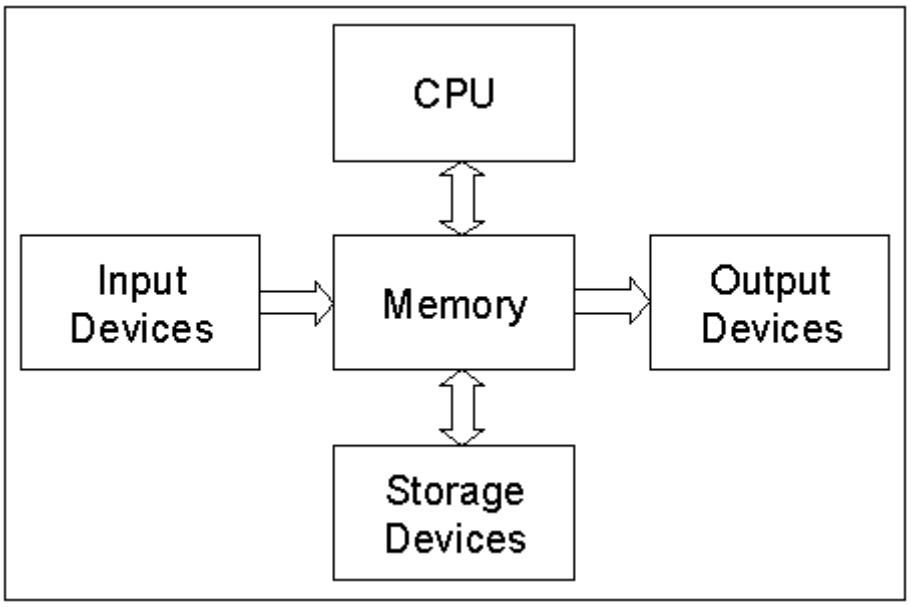
3 Computer Architecture and Assembly Language

3.1 Overview of the Organization of a Computer System

All general-purpose computers require the following hardware components:

- Main memory: enables a computer to store, at least temporarily, data and programs.
- Secondary storage: allows a computer to permanently retain large amounts of data. Common secondary storage devices include disk drives and tape drives.
- Input/output subsystem: includes any operation, program, or device that transfers data to or from a computer.
- Central Processing Unit (CPU): The brains of the computer, this is the component that actually executes instructions.
- Buses: a bus is a digital pathway on which data passes from one place to another.
- System clock: a clock that regulates when operations start and when they finish.



The main modules of a computer system

3.2 The System Bus

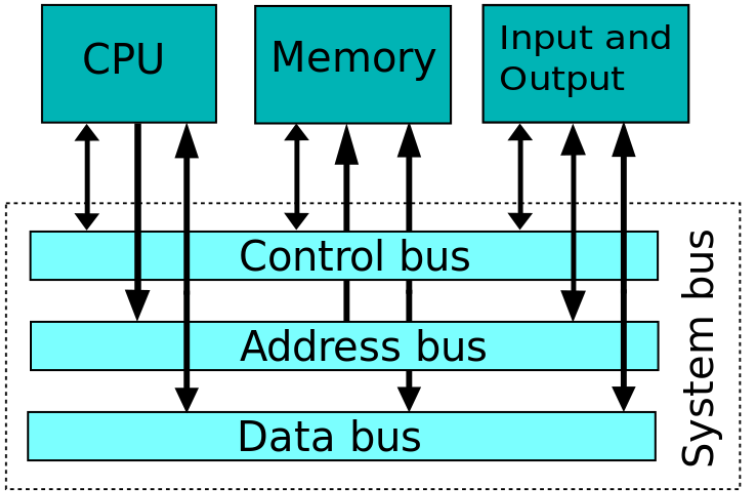
The term 'bus', in computing, is a set of physical connections (cables, printed circuits, etc.) which can be shared by multiple hardware components in order to communicate with one another. The processor and main memory transmit information to each other by means of a bus.

A bus is characterised by the amount of information that can be transmitted at once. This amount, expressed in bits, corresponds to the number of physical

lines over which data is sent simultaneously. A 32-wire ribbon cable can transmit 32 bits in parallel. The term "width" is used to refer to the number of bits that a bus can transmit at once.

Additionally, the bus speed is also defined by its frequency (expressed in Hertz), the number of data packets sent or received per second. Each time that data is sent or received is called a 'cycle'. This way, it is possible to find the maximum transfer speed of the bus, the amount of data which it can transport per unit of time, by multiplying its width by its frequency. A bus with a width of 16 bits and a frequency of 133 MHz, therefore, has a transfer speed equal to:

$16 * 133.10^6 = 2128*10^6$ bit/s,
or $2128*10^6/8 = 266*10^6$ bytes/s
or $266*10^6 /1000 = 266*10^3$ KB/s = 266 MB/s



Buses

Each bus is generally constituted of 50 to 100 distinct physical lines, divided into three subassemblies:

- The address bus (sometimes called the memory bus) transports memory addresses which the processor wants to access in order to read or write data. It is a unidirectional bus.
- The data bus transfers instructions and data coming from or going to the processor. It is a bidirectional bus.
- The control bus (or command bus) transports orders and synchronisation signals coming from the control unit and travelling to all other hardware components. For example it tells RAM whether to read from or write into a memory location. It is a bidirectional bus, as it also transmits response signals from the hardware.

The system bus is the bus that connects the CPU to main memory on the motherboard. I/O buses, which connect the CPU with the systems other components, branch off of the system bus. The system bus is also called the frontside bus, memory bus, local bus, or host bus.

There are generally two buses within a computer:

- The internal bus (sometimes called the front-side bus, or FSB for short). The internal bus allows the processor to communicate with the system's central memory (the RAM).
- The expansion bus (sometimes called the input/output bus) allows various motherboard components (USB, serial, and parallel ports, cards inserted in PCI connectors, hard drives, CD-ROM and CD-RW drives, etc.) to communicate with one another. However, it is mainly used to add new devices using what are called expansion slots connected to the input/output bus.

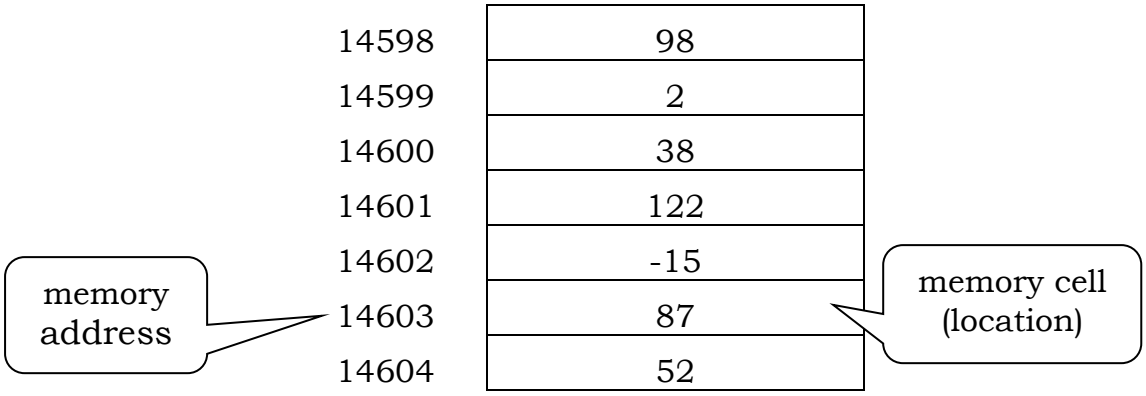
Some bus technologies are ISA, EISA, USB, FireWire and SCSI.

The width of the address bus determines the amount of memory a system can address. For example, a system with a 32-bit address bus can address 2^{32} (4,294,967,296) memory locations. If each memory address holds one byte, the addressable memory space is 4 GB.

3.3 Memory

Primary storage, also known as 'main storage' or 'memory', is the main area in a computer in which data is stored for quick access by the computer's processor. On today's smaller computers, especially personal computers and workstations, the term 'random access memory' (RAM) - or just 'memory' - is used instead of primary or main storage, and the hard disk, diskette, CD, and DVD collectively describe 'secondary storage' or auxiliary storage.

Main memory, to be more exact, consists of all the (electronic) memory that is directly accessible by the processor i.e. RAM, ROM and the various kinds of Cache. Main memory can be considered as a sequence of locations where each location has an address that identifies it. This is shown in the following diagram.



Each memory location is identified by means of an address.

A small part of memory

3.3.1 Mega, Giga, Tera

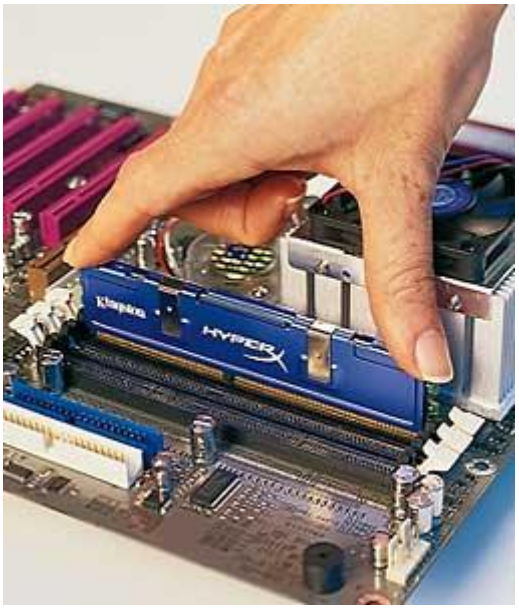
The following table shows the terms used to describe the amounts of memory.

kilo-	k or K	10^3	2^{10}
mega-	M	10^6	2^{20}
giga-	G	10^9	2^{30}
k = 10^3 and K = 2^{10}			

3.3.2 RAM

RAM is an acronym for 'random access memory', a type of computer memory that can be accessed randomly; that is any byte of memory can be accessed without touching the preceding bytes. RAM is the most common type of memory found in computers and other devices, such as printers.

There are two different types of RAM: DRAM (Dynamic Random Access Memory) and SRAM (Static Random Access Memory). The two types differ in the technology they use to hold data, with DRAM being the more common type. In terms of speed, SRAM is faster. DRAM needs to be refreshed thousands of times per second while SRAM does not need to be refreshed, which is what makes it faster than DRAM. DRAM supports access times of about 60 nanoseconds, SRAM can give access times as low as 10 nanoseconds. Despite SRAM being faster, it's not as commonly used as DRAM because it's so much more expensive. Both types of RAM are volatile, meaning that they lose their contents when the power is turned off.



A RAM module being placed in a socket

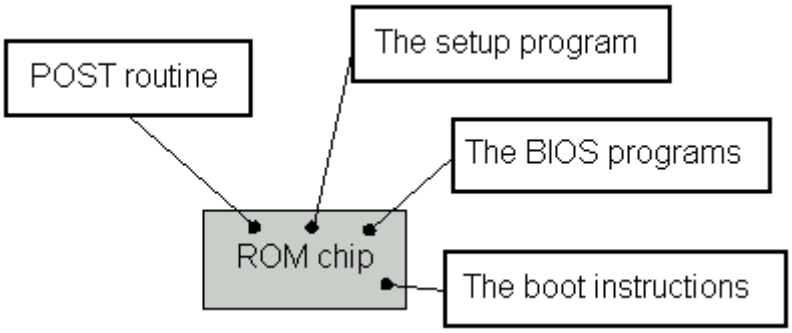
In common usage, the term RAM is synonymous with main memory, the memory available to programs. For example, a computer with 8MB RAM has approximately 8 million bytes of memory that programs can use. In contrast, ROM (read-only memory) refers to special memory used to store programs that boot the computer and perform diagnostics. Most personal computers have a small amount of ROM (a few thousand bytes). In fact, both types of memory (ROM and RAM) allow random access. To be precise, therefore, RAM should be referred to as read/write RAM and ROM as read-only RAM.

3.3.3 ROM

The 'Read-Only Memory' is non-volatile. ROM contains the data needed to start up the computer. The computer is designed in such a way that when it is switched on it executes programs contained in the ROM.

Different ROM-type memories contain these essential start-up data, i.e.:

- The BIOS is a program for controlling the system's main input-output interfaces, hence the name BIOS ROM which is sometimes given to the read-only memory chip of the mother board which hosts it.
- The bootstrap loader: a program for loading the operating system from secondary storage to RAM and launching it. This generally seeks the operating system on the floppy drive then on the hard disk, which allows the operating system to be launched from a system floppy disk in the event of malfunction of the system installed on the hard disk.
- The CMOS (Complementary metal–oxide–semiconductor) Setup holds data like the time, date and configuration information. CMOS requires constant (low) power to keep the information.
- The Power-On Self-Test (POST), a program that runs automatically when the system is booted, thus allowing the system to be tested.



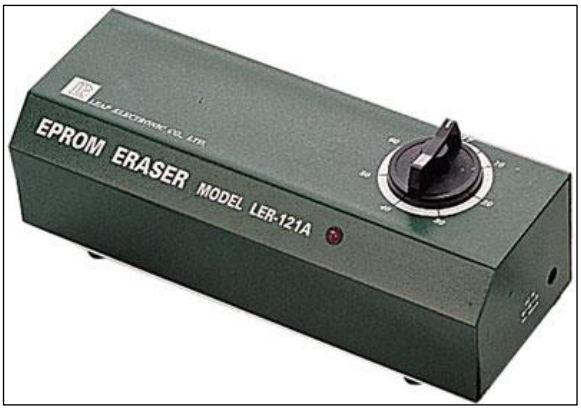
The contents of the ROM chip of a personal computer

Given that ROMs are much slower than RAM memories (access time for a ROM is around 150 ns whereas for SDRAM it is around 10 ns), the instructions given in the ROM are sometimes copied to the RAM at start-up; this is known as shadowing, though is usually referred to as shadow memory).

A CMOS ROM chip containing the BIOS

ROM memories have gradually evolved from fixed read-only memories to memories than can be programmed and then re-programmed:

- ROM: The first ROMs had their programs and data written at the manufacturing stage and the information in them could not be altered.
- PROM: PROM (Programmable Read Only Memory) memories were developed at the end of the 70s. Information in them could "burnt" using a device called a "ROM programmer". Once the information was written in them it could not be altered.
- EPROM: EPROM (Erasable Programmable Read Only Memory) memories are PROMs whose data can be deleted and they could be reprogrammed. However to do this process the chips have to be pulled out of the computer.
- EEPROM: EEPROM (Electrically Erasable Read Only Memory memories are also erasable PROMs, but unlike EPROMs, they can be erased by a simple electric current, meaning that they can be erased even when they are in position in the computer.
    - There is a variant of these memories known as flash memories (also Flash ROM or Flash EPROM). The difference in technology makes the flash memory denser. EEPROMs are thus used preferably to memorise configuration data and the Flash memory is used for programmable code (IT programmes).
    - The action involving reprogramming of an EEPROM is known as flashing.

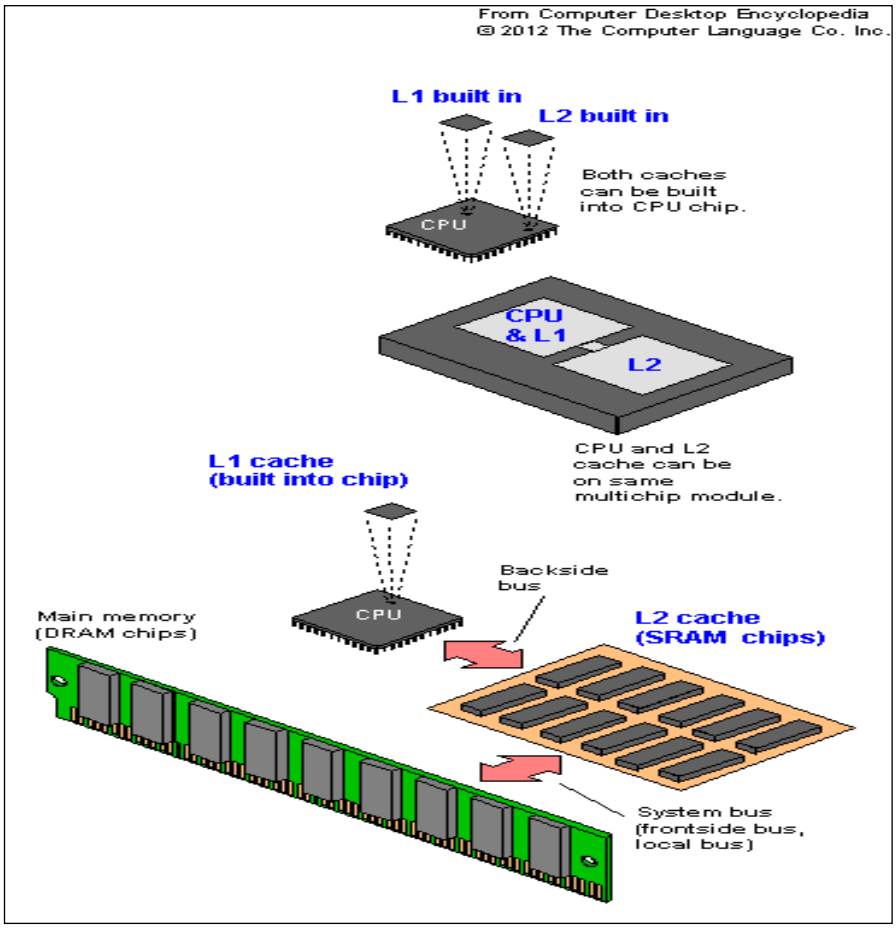


EPROM programmer and eraser

3.3.4 Cache

Cache is a special high-speed storage mechanism. Two types of caching are commonly used in personal computers:

- memory caching
- disk caching

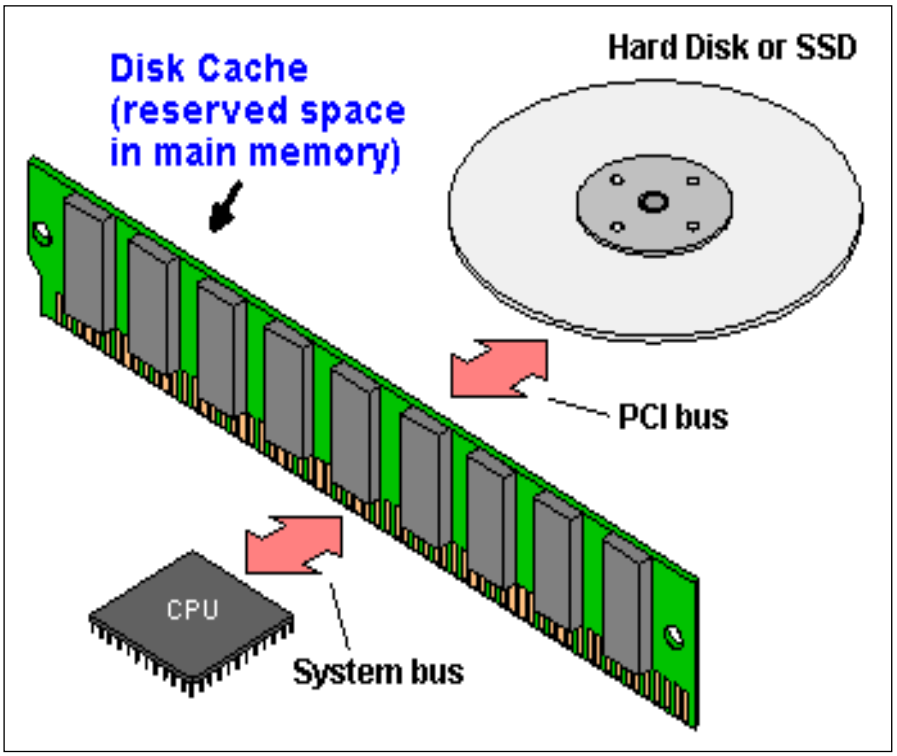


Caches

A memory cache, sometimes called a 'cache store' or 'RAM cache', is a portion of memory made of high-speed static RAM (SRAM) instead of the slower and cheaper dynamic RAM (DRAM) used for main memory. Memory caching is effective because most programs access the same data or instructions over and over. By keeping as much of this information as possible in SRAM, the computer avoids accessing the slower DRAM.

Some memory caches are built into the architecture of microprocessors. The Intel 80486 microprocessor, for example, contains an 8K memory cache, and the Pentium has a 16K cache. Such internal caches are often called Level 1 (L1) caches. Most modern PCs also come with external cache memory, called Level 2 (L2) caches. These caches sit between

the CPU and the DRAM. Like L1 caches, L2 caches are composed of SRAM but they are much larger.



Disk caching works under the same principle as memory caching, but instead of using high-speed SRAM, a disk cache uses conventional main memory. The most recently accessed data from the disk (as well as adjacent sectors) is stored in a memory buffer. When a program needs to access data from the disk, it first checks the disk cache to see if the data is there. Disk caching can dramatically improve the performance of applications, because accessing a byte of data in RAM can be thousands of times faster than accessing a byte on a hard disk.

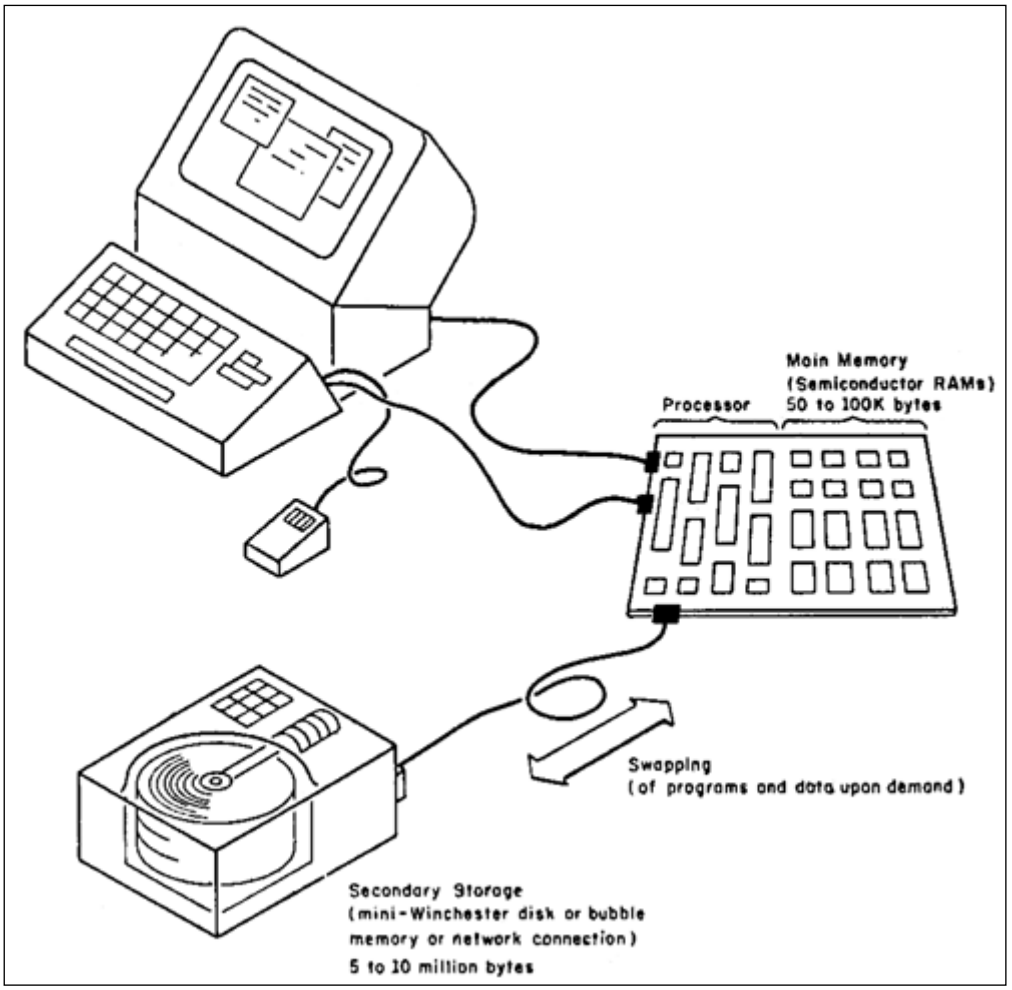
When data is found in the cache, it is called a cache hit, and the effectiveness of a cache is judged by its hit rate. Many cache systems use a technique known as 'smart caching', in which the system can recognize certain types of frequently used data. The strategies for determining which information should be kept in the cache constitute interesting problems in computer science.

3.3.5 Virtual Memory

Virtual Memory is a feature of an operating system that enables the use of a space that is larger than the actual amount of RAM present, temporarily relegating some contents from RAM to a disk.

In a system using virtual memory, the physical memory is divided into equally-sized pages. When a program is opened (let us say it is 30 pages long) and the number of available pages in memory is less than this size (let us say that 8 pages are available) only the most important parts of the program are loaded in RAM. The other parts (22 pages) are only left on the hard-disk. If some page is required from the hard-disk then a page is chosen by the computer so that it is 'emptied' (its contents copied on the hard-disk) and the necessary page copied from hard-disk into the 'emptied' page. This is called 'swapping'. In this way more programs are handled than actually fit in the memory.

The 'emptied' page is normally chosen according to how long it has been occupying a page without being used. This method is called LRU (least recently used). Other methods exist.

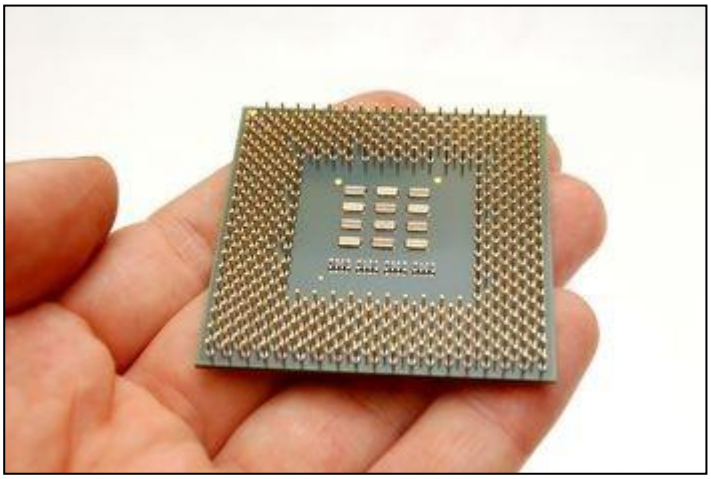


BYTE Magazine, August 1981, describing Virtual Memory

3.4 CPU

The processor in a computer is the module that executes instructions and programs (a program is a sequence of instructions). Today the terms processor and CPU have the same meaning. In the old days the term CPU used to refer to the combination containing the processor and main memory.

The microprocessor is a silicon chip that contains a whole processor. At the heart of all personal computers and most workstations sits a microprocessor.



CPU

The three basic characteristics that differentiate microprocessors are:

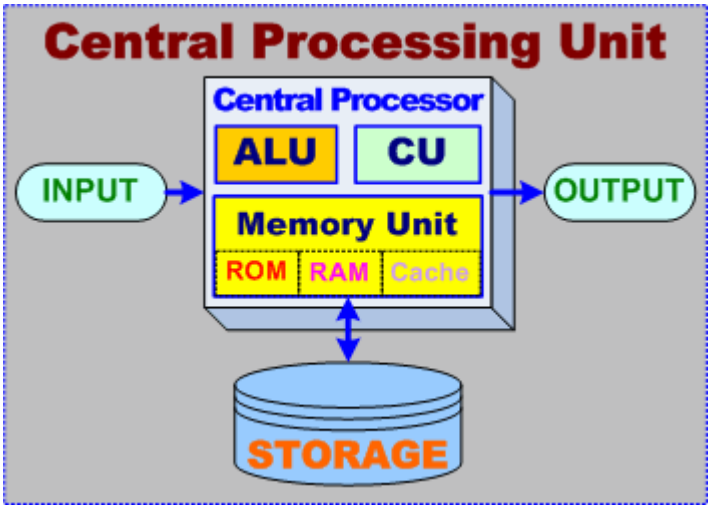
- Instruction set: The set of instructions that the microprocessor can execute.
- Bandwidth: The number of

bits processed in a single instruction.
- Clock speed: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.

In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).

The most important parts of a processor are:

- Control Unit (CU)
- Arithmetic-Logic Unit (ALU)
- Registers
- Cache (L1)

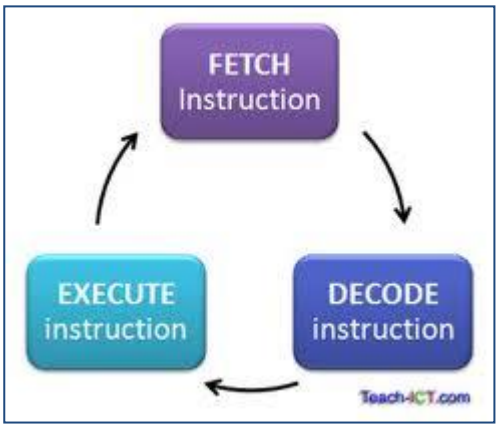


Central Processing Unit

3.4.1 The Fetch-Decode-Execute Cycle

The role of the processor in a computer is to execute instructions. These instructions are given in the form of a program. The processor follows the Fetch-Execute cycle (this is also called the Fetch-Decode-Execute cycle). In very simple terms the Fetch-Execute cycle performs the following sequence of commands.

1. Bring the next command from main memory.
2. Interpret this command.
3. Execute this command.
4. Return to step 1.



The Fetch-Execute Cycle

3.4.2 Control Unit

The control unit extracts instructions from memory and decodes and executes them, and sends the necessary signals to the ALU to perform the operation needed. Control Units are either hardwired or micro-programmed. The control unit communicates with the arithmetic logic unit and the system memory.

3.4.3 Arithmetic-Logic Unit

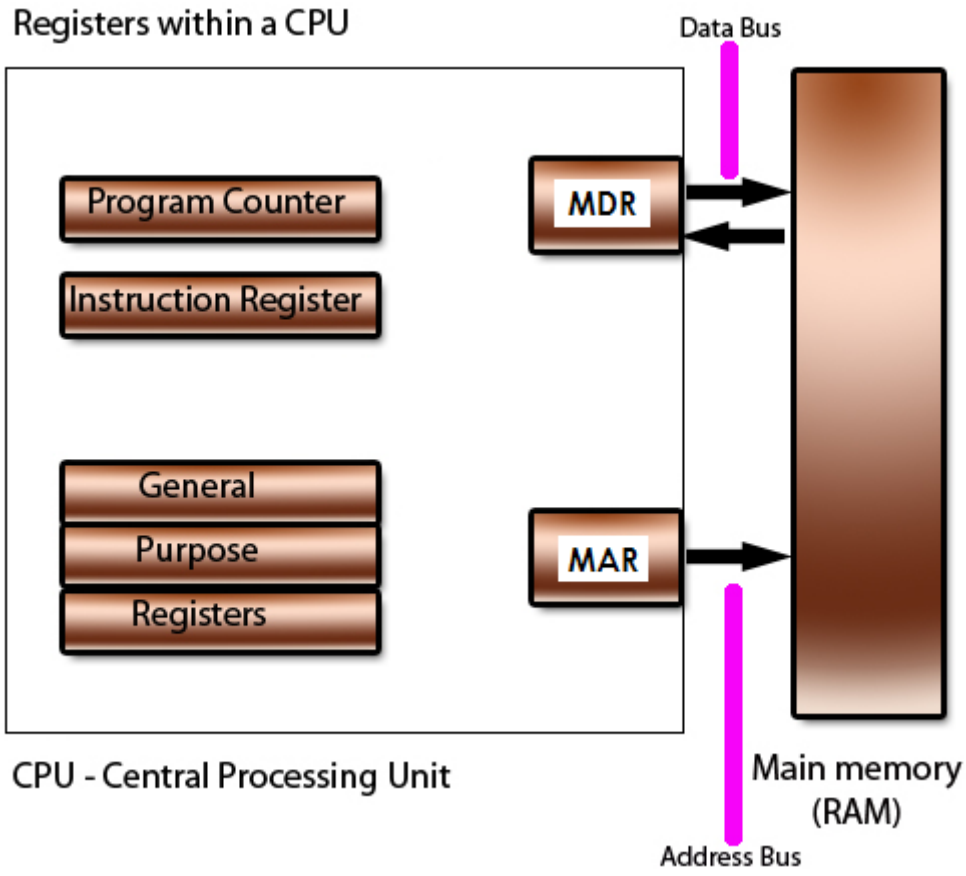
The ALU is where all the arithmetic and logical operations are carried out. Apart from the basic arithmetic operations (addition, subtraction, multiplication, division) the ALU performs operations involving logic (AND, OR, NOT, comparison between two values to see if they are equal or which one is greater than the other).

3.4.4 Registers

The registers are what the CPU uses for temporary storage of data. They are not part of any of the system memory, but are instead additional storage locations that are on the CPU itself. This makes registers very fast for the CPU to use. The registers are controlled by the control unit and are used to hold and transfer instructions, and perform the logical and arithmetic operations.

Registers are assigned specific functions. Some are listed below:

- Accumulators: they hold values are can perform operations with these values
- Address Registers: they can store memory addresses.
- Storage Registers: they can temporarily store data.
- Miscellaneous: general purpose used for several functions.



Registers

Particularly important registers are the following:

- Current Instruction Register (CIR): This register (also called IR) is the part of a CPU's control unit that stores the instruction currently being executed or decoded.
- Program Counter (PC): The PC (also called the 'instruction pointer' or 'instruction address register') holds the address of the next instruction to

be executed. In most processors, the instruction pointer is incremented automatically after fetching a program instruction.

- Memory Address Register (MAR): This register either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored.
- Memory Data Register (MDR): This register contains the data to be stored in the computer storage (e.g. RAM), or the data after a fetch from the computer storage. It acts like a buffer.

A more detailed 'fetch-decode-execute cycle' is the following:

1. The Program Counter (PC) contains the address of the next instruction to be fetched. The address contained in the PC is copied to the Memory Address Register (MAR).
2. The instruction is copied from the memory location contained in the MAR and placed in the Memory Data Register (MDR).
3. The entire instruction is copied from the MDR and placed in the Current Instruction Register (CIR)
4. The PC is incremented so that it points to the next instruction to be fetched
5. The address part of the instruction is placed in the MAR.
6. The instruction is decoded and executed.
7. The processor checks for interrupts (signals from devices or other sources seeking the attention of the processor) and then it either branches to the relevant interrupt service routine or starts the cycle again.

3.4.5 Cache

The Cache is a small amount of fast memory which holds recently accessed data or instructions so that if they are used by the programs again, the cache can supply them faster than main memory. The cache is used in more than one place in a computer system. The one found inside the processor is called L1 cache.

3.4.6 Multi-Core Processors

2-core Processor

A multi-core processor is an integrated circuit to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks. A dual core set-up is somewhat comparable to having two, separate processors installed in the same computer, but because the two processors are actually plugged into the same socket, the connection between them is faster. Ideally, a dual core processor is nearly twice as powerful as a single core processor. In practice, performance gains are said to be about fifty percent: a dual core

processor is likely to be about one-and-a-half times as powerful as a single core processor.

In 2005, the first personal computer dual-core processors were announced and as of 2009 dual-core and quad-core processors are widely used in servers, workstations and PCs while six and eight-core processors will be available for high-end applications in both the home and professional environments.

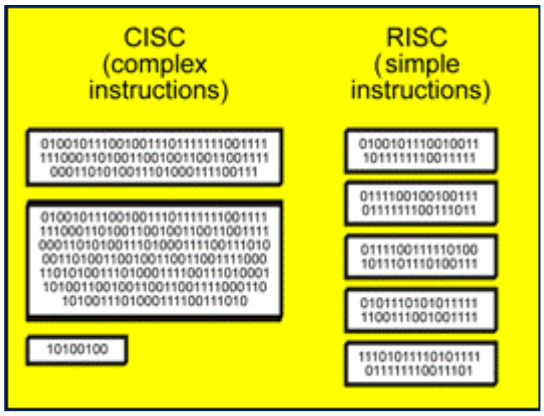
3.4.7 CISC and RISC

CISC stands for Complex Instruction Set Computer. Most personal computers use a CISC architecture, in which the CPU supports as many as two hundred instructions. An alternative architecture, used by many workstations and also some personal computers, is RISC (reduced instruction set computer), which supports fewer instructions.

CISC computers were designed with a full set of computer instructions that were intended to provide needed capabilities in the most efficient way. Later, it was discovered that, by reducing the full set to only the most frequently-used instructions, the computer would get more work done in a shorter amount of time for most applications.

Macintosh computers use a RISC microprocessor. Intel's Pentium microprocessors are CISC.

One advantage of RISC computers is that they can execute their instructions very fast because the instructions are so simple. Another, perhaps more important advantage, is that RISC chips require fewer transistors, which makes them cheaper to design and produce.



CISC and RISC

3.5 Registers

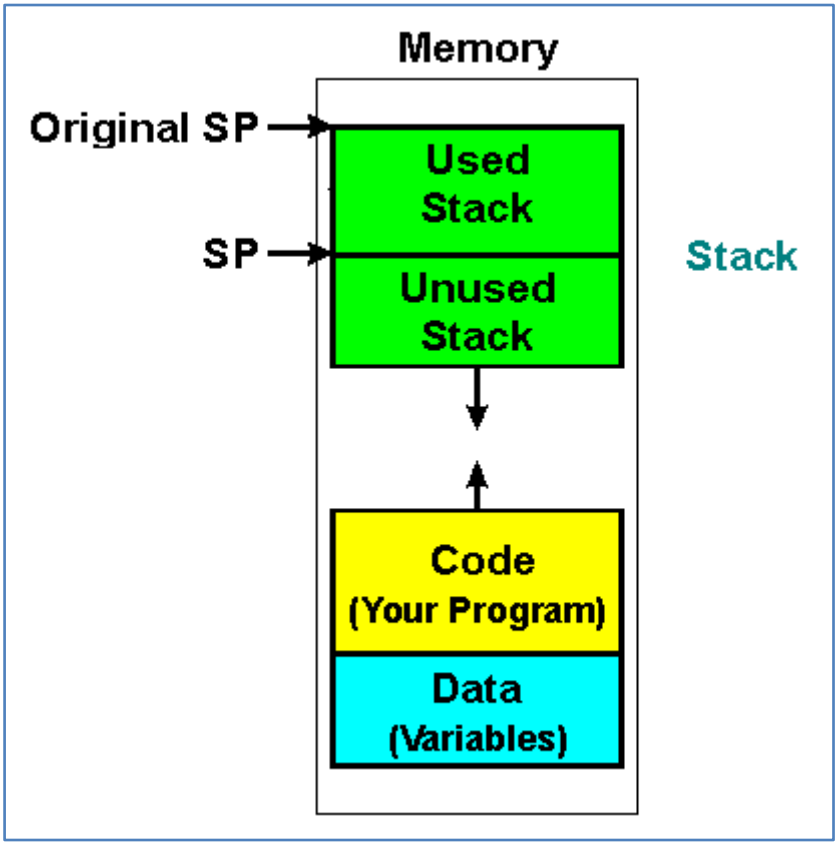
We already mentioned the registers CIR, PC, MAR and MDR. In this section we will mention others.

- AX is the "accumulator"; some of the operations, such as MUL and DIV, require that one of the operands be in the accumulator. Some other operations, such as ADD and SUB, may be applied to any of the registers (that is, any of the eight general- and special-purpose registers) but are more efficient when working with the accumulator.
- BX is the "base" register; it is the only general-purpose register which may be used for indirect addressing. For example, the instruction MOV [BX], AX causes the contents of AX to be stored in the memory location whose address is given in BX.

- CX is the "count'' register. The looping instructions, the shift and rotate instructions all use the count register to determine how many times they will repeat.
- DX is the "data'' register; it is used together with AX for the MUL and DIV operations, and it can also hold the port number for the IN and OUT instructions, but it is mostly available as a convenient place to store data.

All the four registers just mentioned are considered as general-purpose registers.

SP is a register that holds the stack pointer. It indicates the current position of the top of the stack (the stack resides in main memory).



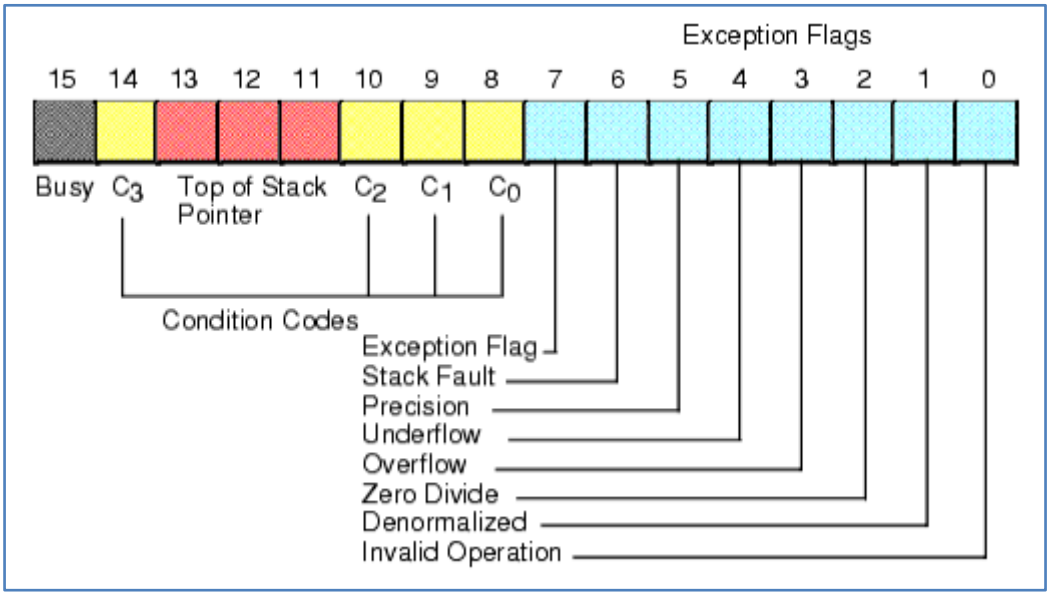
A Stack with Pointer SP

The stack (LIFO) is used to hold information about the state of a process that has to be terporarily suspended. The set of saved information is called a stack frame. This includes for example: the return address and local variables.

Examples of operations which typically affect the stack are: subroutine calls and interrupt calls.

The status register (flag register) is a collection of 1-bit values which reflect the current state of the processor and the results of recent operations. Here are some examples:

- Carry bit: set if the last arithmetic operation ended with a leftover carry bit coming off the left end of the result. This signals an overflow on unsigned numbers.
- Parity bit: set if the low-order byte of the last data operation contained an even number of 1 bits (that is, it signals an even parity condition).
- Zero bit: set if the last computation had a zero result. After a comparison this indicates that the values compared were equal (since their difference was zero).
- Sign bit: set if the last computation had a negative result (a 1 in the leftmost bit).
- Interrupt bit: when set, interrupts are enabled.
- Overflow bit: set if the last arithmetic operation caused an overflow



Status (Flag) Register

An example of an assembly language instruction is: LDA #24.

- The meaning of this instruction is: load (write) 24 in the accumulator.
- LDA is called the operator
- #24 is called the operand
- If 10110 is the binary code for LDA, then 10110 is called the opcode (sometimes LDA itself is called the opcode)
- LDA is also called a mnemonic.
- Example of an assembly language program:

```
rep:    LDA #14     ; load the accumulator with 14
        ADD #31     ; add 31 to the contents of the accumulator
        JZE rep     ; jump to 'rep' if accumulator contains zero
        HLT         ; stop program execution
```

- In the above program 'rep' is called a label.
- The instruction set is the set of all instructions performed by a particular CPU.

Memory address modes tell you how the CPU accesses data.

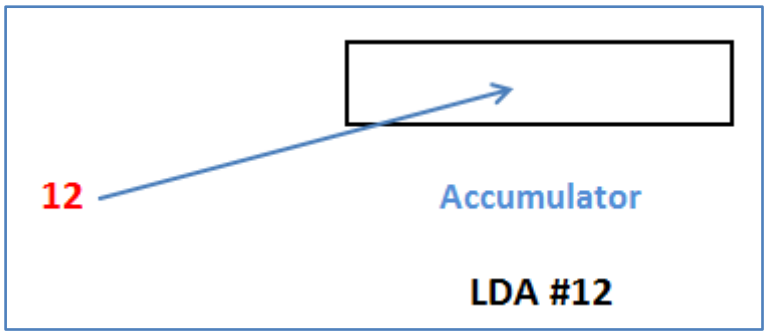
3.5.1 Address Modes

Four commonly-used memory address modes are the following:

- Immediate
- Direct
- Register
- Symbolic

Immediate addressing means that the data is found inside the instruction itself.

Example: 'LDA #12' means: load 12 into A (the accumulator).

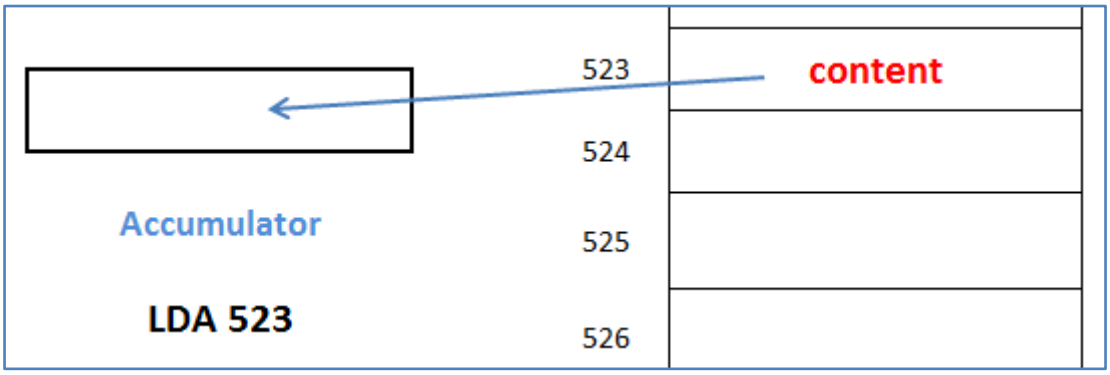


Immediate Addressing

Advantage: very fast.
Disadvantage: the value in the instruction never changes.

Direct addressing means the code refers directly to a location in memory.

Example: 'LDA 523' means: load the contents of address 523 into A.



Direct Addressing

Advantage: fast (but not as fast as immediate addressing).
Disadvantage: the program cannot be relocated.

Re-locatable code refers to code that can be put anywhere in RAM.

Register addressing is a term used to indicate an instruction where the operands are registers for example: Add R4, R3 (which means get the value in register R3 and add it to the value in register R4 placing the value in R4.

Symbolic addressing means that the instruction instead of referring to an address refers to a name (symbol).

Example: 'LDA num' means: load the contents of num into A.
Advantages:

- The program is re-locatable in memory.
- Using symbols makes the software much more understandable.

3.5.2 Examples of instructions

OPCODE	EXAMPLES

MOV	MOV is the mnemonic for Move
	MOV TOTAL, 48 ; Transfer the value 48 to the memory variable
	; TOTAL
	MOV AL, 10 ; Transfer the value 10 to the AL register
	MOV CL, TABLE[2] ; Copies the 3rd element of array TABLE in CL
	MOV [EBX], 72 ; Copies 72 at the address indicated in ECB

ADD	ADD AH, BH ; Add the number in the BH register to the number
	; in the AH register and store in the AH register
	ADD MARKS, 10 ; Add 10 to the variable MARKS
	ADD EBX, [ECX] ; Add the number found in the address indicated in
	; ECX with the number in EBX and store in EBX

SUB	SUB is the mnemonic for Subtract
	SUB EBX, EAX ; Subtract the number found in register EAX from
	; the number found in register EBX and store the
	; result in EBX
	SUB [ad], 4Bh ; Subtract the hex number 4B from the number
	; found in address ad and place the result in
	; address ad

DEC	DEC is the mnemonic for Decrease by 1 DEC EAX ; subtract 1 from the number found in EAX
INC	INC is the mnemonic for Increase by 1 INC EAX ; add 1 to the number found in register EAX
CMP	COMP is the mnemonic for Compare CMP DX, 00 ; Compare the DX value with zero JE L7 ; If DX is equal to 0 then jump to label L7

	CMP EDX, 10 ; Compares whether the counter has reached 10 JLE LP1 ; If it is less than or equal to 10, then jump to LP1
JG	JG is the mnemonic for Jump if Greater CMP ECX, [ad] ; Compare the number in ECX with the number in ; address ad JG here ; If in the previous operation the value of the first ; operand (ECX) was greater than the value of the ; second operand ([ad]) then jump to the label 'here'.
JL	JL is the mnemonic for Jump if Less
JE	JE is the mnemonic for Jump if Equal
PUSH	PUSH AX ; Put the value found in register AX on top of the ; stack.
POP	POP CX ; Put the value found at the top of the stack in ; register CX. Remove the top value from the stack.

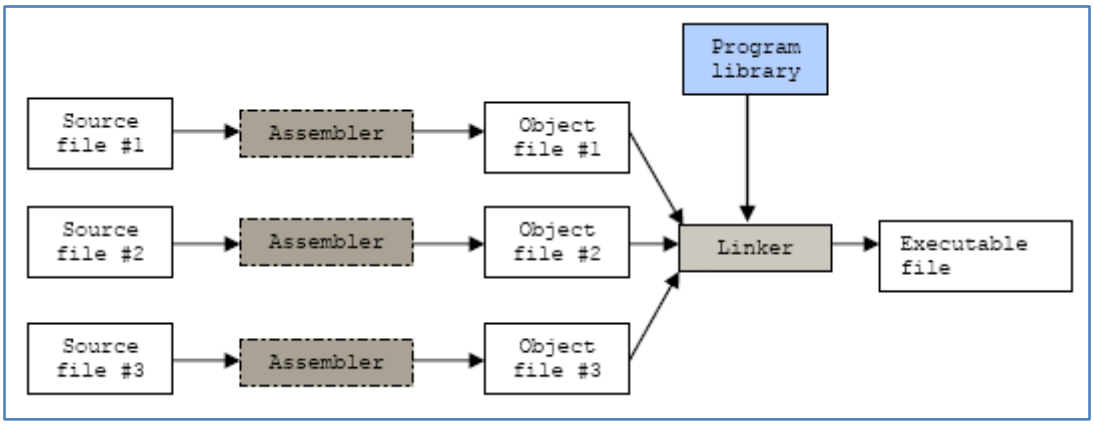
3.6 Assembler, Linker and Loader

An assembler is a program that translates a program written in assembly language into machine code (object code).

A linker (also called link editor and binder) is a program that combines object modules to form an executable program. In addition to combining modules, a linker also replaces symbolic addresses with real addresses.

A loader is an operating system utility that copies programs from a storage device to main memory, where they can be executed. In addition to copying a program into main memory, the loader can also replace virtual addresses with physical addresses.

Most loaders are transparent, i.e., you cannot directly execute them, but the operating system uses them when necessary.



A Linker joining a number of Object Files