Storage Representation of Integers:

There are various methods for storage representation of integers.

(1) Signed and Magnitude method.
(2) 1’s complement method.
(3) 2’s complement method.
(4) Excess Notation.

(1) Signed and Magnitude Method:

Most of computer use one memory word to store an integer. The left most bit is used indicate the sign of number. The remaining bits are used to store magnitude of the number. A ‘0’ in the sign bit-position indicates a +ve sign and “1” in the sign bit-position indicates –ve sign. The storage representation of +10 and -10 in 8-bits using this method will be as below.

+10	0	0001010
-10	1	0001010

So the maximum no. stored in b-bits using method will be 127=2^7-1 and minimum number stored in 8 bit using this method will be -127= -(2^7-1). If computer use m-bits then range of number stored using this method will be – (2^{m-1}-1) to (2^{m-1} -1).

Here we will get two different representation of +40 and -40. This different representation of +0 and -0 is the limitation of this method.

(2) 1’s Complement Method:

The representation of +ve integers are same as representation of +ve integers in signed and magnitude method. The 1’s complement of binary number is obtained by subtracting each digit from 1. In other word 1’s complement of binary no. is obtained by replacing ‘0’ by ‘1’ & vice-versa. The storage representation of –ve no. is in the form of one’s complement. Hence storage representation of the +10 and -10 using 8-bits will be as below.

+10	0	00001010
-10	1	11110101

Maximum +ve no. which can be stored in 8-bits is +127 and minimum no. is -127. If a digit is stored in ‘m’ bits then maximum +ve no. stored would be 2^{m-1} -1 & the minimum no. would be – (2^{m-1}-1). Here also we have two different represe-ntation of +0 and -0.

(3) 2’s Complement Method:

The representation of +ve integers are same as representation of +ve integers in signed magnitude method. The –ve nos. will be stored in 2’s complement form. 2’s complement will be obtained by first getting 1’s complement and adding 1 to the right most bit. The storage representation of +10 & -10 in 8-bits using this method will be as below.

+10	0	00001010
-10	1	11110110

One’s complement of 00001010 is 11110101, so two’s complement will be 11110110. In this method there will not be two different representation of +0 and -0. In 8-bits range will be -128 to 127.

(4) Excess Notation:

For ‘m’ bit nos. this type of notation is known as excess 2m-1. In this method number will be represented as the sum of itself & 2^{m-1} i.e. 2^{m-1} +n.

e.g. for m=8, the method is known as excess 128 & no. is stored as itself plus 128. If one is interested to store -10. In 8-bits using excess notation, then -10 becomes -10+128 =118.

So, -10 will be represented by the 8-bit binary no. of 118.

2	118	
2	59	0
2	29	1
2	14	1
2	7	0
2	3	1
2	1	1
	0	1

1110110

So, the storage representation of -10 using 8 bits will be

01110110

If one want to store 10, in 8 bits using excess notation then +10 will be +10+128=138. 10 will be represented by the 8-bit binary number of 138.

For m=8 bits, i.e. in excess 128 notation minimum number stores will be -128 & maximum no. stored will be 127 i.e. range will be -128 to 127. In m-bits range will be -2^{m-1} to 2^{m-1}-1.

The Error Detection and Correction of One-Bit Parity Method

Parity Bit Method (Error detection):

Just human being make mistake, computer may make mistake. The CPU and memory do not have any moving part. So they are highly reliable but the I/O devices involve physical motion and so they are less reliable. Error can be caused by dust on read/write head or due to fluctuation in current. A very simple and widely used method for detecting a single bit error that is a bit change from 0 to 1 and 1 to 0 is to add a parity bit to each character. In an odd parity, the parity bit is so chosen that the number of 1's in the character including parity bit is an odd number. The following example shows the creation of an eight bit odd parity code from 7-bit ASCII character.

ODD		EVEN
1 1000001		0 1000001
1 1010101		0 1010101
1 0101011	parity bit $\longrightarrow$	0 0101011
0 1111010		1 1111010

If the bit is change during transformation the no. of 1 in the received character would be even. And hence the receiver will know that error has occurred. In an even parity code, the parity bit is so chosen that no. of 1 in the character including parity bit is a even number.

Hamming Code (Error detection and correction – One bit only):

Hamming code will not detect one bit error but also it will correct it. It was developed by Richard Hamming in 1950. In Hamming code K parity bits are added to an n-bit character which will form a n+k bits character code. All bits whose bit no. is power of 2 is a parity bit and rest are used as data bits. i.e. bit 1, 2, 4, 8 are parity bits and bits 3, 5, 6, 7, 9, 10, 11 are data bit. Each parity bit checks specific bit position. For even parity computer the parity bit is set so that total number of 1 in checked position is even. The parity bit position

1 checks bits 1, 3, 5, 7, 9, 11
2 checks bits 2, 3, 6, 7, 10, 11
4 checks bits 4, 5, 6, 7
8 checks bits 8, 9, 10, 11

In general bit 'n' is checked by those J parity bits such that $b_1+b_2+\ldots\ldots+b_j=n$

E.g bit '5' is checked by parity bit 1 and 4 because 1+4=5

Construct Hamming code for ‘b’ where ASCII code for b is ‘98’.
$(98)_d=(1100010)_b$

0	0	1	1	1	0	0	1	0	1	0
1	**2**	3	**4**	5	6	7	**8**	9	10	11

So Hamming code for ‘b’ is 00111001010, if the left most bit is changed during transmission, the revised character will be 10111001010. The parity bit will be checked by the receiver with the following results.

Parity bit 1 is incorrect
Parity bit 2 is correct
Parity bit 4 is correct
Parity bit 8 is correct

Here parity bit ‘1’ is incorrect any of the 1, 3, 5, 7, 9, 11 will be incorrect but parity 2 is correct so bits 2, 3, 6, 7, 10, 11 are correct bits. So incorrect bit will be either 1 or 5 or 9. But parity bit 4 is correct will result bit 4, 5, 6, 7 are correct bits. Therefore incorrect bit are either 1 or 9. But parity bit 8 is correct. Therefore bits 8, 9, 10, 11 are correct bits. Hence incorrect bit will be 1.

Character Codes

What is ASCII?

ASCII (American Standard Code for Information Interchange) is the most common format for text files in computers and on the Internet. In an ASCII file, each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined.

UNIX and DOS-based operating systems use ASCII for text files. Windows NT and 2000 uses a newer code, Unicode. IBM's S/390 systems use a proprietary 8-bit code called EBCDIC. Conversion programs allow different operating systems to change a file from one code to another.

ASCII was developed by the American National Standards Institute (ANSI).

ASCII has been very popular in the computer world. It contains seven bits to define each letter or character excluding eighth bit for error-checking function. There are 128 specific characters including capital letters, small letters, 0 to 9 digit, special symbols and some specific character having specific different functions. Thirty-three codes are used to represent things other than specific characters. The first 32 (0-31) codes represent a chime sound, used to feed line as well as to start of a header. The final code, 127 represents a backspace while the first 31 bits are the printable characters. Bits ranging from 48 to 57 represent the numeric digits and 65 to 90 represents the capital letters, while bits 97 to 122 are the lower-case letters. The rest bits represent symbols of punctuation, mathematical symbols, and other symbols such as the pipe and tilde.

What is Unicode?

Unicode provides a unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.

Unicode is the universal character encoding, maintained by the Unicode Consortium. This encoding standard provides the basis for processing, storage and interchange of text data in any language in all modern software and information technology protocols. Unicode covers all the characters for all the writing systems of the world, modern and ancient. It also includes technical symbols, punctuations, and many other characters used in writing text. The Unicode Standard is intended to support the needs of all types of users, whether in business or academia, using mainstream or minority scripts.

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystems, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others. Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.

Incorporating Unicode into client-server or multi-tiered applications and websites offers significant cost savings over the use of legacy character sets. Unicode enables a single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

About the Unicode Consortium

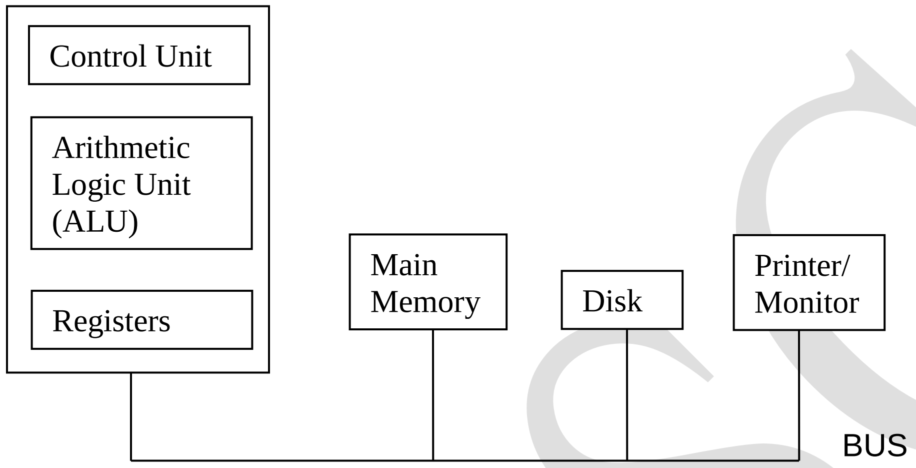
The Unicode Consortium is a non-profit organization founded to develop, extend and promote use of the Unicode Standard, which specifies the representation of text in modern software products and standards. The membership of the consortium represents a broad spectrum of corporations and organizations in the computer and information processing industry. The consortium is supported financially solely through membership dues. Membership in the Unicode Consortium is open to organizations and individuals anywhere in the world who support the Unicode Standard and wish to assist in its extension and implementation.

Processor and Its Functions:

Functions and Components of processor:

A digital computer consists of an interconnected system of processors, memories and input/output devices as shown in below figure.

Central Processing Unit (CPU):



Organization of Simple Computer with one CPU and two I/O devices.

The central processing unit (CPU) is brain of computer. Its major function is to execute programs stored in the main memory by fetching their instructions, examining them and then executing them one after other.

CPU is composed of several distinct parts.

1. Control Unit: It is responsible for fetching instructions from main memory and determining their type.
2. Arithmetic Logic Unit (ALU): It performs operations such as mathematical and Boolean, to carry out the instructions.
3. Registers: CPU also contains small, high-speed memory units used to store temporary results and certain control information. Each register has a certain size and faction.

Instruction Execution Cycle:

CPU is responsible for fetching and executing the instructions of the program one after another. To carry out its operation it performs a cyclic run known as instruction execution cycle. CPU executes each instruction in a series of small steps. These steps are as follows:

1. Fetch the next instruction from memory into instruction register.
2. Change the program counter to point to the following instruction.
3. Determine the type of instruction just fetched.

4. If the instruction uses a word in memory then determine where it is.
5. Fetch the word, if needed into CPU register.
6. Execute the instruction.
7. Go to step 1 to begin executing the next instruction.

This sequence of steps is frequently referred to as the fetch-decode-execute cycle. It is central to the operation of all computers.

CPU has two important registers called Program Counter and Instruction Register. Program counter points to the next instruction to be executed after finishing the current instruction and Instruction Register stores the instruction currently in execution.

When a program is loaded into main memory for execution, the program counter points to the first instruction. CPU loads the instruction pointed by the program counter in to instruction register and program counter is incremented by 1 to point to the next instruction. CPU then decides the type of the instruction loaded in the instruction register to determine how to execute it. It checks whether current instruction required any data operation or not. If it performs any operation on data then it loads the data required, form main memory to the registers. Then it executes the instruction. If any result is produced then the result is stored back to either register or to the main memory, as per the instruction. After finishing the execution if loads the instruction pointer by program counter into instruction register and whole process is repeated till all the instructions of the program are executed.

CPU Organization: Data Path of a Typical Von Neumann Machine

The internal organization of part of a typical Von Neumann CPU is as shown in the figure. This part is called the data path and consists of the registers (1 to 32), the ALU and several buses connecting these devices.

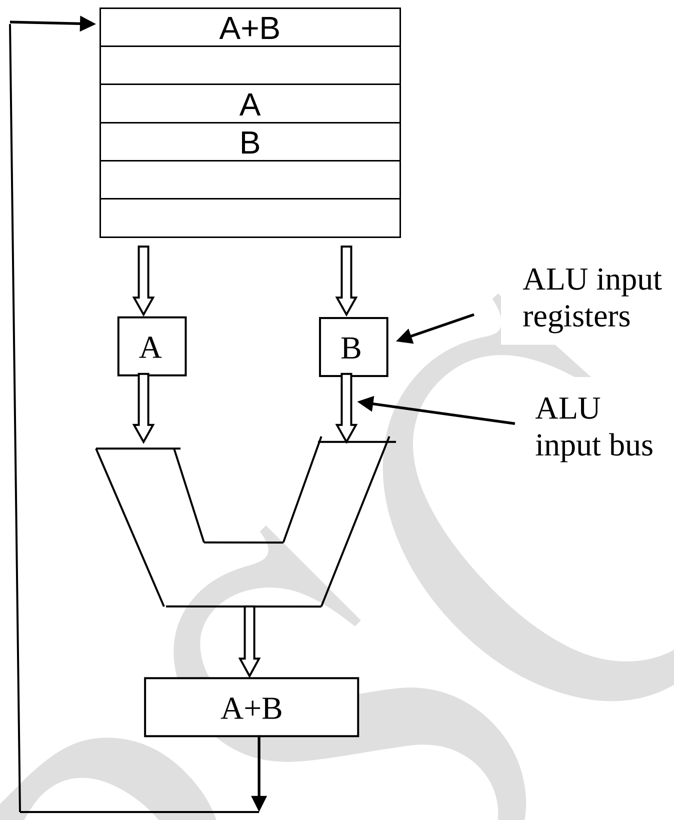
The registers content is fed into two ALU registers labeled A & B in the figure. These registers hold the ALU input while the ALU is computing. ALU performs various arithmetic and logic operations and produces result, which is stored in the output register. The value in the output register can be stored back to the CPU register and then to the main memory, if required as per the instruction.

The process of running two operands through the ALU and storing the result is called the **data path cycle**. This is the heart of the CPU. The faster the data path cycle is, the faster the machine runs.

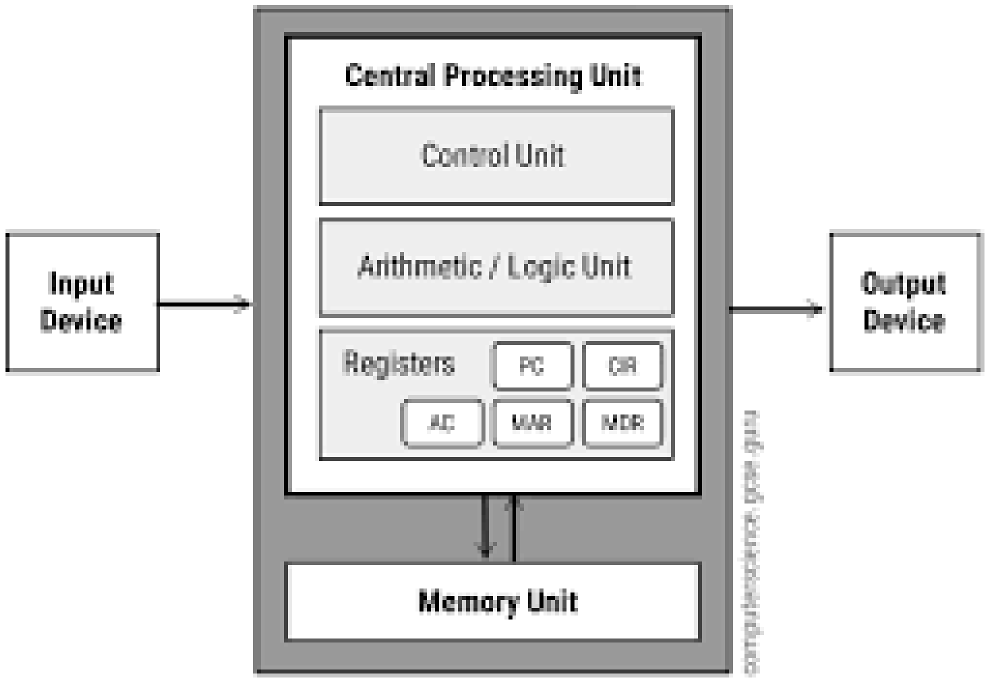
Most instructions can be divided into two categories.
1. Register-Memory instruction and
2. Register-Register instruction.

1. **Register-Memory instruction:** Such instructions allow memory words to be fetched into registers, where they can be used as ALU inputs. Some register-memory instructions allow registers to be stored back into memory.

2. **Register-Register instruction:** Such instructions fetch two operands from the registers into ALU registers, and the result is stored back to one of the registers.



Von-Neumann Machine Diagram:


Central Processing Unit
Control Unit
Arithmetic / Logic Unit
Registers
PC
CIR
AC
MAR
MDR
Memory Unit
Input Device
Output Device