Computer Programming Languages

A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. Most programming languages consist of instructions for computers.

Different Programming Paradigms:

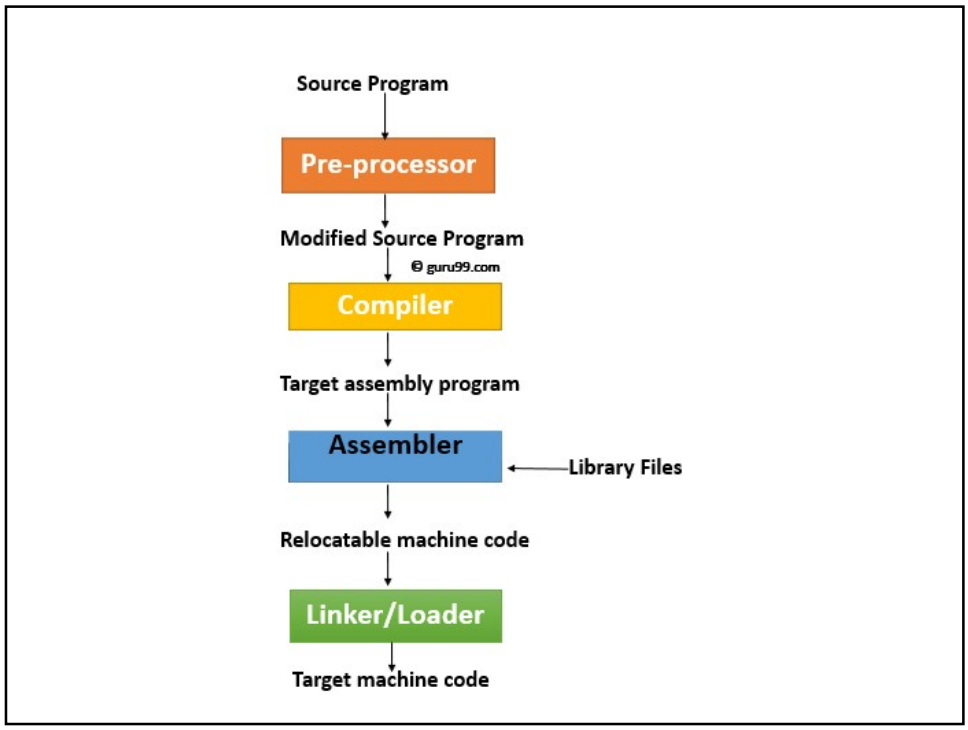
Based on the **programming** philosophy, style, or general approach to writing code there different programming paradigms such as:

- imperative in which the programmer instructs the machine how to change its state,
- procedural which groups instructions into procedures, ← ***(OUR CURRENT FOCUS)***
- object-oriented which groups instructions together with the part of the state they operate on,
- declarative in which the programmer merely declares properties of the desired result, but not how to compute it
- functional in which the desired result is declared as the value of a series of function applications,
- logic in which the desired result is declared as the answer to a question about a system of facts and rules,
- mathematical in which the desired result is declared as the solution of an optimization problem
- Symbolic techniques such as reflection, which allow the program to refer to itself, might also be considered as a programming paradigm. However, this is compatible with the major paradigms and thus is not a real paradigm in its own right

Classification of Programming Languages

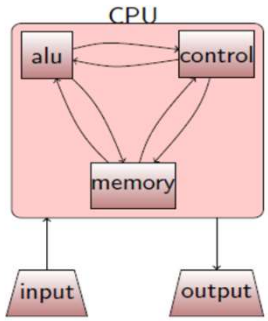
The programming language in terms of their performance reliability and robustness can be grouped into five different generations:

- **First generation** languages (**1GL**)
  - Low-level languages that are machine language.
- **Second generation** languages (**2GL**)
  - Low-level assembly languages. They are sometimes used in kernels and hardware drives, but more commonly used for video editing and video games.
- Third generation languages (**3GL**)
  - High –Level languages, such as C, C++, Java, JavaScript, and Visual Basic.
- Fourth generation languages (**4GL**)
  - languages that consist of statements similar to statements in a human language. Fourth generation languages are commonly used in database programming and scripts examples include Perl, PHP, Python, Ruby, and SQL.
- **Fifth generation** languages (**5GL**)
  - programming languages that contain visual tools to help develop a program. Examples of fifth generation languages include Mercury, OPS5, and Prolog



Quick Look at 1GL & 2GL

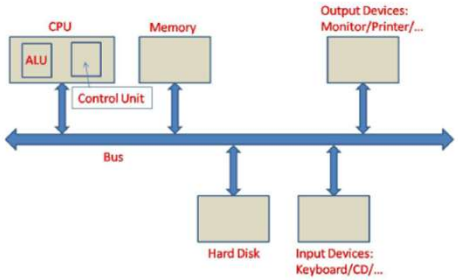
von Neumann architecture



Basic instructions

- Read/write data from memory/IO devices,
- Perform arithmetic logical operations on data,
- Read instructions from memory and interpret them.
- Proceed according to the instructions.

Functional Units



CPU: Central Processing Unit (ALU: Arithmetic and Logic Unit; Control Unit: Executes instructions)

Memory: Storage area; quickly accessible from CPU

Hard Disk: Storage area is not so quickly accessible to CPU

Binary Format

- In a computer, everything is stored in binary format: a sequence of 0's and 1's.
- The components of a computer understand only binary format.
- Number 4 is stored as 00000100, 1 is stored as 00000001 etc.

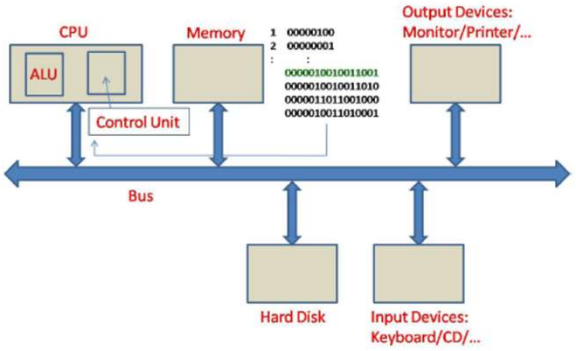
Execution in a Computer

- To begin with, all the data and commands related to a computation is stored in Memory.
- Commands are then brought into the CPU through the Bus, one at a time.
- Each command is executed inside the CPU in the following way:
  - If the command requires data, it is brought to CPU from Memory
  - Command is then executed using the data
  - The command may be for storing data present inside the CPU to Memory
- A program is a collection of commands.

A Small Program

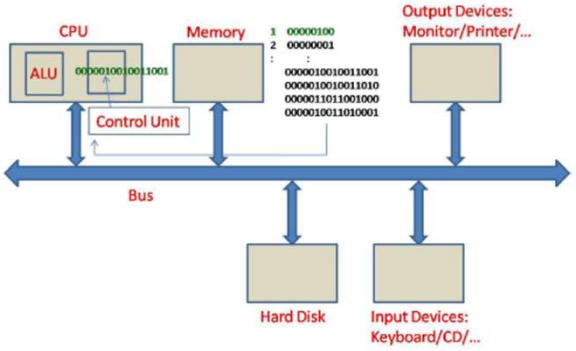
0000010010011001 - read memory location 001
0000010010011010 - read memory location 010
0000011011001000 - add two numbers read
0000010011010001 - store the result in memory location 001

EXECUTION



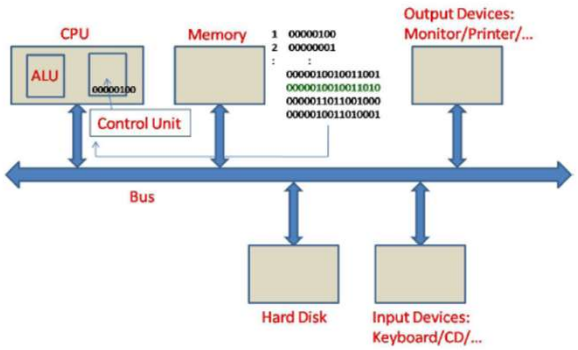
Step 1: Bring 0000010010011001 to CPU
0000010010011001 = Bring data stored in memory location 001 to CPU

EXECUTION



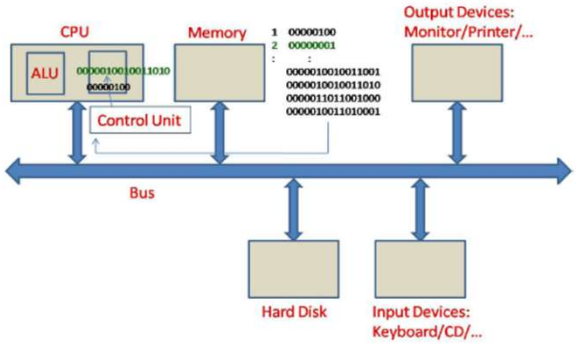
Step 2: Execution of 0000010010011001 in Control Unit
Brings data stored in location 1, number 4, to CPU

EXECUTION



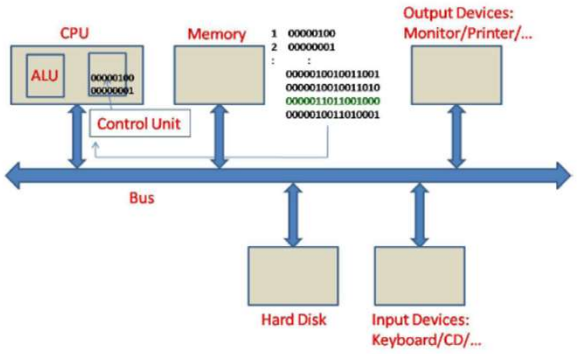
Step 3: Bring 0000010010011010 to CPU
0000010010011010 = Bring data stored in memory location 010 to CPU

EXECUTION



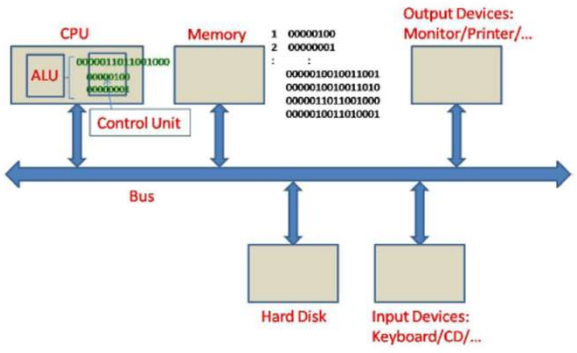
Step 4: Execution of 0000010010011010 in Control Unit
Brings data stored in location 2, number 1, to CPU

EXECUTION



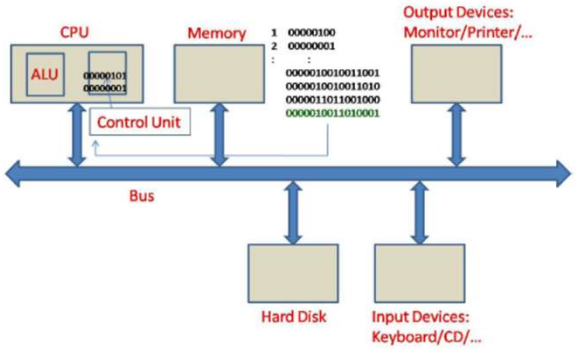
Step 5: Bring 0000011011001000 to CPU
0000011011001000 = Add two stored numbers

EXECUTION



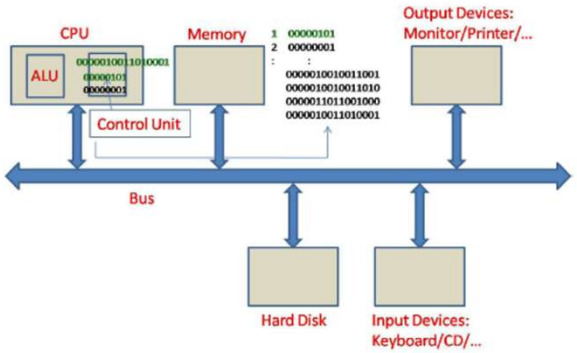
Step 6: Execution of 0000011011001000 in Control Unit
Adds two numbers inside CPU using ALU

EXECUTION



Step 7: Bring 0000010011010001 to CPU
0000010011010001 = Store number in ALU to memory location 001

EXECUTION

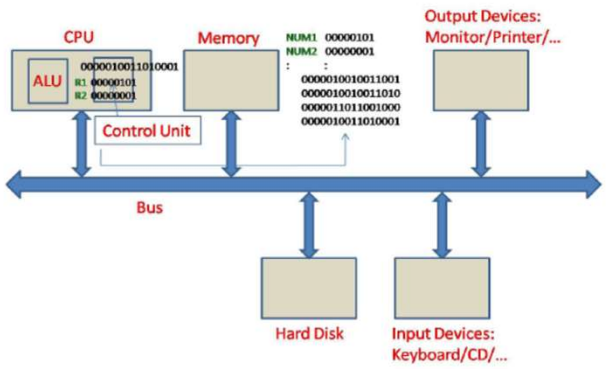


Step 8: Execution of 0000010011010001 in Control Unit
Stores data stored in ALU , number 5, in location 1

Assembly Language

- It is very difficult for us to understand this binary language, called machine language!
- And this is the only language that computers understand!!
- To make it more readable, assembly language was introduced.
- In assembly language, operations and memory locations are addressed by names.

Naming Locations in Example Program



Named Memory and CPU Locations

Example Program in Assembly Language

```
0000010010011001      MOVE  NUM1, R1
0000010010011010      MOVE  NUM2, R2
0000011011001000      ADD   R1, R2
0000010011010001      MOVE  R1, NUM1
```

Move contents of memory location NUM1 to CPU register R1
Move contents of memory location NUM2 to CPU register R2
Add contents of R1 and R2 and store the result in R1
Move the contents of R1 to memory location NUM1

Assemblers

- An assembly language program eventually must be translated to machine language.
- An assembler does this job.
- It maps the names to the corresponding binary values.
- It is also a program!

I/O

- The example only shows how to add numbers already present in the Memory.
- How does one add numbers provided by the user through the keyboard?
- This is the job of another program, called the Operating System (OS in short).

OS

- OS picks the input given by the user and stores it in appropriate locations of Memory.
- It also picks result of computations from Memory and displays to the user.
- It does many other housekeeping jobs that make the interaction of user with the computer easy.
- Examples of OS: Linux, Windows, Mac OS.

Need for a Better Language

- It is very difficult to write large programs in assembly language.
- Several language were created during 1960-80 to simplify the task of the programmer.
- The prominent ones are: COBOL, Fortran, Pascal, C.
- These are called High-level programming languages.
- Assembly and machine language are called low level programming languages.

Lec-4: Introduction to C

Summary of Previous Lecture

- **Classification of Programming Languages**
- **1GL to 5 GL**
- **Look at 1GL/2GL code**

Example Program in Assembly Language

```
0000010010011001      MOVE  NUM1, R1
0000010010011010      MOVE  NUM2, R2
0000011011001000      ADD   R1, R2
0000010011010001      MOVE  R1, NUM1
```

Move contents of memory location NUM1 to CPU register R1
Move contents of memory location NUM2 to CPU register R2
Add contents of R1 and R2 and store the result in R1
Move the contents of R1 to memory location NUM1

Lec-4: Introduction to C

Outline of Today'sLecture

- **Startup on Syntax and Semantics of C-Language**
- **Discussion on "Welcome.c"**
- **Basic Data Types**
- **1GL Vs 2GL Vs 3GL**
- **C-Program to discuss Basic Data Types**

Syntax and Semantics of introductory C-progam

Lets start with a welcoming note C-Program

Lets print the words - " Welcome to the Course MTH-409a" using C

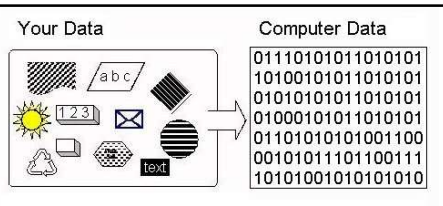
```
#include<stdio.h>
main()
  {
     printf(" Welcome to the Course MTH-409a \n");
  }
```

- Now save it in a file called: welcome.c
- Compile it , say using gcc as follows:
  gcc welcome.c
- On successful compilation "a.out" will be generated.
- On execution from command line prompt i.e.
  > a.out ()
- We will see:
  Welcome to the Course MTH-409a

Note: You will be using PRUTOR environment for programming and execution.

Discussion on "welcome.c"

#include<stdio.h>	•Include I/O library during compilation
main()	•Define a function called main() that receives NO Input
{	•Statements of main are enclosed in braces
printf(" Welcome to the Course MTH-409a \n");	•main() calls a function call printf() to print a sequence of characters. •\n denotes a new line character •";" denotes the end of a line
}	



Input

01001001 01110100 00100000 01101001 01110011 00100000 01110010 01100001
01101001 01101110 01101001 01101110 01100111 00100000 01110100 01101111
01100100 01100001 01111001 00101110 00100000 01001001 01110100 00100000
01101101 01110101 01110011 01110100 00100000 01100010 01100101 00100000
01110111 01101001 01101110 01110100 01100101 01110010 00100000 01101001
01101110 00100000 01010110 01100001 01101110 01100011 01101111 01110101
01110110 01100101 01110010 00111111

Formula Test

It is raining today. It must be winter in Vancouver?

Custom Function Test

It is raining today. It must be winter in Vancouver?

Converting the text "hope" into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

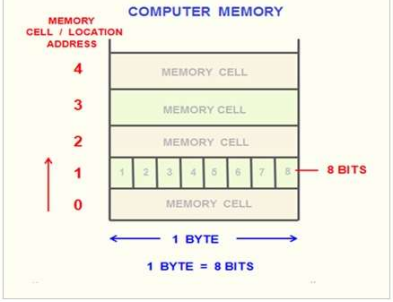
ASCII Desc	Decimal	Hexadecimal	Octal	Binary	fBinary	Char
>	62		76	111110	00111110	>
?	63		77	111111	00111111	?
@	64	40	100	1000000	01000000	@
A	65	41	101	1000001	01000001	A
B	66	42	102	1000010	01000010	B
C	67	43	103	1000011	01000011	C
D	68	44	104	1000100	01000100	D
E	69	45	105	1000101	01000101	E
F	70	46	106	1000110	01000110	F
G	71	47	107	1000111	01000111	G
H	72	48	110	1001000	01001000	H
I	73	49	111	1001001	01001001	I
J	74		112	1001010	01001010	J

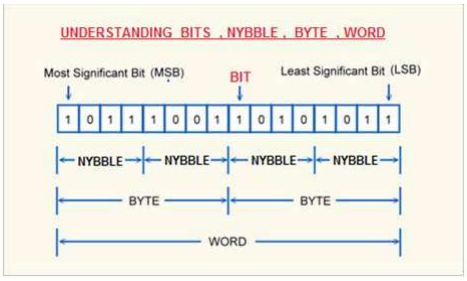
Sheet1!_Binary (Number)

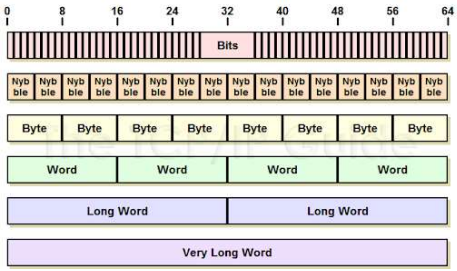
Random Access Memory (RAM)

- Content of a 1024 x 16-bit memory:

Memory address		Memory content
binary	decimal	
0000000000	0	1011010111011101
0000000001	1	1010000110000110
0000000010	2	0010011101110001
:	:	:
:	:	:
1111111101	1021	1110010101010010
1111111110	1022	0011111010101110
1111111111	1023	1011000110010101

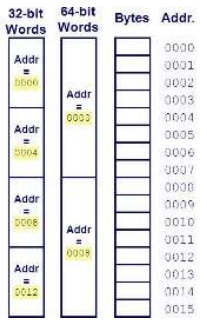






Addresses Specify Byte Locations

- Address of first byte in word
- Addresses of successive words differ by 4 (32-bit) or 8 (64-bit)



MEMORY CHIP REPRESENTATION

64K X 8

This indicates the number of cells in the memory chip i.e. 64K cells(here)

This indicates the size of the Cell (the number of bits that can be stored in the Cell) i.e. 8 bits(here)

Memory Capacity Conversion Chart

Term (Abbreviation)	Approximate Size
Byte (B)	8 bits
Kilobyte (KB)	1024 bytes / 10^3 bytes
Megabyte (MB)	1024 KB / 10^6 bytes
Gigabyte (GB)	1024 MB / 10^9 bytes
Terabyte (TB)	1024 GB / 10^{12} bytes
Petabyte (PB)	1024 TB / 10^{15} bytes
Exabyte (EB)	1024 PB / 10^{18} bytes
Zettabyte (ZB)	1024 EB / 10^{21} bytes
Yottabyte (YB)	1024 ZB / 10^{24} bytes

American Standard Code for Information Interchange (ASCII)

- Character encoding is the American Standard Code for Information Interchange, and is the US precursor to ISO 646 (internationally defined character sets).
- ASCII is a 7-bit code, meaning that 128 characters (27) are defined.
- The code consists of 33 non-printable and 95 printable characters and includes both letters, punctuation marks, numbers and control characters.
- **Each character corresponds to a seven-digit sequence of zeroes and ones, which can then be represented as a decimal number, or as a hexadecimal number. The ASCII characters can be divided into several groups.**
- **Control Characters (0–31 & 127): Control characters are not printable characters. They are used to send commands to the PC or the printer and are based on telex technology. With these characters, you can set line breaks or tabs. Today, they are mostly out of use.**
- **Special Characters (32–47 / 58–64 / 91–96 / 123–126): Special characters include all printable characters that are neither letters nor numbers. These include punctuation or technical, mathematical characters. ASCII also includes the space (a non-visible but printable character), and, therefore, does not belong to the control characters category, as one might suspect.**
- **Numbers (30–39): These numbers include the ten Arabic numerals from 0-9.**
- **Letters (65–90 / 97–122): Letters are divided into two blocks, with the first group containing the uppercase letters and the second group containing the lowercase.**

ASCII table: An overview of all ASCII codes

Bin.	Hex.	Dec.	ASCII Symbol	Explanation	Group
0000000	0	0	NUL	The null character prompts the device to do nothing	Control Character
0000001	1	1	SOH	Initiates a header (Start of Heading)	
0000010	2	2	STX	Ends the header and marks the beginning of a message. (start of text)	
0000011	3	3	ETX	Indicates the end of the message (end of text)	
0000100	4	4	EOT	Marks the end of a completes transmission (End of Transmission)	
0000101	5	5	ENQ	A request that requires a response (Enquiry)	
0000110	6	6	ACK	Gives a positive answer to the request (Acknowledge)	
0000111	7	7	BEL	Triggers a beep (Bell)	
0001000	8	8	BS	Lets the cursor move back one step (Backspace)	
0001001	9	9	TAB (HT)	A horizontal tab that moves the cursor within a row to the next predefined position (Horizontal Tab)	
0001010	A	10	LF	Causes the cursor to jump to the next line (Line Feed)	
0001011	B	11	VT	The vertical tab lets the cursor jump to a predefined line (Vertical Tab)	
0001100	C	12	FF	Requests a page break (Form Feed)	
0001101	D	13	CR	Moves the cursor back to the first position of the line (Carriage Return)	
0001110	E	14	SO	Switches to a special presentation (Shift Out)	
0001111	F	15	SI	Switches the display back to the normal state (Shift In)	
0010000	10	16	DLE	Changes the meaning of the following characters (Data Link Escape)	
0010001	11	17	DC1	Control characters assigned depending on the device used (Device Control)	
0010010	12	18	DC2		
0010011	13	19	DC3		
0010100	14	20	DC4		
0010101	15	21	NAK	Negative response to a request (Negative Acknowledge)	
0010110	16	22	SYN	Synchronizes a data transfer, even if no signals are transmitted (Synchronous Idle)	
0010111	17	23	ETB	Marks the end of a transmission block (End of Transmission Block)	
0011000	18	24	CAN	Makes it clear that a transmission was faulty and the data must be discarded (Cancel)	
0011001	19	25	EM	Indicates the end of the storage medium (End of Medium)	
0011010	1A	26	SUB	Replacement for a faulty sign (Substitute)	
0011011	1B	27	ESC	Initiates an escape sequence and thus gives the following characters a special meaning (Escape)	
0011100	1C	28	FS	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit.(File Separator, Group Separator, Record Separator, Unit Separator)	
0011101	1D	29	GS		
0011110	1E	30	RS		
0011111	1F	31	US		
0100000	20	32	SP	Blank space (Space)	Special Character
0100001	21	33	!	Exclamation mark	
0100010	22	34	"	Only quotes above	
0100011	23	35	#	Pound sign	
0100100	24	36	$	Dollar sign	
0100101	25	37	%	Percentage sign	
0100110	26	38	&	Commericial and	
0100111	27	39	'	Apostrophe	
0101000	28	40	(	Left bracket	
0101001	29	41	)	Right bracket	
0101010	2A	42	*	Asterisk	
0101011	2B	43	+	Plus symbol	
0101100	2C	44	,	Comma	
0101101	2D	45	-	Dash	
0101110	2E	46	.	Full stop	
0101111	2F	47	/	Forward slash	
0110000	30	48	0		Numbers
0110001	31	49	1		
0110010	32	50	2		
0110011	33	51	3		
0110100	34	52	4		
0110101	35	53	5		
0110110	36	54	6		
0110111	37	55	7		
0111000	38	56	8		
0111001	39	57	9		
0111010	3A	58	:	Colon	Special characters
0111011	3B	59	;	Semicolon	
0111100	3C	60	<	Small than bracket	
0111101	3D	61	=	Equals sign	
0111110	3E	62	>	Bigger than symbol	
0111111	3F	63	?	Question mark	
1000000	40	64	@	At symbol	
1000001	41	65	A		Capital letters
1000010	42	66	B		
1000011	43	67	C		
1000100	44	68	D		
1000101	45	69	E		
1000110	46	70	F		
1000111	47	71	G		

1001000	48	72	H		
1001001	49	73	I		
1001010	4A	74	J		
1001011	4B	75	K		
1001100	4C	76	L		
1001101	4D	77	M		
1001110	4E	78	N		
1001111	4F	79	O		
1010000	50	80	P		
1010001	51	81	Q		
1010010	52	82	R		
1010011	53	83	S		
1010100	54	84	T		
1010101	55	85	U		
1010110	56	86	V		
1010111	57	87	W		
1011000	58	88	X		
1011001	59	89	Y		
1011010	5A	90	Z		
1011011	5B	91	[	Left square bracket	Special character
1011100	5C	92	\	Inverse/backward slash	
1011101	5D	93	]	Right square bracket	
1011110	5E	94	^	Circumflex	
1011111	5F	95	_	Underscore	
1100000	60	96	`	Gravis (backtick)	
1100001	61	97	a		Lowercase letters
1100010	62	98	b		
1100011	63	99	c		
1100100	64	100	d		
1100101	65	101	e		
1100110	66	102	f		
1100111	67	103	g		
1101000	68	104	h		
1101001	69	105	i		
1101010	6A	106	j		
1101011	6B	107	k		
1101100	6C	108	l		
1101101	6D	109	m		
1101110	6E	110	n		
1101111	6F	111	o		
1110000	70	112	p		
1110001	71	113	q		
1110010	72	114	r		
1110011	73	115	s		

1110100	74	116	t		
1110101	75	117	u		
1110110	76	118	v		
1110111	77	119	w		
1111000	78	120	x		
1111001	79	121	y		
1111010	7A	122	z		
1111011	7B	123	{	Left curly bracket	Special characters
1111100	7C	124	|	Vertical line	
1111101	7D	125	}	Right curly brackets	
1111110	7E	126	~	Tilde	
1111111	7F	127	DEL	Deletes a character. Since this control character consists of the same number on all positions, during the typewriter era it was possible to invalidate another character by punching out all the positions (Delete)	Control characters

Basic Data Types

- **Variables and constants are basic DATA types in C**

•**Data Types and Sizes:**

•Few basic data types in C:

•char a single byte, capable of holding one character in the local character set

•int an integer, typically reflecting the natural size of integers on the host machine

•float single-precision floating point

•double double-precision floating point

There are some qualifiers:

- ✓short int sh;
- ✓long int counter;