**Von Neumann Mimarisinin Bileşenleri**

1 – Bellek
2 – Merkezi İşlem Birimi
3 – Giriş/Çıkış Birimleri

Yazmaçlar
İletişim Yolları

Bileşenler arasındaki iletişim ise **iletişim yolları** adı verilen kanallar yardımı ile gerçekleşir:

1 – Veri Yolları
2 – Adres Yolları
3 – Kontrol Yolları

# İLETİŞİM YOLLARI

Bileşenler arasındaki iletişim ise **iletişim yolları** adı verilen kanallar yardımı ile gerçekleşir:

1 – Veri Yolları
2 – Adres Yolları
3 – Kontrol Yolları

İletişim yollarında veriler taşınır ve veriler 0 – 1 şeklindedir.

İletişim hızı Hertz olarak birimleştirilmiştir. Hertz, saniye başına düşen titreşim sayısının birimidir ve bilgi işlem sürecinde ise bir titreşim anında kaç komut işlenecek demektir.

Veri iletişim hızı (Hertz) ile veri aktarım hızı (bit/s) karıştırılmamalıdır.

İletişim yolları anakart üzerinde genellikle bakır elementinden oluşan kanallar kastedilmektedir.

İlk mikro işlemcilerde bu kanallar 8 bitlik idi. Daha sonra 16 bitlik iletişim yolları üretildi.  Günümüzde ise 32 bitlik iletişim yolları kullanılmaktadır.

## 1 – Veri Yolu (Data Bus)

Bu veri yolu, işlemciden belleğe veya G/Ç birimlerine veri göndermede ya da bu birimlerden işlemciye veri taşınımında kullanılır. Bu nedenle iki yönlüdür.

Veri yolları genel olarak yazmaçlar ile aynı büyüklüktedir.

İçsel Veri Yolları: Merkezi işlem biriminin bileşenleri (AMB, KB, yazmaçlar) arasındaki iletişimde kullanılır.

Dışsal Veri Yolları: Merkezi işlem birimi ile bellek ya da G/Ç birimleri arasındaki veri iletişiminde kullanılır.

## 2 – Adres Yolu (Address Bus)

İşlenmiş bilginin saklanacağı konumu taşıyan kanallara Adres Yolu adı verilir. Tek yönlüdür. Program sayacına bağlıdır.

## 3 – Kontrol Yolu (Control Bus)

Birimlerin düzenli çalışmasını sağlayan özel iletileri taşıyan kanallardır. Bu iletiler sırasıyla; kesme isteği (işlemciye müdahale yönündedir), yön isteği (bellekten işlemciye ya da işlemciden belleğe yön istemi), zamanlama sinyalleri.

# YAZMAÇLAR

Yazmaçlar: Bilgilerin ya da verilerin çok kısa süre için tutuldukları yerlerdir.

Ana bellekten farkı ise,
* çok kısa süreli bilgi saklamaları,
* belleğe göre çok daha hızlı çalışırlar,
* bir birime bağlı olarak çalışmaları (AMB yazmaçları, kontrol birimi yazmaçları, bellek yazmaçları gibi)
* adreslemelerinin olmaması.

Yazmaçlar, AMB yazmaçları, kontrol birimi yazmaçları ve bellek yazmaçları ibi sınıflanabileceği gibi, Segment yazmaçları ve Genel Amaçlı Yazmaçlar olarakta sınıflanmaktadır.

Bu konuya ileride geri dönülecektir.

# BELLEK

Operatör ve operandların geçici süreler olarak saklandığı yerlere rasgele erişimli bellek (Random Access Memory) adı verilir. Butür belleklerden hem okuma hem de yazma işlemi yapılabilir.

Yalnızca okunabilen bellek türleri (ROM) ise kartların üretici firmaları tarafından programlanırlar.

Bunların yanısıra yazmaçlar, tampon bellek gibi küçük bellek türleri de vardır.

Von Neumann mimarisinde söz konusu olan genellikle RAM bellektir.

**Bellek**, işletilecek olan komutu ve operandı içeren ya da saklayan bilgisayar birimidir. Bir byte'lık hücrelerden oluşan bir boyutlu dizi görünümündedir. RAM (Random Access Memory)'de, hücre adreslerinden dolayı doğrudan bellek hücrelerine erişmek olanaklıdır.

Bellek adresleri tamsayılardan oluşmaktadır. Ancak bu durumda şu şekilde bir sorun ortaya çıkmaktadır.

Tam sayılar 2 byte'lık bir veri uzunluğuna sahiptirler. Bir diğer ifade ile maksimum 65535 değerini alabilir. Bu nedenle bellekte 65536 adet byte adreslenebilir.

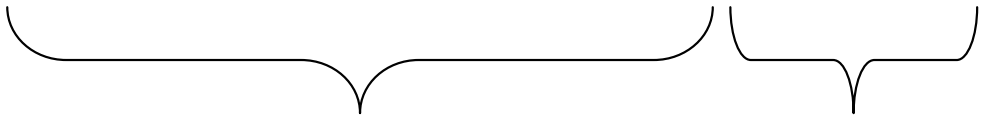
$2^{10}$=1024 B → 1 KB
$2^{20}$=1 048 576 B → 1MB
$2^{30}$=1 073 741 824 B → 1GB

Bu durumda nasıl $2^{20}$=1 MB'lık bir bellek nasıl adreslenecek?

Bir diğer sorun, veri (veri yolları) ve adres (adres yolları) transferleri maksimum 16 bitlik verilerle olanaklıdır. Bu durumda 20 bitlik bellek adresi nasıl taşınacak?

| Desimal | Binary | Heksadesimal |
|---|---|---|
| 0 | 0000 0000 0000 0000 0000 | 00000 |
| 1 | 0000 0000 0000 0000 0001 | 00001 |
| 2 | 0000 0000 0000 0000 0000 | 00002 |
| 3 | 0000 0000 0000 0000 0000 | 00003 |
| | : : : : | |
| 10 | 0000 0000 0000 0000 1010 | 0000A |
| 11 | 0000 0000 0000 0000 1011 | 0000B |
| 12 | 0000 0000 0000 0000 1100 | 0000C |
| 13 | 0000 0000 0000 0000 1101 | 0000D |
| 14 | 0000 0000 0000 0000 1110 | 0000E |
| 15 | 0000 0000 0000 0000 1111 | 0000F |
| 16 | 0000 0000 0000 0001 0000 | 00010 |

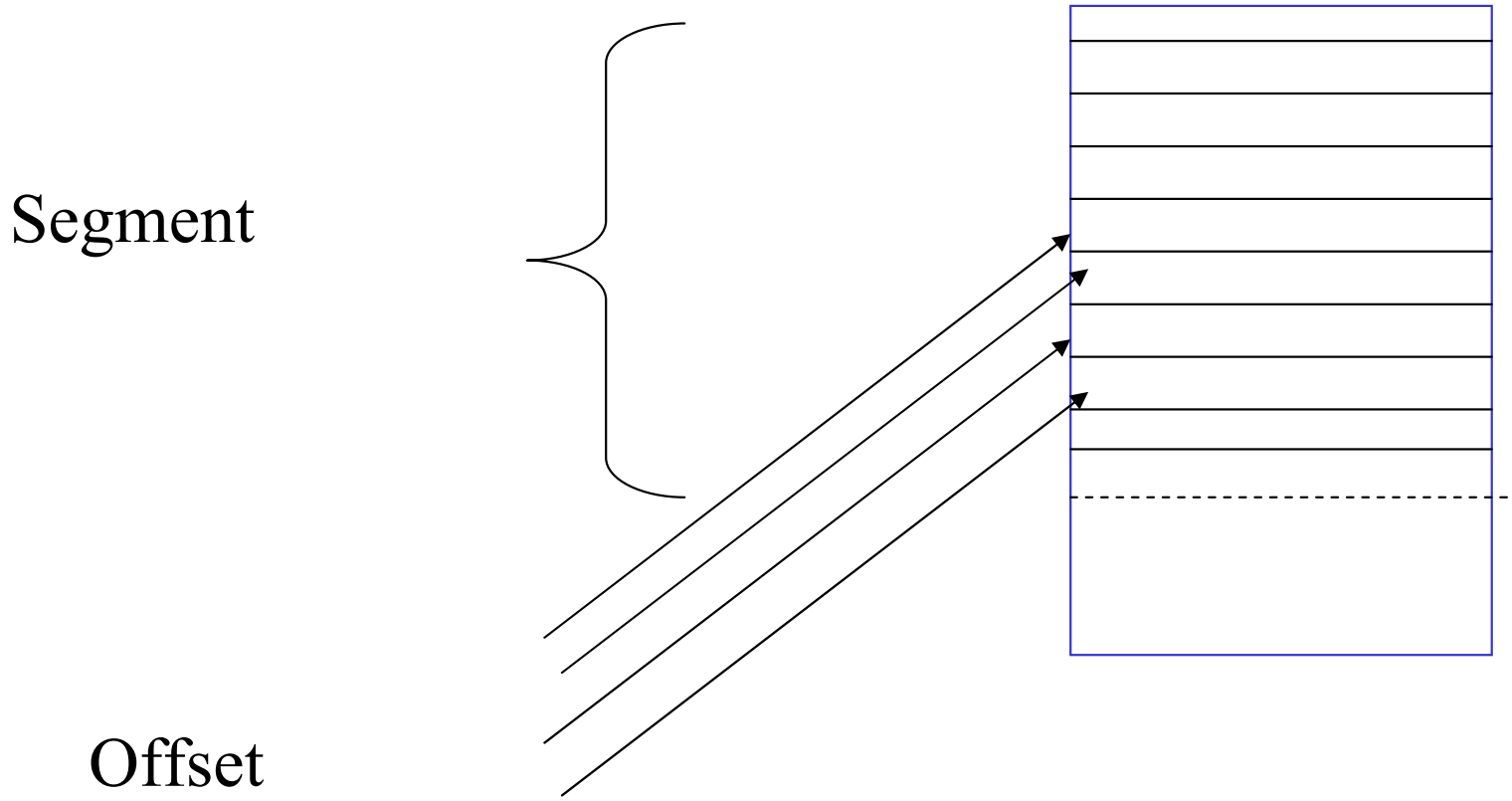
| Desimal | Binary | Heksadesimal |
|---|---|---|
| 0 | 0000 0000 0000 0000 0000 | 00000 |
| 16 | 0000 0000 0000 0001 0000 | 00010 |
| 32 | 0000 0000 0000 0010 0000 | 00020 |
| 48 | 0000 0000 0000 0011 0000 | 00030 |
| 64 | 0000 0000 0000 0100 0000 | 00040 |
| 80 | 0000 0000 0000 0101 0000 | 00050 |
| 96 | 0000 0000 0000 0110 0000 | 00060 |



16 Bit

4 Bit

Segment ile offset yaklaşımı ile bütün bellek adreslenebilir. Bunun için bellek 65536 byte büyüklüğünde bloklara ayrılır, çünkü elimizde 65536 adet adres var. Her bir blok'a Segment adı verilir.




Segmentler kendi içerisinde 16 bitlik işaretsiz tam sayılar ile yeniden adreslenebilir. Bu durumda segment içinde yeralan bir bellek bölgesinin adresini gösteren 16 bitlik işaretsiz tamsayıya Offset denilir.

Diğer bir ifade ile bellek bölgelerine segment, belirli bir bellek bölgesi içerisindeki bellek hücresini gösteren adrese ise offset adı verilir.

Kullanım kolaylığı nedeniyle bellek adresleri onaltılık sayı sistemi (heksadesimal) ile gösterilir.

Segment ve offset yapısı ile parçalanmış bellek adresleri

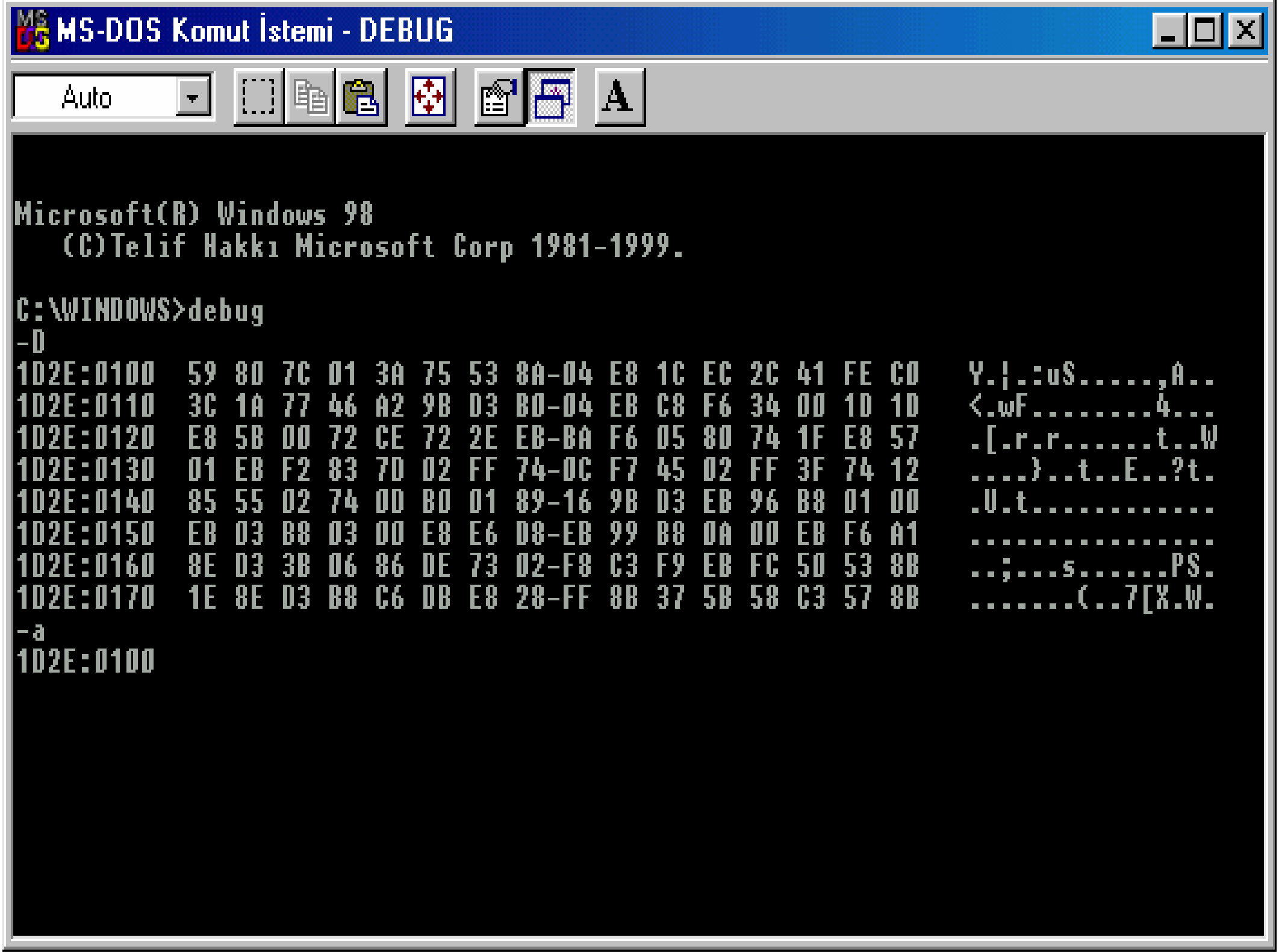
Segment : Offset

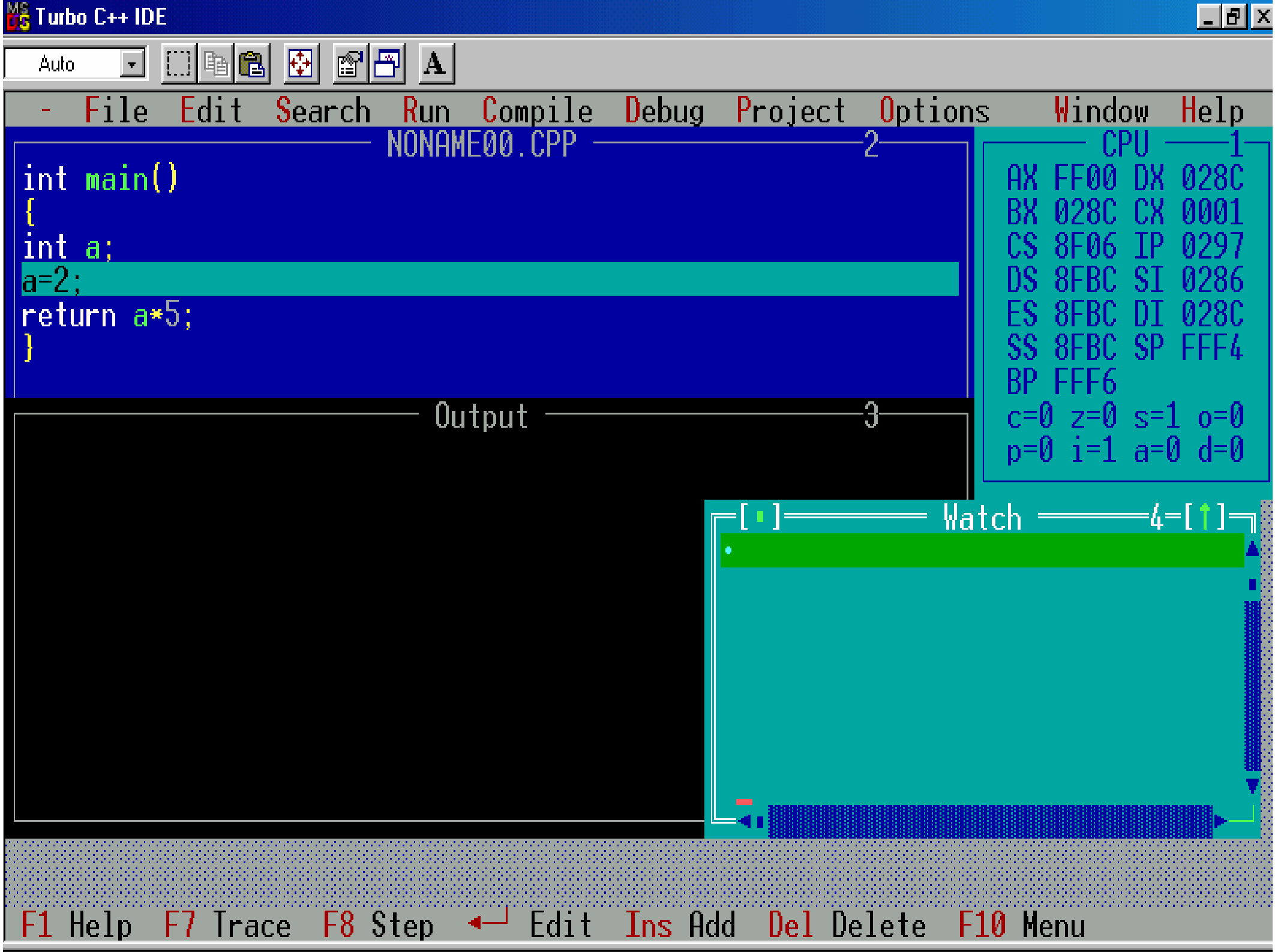
Biçiminde gösterilir. 1D2E:0120

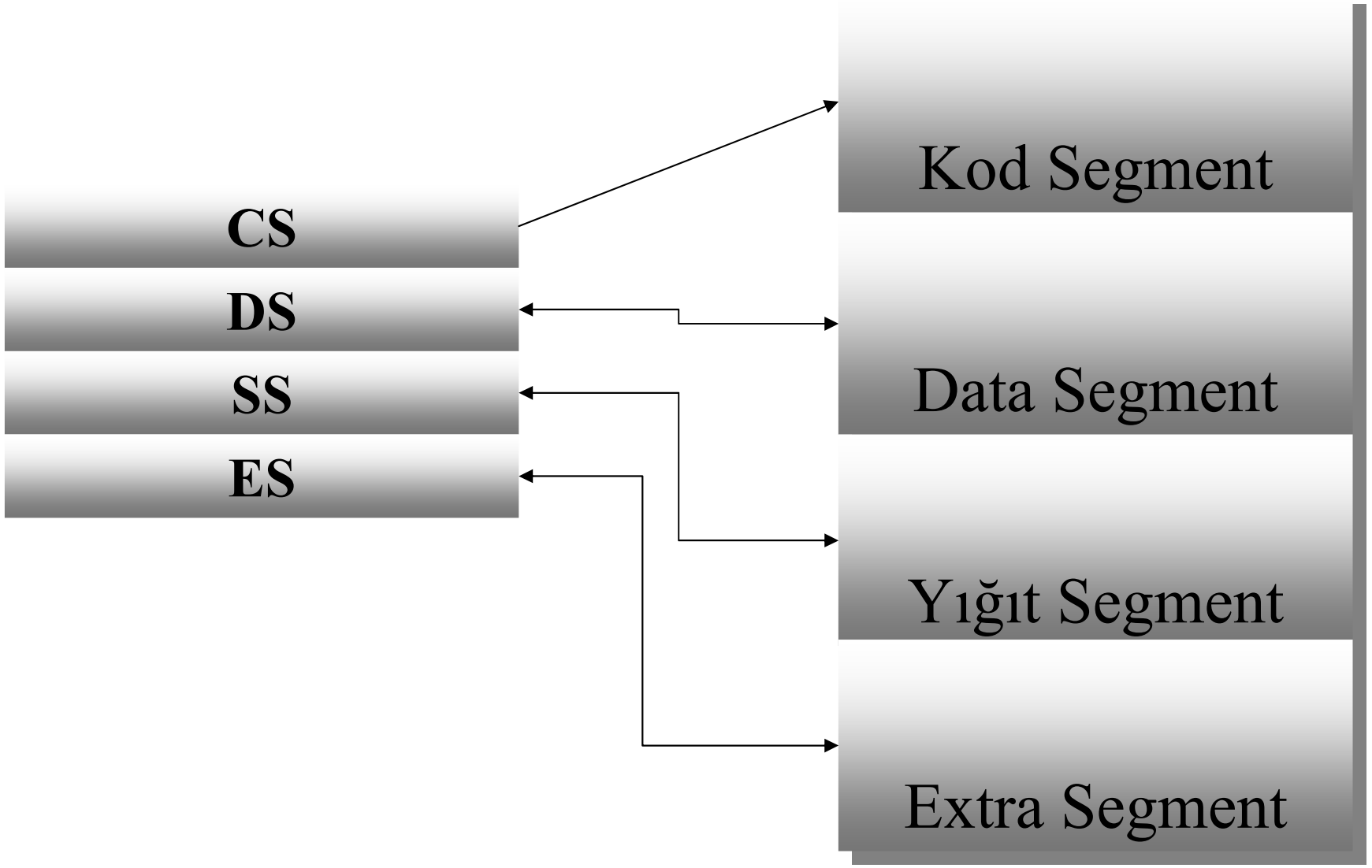
Eğer bellekteki bir operand için adresleme: DS:BX
Eğer bellekteki bir operatör için adresleme: CS:IP

Şeklinde yapılır.

Debug komutu ile olası bellek adreslerine bakalım:


MS-DOS Komut İstemi - DEBUG
Auto
Microsoft(R) Windows 98
(C)Telif Hakkı Microsoft Corp 1981-1999.
C:\WINDOWS>debug
-D
1D2E:0100 59 80 7C 01 3A 75 53 8A-04 E8 1C EC 2C 41 FE C0 Y.¦.:uS.....,A..
1D2E:0110 3C 1A 77 46 A2 9B D3 B0-04 EB C8 F6 34 00 1D 1D <.wF........4...
1D2E:0120 E8 5B 00 72 CE 72 2E EB-BA F6 05 80 74 1F E8 57 .[.r.r......t..W
1D2E:0130 01 EB F2 83 7D 02 FF 74-0C F7 45 02 FF 3F 74 12 ....}..t..E..?t.
1D2E:0140 85 55 02 74 0D B0 01 89-16 9B D3 EB 96 B8 01 00 .U.t............
1D2E:0150 EB 03 B8 03 00 E8 E6 D8-EB 99 B8 0A 00 EB F6 A1 ................
1D2E:0160 8E D3 3B 06 86 DE 73 02-F8 C3 F9 EB FC 50 53 8B ..;...s......PS.
1D2E:0170 1E 8E D3 B8 C6 DB E8 28-FF 8B 37 5B 58 C3 57 8B .......(..7[X.W.
-a
1D2E:0100


Turbo C++ IDE
Auto
File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP 2
int main()
{
int a;
a=2;
return a*5;
}
CPU 1
AX FF00 DX 028C
BX 028C CX 0001
CS 8F06 IP 0297
DS 8FBC SI 0286
ES 8FBC DI 028C
SS 8FBC SP FFF4
BP FFF6
c=0 z=0 s=1 o=0
p=0 i=1 a=0 d=0
Output 3
Watch 4
F1 Help F7 Trace F8 Step Edit Ins Add Del Delete F10 Menu




Bellek Alanları ve Segment Yazmaçları

Belleğin, operatörleri (komutları) sakladığı bölümüne kod segment (Code Segment) adı verilir ve belleğin bu bölgesinin başlangıç adresi CS yazmacında tutulur. -CS:IP-

Aynı şekilde, operandların saklandığı bölüme ise veri segment (Data Segment) adı verilir. Bu bellek bölgesinin başlangıç adresi DS yazmacında tutulur. -DS:BX-, -DS:SI-

Yığıt Segmentleri (Stack Segment), verilerin geçici olarak bellekte tutulduğu bölgeyi ifade eder. SS yazmacı bu bölgenin başlangıç adresini belirtir. -SS:SP-, -SS:BP-

Değişik amaçlarla kullanılan (örneğin string işlemleri) bellek bölgesine ise Extra Segment adı verilir ve adresi ES yazmacında tutulur. -ES:DI-


AX
Akümülatör
BX
Taban Adres İndisçisi
CX
Sayaç
DX
Veri
SI
Kaynak İndisçisi
DI
Hedef İndisçisi
SP
Yığıt İşaretçisi
BP
Yığıt Taban İşaretçisi
IP
Komut İşaretçisi
FLAGS
Bayraklar

AX: AH-AL EAX

**AX:** Akümülatör, genellikle çarpma, bölme, giriş ve çıkış işlemlerinde kullanılır. En temel yazmaçtır.

**BX:** Base Address Register; bellek içindeki verilerin adreslenmesinde kullanılır.

**CX:** Counter Register; Daha çok dizinlerdeki indisi ve de döngülerdeki sayacı tutar.

**DX:** Data Register; Genel amacı AX'e yardımcı olmaktır. Büyük hacimli sayılarda, ya da AX'in kullandığı G/Ç işlemlerindeki port adresini tutar.

İşaretçi ve İndis Yazmaçları, genellikle offsetleri gösterir.

**IP:** Instruction Pointer; Bellekte yer alan komut, kod segmenti içerisinde herhangi bir offset adresindedir. Kod segmentini CS, komutun offset adresini ise IP gösterir ve CS:IP şeklinde ifade edilir.

**SP:** Stack Pointer; SS:SP şekline kullanılır ve o esnada bellekteki yığıt segmentine ilişkin offset adresini taşır. Base Pointer (BP) yazmacı da aynı işlemi görür.

**SI ve DI:** Source index, destination index; DS:SI ve ES:DI olarak kullanılırlar.

**Flags:** Bayraklar ile belirli durumları bildiren yazmaçtır. İşlem esnasında ya da işlem sonucunda durum bilgileri taşır.

Bayraklar:

| N | V | +5V | B | D | I | Z | C |
|---|---|---|---|---|---|---|---|
| Negatif | Taşma | | Dur | Desimal | Kesme | Zero | Elde |