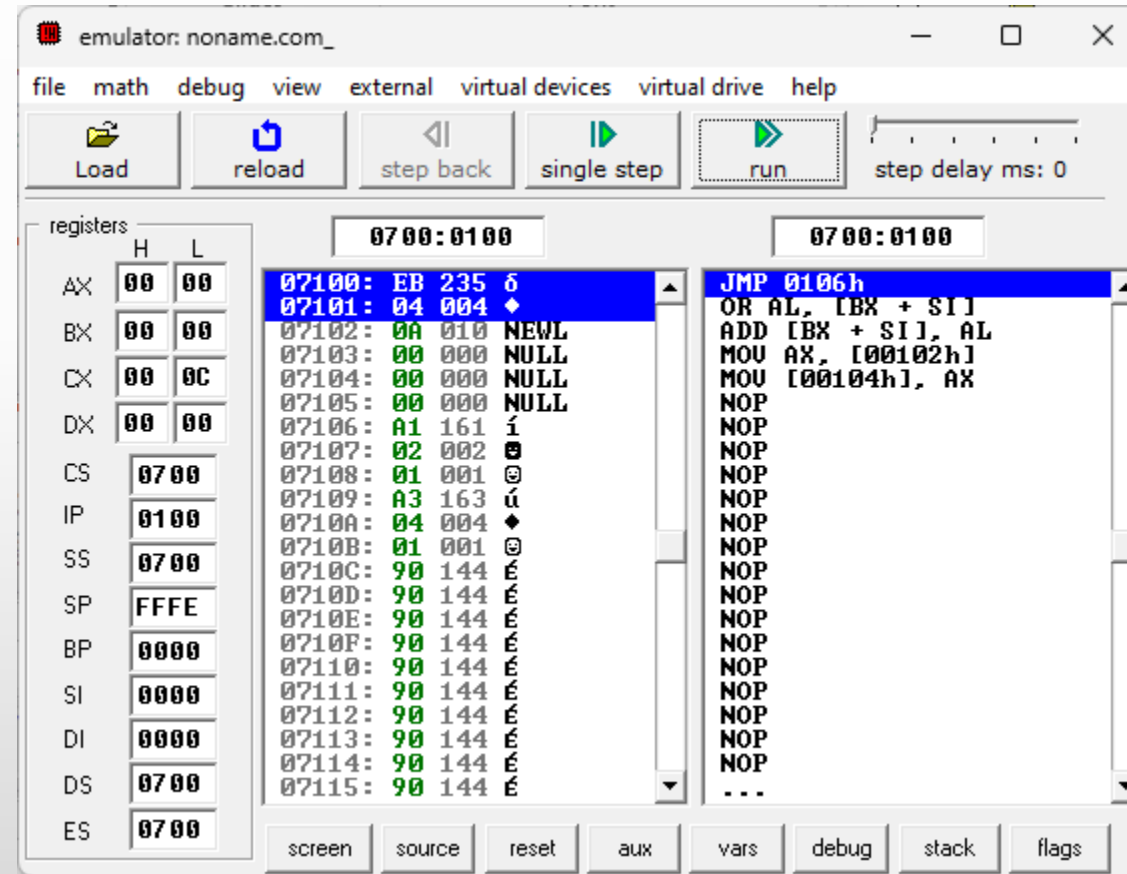
MOV AX, value1 ; value1 değişkeninin değerini AX yazmacına taşı

MOV value2, AX ; AX yazmacındaki değeri value2 değişkenine taşı

end start

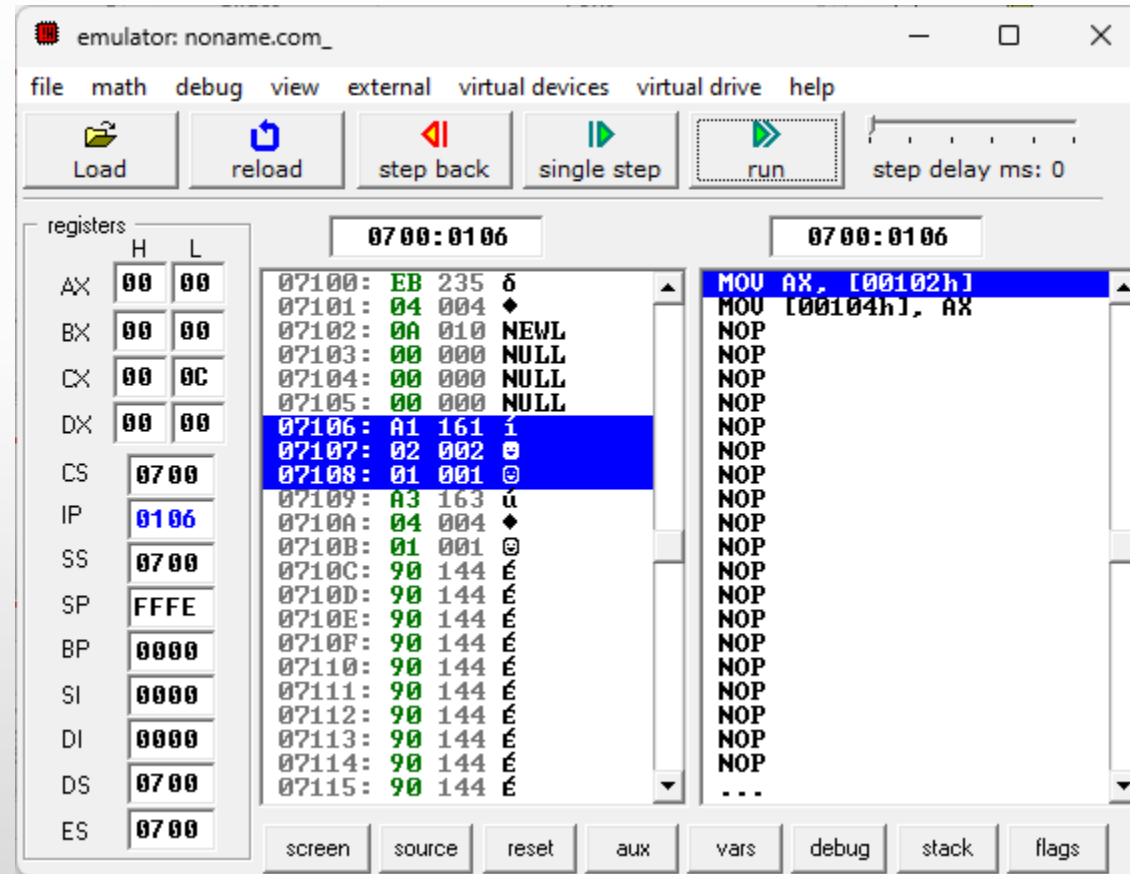


Değişkenler Arasında Veri Taşıma





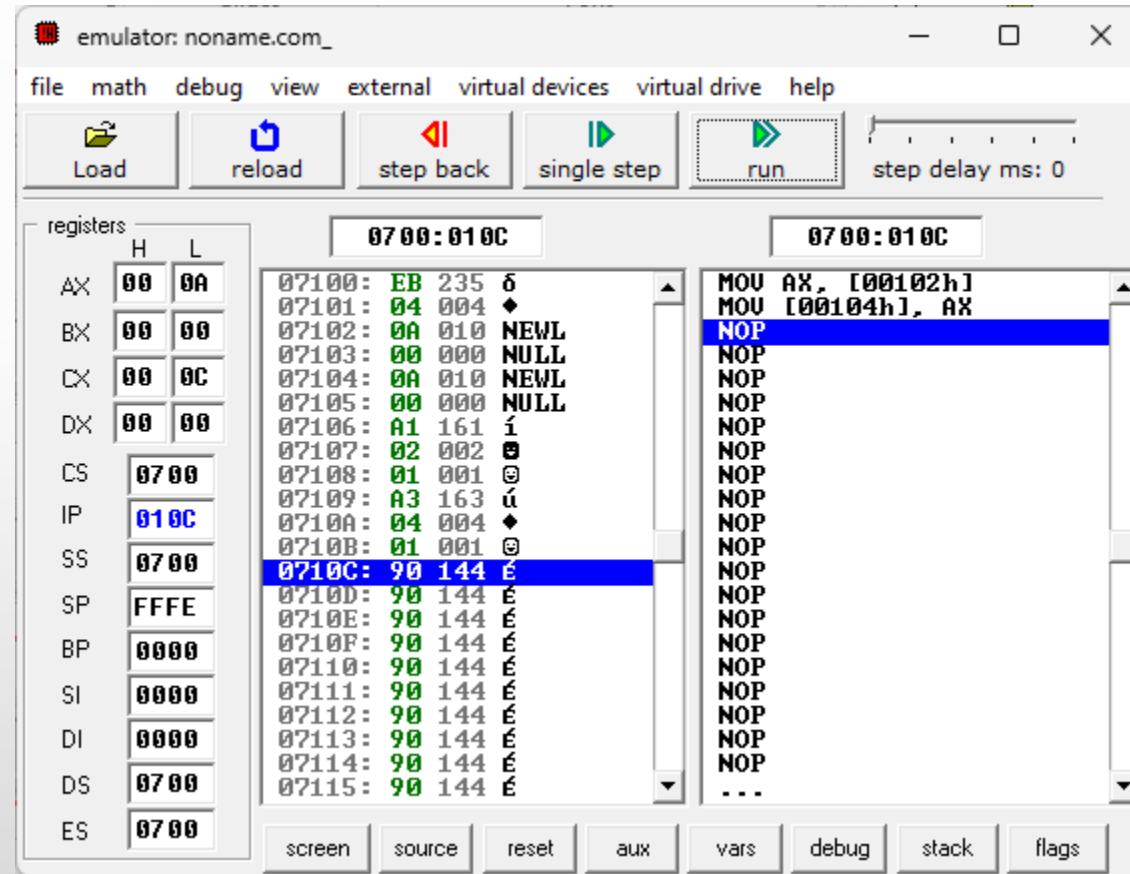
Değişkenler Arasında Veri Taşıma







Değişkenler Arasında Veri Taşıma





Değişkenleri Toplama İşlemi

jmp start

value1 DW 105h ; İki byte'lık değişken tanımla, başlangıç değeri 105h

value2 DW 103h ; İkinci bir değişken tanımla

result DW ? ; Sonucu saklayacak değişken, başlangıç değeri belirsiz (?)

start:

MOV AX, value1 ; value1 değişkeninin değerini AX yazmacına yükle

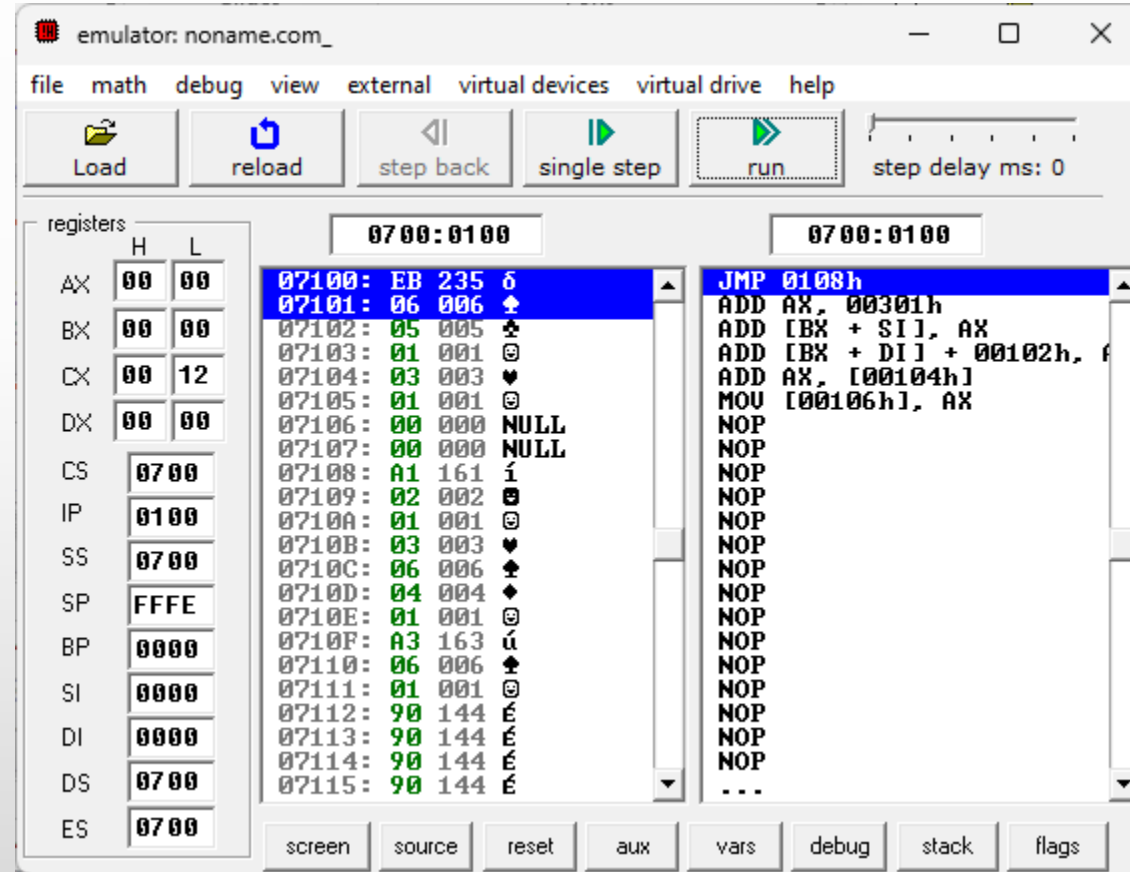
ADD AX, value2 ; value2 değişkeninin değerini AX yazmacına ekle

MOV result, AX ; AX yazmacındaki değeri result değişkenine taşı

end start

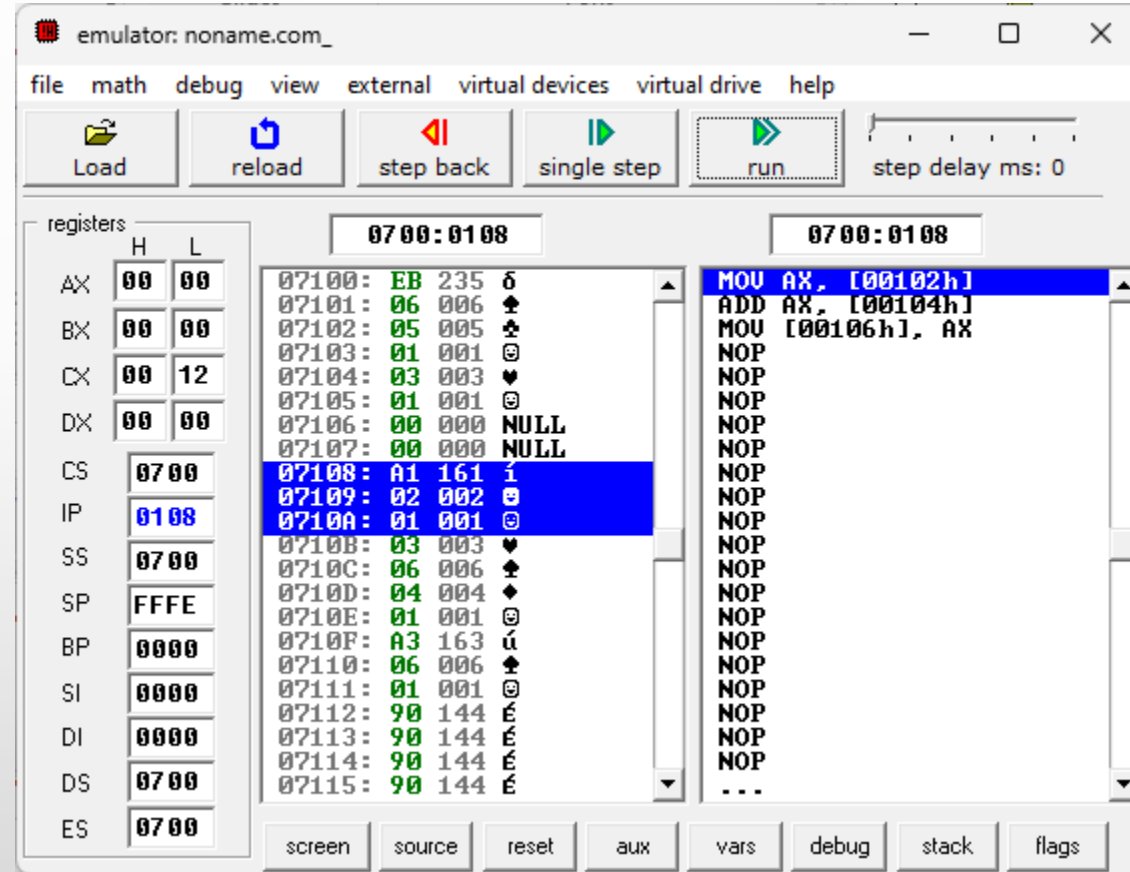


Değişkenleri Toplama İşlemi



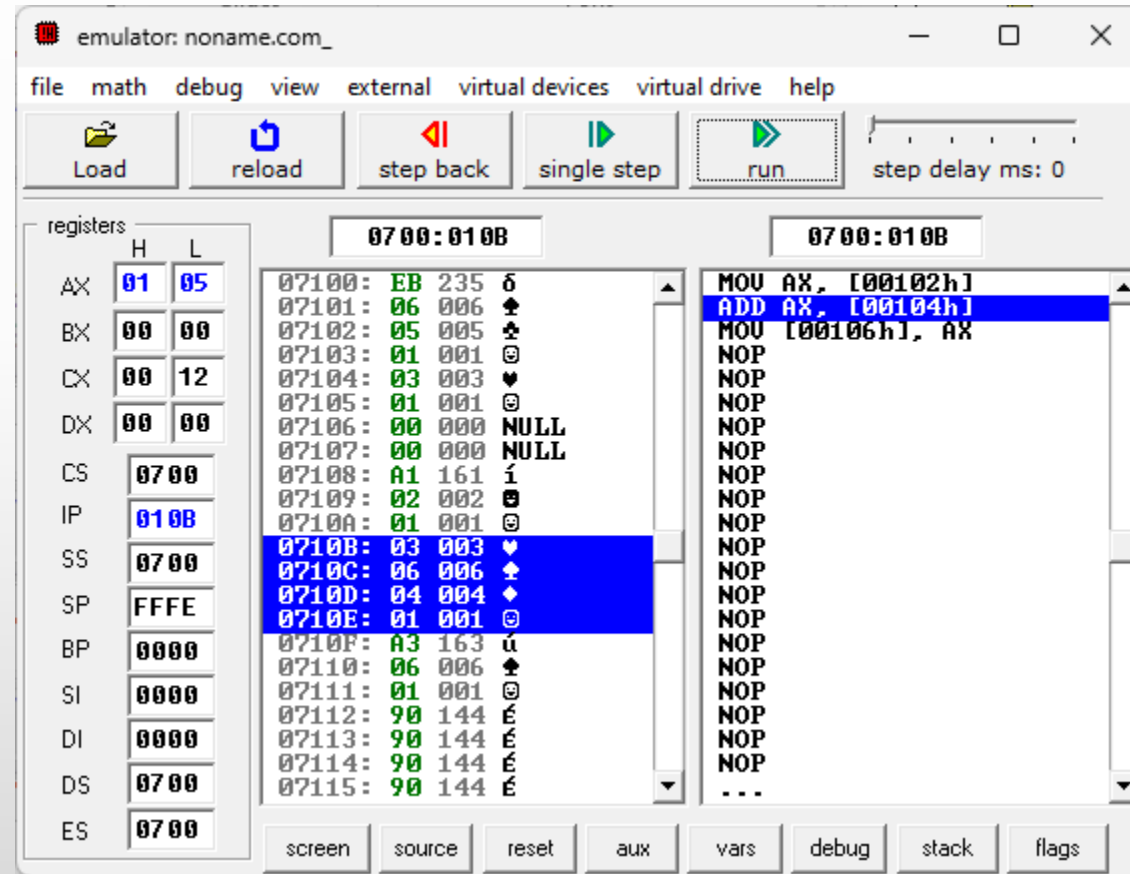


Değişkenleri Toplama İşlemi



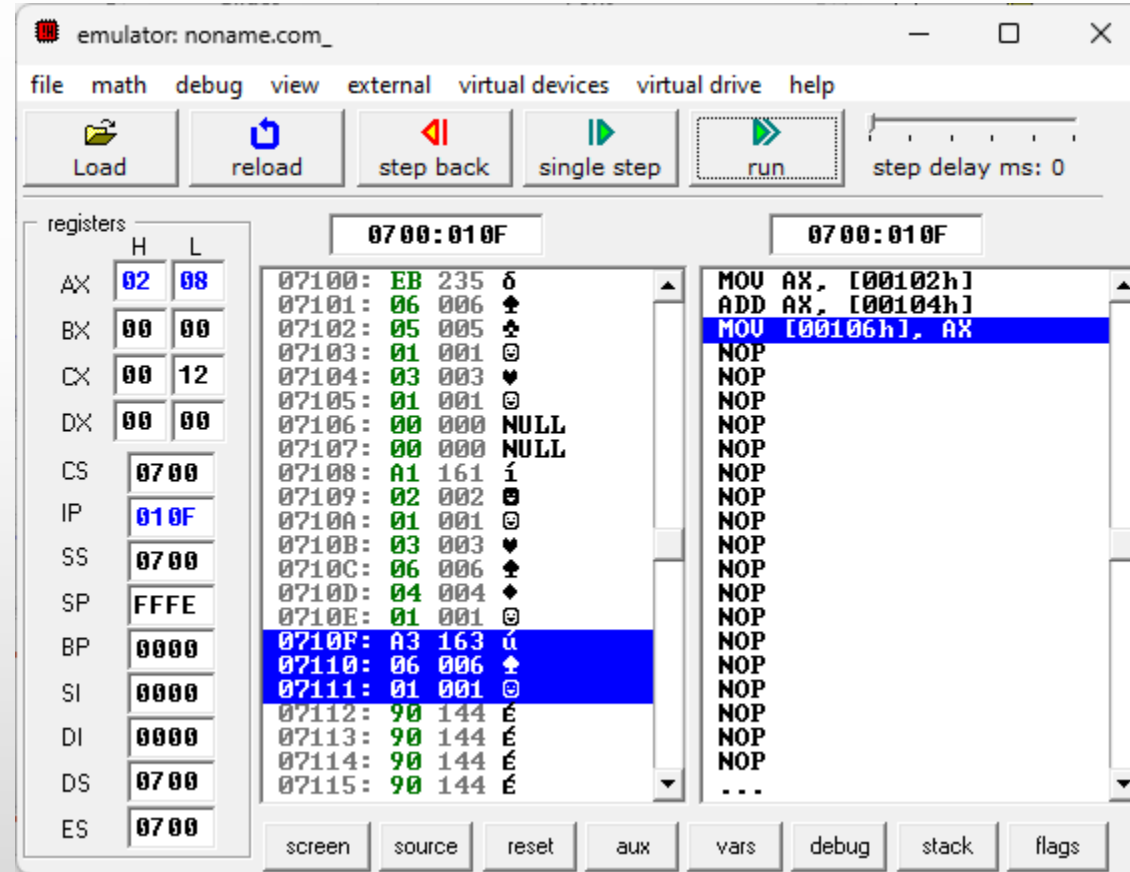


Değişkenleri Toplama İşlemi



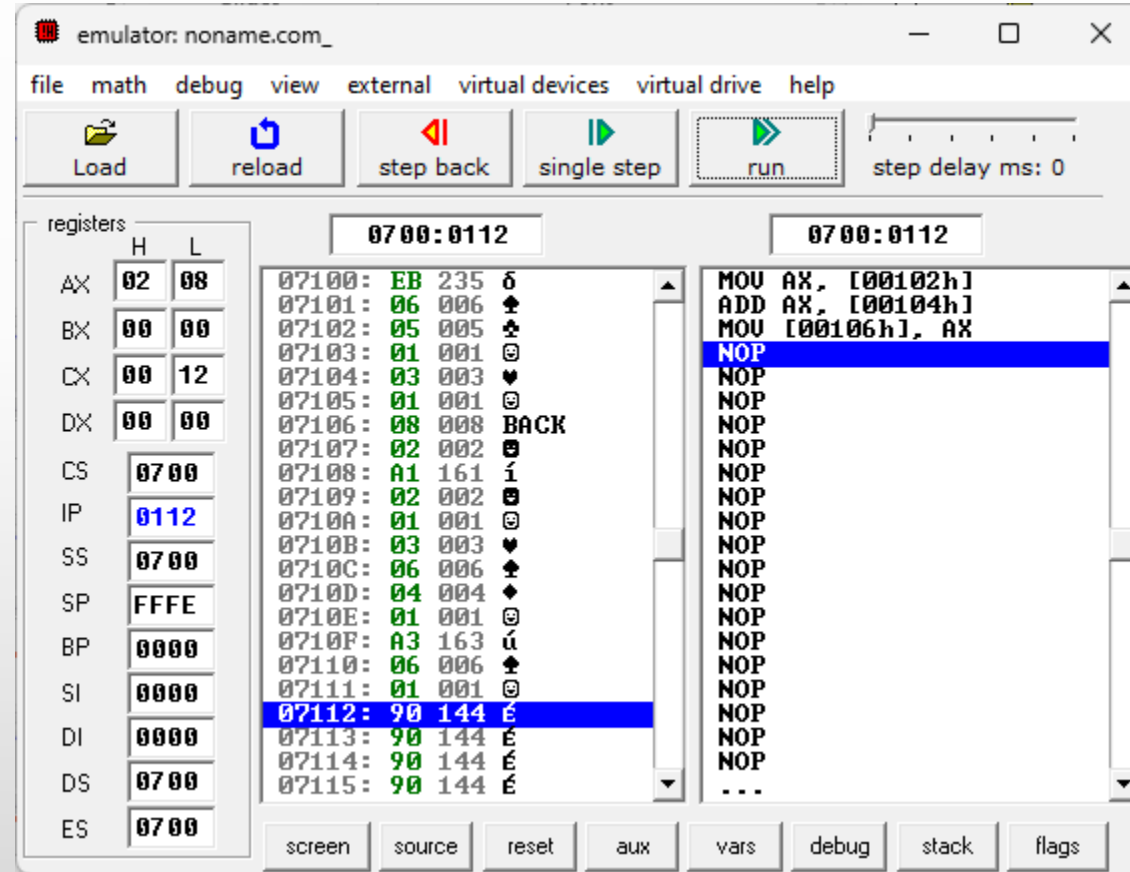


Değişkenleri Toplama İşlemi



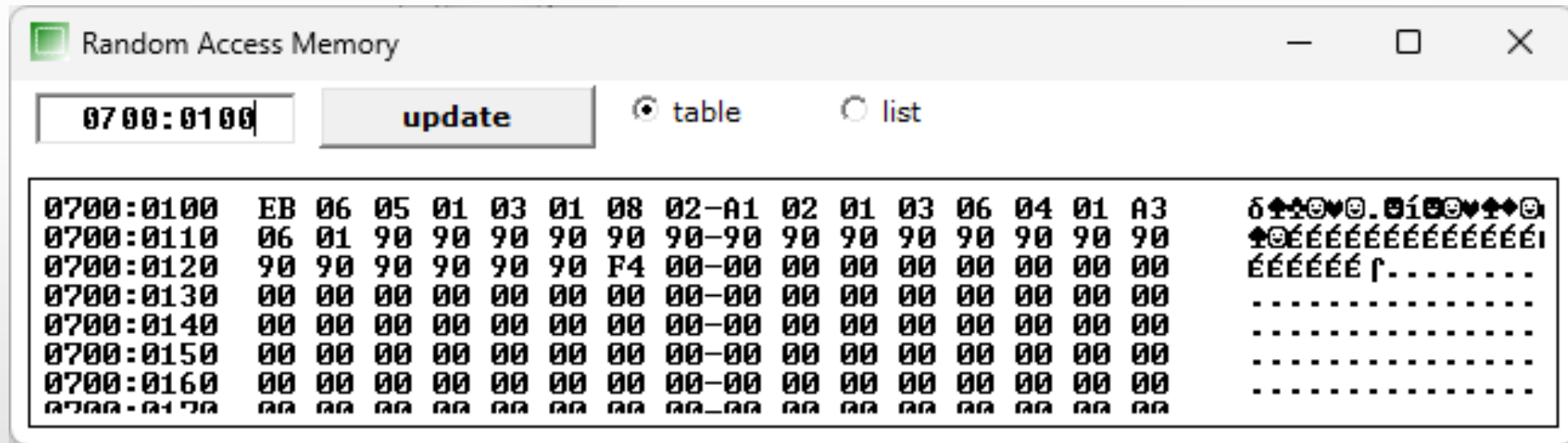


Değişkenleri Toplama İşlemi





Değişkenleri Toplama İşlemi





İki Değişkenin Değerini Karşılaştırma

START:

```
MOV AX, value1 ; Birinci değişkenin değerini AX yazmacına yükle
CMP AX, value2 ; AX yazmacındaki değeri ikinci değişken ile karşılaştır
JGE greater_or_equal ; value1, value2'den büyük veya eşitse atla
MOV AX, value2 ; İkinci değişkenin değerini AX yazmacına yükle
JMP store_max ; Karşılaştırma sonrası büyük değeri saklayan kısma atla
```

greater_or_equal:

```
MOV AX, value1 ; Birinci değişkenin değerini AX yazmacına yükle
```

store_max:

```
MOV max_value, AX ; AX yazmacındaki değeri max_value değişkenine taşı
value1 DW 10 ; Birinci değişken, başlangıç değeri 10
value2 DW 5 ; İkinci değişken, başlangıç değeri 5
max_value DW ? ; Büyük değeri tutacak değişken, ilk değeri belirsiz (?)
```

END START



İki Değişkenin Değerini Karşılaştırma

emulator: noname.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	1A
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100

07100:	A1	161	i
07101:	14	020	¶
07102:	01	001	⊖
07103:	3B	059	;
07104:	06	006	♣
07105:	16	022	=
07106:	01	001	⊖
07107:	7D	125	>
07108:	05	005	♣
07109:	A1	161	i
0710A:	16	022	=
0710B:	01	001	⊖
0710C:	EB	235	δ
0710D:	03	003	♥
0710E:	A1	161	i
0710F:	14	020	¶
07110:	01	001	⊖
07111:	A3	163	ú
07112:	18	024	↑
07113:	01	001	⊖
07114:	0A	010	NEWL
07115:	00	000	NULL

0700:0100

```
MOV AX, [00114h]
CMP AX, [00116h]
JNL 010Eh
MOV AX, [00116h]
JMP 0111h
MOV AX, [00114h]
MOV [00118h], AX
OR AL, [BX + SI]
ADD AX, 00000h
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



İki Değişkenin Değerini Karşılaştırma

emulator: noname.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	0A
BX	00	00
CX	00	1A
DX	00	00
CS	0700	
IP	0103	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0103

07100:	A1	161	i
07101:	14	020	¶
07102:	01	001	⊖
07103:	3B	059	;
07104:	06	006	⬆
07105:	16	022	—
07106:	01	001	⊖
07107:	7D	125	>
07108:	05	005	♣
07109:	A1	161	i
0710A:	16	022	—
0710B:	01	001	⊖
0710C:	EB	235	δ
0710D:	03	003	♥
0710E:	A1	161	i
0710F:	14	020	¶
07110:	01	001	⊖
07111:	A3	163	ú
07112:	18	024	↑
07113:	01	001	⊖
07114:	0A	010	NEWL
07115:	00	000	NULL

0700:0103

```
MOV AX, [00114h]
CMP AX, [00116h]
JNL 010Eh
MOV AX, [00116h]
JMP 0111h
MOV AX, [00114h]
MOV [00118h], AX
OR AL, [BX + SI]
ADD AX, 00000h
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



İki Değişkenin Değerini Karşılaştırma

emulator: noname.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	0A
BX	00	00
CX	00	1A
DX	00	00
CS	07 00	
IP	01 07	
SS	07 00	
SP	FF FE	
BP	00 00	
SI	00 00	
DI	00 00	
DS	07 00	
ES	07 00	

07 00: 01 07

07100:	A1	161	i
07101:	14	020	¶
07102:	01	001	⊖
07103:	3B	059	;
07104:	06	006	♣
07105:	16	022	=
07106:	01	001	⊖
07107:	7D	125	>
07108:	05	005	♣
07109:	A1	161	i
0710A:	16	022	=
0710B:	01	001	⊖
0710C:	EB	235	δ
0710D:	03	003	♥
0710E:	A1	161	i
0710F:	14	020	¶
07110:	01	001	⊖
07111:	A3	163	ú
07112:	18	024	↑
07113:	01	001	⊖
07114:	0A	010	NEWL
07115:	00	000	NULL

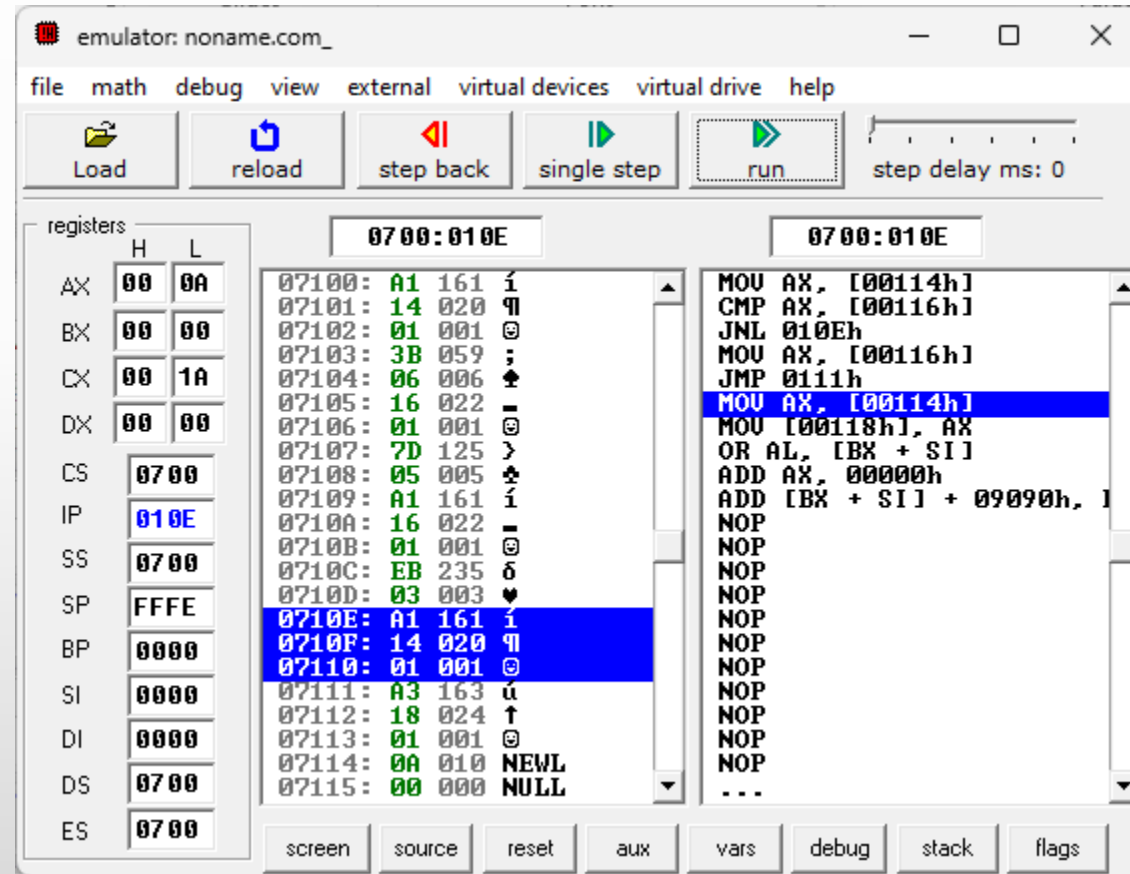
07 00: 01 07

```
MOV AX, [00114h]
CMP AX, [00116h]
JNL 010Eh
MOV AX, [00116h]
JMP 0111h
MOV AX, [00114h]
MOV [00118h], AX
OR AL, [BX + SI]
ADD AX, 00000h
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



İki Değişkenin Değerini Karşılaştırma





İki Değişkenin Değerini Karşılaştırma

emulator: noname.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	0A
BX	00	00
CX	00	1A
DX	00	00
CS	0700	
IP	0111	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0111

```
07100: A1 161 i
07101: 14 020 q
07102: 01 001 @
07103: 3B 059 ;
07104: 06 006 +
07105: 16 022 =
07106: 01 001 @
07107: 7D 125 >
07108: 05 005 +
07109: A1 161 i
0710A: 16 022 =
0710B: 01 001 @
0710C: EB 235 d
0710D: 03 003 v
0710E: A1 161 i
0710F: 14 020 q
07110: 01 001 @
07111: A3 163 u
07112: 18 024 t
07113: 01 001 @
07114: 0A 010 NEWL
07115: 00 000 NULL
```

0700:0111

```
MOV AX, [00114h]
CMP AX, [00116h]
JNL 010Eh
MOV AX, [00116h]
JMP 0111h
MOV AX, [00114h]
MOV [00118h], AX
OR AL, [BX + SI]
ADD AX, 00000h
ADD [BX + SI] + 09090h, 1
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags



SON