



ALGORİTMA VE PROGRAMLAMA

Öğr. Gör. Dr. Umut Engin AYTEN



Dersin İçeriği

- ❖ Temel Kavramlar ve Tanımlar
- ❖ Problem Çözme ve Algoritmalar
- ❖ Sözde Kod ve Akış Diyagramı Uygulamaları(Sıralama, arama, ..)
- ❖ Programlama Dillerine Giriş, Temel Kavramlar
- ❖ MATLAB Programı
- ❖ Temel Komutlar, Koşul ve Döngü Komutları, Vektör İşlemleri
- ❖ Grafik Komutları, Dosya İşlemleri
- ❖ MATLAB Programı ile GUI Hazırlama
- ❖ MATLAB Toolbox'lar ve İçerikleri
- ❖ MATLAB Simulink'i Kullanma
- ❖ MATLAB'te Çeşitli Uygulamalar

1.BÖLÜM

Temel Kavramlar ve Tanımlar



Bilgisayar

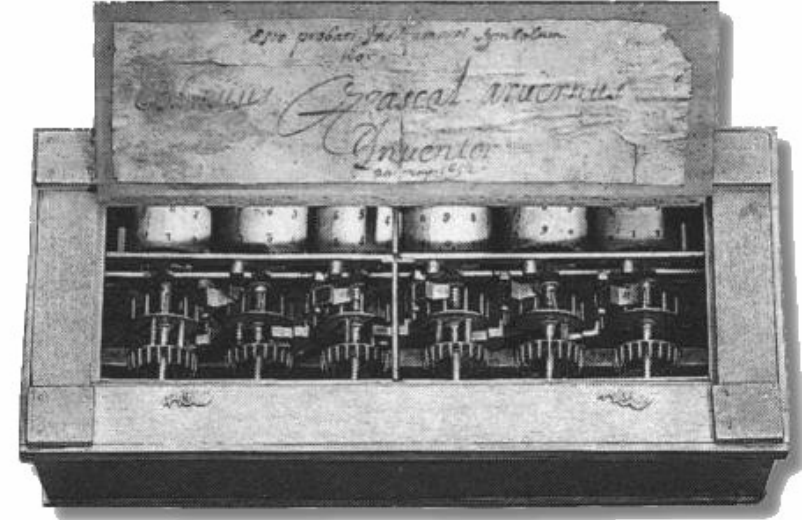
Verilen bilgileri saklayan, gerektiğinde bu bilgileri hızlı bir şekilde istenilen amaca uygun kullanmayı sağlayan/işleyen, mantıksal ve aritmetiksel işlemleri çok hızlı biçimde yapan bir makinedir. Bilgisayar terimi İngilizce “computer” kelimesinin dilimize çevrilmiş halidir. Bu terim de Latince “compurate” kelimesinden gelmektedir.

Bilgisayarların Gelişimi

Mekanik Çağ

Blaise Pascal (1642)

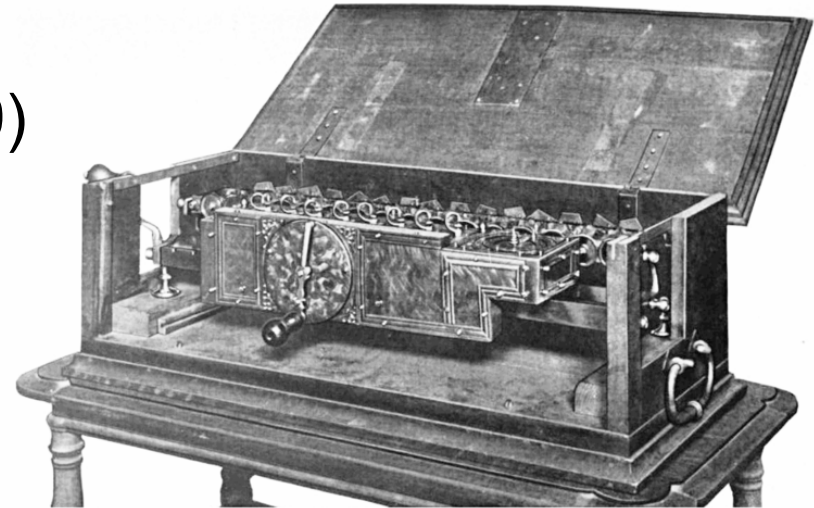
Vites tabanlı toplama makinası



Gottfried Wilhelm von Leibniz (1670)

Toplama, çıkarma, çarpma, bölme

Mekanik olarak sık sık arızalanırdı.



Bilgisayarların Gelişimi



Joseph Jacquard (1810)
Bilgisayar tabanlı halı dokuma makinesi



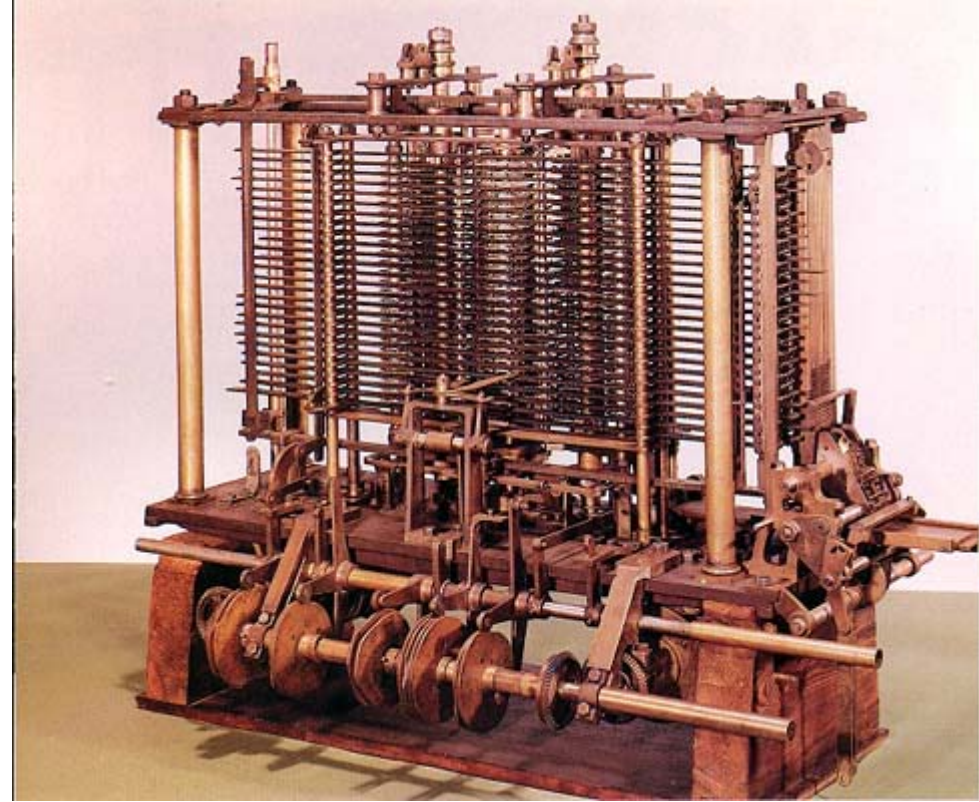
Delikli Kart (Punch Card)

Bilgisayarların Gelişimi



The Difference Engine (1822)

- Charles Babbage



The Analytical Engine

- Punch card'lar üzerinde yazılan programları işleyebiliyordu
- Bilgiyi belleğinde saklayabiliyordu

Bilgisayarların Gelişimi

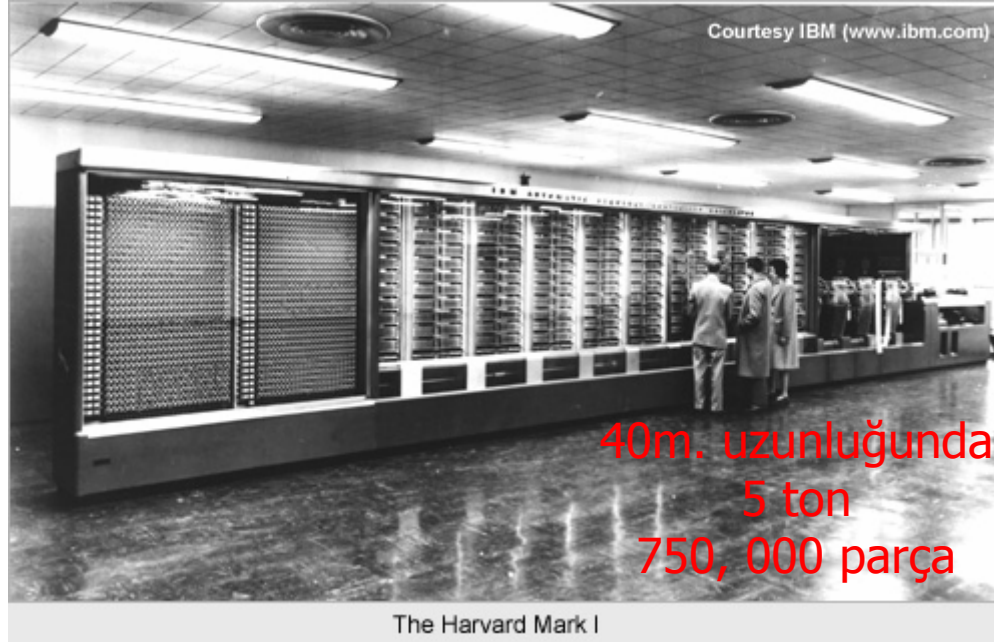
Elektro-mekanik Çağ (1840 – 1940)



- Hermann Hollerith (19'uncu yüzyılın sonları)
- Amerikan oy sayımlarına kullanıldı.
- Elektrik ile çalışıyor.
- Bilgi punch card ile veriliyor.

- Nüfus: 63 milyon; 6 hafta
- International Business Machines (IBM)'in ilk ürünü

Bilgisayarların Gelişimi



Mark I

Bir bilgisayar ile bir hesap makinesi arasında ne fark var?

- Howard Aiken + IBM + Harvard (1930)
- Veri depolama: Mekanik röle telefon anahtarları (switch)
- Girdi: Punch Card

Bilgisayarların Gelişimi

İlk yazılım Bug'ı

9/9

0800 Antan started
1000 " stopped - antan ✓
1300 (033) MP-MC ~~1.13047645~~ 1.2700 9.037 847 025
(033) PRO 2 2.13047645 9.037 846 995 count
count 2.13067645 4.615925059(-2)
Relays 6-2 in 033 failed special speed test
in relay .. 11.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antan started.
1700 closed down.

- Grace Hopper (1909 – 1992)
- Mark I'in ilk programcılarında.
- Derleyicinin mucidi.

Bilgisayarların Gelişim

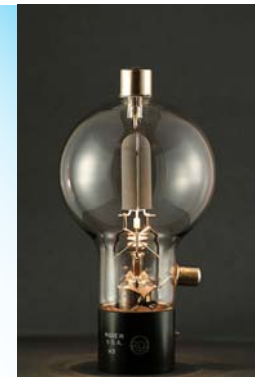
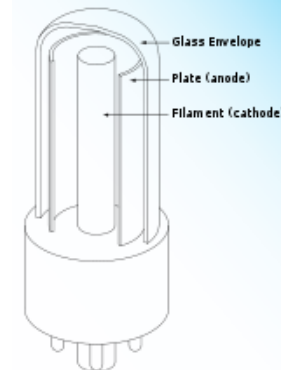
Elektronik Çağ (1840 – Bugün)

❖ Elektronik ile ilgili ilk deneylerin vakum tüplerinde yapılan çalışmalar olduğu kabul edilir. Heinrich Geissler (1814-1879), cam tüpün içinden havanın çoğunu çıkartmış ve bu tüpün içinden elektrik akımı geçirildiğinde tüpün parıldadığını görmüştür.

❖ Sir William Crookes (1832-1919) havası alınmış cam tüp'ün (vakum tüp) içinden akım geçirdiğinde, geçen akımın parçacıklardan oluştuğunu görmüştür.

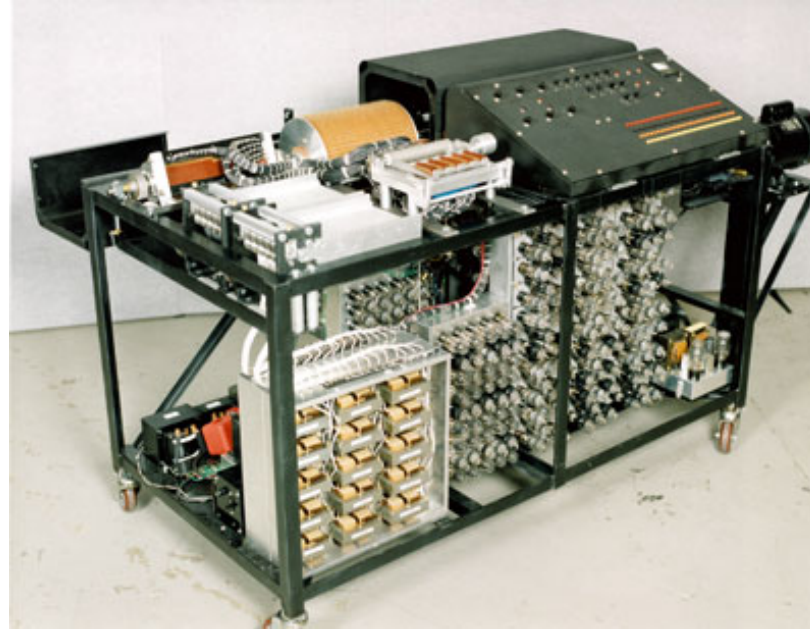
❖ Sir Joseph Thompson (1856-1940) bu parçacıkları ölçmeyi başarmıştır ve bu parçacıklara daha sonra **elektron** denilmiştir.

❖ John Ambrose Fleming, 1904 yılında, vakum tüpünü kullanarak akımın tek yönlü olarak akmasına izin veren vakum tüp diode'u geliştirmiştir. Bu cihaza "Fleming valve" veya radio tube'de denir.



Bilgisayarların Gelişimi

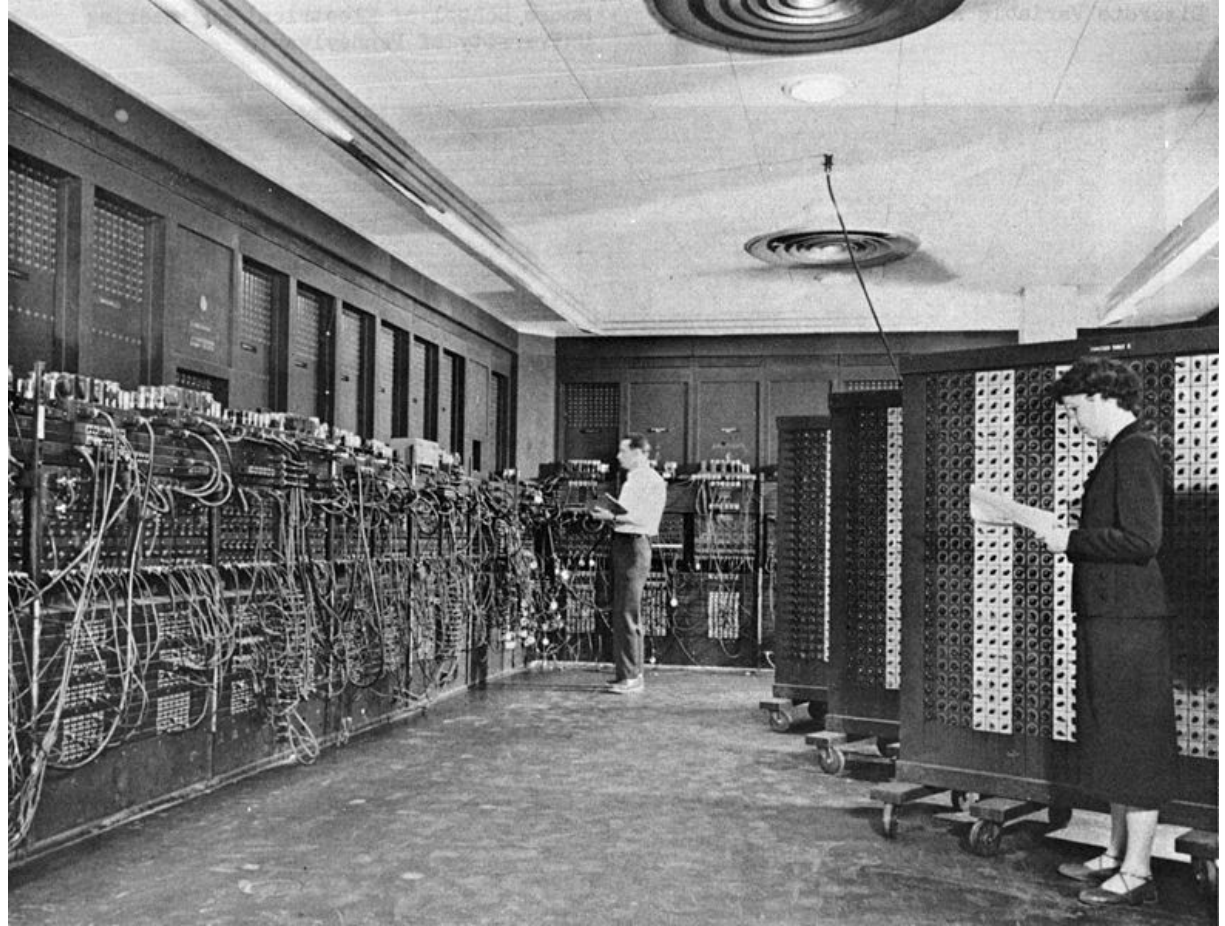
- ❖ 1930 yıllarında, elektronik dünyasında bir çok gelişme olmuştur. Bu yıllarda ilk elektronik hesap makineleri geliştirilmeye başlanmıştır.
- ❖ John Atanasoff ve lisansüstü öğrencisi Clifford Berry, 1939 yılında, ABC (Atanasoff-Berry Computer) olarak adlandırılan ilk ikili sayı sisteminde çalışan makineyi icat geliştirmişlerdir. Bu makinada lojik işlemler için vakum tüpleri ve hafıza için kondansatörler kullanılmıştır.



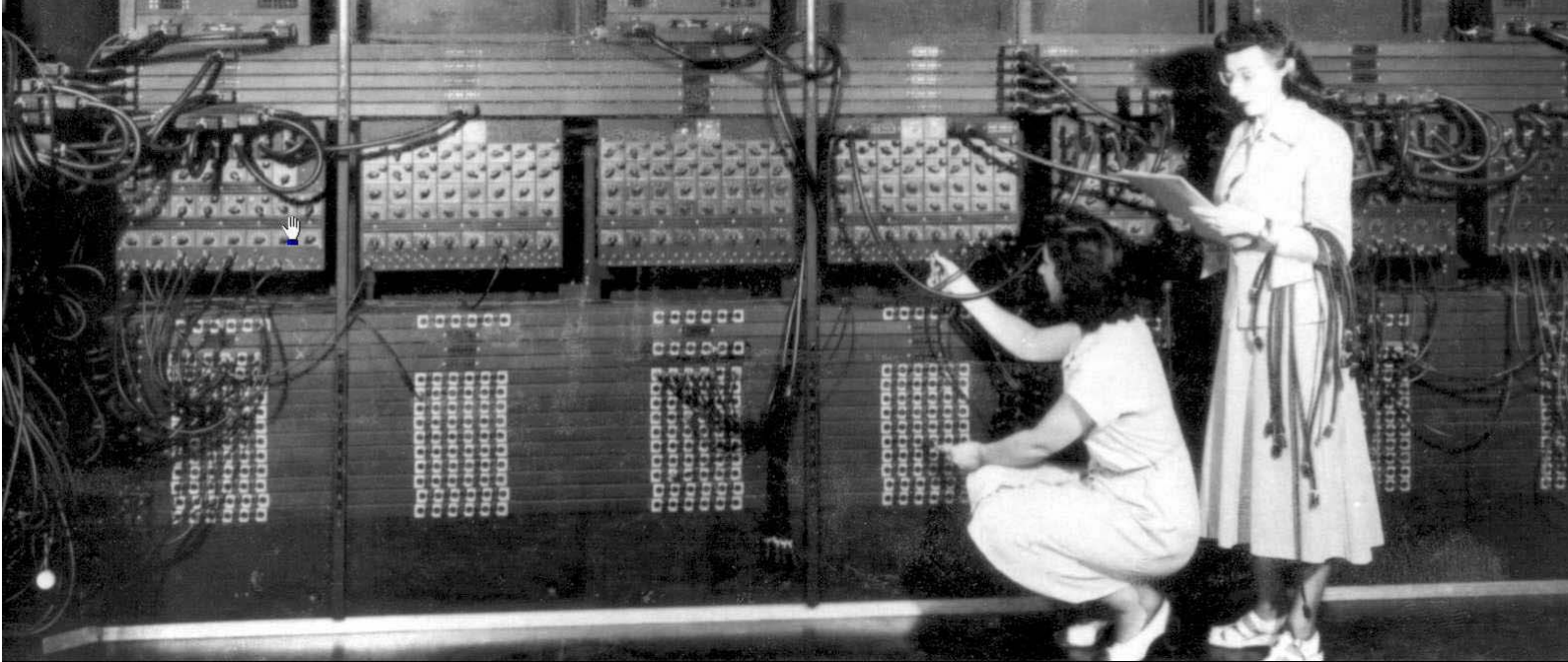
Bilgisayarların Gelişimi

Savaş sırasında bilgisayar konusundaki çalışmalar çok daha hızlı bir şekilde geliştirilmiştir.

John von Neumann, 1946 yılında, ilk bilgisayar olarak kabul edilen Eniac'ı geliştirmiştir.



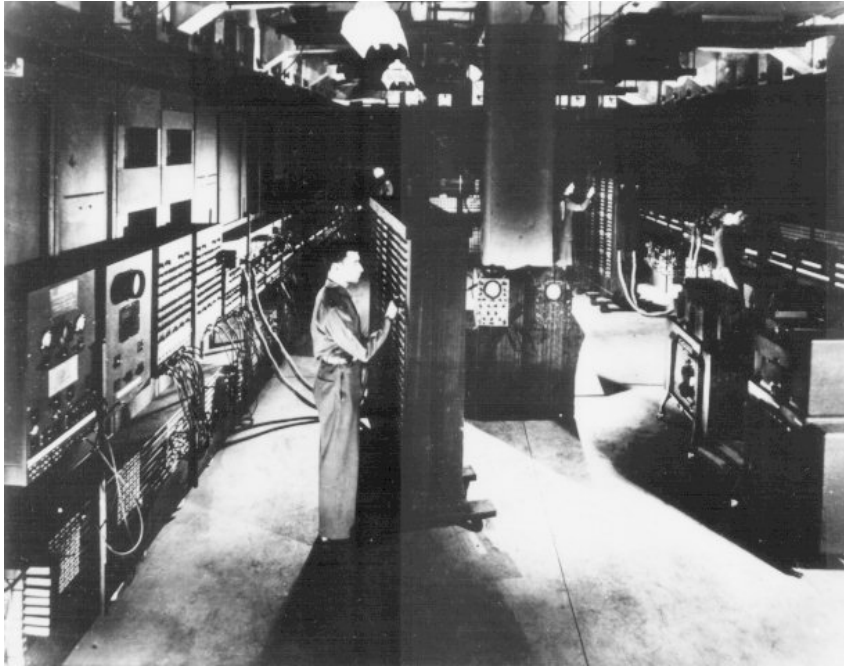
Bilgisayarların Gelişimi



Electronic Numerical Integration and Calculator (ENIAC)

- John Mauchly and J. Presper Eckert (1946'da tamamlandı)
- İlk olarak 2'inci dünya savaşında gizli bir proje olarak başladı.
- University of Pennsylvania

Bilgisayarların Gelişimi



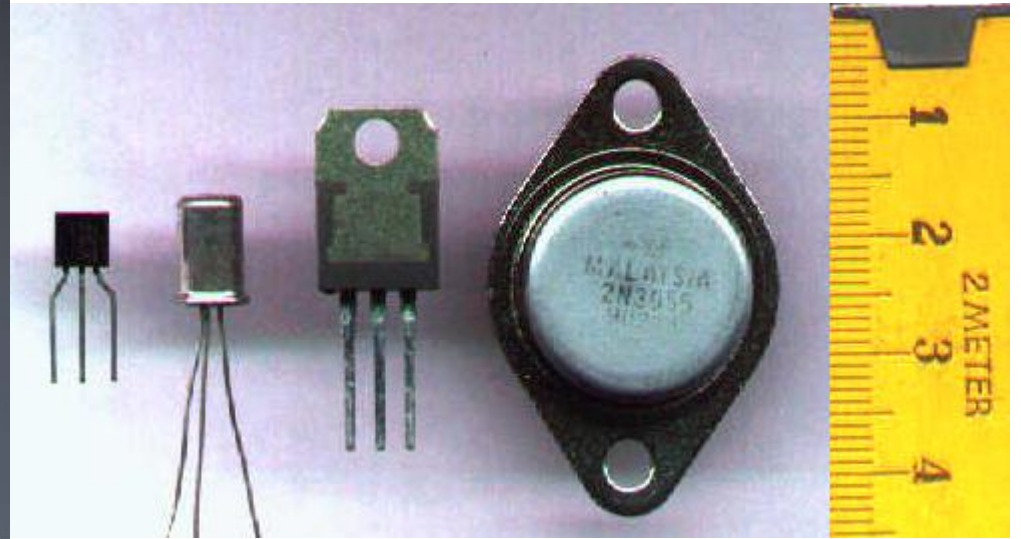
ENIAC

- 1000 metre kare alan
- 30 tons
- vacuum tüpleri kullanıyordu
- >17,000
- Karar verebiliyordu: **ilk gerçek bilgisayar**
- Programlama kablo temasları ve switch ayarları ile yapılıyordu.

Bilgisayarların Gelişimi

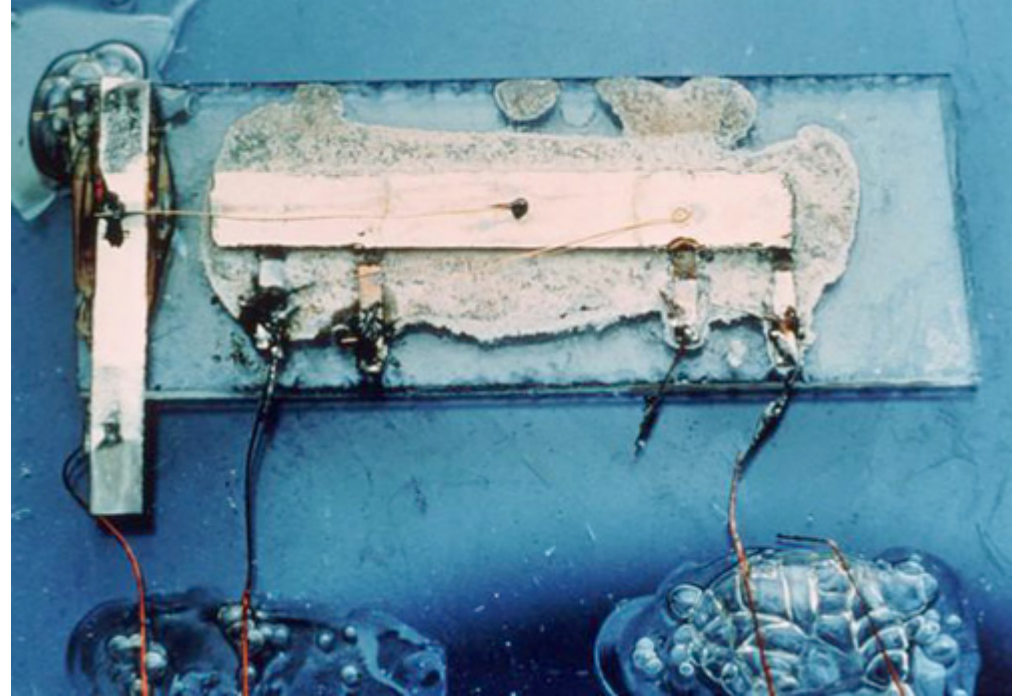
- ❖ 1945 yılında Bell laboratuvarlarında bir araştırma grubu kurulmuştur. Grubun amacı: iletkenler, yarıiletkenler, yalıtkanlar, piezoelektrik malzemeler ve manyetik malzemeler üzerinde temel araştırmalar yapmak, olarak tanımlanmıştır. Burada yapılan yarıiletkenler konusundaki çalışmalar sonucunda, Walter Brattain, John Bardeen ve William Shockley tarafından tranzistör icat edilmiştir. 1950 yılında bu yeni devre elemanının patenti alınmış ve 1951 yılında da Allentown Pennsylvania'da ticari olarak üretilmeye başlanmıştır.
- ❖ Tranzistörün icadı elektronikte devrim niteliğindedir.

Bilgisayarların Gelişimi



Bilgisayarların Gelişimi

1950'li yıllarda yapılan araştırmalar sonucunda çok sayıda tranzistör, diyot ve kapasiteden oluşan devrelerin bir bütün olarak gerçekleştirilmesi yolu bulunmuştur. Böylece ortaya tümdevreler veya entegre devreler (integrated circuit) çıkmıştır.

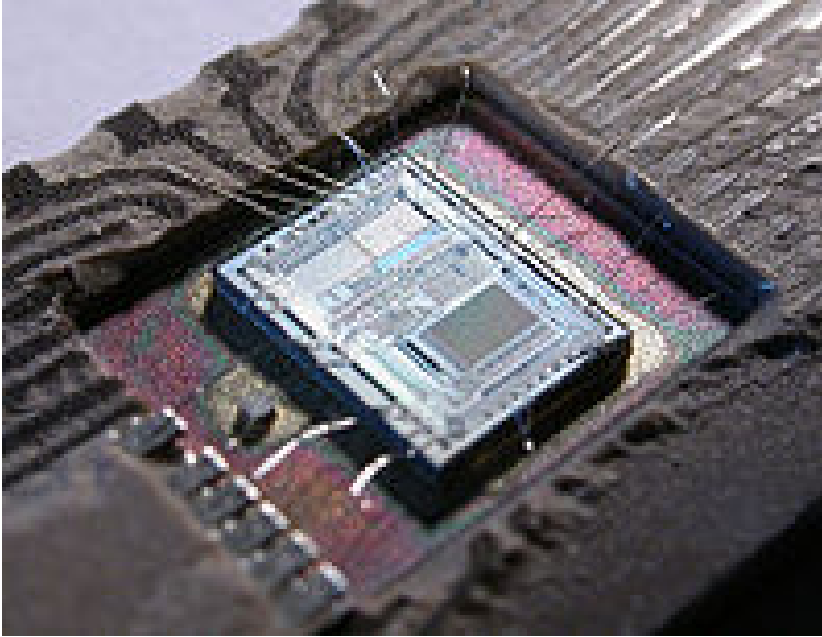


- ❖ Jack Kilby, 1958 yılında, Texas Instruments firmasında ilk tümdevreyi gerçekleştirmiştir.

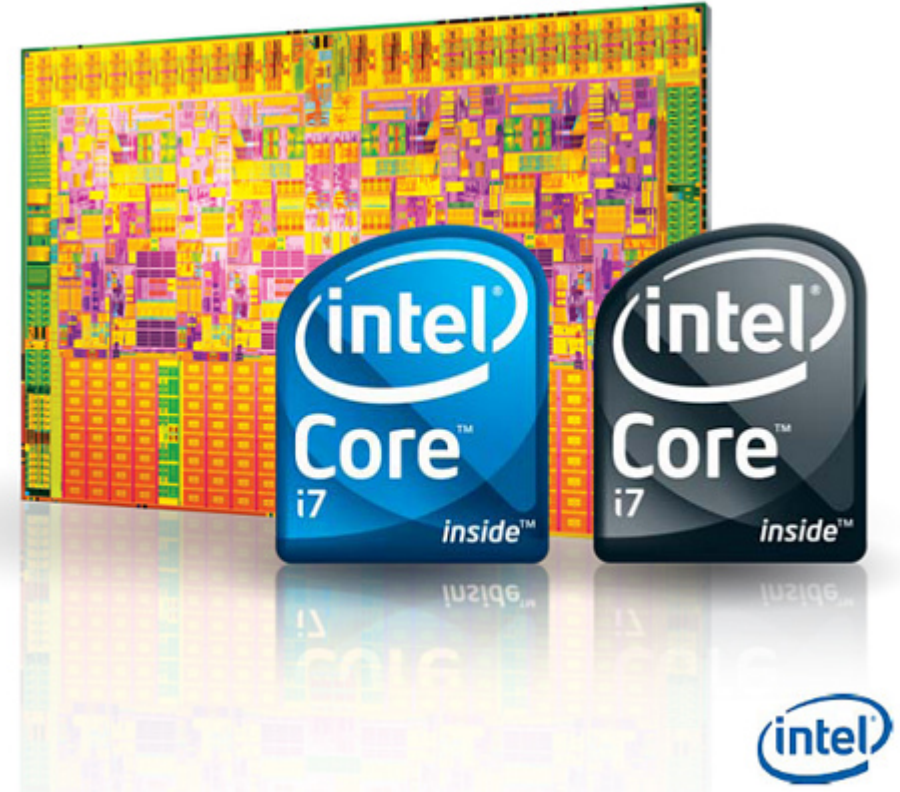
Bilgisayarların Gelişimi

- ❖ 1960 ve 1962 yılında yapılan çalışmalarda tümdevre teknolojisine BJT'lere göre daha uygun olan Metal-oksit-yarıiletken alan etkili tranzistör (metal-oxide-semiconductor field effect transistor-MOSFET) geliştirilmiştir (Kahng ve Atalla, 1960), (Hofstein ve Heinman, 1963).
- ❖ MOSFET transistorlerin gelişmesi ile birlikte tümdevre içine çok daha fazla sayıda transistor yerleştirilebilmiştir. Bir tümleşik devredeki eleman sayısı 1964'te 40'a ve 1972'de 1200'e yükselmiştir. 1982'li yıllarda VLSI (Very Large-Scale Integration) olarak isimlendirilen sistemlerde 100,000'ler mertebesinde eleman içeren tümleşik devreler gerçekleştirilmiştir.
- ❖ Günümüzde bu eleman sayıları çok daha büyük değerlere ulaşmıştır.

Bilgisayarların Gelişimi



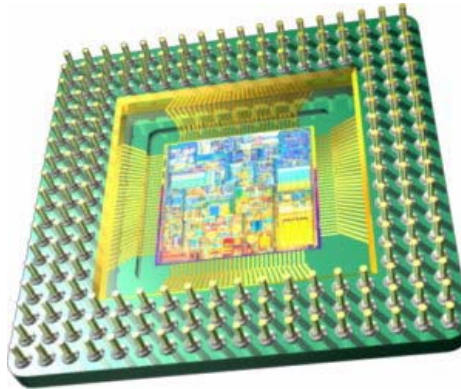
Intel 8742 8-bit mikrokontroller. İşlemci hızı 12 MHz, 128 bytes Ram, 2048 bytes EPROM, giriş çıkış uçları. Hepsi bir tümdevrede



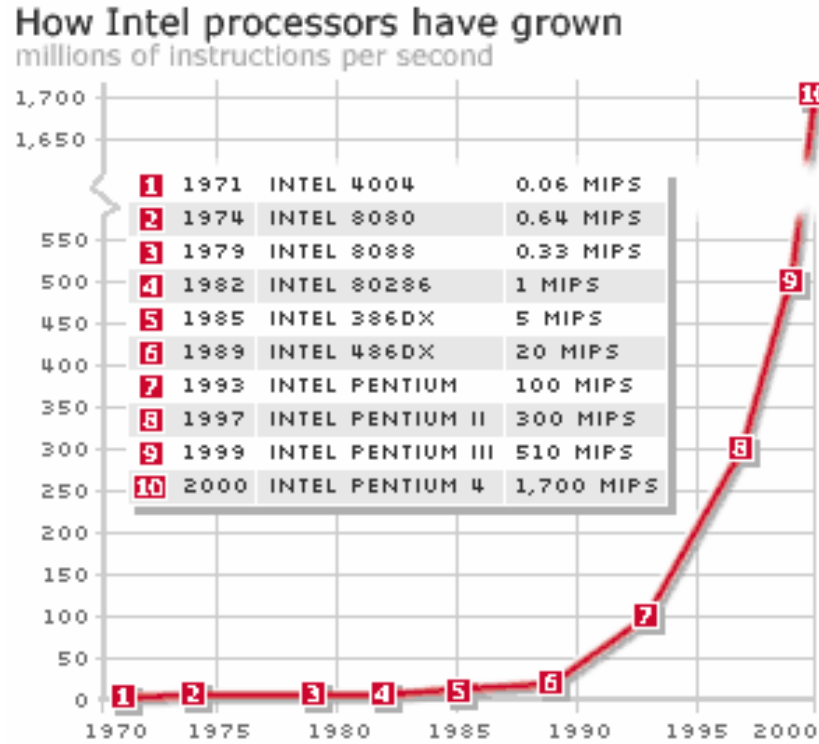
781 Milyon tranzistör
bir tümdevrenin içinde

Bilgisayarların Gelişimi

- **Microprocessor**: CPU içeren tek bir chip
 - İlk olarak 1970 yılında Marcian Hoff (Intel Corporation) tarafından tasarlandı
- **Microcomputer**: masaüstü boyutlarında bilgisayar
 - ALTAIR (1975)
 - Apple (Stephen Wozniak ve Steven Jobs; 1977)



Bilgisayarların Gelişimi



From the BBC

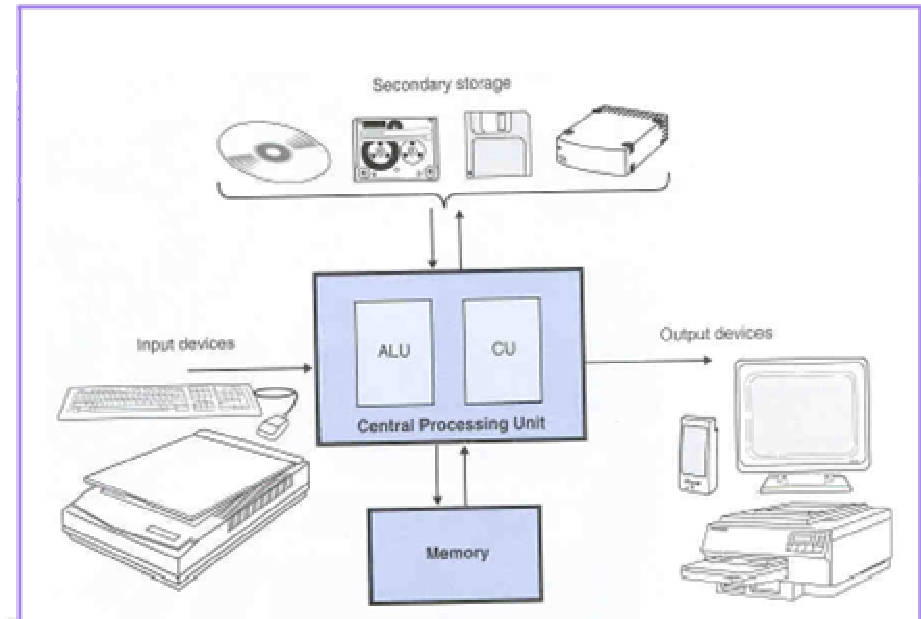
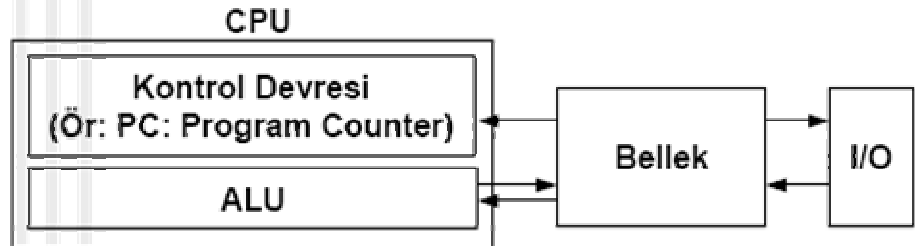
- Her 18-24 ayda bir işlemci gücü ikiye katlanıyor.

Bilgisayarlar Mimarileri

Temel Bilgisayar Mimarileri

- Von Neumann mimarisi
- Harvard Mimarisi

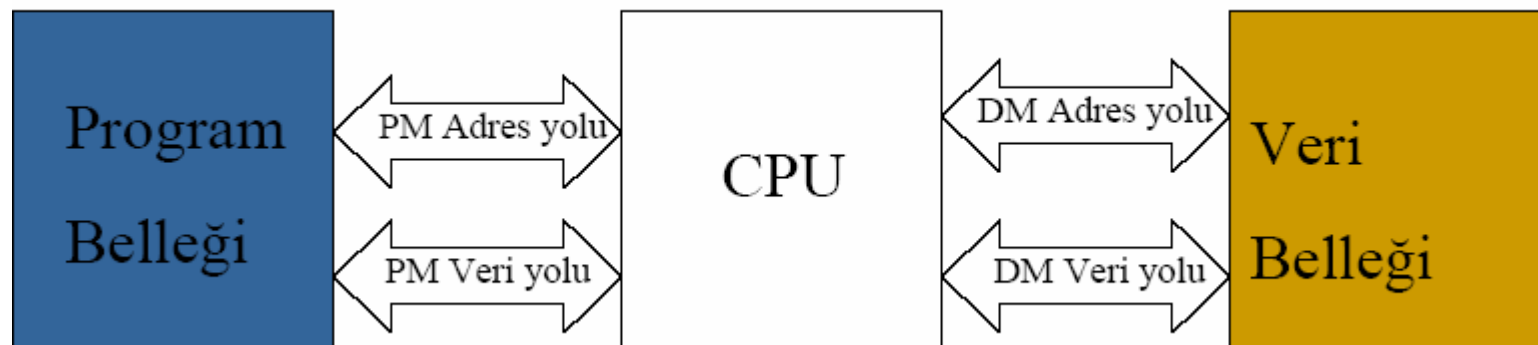
Von-Neumann Mimarisi



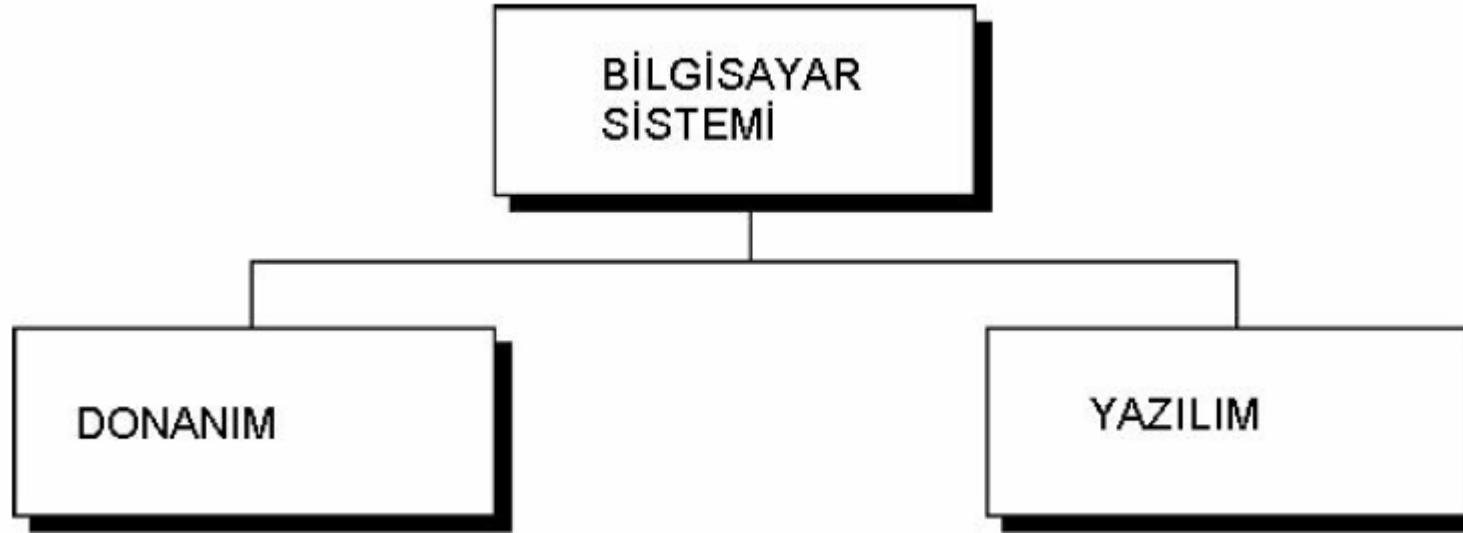
Bilgisayarlar Mimarileri

Harvard Mimarisi

Günümüz tipik bilgisayarları Von-Neumann Mimarisine sahip Mikroişlemciler kullanırken (Intel x86, Pentium, AMD Athlon..) , Özellikle Görüntü, ses işleme, yüksek hız gerektiren uygulamalarda Harvard mimarisine sahip mikroişlemciler (DSP'ler, ARM Cortex..)

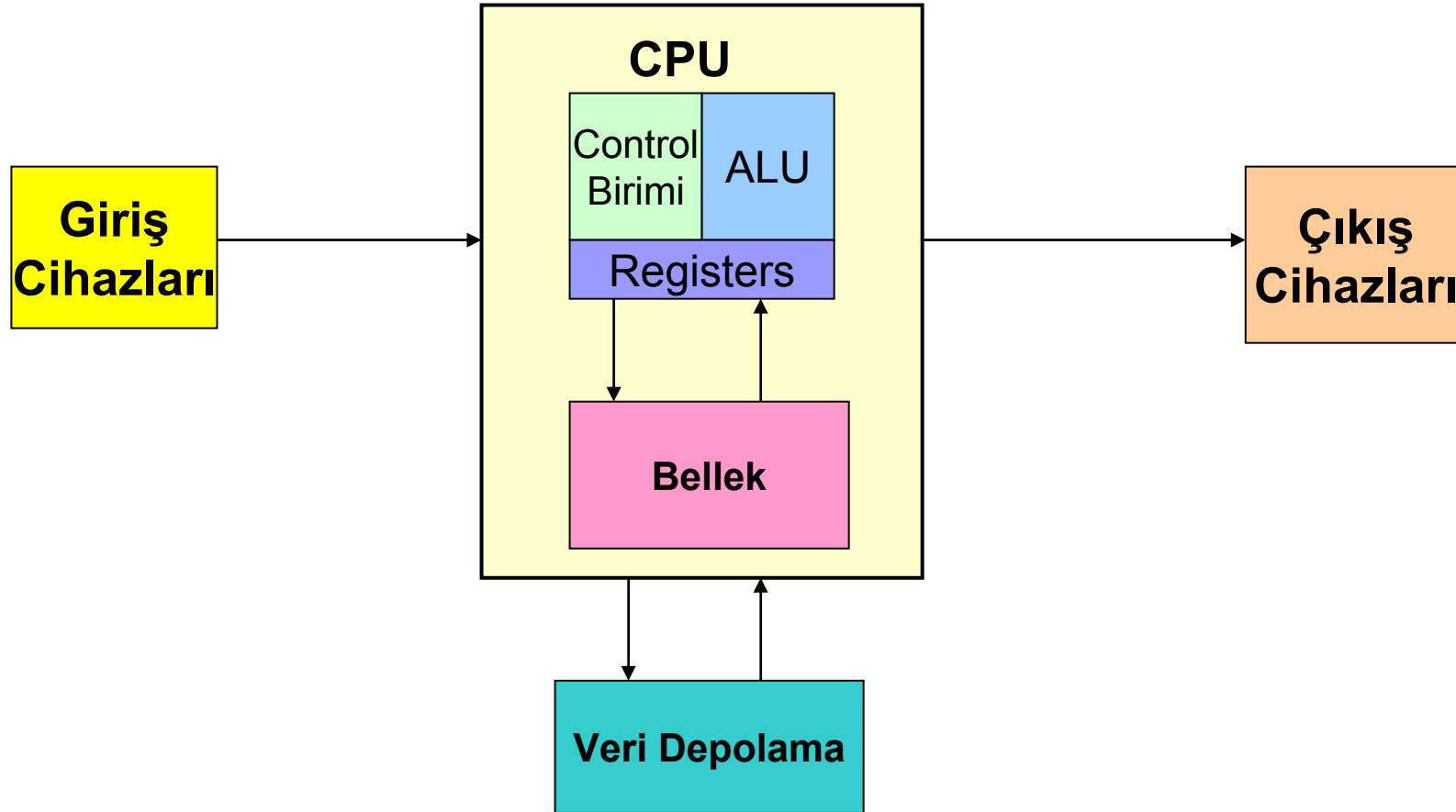


Bilgisayar Sistemi



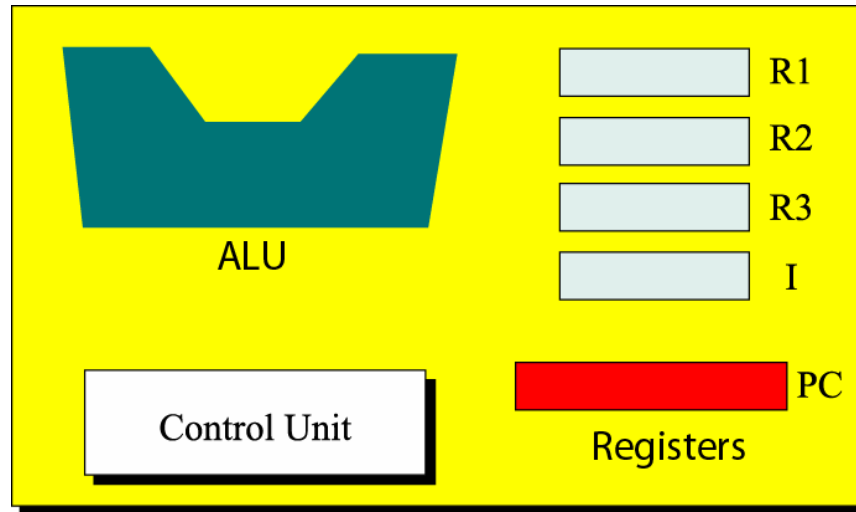
1. Donanım fiziksel aygıtlardır.
2. Yazılım ise yapılması gereken işleri yapabilmek için donanıma komutlar veren programlar topluluğudur.

Donanım



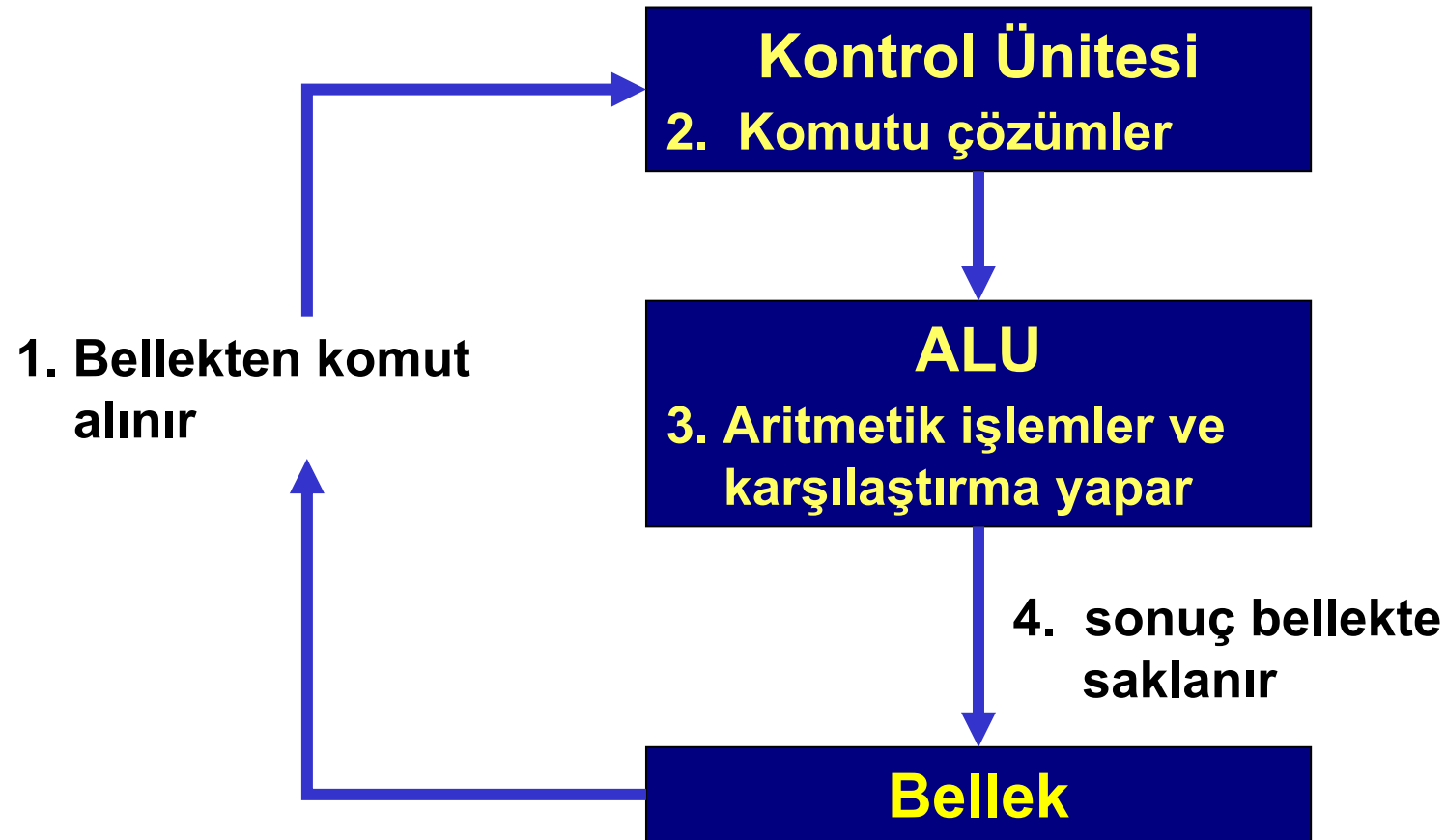
Donanım

- Görevleri yapabilmek için komutları işleyen mikroişlemciye CPU denir.
- CPU nelerden oluşur:
 - Kontrol Ünitesi
 - Aritmetik mantık ünitesi (Arithmetic Logic Unit)
 - Register



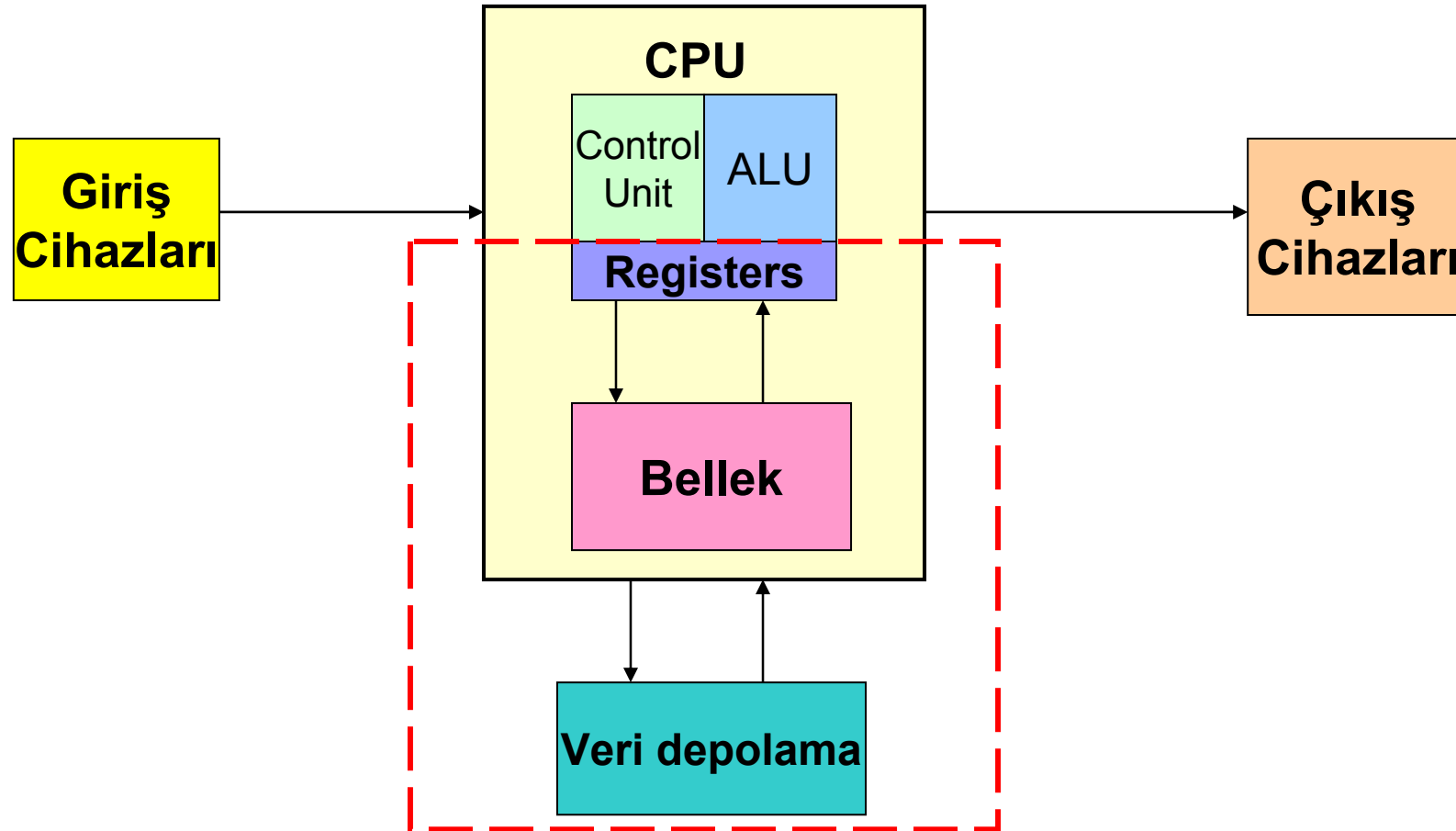
Donanım

CPU'daki Komut Döngüsü



Donanım

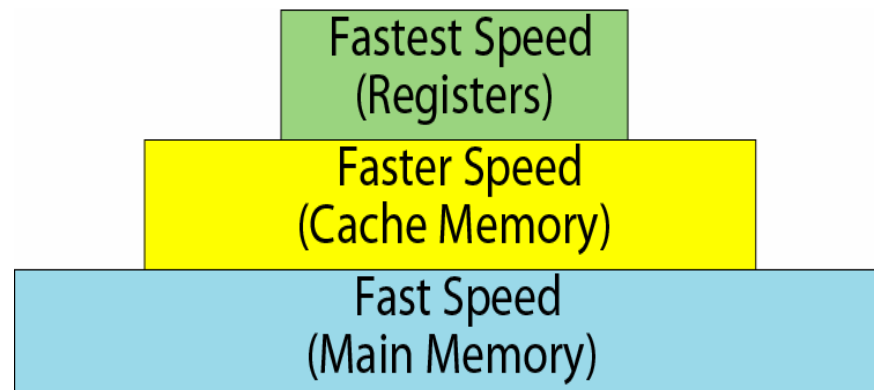
Veri Depolama Üniteleri



Donanım

Bellek Türleri

- **Registers** – CPU'nun bir parçası; çok hızlı; sınırlı büyüklük
- **Cache Memory** – CPU'nun bir parçası; RAM'den daha hızlı
- **Read-only Memory (ROM)**
 - Bilgisayarın sürekli ihtiyaç duyduğu sistem komutlarını barındıran chip
- **Random Access Memory (RAM)** – Ana karta eklenen bellek; program komutları ve veriler için birincil depo



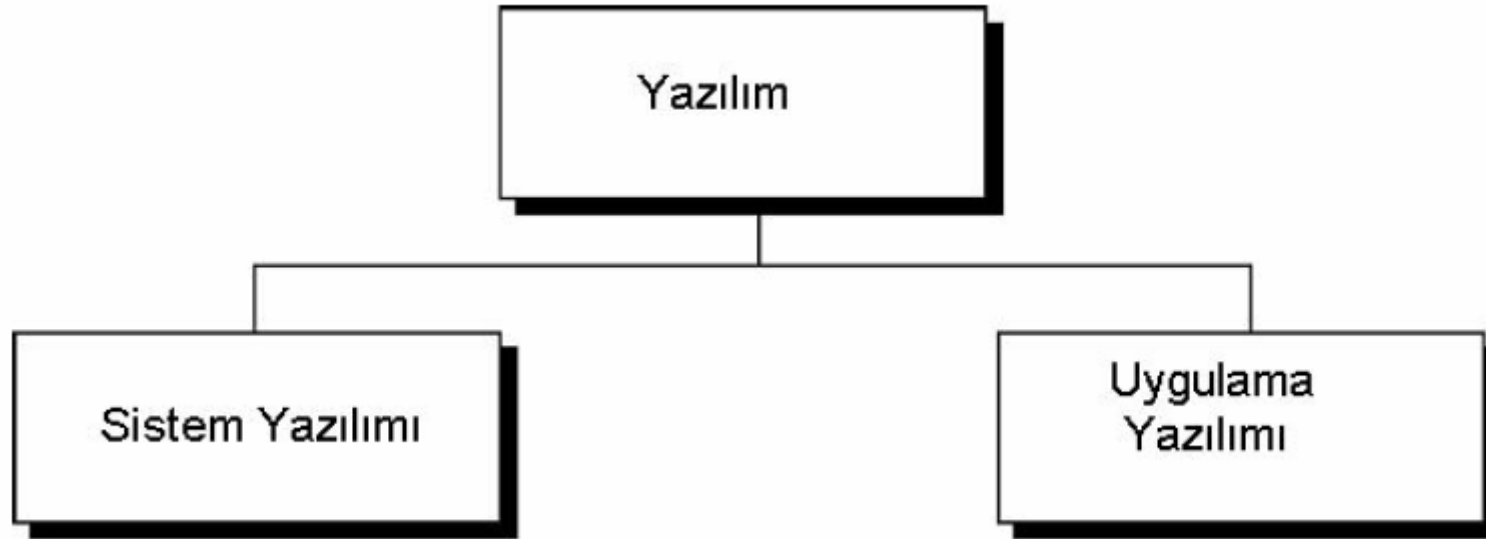


Donanım

Diğer Bilgisayar Bileşenleri

- Veri depolama sistemi
 - Hard disk, tape, floppy, DVD vs.
 - Geniş alan, ucuz, yavaş, manyetik ve optik
- Input Cihazları
 - Klavye, Fare, Dokunmatik ekran, Tarayıcı, Webcam, Joystick, Mikrofon
- Output Cihazları
 - Monitör, Yazıcı, Plotter, Hoparlör

Yazılım



Sistem Yazılımı:

1. Aygıt Yazılımı (Firmware) (BIOS).
2. İşletim Sistemi
3. Sistem destek yazılımı
4. Sistem Geliştirme Yazılımı

Uygulama yazılımı:

1. Genel Amaçlı
2. Uygulamaya Özel



Yazılım

Sistem Yazılımı

•**Aygıt Yazılımı**: Sistemi oluşturan donanımların çalışması için gerekli olan yazılımlardır.

•**İşletim sistemi**: Kullanıcı arayüzü, ağ bağlantı arayüzleri, Dosya erişimi ve organizasyonu, Çoklu çalışma gibi hizmetleri sağlayan yazılımlardır. Örneğin: DOS, Windows, Linux, PARDUS, Unixvs..

•**Sistem destek yazılımları**: Sistemle ilişkili faydalı yazılımlardır. Örneğin, Disk formatlayıcı, hesap makinesi, test ve iletişim yazılımları, Hyperterminal, Telnet vs..

•**Sistem Geliştirme Yazılımları**: Bunlar, çeşitli kütüphaneler, Uygulama Programı arayüzü(API) (Winsock, setupapi, mmttools, SAPI, DDK..), Derleyiciler, Debugger'lar..



Yazılım

Uygulama Yazılımları

•Genel Amaçlı

- Kelime işlem programları: MS-Word, Word-Pro, ...
- Veri tabanı yönetim programları: Oracle, Access, SQL, ...
- Hesap Tablosu programları: MS-Excel, Lotus, ...
- Grafik ve çizim programları: AutoCAD, 3D MAX, Photoshop, Corel Draw, ...
- Matematik tabanlı programlar: MATLAB, MatCAD, Mathematica, ...
- ...

•Özel yazılımlar



Program ve Yazılımın Tanımı

Program: belirli bir işi gerçekleştirmek için gerekli komutlar dizisi olarak tanımlanabilir.

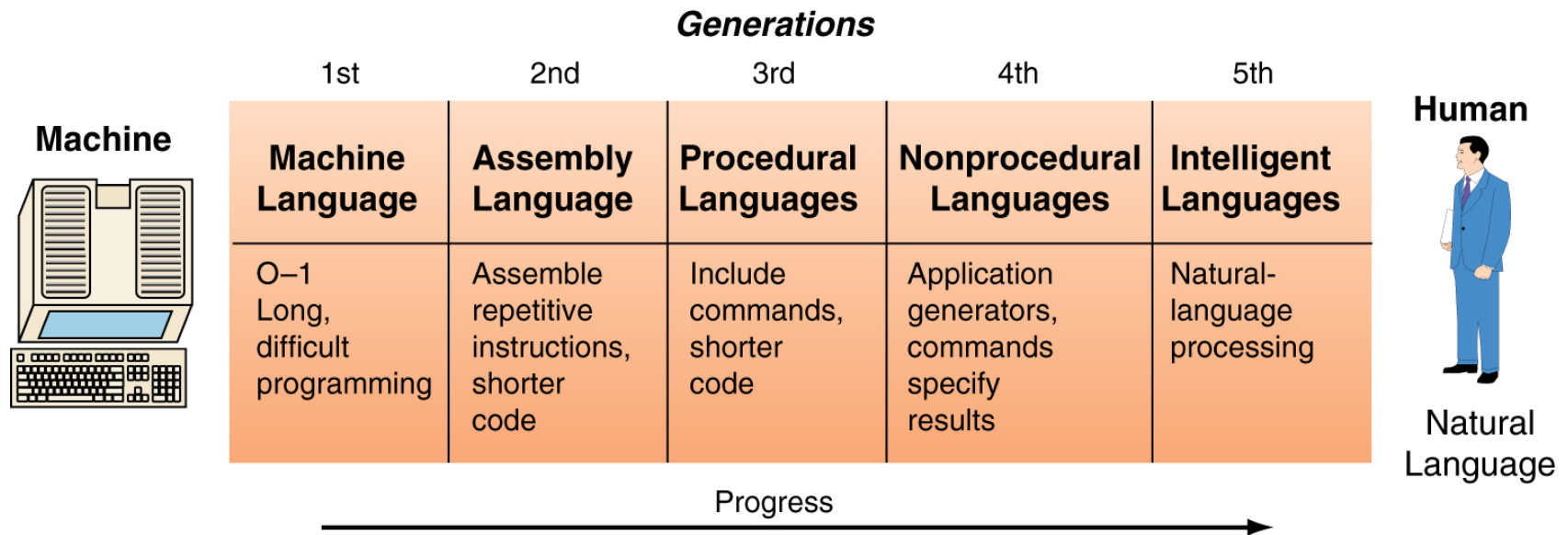
Programlama: Bir programı oluşturabilmek için gerekli komutların belirlenmesi ve uygun biçimde kullanılmasıdır.

Programlama Dilleri: Bir programın oluşturulmasında kullanılan komutlar, tanımlar ve kuralların belirtildiği programlama araçlarıdır.

Yazılım: Belirli bir amacı sağlayan, program yada programlar ve ilgili dokümantasyonlardır.

Programlama Dilleri

- Bilgisayarlara ne yapmaları gerektiğini söylememizi sağlayan özel bir dil
- Tüm yazılımlar programlama dilleri ile yazılır.



Programlama Dilleri

- Makine dili (**birinci seviye**)
 - Bilgisayarın ana dilidir.
 - İkicil (binary) kodlardan oluşur (0'lar ve 1'ler)
 - Örn. **0110 1001 1010 1011**
 - Bilgisayarın anlayabildiği tek dildir.
- Assembly Dili (**ikinci seviye**)
 - Makine diline birebir çevrilebilir
 - Makine dilinden daha kolay anlaşılabilir (ama çok da değil)
 - Örn. **ADD X Y Z**
 - Assembler – assembly dilini makine diline çeviren program



Programlama Dilleri

- **Procedural diller (üçüncü seviye)**
 - Bir komut pek çok makine dili komutuna karşılık gelir
 - Programlarda bilgisayarın işlem akışını adım adım tasarlayabilirsiniz.
 - İnsan diline daha çok benzer; bilinen kelimeleri kullanır
 - Örnek: C, C++, Java, Fortran, QuickBasic
 - Derleyici (compiler) – programın tümünü assembly veya makine diline çevirir (C++, Pascal, Ada).
 - İnterpreter – program çalıştırıldığında adım adım programı makine koduna çevirir (Basic, Javascript, LISP)

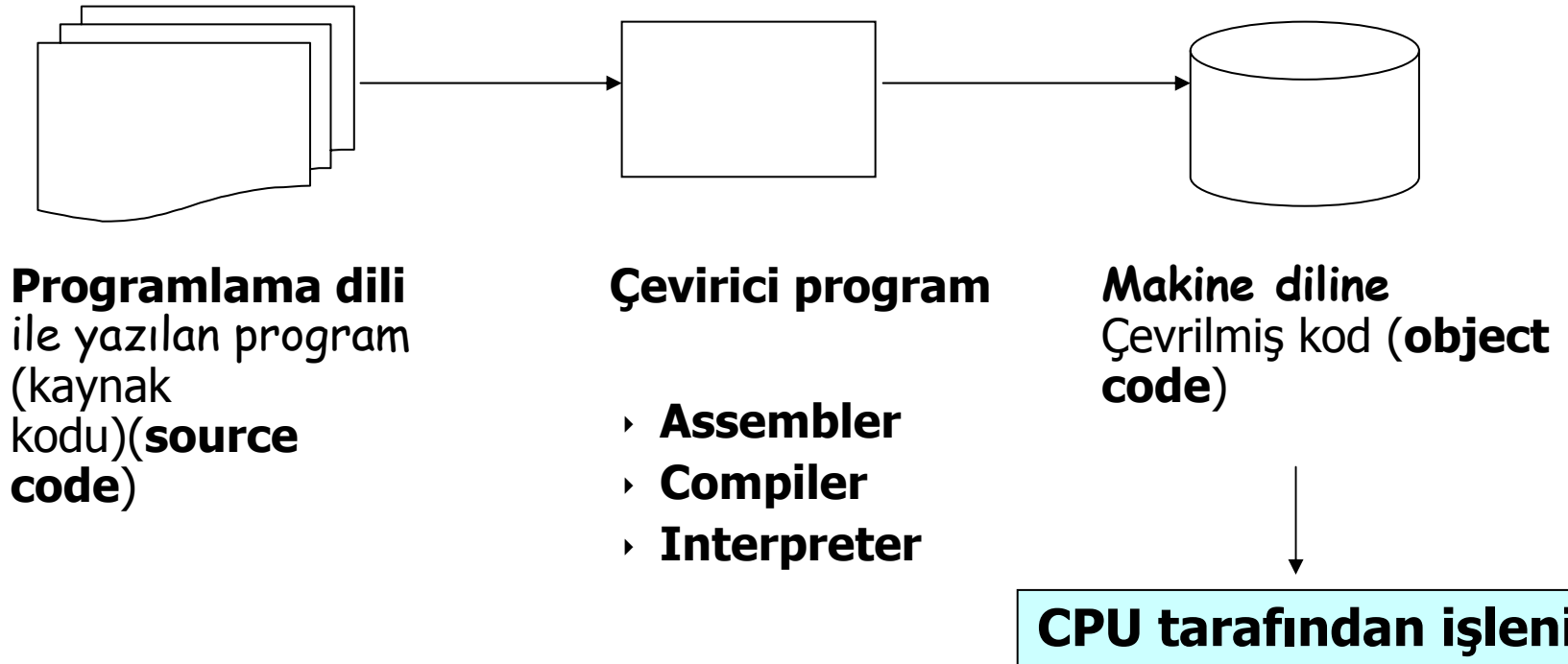


Programlama Dilleri

- **Nonprocedural Diller (dördüncü seviye)**
 - Kullanıcının sadece gerekli sorguyu göndermesi sonuca ulaşması için yeterlidir.
 - Örnek: – veritabanı sorgulama dili- SQL
 - Teknik olmayan insanlar tarafından da kullanılabilir.
- **Natural Language Programming Languages (beşinci seviye (akıllı diller))**
 - İnsan dilini programlama diline çevirir.
 - Oldukça karmaşık ve yenidir.

Programlama Dilleri

■ Dilden dile çevrim



Programlama Dilleri

Makine Dilinde Çarpma İşlemi

	00000000	00000100	000000000000000000
01011110	00001100	11000010	000000000000000010
	11101111	00010110	000000000000000101
	11101111	10011110	000000000000001011
11111000	10101101	11011111	00000000000010010
	01100010	11011111	00000000000010101
11101111	00000010	11111011	00000000000010111
11111100	10101101	11011111	00000000000011110
00000011	10100010	11011111	00000000000100001
11101111	00000010	11111011	00000000000100100
01111110	11110100	10101101	
11111000	10101110	11000101	00000000000101011

Programlama Dilleri

Bir Assembly programı Örneği:

```
LDI temp,0x80 ; Analog Comparator disabled
OUT ACSR,temp
LDI temp,0x00
OUT DDRB,temp ; PORTB giriş
LDI temp,0b01110000 ; PD0,PD1,PD2,PD3 inputdiğerleri output
OUT DDRD,temp
LDI temp,0b01000000 ; initPORTD
OUT PORTD,temp
CLR hat1_time_out ;ilk deđerleri atama bölümü
CLR hat2_time_out
CLR temp
LDI ZH,0 ;hat1 temp buffer'ı boşalt
LDI ZL,hat1_temp_adres
ST Z,temp
LDI YH,0 ;hat2 temp buffer'ı boşalt
LDI YL,hat2_temp_adres
ST Y,temp
LDI XH,0
```



Programlama Dillerinin Tarihçesi

İlk programın, Ada Lovelace tarafından Charles Babbage'ın tanımlamış olduğu "Analytical Engine" ile Bernoulli sayılarının hesaplanmasına ilişkin makalesinde olduğu söylenmektedir. Bu nedenle ilk gerçek anlamdaki programlama dillerinden birinin adı Ada Lovelace anısına ADA olarak isimlendirilmiştir. 1940 larda ilk modern bilgisayar ortaya çıktığında, programcılar yalnızca assembly dili kullanarak yazılım yapabiliyorlardı.

1950 –1960

- FORTRAN (1955), the "**FOR**mula **TRAN**slator
- LISP, the "**LI**St **P**rocessor",
- COBOL, the **CO**mmon **B**usiness **O**riented **L**anguage
- ALGOL **A**lgorithmic **L**anguage

Programlama Dillerinin Tarihçesi

- 1962 - APL
- 1964 - BASIC
- 1964 - PL/I
- 1970 - Pascal → Yapısal programlama
- 1970 - Forth
- 1972 - C
- 1972 - Prolog
- 1978 - SQL → Nesne yönelimli dillerin ortaya çıkışı
- 1983 - Ada
- 1983 - C++
- 1987 - Perl

1990 lar, Internet

- 1991 - Python
- 1991 - Java
- 1995 - PHP
- 2000 - C#

Tamamı nesne yönelimli dillerdir. Yeni programlama kavramlarından ziyade, programlamanın kolaylaşmasını ve taşınabilirliği amaçlamaktadırlar

Yazılım Geliştirme Araçları

- Editörler-Tümleşik geliştirme ortamları(IDE)
- Derleyicilerle birlikte kullanılır
- Derleyiciler-Bağlayıcılar (Compilers–Linkers)
- Yorumlayıcılar (Interpreter)

Editörler

Program kodlarını yazmak için kullanılan, metin düzenleyicilerdir. Program kodları saf metin biçiminde yazıldığından, herhangi bir metin düzenleyicisi, program yazılımı için kullanılabilir.

- Kodlamadaki hatalar görülmez.
- Anahtar kelimeler, fonksiyonlar ve parametreleri, vb.. Tanımlar ayrı renklendirilmediğinden kod yazmak daha zordur.
- Breakpoint, yada watch gibi, hata ayıklama unsurları yoktur.
- Notepad, Wordpad.. editör olarak kullanılabilir.
- Program derleme ve bağlama işlemi editör dışında genellikle komut satırı üzerinde yapılır.



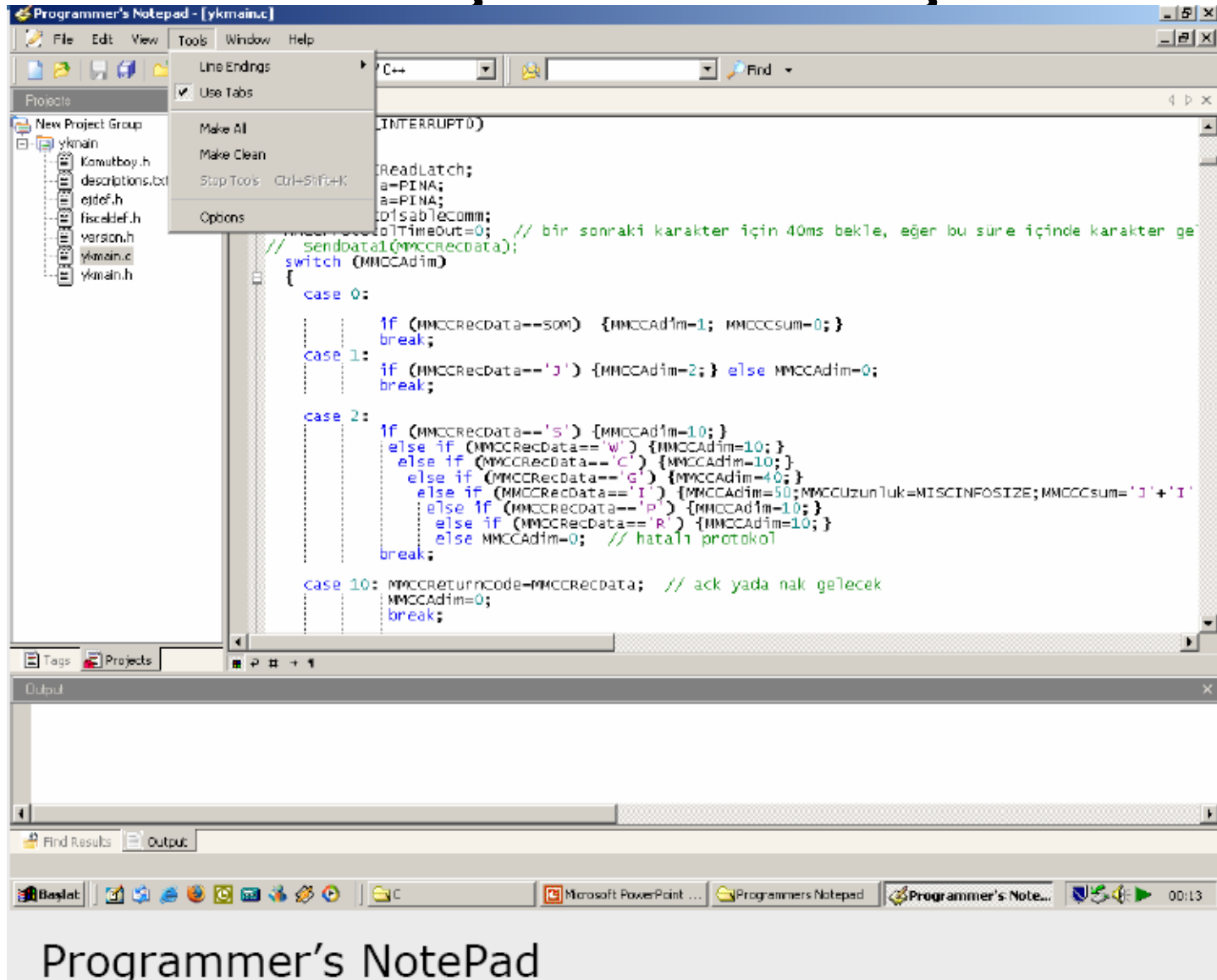
Yazılım Geliştirme Araçları

IDE (Tümleşik geliştirme ortamı)

Tümleşik geliştirme ortamları, Genellikle derleyicileri –bağlayıcıları ortam içinden kullanabilmeyi yada derleyici ve bağlayıcıya ortam içinden erişme yollarını sağlarlar (Makefilevs..). Bunun yanı sıra;

- Derleyici ve bağlayıcı tümleşik olan yapılarda Hata ayıklama, Gözlem penceresi gibi bileşenler mevcuttur.
- Yazım işlemini kolaylaştıracak vurgulamalar ve uyarılar mevcuttur.
- Derleyici ve bağlayıcı parametreleri menülerden ayarlanabilir.
- Yardımlar mevcuttur.
- Her yazılım dilinin kendi IDE si mevcuttur. Ancak bazı IDE'ler birden fazla yazılım dili için ortam sağlayabilir.

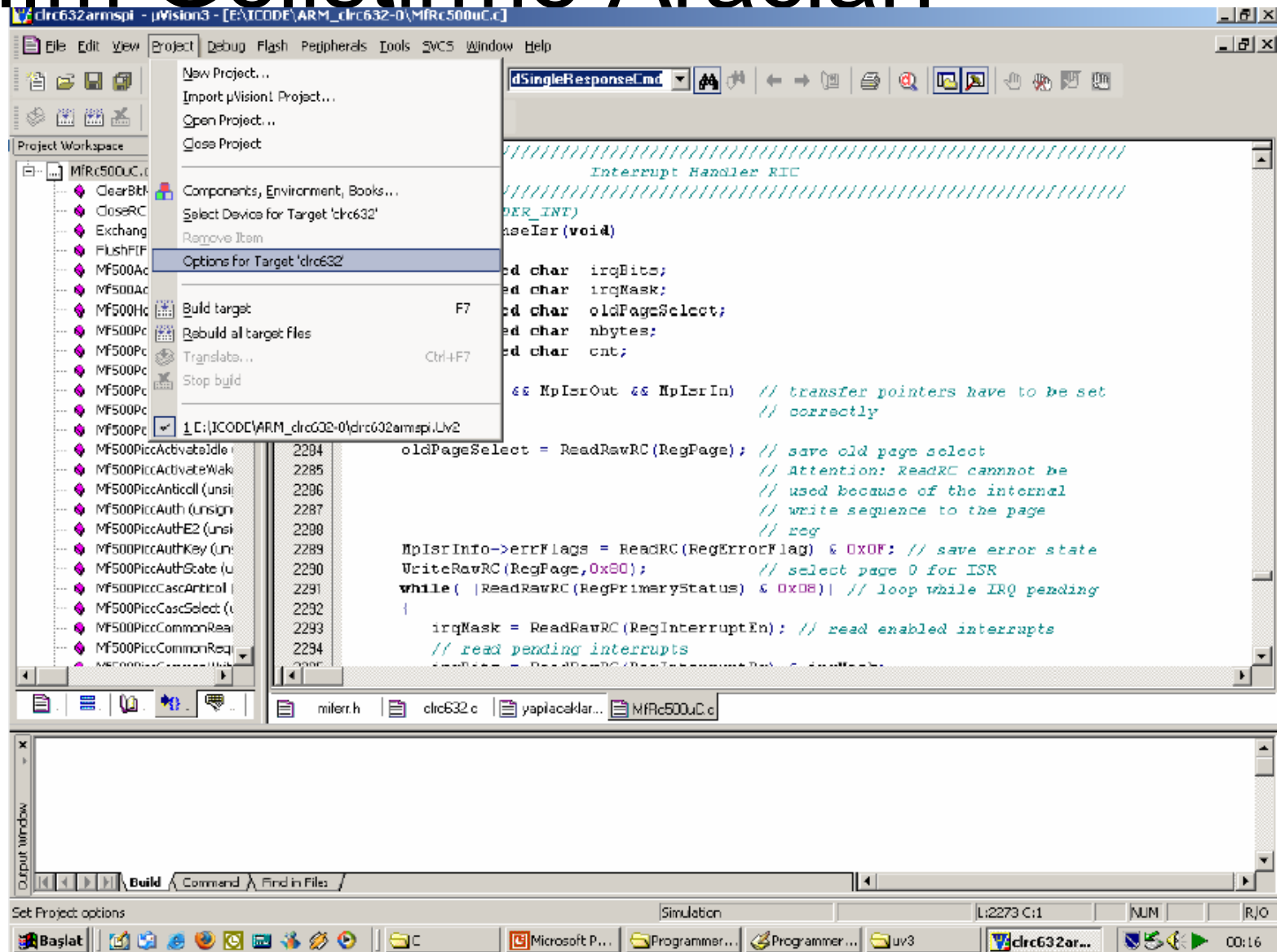
Yazılım Geliştirme Araçları



Programmer's Notepad

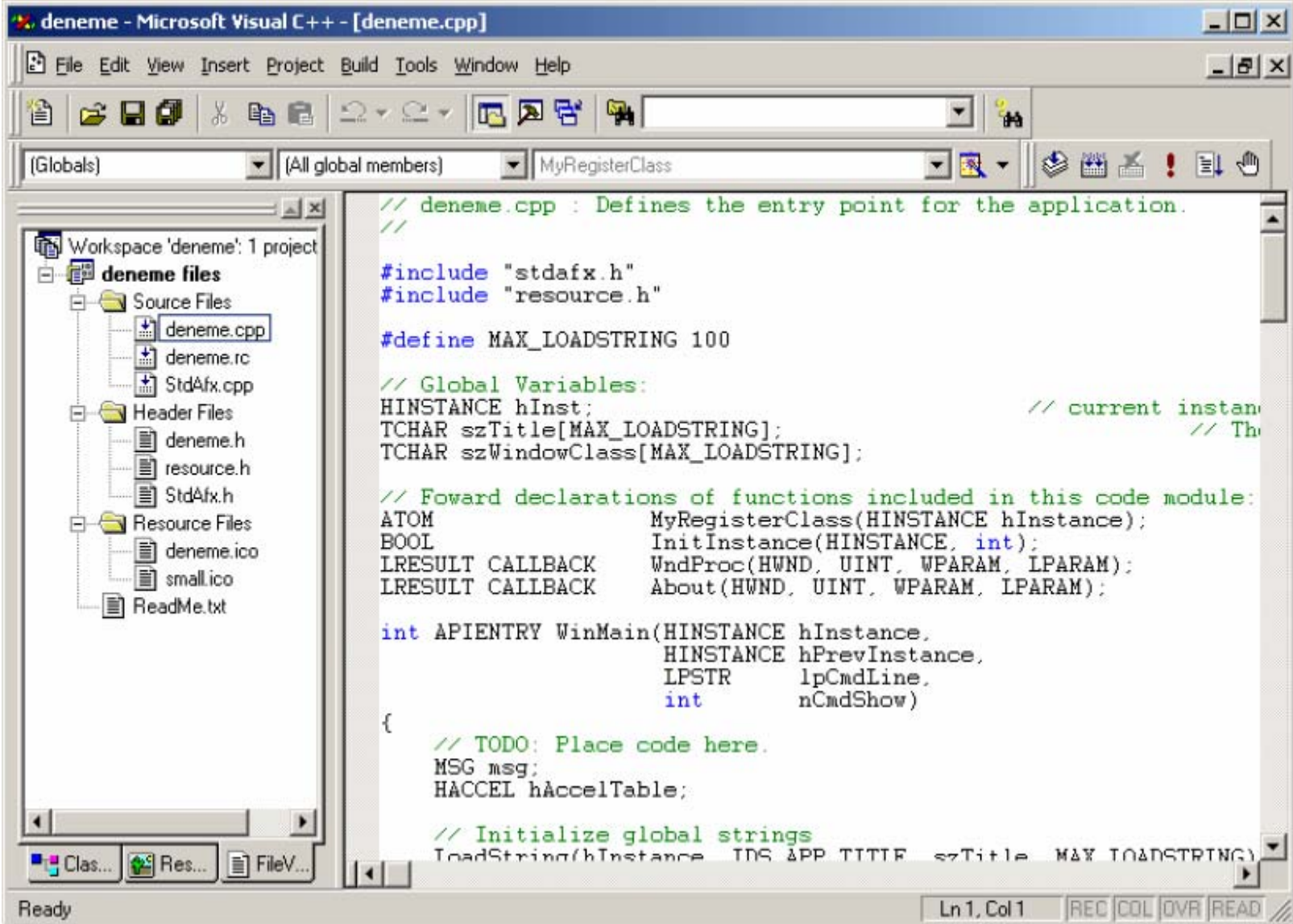
Yazılım Gelistirme Araçları

Gömülü sistem
programlaması
için Keil
uVision3 IDE



Yazılım Geliştirme Araçları

Windows
Ortamındaki
geliştirmeler
için MS VS
C++ 6.0



```
// deneme.cpp : Defines the entry point for the application.
//
#include "stdafx.h"
#include "resource.h"

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst;                                // current instance
TCHAR szTitle[MAX_LOADSTRING];                 // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];

// Forward declarations of functions included in this code module:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK    About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR     lpCmdLine,
                    int       nCmdShow)
{
    // TODO: Place code here.
    MSG msg;
    HACCEL hAccelTable;

    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
```

Yazılım Geliştirme Araçları

Derleyiciler: Bir derleyici, bir metin editörü yada IDE üzerinde yazılan program kodlarını, makinenin anlayabileceği OBJ kodlara dönüştüren bir uygulama yazılımıdır. Derleyicilerin birçoğu, Program dilinin yanısıra makine dilinin(assembly) de kullanılmasına izin verir.

Bağlayıcılar: Bir bağlayıcı, derleyici tarafından derlenmiş olan OBJ program kodlarını uygun bellek bölgelerine yerleştirerek, değişkenlerin ve sabitlerin bellek atamalarını ve ilklemelelerini gerçekleyerek tek bir çalıştırılabilir program elde eden bir uygulama yazılımıdır (windows için exe dosya).

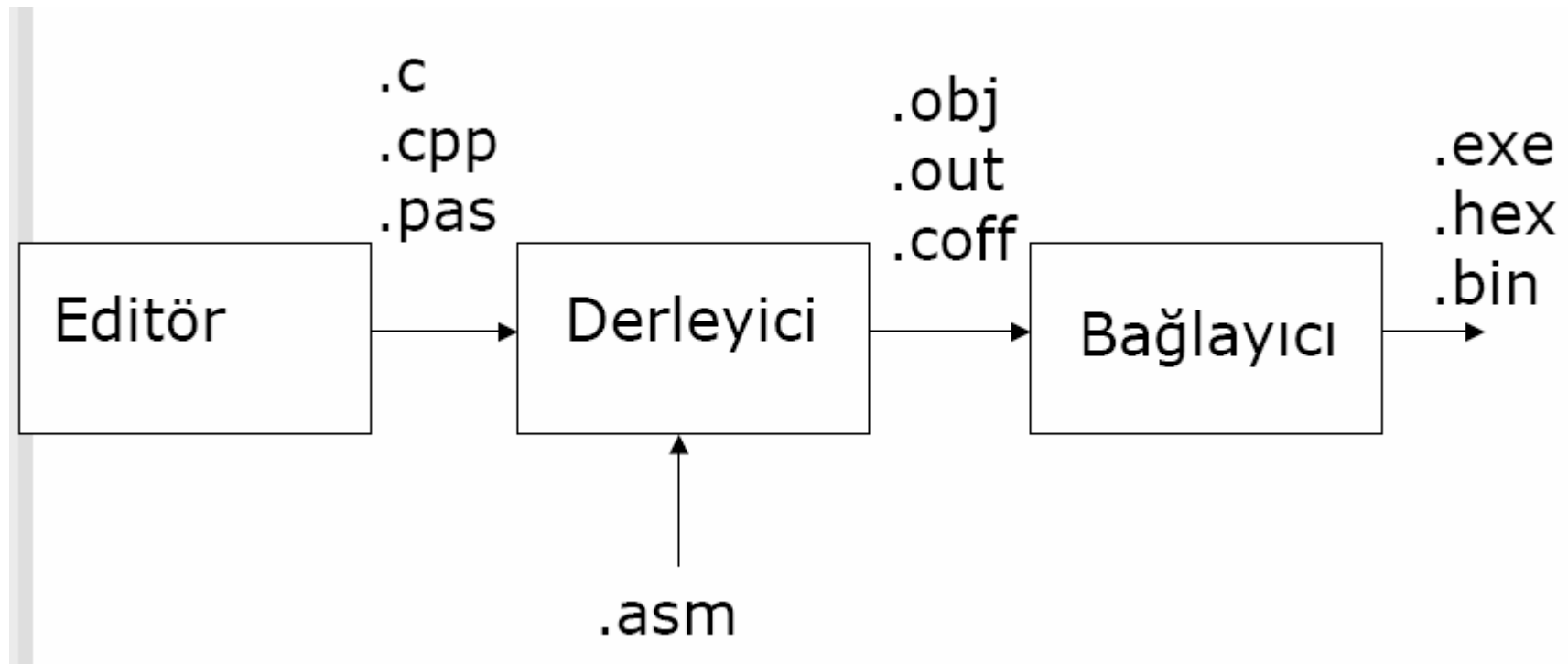
Örnek derleyiciler ve bağlayıcılar:

MS VC++ 6.0 için CL.exe derleyici, Link.exe bağlayıcı

Keil uVision 8051 için c51.exe derleyici, Ld51.exe bağlayıcı

gcc.exe açık kaynaklı ücretsiz bir derleyici ve bağlayıcı

Yazılım Geliştirme Araçları





Yazılım Geliştirme Araçları

YORUMLAYICILAR (INTERPRETERS)

Yorumlayıcılar, program kodunu bir bütün olarak değerlendirmez. Program kodunu satır, satır yorumlayarak çalıştırırlar. Bu nedenle günümüzde derleyicilere göre daha kısıtlı uygulamalara sahiptirler, internet uygulamaları ve bilimsel alanda yaygın kullanılmaktadırlar.

- Bazı yorumlayıcılar, yazılan program satırını, daha etkin bir biçime çevirip, hemen uygularlar. Bunlar arasında: *Perl*, *Phyton*, *Matlab*, *Mathcad* gibi yorumlayıcılar sayılabilir.
- Bazı yorumlayıcılar ise, yorumlayıcı sistemin bir parçası olan bir derleyici tarafından önceden derlenip saklanmış kodları uygularlar. *Java* bunlar arasında sayılabilir.

Sayı Sistemleri

Günlük yaşantımızda 10 luk sayı sistemi kullanılır. Ancak, bilgisayar sistemleri 2 lik sayı sistemini kullanırlar. 10 luk sistemde taban 10, ikilik sistemde taban 2 dir.

Sayı sistemlerinde sayıyı oluşturan her bir rakam **digit** olarak adlandırılır. Onluk sayı sistemlerinde her bir rakam **decimal digit** yada sadece **digit**ken, ikilik sistemde binary digit yada kısaca **bit** olarak adlandırılır.

123456 6 digitlik onlu sayı

100101 6 bitlik ikili sayı

Sayı sembolleri 0 .. (Taban-1) arasındadır.

Onluk düzende rakamlar 0..9, ikilik düzende rakamlar 0 , 1 den oluşur.

Sayıların oluşturulması

$$123456 = 1 \cdot 10^5 + 2 \cdot 10^4 + 3 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0$$

$$100101 = 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$



Sayı Sistemleri

Sekiz bitlik ikili sayılara bir byte lık sayılar denir

10011101 8 bit yada bir **byte**dir.

16 bit uzunluklu sayılara 1 word luk sayılar sayılar denmesine rağmen, bu kavram bazen işlemcinin veri yolu uzunluğu kadar bit sayısı ile de eşleştirilmektedir.

11001001 11100011 2 byte lık yada 1 **word**luk sayı.

Ayrıca her 4 bit, bir **Nibble**olarak adlandırılır.

Sayı Sistemleri

POZİTİF VE NEGATİF SAYILAR

Bir byte'lık en küçük ve en büyük pozitif sayılara bakalım

00000000 (decimal 0)

11111111 (decimal 255)

Buradaki tüm sayılar, pozitifdir. Bir başka deyişle sayı işaretsizdir. Negatif sayılar söz konusu olduğunda bu sayıların yarısının pozitif, yarısının negatif olduğu söylenebilir.

Örneğin 1 byte'lıksayı -127 ile +127 arasında değişecektir.

İkilik sistemde negatif sayılar, çıkarma işleminin toplama aracılığıyla yapılabilmesini sağlamak amacıyla tümleyen sayılarla gösterilir.

Tümleyen sayı, verilen sayıyı, o bit sayısı için temsil edilen en büyük sayıya tamamlayan sayıdır. (Pratikte bit evirerek yapılır.)

Örneğin 00001010 ın tümleyeni 11110101 dir. (255 –10). Bu türden tümleyene **1'e tümleyen** sayı denir. Dikkat edilirse en ağırlıklı (en soldaki) bit negatif sayılar için 1 olmaktadır. Pratikte pek kullanılmaz, çünkü burada iki tane 0 söz konusudur (0000 0000 ve 1000 0000) ve işlemcinin doğrudan toplamasıyla çıkarma elde edilemez.

Sayı Sistemleri

İkiye Tümlenen (2's Complement)

İkiye tümlenen sayıda tek bir 0 vardır. Sayılar $-2^{\text{bitsayısı}-1} \dots 0 \dots (2^{\text{bitsayısı}-1} - 1)$ arasında değişir. Örneğin 8 bit için $-128 \dots 0 \dots 127$.

İkiye tümlenen sayı aşağıdaki şekilde elde edilir.

$$2^{\text{bitsayısı}} - \text{sayı}$$

Örnek : -20 $-(0001\ 0100)$ sayısını 8 bitlik ikili sayı ile ifade edelim
 $2^8 - 20 = 256 - 20 = 236 = 1110\ 1100$

Pratik yöntem sayıyı evirip 1 eklemektir.

0001 0100

$$1110\ 1011 + 1 = 1110\ 1100$$

Çıkarma işlemi

Örneğin $40 - 20$ yi 2'ye tümlenenle hesaplayalım

$$\begin{array}{r} 0010\ 1000 \\ + 1110\ 1100 \\ \hline \end{array}$$

$$0001\ 0100 = (20 \text{ dec})$$

Sayı Sistemleri

Hexadecimal sayılar (Hex)

Bilgisayar sistemlerinde uzun bit dizilerini temsil etmek zor olacağı için yazım biçimi olarak hexadecimal sayılar tercih edilmektedir. Hex sayılar 16 lık sayılardır. Herbir Nibble bir Hex sayı ile temsil edilebilir. Böylelikle ikili sayının yazım uzunluğu 4 te bir digite düşecektir.

Hex sistemde sayılar 16 sembolden oluşur ve aşağıdaki gibidir.

0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Örnek: 0011 1010 = 3A Hex, 1110 0101 = E5 Hex
0101 1101 1100 1001 0110 0111 =5DC967 Hex



Kod Sistemleri

Bilgisayarlar yalnızca sayılarla çalışırlar, oysa bizim harflere ve diğer sembollere de gereksinimimiz vardır. Bu semboller de sayılara karşılık düşürülecek biçimde kodlanırlar. Program örneğin bu sayı ile karşılaşırsa ekrana karşılık düşen sembolü basar, yada klavyeden gelen sayının sembolik karşılığını , yazıcıdan çıkarır.

Bir çok kodlama türü olmasına karşın dünyada bilgisayar ortamlarında ANSI tarafından 1963 yılında standartlaştırılan **ASCII**(**A**merican **N**ational**C**ode for **I**nformation **I**nterchange) kodlaması yoğun olarak kullanılmaktadır. Ancak günümüzde , ASCII kodları çok dilliği sağlayabilmek için yetersiz kaldığından UNICODE kodlaması yaygınlaşmaktadır. Ancak pek çok uygulamada ASCII kodlaması hala geçerliliğini korumaktadır.

ASCII temel olarak 7 bit' tir. 127 karakterden oluşur. Ama Extended kısmıyla birlikte 8 bit kullanılmaktadır. Ancak genişletilmiş kısımdaki semboller yazılım ortamına göre değişebilmektedir.

ASCII ilk 128 Sembol

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ASCII genişletilmiş kısım

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ù	161	A1	í	193	C1	ł	225	E1	β
130	82	é	162	A2	ó	194	C2	Ť	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ł	227	E3	Π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	†	229	E5	σ
134	86	å	166	A6	ª	198	C6	‡	230	E6	μ
135	87	ç	167	A7	º	199	C7	‡	231	E7	τ
136	88	ê	168	A8	¿	200	C8	‡	232	E8	φ
137	89	ë	169	A9	ƒ	201	C9	ƒ	233	E9	θ
138	8A	è	170	AA	¬	202	CA	‡	234	EA	Ω
139	8B	ì	171	AB	½	203	CB	‡	235	EB	ϑ
140	8C	í	172	AC	¼	204	CC	‡	236	EC	∞
141	8D	î	173	AD	¡	205	CD	=	237	ED	∞
142	8E	Ë	174	AE	«	206	CE	‡	238	EE	τ
143	8F	Ä	175	AF	»	207	CF	±	239	EF	∩
144	90	É	176	B0	⋄	208	D0	‡	240	FO	≡
145	91	æ	177	B1	⋄	209	D1	‡	241	F1	±
146	92	Æ	178	B2	⋄	210	D2	‡	242	F2	≥
147	93	ó	179	B3		211	D3	‡	243	F3	≤
148	94	ö	180	B4	†	212	D4	‡	244	F4	[
149	95	ò	181	B5	†	213	D5	ƒ	245	F5]
150	96	û	182	B6	‡	214	D6	ƒ	246	F6	÷
151	97	ù	183	B7	‡	215	D7	‡	247	F7	∞
152	98	ÿ	184	B8	‡	216	D8	‡	248	F8	°
153	99	Ö	185	B9	‡	217	D9	‡	249	F9	•
154	9A	Û	186	BA	‡	218	DA	ƒ	250	FA	·
155	9B	◊	187	BB	‡	219	DB	■	251	FB	√
156	9C	£	188	BC	‡	220	DC	■	252	FC	∂
157	9D	¥	189	BD	‡	221	DD	■	253	FD	ε
158	9E	℔	190	BE	‡	222	DE	■	254	FE	■
159	9F	f	191	BF	‡	223	DF	■	255	FF	□



İşlemler

Bilgisayar programları ile gerçekleştirilen işlemler;

- 1) Matematiksel İşlemler
- 2) Karşılaştırma(karar) İşlemleri
- 3) Mantıksal(lojik) İşlemler

- Matematiksel İşlemler
 - Temel aritmetik işlemler
 - toplama, çıkarma, çarpma, bölme
 - Matematiksel fonksiyonlar
 - Üstel, logaritmik, trigonometrik, hiperbolik) vb

Matematiksel İşlemler

<i>İşlem</i>	<i>Matematik</i>	<i>Bilgisayar</i>
<i>Toplama</i>	$a+b$	$a+b$
<i>Çıkarma</i>	$a-b$	$a-b$
<i>Çarpma</i>	$a.b$	$a*b$
<i>Bölme</i>	$a:b$	a/b
<i>Üs alma</i>	a^b	a^b

Matematiksel işlemlerin öncelik sırası ?

<i>Sıra</i>	<i>İşlem</i>	<i>Bilgisayar dili</i>
1	<i>Parantezler</i>	$((.....))$
2	<i>Üs alma</i>	a^b
3	<i>Çarpma ve bölme</i>	$a*b$ ve a/b
4	<i>Toplama ve çıkarma</i>	$a+b$ ve $a-b$

NOT: Bilgisayar diline kodlanmış bir matematiksel ifade, aynı önceliğe sahip işlemler mevcut ise bilgisayarın bu işlemleri gerçekleştirme sırası soldan sağa(baştan sona) doğrudur.

Örneğin ; $Y=A*B/C$
Önce $A*B$ işlemi yapılacak, ardından bulunan sonuç C ye bölünecektir.

Matematiksel İşlemler

Matematiksel Yazılım	Bilgisayara Kodlanması
$a+b-c+2abc-7$	$a+b-c+2*a*b*c-7$
$a+b^2-c^3$	$a+b^2-c^3$
$a - \frac{b}{c} + 2ac - \frac{2}{a+b}$	$a-b/c+2*a*c-2/(a+b)$
$\sqrt{a+b} - \frac{2ab}{b^2-4ac}$	$(a+b)^{(1/2)}-2*a*b/(b^2-4*a*c)$
$\frac{a^2+b^2}{2ab}$	$(a^2+b^2)/(2*a*b)$

Karşılaştırma (Karar) İşlemleri

- İki büyüklükten hangisinin büyük veya küçük olduğu,
- İki değişkenin birbirine eşit olup olmadığı gibi konularda karar verebilir.

<i>İşlem sembolü</i>	<i>Anlamı</i>
=	<i>Eşittir</i>
<>	<i>Eşit değil</i>
>	<i>Büyükdür</i>
<	<i>Küçüktür</i>
<i>>= veya =></i>	<i>Büyük eşittir</i>
<i><= veya =<</i>	<i>Küçük eşittir</i>

Mantıksal İşlemler

<i>Mantıksal işlem</i>	<i>Matematiksel sembol</i>	<i>Komut</i>
<i>Ve</i>	.	<i>And</i>
<i>Veya</i>	+	<i>Or</i>
<i>değil</i>	'	<i>Not</i>

“ve,veya,değil “ operatörleri hem matematiksel işlemlerde hem de karar ifadelerinde kullanılırlar.

- Bütün şartların sağlatılması isteniyorsa koşullar arasına VE
- Herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA
- Koşulu sağlamayanlar isteniyorsa DEĞİL
mantıksal operatörü kullanılır.

Mantıksal İşlemler

Örnek; Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.

Burada iki koşul vardır ve bu iki koşulun da doğru olması gerekir. Yani;

Eğer $\underbrace{\text{Yaş} > 23}_{1. \text{KOŞUL}}$ VE $\underbrace{\text{maaş} = \text{asgari ücret ise}}_{2. \text{KOŞUL}}$ ismi Yaz

Yaz komutu 1. ve 2.koşulun her ikisi de sağlanıyorsa çalışır.

Örnek; Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65 in üzerinde not alanların isimleri istenmektedir.

Burada 3 koşul vardır ve Bilgisayar dersinden 65 in üzerinde not almış olmak temel koşuldur. Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gerekir.

Eğer ;

Bilg.Not>65 **ve** (TDili not>65 **veya** YDil not>65) ismi Yaz

2. BÖLÜM

Problem Çözme ve Algoritmalar

Problem Çözme

Problem Çözme Tekniđi (Descartes'e göre):

1. Doğruluđu kesin olarak kanıtlanmadıkça, hiçbir şeyi doğru olarak kabul etmeyin; tahmin ve önyargılardan kaçının.
2. Karşılaştığınız her güçlüđu mümkün olduđu kadar çok parçaya bölün.
3. Düzenli bir biçimde düşünün; anlaşılması en kolay olan şeylerle başlayıp yavaş yavaş daha zor ve karmaşık olanlara doğru ilerleyin.
4. Olaya bakışınız çok genel, hazırladığınız ayrıntılı liste ise hiçbir şeyi dışarıda bırakmayacak kadar kusursuz ve eksiksiz olsun.

Problem çözme sırası

- 1. Problemi anlama** (Understanding, Analyzing),
- 2. Bir çözüm yolu geliştirme** (Designing),
- 3. Algoritma ve program yazma** (Writing),
- 4. Tekrar tekrar test etme** (Reviewing)

Problem Çözme

Bir problemi çözmek için yazılacak programda, genel olarak, aşağıdaki yazılım geliştirme aşamaları uygulanmalıdır.

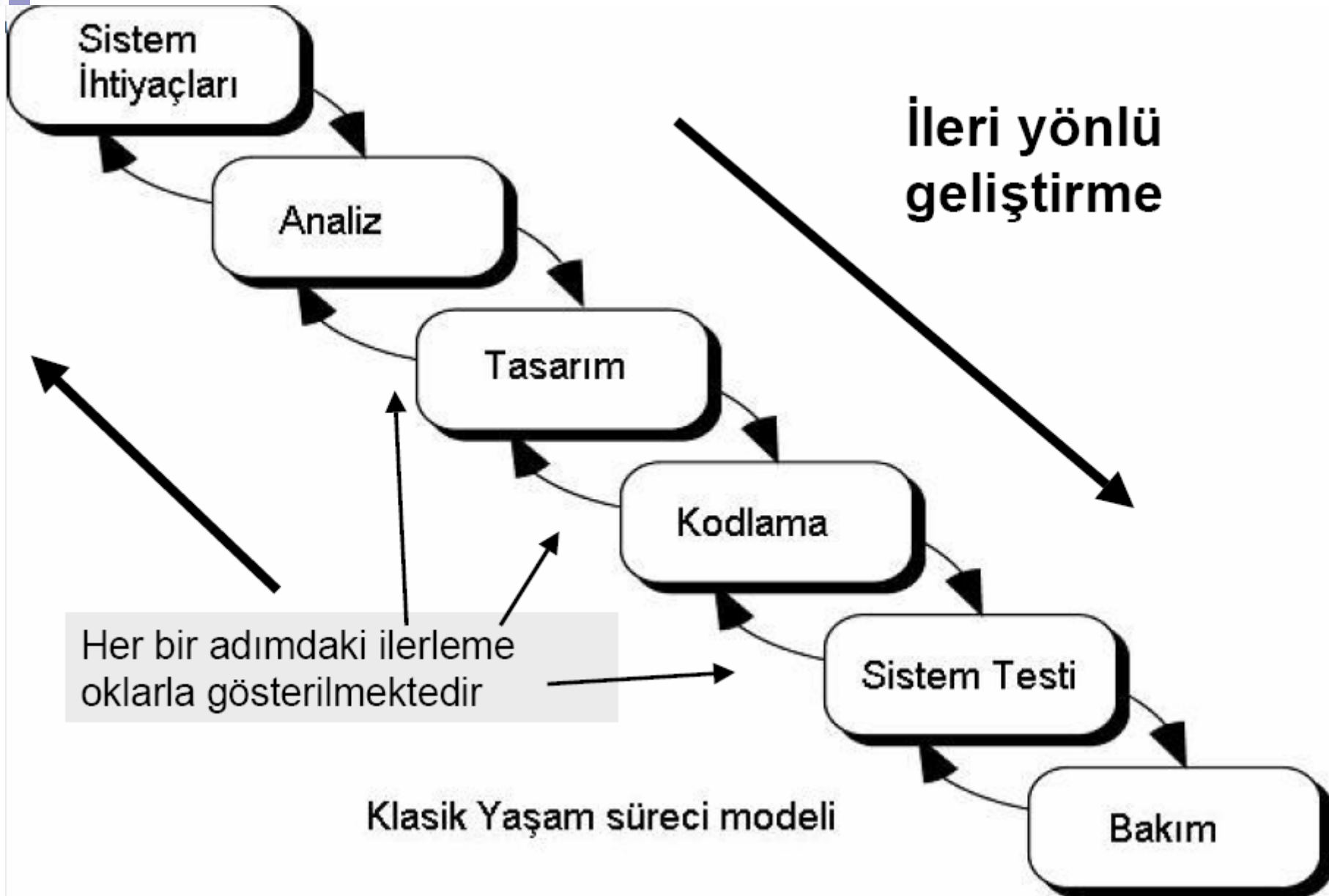
Yazılım Geliştirme Aşamaları

1. Problemin Analizi: Problemin tam olarak ne olduğunun anlaşılmasıdır. Bu nedenle, problemin çözümünden neler beklendiği ve yaratacağı çözümün girdi ve çıktılarının neler olacağı kesin olarak belirlenmelidir.
2. Tasarım: Problemi çözmek için kullanılacak çözüm adımlarını gösteren bir liste yapılması gereklidir. Bir problemin çözüm adımlarını gösteren bu listeye algoritma denir. Böyle bir liste tasarlanırken, ilk önce problemin ana adımları çıkarılır; daha sonra her adım için, gerekiyorsa, daha ayrıntılı bir çözüm tasarlanır.
3. Kodlama: Kağıt üzerinde geliştirilen algoritma, programcının tercih ettiği bir programlama diline çevrilir.



Problem Çözme

4. Test etme: Program değişik girdiler ile çalıştırılarak ürettiği sonuçlar kontrol edilerek test işlemi gerçekleştirilir. Sonuçlar beklendiği gibi ise , programın doğru çalıştığı kanıtlanmış olunur; değilse doğru çalışmayan kısımları tespit edilerek düzeltilir.
5. Belgeleme: Bütün bu çalışmaların belli bir dosyalama sistemi içinde belgeler halinde saklanmasının sağlandığı aşamadır.
6. Bakım: Programın güncel koşullara göre yeniden düzenlenmesini içeren bir konudur. Oluşan hataların giderilmesi,, yeni eklemeler yapılması ya da programın teknolojisinin yenilenmesi gibi işlemlerdir.





Problem Çözme

Bir problem çözülrken biri algoritmik, diğeri herustic(sezgisel) olarak adlandırılan iki yaklaşım vardır.

Algoritmik yaklaşımda, çözüm için olası yöntemlerden en uygun olanı seçilir ve yapılması gerekenler adım adım ortaya konulur.

Herustic yaklaşımda ise, çözüm açıkça ortada değildir. Tasarımcının deneyimi, birikimi ve o andaki düşüncesine göre problemi çözecek bir şeylerin şekillendirilmesiyle yapılır. Program tasarımcısı, algoritmik yaklaşımla çözemediği, ancak çözmek zorunda olduğu problemler için bu yaklaşımı kullanır.

Algoritmik Yaklaşım

Algoritma, herhangi bir sorunun çözümü için izlenecek yol anlamına gelmektedir. Çözüm için yapılması gereken işlemler hiçbir alternatif yoruma izin vermeksizin sözel olarak ifade edilir. Diğer bir deyişle algoritma, verilerin, bilgisayara hangi çevre biriminden girileceğinin, problemin nasıl çözüleceğinin, hangi basamaklardan geçirilerek sonuç alınacağıının, sonucun nasıl ve nereye yazılacağıının sözel olarak ifade edilmesi biçiminde tanımlanabilir.

Algoritma hazırlanırken, çözüm için yapılması gerekli işlemler, öncelik sıraları gözönünde bulundurularak ayrıntılı bir biçimde tanımlanmalıdırlar.

Örnek 1: Verilen iki sayının toplamının bulunmasının algoritması aşağıdaki gibi yazılır.

Adım 1 – Başla

Adım 2 – Birinci Sayıyı Oku

Adım 3 – İkinci Sayıyı Oku

Adım 4 – İki Sayıyı Topla

Adım 5 – Dur



Algoritmik Yaklaşım

Algoritmalar iki farklı şekilde kağıt üzerinde ifade edilebilirler;

1. Pseudo Code (*Kaba Kod veya Yalancı Kod veya Sözde Kod*), bir algoritmanın yarı programlama dili kuralı, yarı konuşma diline dönük olarak ortaya koyulması/ tanımlanmasıdır. Bu şekilde gösterim algoritmayı genel hatlarıyla yansıtır.
2. Akış şeması, algoritmanın görsel/şekilsel olarak ortaya koyulmasıdır. Problemin çözümü için yapılması gerekenleri, başından sonuna kadar, geometrik şekillerden oluşan simgelerle gösterir.

Sözde (Pseudo) Kod

Sözde programlar, doğrudan **konuşma dilinde** ve programlama mantığı altında, **eğer, iken** gibi koşul kelimeleri ve **> = <** gibi ifadeler ile beraber yazılır. İyi bir biçimde yazılmış, sözde koddan, programlama diline kolaylıkla geçilebilir.

Örnek: Verilen bir sıcaklık derecesine göre suyun durumunu belirten bir sözde program yazınız.

•Örnek Giriş/Çıkış

–Bu Program, Sıcaklığa göre suyun durumunu gösterir

–Su, Buz, Buhar

–Lütfen derece cinsinden sıcaklığı giriniz: 140

–BUHAR elde edeceksiniz.



Sözde (Pseudo) Kod

İstenilen programın Pseudo Kodu:

1. Program açıklama mesajı yaz.
2. Kullanıcın sıcaklığı girmesi için bir uyarı mesajı yaz.
3. Girilen Sıcaklığı Oku.
5. Eğer Sıcaklık < 0 ise Durum="Buz"
6. Eğer Sıcaklık ≥ 100 ise Durum="Buhar"
7. Değilse Durum = "Su"
8. Sonucu Yaz.

Akış Diyagramları (Şemaları)

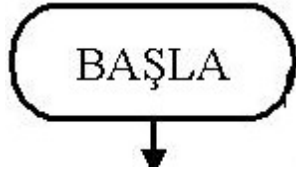
Algoritmanın, görsel olarak simge ya da sembollerle ifade edilmiş şekline “akış şemaları” veya FLOWCHART adı verilir. Akış şemalarının algoritmadan farkı, adımların simgeler şeklinde kutular içine yazılmış olması ve adımlar arasındaki ilişkilerin ve yönünün oklar ile gösterilmesidir.

Programın saklanacak esas belgeleri olan akış şemalarının hazırlanmasına, sorun çözümlenmesi sürecinin daha kolay anlaşılır biçime getirilmesi, iş akışının kontrol edilmesi ve programın kodlanmasının kolaylaştırılması gibi nedenlerle başvurulur. Uygulamada çoğunlukla, yazılacak programlar için önce programın ana adımlarını (bölümlerini) gösteren genel bir bakış akış şeması hazırlanır. Daha sonra her adım için ayrıntılı akış şemalarının çizimi vardır.

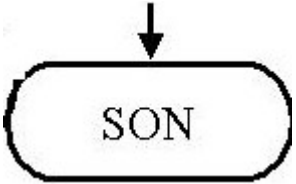
En basit şekliyle dikdörtgen kutulardan ve oklardan oluşur. Akış şeması sembolleri ANSI (American National Standards Institute) standardı olarak belirlenmiş ve tüm dünyada kullanılmaktadır.

Akış Diyagramları (Şemaları)

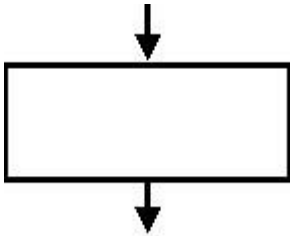
Her simge, yapılacak bir işi veya komutu gösterir. Akış şemalarının hazırlanmasında aşağıda yer alan simgeler kullanılır.



Bir algoritmanın başladığı konumu göstermektedir. Tek çıkışlı bir şekildir.

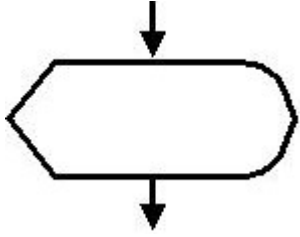


Bir algoritmanın bittiği konumu göstermektedir. Tek girişli bir şekildir.

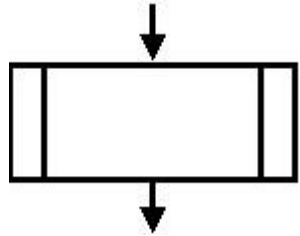


Bir algorithmanda aritmetik işlem yapılmasını sağlayan şekildir. Bu dörtgen kutu içerisine yapılmak istenen işlem yazılır. Tek girişli ve tek çıkışlı bir şekildir.

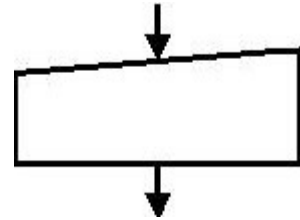
Akış Diyagramları (Şemaları)



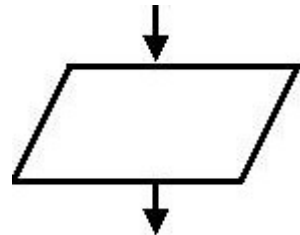
Algoritmada bir bilginin ekrana yazılacağı konumu gösteren şekildir. Ekrana yazılacak ifade ya da değişken bu şekil içerisine yazılır.



Bir algoritmada başka bir yerde tanımlanmış bloğun yerleştiği konumu gösteren şekildir. Kutu içerisine bloğun adı yazılabilir.

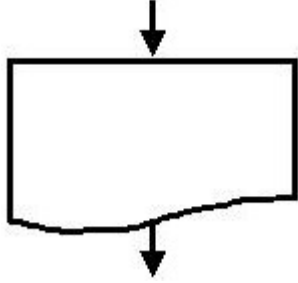


Klavyeden Bilgisayara bilgi girilecek konumu belirten şekildir. Girilecek bilginin hangi değişkene okunacağını kutu içerisine yazabilirsiniz.

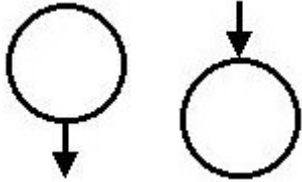


Giriş Çıkış komutunun kullanılacağı yeri belirler ve kutu içerisine hangi değişkeni ve OKUma mı YAZ mı yapılacağını belirtmeniz gerekir.

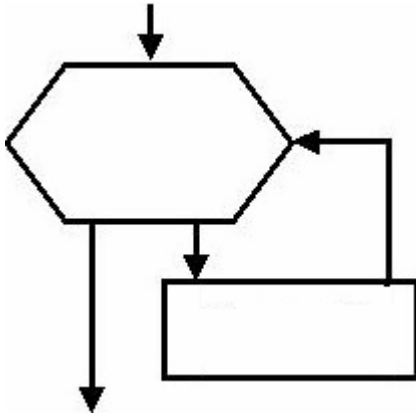
Akış Diyagramları (Şemaları)



Bilginin Yazıcıya yazılacağı konumu gösteren şekildir.

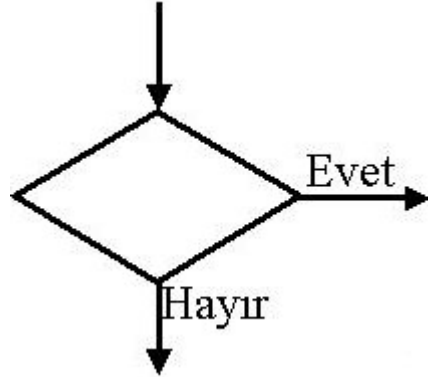


Bir algoritmanın birden fazla alana yayılması durumunda bağlantı noktalarını gösteren şekildir. Tek girişli veya tek çıkışlı olarak kullanılırlar.



Birçok programda; belirli işlem blokları ardışık değerlerle veya bazı koşullar sağlanıncaya kadar tekrarlanır. Bu tekrarlamalı işlemler döngü olarak isimlendirilir. Döngü şeklinin içine; döngü (çevrim, kontrol) değişkeni, başlangıç değeri, bitiş değeri ve adımı yazılır.

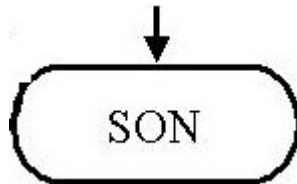
Akış Diyagramları (Şemaları)



Bir alıgoritmada bir kararın verilmesini ve bu karara göre iki seenekten birinin uygulanmasını saėlayan şekildir. burada eşkenar dörtgen ierisine kontrol edilecek mantıksal koşul yazılır. Program akışı sırasında koşulun doėru olması durumunda "Evet" yazılan kısma Yanlıř olması durumunda "Hayır" yazılan kısma sapılır. Tek giriřli ve ift ıkıřlı bir şekildir.



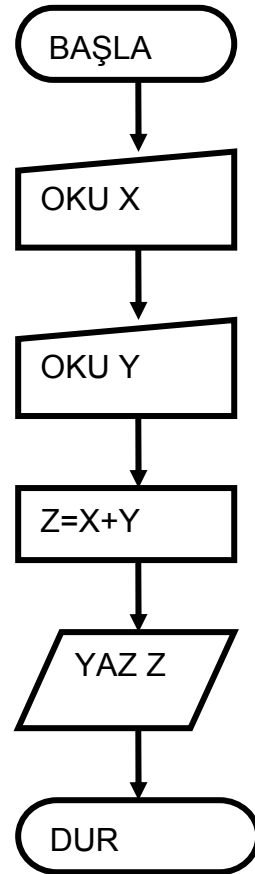
Akış ubuėu



Programın bittiėi yer ya da yerleri gsteren bir şekildir.

Akış Diyagramları (Şemaları)

Klavyeden girilen İki sayının toplamını hesaplayıp yazan pseudo kod ve akış şemasını hazırlayınız.



(X: Birinci sayı, Y: İkinci sayı, Z: toplam)

A1 : Başla

A2 : Klavyeden oku X

A3 : Klavyeden oku Y

A4 : Hesapla $Z = X + Y$

A5 : Yaz Z

A6 : Dur

Algoritmelerde Kullanılan Operatörler

- İşlemleri belirten sembollere bilgisayar dilinde “operatör” denir.
- Algoritmada kullanılan operatörler Tabloda verilmiştir.

<i>Matematiksel İşlem oper</i>	
<i>Üs alma</i>	\wedge
<i>Çarpma</i>	$*$
<i>Bölme</i>	$/$
<i>Toplama</i>	$+$
<i>Çıkarma</i>	$-$
<i>Tam ve onda ayırma</i>	$.$
<i>Mantıksal İşlem Operatörleri</i>	
<i>Değil</i>	$'$
<i>Ve</i>	$.$
<i>Veya</i>	$+$

<i>Karşılaştırma Operatörleri</i>	
<i>Eşittir</i>	$=$
<i>Eşit değildir</i>	$<>$
<i>Küçüktür</i>	$<$
<i>Büyükür</i>	$>$
<i>Büyük eşittir</i>	$>=$
<i>Küçük eşittir</i>	$<=$
<i>Genel İşlem Operatörleri</i>	
<i>Aktarma</i>	$=$
<i>Parantez</i>	$()$

Algoritmalarda Kullanılan Terimler

- Tanımlayıcı
- Değişken
- Sabit
- Aktarma
- Sayaç
- Döngü
- Ardışık Toplama
- Ardışık Çarpma

Tanımlayıcı

- Programcı tarafından oluşturulur.
- Programdaki değişkenleri,sabitleri, kayıt alanlarını, özel bilgi tiplerini vb adlandırmak için kullanılan kelimeler
- Tanımlayıcılar, yerini tuttıkları ifadelere çağrışım yapacak şekilde seçilmelidir.
- İngiliz alfabesindeki A-Z veya a-z arası 26 harften
- 0-9 arası rakamlar kullanılabilir
- Sembollerden sadece alt çizgi (_) kullanılabilir.
- Tanımlayıcı isimleri harfle veya alt çizgiyle başlayabilir.
- Tanımlayıcı ismi,rakamla başlayamaz veya sadece rakamlardan oluşamaz.

Algoritmalarda Kullanılan Terimler

Değişken

- Programın her çalıştırılmasında, farklı değerler alan bilgi/bellek alanlarıdır.
- Değişken isimlendirilmeleri, yukarıda sayılan tanımlayıcı kurallarına uygun biçimde yapılmalıdır.

Örneğin ;

Bir ismin aktarıldığı değişken ; **ad**

Bir isim ve soy ismin aktarıldığı değişken; **adsoyad**

Ev telefon no sunun aktarıldığı değişken; **evtel**

Ev adresinin aktarıldığı değişken; **evadres**

İş adresinin aktarıldığı değişken; **isadres**

Sabit

Programdaki değeri değişmeyen ifadelere “sabit” denir. “İsimlendirme kuralları”na uygun olarak oluşturulan sabitlere, sayısal veriler doğrudan; alfa sayısal veriler ise tek/çift tırnak içinde aktarılır.

Algoritmalarda Kullanılan Terimler

Örnek Algoritma

Başla

Bir isim giriniz(A)

“İlk algoritmama Hoş geldin” mesajı (B)

B VE A yı Yaz

Dur

- Yukarıdaki algoritma, dışarıdan girilen bir A değişkeni ile B sabitini birleştirip ekrana yazar.

<i>A değişkeni</i>	<i>B sabiti</i>	<i>Sonuç</i>
Onur	İlk Algoritmama Hoş geldin	İlk Algoritmama Hoş geldin Onur
Emre	İlk Algoritmama Hoş geldin	İlk Algoritmama Hoş geldin Emre

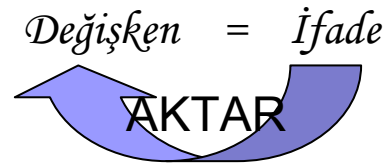
Algoritmalarda Kullanılan Terimler

Aktarma

- Herhangi bir bilgi alanına, veri yazma; herhangi bir ifadenin sonucunu başka bir değişkende gösterme vb görevlerde “aktarma” operatörü kullanılır.

değişken=ifade

- Değişken yazan kısım herhangi bir değişken ismidir.
- İfade yazan kısımda ise matematiksel, mantıksal veya alfa sayısal ifade olabilir.
- = sembolü, aktarma operatörüdür ve sağdaki ifadenin/işlemin sonucunu soldaki değişkene aktarır. Bu durumda değişkenin eğer varsa bir önceki değeri silinir.



1.işlem: sağdaki ifadeyi gerçekleştir veya sağdaki işlemi yap

2.işlem: Bulunan sonucu soldaki değişkene aktar.

Algoritmalarda Kullanılan Terimler

Sayaç

Programlarda bazı işlemlerin belirli sayıda yaptırılması veya işlenen/üretilen değerlerin sayılması gerekebilir.

say=say+1

Bu işlemde sağdaki ifadede değişkenin eski değerine 1 eklenmekte; bulunan sonuç yine kendisine yeni değer olarak aktarılmaktadır.

Bu tür sayma işlemlerine algoritmada sayaç adı verilir.

Sayacın genel formülü;

Sayaç değişkeni=sayaç değişkeni+adım

Örnek; $X=X+3$

Örnek; $S=S-5$

Algoritmalarda Kullanılan Terimler

Örnek

- Aşağıdaki algorithmada 1-5 arası sayılar, ekrana yazdırılmaktadır. 1-5 arası sayıları oluşturmak için sayaç($s=s+1$) kullanılmıştır.

1. **Başla**
2. **S=0**
3. **Eğer $s>4$ ise git 7**
4. **S=S+1**
5. **Yaz S**
6. **Git 3**
7. **Dur**

Eski S	Yeni S	Ekrana Yazılan
0	$0+1=1$	1
1	$1+1=2$	2
2	$2+1=3$	3
3	$3+1=4$	4
4	$4+1=5$	5

Algoritmalarda Kullanılan Terimler

Döngü

- Bir çok programda bazı işlemler, belirli ardışık değerlerle gerçekleştirilmekte veya belirli sayıda yaptırılmaktadır. Programlardaki belirli işlem bloklarını, verilen sayıda gerçekleştiren işlem akış çevrimlerine “döngü” denir.
- Örneğin 1 ile 1000 arasındaki tek sayıların toplamını hesaplayan programda $T=1+3+5 \dots$ yerine 1 ile 1000 arasında ikişer artan bir döngü açılır ve döngü değişkeni ardışık toplanır.

Algoritmalarda Kullanılan Terimler

Örnek

Aşağıdaki algoritmada, 1 ile 10 arası tek sayıların toplamı hesaplanmaktadır.

1. **Başla**

2. **T=0**

3. **J=1**

4. **Eğer $j > 10$ ise git 8**

5. **T=T+J**

6. **J=J+2**

7. **Git4**

8. **Dur**

DÖNGÜ

Eski J	Eski T	Yeni T	Yeni J
1	0	0+1=1	3
3	1	1+3=4	5
5	4	4+5=9	7
7	9	9+7=16	9
9	16	16+9=25	11
11	-	-	-

Algoritmalarda Kullanılan Terimler

Ardışık Toplama

Programlarda, aynı değerin üzerine yeni değerler eklemek için kullanılır.

$$\text{Toplam değışkeni} = \text{Toplam değışkeni} + \text{Sayı}$$

Örnek: Klavyeden girilen 5 sayısının ortalamasını bulan programın algoritması.

1. Başla
2. $T=0$
3. $S=0$
4. Eğer $S>4$ ise git 9
5. $S=S+1$
6. Sayıyı (A) gir
7. $T=T+A$
8. Git 4
9. $\text{Ortalama} = T/5$
10. Yaz Ortalama
11. Dur

Algoritmalarda Kullanılan Terimler

Ardışık Çarpma

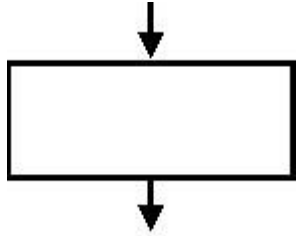
Ardışık veya ardışıl çarpma işleminde; aynı değer, yeni değerlerle çarpılarak eskisinin üzerine aktarılmaktadır.

$$\text{Çarpım değişkeni} = \text{Çarpım değişkeni} * \text{Sayı}$$

Örnek: Klavyeden girilen N sayısının faktöriyelini hesaplayan programın algoritmasını yazınız.

1. Başla
2. N sayısını gir
3. Fak=1
4. S=0
5. Eğer $S > N-1$ ise git 9
6. $S=S+1$
7. $Fak=Fak*S$
8. Git 5
9. Yaz Fak
10. Dur

Akış Diyagramlarında Kullanılan Temel Şekiller



Programın çalışması sırasında yapılacak işlemleri ifade etmek için kullanılan şekildir. İçine işlem cümleleri/ifadeleri aynen yazılır. Program akışı buraya geldiğinde, şeklin içerisindeki yazılı işlem gerçekleştirilir. Birden fazla işlem; aynı şekil içinde, aralarına virgül konularak veya alt alta yazılarak gösterilebilir.

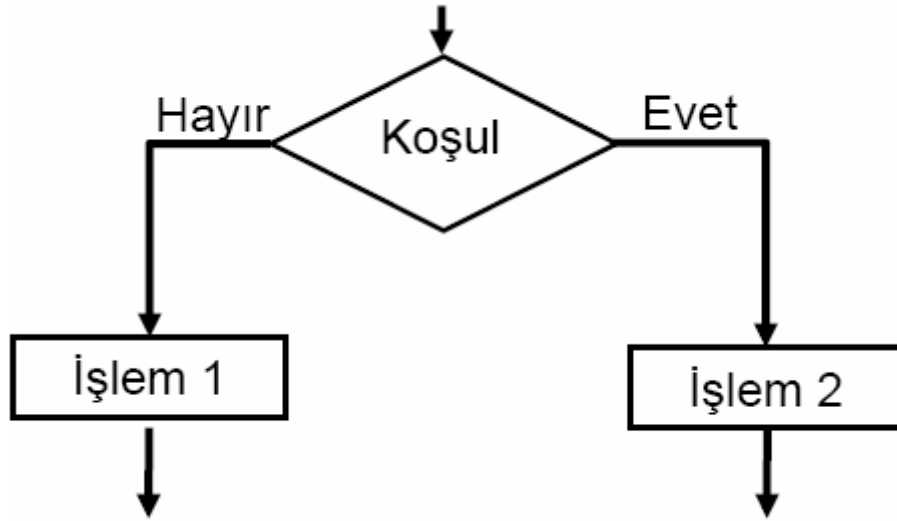
$C=(a^2+b^2)^{(1/2)}$

Örnek: İşlem akışı bu şekle gelince, program $c = \sqrt{a^2 + b^2}$ işlemini yapar. İfadedeki 'a' ve 'b' daha önceki adımlarda girilmiş olan değerlerdir.

Akış Diyagramlarında Kullanılan Temel Şekiller

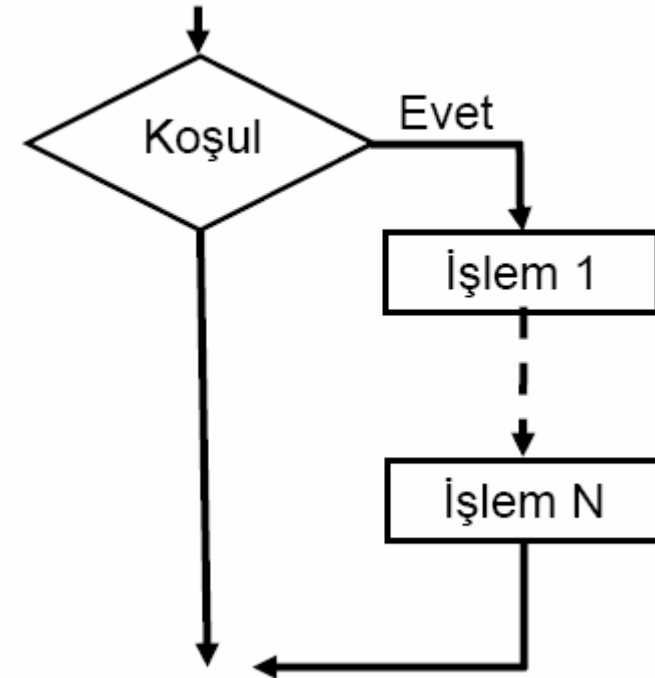
Karar Verme

1. Durum:



a) Koşulun durumuna bağlı olarak 2 farklı işlem vardır.

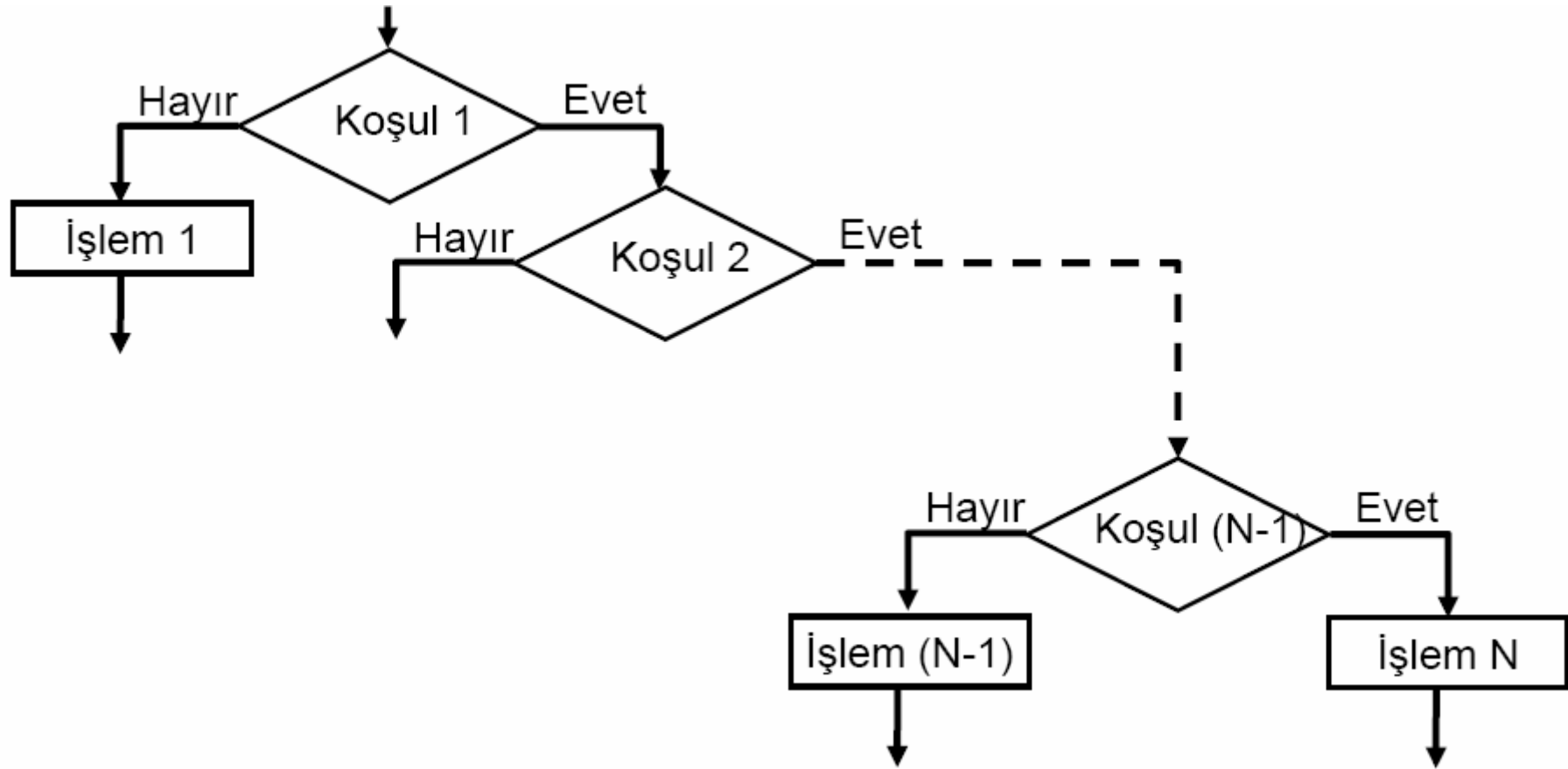
2. Durum:



b) Olumsuz koşulda yapılacak işlem yoktur; olumlu olması durumunda ise N adet işlem yapılacaktır.

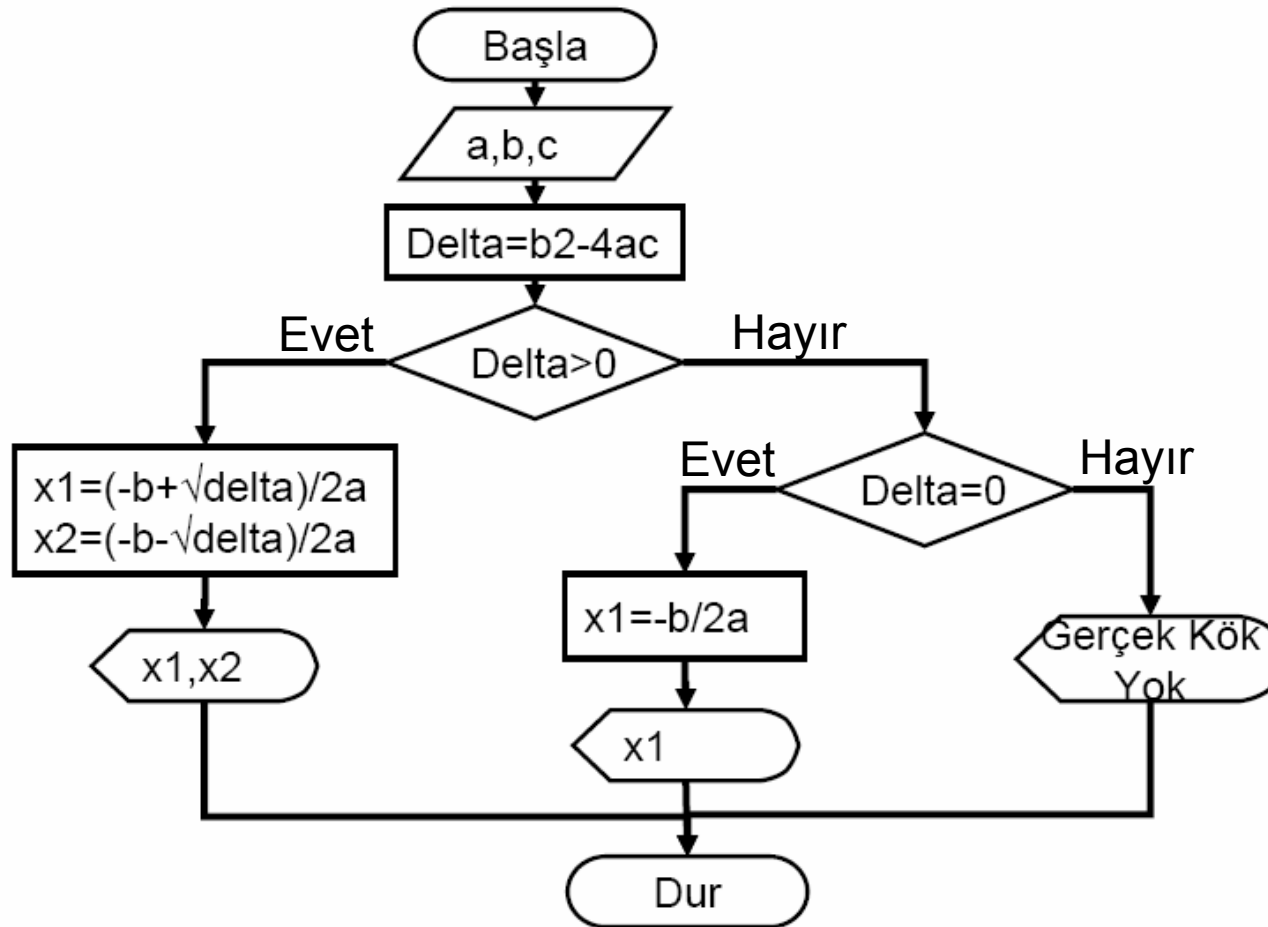
Akış Diyagramlarında Kullanılan Temel Şekiller

Bu yapıyı art arda birden çok kez kullanıp aşağıdaki gibi bir kaşılaştırma dizisi oluşturulabilir.



Akış Diyagramlarında Kullanılan Temel Şekiller

Örnek: $ax^2 + bx + c = 0$ şeklindeki ikinci dereceden bir denklemin köklerini bulan algoritmayı tasarlayıp akış şeması ile gösteriniz.





Akış Diyagramlarında Kullanılan Temel Şekiller

Döngü Yapısı

Bu yapı kullanılırken, döngü sayacı, koşul bilgisi ve sayacın artım bilgisi verilmelidir. Döngü sayacı kullanılmıyorsa sadece döngüye devam edebilmek için gerekli olan koşul bilgisi verilmelidir.

Genel olarak çoğu programlama dilinin döngü deyimleri ;

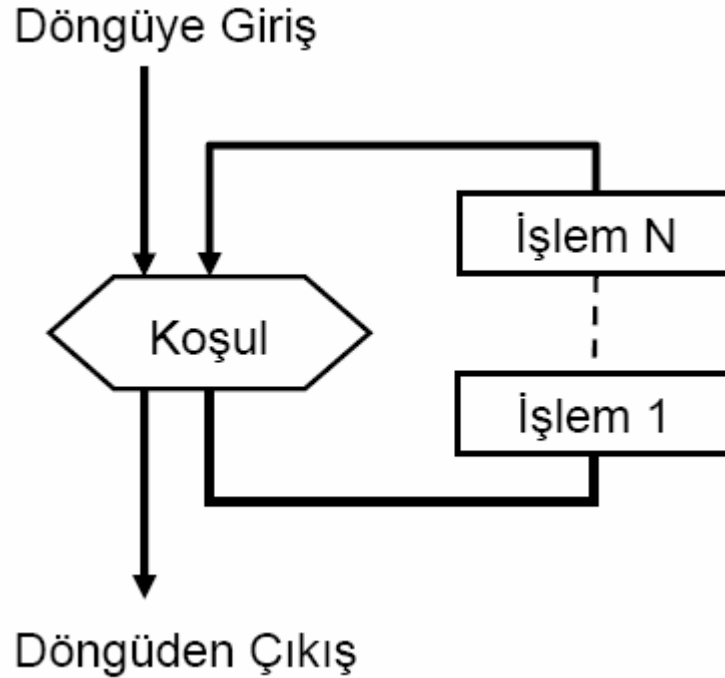
- While
- Do-while
- For

gibi yapılar üzerine kurulmuştur. Farklı dillerde bu yapılara farklı alternatifler olsa da döngülerin çalışma mantığı genel olarak benzerdir.

Akış Diyagramlarında Kullanılan Temel Şekiller

1. Durum (While)

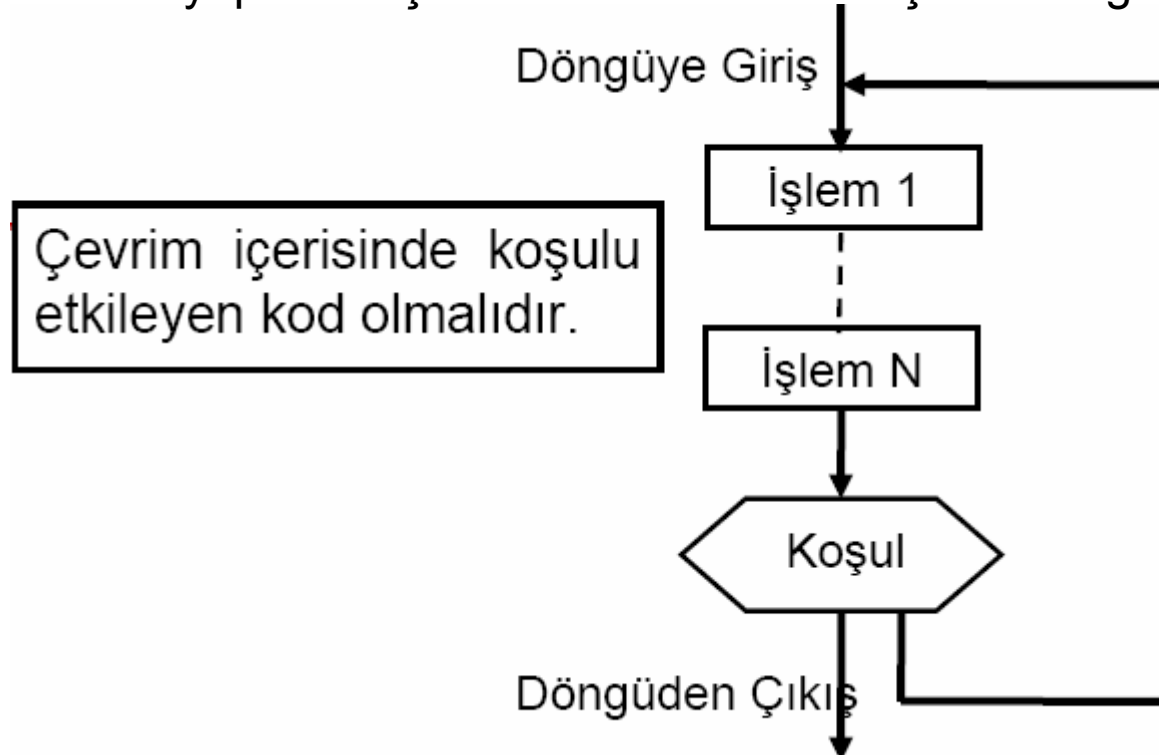
Koşul daha çevrim içerisine girmeden sınanır. Koşul olumsuz olduğunda çerime hiç girilmez ve döngü içerisinde yapılması gerekenler atlanır.



Akış Diyagramlarında Kullanılan Temel Şekiller

2. Durum (Do-While)

Bu döngü deyiminde, çevrim en az bir defa olmak üzere gerçekleşir. Çünkü koşul sınaması döngü sonunda yapılmaktadır. Eğer koşul sonucu olumsuz ise bir sonraki çevrime geçilmeden döngüden çıkılır. Çevrimin devam edebilmesi için her döngü sonunda yapılan koşul testinin olumlu sonuçlanması gerekir.

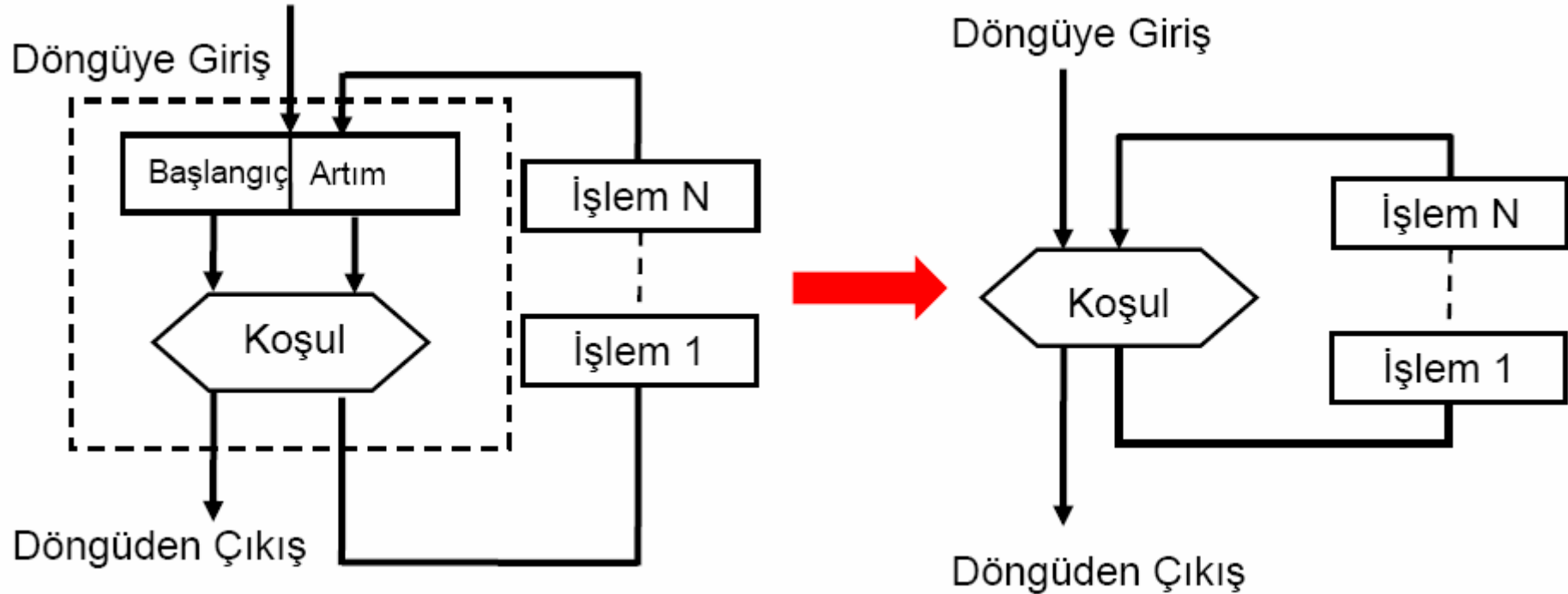


Akış Diyagramlarında Kullanılan Temel Şekiller

3. Durum (For)

Diğer deyimlerden farklı olarak, döngü sayacı doğrudan koşul parametreleri düzeyinde verilir.

Döngü girmeden önce sayaç değişkenine başlangıç değeri atanmakta ve daha sonra koşula bakılmaktadır. Döngü içerisinde belirtilen işlemler yapıldıktan sonra sayaç değişkeni artırılmaktadır.

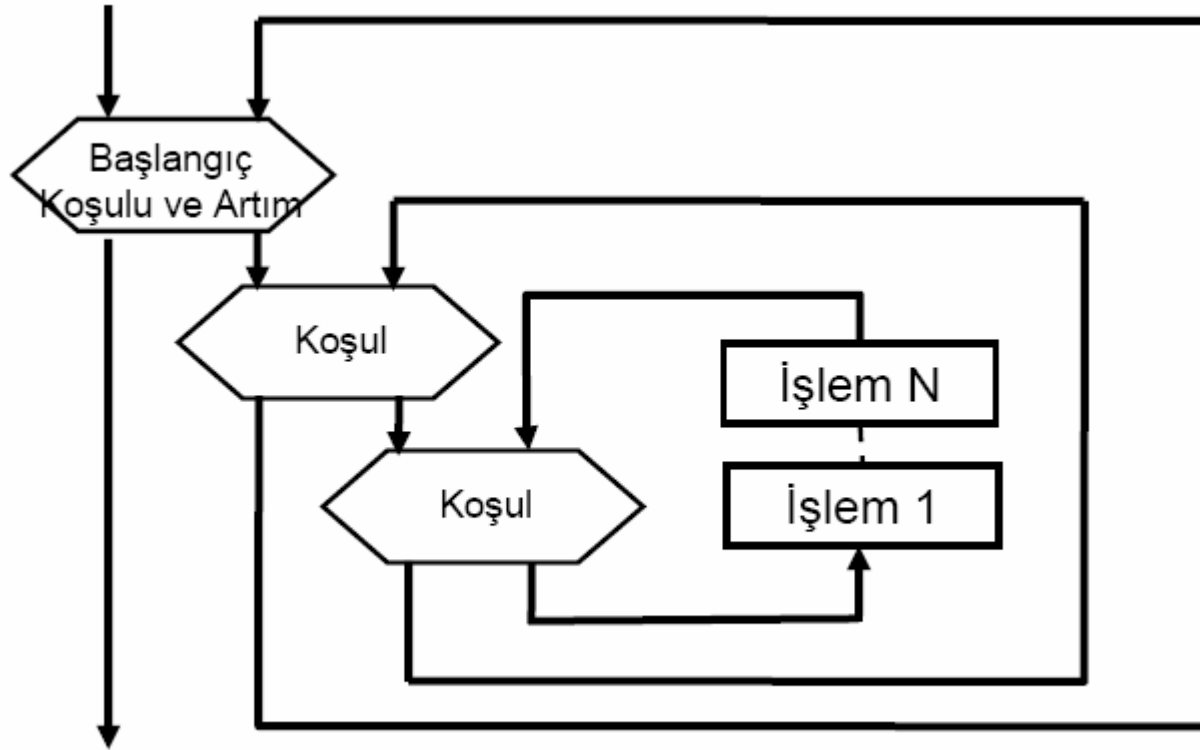


Akış Diyagramlarında Kullanılan Temel Şekiller

İç içe Döngülerin Kullanılması

İç içe döngü kurulurken en önemli unsur, içteki döngü sonlanmadan bir dıştaki döngüye geçilmemesidir. Diğer bir deyişle döngüler birbirlerini kesmemelidir.

Döngüye Giriş

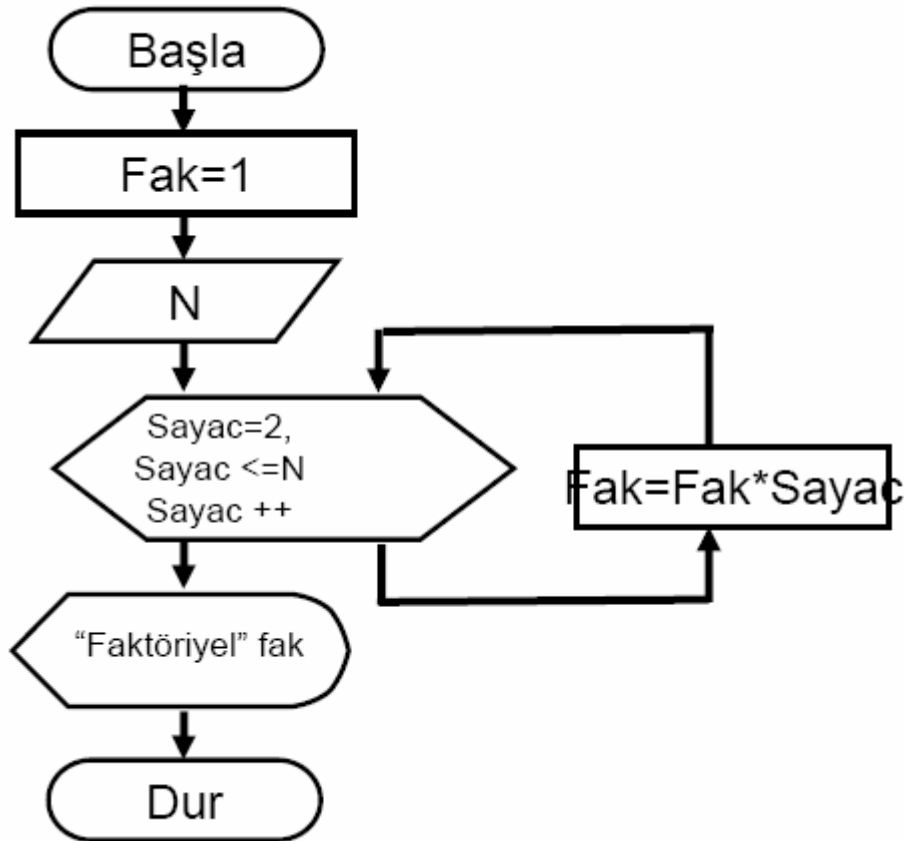


En içteki döngü bir dıştaki döngünün her adımında N kez tekrarlanır.

Döngüden Çıkış

Akış Diyagramlarında Kullanılan Temel Şekiller

Örnek: Klavyeden girilen N sayısının faktöriyelini alan algoritmanın akış diyagramını çiziniz.



- N ile hangi sayının faktöriyelini hesaplanacağı belirlenir ve N çevrimlik bir döngü kurulur.

- İlk çevrimde 1!, ikinci çevrimde 2! ve sırayla N'inci çevrim sonucunda da N! değeri hesaplanmış olur.

- Sayac > N koşulu oluştuğunda döngüden çıkılır ve elde edilen sonuç dış ortama aktarılır.



UYGULAMALAR

KAYNAKLAR

1. Rifat Çölkesen, "Veri yapıları ve algoritmalar", Papatya Yayınları, İstanbul, 2002.
2. Fahri Vatansever, "Algoritma geliştirme ve programlamaya giriş", Seçkin Yayınları, Ankara, 2009.
3. Aslan İnan, "MATLAB ve programlama", Papatya Yayınları, İstanbul, 2004.
4. <http://www.yildiz.edu.tr/~kunal/bilgisayarbil.html>, "Temel Bilgisayar Bilimleri Ders Notları-Ünal Küçük".
5. Soner Çelikkol, "Programlamaya giriş ve algoritmalar", Akademi Yayınları, İstanbul, 2001.
6. Feridun Karakoç, "Algoritma geliştirme ve programlamaya giriş", Temel Bilgisayar Bilimleri Ders Notları.
7. Maltepe Üniversitesi, "Programlamanın Temelleri Ders Notları".
8. www.akademi.itu.edu.tr/buzluca, "Feza Buzluca Bilgisayar Mimarisi Ders Notları".