



TEMEL ALGORİTMA VE PROGRAMLAMA

Ders 1

Programa Dillerinin Genel Yapısı

Doç.Dr. İlker ÜNAL
Burdur Mehmet Akif Ersoy Üniversitesi
Fen-Edebiyat Fakültesi
Hassas Tarım ve Tarımsal Robotlar Bölümü

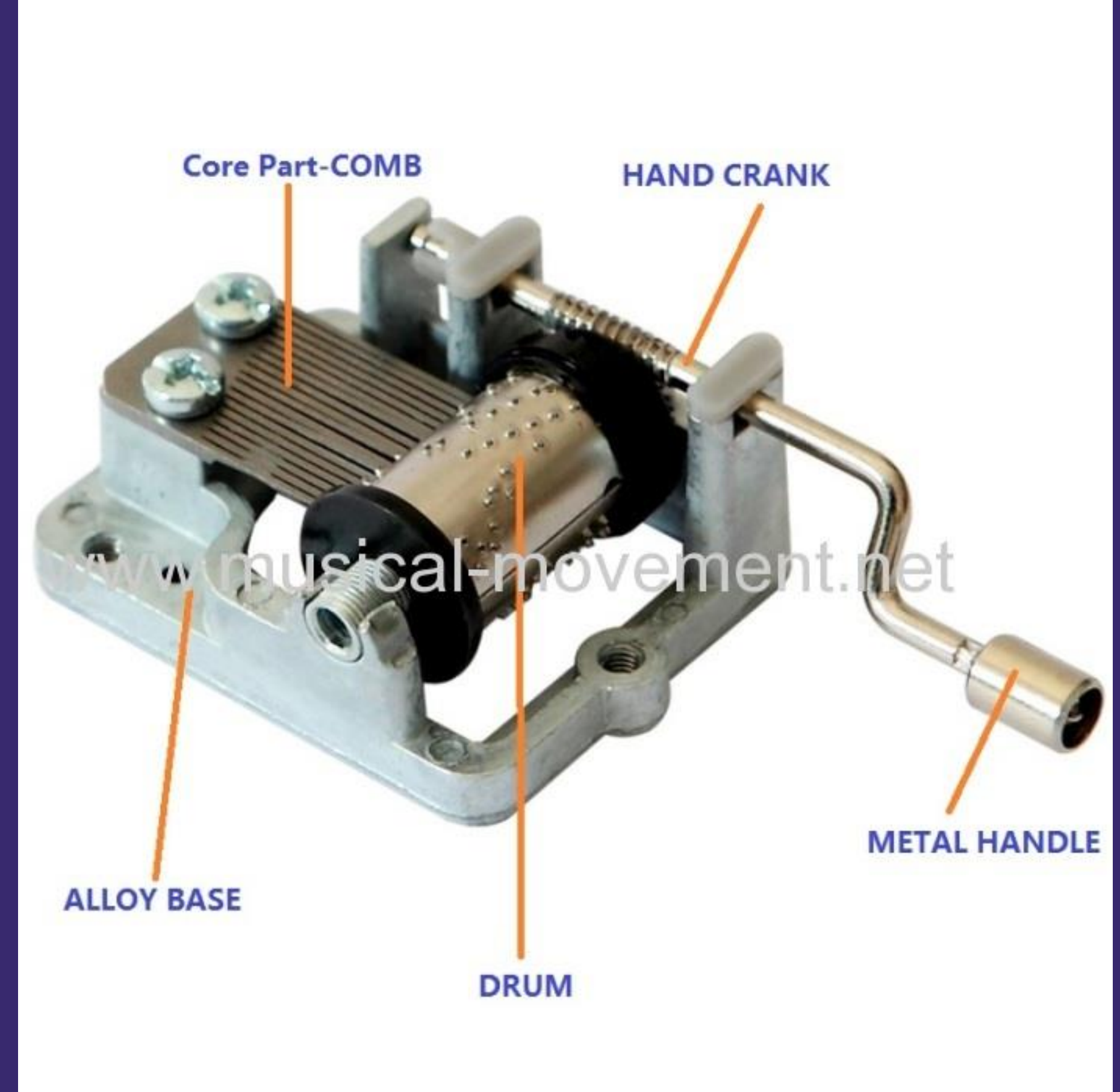
- Dersin işlenişi: 3 saat uygulamalı konu anlatımı
- Dersin web sayfası: **www.ilkerunal.com**
- Derse devam zorunludur. En fazla 4 hafta devamsızlık yapılabilir.
- Ders başlangıç saatlerine özen gösteriniz.
- Dersin başlangıcında yoklama alınacaktır.
- Ders esnasında lütfen kendi aranızda (veya kendi kendinize) konuşmayın, fısıldaşmayın, mesajlaşmayın.
- Dersi anlatan ve dinleyen kişilere lütfen saygı gösterin.
- Anlatılan, anlatılmayan, merak ettiğiniz her konuda soru sormaktan çekinmeyin.
- Cep telefonu vb. kişisel taşınabilir iletişim cihazlarınızı ders süresince mutlaka kapalı tutunuz.

- Vize %30
- Devam %10
- Final %60
- *Devamsızlık her hafta alınacak ancak devamsızlıktan kalmak yok. %10 devam için kanaat notu kullanılacaktır.*

DERS	KONULAR
01	Programa Dillerinin Genel Yapısı
02	Donanım ve Program Arasındaki Bağlantı
03	Nesne Tabanlı Programlama Mantığı
04	Sınıf, Metot ve Nesne Yapıları
05	Konsol Yapısı ve Program Yazma Ortamı
06	Algoritma ve Akış Şemaları
07	Değişkenler ve Veri Türleri
08	Operatörler
09	Kontrol Yapıları
10	Döngüler
11	Diziler
12	Matematiksel ve Metinsel İşlemler
13	Dosya İşlemleri



Bu ikonik fotoğrafta, öncü bilgisayar bilimcisi **Margaret Hamilton**, kendisinin ve ekibinin Apollo uzay aracını aya götürmek için yazdığı bilgisayar kodunun yanında duruyor!

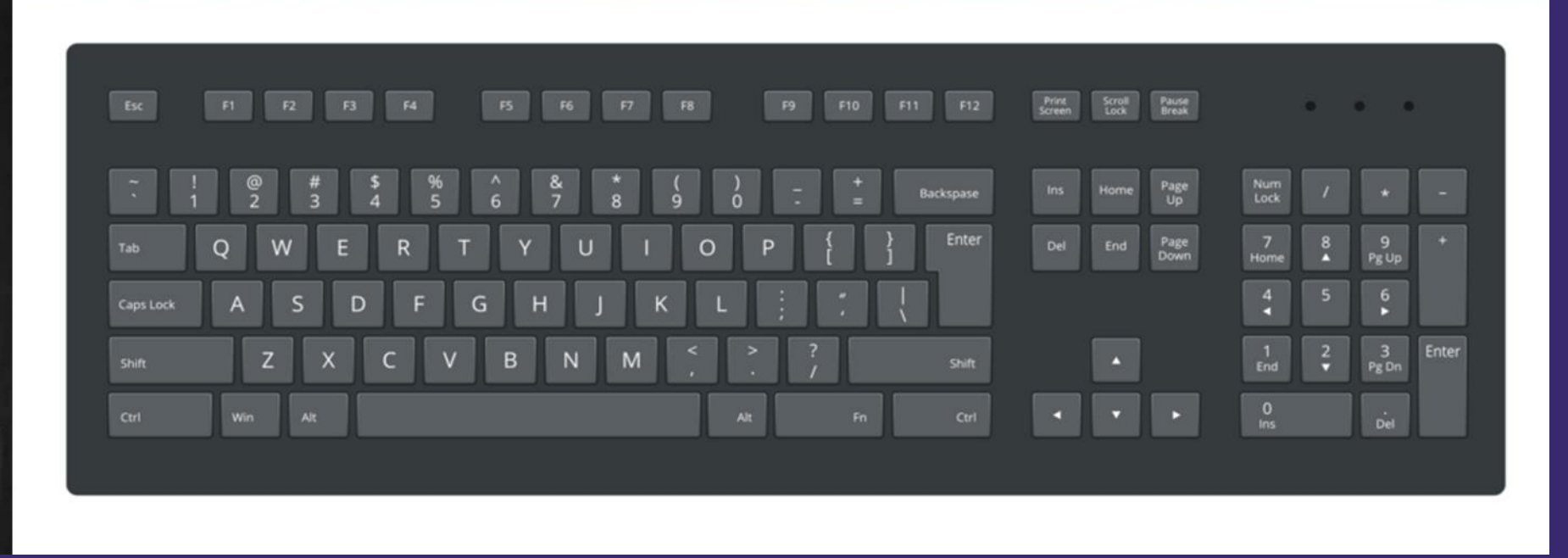
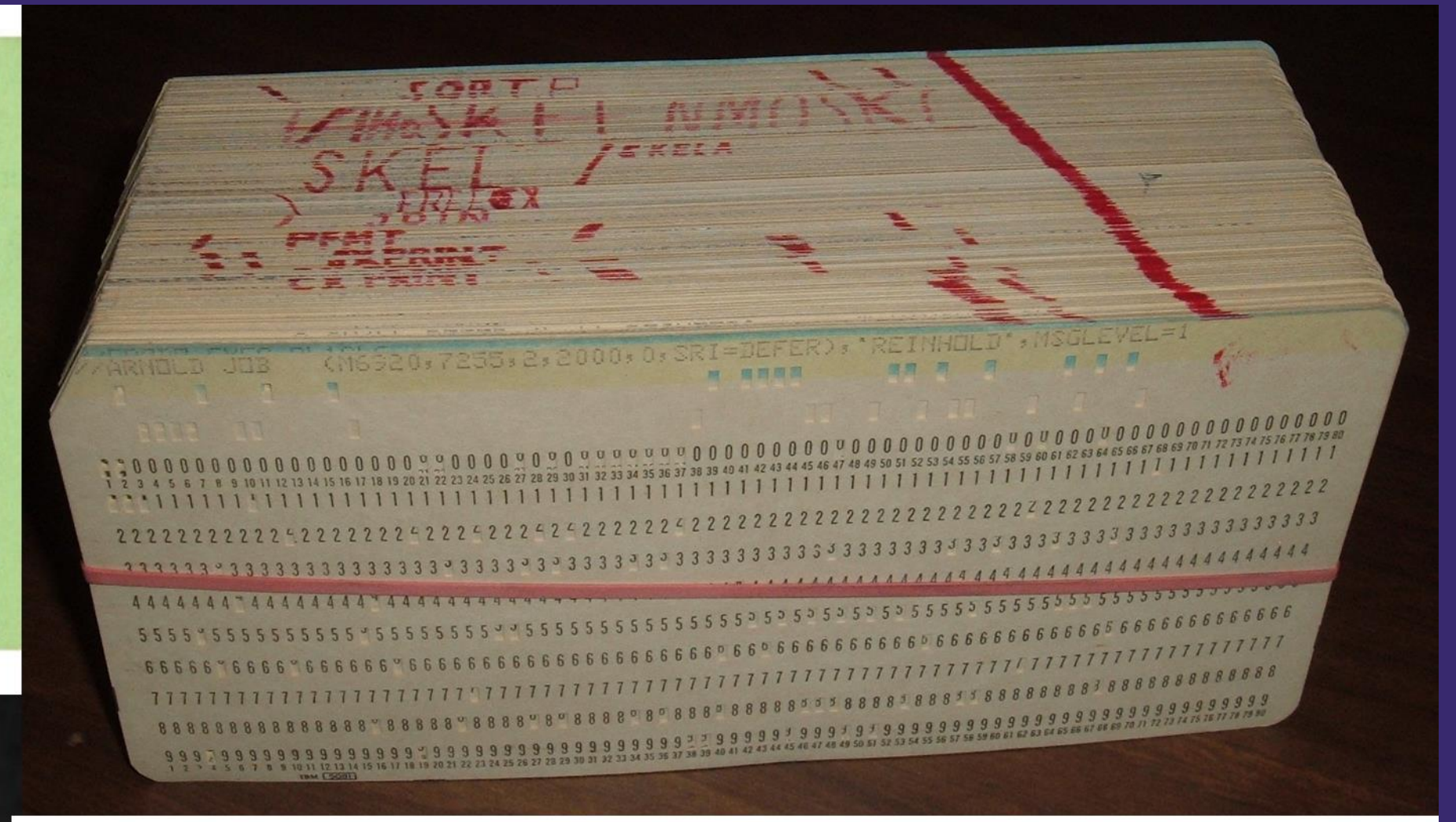
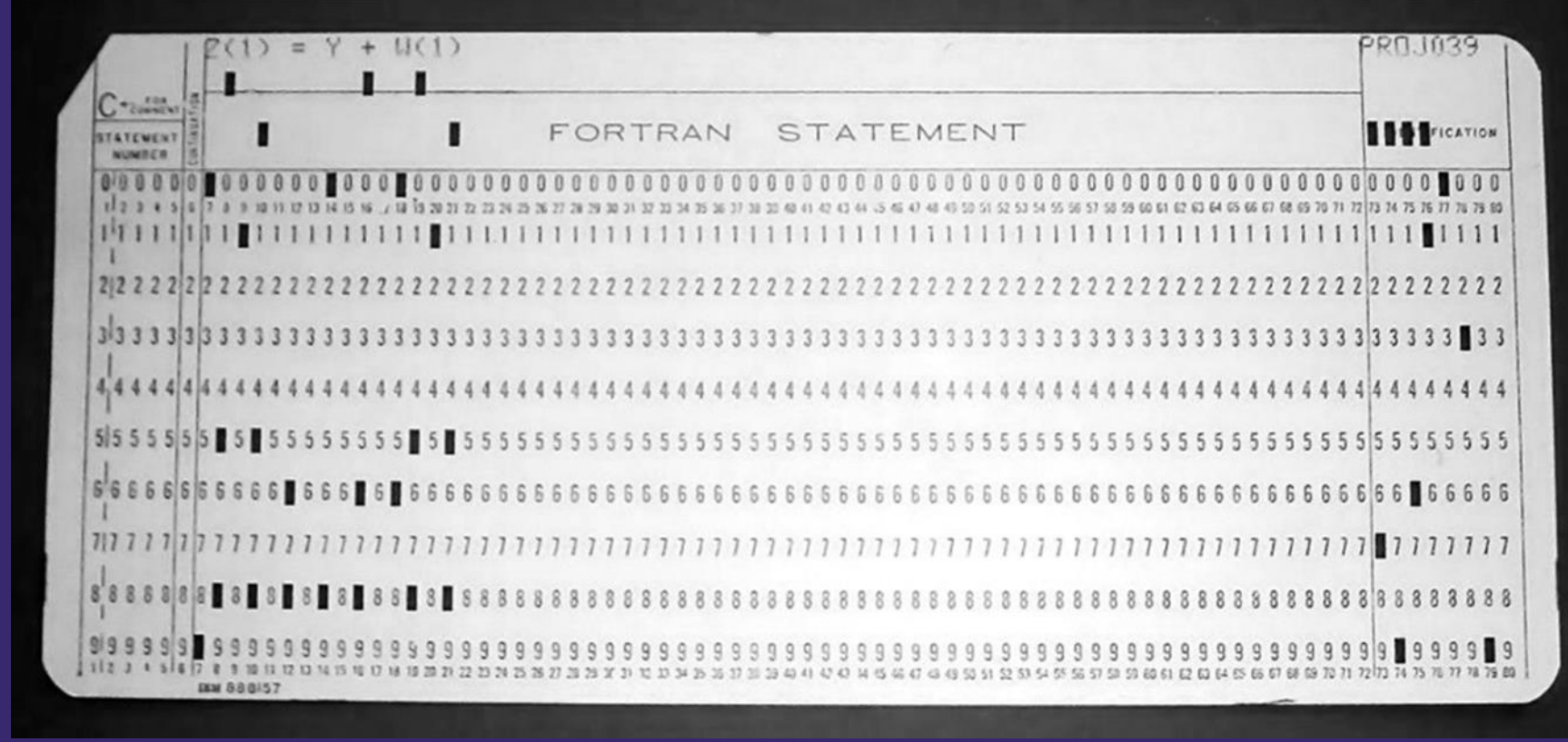
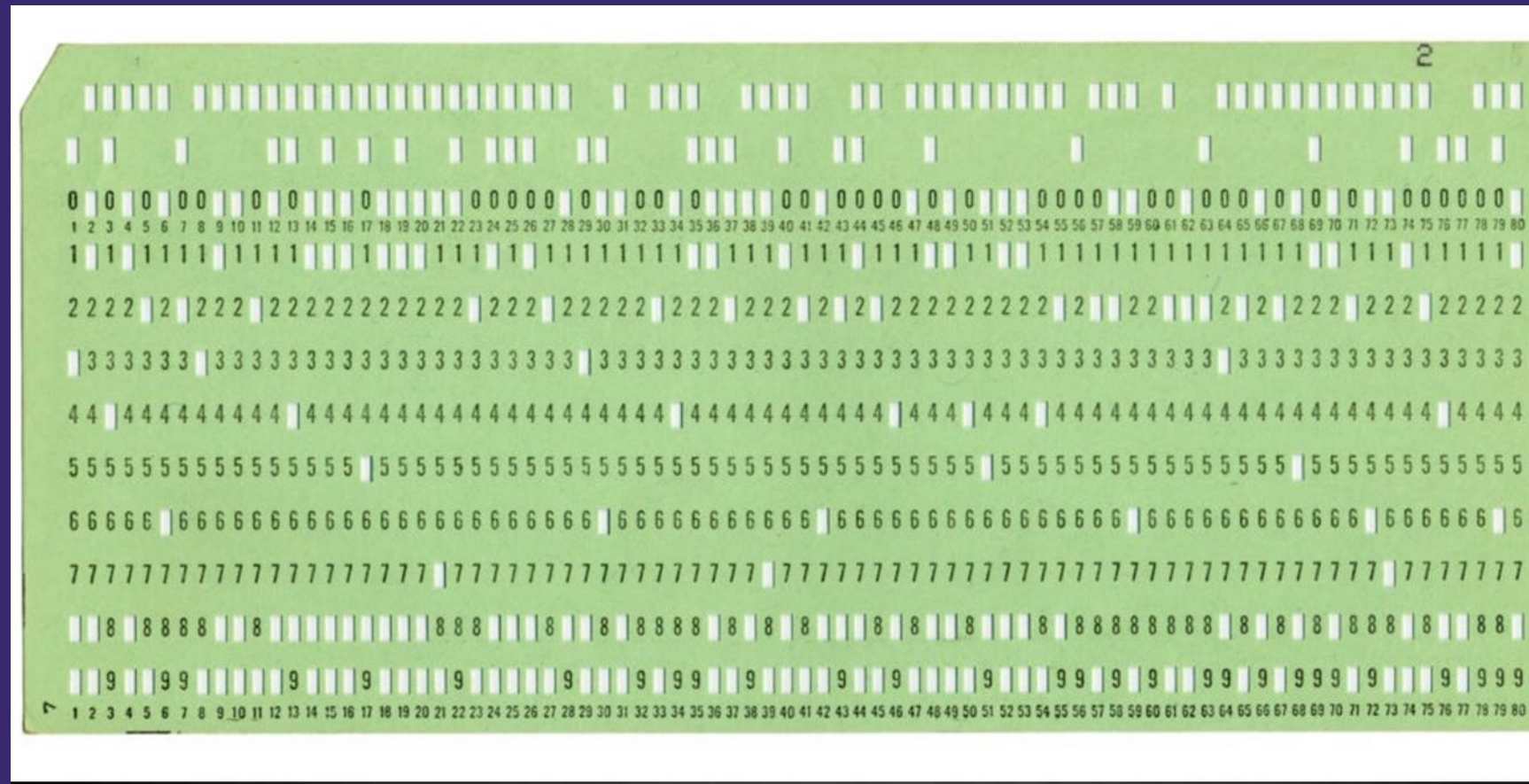




Fransa'nın Lyon kentinde Joseph Marie Jacquard (1752-1834) 1801'de vasıfsız işçilerin ipek üzerinde karmaşık desenler dokumalarına olanak tanıyan bir dokuma tezgahı tasarlamıştır. Jacquard Tezgahı, mekiğin her geçişinde kumaş çözgüsünün hangi kordonlarının kaldırılacağını belirleyen deliklerle delinmiş çoklu kartlardan oluşan bir zincir tarafından kontrol edilmektedir. Karmaşık işlemleri depolama ve otomatik olarak yeniden üretme yeteneği, tekstil üretiminde geniş uygulama alanı bulmuştur.



Delikli kartlar, bilgisayarların ilk zamanlarında yaygın olarak kullanılan bir tür veri depolama ve işleme teknolojisiydi. İlk olarak 17. yüzyılda dokuma tezgahlarında kullanılmak üzere icat edildiler ve daha sonra bilgisayarlarda kullanılmak üzere uyarlandılar. Delikli kartlar sert karton veya kağıttan yapıldı ve standart bir düzende içlerine delik sıraları açılırdı. Delikler ikili verileri temsil ederdi; delik "1"i, delik olmaması ise "0"ı temsil ederdi.



Delikli kartlar, Hollerith kartları veya IBM kartları olarak da bilinir, bilgisayar verilerini ve talimatlarını temsil etmek için deliklerin elle veya makineyle delinebildiği kağıt kartlardır. Bunlar, erken bilgisayarlara veri girmek için yaygın olarak kullanılan bir araçtı. Kartlar, delik dizisini makine dili olarak bilinen dijital bilgilere dönüştüren bir bilgisayara bağlı bir kart okuyucuya beslenirdi.

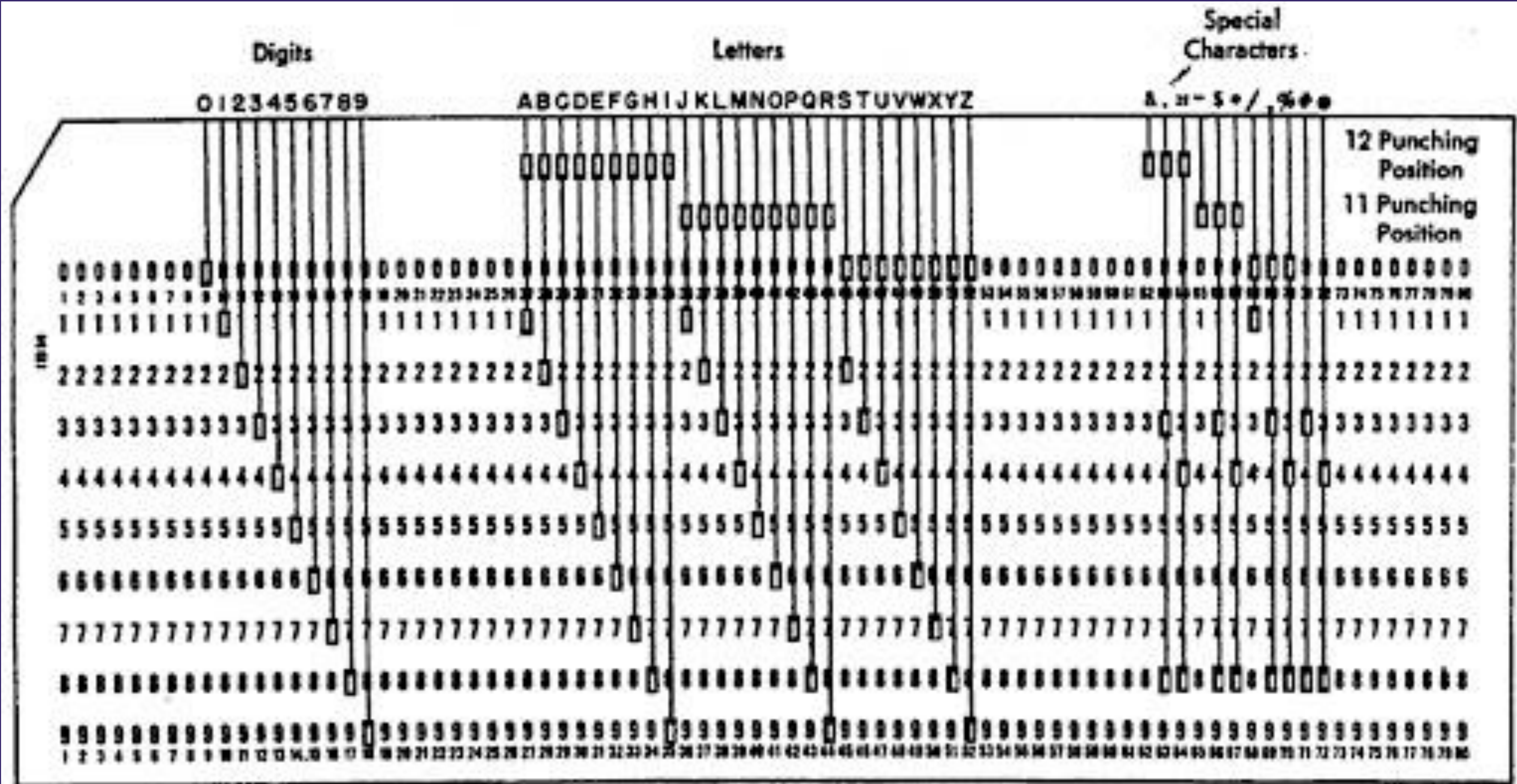
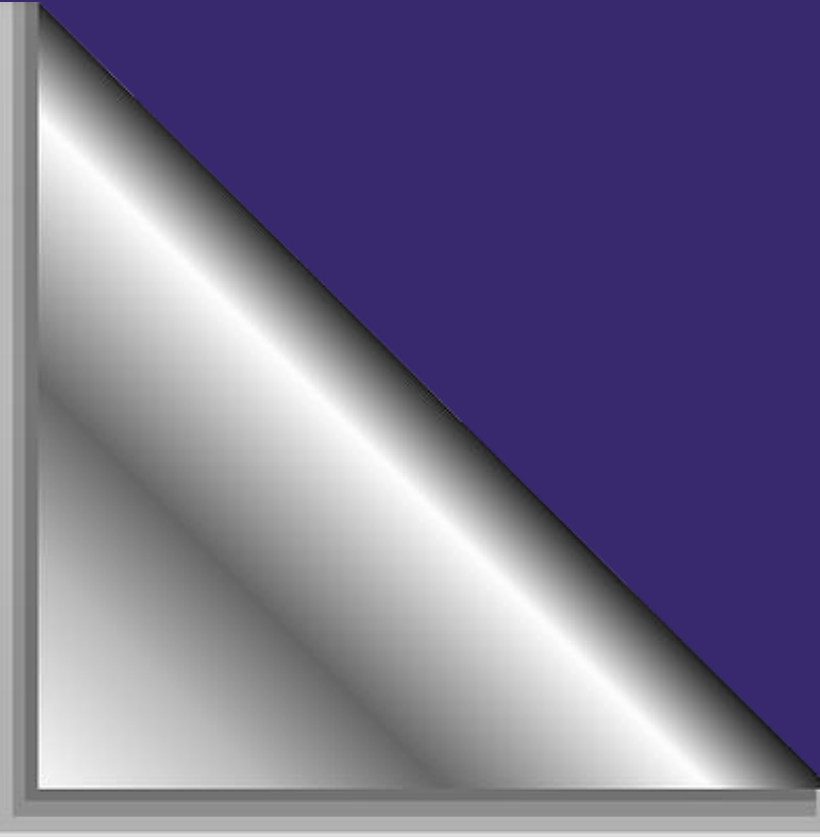


Figure 2. Punching Positions in Card



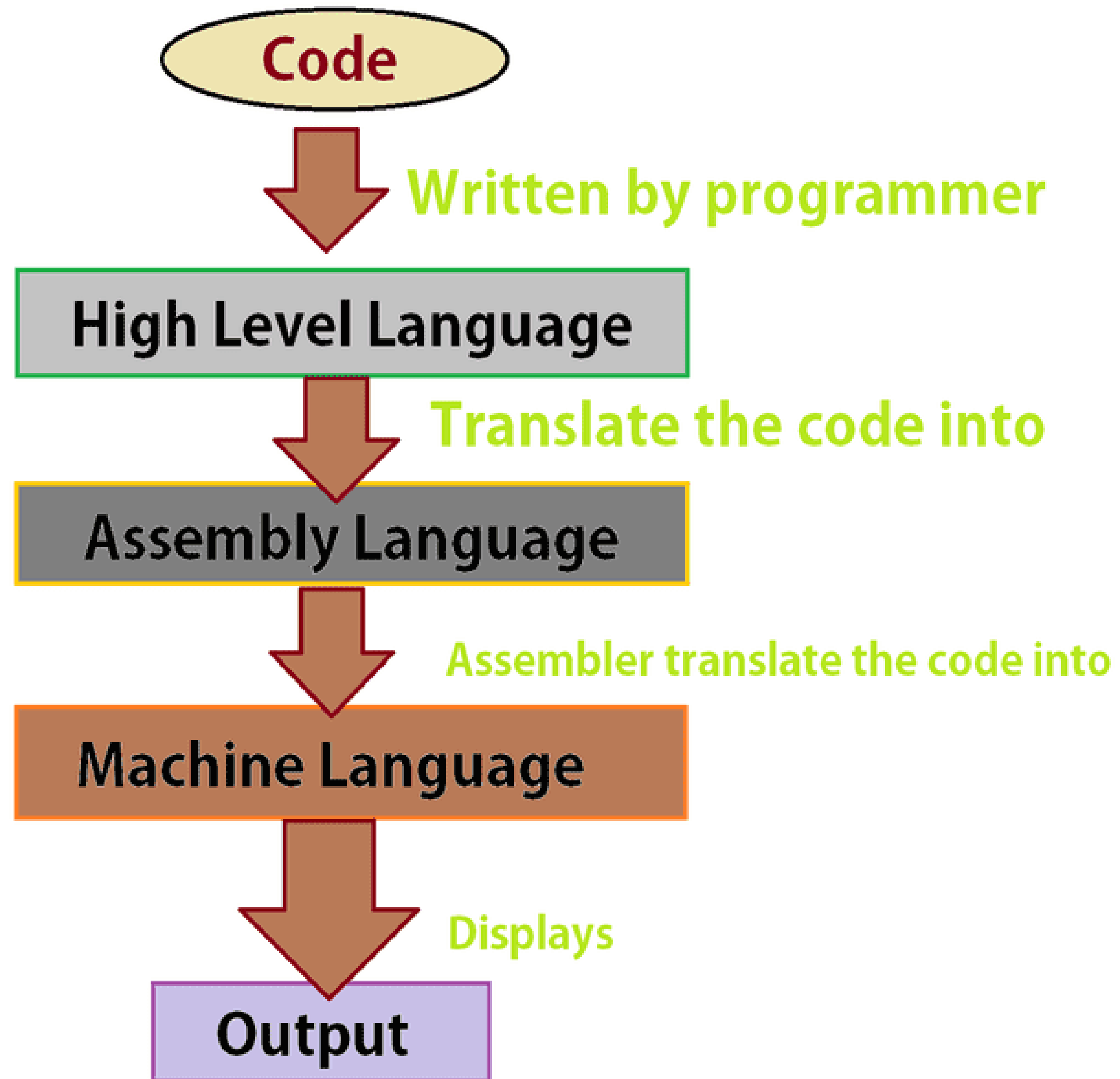
```
0110000101100110011000010
1100011011001010111011101
1000100110100101101010011
0101001101110001000000110
1000011010100110101001100
1010110111001111001011001
0100100000011001010110100
1011010000110001001100101
0010000000100000011010010
1111001011001010110111001
1011010111001101100011011
110010111011101100010011001101101001000
0001110011011110010110010100100000011000110
11001010111001001100010011001000001000000110
0101011010000111001101100001011011010111100
000100000011110010111011100010110010101
1011010111100001100100011101010111000101101
0000111001001100001011011100110101000100000
0110010001101001011010000110001001101000011
0001001011001010110100101101000011000100110
0101001000000010000001101001011110010110010
1011011100110110101110011011000110111100101
1101110110001001100110011011010010000001110
0110111100110010101101001011010000110001001
1001010010000000100000011010010111100101100
1010110111001101101011100110110001101111001
0111011101100010011001100110110100100000011
1001101111001100101011010010110100001100010
0110010100100000001000000110100101111001011
0010101101110011011010111001101100011011110
0101110111011000100110011001101101001000000
1110011011110011001010110100101101000011000
1001100101001000000010000001101001011110010
```

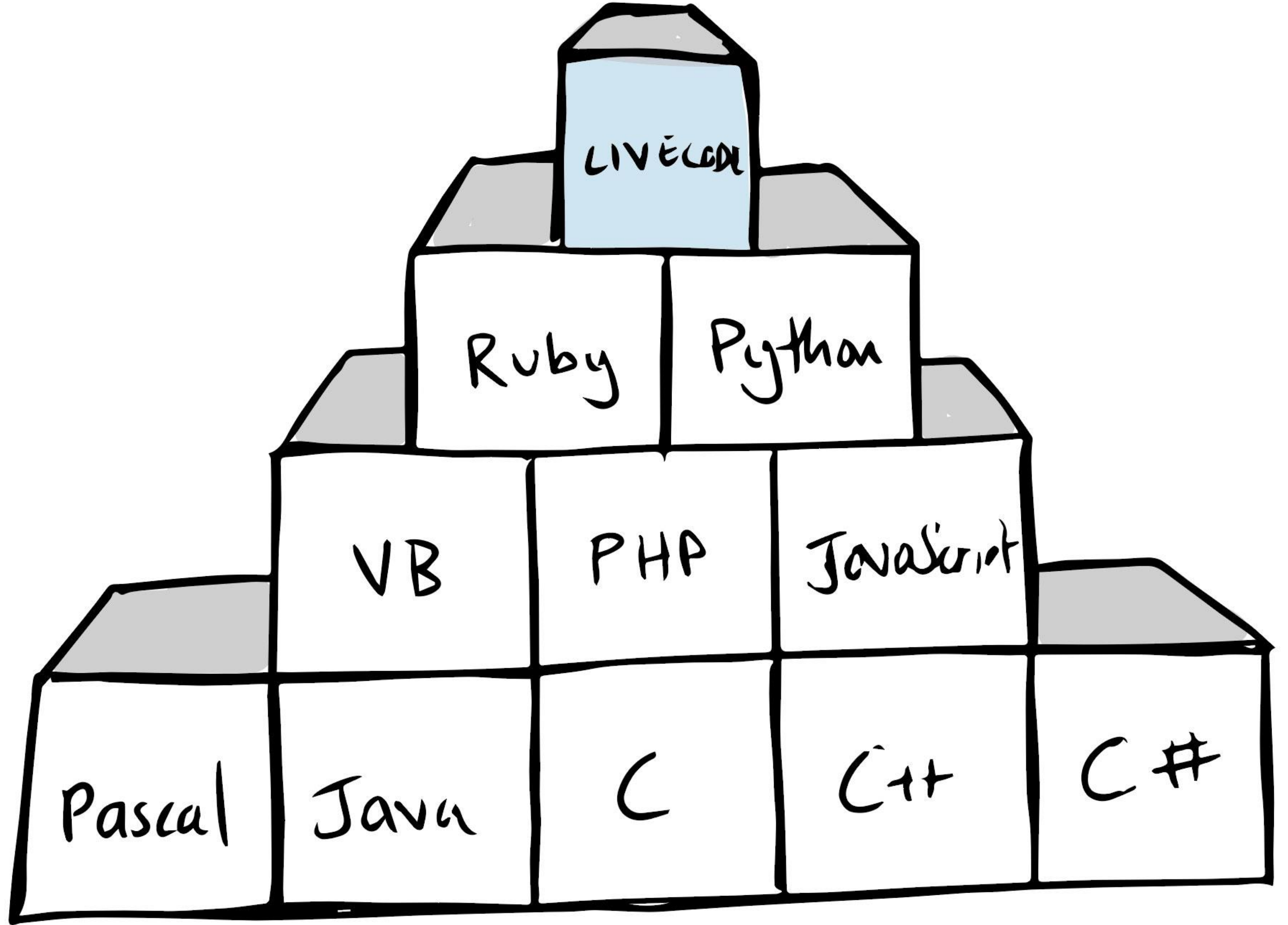
İkili dosya, 0'lar ve 1'lerden oluşan bir ikili kodlama şeması kullanarak veri depolayan bir bilgisayar dosyasıdır. Karakterleri veri olarak kullanan metin dosyalarının aksine, ikili dosyalar bilgileri doğrudan ikili biçime kodlar. Bu, bilgisayarların yürütülebilir programlar, resimler, ses, video ve diğer dijital bilgi türleri dahil olmak üzere karmaşık verileri verimli bir şekilde depolamasını ve işlemesini sağlar. İkili dosyalar, yazılım geliştirmeden multimedya işleme ve veri depolamaya kadar çeşitli bilgi işlem görevleri için gereklidir.

00000000	01	00	FF	FF	00	00	00	00	00	00	00	00	40	00	CC	80	@...
00000010	0C	00	00	00	00	00	26	01	8F	00	00	00	00	00	53	00	&.....S.
00000020	65	00	6C	00	65	00	63	00	74	00	20	00	52	00	75	00	e.l.e.c.t. R.u.
00000030	6C	00	65	00	00	00	08	00	00	00	00	01	4D	00	53	00	l.e.....M.S.
00000040	20	00	53	00	68	00	65	00	6C	00	6C	00	20	00	44	00	.S.h.e.l.l. .D.
00000050	6C	00	67	00	00	00	00	00	00	00	00	00	00	02	00	00	l.g.....
00000060	03	01	A1	50	53	00	3A	00	C3	00	36	00	32	25	00	00	...PS.....6.2%..
00000070	FF	FF	83	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000080	03	00	01	50	0E	00	56	00	41	00	0A	00	4A	26	00	00	...P..V.A...J&..
00000090	FF	FF	80	00	26	00	41	00	70	00	70	00	6C	00	79	00	...&A.p.p.l.y.
000000a0	20	00	74	00	6F	00	20	00	61	00	6C	00	6C	00	00	00	..t.o..a.l.l....
000000b0	00	00	00	00	00	00	00	00	00	00	00	00	01	00	01	50	P
000000c0	7E	00	7D	00	32	00	0E	00	01	00	00	00	FF	FF	80	00	~}.2.....
000000d0	4F	00	4B	00	00	00	00	00	00	00	00	00	00	00	00	00	O.K.....
000000e0	00	00	01	50	B4	00	7D	00	32	00	0E	00	02	00	00	00	...P..}.2.....
000000f0	FF	FF	80	00	43	00	61	00	6E	00	63	00	65	00	6C	00	...C.a.n.c.e.l.
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	50	P
00000110	EA	00	7D	00	32	00	0E	00	09	00	00	00	FF	FF	80	00	..}.2.....
00000120	26	00	48	00	65	00	6C	00	70	00	00	00	00	00	00	00	&H.e.l.p.....
00000130	00	00	00	00	00	00	00	00	80	08	81	50	0E	00	3A	00	P....
00000140	3B	00	0E	00	2F	25	00	00	FF	FF	81	00	00	00	00	00	.../%.....
00000150	00	00	00	00	00	00	00	00	00	00	02	50	0E	00	30	00	P..0.
00000160	1E	00	08	00	EE	25	00	00	FF	FF	82	00	46	00	69	00	%.F.i.
00000170	6C	00	65	00	20	00	54	00	79	00	70	00	65	00	00	00	l.e...T.y.p.e...
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	02	50	P
00000190	54	00	30	00	2C	00	08	00	EF	25	00	00	FF	FF	82	00	T.O.....%
000001a0	50	00	61	00	72	00	73	00	69	00	6E	00	67	00	20	00	P.a.r.s.i.n.g...
000001b0	52	00	75	00	6C	00	65	00	73	00	00	00	00	00	00	00	R.u.l.e.s.....
000001c0	00	00	00	00	00	00	00	00	07	00	00	50	06	00	07	00	P....
000001d0	1A	01	71	00	ED	25	00	00	FF	FF	80	00	00	00	00	00	..q.%......
000001e0	00	00	00	00	00	00	00	00	00	00	02	50	0E	00	11	00	P....
000001f0	3E	00	08	00	EC	25	00	00	FF	FF	82	00	53	00	65	00	>...%.S.e.
00000200	6C	00	65	00	63	00	74	00	20	00	52	00	75	00	6C	00	l.e.c.t. R.u.l.
00000210	65	00	20	00	46	00	6F	00	72	00	20	00	46	00	69	00	e..F.o.r. F.i.
00000220	6C	00	65	00	00	00	00	00	00	00	00	00	00	00	00	00	l.e.....
00000230	80	08	81	50	0E	00	1B	00	08	01	0E	00	EB	25	00	00	...P.....%
00000240	FF	FF	81	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000250	00	00	02	50	19	00	61	00	37	00	08	00	6B	26	00	00	...P..a.7...k&..
00000260	FF	FF	82	00	00	00	00	00									

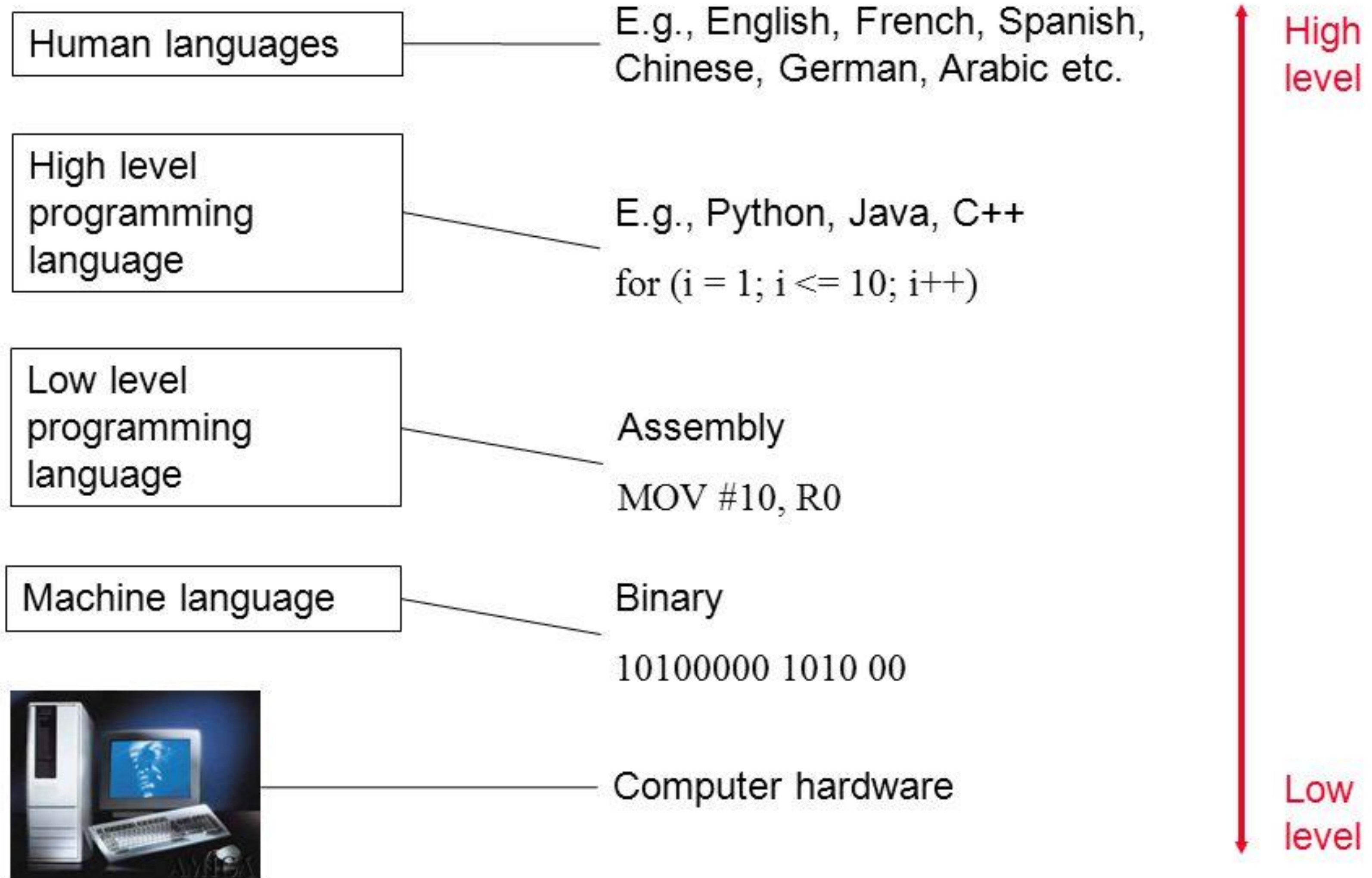
Bilgisayarların düşük seviyede depoladığı standart ikili sisteme kıyasla onaltılık sistemin tercih edilmesinin birkaç belirgin nedeni vardır.

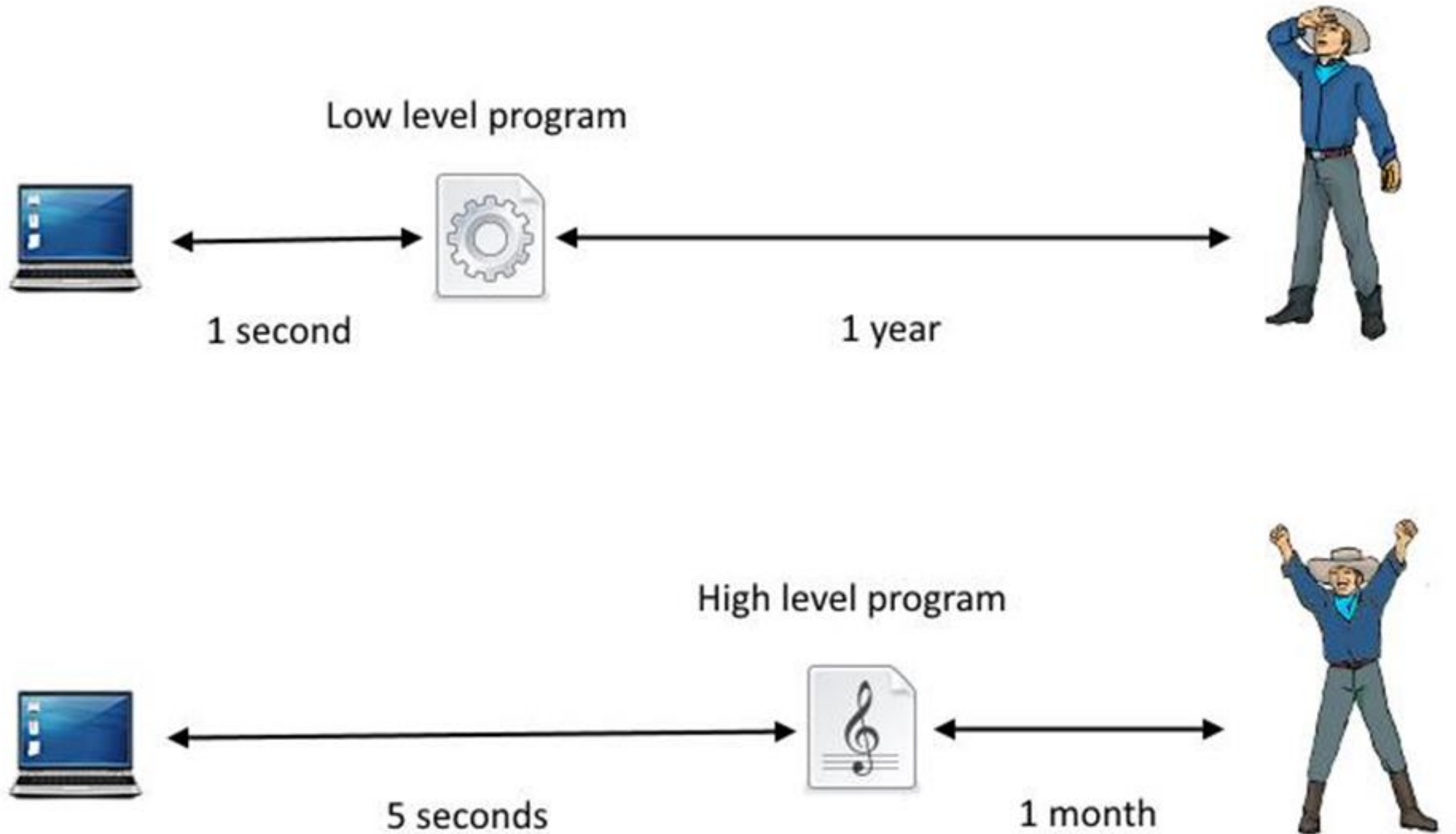
- Okunabilirlik. Onaltılık sayı sistemi, bizim alıştığımız 10 tabanlı sayma sistemimize daha çok benzeyen rakamlar kullanır ve bu nedenle E7 gibi bir sayının 11100111'e kıyasla ne kadar büyük olduğuna bir bakışta karar vermek daha kolaydır.
- Daha yüksek bilgi yoğunluğu. 2 onaltılık basamakla, 0'dan 255'e kadar herhangi bir sayıyı ifade edebiliriz. Aynısını ikili sistemde yapmak için 8 basamağa ihtiyacımız var. Sayılar büyüdükçe, daha fazla basamağa ihtiyaç duymaya başlarız ve başa çıkmak daha da zorlaşır.



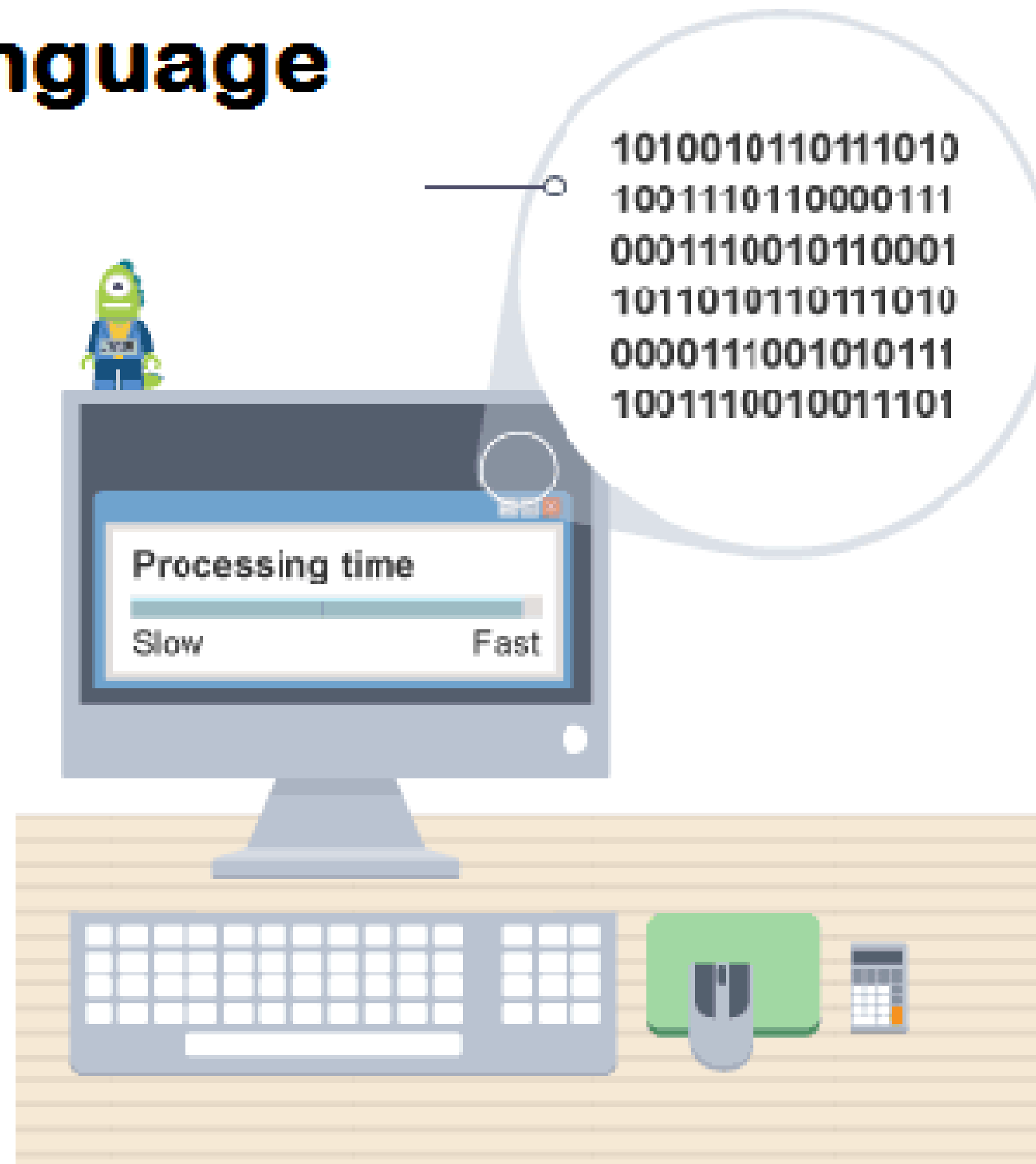


High Vs. Low Level Languages

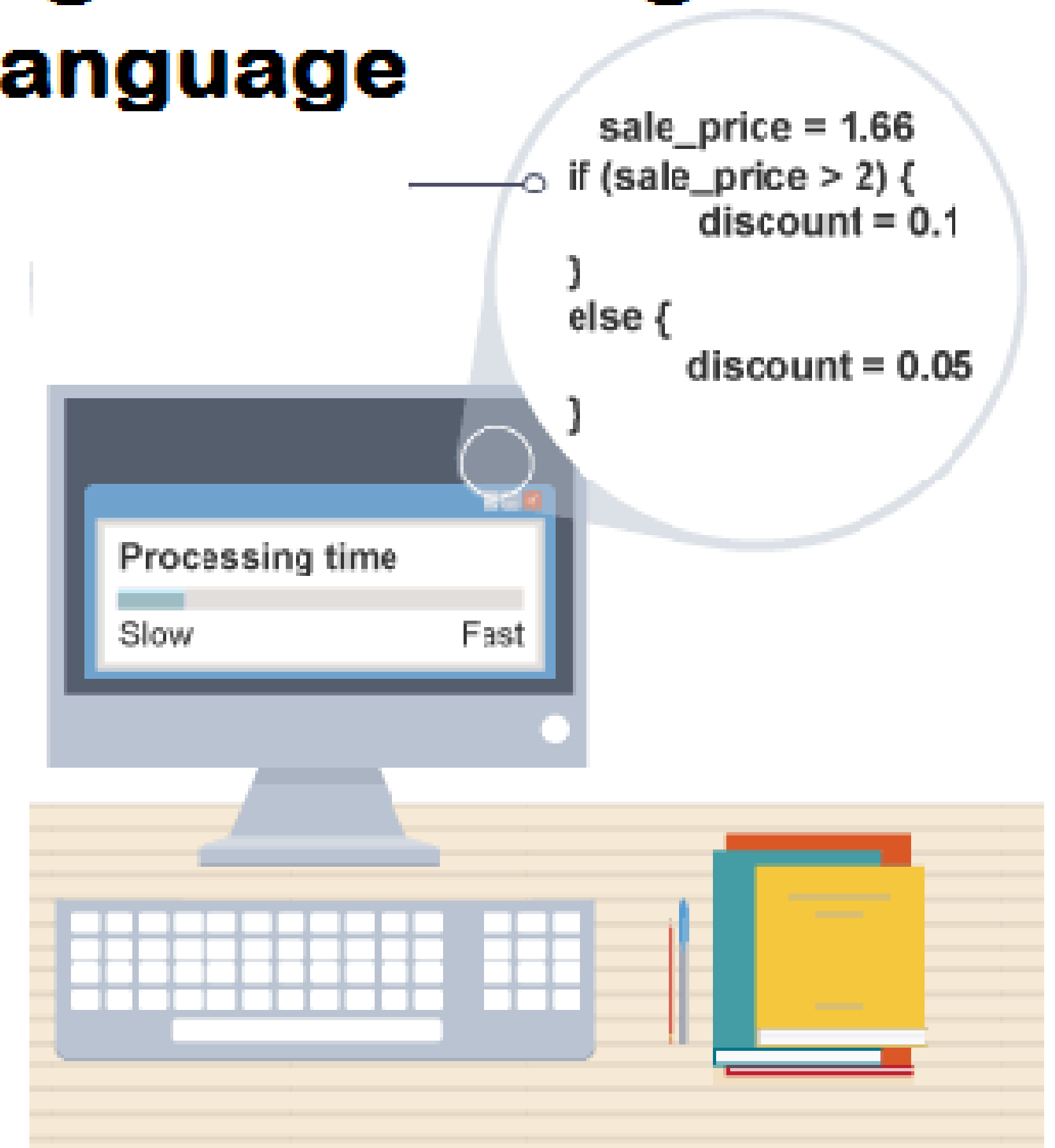




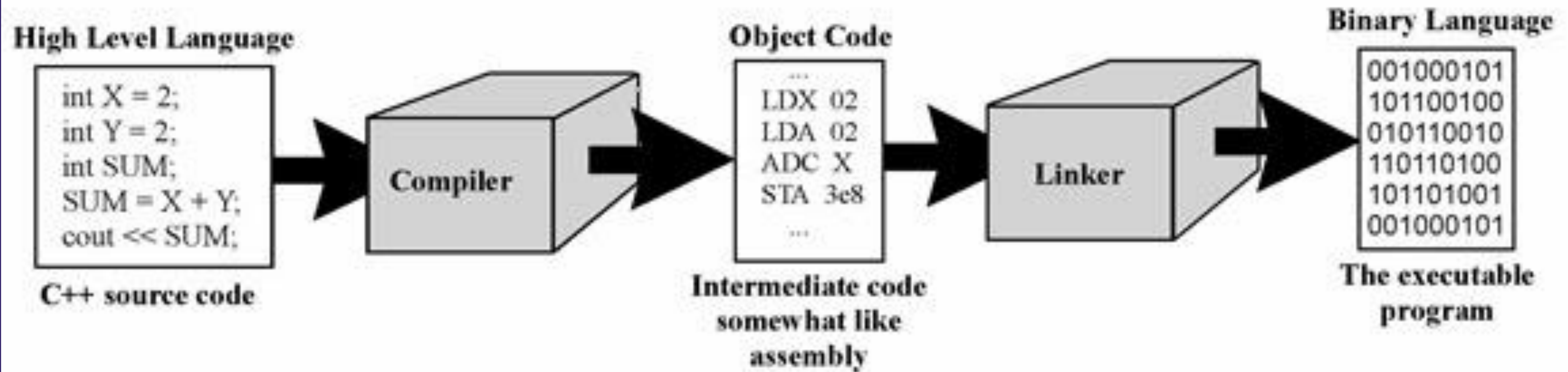
Low Level Programming Language



High Level Programming Language



Translating a High Level Language into Binary



```
while(n>0)
{
    sum = sum + n;
    --n;
}
```

Compile

```
L28  movf    _n,f
      btfsc  STATUS,Z
      goto  L41
      movf  _n,f
      addwf _sum,f
      btfsc  STATUS,C
      incf  _sum+1,f
      decf  _n,f
      goto  L28
```

L41

(a) First, compile to assembly-level code.

```
L28  movf    _n,f
      btfsc  STATUS,Z
      goto  L41
      movf  _n,f
      addwf _sum,f
      btfsc  STATUS,C
      incf  _sum+1,f
      decf  _n,f
      goto  L28
```

Assemble

```
0000100010010011
0001100100000011
0010100000001111
0000100000010011
0000100000010011
0000100000010011
0000011110010100
0001100000000011
0000101010010101
0111100000000111
```

L41

(b) Second, assemble-link to machine code.

Assembly Language

```
mov eax, ebx  
xor  eax, eax  
add  eax, 0xff
```



Assembler + Linker

Translator

Machine Language

```
010110100101  
111010101010  
101010101010
```



High or Low Level Programming Language? Justify your decision.

```
10110110010010101
01100011100100010
```

Machine

```
# Ask the player for a name
inName = ""
inName = input ("Please enter your name: ")
while ((len(inName) == 0) or (len(inName)>12)):
    # Validate name using length check for 12 characters
    print ("Name must be 12 characters long\n")
    inName = input ("Please enter your name: ")
```

Python

```
38      MOV      R1,#3
39      MOV      R2,#0
40  LOOP:  ADD      R2,R1,R2
41      SUB      R1,#1
42      CMP      R1,#0
43      BNE      LOOP
44  END:
```

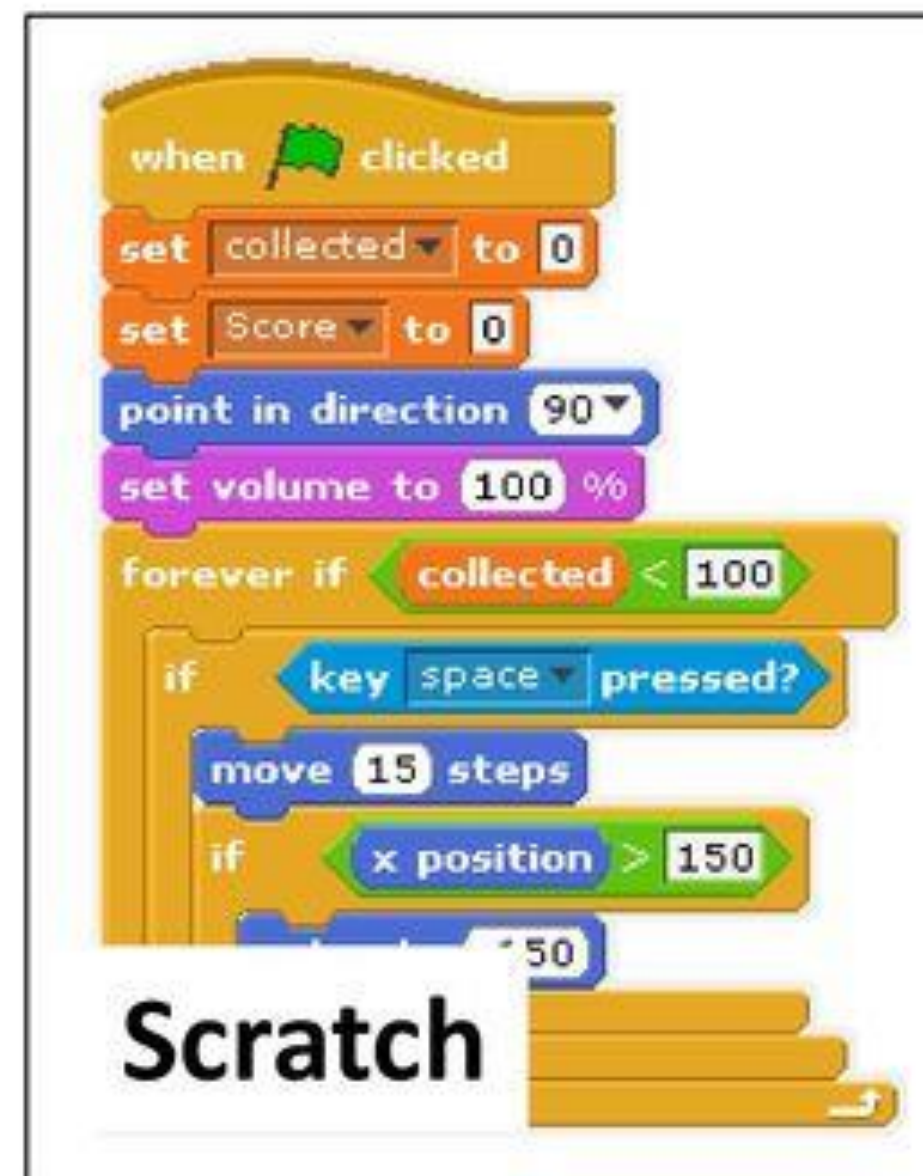
Assembly

```
System.out.println ("----- Main Menu -----");
System.out.println ("  1.    Customer");
System.out.println (" 99.    Exit ");
System.out.println ("");
System.out.println (" Select an option: ");

choice = inNumbers.nextInt();
eat = inNumbers.nextLine();

return (choice);
```

Java



Scratch

MOST USED PROGRAMMING LANGUAGES IN 2026


JavaScript
66.0%


SQL
58.6%


Python
57.9%


Bash/Shell
48.7%


TypeScript
43.6%


Java
29.4%


C#
27.8%


C++
23.5%


PowerShell
23.2%


C
22.0%


PHP
18.9%


Go
16.4%


Rust
14.8%


Kotlin
10.8%

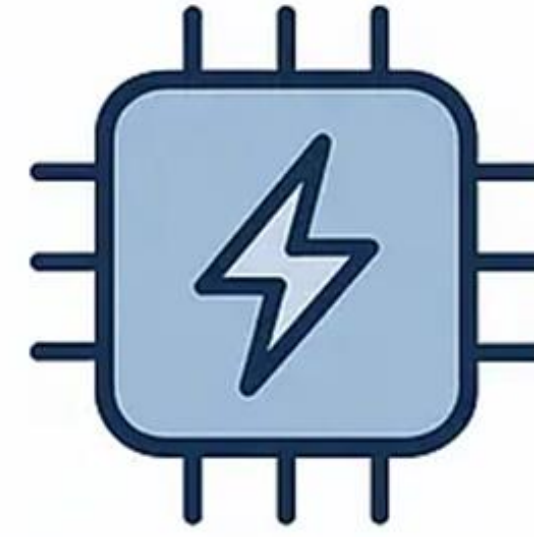

Dart
5.9%


Swift
5.4%

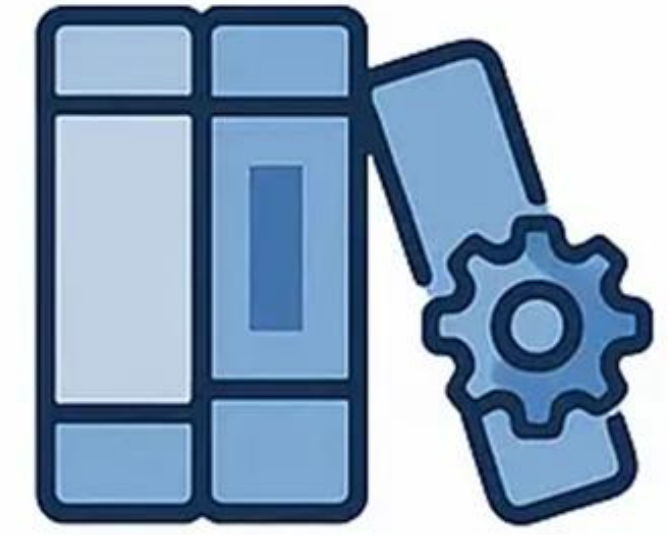
Why Programming Languages Matter in AI



Rapid Prototyping



Performance



Library Support



Community
& Ecosystem

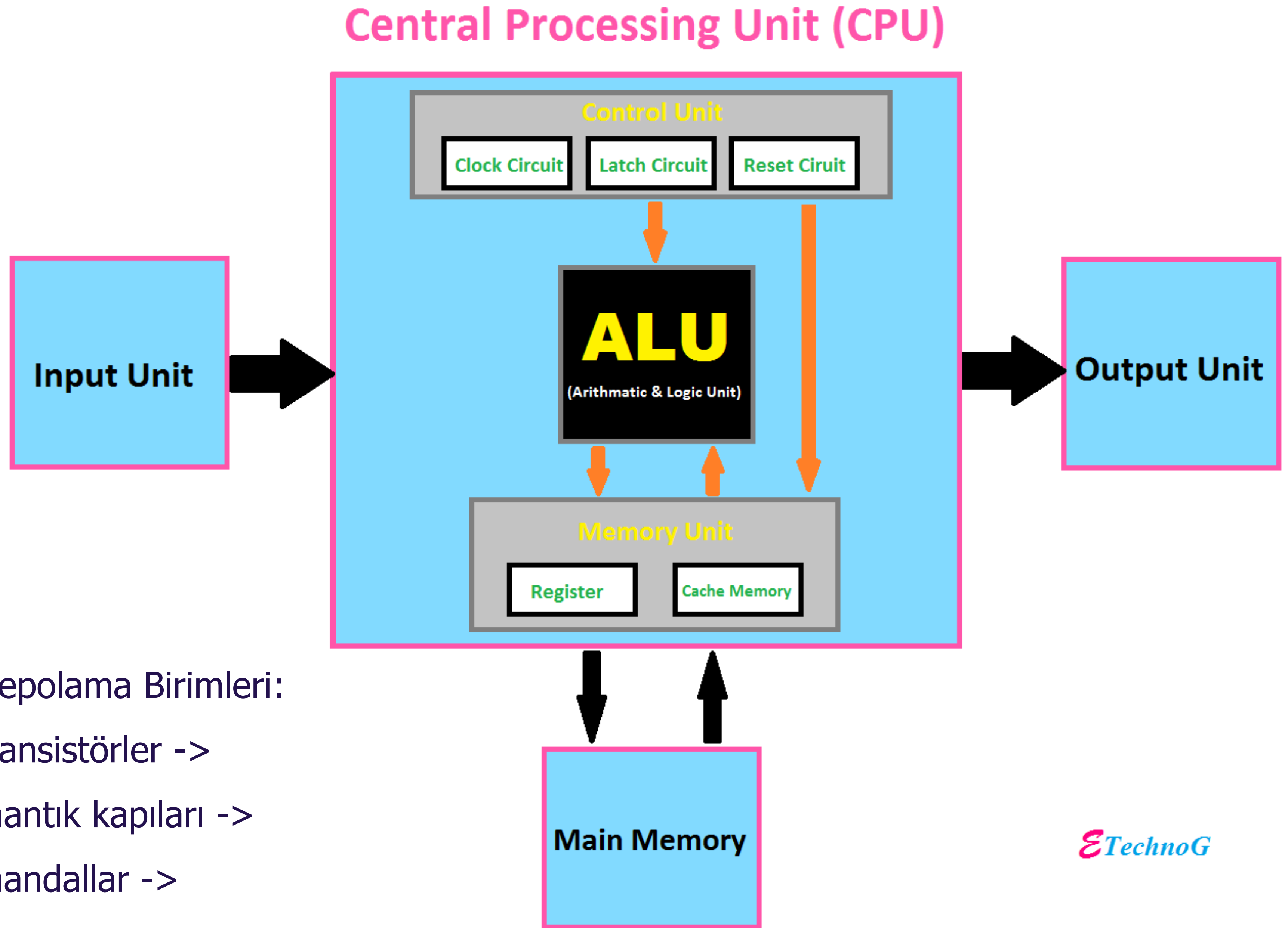


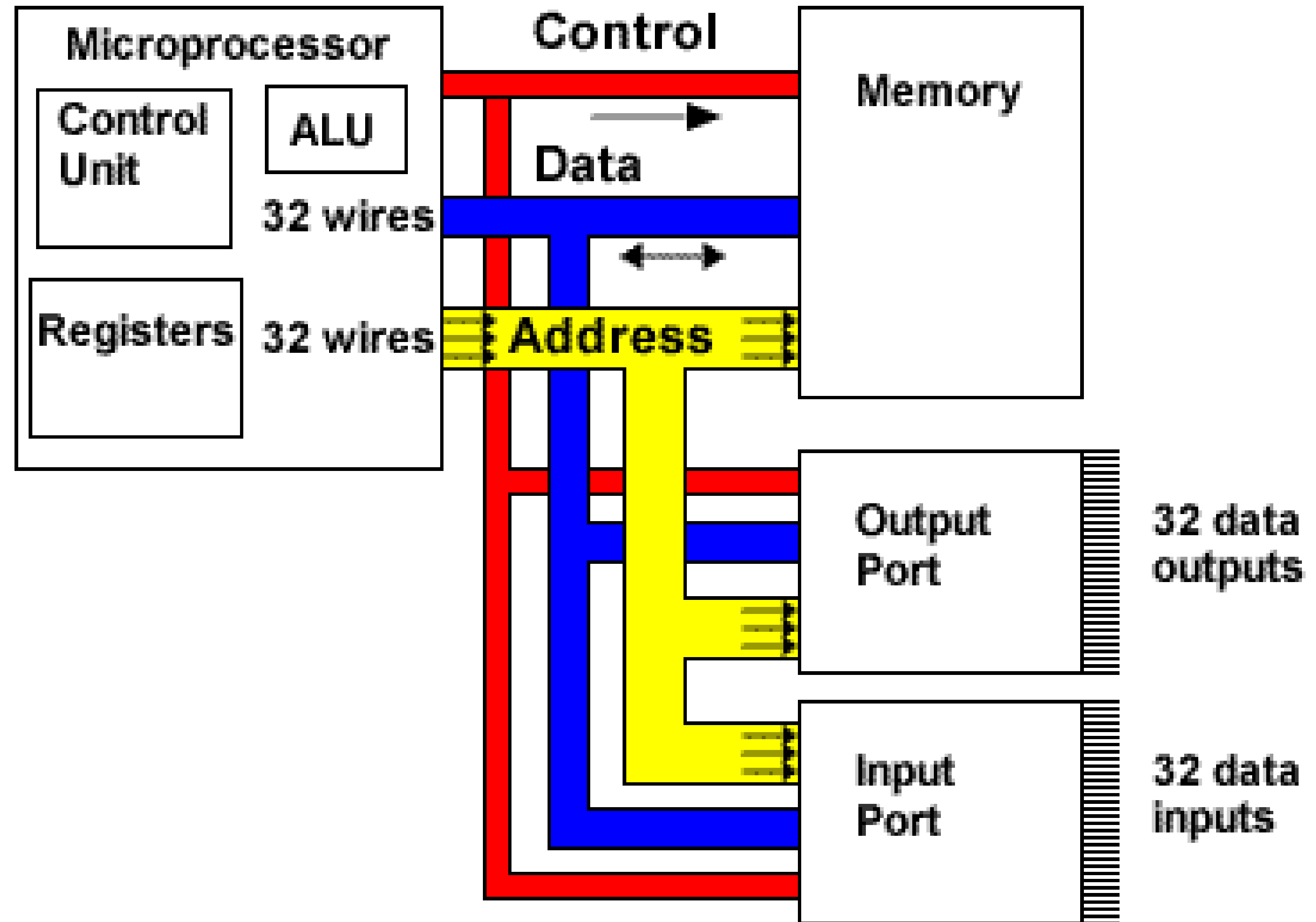
Deployment &
Integration

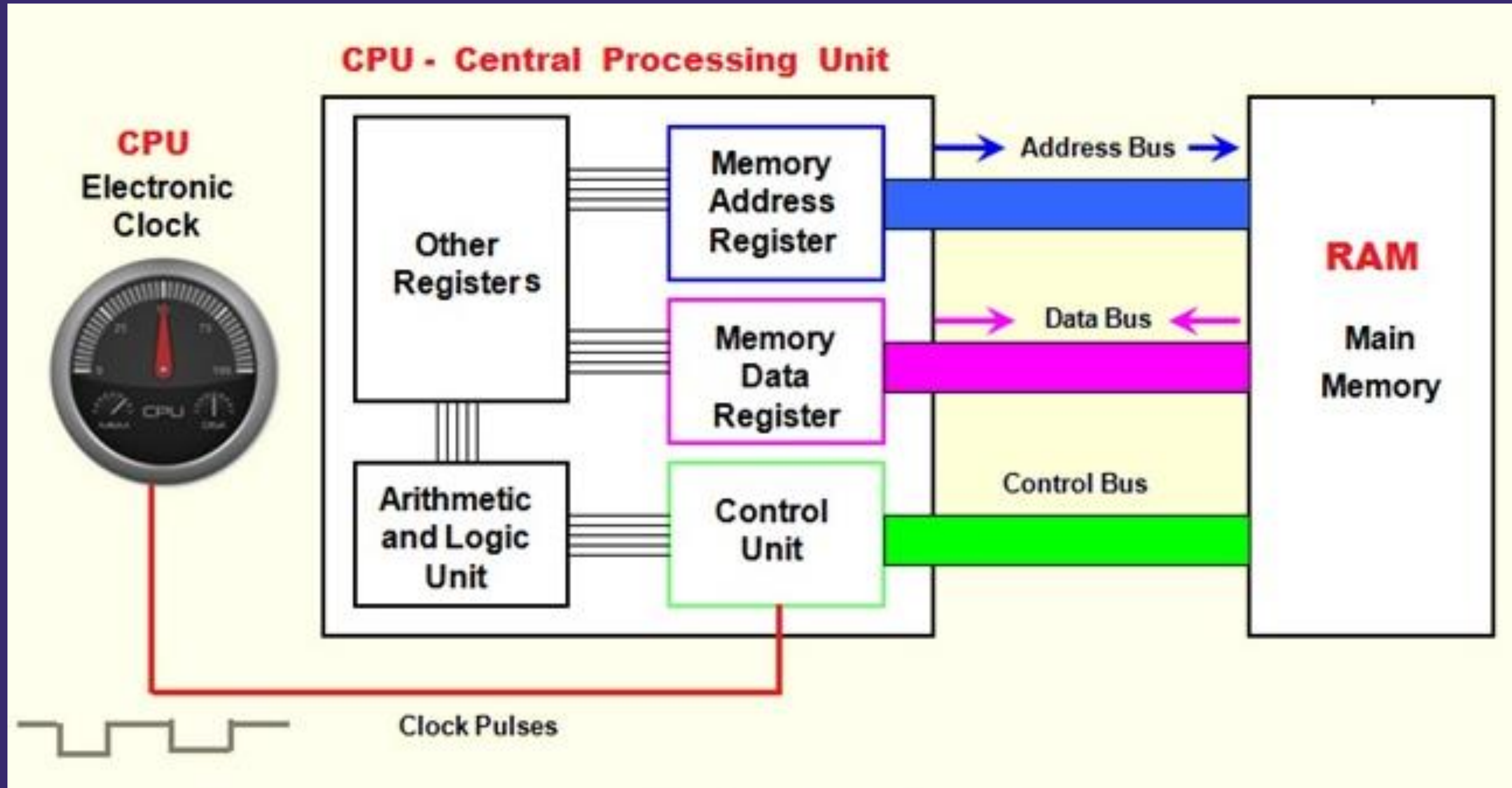
Python
C++
Java

Language Fit
for Purpose

Depolama Birimleri:
transistörler ->
mantık kapıları ->
mandallar ->
flip-floplar.







General Purpose Registers

AX	AH	AL	Accumulator Register
BX	BH	BL	Base Register
CX	CH	CL	Counter Register
DX	DH	DL	Data Register
SI			Source Index Register
DI			Destination Index Register
BP			Base Pointer Register
SP			Stack Pointer Register
IP			Instruction Pointer Register

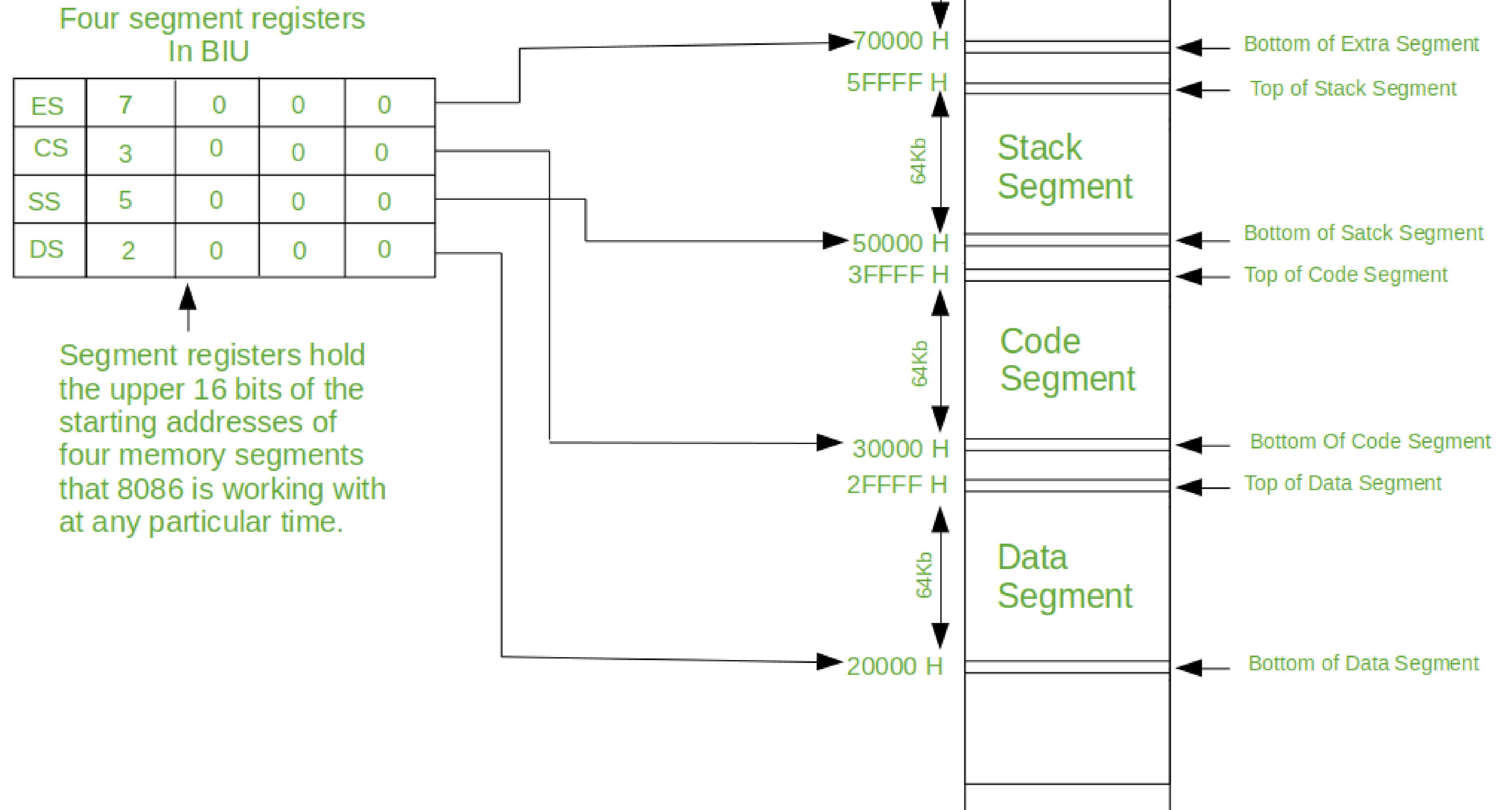
Segment Registers

CS		Code Segment Register
DS		Data Segment Register
ES		Extra Segment Register
SS		Stack Segment Register

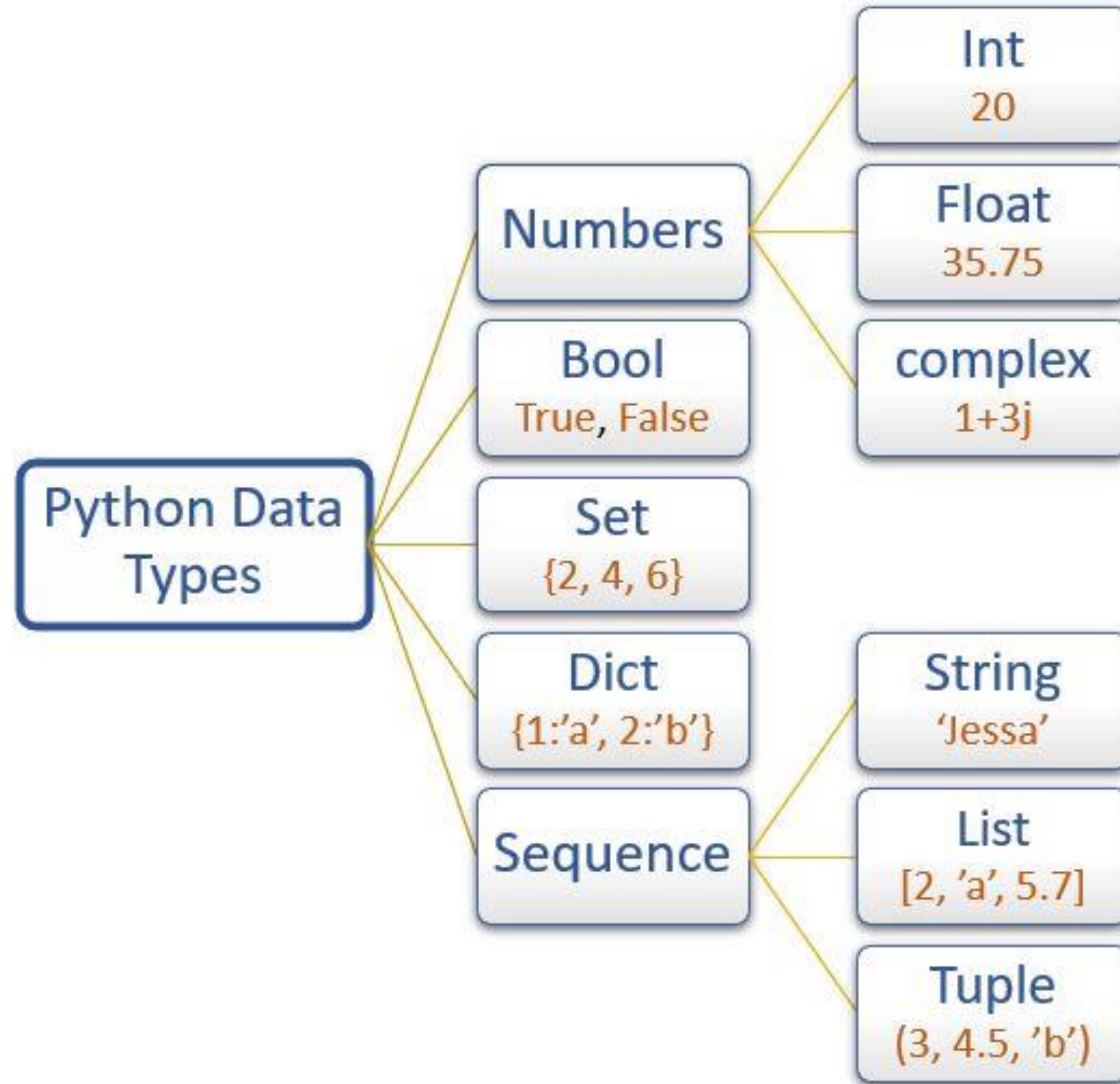
FLAGS					O	D	I	T	S	Z		A		P		C	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Flags Register

Posizione bit

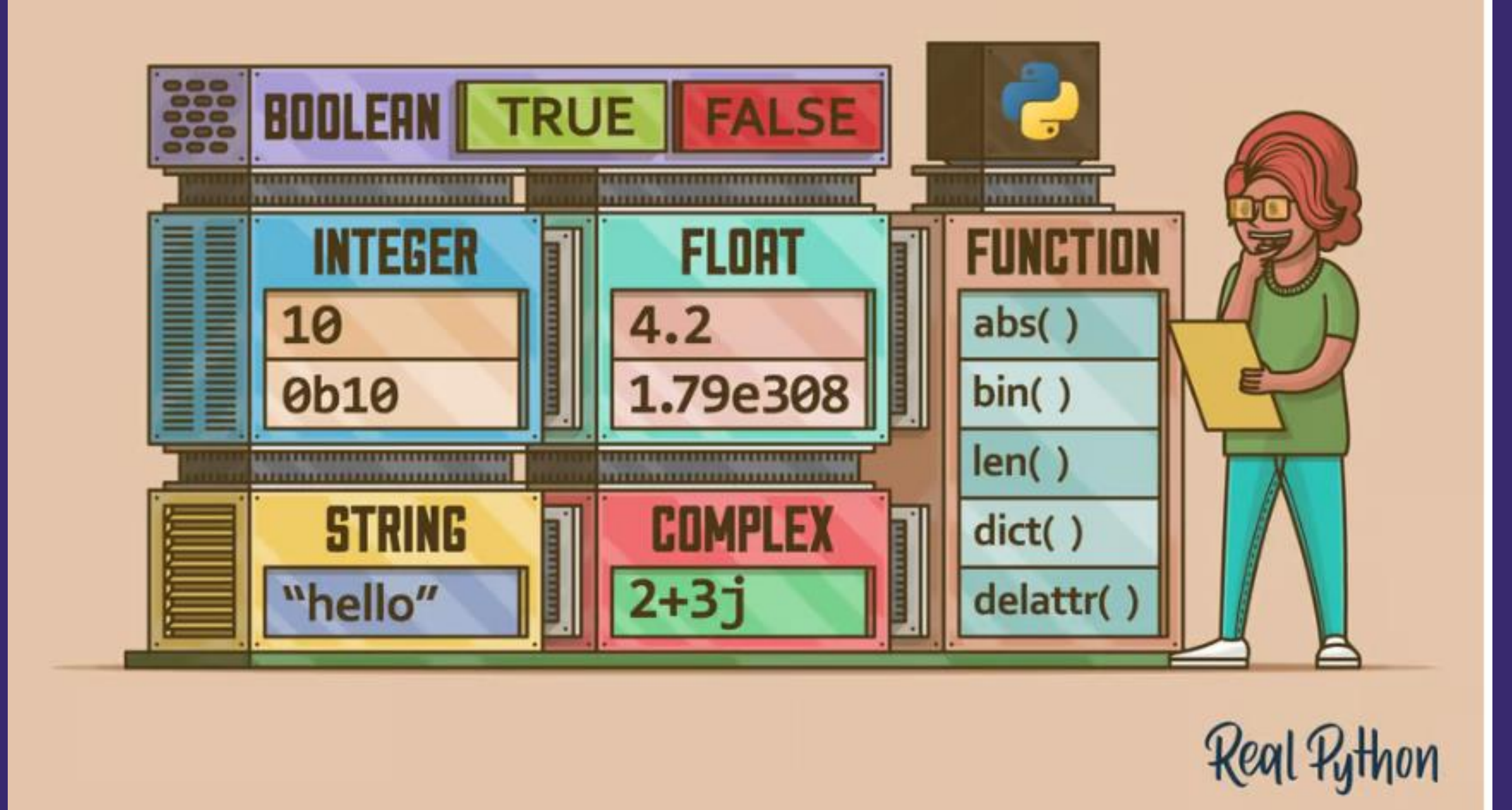


Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)



VERİ TİPLERİ

31



- Python programlarımızda geçici olarak veri saklamak için oluşturduğumuz alanlara değişken denir.
- Python'da değişkenler farklı türdeki verileri saklayabilir. Her veri türü, farklı işlemleri yapmamıza olanak tanır. Bu işlemler sayısal (toplama, çıkarma vb), metinsel (slicing) , listeleme(listeye eleman ekleme/çıkarma) ve mantıksal işlemler olabilir. Veri tipleri ise hangi işlemleri yapabileceğimizi söyler.
- Değişken tanımlama işlemi bellekte uygun bir alanın ayrılması (bu alan değişkenin veri tipine ve boyutuna göre değişir), değişkenin adlandırılması, değer atanması ve bu değere erişim sağlanması aşamalarını içerir. Bu sayede program, değişkenleri kullanarak veri saklayabilir ve işlemler gerçekleştirebilir.

- Python'da bir değişkenin hangi veri tipinde saklandığı bulmak **type()** metodu kullanılır.

```
x = 5.4|  
veri_tipi = type(x)  
print(veri_tipi)
```

```
<class 'float'>
```

- Bu kodda "x" adında bir değişken oluşturduk ve o değişkene "5.4" değerini atadık. Böylece, programın ilerleyen kısımlarında bu değere erişmek istediğimiz zaman "x" değişkenini kullanırız.

- Python'da **integer (int)** veri tipi tam sayıları ifade eder. Ondalık veya kesirli olmadan pozitif, negatif veya sıfır değerlerini alabilir.

```
x = 8  
print(type(x))  
  
<class 'int'>
```

- Python'da **float** veri tipi ondalıklı sayıları ifade eder. Pozitif veya negatif olabilir. Bilimsel sayıları tanımlarken genellikle float veri tipi kullanılır.

```
x = 2.488  
print(type(x))  
  
<class 'float'>
```

- **Complex** tip karmaşık sayıları ifade etmek için kullanılan bir nümerik veri tipidir. Karmaşık sayılar gerçek ve sanal kısımdan oluşur. Gerçek kısım integer ve float ile ifade edilirken sanal kısım "j" ile ifade edilir. Bu veri tipini bilimsel hesaplamalarda, sinyal işleme, görüntü işleme , matematiksel modelleme alanlarında kullanılır.

```
x = 4 - 5j  
print(type(x))
```

```
<class 'complex'>
```

- **Dictionary**, verileri key-value şeklinde saklamamızı sağlayan bir yapıdır. Kullanımı **key : value** şeklindedir. Dictionary'deki (kısaca dict.) değerlere key üzerinden erişim sağlarız. Dictionary'ler süslü parantez {} içerisinde tanımlanır. Ve key-value çifti virgüllerle ayrılır. Dictionary'ler diğer veri yapılarında olduğu gibi farklı veri tipleri içerebilir. Dictionary veri tipi değiştirilebilir (mutable) bir veri tipidir. Yani anahtar-değer çiftleri eklenebilir, güncellenebilir veya silinebilir. Anahtar kısmı için immutable (değiştirilemez) veri tipleri kullanılır. Çünkü anahtarlar benzersiz olmalıdır. Anahtar kısmında string, integer, float, boolean, tuple dictionary anahtarları olarak kullanılabilir. Değer kısmı için ise tüm veriler kullanılabilir.

```
my_dictionary = {"çilek" : 40 , "erik" : 50 , "ananas" : 100}

customers = {"irem" : [23, "adana"] , "ayça" : [23, "aydın"]}

colors = {
    ("red", "green", "blue"): "RGB",
    ("cyan", "magenta", "yellow"): "CMY",
    ("black", "white"): "BW"
}

print(type(colors))

<class 'dict'>
```

- **Boolean** veri tipi True ya da False olmak üzere sadece iki değere sahiptir.

```
print(type(True))
print(type(False))
```

```
<class 'bool'>
<class 'bool'>
```

- Python'da **set** veri tipi, benzersiz ve sırasız elemanların bir koleksiyonunu temsil eder. Setler, süslü parantezler içinde tanımlanır ve virgülle ayrılan öğelerden oluşur. Bir set, her bir elemandan yalnızca bir tane içerir. Setler, matematiksel küme işlemlerini destekleyen bir veri tipidir. İki veya daha fazla seti birleştirme, kesişimini bulma, farkını bulma gibi işlemleri gerçekleştirmek için setler kullanılabilir.

```
örnek_set = {"a", "b", "c", "d"}
print(type(örnek_set))
```

```
<class 'set'>
```

- Python'da "**strings**" (dizeler) metinsel verileri temsil etmek için kullanılan veri tipleridir. Bir string, bir veya daha fazla karakterin bir araya gelerek oluşturduğu bir sıralı veridir. Stringler, tek tırnak (' '), çift tırnak (" "), veya üçlü tırnak içinde tanımlanabilir.

```
say_hi = "hello world"  
say_hi = 'hello world'  
say_hi = """hello world  
            how are you?"""  
print(type(say_hi))
```

```
<class 'str'>
```

- Python'da "listeler" (**lists**), birden çok veri ögesini bir araya getirmek için kullanılan ve değiştirilebilen veri tipleridir. Elemanları ekleme, çıkarma ve değiştirme gibi işlemlerle değiştirilebilir. Listeler, köşeli parantezler içinde tanımlanır ve virgülle ayrılan öğelerden oluşur.

```
#boş liste oluşturma
my_list = []

#liste oluşturma
fruits = ['apple', 'banana', 'watermelon']
numbers = [1, 2, 3, 4]
mixed_list = ['hello', 88, False, [1,2,3]]

print(type(mixed_list))

<class 'list'>
```

- Python'da "**tuple**" (demet), birden çok veri ögesini bir araya getirmek için kullanılan ve değiştirilemez (immutable) veri tiplerinden biridir. Tuple'lar, parantezler içinde tanımlanır ve virgülle ayrılan öğelerden oluşur. Tuple'lar, listelere benzerdir ancak değiştirilemez özellikleri vardır, yani bir kez oluşturulduktan sonra öğelerini değiştiremezsiniz.

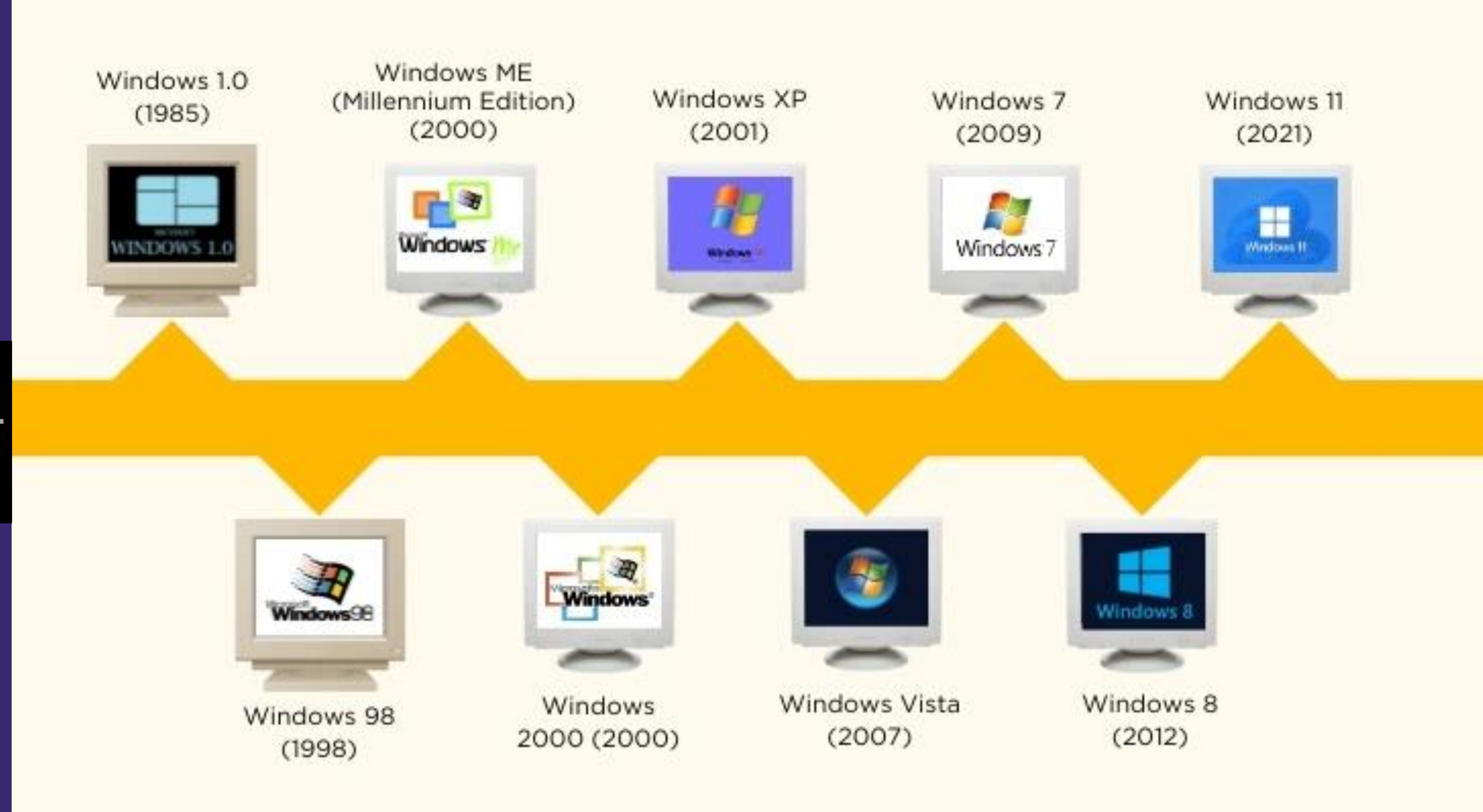
```
my_tuple = ("bird", 4.5, "a", 2, 9)
print(type(my_tuple))

<class 'tuple'>
```

MICROSOFT İŞLETİM SİSTEMİ TARİHİ

41

```
Starting MS-DOS...  
C:\>_
```



```
C:\WINDOWS\system32\cmd.exe
10/04/2007  04:51 PM    <DIR>          Start Menu
01/27/2003  03:15 PM    <DIR>          Templates
02/07/2003  02:35 PM    <DIR>          WINDOWS
                5 File(s)      238,543 bytes
                19 Dir(s)   47,378,472,960 bytes free

C:\Documents and Settings\kheintz>cd ..

C:\Documents and Settings>cd ..

C:\>dir
Volume in drive C is media 02
Volume Serial Number is BC2E-0ED8

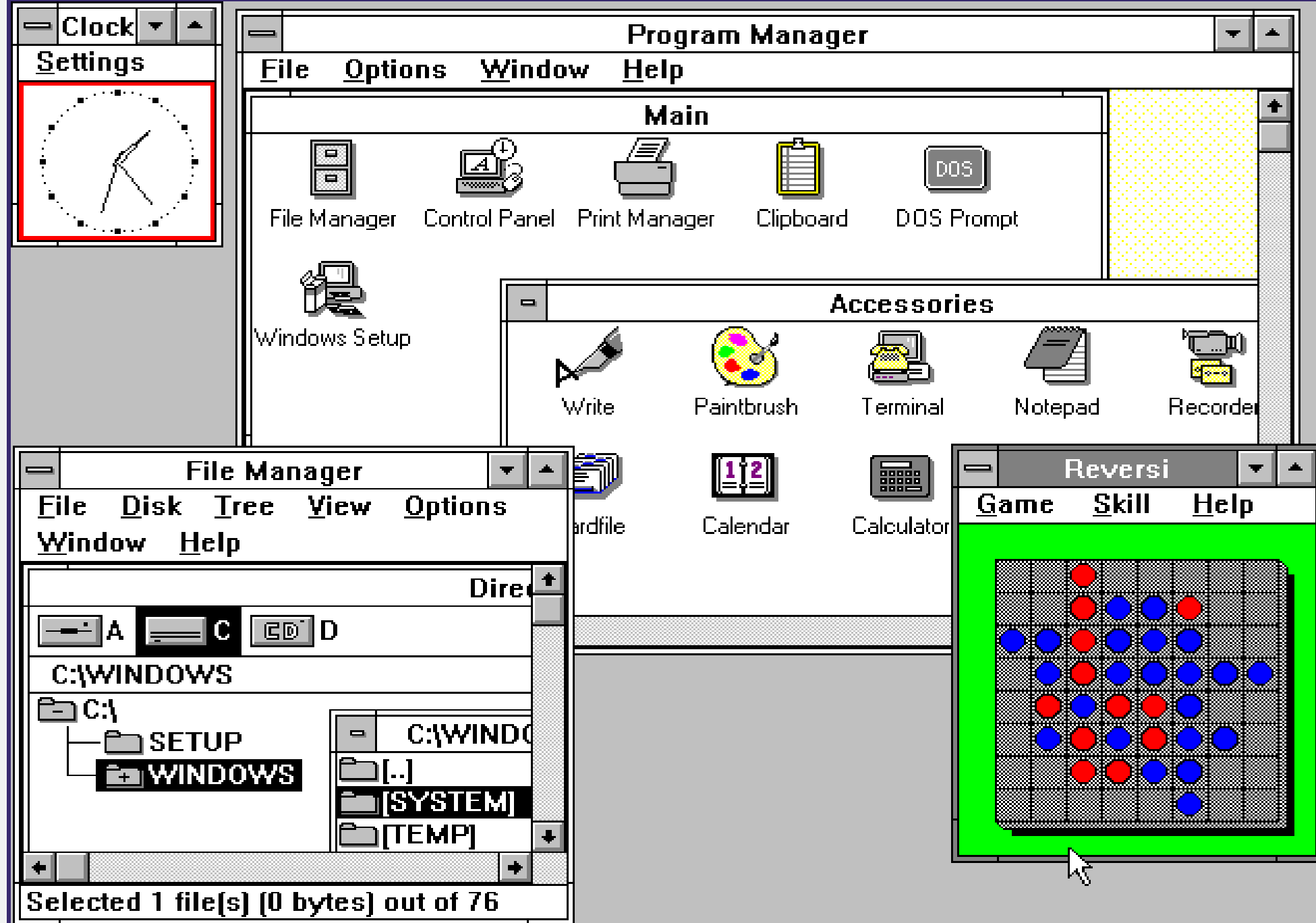
Directory of C:\

06/10/2004  03:59 PM                0 0000_
09/15/2003  10:01 AM            90 AUTOEXEC.BAT
02/07/2003  12:33 PM                2 autoexec.ixl
10/21/2008  01:18 PM    <DIR>          Batch_Upload
01/27/2003  03:19 PM                0 CONFIG.SYS
06/13/2005  02:58 PM            3,197 DEBUG.TXT
02/28/2005  02:47 PM    <DIR>          dell
01/30/2009  07:12 PM    <DIR>          divx
10/13/2008  03:26 PM    <DIR>          Documents and Settings
02/28/2005  02:48 PM    <DIR>          drvrtmp
04/20/2009  01:52 PM    <DIR>          ev97dos
02/11/2009  12:14 PM            330,347 fwlog.txt
02/10/2009  05:27 PM    4,194,402 fwlog.txt.old
05/30/2008  03:58 PM    <DIR>          KPCMS
03/09/2004  11:12 AM    <DIR>          Microtek
04/16/2009  05:59 PM    <DIR>          My Downloads
01/27/2003  05:19 PM    <DIR>          My Music
10/24/2005  01:42 PM    <DIR>          NPS
11/05/2008  04:43 PM    <DIR>          prntdrv
04/16/2009  06:10 PM    <DIR>          Program Files
12/09/2003  06:27 PM            56,710 service.txt
11/09/2005  11:33 AM    <DIR>          spoolerlogs
02/20/2008  10:45 AM    <DIR>          Temp
09/15/2003  10:01 AM                0 Tempfile.txt
04/16/2009  06:15 PM    <DIR>          WINDOWS
11/06/2003  02:17 PM    <DIR>          WUTemp
                9 File(s)      4,584,748 bytes
                17 Dir(s)   47,378,472,960 bytes free

C:\>
```

WINDOWS İŞLETİM SİSTEMİ

43



İLK PROGRAMLARIN YAPISI

44

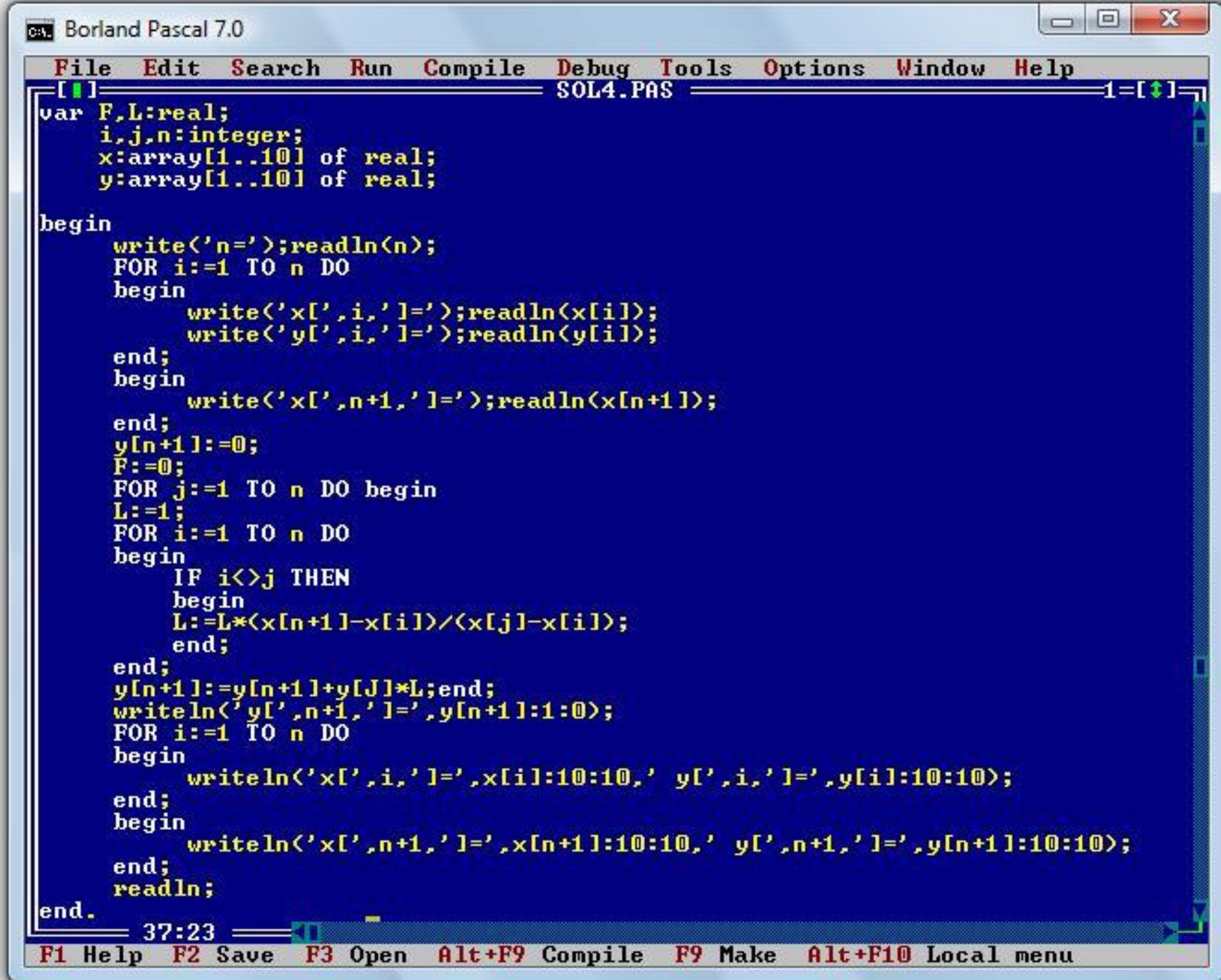
```

240 REM THE METROPOLIS-HASTINGS ALGORITHM
250 FOR I = 1 TO NSAMPLES - 1
260   FOR J = 0 TO NPARAMS - 1
270     PARAMS†(J) = SAMPLES†(I - 1, J)
280   NEXT J
285   REM
290   GOSUB 520
300   CURRDENS† = LOGDENS†
310
320   FOR J = 0 TO NPARAMS - 1
330     PROPOSAL†(J) = SAMPLES†(I - 1, J) + RND - .5
340     PARAMS†(J) = PROPOSAL†(J)
350   NEXT J
360   REM
370   GOSUB 520
380   PROPDENS† = LOGDENS†
390
400   IF RND < EXP(PROPDENS† - CURRDENS†) THEN
410     FOR J = 0 TO NPARAMS - 1
420       SAMPLES†(I, J) = PROPOSAL†(J)
430     NEXT J
440   ELSE
450     FOR J = 0 TO NPARAMS - 1
460       SAMPLES†(I, J) = SAMPLES†(I - 1, J)
470     NEXT J
480   END IF
490 NEXT I
500 RETURN

```

İLK PROGRAMLARIN YAPISI

45



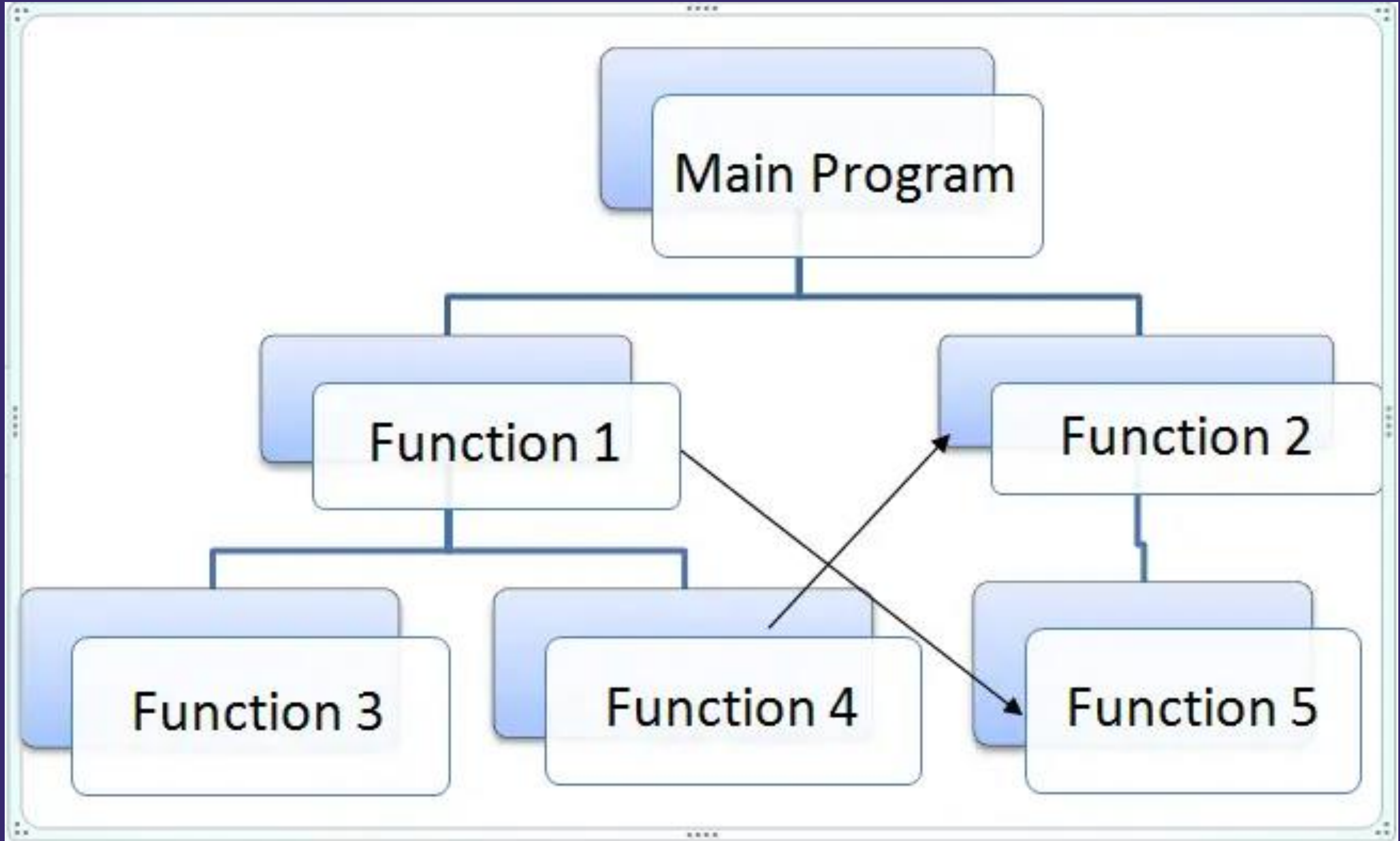
```
File Edit Search Run Compile Debug Tools Options Window Help
[1] SOL4.PAS 1=[1]
var F,L:real;
    i,j,n:integer;
    x:array[1..10] of real;
    y:array[1..10] of real;

begin
    write('n=');readln(n);
    FOR i:=1 TO n DO
    begin
        write('x[' ,i, ' ]=');readln(x[i]);
        write('y[' ,i, ' ]=');readln(y[i]);
    end;
    begin
        write('x[' ,n+1, ' ]=');readln(x[n+1]);
    end;
    y[n+1]:=0;
    F:=0;
    FOR j:=1 TO n DO begin
        L:=1;
        FOR i:=1 TO n DO
        begin
            IF i<>j THEN
            begin
                L:=L*(x[n+1]-x[i])/(x[j]-x[i]);
            end;
        end;
        y[n+1]:=y[n+1]+y[j]*L;end;
        writeln('y[' ,n+1, ' ]=',y[n+1]:1:0);
        FOR i:=1 TO n DO
        begin
            writeln('x[' ,i, ' ]=',x[i]:10:10,' y[' ,i, ' ]=',y[i]:10:10);
        end;
        begin
            writeln('x[' ,n+1, ' ]=',x[n+1]:10:10,' y[' ,n+1, ' ]=',y[n+1]:10:10);
        end;
        readln;
    end.
37:23
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu
```

```
1 def calculate(operator, number1, number2):
2     if operator == "+":
3         return number1 + number2
4     elif operator == "-":
5         return number1 - number2
6     elif operator == "*":
7         return number1 * number2
8     else:
9         raise Exception(f"Unknown operator: {operator}")
```

İLK PROGRAMLARIN YAPISI

47



```
PROGRAM MaxFunction;
```

```
VAR A, B, C, D, G: INTEGER;
```

```
FUNCTION Maximum(X, Y: INTEGER)  
    : INTEGER;
```

```
BEGIN
```

```
    IF X > Y THEN
```

```
        Maximum := X
```

```
    ELSE
```

```
        Maximum := Y;
```

```
    END; { Maximum }
```

```
BEGIN
```

```
    WriteLn('Enter four values');
```

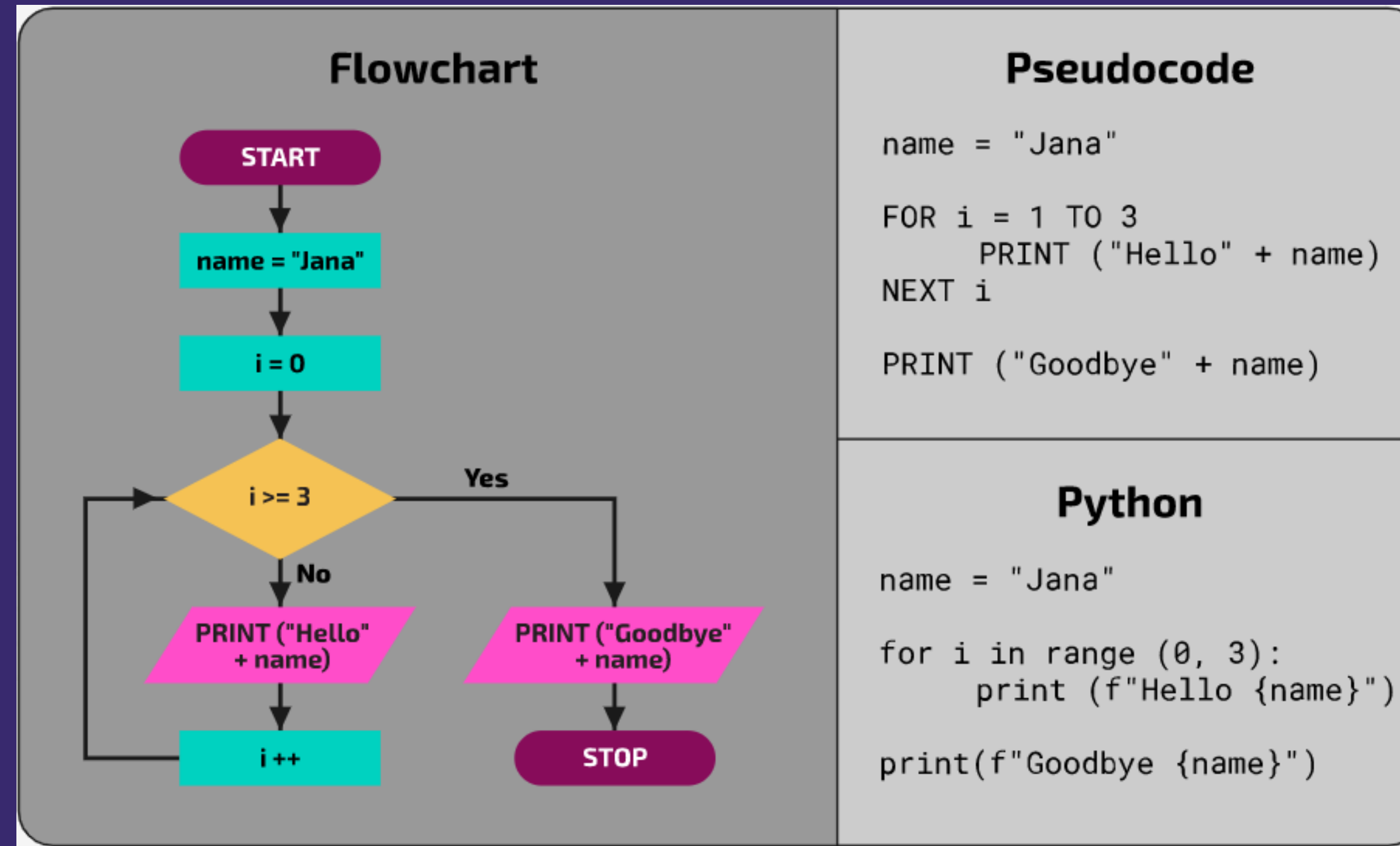
```
    ReadLn(A, B, C, D);
```

```
    G := Maximum(Maximum(A, B),  
                  Maximum(C, D));
```

```
    WriteLn('The maximum is ', G:3);
```

```
END. { MaxFunction }
```

- Bilgisayar, sadece programcının kendisine söylediği şeyi nasıl yapacağını bilir.
- Bundan dolayı programcı bilgisayara problemi nasıl çözeceğini bildirmelidir.
- Bilgisayara nasıl iş yaptıracak, nasıl iletişim kuracaksınız? Bir "Program" ile. Bilgisayarlar program olmadan çalışmazlar.
- Bilgisayarın dili makine dilidir. Onunla makine mantığı ile iletişim kurabiliriz. Bu da **ALGORİTMA** (talimat, rutin, reçete) ile olur.



- Program yazma, çözülmüş bir problemin bir bilgisayar dili ile kodlarını yazmaktır (coding). Bu, en son iştir.
- Kodlamadan önce algoritma yazılmalıdır.

İki Sayının Toplamasını Bulan Algoritma

- 1.Adım: Başla
- 2.Adım: 1.Sayıyı girin
- 3.Adım: 2.Sayıyı girin
- 4.Adım: Sayıları topla
- 5.Adım: Sonucu ekrana yaz
- 6.Adım: Bitir

$$10 + 5 = 15$$



İki Sayının Toplamasını Bulan Bilgisayar Algoritması

- 1.Adım: Başla
- 2.Adım: 1.Sayıyı girin, Sayı1
- 3.Adım: 2.Sayıyı girin, Sayı2
- 4.Adım: Sonuç = Sayı1 + Sayı2
- 5.Adım: Sonuç'u ekrana yaz
- 6.Adım: Bitir

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

Polya, George (1957) 'How To Solve It', Princeton University Press, 2nd Edition

Problem Çözme Aşaması	Problemin tanımlanması
	Çözümün ana hatlarının ortaya konulması
	Ana hatlara bağlı bir algoritma geliştirilmesi
	Algoritmanın doğruluğunun sınanması
Gerçekleştirim Aşaması	Algoritma kodları belirli bir programlama diline dönüştürülür.
	Program bilgisayarda çalıştırılır.
	Program belgelemesi ve bakımı yapılır.

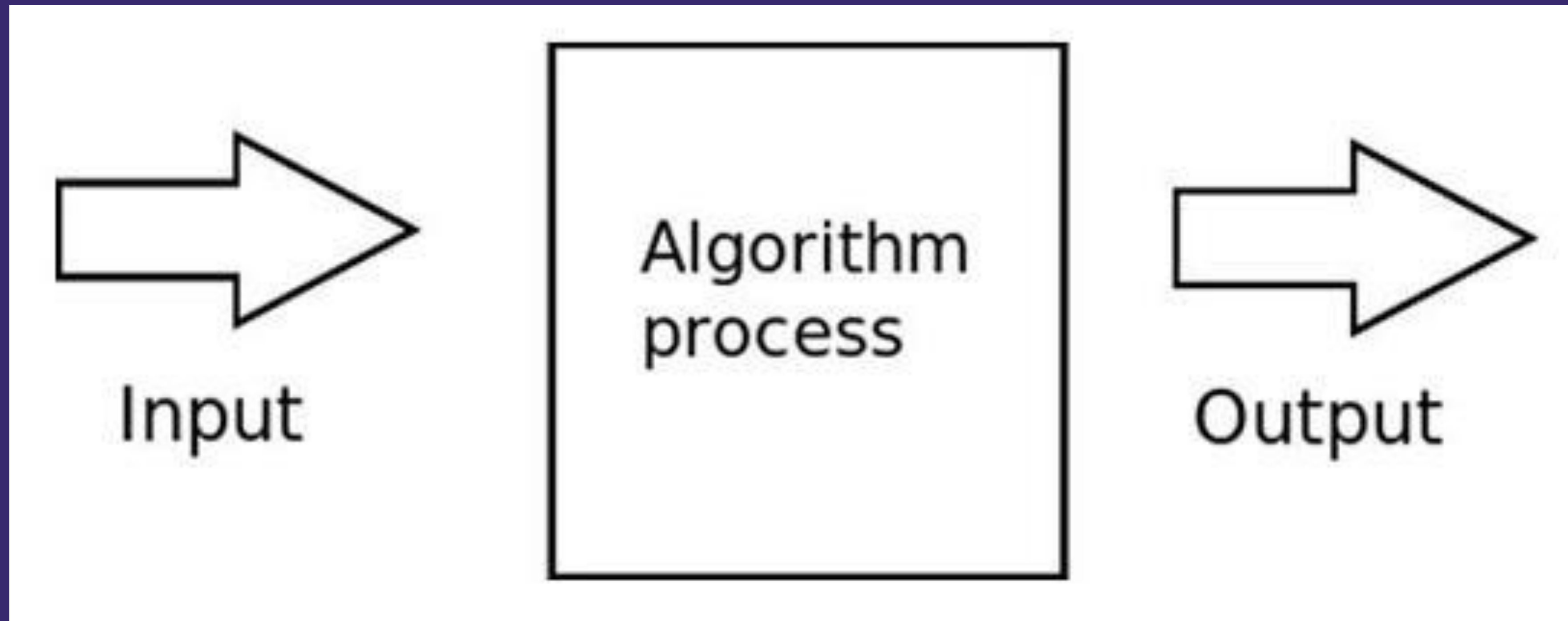
- **İstenenler analizi (Requirements Analysis):** Problemi bütün boyutları ile tanıma ve ne yapılması istendiğini net olarak anlama, problem çözülebilir mi?
- **Çözüm yolu bulma (Design):** Bir çözüm yolu geliştirme ve algoritmasını yazma.
- **Uygulama (Implementation):** Çözüm algoritmasını uygun bir program diline çevirme
- Test etme, hatalardan ayıklama, kurma ve bakımını yapma (Testing, debugging, installing and maintenance)

➤ **Örnek:** 2 saatte 190 km yol alan arabanın ortalama hızı nedir?

Input: gidilen mesafe ve süre

Process: $ortalama\ hız = \frac{mesafe}{süre}$

Output: saatte ortalama hız



Algoritma kelimesi, “Hesaplamalarda veya diğer problem çözme işlemlerinde uyulması gereken sonlu kurallar veya talimatlar kümesi” veya “Sık sık yinelemeli işlemler içeren sonlu sayıda adımda bir matematik problemini çözme yöntemi” anlamına gelir .

El-Harizmi. “Kitab el-cebr ve'l-mukabele” (Cebir denklemlerini çözmenin kuralları)

Doğru çözüm için doğru sırada atılması gereken adımlar:

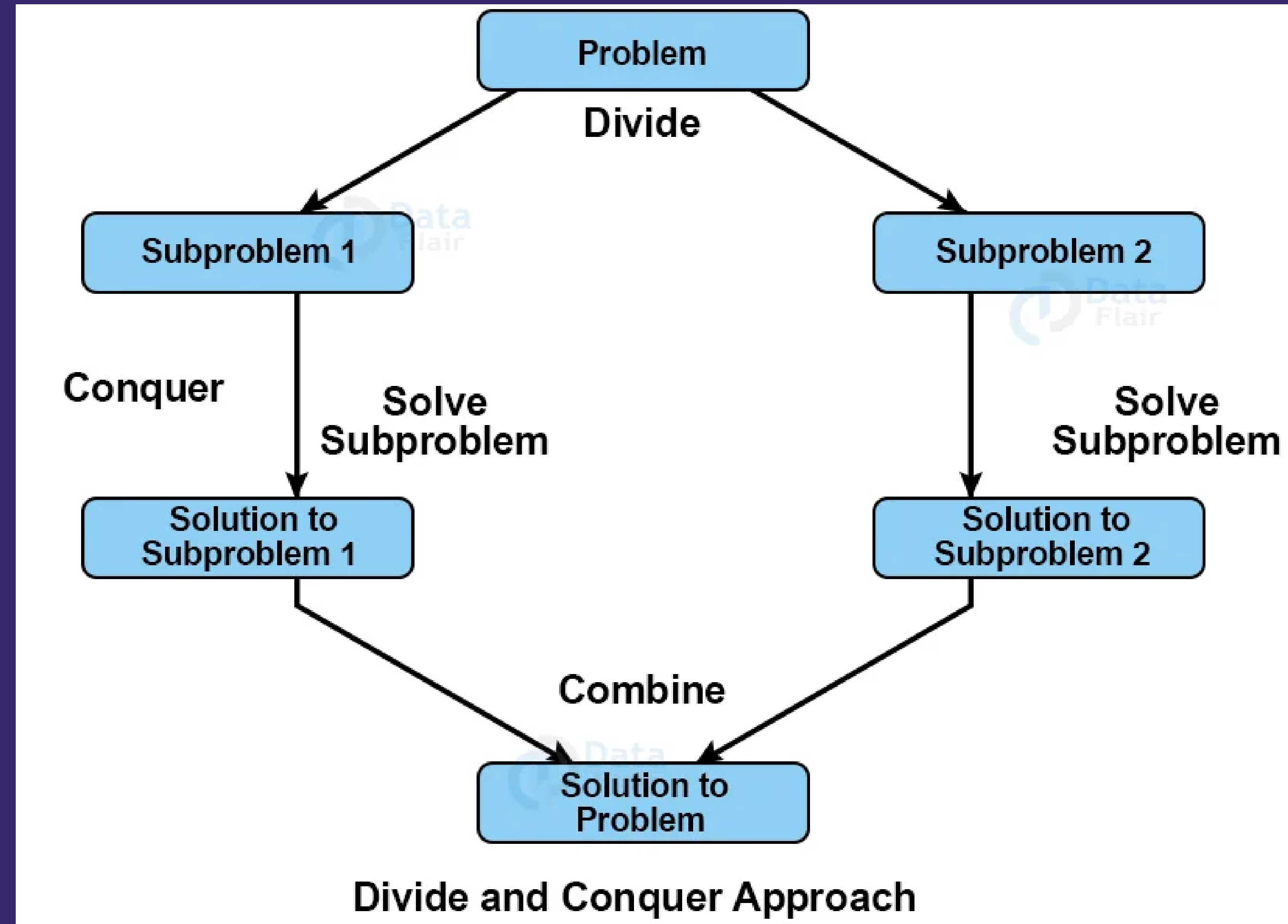
- Başlangıç ve bitişi belli
- Sınırlı sayıda adım. Ne kadar uzun olursa olsun mutlaka bir sonu olmalı.
- Her adımda yapılacak işlemler açık ve kesin olmalı

Algoritmaları bilgisayar programına çeviren üç teknik vardır:

- **Akış diyagramı (Flowcharts)**- Bir işi yapan mantıksal adımları ve adımlar arası geçişlerin grafiksel gösterimi.
- **Satır algoritma (Pseudocode)** – Programın ana hatlarını ve adımlarını gösteren kısa komutlar.
- **Sıradüzeni grafiği (Hierarchy charts)** - Programın farklı parçalarının birbirleriyle ilişkilerini gösterir.



- Çok büyük problemleri parçalara ayırarak daha kolay çözeriz.
- Çözülecek kadar küçülttüğün parçaları mantıklı bir sıraya koy.
- Her parçayı ayrı modüller olarak çöz ve birbirine bağla (entegre olsun)



- **Sıralı (Sequence) yapılar** – işlem hiç satır atlamadan doğrusal bir sıra ile olur
- **Karar (Decision) yapıları** – İçinde “evet”-“hayır” soruları soran ve cevaba göre farklı yönde farklı işlem yapan yapılar
- **Döngü (Looping) yapıları** – Bir şart gerçekleşinceye kadar tekrar tekrar yapılan işlemler
- **Koşulsuz atlama** (GOTO Programın bir yerinden başka satıra atlama unconditional branch. Bu yapısal programlamada yok.

**KATILIMINIZ VE DİKKATİNİZ İÇİN
TEŞEKKÜR EDERİM.**