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).

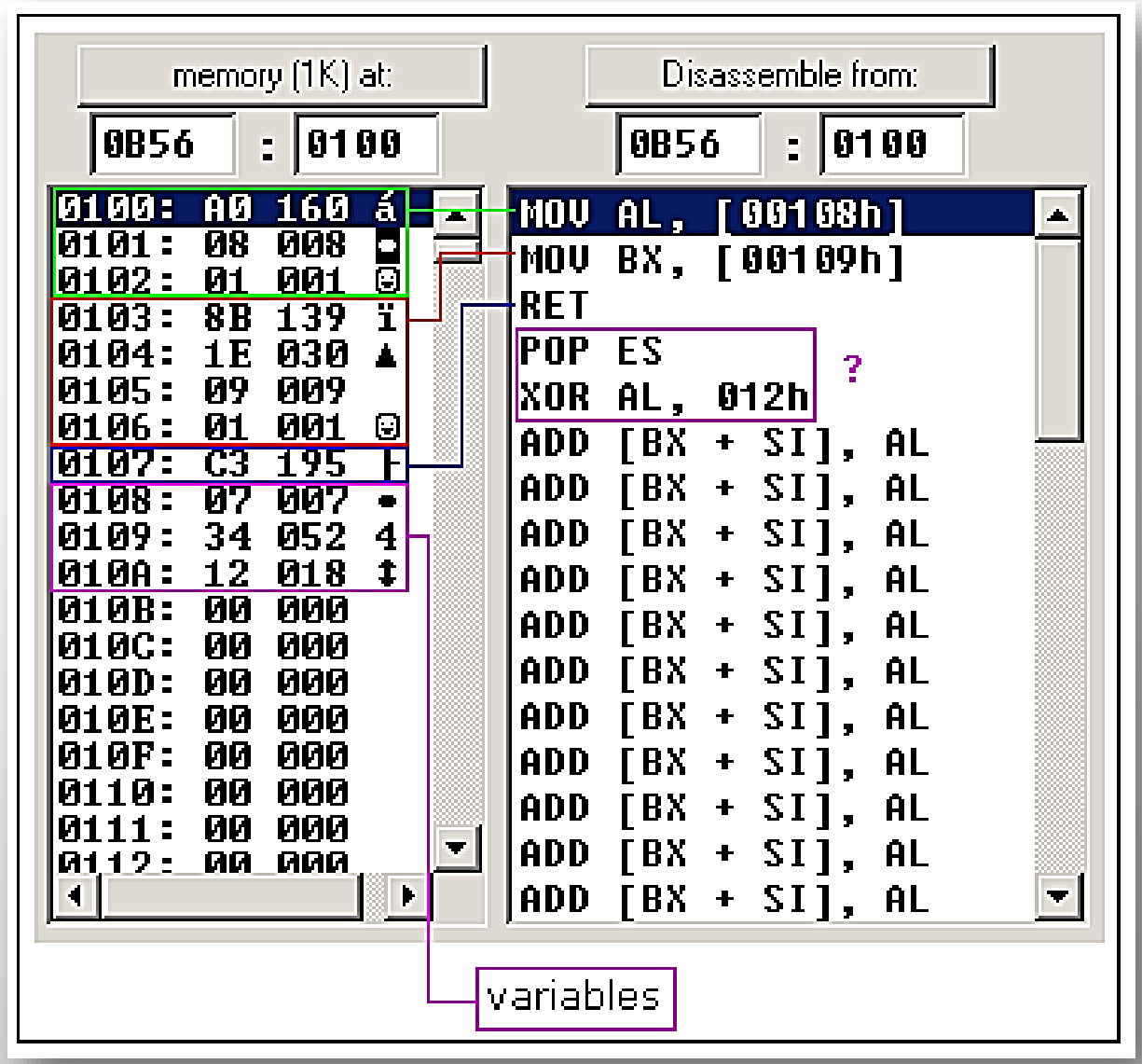
Değişkenler

- 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



Örnek Kod

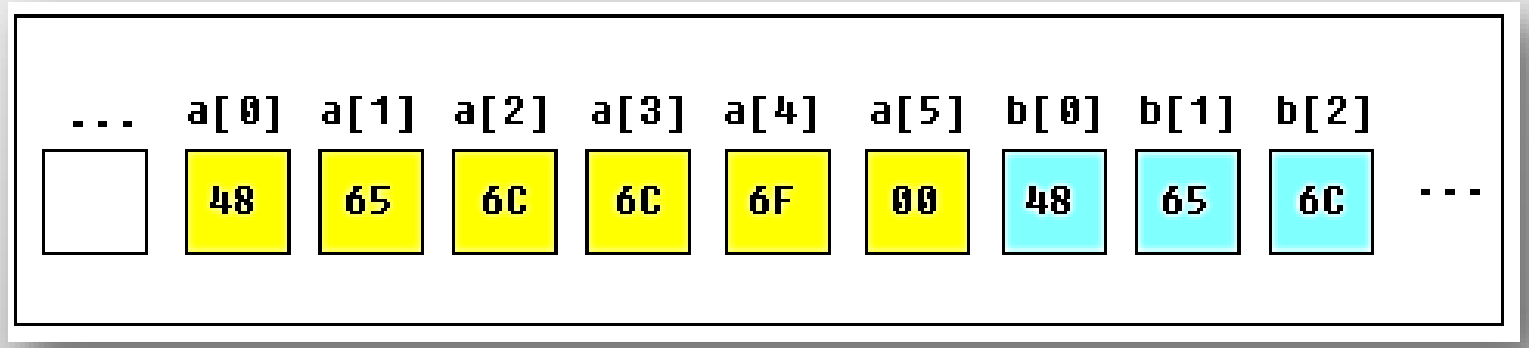
- Derleyici,
  - Kaynak kodu bir dizi bayta dönüştürür.
  - Değişken adlarını bellek konumlarıyla değiştirir.
  - Büyük-küçük harfe duyarsızdır ("VAR1" ve "var1" aynı değişken).
- COM dosyaları yüklendiğinde DS, CS yazmacının değerini alır.
- Bağıl konum (*offset*), bellek konumunun başlangıcından olan 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.

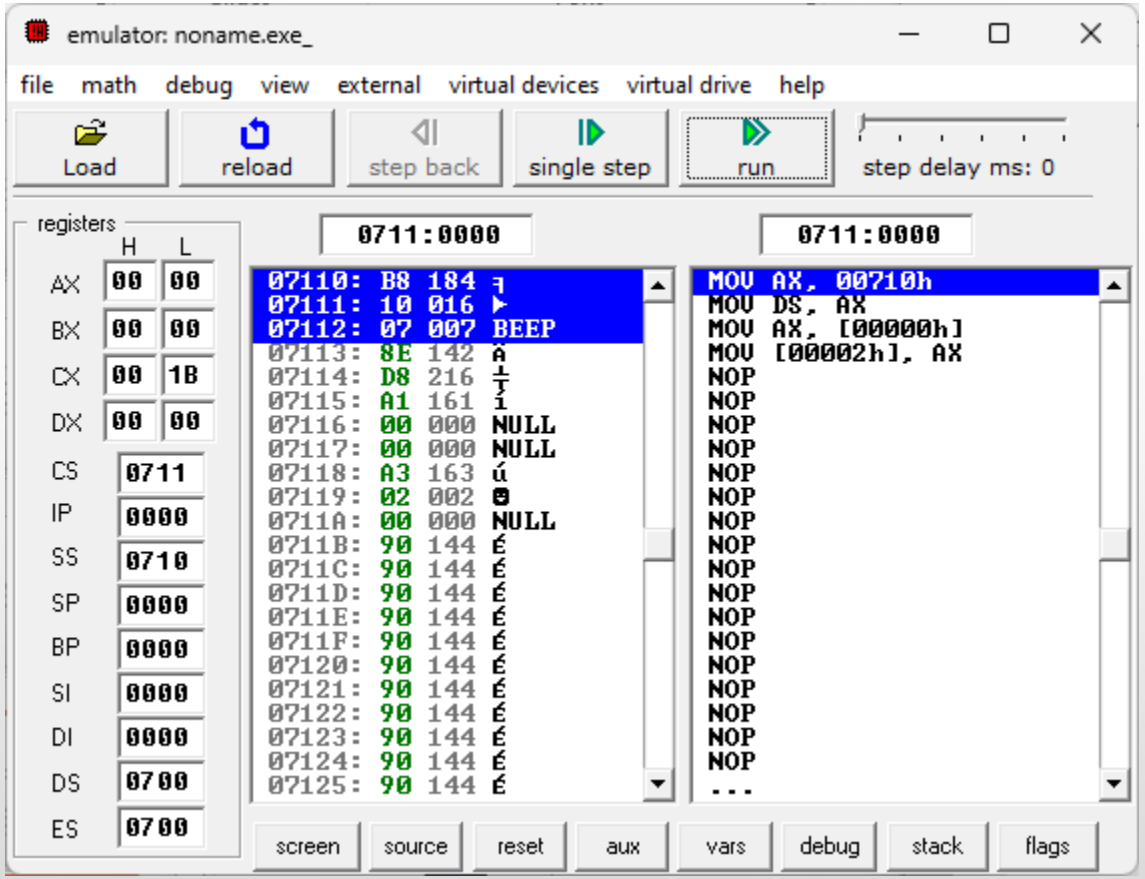
Sabitler (Constants)

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

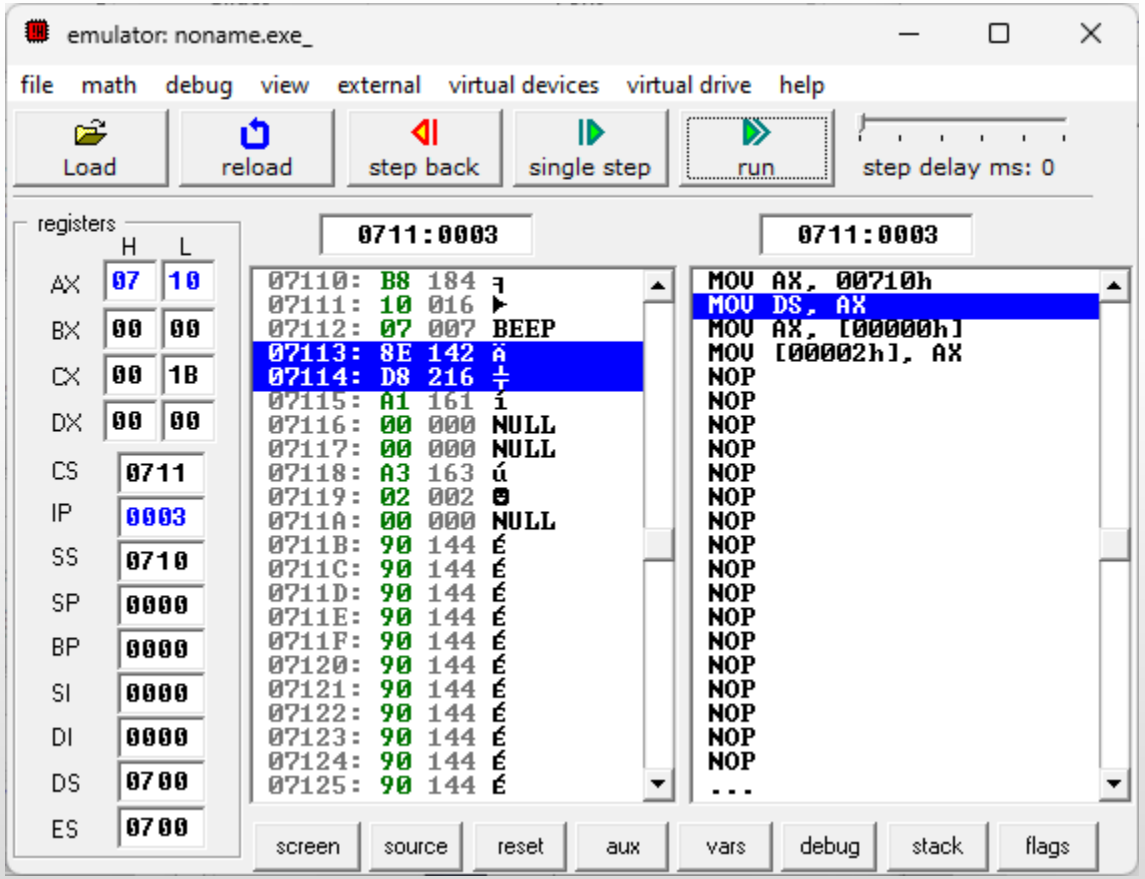
Değişkenler Arasında Veri Taşıma

```
DATA SEGMENT
    value1 DW 10      ; İki byte'lık bir değişken tanımlama
    value2 DW 0       ; İkinci bir değişken tanımlama
DATA ENDS
CODE SEGMENT
START:
    MOV AX, DATA     ; DATA kesiminin adresini AX yazmacına taşı
    MOV DS, AX       ; DS yazmacına DATA kesiminin adresini yükle
    MOV AX, value1   ; value1 değişkeninin değerini AX yazmacına taşı
    MOV value2, AX   ; AX yazmacındaki değeri value2 değişkenine taşı
END START
CODE ENDS
```

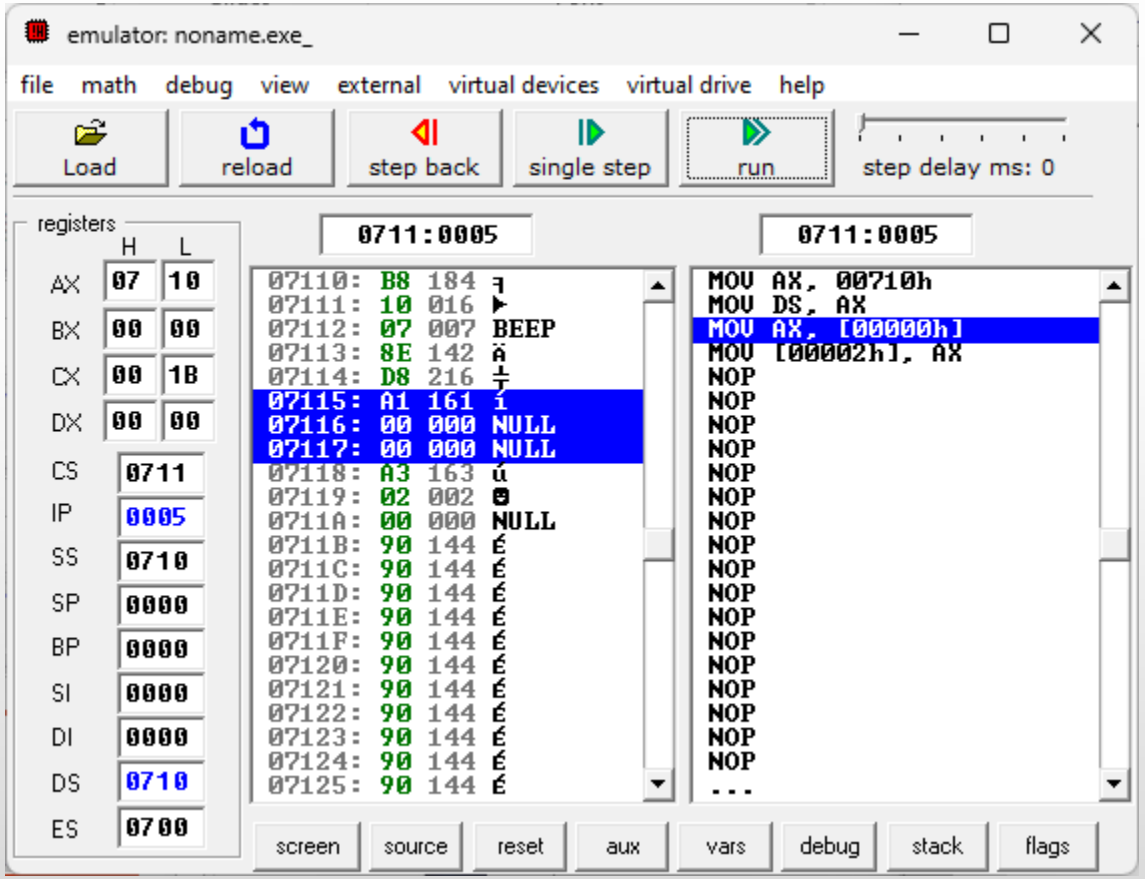
Değişkenler Arasında Veri Taşıma



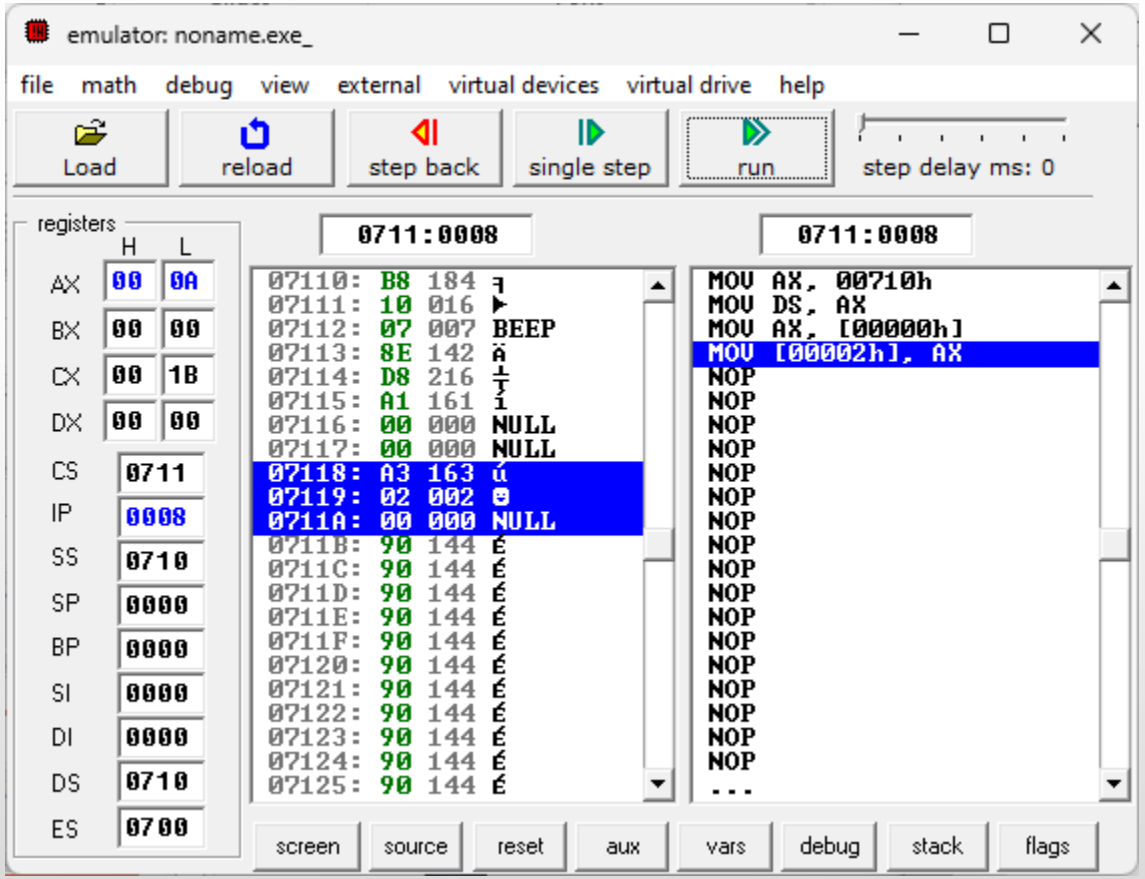
Değişkenler Arasında Veri Taşıma



Değişkenler Arasında Veri Taşıma



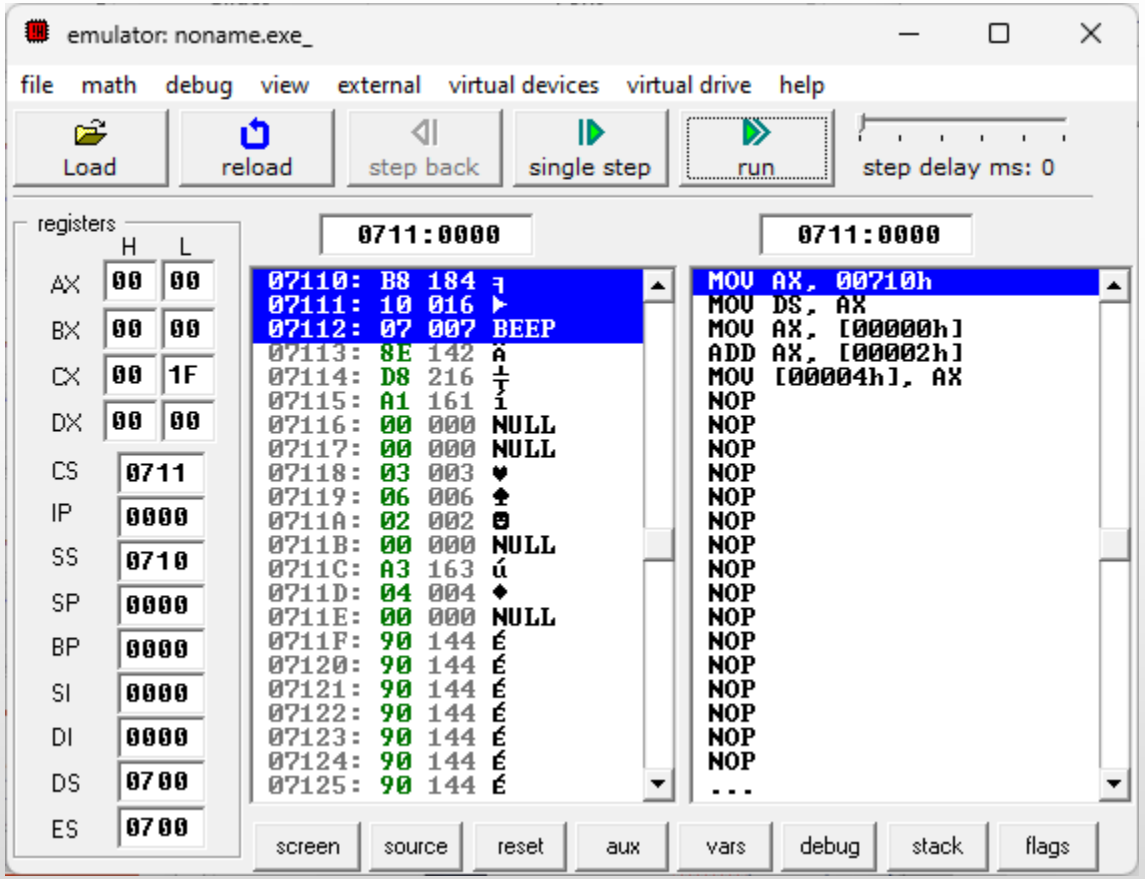
Değişkenler Arasında Veri Taşıma



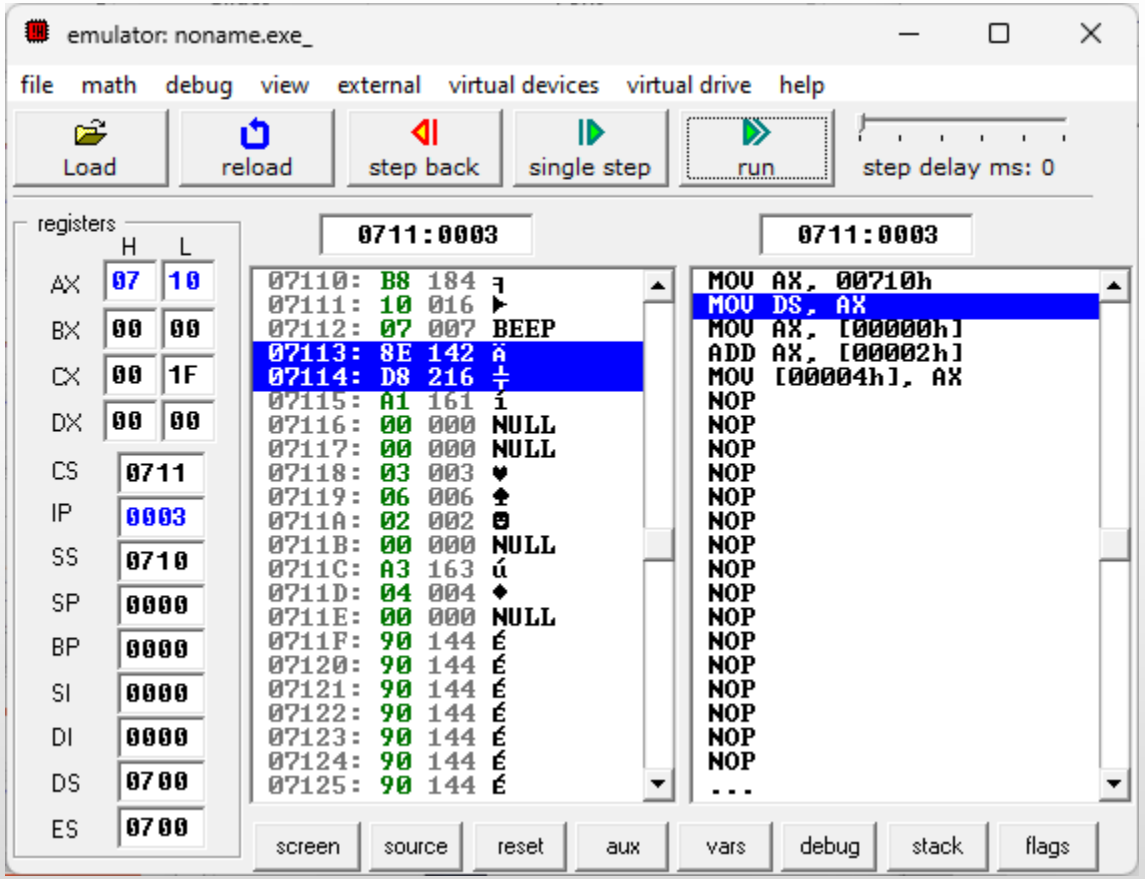
Değişkenleri Toplama İşlemi

```
DATA SEGMENT
    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 (?)
DATA ENDS
START:
    MOV AX, DATA     ; DATA kesiminin adresini AX yazmacına yükle
    MOV DS, AX       ; DS yazmacına DATA kesiminin adresini yükle
    MOV AX, value1   ; value1 değişkeninin değerini AX yazmacına yükle
    ADD AX, value2   ; value2 değişkeninin değerini AX yazmacındaki değere ekle
    MOV result, AX   ; AX yazmacındaki değeri result değişkenine taşı
END START
```

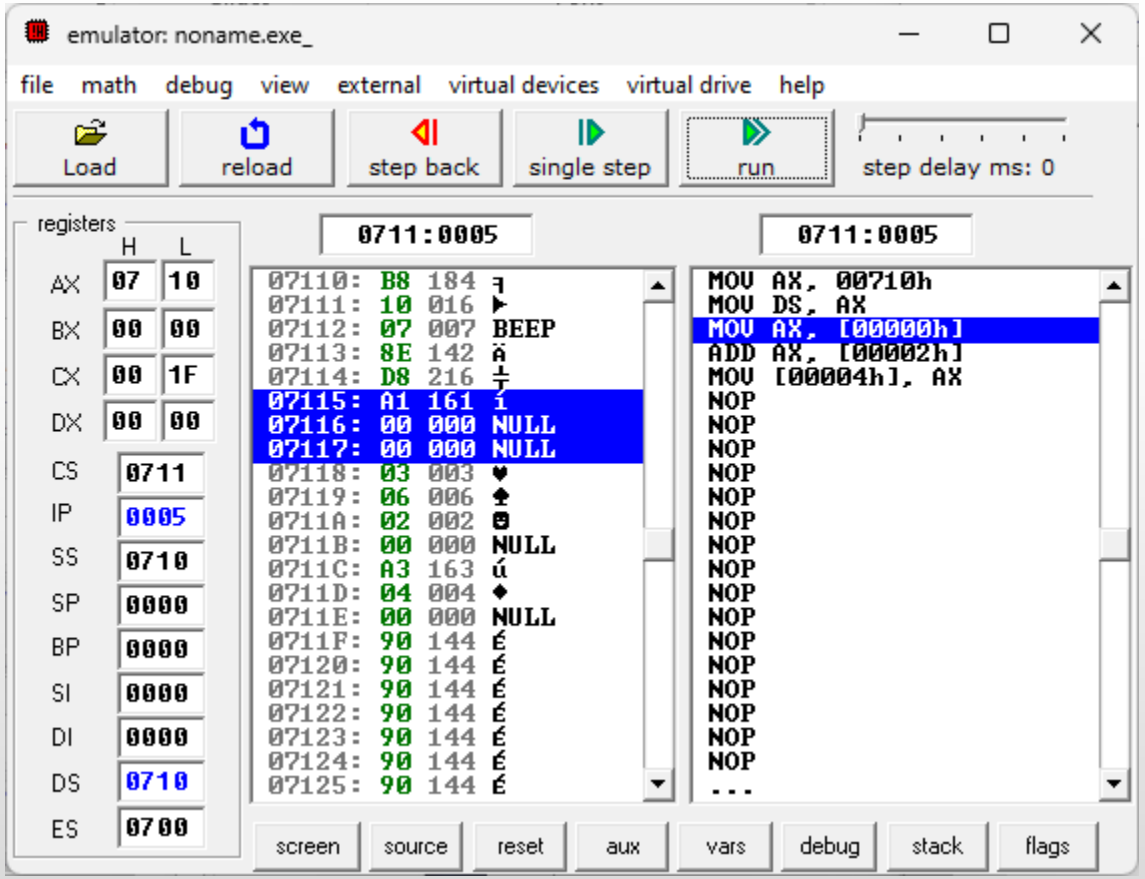
Değişkenleri Toplama İşlemi



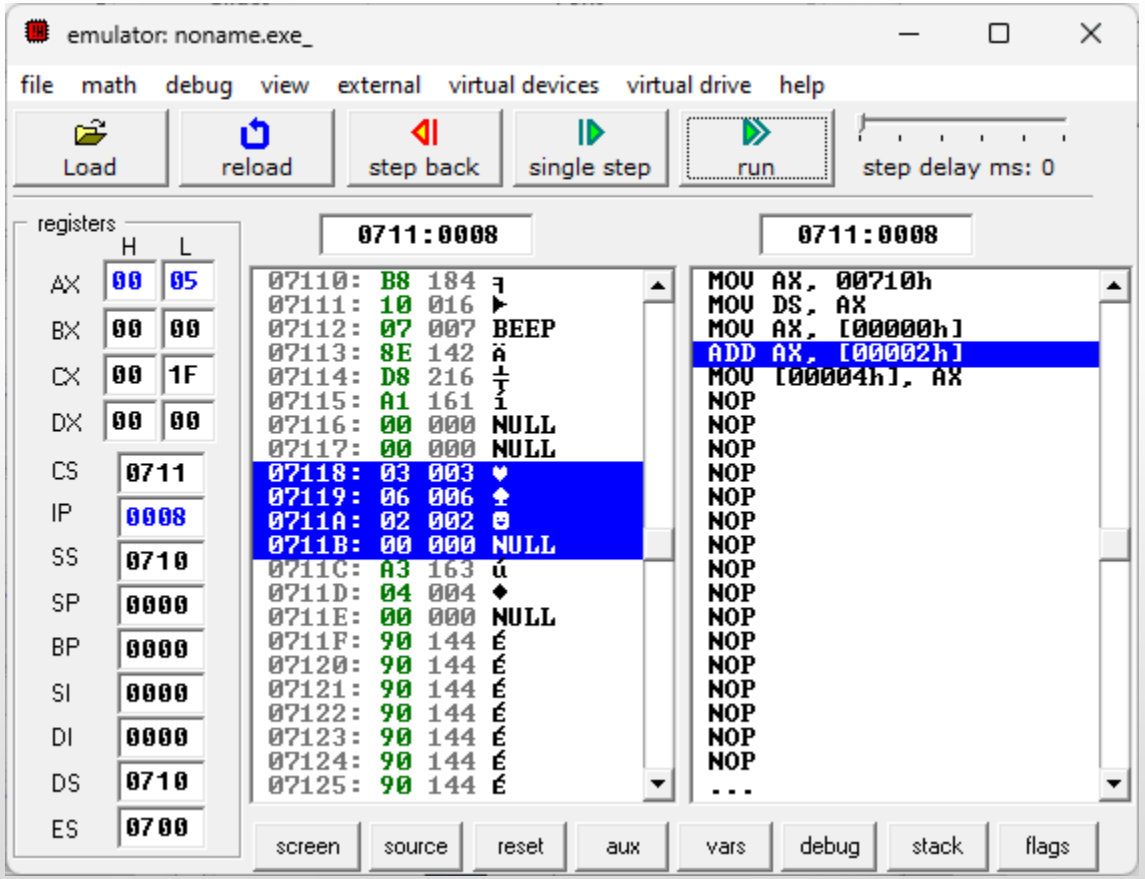
Değişkenleri Toplama İşlemi



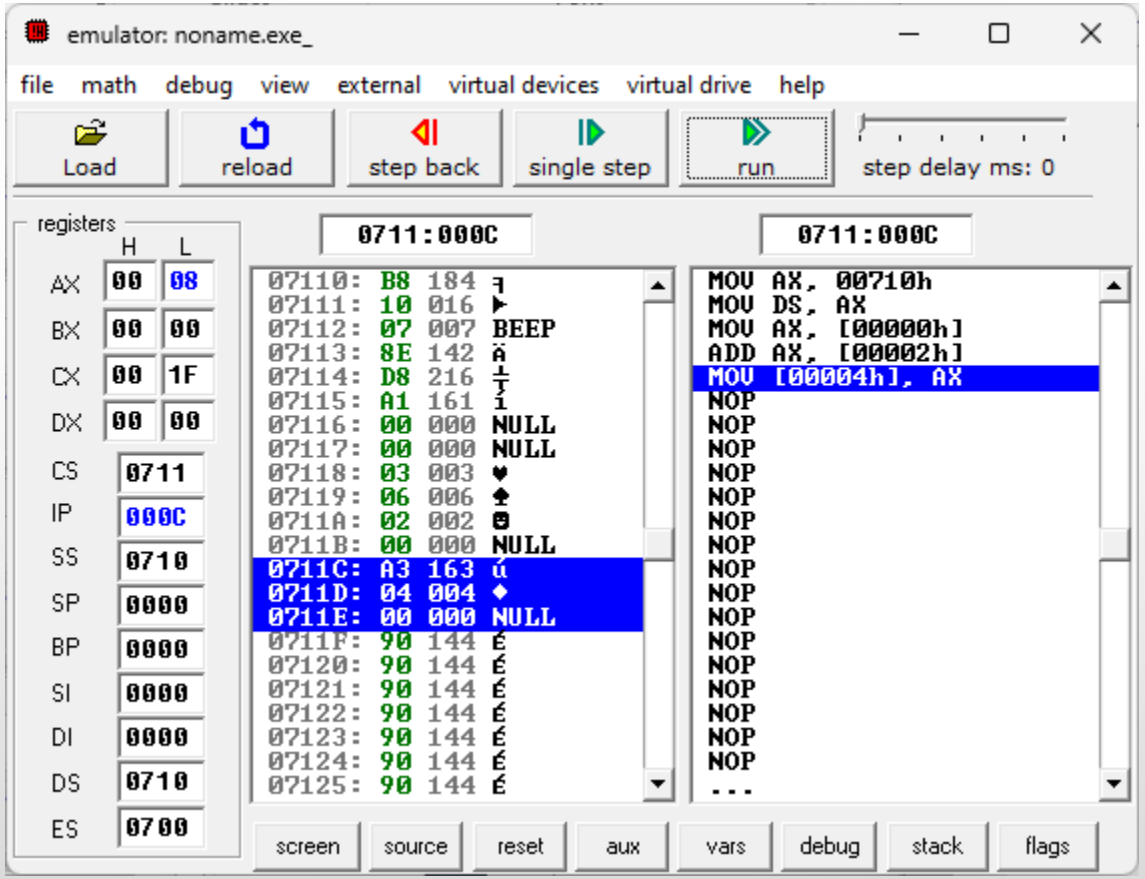
Değişkenleri Toplama İşlemi



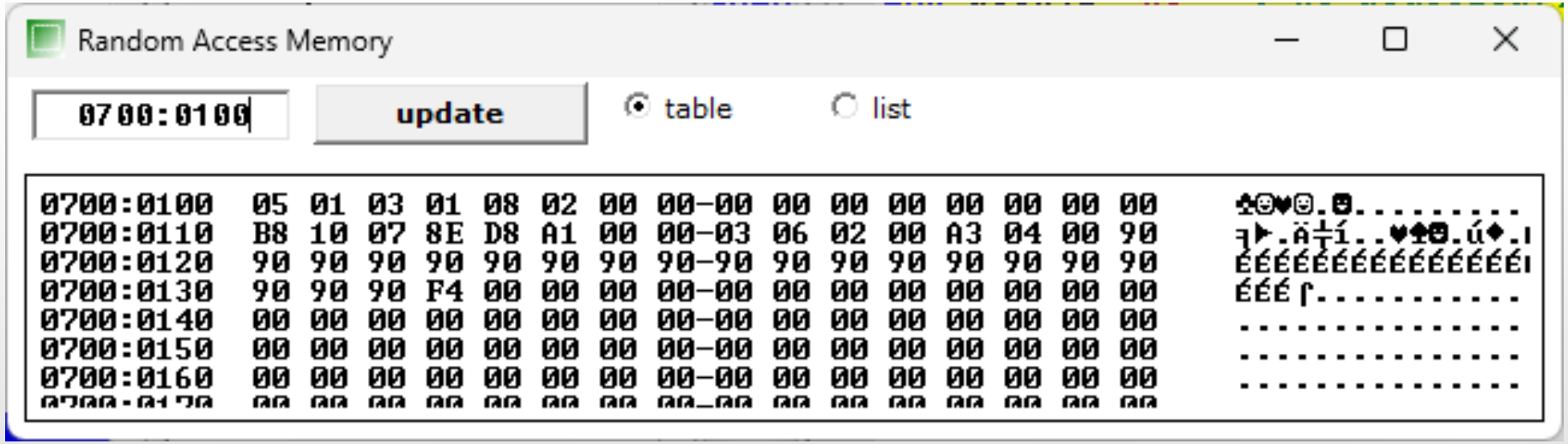
Değişkenleri Toplama İşlemi



Değişkenleri Toplama İşlemi



Değişkenleri Toplama İşlemi



İki Değişkenin Değerini Karşılaştırma

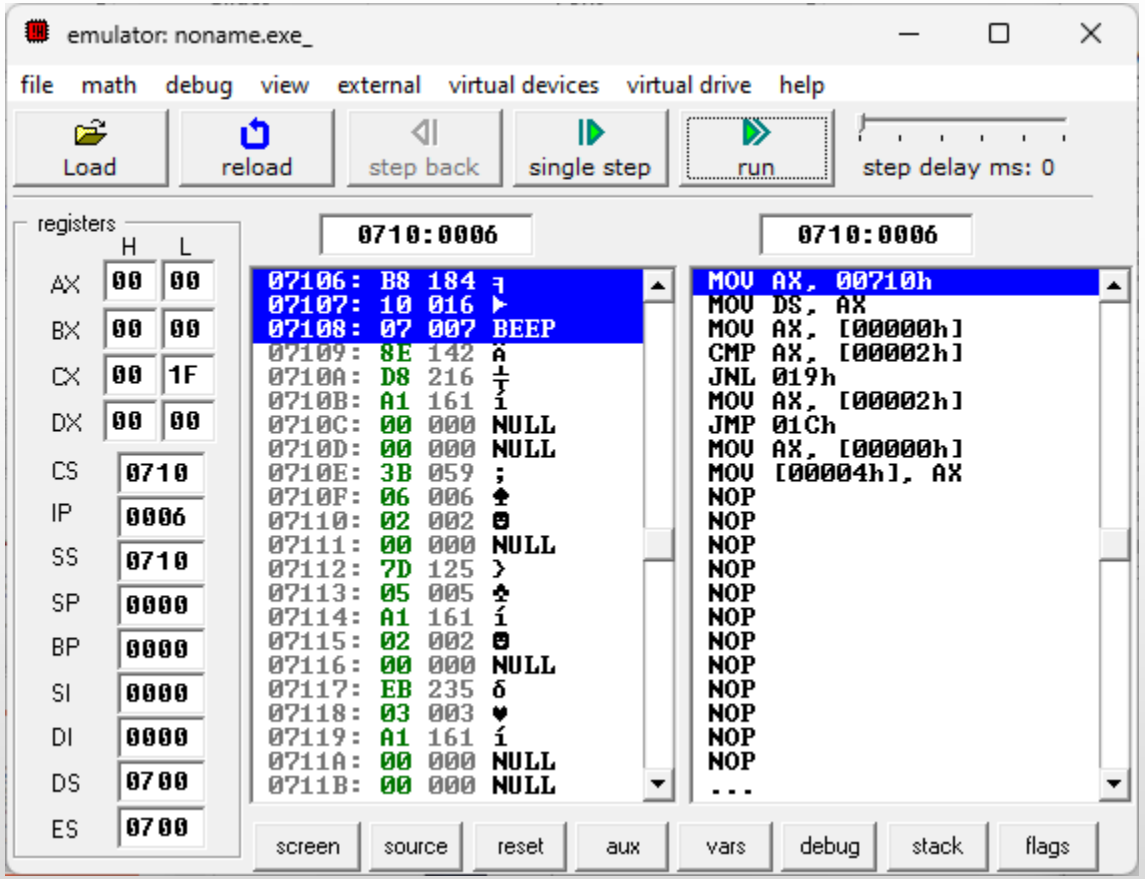
```
DATA SEGMENT
    value1 DW 10      ; Birinci değişken, başlangıç değeri 10
    value2 DW 5       ; İkinci değişken, başlangıç değeri 5
    max_value DW ?    ; Büyük değeri tutacak değişken, ilk değeri belirsiz (?)
DATA ENDS

ASSUME DS:DATA        ; DATA kesiminin adresini DS yazmacına ata
```

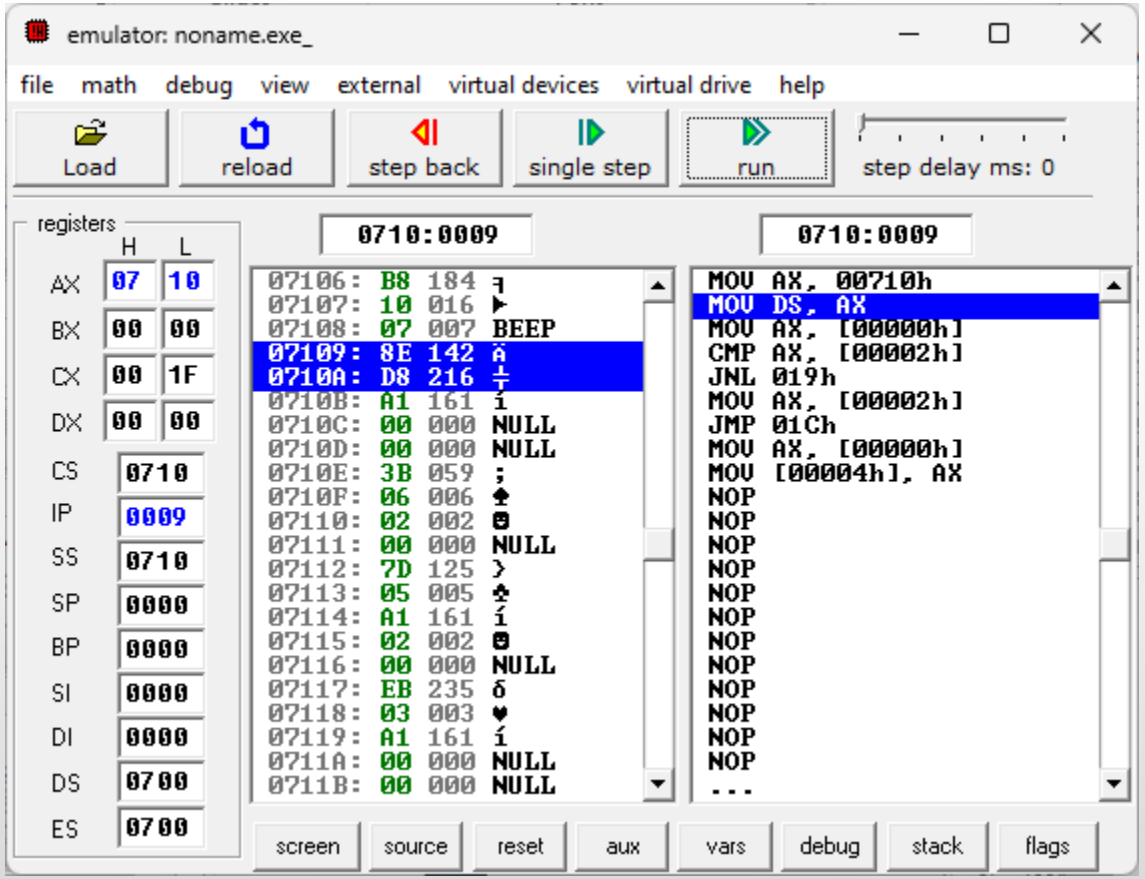
İki Değişkenin Değerini Karşılaştırma

```
START:
    MOV AX, DATA     ; DATA kesiminin adresini AX yazmacına yükle
    MOV DS, AX       ; DS yazmacına DATA kesiminin adresini yükleme
    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
END START
```

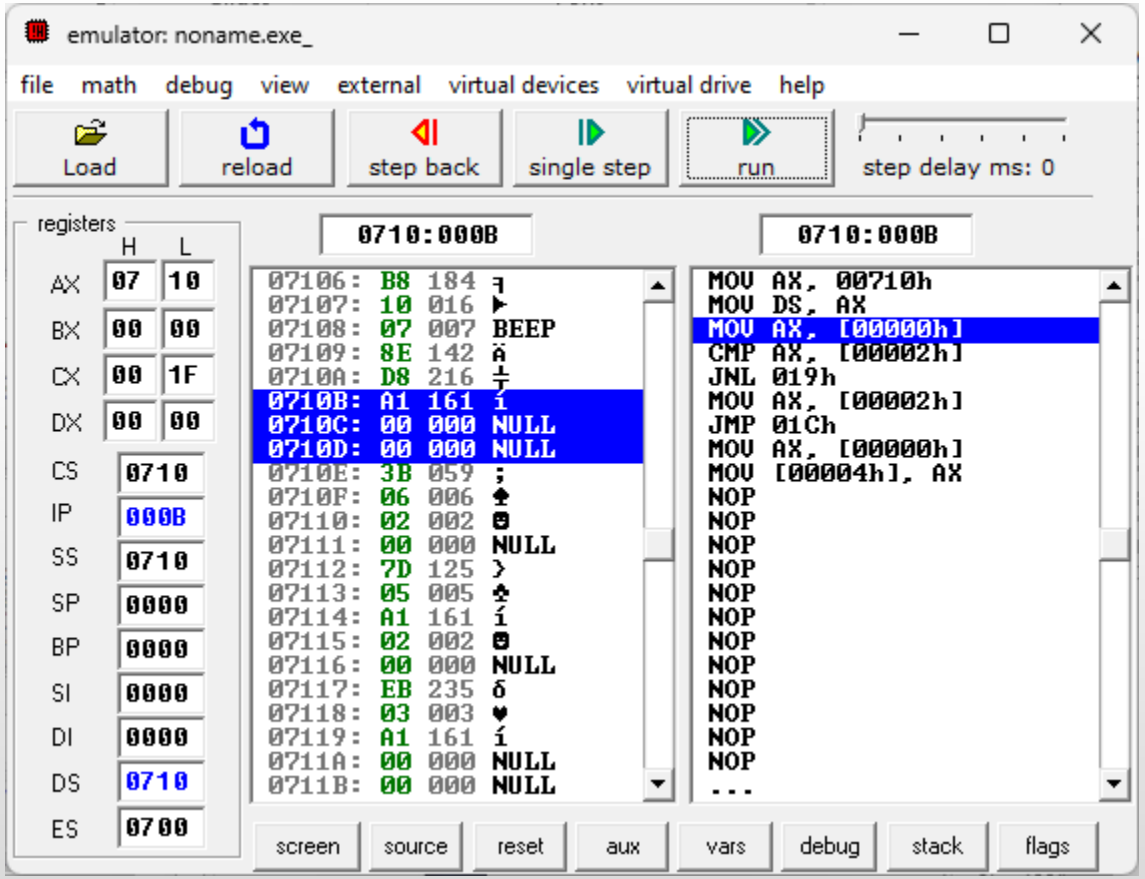
İki Değişkenin Değerini Karşılaştırma



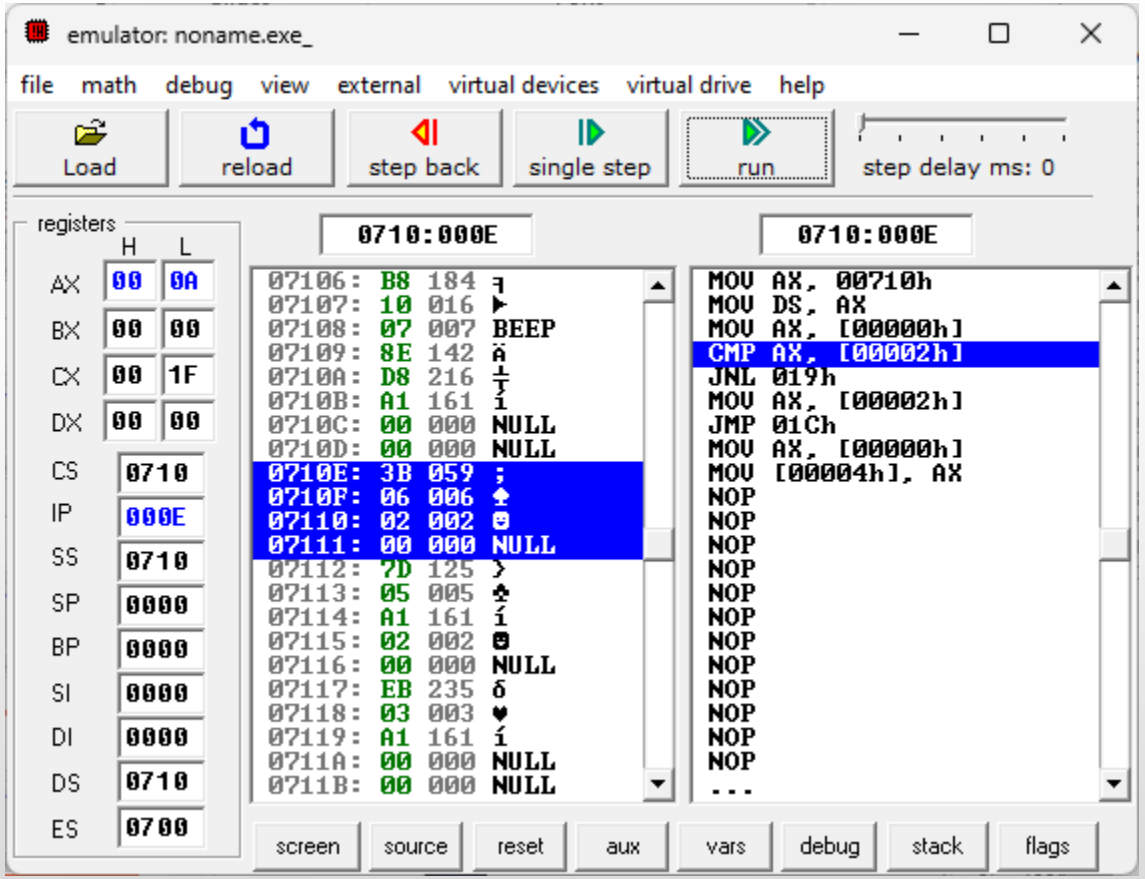
İki Değişkenin Değerini Karşılaştırma



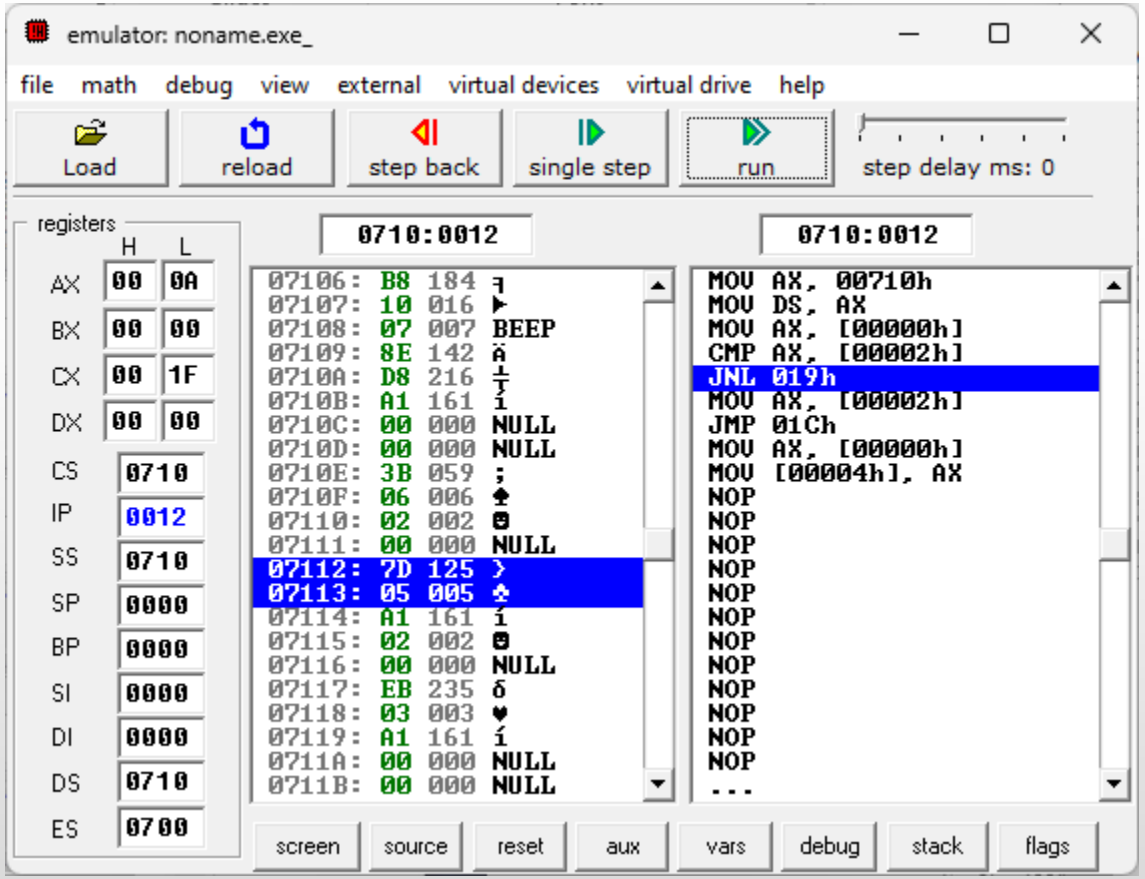
İki Değişkenin Değerini Karşılaştırma



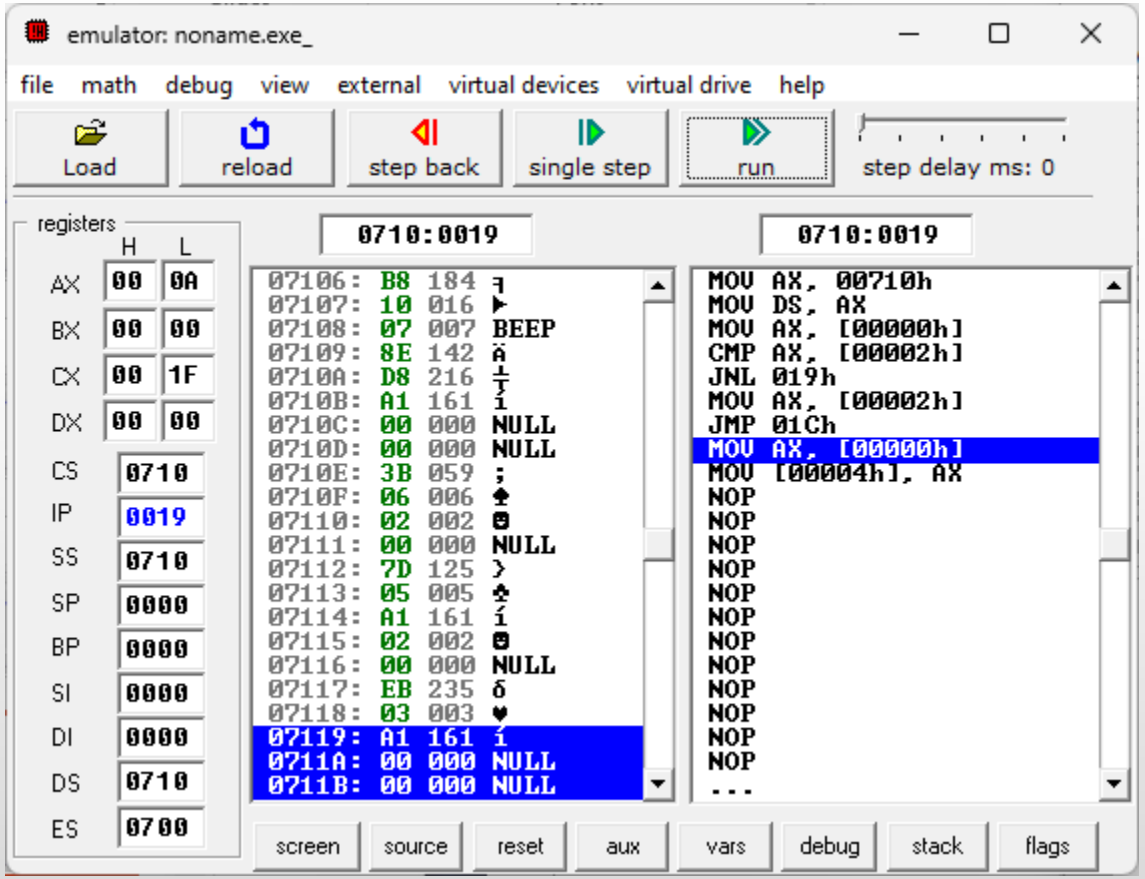
İki Değişkenin Değerini Karşılaştırma



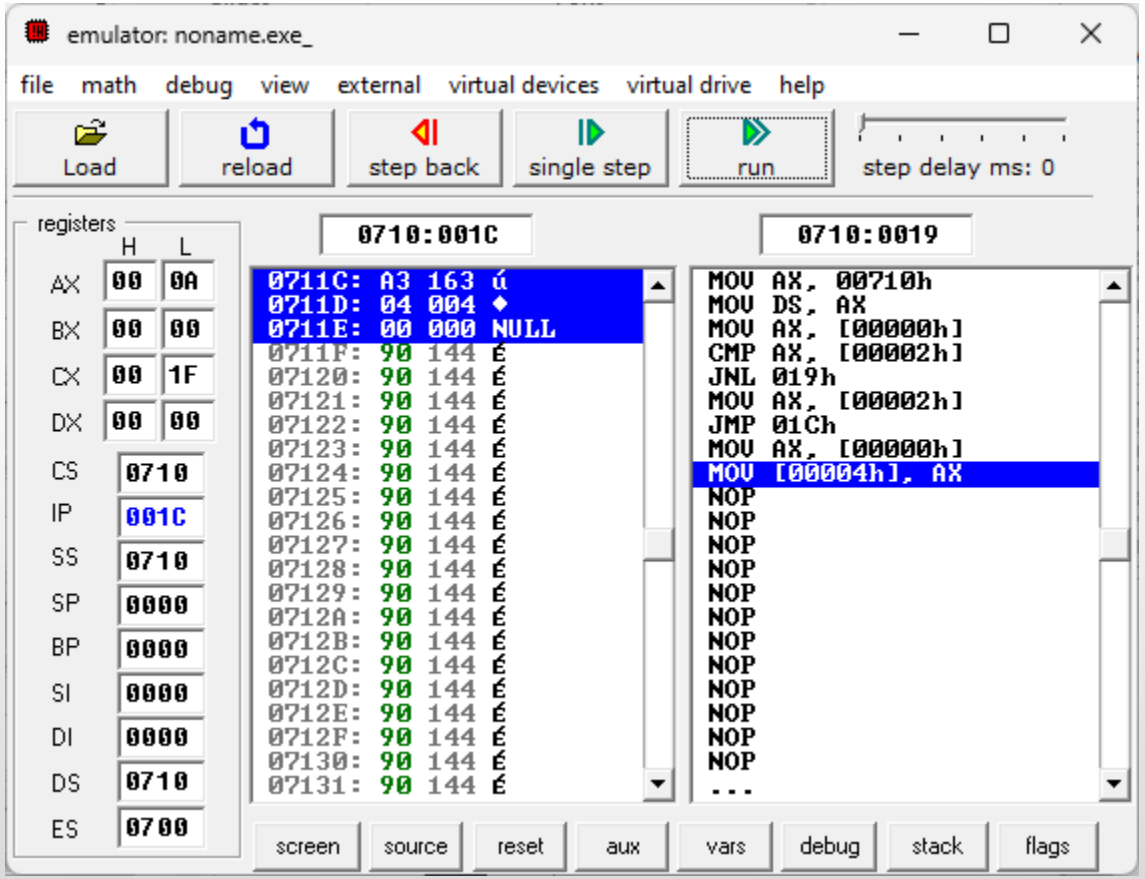
İki Değişkenin Değerini Karşılaştırma



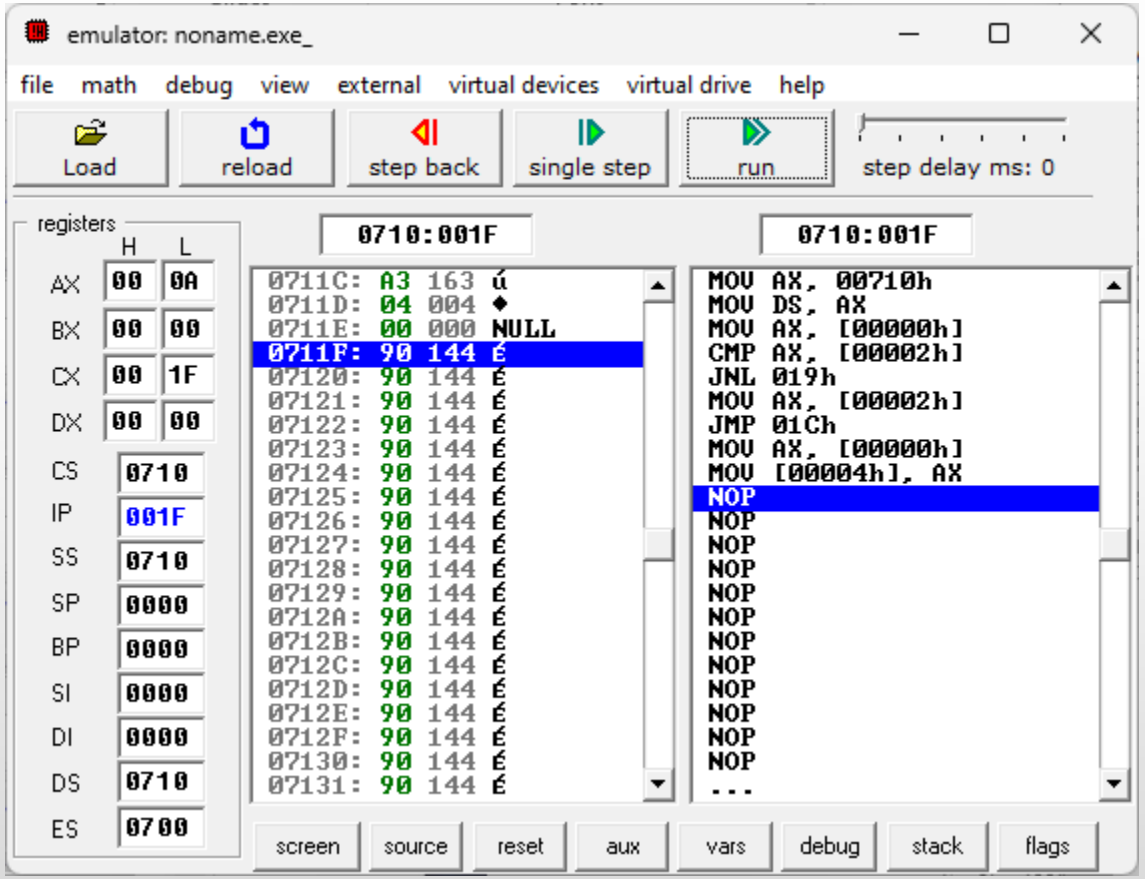
İki Değişkenin Değerini Karşılaştırma



İki Değişkenin Değerini Karşılaştırma



İki Değişkenin Değerini Karşılaştırma



SON