

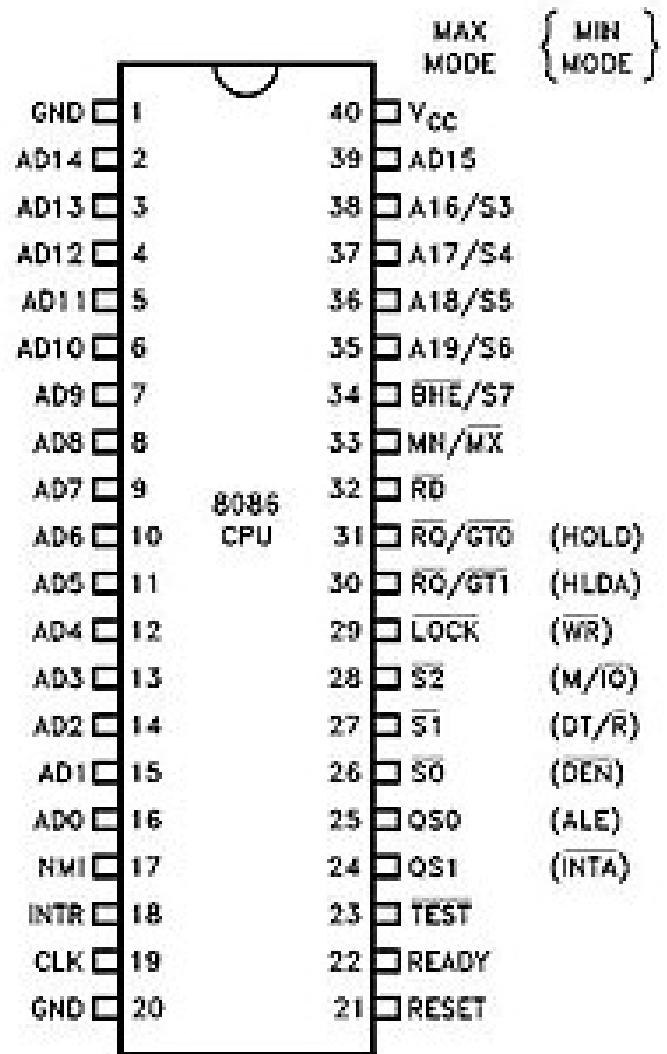
MİKROİŞLEMCI SİSTEMLERİ

Dr. Öğr. Üyesi Meltem KURT PEHLİVANOĞLU

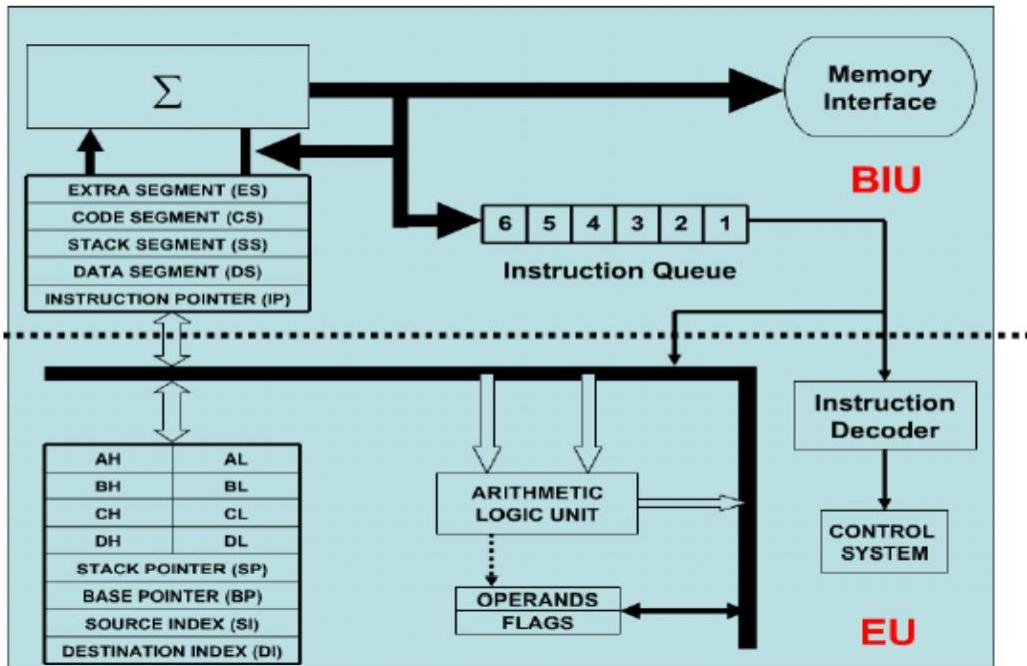
W-3

8086 Mikroişlemci

Ana Kayıtçılar (register)																					
AH				AL				AX (primary accumulator)													
BH				BL				BX (base, accumulator)													
CH				CL				CX (counter, accumulator)													
DH				DL				DX (accumulator, other functions)													
İndeks kayıtçıları																					
SI									Source Index												
DI									Destination Index												
BP									Base Pointer												
SP									Stack Pointer												
Durum Kayıtçıları																					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(bit position)					
-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C	Flags					
Segment kayıtçıları																					
CS									Code Segment												
DS									Data Segment												
ES									ExtraSegment												
SS									Stack Segment												
Eğitim noktası																					
IP									Instruction Pointer												



8086 Mikroişlemci



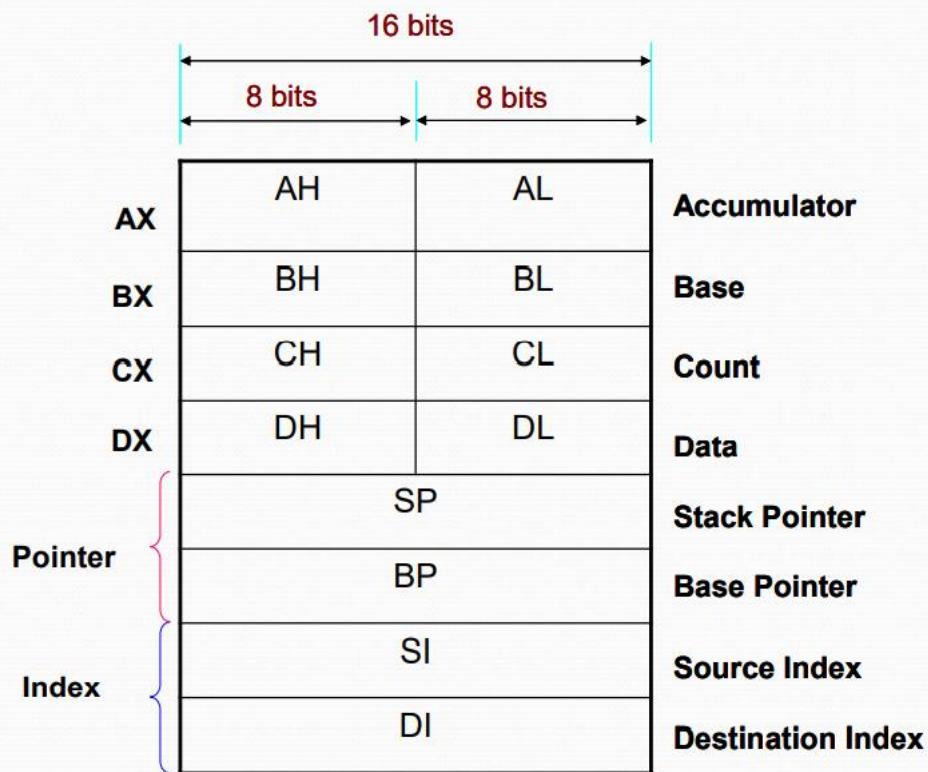
- BIU (Bus Interface Unit): Bus'lar üzerindeki tüm veri ve adres hareketlerini EU için halledebilir.
 - emirleri getirme (instruction fetching),
 - operandların adreslerini hesaplama,
 - belleğe operand yazma/bellekten operand okuma,
 - emir byte'larını emir kuyruğuna (instruction queue) transfer etme, gibi tüm bus işlemlerini yapar
- EU: Emirleri veya verileri hangi adreslerden getireceğini BIU birimine söyler
 - Emirlerin kodunu çözer / Emirleri icra eder (decode & execute)

8086 mimarisinin diyagramı

8086 Mikroişlemci

- x86 komut kümesini kullanan bütün mikroişlemciler (örneğin Intel Pentium ve AMD Athlon serisi) 8086 ile geriye dönük olarak uyumludur. 8086 işlemcisi için yazılan tüm komutlar bugünkü x86 tabanlı bilgisayarlarda çalışabilmektedir.
- Ancak günümüz bilgisayarları için yazılmış olan ve genişletilmiş x86 komut kümesi kullanan yazılımlar 8086 mikroişlemcisi üzerinde çalışmamaktadır.

8086 Mikroişlemci



Register	Purpose
AX	Word multiply, word divide, word I/O
AL	Byte multiply, byte divide, byte I/O, decimal arithmetic
AH	Byte multiply, byte divide
BX	Store address information
CX	String operation, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect I/O (Used to hold I/O address during I/O instructions. If the result is more than 16-bits, the lower order 16-bits are stored in accumulator and higher order 16-bits are stored in DX register)

8086 Mikroişlemci

- 16 bit veri yolu; tek seferde 16 bit veri işleyebilme özelliği (16 bitlik mikroişlemci)
- 20 bitlik adresleme özelliği; $2^{20} = 1\text{MB}$ lık belleğe erişebilme
- 8 adet genel amaçlı 16 bitlik register

- **AX** - accumulator register – akümülatör (**AH / AL**).
- **BX** - the base address register – adres başlangıcı (**BH / BL**).
- **CX** - the count register – sayma (**CH / CL**).
- **DX** - the data register – veri (**DH / DL**).
- **SI** - source index register – kaynak indisi.
- **DI** - destination index register – hedef indisi.
- **BP** - base pointer – temel gösterici.
- **SP** - stack pointer – yiğit gösterici.

8086 Mikroişlemci

- Segment registerlarının özel amaçları vardır, bellekte ulaşılabilir bazı bölümleri işaretler.
- Segment registerleri, genel amaçlı registerleri ile birlikte çalışarak hafızada herhangi bir bölgeyi işaretleyebilir.
- **CS** – (Code Segment) Mevcut programın bulunduğu bölümü işaretler.
- **DS** – (Data Segment) Genellikle programda bulunan değişkenlerin bulunduğu bölümü işaretler.
- **ES** – (Extra Segment) Bu register'in kullanımı, kullanıcıya bırakılmıştır.
- **SS** – (Stack Segment) yiğinin bulunduğu bölümü işaretler.

8086 Mikroişlemci

- **20 bitlik adres hattı;** ($2^{20}=1048576$ byte=1MB) **1MBlık belleği** adresleyebilir.
- **16 bitlik adres hattı;** ($2^{16}=65,536$ byte=64KB) **64 KBlık belleği** adresleyebilir.

8086 Mikroişlemci

- Ancak 16-bitlik registerlar ile en fazla adreslenebilir bellek uzayı 64 KB büyüğündedir. Çünkü tüm dahili registerlar 16-bit büyüğündedir ($2^{16}=65536$ byte=64KB (örn. 0000h-FFFFh))
- 64 KB'lık sınırların dışında programlama yapabilmek için extra operasyonlar (adres bölütleme-segmentation) kullanmak gereklidir.

8086 Mikroişlemci

- 64 KB'lık erişilebilecek adres uzayı bölütlü (segment) adresleme ile 1MB a (00000h-FFFFFh) çıkarılır;
- Şöyle ki

Fiziksel adres = F A = bölüm (segment) adresi \times (16 (decimal ise adres 16 ile çarp) 10(hex ise adres 10 ile çarp)) + offset adresi

Böylece 20 bitlik fiziksel bellek adresi hem adres registerindaki offset, hem de segment registerindaki paragraf adresi değerinden hesaplanır.

(İpucu: program kodu için kod bölümü CS, veri adresleme için veri bölümü DS, extra bölüm ES, yığın adresleme için SS)

8086 Mikroişlemci

- Örneğin, fiziksel adres 12345h (hexadecimal) işaretlenmesi isteniyor ise, DS = 1230h ve SI = 0045h olmalıdır.
- CPU, segment register'ı 10h ile çarpar (çünkü fiziksel adres hexadecimal- decimal olsaydı 16 ile çarpacaktı) ve genel amaçlı register'da bulunan değeri de ilave eder ($1230h \times 10h + 45h = 12345h$).

(DS:Data Segment, SI: Source Index)

8086 Mikroişlemci

- Tüm fiziksel bellek adresleri **segment adresi+ offset adresi** ilave edilmesi ile bulunur.
 - **segment adresi:** Herhangi bir 64 KB'lık hafıza bölümünün başlangıcını gösterir.
 - **offset adresi:** 64 KB'lık hafıza bölümünde herhangi bir satırı belirtir.

8086 Mikroişlemci

Segment ve adres register çiftleri:

CS	IP
	SP
SS	BP
	BX
DS	SI
	DI
ES	DI

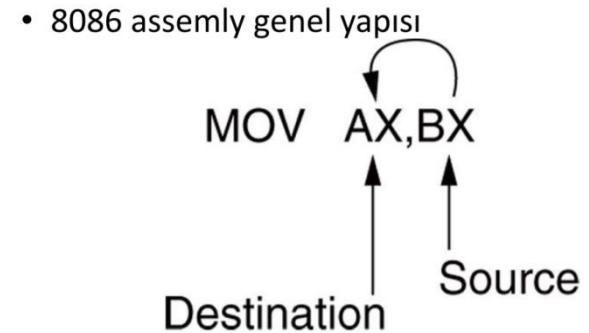
8086 Mikroişlemci

- CS(Code Segment) kod bölümünün başlangıcına işaret eder. IP(Instruction Pointer) kod bölümü içerisinde bir sonraki komutun bulunduğu bellek adresine işaret eder.
- Eğer **CS=1400h** ve **IP=1200h**, mikroişlemci, bir sonraki komutu
 - 14000h ($1400\text{h} \times 10\text{h} = 14000\text{h}$) + $1200\text{h} = 15200\text{h}$ adresinden okur.

8086 Mikroişlemci Adresleme Modları

- Adresleme modları belleğin nasıl kullanıldığını, belleğe nasıl erişileceğini ve verilerin belleğe nasıl yerleştirileceğini belirler.
- REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.
- SREG: DS, ES, SS, and only as second operand: CS.
- bellek: [BX], [BX+SI+7] ...
Belleğe erişmek için bu 4 register kullanılır: BX, BP, SI, DI,
- immediate: 5, -24, 3Fh, 10001101b ...

CS	IP
	SP
SS	BP
	BX
DS	SI
	DI
ES	DI



8086 Mikroişlemci

Hemen Adresleme (Immediate Addressing)

- **Sabit bir değer** (immediate) bir register'a aktarılır.
- Sabit değerin büyüklüğü ile register uyumlu olmalıdır.
- Örneğin 8 bitlik bir register'a 16 bitlik bir değer yüklenemez.
- Genel Kullanımı: **KOMUT register, immediate**
 - MOV CL, 16h
 - MOV DI, 2ABFh
 - MOV AL, 4567h (Yanlış kullanım-8 bitlik register AL)

8086 Mikroişlemci

Doğrudan adresleme (Direct addressing)

- Doğrudan bir adres değeri kullanılır diğer bir ifadeyle erişilecek hafıza gözünün doğrudan gösterildiği durumdur.
- Bir adresten bir registera veri aktarımı gerçekleştirilir. Bir başka ifade ile operandlardan birisi adres belirtir.
- Genel kullanımı:
 - KOMUT register, memory veya
 - KOMUT memory, register
- Örnek:

MOV AX, [1000h] ; 1000h adresindeki değeri AX e ata

8086 Mikroişlemci Adresleme Modları

MOV CL, 'A'

MOV CH, 'B'

MOV BX, 15Eh

MOV [BX], CX ; DS:BX adresine BA veri aktarılır

8086 Mikroişlemci

Kaydedici Adresleme (Register Addressing)

- Bu adresleme modunda her iki operand da registerdir.
- Genel kullanımı: **KOMUT register, register**

Örnek:

MOV AL, BL

INC BX

DEC AL

SUB DX, CX

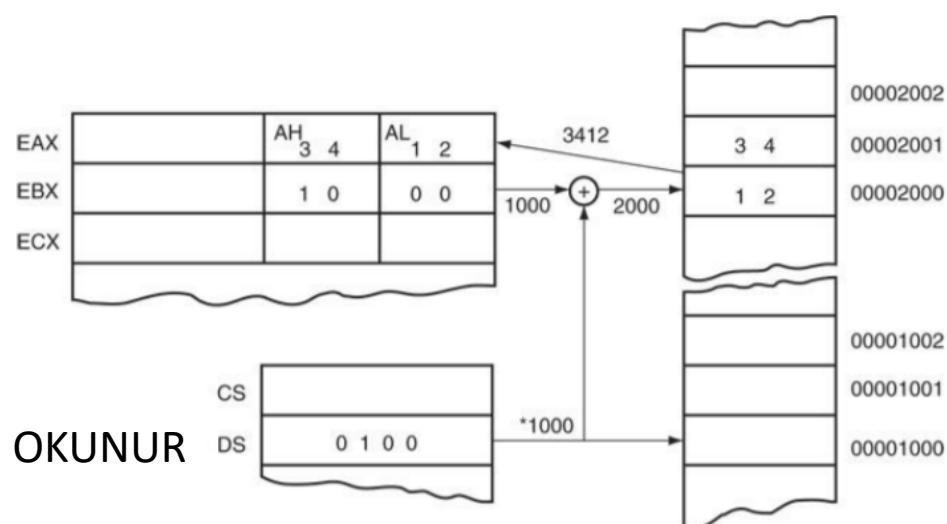
8086 Mikroişlemci

Register Dolaylı adresleme (Register Indirect addressing)

- Etkin adres değeri (**offset**) BX, BP, SI, DI registerlarından birinde bulunur.
- BP (SS ile), BX, DI ve SI (DS ile) yazmaçları ile kullanılabilir
- Genel kullanımı:
 - **KOMUT register, [BX/BP/SI/DI] veya**
 - **KOMUT [BX/BP/SI/DI], register**

- Örnek:

MOV AX, [BX] ; AX \leftarrow DS:BX

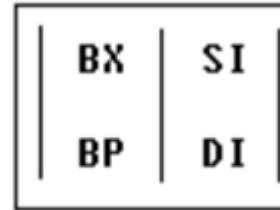


SEGMENT ADRES:OFFSET ADRES

Komut SEGMENT ADRES*10+OFFSET ADRESTEN OKUNUR

8086 Mikroişlemci

BASE+INDEX Adresleme



[BX + SI]
[BX + DI]
[BP + SI]
[BP + DI]

- Temelde bir dolaylı adresleme modudur
 - Base registeri (BX veya BP) işlem yapılacak bellek konumunun başlangıcını göstermek için kullanılır
 - Index registerleri (DI veya SI) verinin bu başlangıç adresine görece yerini tutmak için kullanılır
 - Genel kullanımı:
 - **KOMUT register, [BX+SI] / [BX+DI] / [BP+SI] / [BP+DI] veya**
 - **KOMUT [BX+SI] / [BX+DI] / [BP+SI] / [BP+DI], register**
- MOV DX, [BX+DI]

BX ve BP asla toplanmaz!

8086 Mikroişlemci

[SI + d8]	[SI + d16]
[DI + d8]	[DI + d16]
[BP + d8]	[BP + d16]
[BX + d8]	[BX + d16]

Yazmaç Göreli Adresleme (Register Relative Addressing)

- Base (BP veya BX) veya Index (DI, SI) yazmaçlarının bir sabit ofset değeri ile kullanılmasını ifade eder

Sabit offset değerleri:d8 veya d16

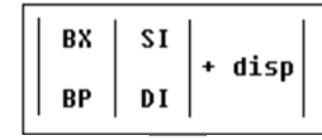
d8 - 8 bit işaretli immediate (örneğin: 22, 55h, -1,...)

d16 -16 bit işaretli immediate (örneğin: 300, 5517h, -259,...).

- Genel kullanımı:
 - KOMUT register, [BX+d8] / [DI+d8] / [BP+d8] / [SI+d8] veya
 - KOMUT [BX+d8] / [DI+d8] / [BP+d8] / [SI+d8], register
 - KOMUT register, [BX+d16] / [DI+d16] / [BP+d16] / [SI+d16] veya
 - KOMUT [BX+d16] / [DI+d16] / [BP+d16] / [SI+d16] , register

8086 Mikroişlemci

Base Relative + Index Addressing



[BX + SI + d16]	[BX + SI + d8]
[BX + DI + d16]	[BX + DI + d8]
[BP + SI + d16]	[BP + SI + d8]
[BP + DI + d16]	[BP + DI + d8]

- Base index ve yazmaç göreceli adresleme modlarının birleşimi gibi düşünebiliriz.
- Genel kullanımı:
 - KOMUT register, [BX+SI+d8] / [BX+DI+d8] / [BP+SI+d8] / [BP+DI+d8] / [BX+SI+d16] / [BX+DI+d16] / [BP+SI+d16] / [BP+DI+d16] veya
 - KOMUT [BX+SI+d16] / [BX+DI+d16] / [BP+SI+d16] / [BP+DI+d16] / [BX+SI+d8] / [BX+DI+d8] / [BP+SI+d8] / [BP+DI+d8], register

MOV AX, [BX + SI] + 25 ile MOV AX, [BX + SI+25] aynıdır

8086 Mikroişlemci

Base Relative + Index Addressing

- Örneğin;

DS = 100,

BX = 30,

SI = 70

Mikroişlemci tarafından hesaplanan fiziksel adres $[BX + SI] + 25$

$$= 100 * 16 + 30 + 70 + 25 = 1725 \text{ olur}$$

MOV AX, $[BX + SI] + 25$; 1725 fiziksel adresindeki değeri AX registerine at

8086 Mikroişlemci

- KOMUT **memory, memory** şeklinde bir kullanım **geçerli değildir**. Bir bellek bölgesinden başka bir bellek bölgесine veri aktarımı yoktur.
- **Sabit bir değer** doğrudan **Segment Registerine atanamaz**

MOV DS,1234h Yanlıştır.

Bunun yerine aşağıdaki kullanım geçerlidir.

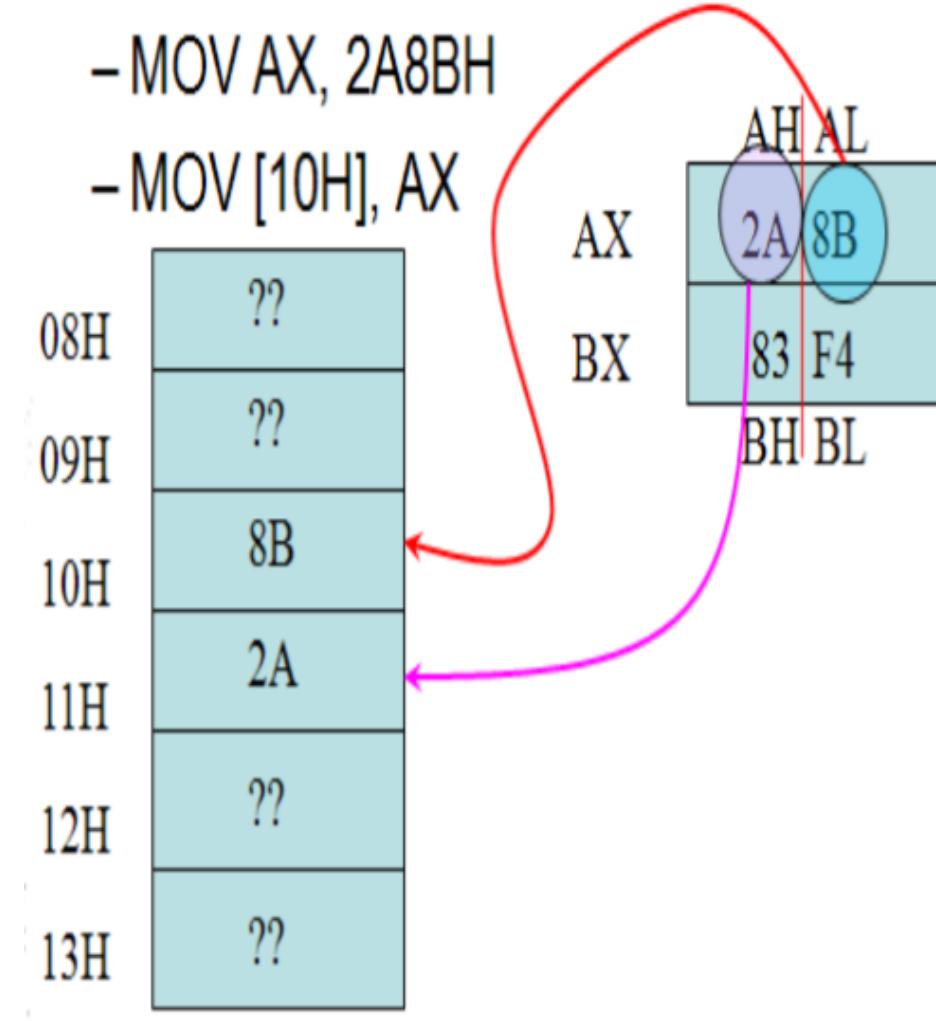
MOV AX, 1234h ; önce registera

MOV DS, AX ; sonra segment kaydedicisine değer atanır.

8086 Mikroişlemci

16 Bitlik Verinin Belleğe Yerleşmesi

- Belleğin her bir hücresi 8 bit olduğundan 16 bitlik verinin düşük değerlikli 8 bitlik kısmı adresin düşük değerlikli kısmına, yüksek değerlikli 8 bitlik kısmı adresin yüksek değerlikli kısmına yerlesir.
- Yanda verilen örnekte AX kaydedicisinde 2A8BH değeri yer almaktadır.
- Düşük değerlikli değer AL de yer almaktır ve 10H adresine yerleşmektedir. (alt byte yani AL ilk)
- Yüksek değerlikli kısımda yer alan AH daki değer 2AH değeri ise 11H adresine yerleşmektedir (Üst byte (AH) ise bir sonraki adreste tutulur)



8086 Mikroişlemci

Yığın Adresleme

- Tüm yazmaçlardan yığına veri basılabilir
- CS hariç tüm yazmaçlara yığından veri çekilebilir

PUSH CS ; çalışır

POP CS ; assembler hatası verir

- 8086'da yığın geçici veri saklamak için ve fonksiyonlardan dönüşlerde dönüş adreslerini saklamak için kullanılır

8086 Mikroişlemci

Yığın Adresleme

- Yığın LIFO mantığında çalışır (Last In First Out)
- Yığın ile ilgili PUSH ve POP komutları kullanılır.
- SP yazmacı programcının tanımladığı yığının genişliğini gösterecek şekilde ilk değer alır
- Her PUSH işleminde SP-1 ve SP-2 adreslerine 2 byte veri yazılır ve SP değeri 2 azaltılır
- Her POP işleminde SP+1 ve SP+2 adreslerinden 2 byte veri okunur ve SP değeri 2 arttırılır

8086 Mikroişlemci Ayrılmış Bölgeler

- Belleğin bazı bölgeleri, bir takım özel verileri tutmak için ayrılmıştır.
- Örneğin belleğin 0000H ile 03FFH adresleri arasında kalan bellek alanı kesme yöneyleri (vektörleri) için ayrılmıştır.
- FFFF0H ya da FFFFFH adresleri bilgisayar ilk açıldığında ya da yeniden başlatıldığında başlangıç adresi olarak kullanılır.

8086 Mikroişlemci

Kesmeler

- Kesmeler, işlemcinin içinde çalışan programın akışını durdurup denetimi özel bir kesme yordamına aktaran özel komutlar ya da sinyallerdir.
- Çoğu kesme komutunun işlenmesinin ardından kesme yordamı işini tamamladığında program kaldığı yerden çalışmayı sürdürür.
- Kesmeler yazılım ya da donanım kullanılarak uygulanabilir.

8086 Mikroişlemci Komutlar

8086 mikroişlemcisinin komut kümesi şu alt başlıklara ayrılabilir:

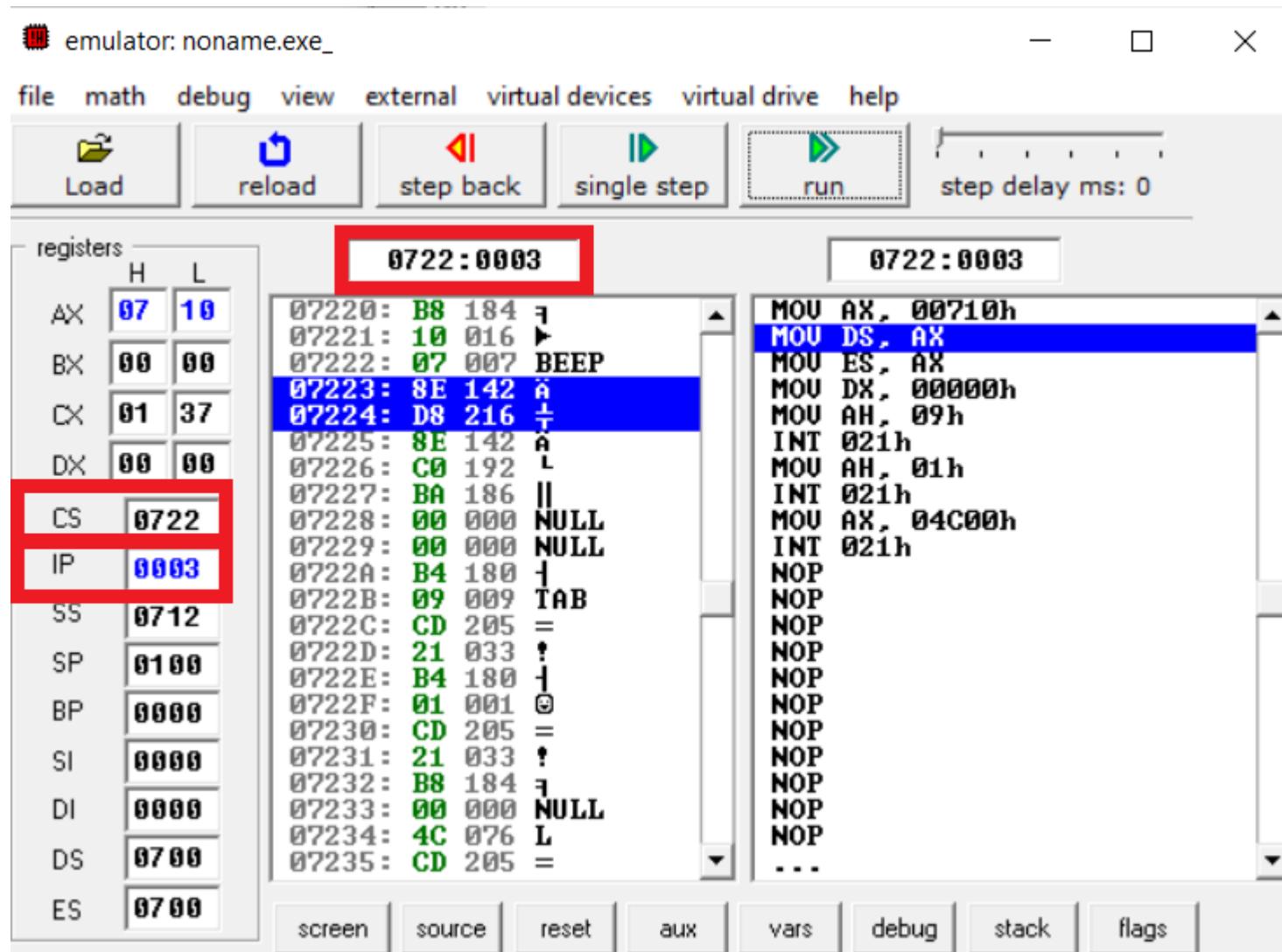
Veri Taşıma Komutları: MOV, PUSH, POP, XCHG, IN, OUT, XLAT, XLATB, LEA, LDS, LES, LAHF, SAHF, PUSHF, POPF

- Aritmetik İşlemi Komutları: ADD, ADC, INC, AAA, DAA, SUB, SBB, DEC, NEG, CMP, AAS, DAS, MUL, IMUL, AAM, DIV, IDIV, AAD, CBW, CWD
- Mantık İşlemi Komutları: NOT, SHL, SAL, SHR, SAR, ROL, ROR, RCL, RCR, AND, TEST, OR, XOR
- Dizgi İşleme Komutları: REP / REPE / REPNE / REPNZ / REPZ, MOVS / MOVSW, CMPS / CMPSW, SCAS / SCASW, LODS / LODSW, STOS / STOSW
- Denetimi Aktarma Komutları: CALL, JMP, RET / RETF, JE / JZ, JL / JNGE, JLE / JNG, JB / JNAE, JBE / JNA, JP / JPE, JO, JS, JNE / JNZ, JNL / JGE, JNLE / JG, JNB / JAE, JNBE / JA, JNP / JPO, JNO, JNS, LOOP, LOOPZ / LOOPE, LOOPNZ / LOOPNE, JCXZ, INT, INTO, IRET
- İşlemci Denetimi Komutları: CLC, CMC, STC, CLD, STD, CLI, STI, HLT, WAIT, ESC, LOCK, NOP

8086 16-Bit Mikroişlemci

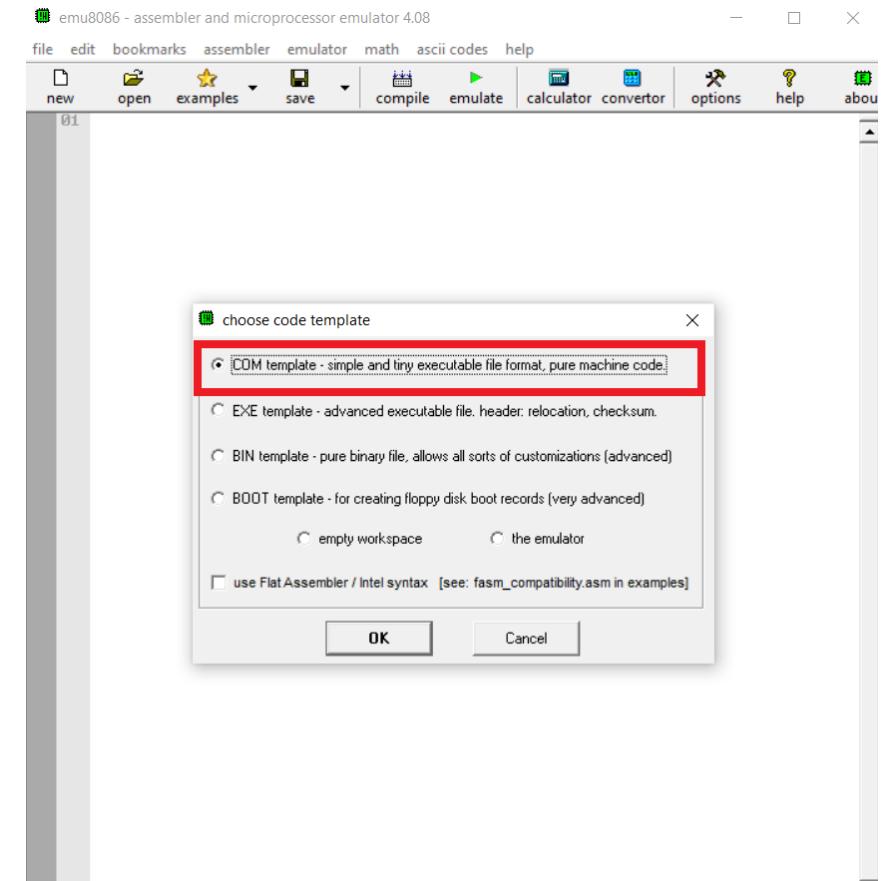
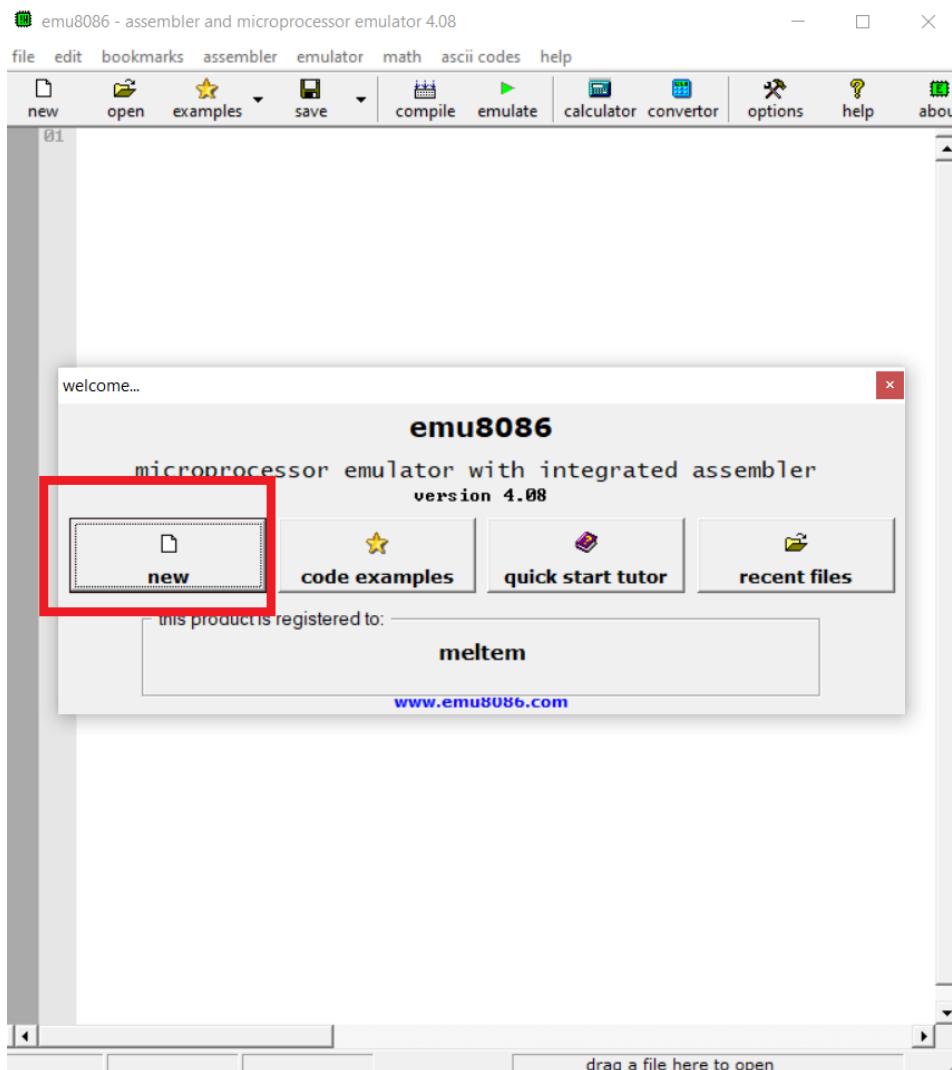
- Fiziksel Adres: **segment adresi + offset adresi**
 - Segment Adres: Hex ise;
Segment Register \times 10H ile çarpar
 - Segment Adres: Decimal ise;
Segment Register \times 16 ile çarpar
- Örn. **CS=0722H** ve **IP=0003H**, mikroişlemci, bir sonraki komutu
 - **07220H** (**0722H** \times 10H=**07220H**) + 0003H = **07223H** adresinden okur.

8086 16-Bit Mikroişlemci



8086 16-Bit Mikroişlemci

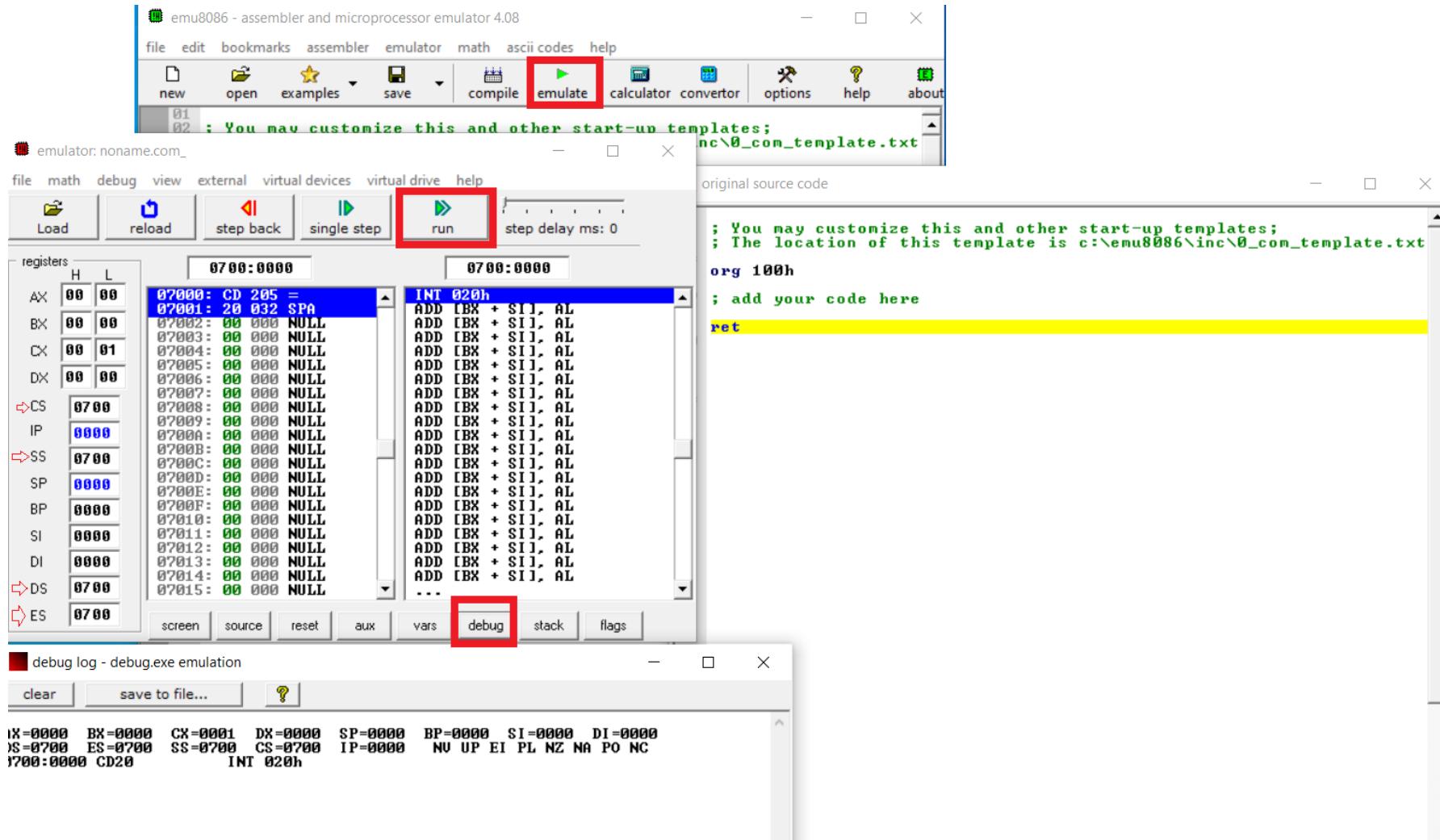
EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

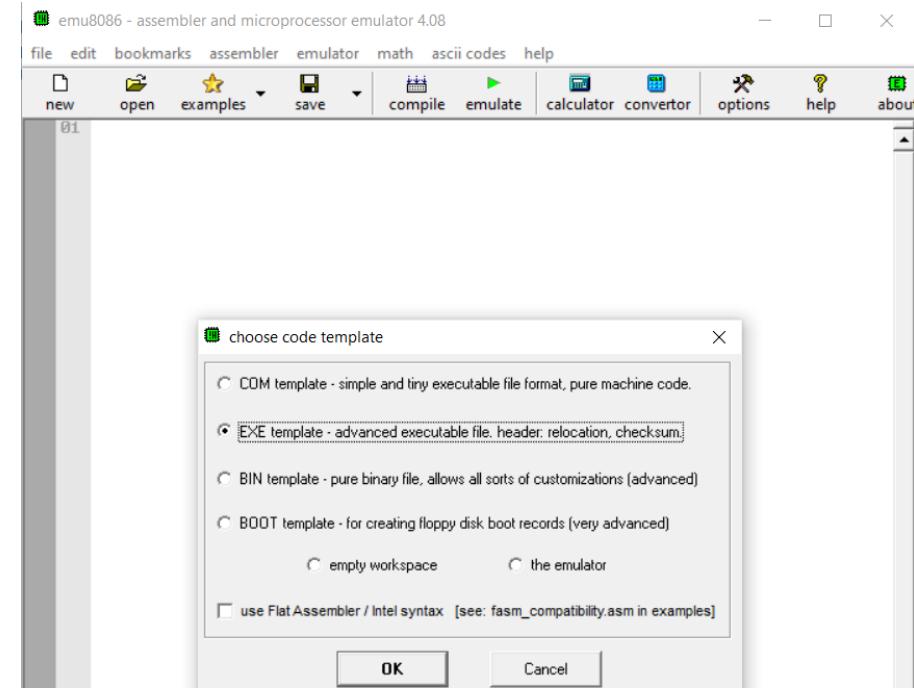
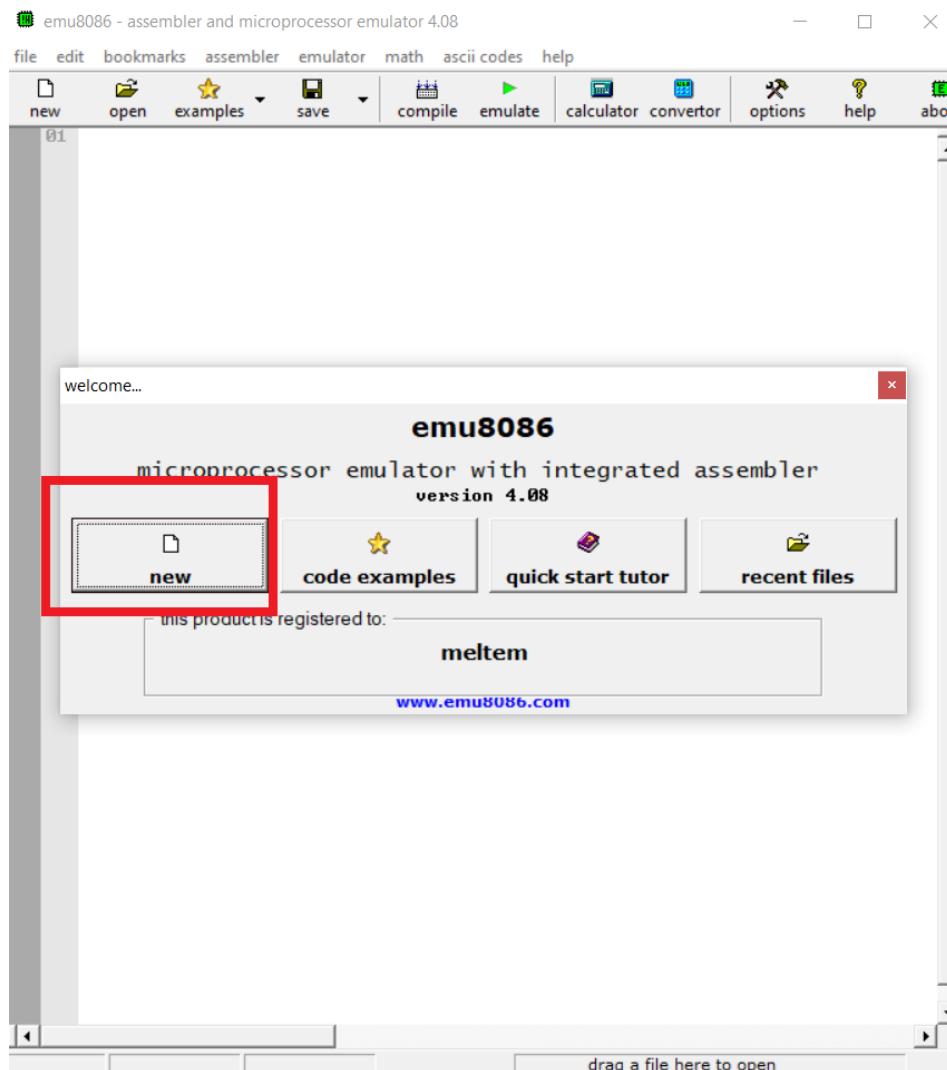
EMU 8086-MICROPROCESSOR EMULATOR

- Com template oluşturduk: 64 KB segment var tüm segment registerleri aynı adresden başlıyor



8086 16-Bit Mikroişlemci

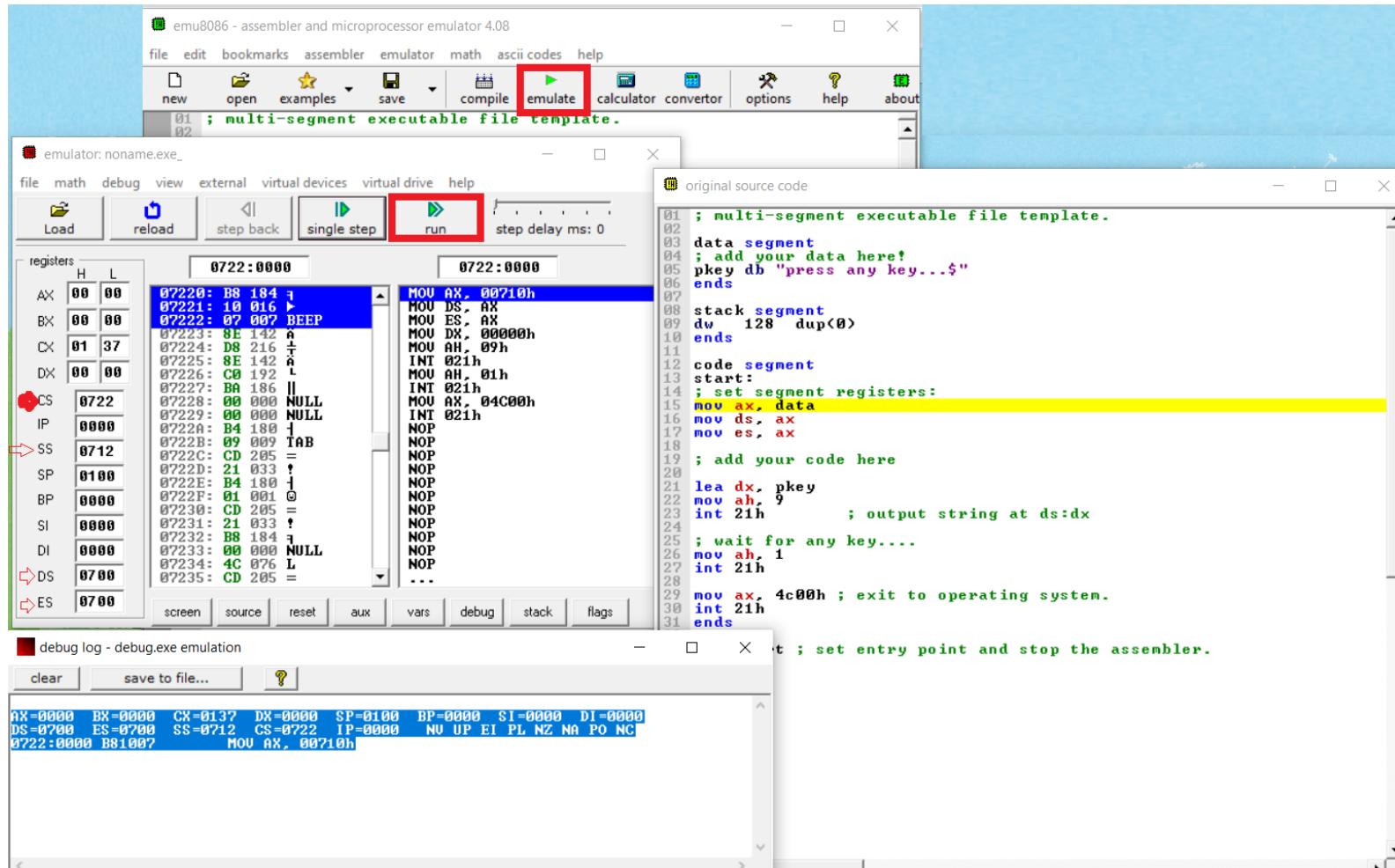
EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

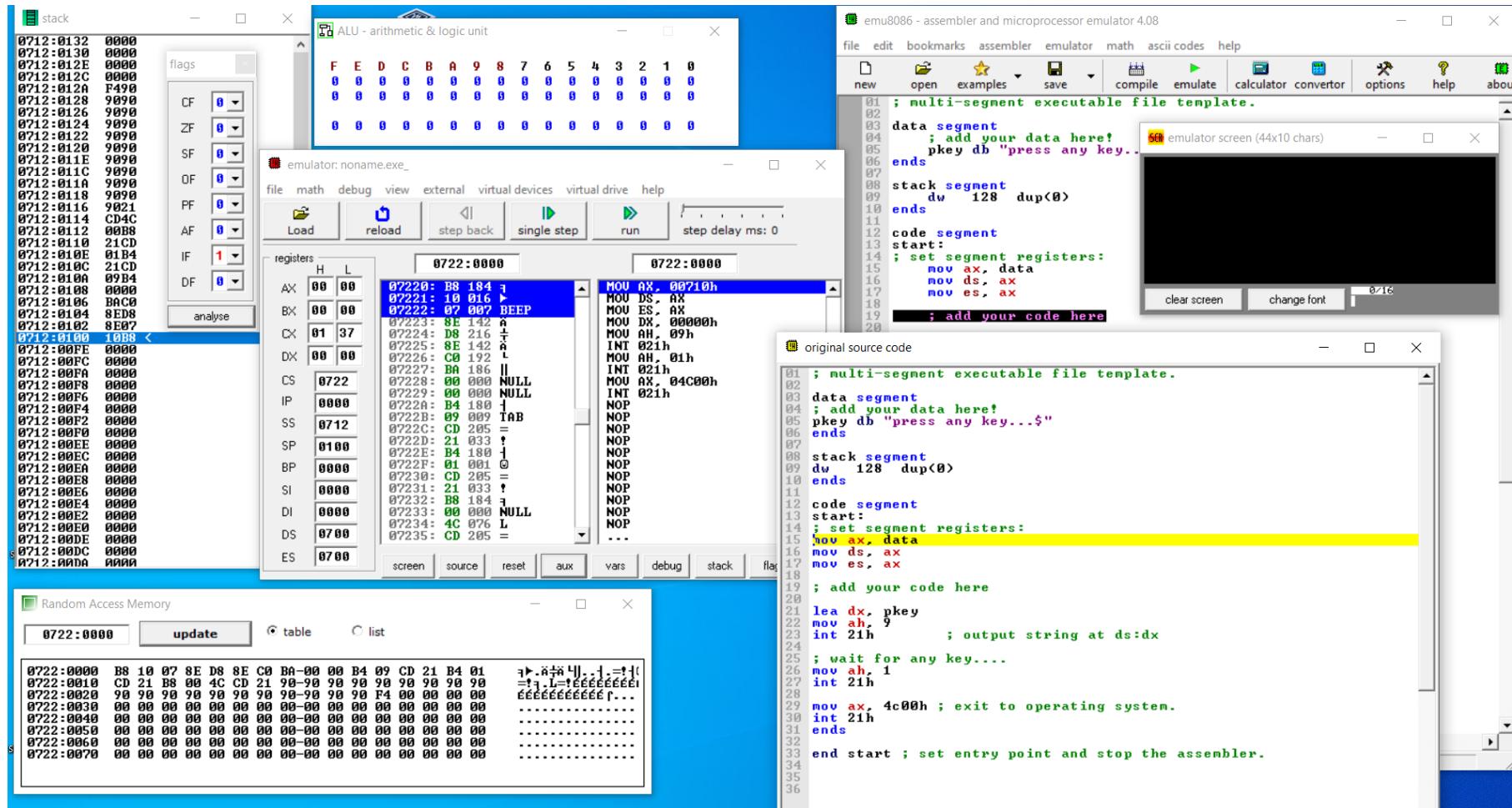
EMU 8086-MICROPROCESSOR EMULATOR

- Exe template oluşturduk: Ayrı ayrı segment registerlarının tanımlamalarını yapabiliyoruz



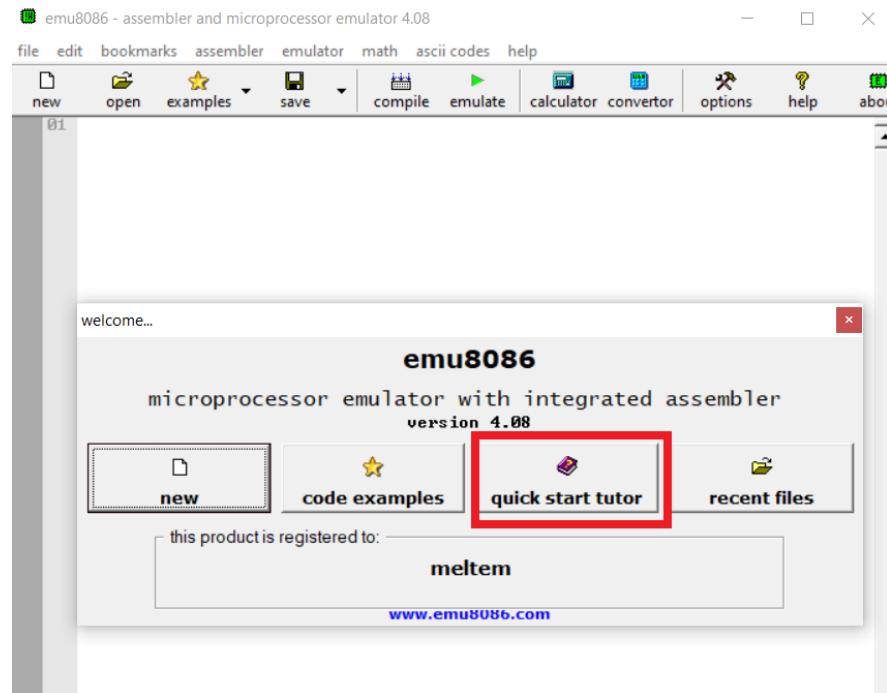
8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



A screenshot of a web browser displaying the documentation for emu8086. The URL is "C:/emu8086/documentation/start.html". The page header includes links for Documentation Index, Licence, Tutorials, 8086 Instruction Set, and Interrupts. The main content is titled "documentation for emu8086 - assembler and microprocessor emulator" and lists several topics: Where to start?, Assembly Language Tutorials, Working with The Editor, How to Compile The Code, Working with The Emulator, Complete 8086 Instruction Set, Supported Interrupt Functions, Global Memory Table, Custom Memory Map, Masm / Tasm compatibility, and I/O ports and Hardware Interrupts. At the bottom, a note states that the reference and tutorials were checked and partly re-written by Daniel B. Sedory (aka The Starman) and provides a link to his realm.

- **quick start tutor** sekmesinden 8086 ile ilgili tüm detaylı bilgilere ulaşabilirsiniz.

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

; ile açıklama satırı ekleniyor

- emu8086 büyük küçük harf duyarlı değil bu nedenle değişken tanımlamalarına dikkat etmek gereklidir çünkü ab=Ab=AB=aB dir

komut operand1, operand2

Komutlar (instruction) en fazla **2 operand** alabilir

Tek operand ve operandsız komutlar bulunur

exe uzantılı dosya için aşağıdaki kod işletim sistemine donus komutudur her programda eklenmeli:

mov ax, 4c00h

int 21h ; (int: interruptın kısaltmasıdır)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

Degiskenadi DB deger (DB:define byte 8-bit)

Degiskenadi DW deger (DW:define word16-bit)

sayi1 db 12

Sayi2 dw 15h

sayilar2 db 3 dup(8) ; 3 tane 8 sakla dup(duplicate)

sayilar3 db 2 dup(1,2,3) ; 1,2,3,1,2,3 sakla

sayilar4 db 1,2,4 dup(3),4 ; 1,2,3,3,3,3,4 sakla

sayilar5 db 3 dup(?) ; 3 tane bos alan bırak

sayilar6 dw 13h,124Fh,0021h ; sayı dizisi tanımlama

metin db 'okul'

faiz equ 8 ; faiz diye bir sabit tanımlıyoruz (Constants are just like variables, but they exist only until your program is compiled (assembled). After definition of a constant its value cannot be changed. To define constants EQU directive is used: name EQU < any expression >)

mov ax, faiz ; bu komutu kullandığımızda AX registerine faiz degiskeninin degerini atamış oluyoruz

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

The screenshot displays the Emu8086-Microprocessor Emulator interface with the following windows:

- Registers Window:** Shows the CPU register values at address 0700:0100. Registers include AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES. Values are mostly 00, except for CS (0700), IP (0100), and DS (0700).
- Assembly Window:** Displays the assembly code starting at address 0700:0100. The code includes instructions like MOV AX, 0000h, RET, ADD, ADC, DEC, JNE, NOP, and HLT.
- Original Source Code Window:** Shows the assembly source code with comments explaining variable definitions and memory locations.
- Variables Window:** Lists memory variables and their addresses: SAYI1 (0Ch), SAYI2 (0015h), SAYILAR2 (08h), SAYILAR3 (01h, 02h, 03h, 01h, 02h, 03h), SAYILAR4 (01h, 02h, 03h, 03h, 03h, 04h), SAYILAR5 (00h), SAYILAR6 (0013h, 124Fh, 0021h), and METIN ('o', 'k', 'u', 'l').

```
emu8086 - assembler and emulator: noname.com_
```

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 mov ax, faiz ; bu komutu kullandığımızda AX registerine faiz degisk
07 ret
08
09 sayil1 db 12
10 Sayi2 dw 15h
11
12 sayilar2 db 3 dup(8) ; 3 tane 8 sakla dup(duplicate)
13 sayilar3 db 2 dup(1,2,3) ; 1,2,3,1,2,3 sakla
14 sayilar4 db 1,2,4 dup(3),4 ; 1,2,3,1,2,3,3,4 sakla
15 sayilar5 db 3 dup(?) ; 3 tane ?
16 sayilar6 dw 13h,124Fh,0021h ; sayi dizisi tanimlama
17
18 metin db "okul"
19
20 faiz equ 8 ; faiz diye bir sabit tanimliyoruz
21
22
23
24
25
```

Registers:

	H	L
AX	00	00
BX	00	00
CX	00	24
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Stack:

	screen	source	reset	aux	vars	debug	stack	flags
07100	B8	184						
07101	00	008	BACK					
07102	00	000	NULL					
07103	C3	195	F					
07104	0C	012	S					
07105	15	021	S					
07106	00	000	NULL					
07107	08	008	BACK					
07108	08	008	BACK					
07109	08	008	BACK					
0710A	01	001	S					
0710B	02	002	S					
0710C	03	003	V					
0710D	01	001	S					
0710E	02	002	S					
0710F	03	003	V					
07110	01	001	S					
07111	02	002	S					
07112	03	003	V					
07113	03	003	V					
07114	03	003	V					
07115	03	003	V					
07116	04	004	D					
07117	06	000	NULL					
07118	08	000	NULL					
07119	00	000	NULL					
0711A	13	019	!!					
0711B	00	000	NULL					
0711C	4F	079	O					
0711D	12	018	T					
0711E	21	033	!					
0711F	00	000	NULL					
07120	6F	111	O					
07121	6B	107	K					
07122	75	117	U					
07123	6C	108	L					
07124	98	144	E					
07125	98	144	E					
07126	98	144	E					
07127	98	144	E					
07128	98	144	E					
07129	98	144	E					
0712A	98	144	E					
0712B	98	144	E					
0712C	98	144	E					
0712D	98	144	E					
0712E	98	144	E					
0712F	98	144	E					
07130	98	144	E					

Variables:

size:	word	elements:	1
edit	hex		
SAYI1	0Ch		
SAYI2	0015h		
SAYILAR2	08h		
SAYILAR3	01h, 02h, 03h, 01h, 02h, 03h		
SAYILAR4	01h, 02h, 03h, 03h, 03h, 04h		
SAYILAR5	00h		
SAYILAR6	0013h, 124Fh, 0021h		
METIN	'o', 'k', 'u', 'l'		

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

LEA (Load Effective Address) KOMUTU:

Register, Memory

LEA operand1,operand2: **operand2** de gösterilen bellek hücresinin adresini **operand1** e aktarır.

- Başlangıç adres tutucuları (**BX,BP**) ve indeks (**SI,DI**) registerleri kullanılmalıdır.
- **LEA: MOV** operand1, **offset** operand2 ifadesi ile aynı işlemi yapar (buradaki offset anahtar kelimesi ile direkt operand2 nin efektif adresini alabilirsiniz)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
; MOV ve LEA kullanımı
```

```
LEA BX, sayı ; sayı değişkeninin tutuldugu bellek adresi BX registerine  
atandi
```

```
MOV AX, [BX] ; BX registerinin gösterdiği bellek hücresindeki değeri AX  
registerine atadık
```

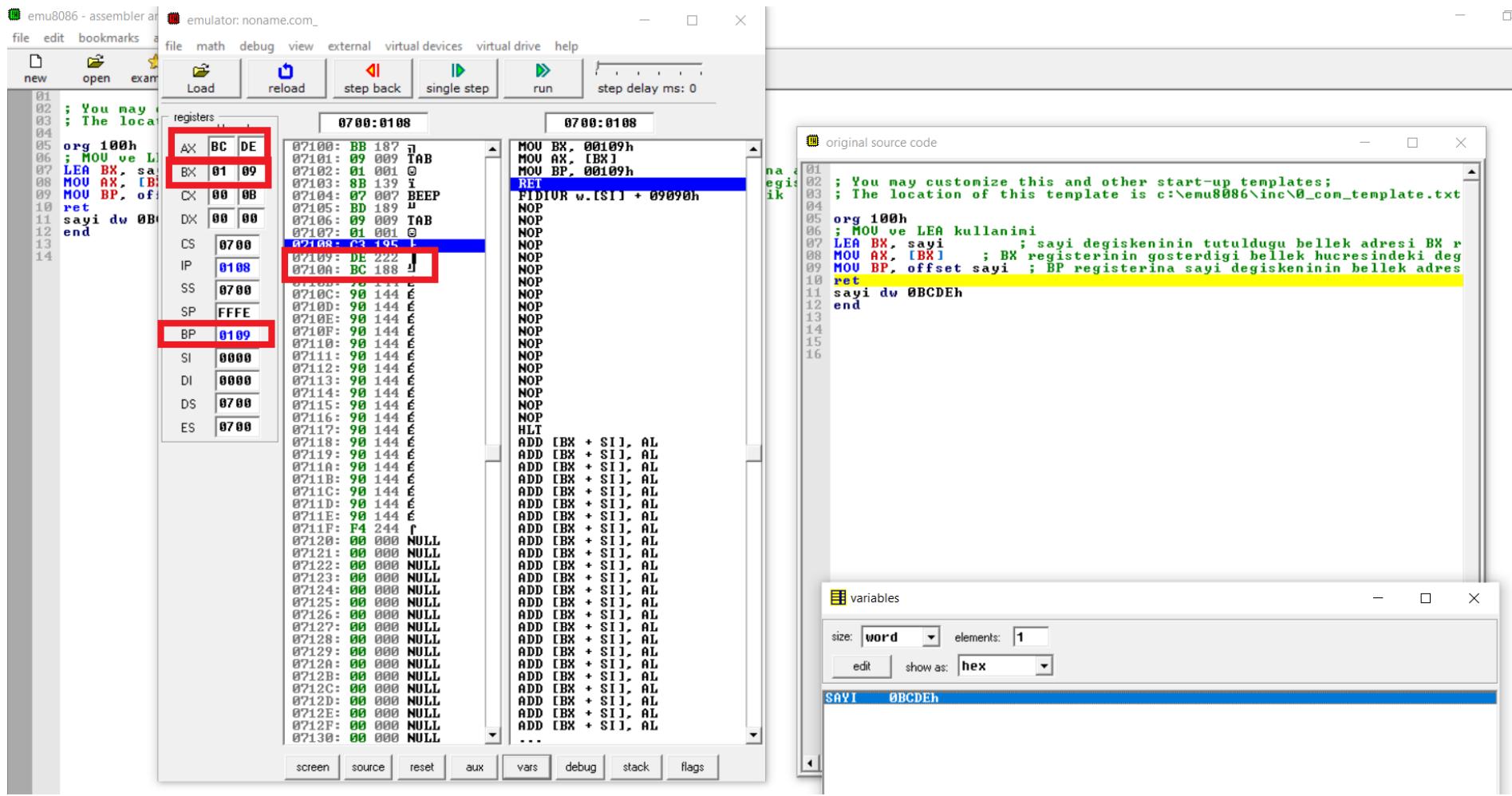
```
MOV BP, offset sayı ; BP registerine sayı değişkeninin bellek adresini atadık  
ret
```

```
sayı dw 0BCDEh
```

```
end
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h  
; MOV ve LEA kullanımı
```

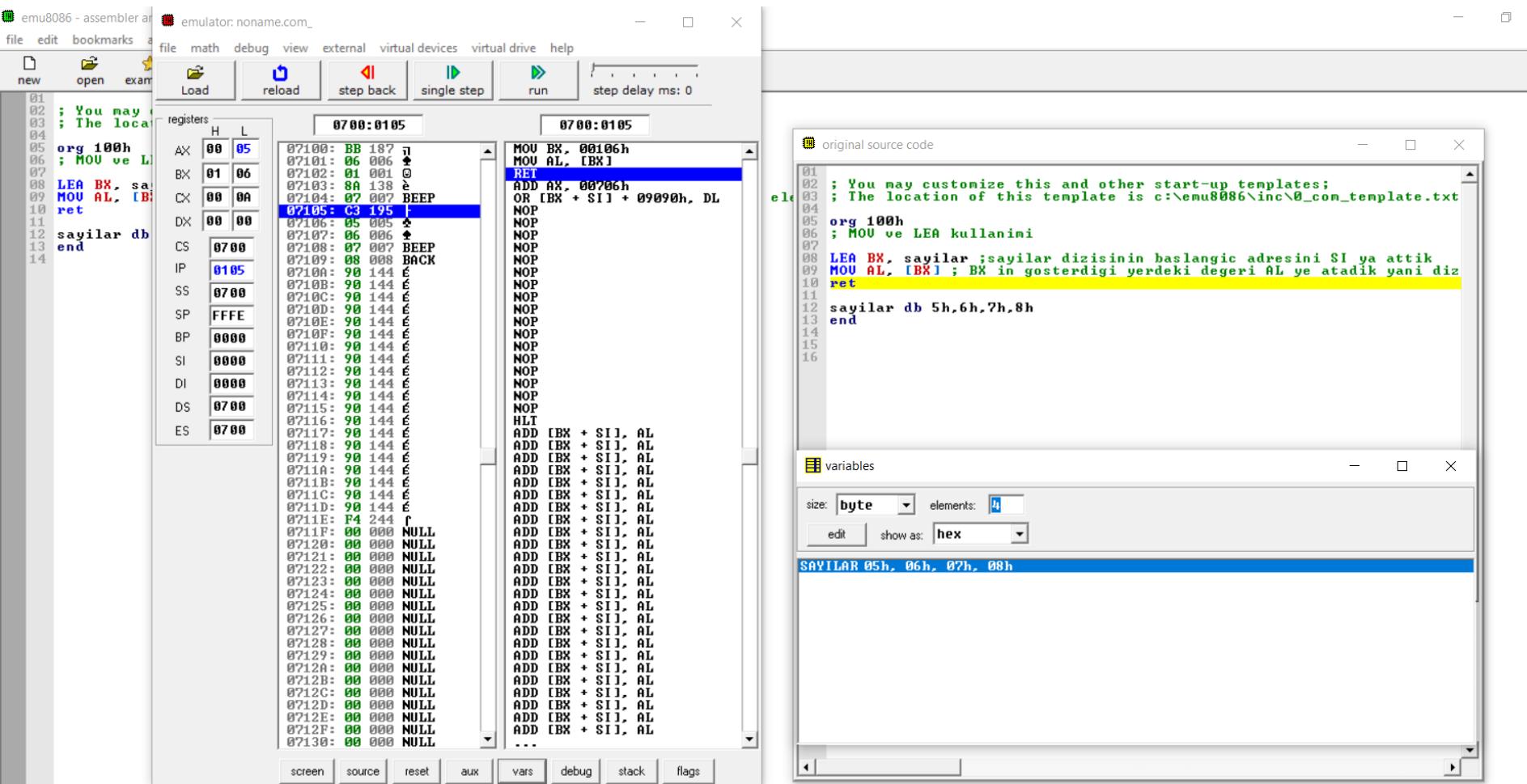
LEA BX, sayilar ;sayilar dizisinin baslangic adresini BX e attik
MOV AL, [BX] ; BX in gösterdiği yerdeki değeri AL ye atadık yani dizinin ilk elemanı

```
ret
```

```
sayilar db 5h,6h,7h,8h  
end
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

;dizide bir sonraki adrese ulaşma:

MOV AL, [BX+1] ; BX in gosterdigi yeri 1 arttır
ordaki degeri AL ye atadik (byte tipinde olduğu
için +1)

MOV AL, [BX+2] ; word tipinde olduğu için +2
yapmak gereklidir çünkü word tipindeki her
değişken bellekte 2 byte yer kaplıyor

NOT: DİZİNİN İNDİS DEĞERİ 0 DAN BAŞLAR!!

8086 16-Bit Mikroişlemci

org 100h

MOV BX,0000h

MOV CX, sayı[BX]

MOV CX, sayı[BX+2]

ret

sayı dw 1234H, 5678H

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

XCHG (exchange) KOMUTU:

operand1, operand2

XCHG operand1,operand2: **operand1 ve operand2 değerleri birbirleriyle yer değiştirir**

- Tek seferde iki değeri yer değiştirmek için kullanılır

REG, memory

Memory, REG

REG,REG

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h  
; XCHG kullanimi
```

```
MOV BL,2h  
MOV BH,3h  
; BL ile BH i yer degistirmek istersek
```

```
MOV AL,BL  
MOV AH,BH  
MOV BL,AH  
MOV BH,AL
```

```
ret
```

```
sayilar db 5h,6h,7h,8h  
end
```

```
org 100h  
; XCHG kullanimi  
MOV BL,2h  
MOV BH,3h  
; BL ile BH i yer degistirmek istersek
```

```
XCHG BL,BH  
ret
```

```
sayilar db 5h,6h,7h,8h  
end
```



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

The screenshot shows the emu8086 emulator interface with three main windows:

- Left Window (Registers):** Displays the CPU registers (AX, BX, CX, DX, SP, BP, SI, DI, DS, ES) in both H (High) and L (Low) bytes. The BX register is highlighted with a red box.
- Middle Window (Memory Dump):** Shows memory starting at address 0700:010C. The BX register's value (02 03) is present at address 0700:010C. The entire assembly code listing is visible.
- Right Window (Original Source Code):** Displays the assembly source code with syntax highlighting. The BX register's value (02 03) is also present in the original source code at line 17.

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ; XCHG kullanimi
07 ; MOU BL,2h
08 MOU BH,3h
09
10 ; BL ile BH i yer degistirmek istersek
11
12 ; BL ile BL
13 MOU AL,BL
14 MOU AH,BH
15 MOU BL,AH
16 MOU BH,AL
17 MOU BH,AL
18
19 ret
20
21 sayilar db 5h,6h,7h,8h
22 end
23
24
25
```

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ; XCHG kullanimi
07 ; MOU BL,2h
08 MOU BH,3h
09
10 ; BL ile BH i yer degistirmek istersek
11
12 ; BL ile BL
13 MOU AL,BL
14 MOU AH,BH
15 MOU BL,AH
16 MOU BH,AL
17 MOU BH,AL
18
19 ret
20
21 sayilar db 5h,6h,7h,8h
22 end
23
24
25
26
27
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

The screenshot shows the EMU 8086-Microprocessor Emulator interface. On the left, the assembly code window displays the following source code:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05 ; XCHG kullanimi
06
07 MOU BL,2h
08 MOU BH,3h
09 ; BL ile BH
10 ; BL ile BH i yer degistirmek istersek
11 XCHG BL,BH
12 ret
13
14 sayilar db
15 end
16
17
18
```

The assembly code window also shows the memory dump for address 0700:0106, which contains the instruction `XCHG BL, BH`. The registers window shows the following values:

	H	L
AX	00	00
BX	02	03
CX	00	00
DX	00	00
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The status bar at the bottom of the assembly code window includes buttons for screen, source, reset, aux, vars, debug, stack, and flags.

On the right, the original source code window shows the same assembly code. The instruction `XCHG BL, BH` is highlighted in red, indicating it is the current instruction being executed.

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **INC KOMUTU:**

REG

MEMORY

; değeri 1 artırma komutudur

MOV AH,5

INC AH ; AH=6 olur

- **DEC KOMUTU:**

REG

MEMORY

; değeri 1 azaltma komutudur

MOV AH,5

DEC AH ; AH=4 olur

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **XLATB KOMUTU:**
operand almaz
 - Dizinin **başlangıç adresini BX registeri içine atamak** zorunludur.
 - Erişmek istediğimiz dizinin elemanın konumu **AL registerine yüklemeniz** zorunludur
 - **XLATB komutu** yazılır sonra istenen eleman **AL registeri içine kaydedilmiş** olur.

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

ORG 100h

LEA BX, sayilar ; sayilar dizisinin baslangic adresini
BX registerina atadik

MOV AL, 2 ; kacinci indise ulasmak istiyorsak onu
AL register icine atiyoruz dizi indisleri 0 dan baslar

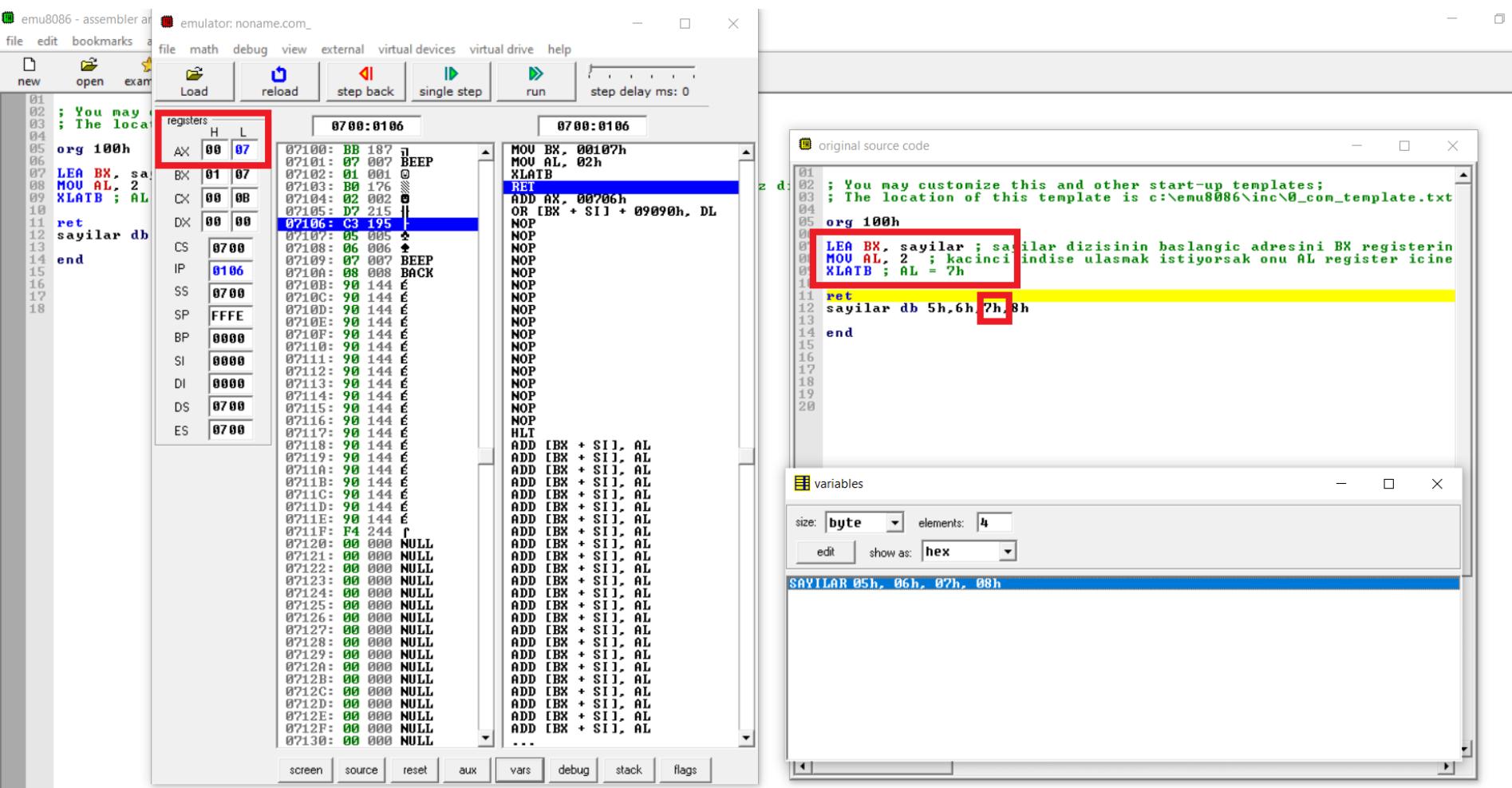
XLATB ; AL = 7h

RET

sayilar db 5h,6h,7h,8h

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



Kaynaklar

- <https://www.electricaltechnology.org/2020/05/types-of-microprocessors.html>
- <http://www.lojikprob.com/elektronik/mikroislemci-mimarisi-2-adres-veri-ve-kontrol-yollari/>
- https://www.academia.edu/11730974/KTU_M%C4%B0KRO%C4%B0%C5%9ELEM%C4%B0LER_DE_RS_NOTLARI
- [https://tr.wikipedia.org/wiki/Intel_8086#:~:text=8086%20\(ayr%C4%B1ca%20iAPX86%20de%20denir,aylar%C4%B1nda%20ilk%20%C3%A7ip%20piyasaya%20s%C3%BCr%C3%BCm%C3%BCn%C3%BCf%C3%BCt%C3%BCr.](https://tr.wikipedia.org/wiki/Intel_8086#:~:text=8086%20(ayr%C4%B1ca%20iAPX86%20de%20denir,aylar%C4%B1nda%20ilk%20%C3%A7ip%20piyasaya%20s%C3%BCr%C3%BCm%C3%BCn%C3%BCf%C3%BCt%C3%BCr.)
- https://www.tutorialspoint.com/microprocessor/microprocessor_8086_overview.htm
- <http://rakeshharsha.blogspot.com/2017/05/the-8086-microprocessor-architecture.html>
- <https://tr.wikipedia.org/wiki/X86>
- <https://electronicsdesk.com/8086-microprocessor.html>
- <https://www.intel.com.tr/content/www/tr/tr/gaming/resources/cpu-clock-speed.html>
- https://tr.wikipedia.org/wiki/%C4%B0%C5%9Flemci_%C3%B6nbelle%C4%9F
- <https://www.karel.com.tr/blog/ram-ve-onbellek-arasindaki-fark-nedir>
- <http://www.scozturk.com/wp-content/uploads/permanent/scozturk.com%20-%20Intel%208086%20ile%20Mikroi%C5%9Flemci%20Programlamaya%20Giri%C5%9F%20-%20Kitap.pdf>