Unit III Basic Computer Architecture

Introduction to Computer Architecture

Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with.

As with other contexts and meanings of the word architecture, computer architecture is likened to the art of determining the needs of the user/system/technology, and creating a logical design and standards based on those requirements.

A very good example of computer architecture is von Neumann architecture, which is still used by most types of computers today. This was proposed by the mathematician John von Neumann in 1945. It describes the design of an electronic computer with its CPU, which includes the arithmetic logic unit, control unit, registers, memory for data and instructions, an input/output interface and external storage functions.

There are three categories of computer architecture:

a) **System Design:** This includes all hardware components in the system, including data processors aside from the CPU, such as the graphics processing unit and direct memory access. It also includes memory controllers, data paths and miscellaneous things like multiprocessing and virtualization.

b) **Instruction Set Architecture (ISA):** This is the embedded programming language of the central processing unit. It defines the CPU's functions and capabilities based on what programming it can perform or process. This includes the word size, processor register types, memory addressing modes, data formats and the instruction set that programmers use.

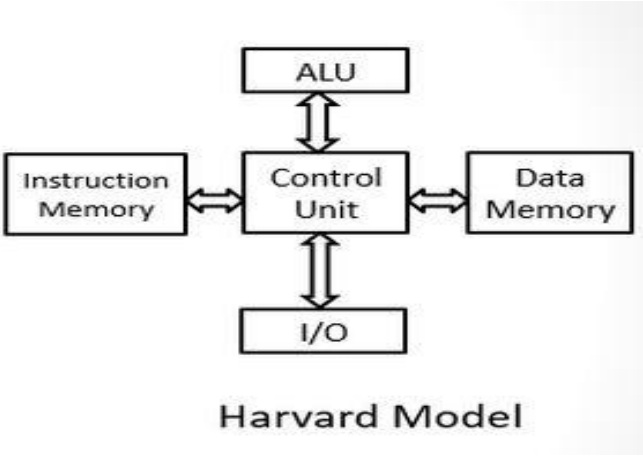
c) **Microarchitecture:** Otherwise known as computer organization, this type of architecture defines the data paths, data processing and storage elements, as well as how they should be implemented in the ISA.

History of Computer Architecture

The first document of Computer Architecture was a correspondence between Charles Babbage and Ada Lovelace, that describes the analytical engine. Here is the example of other early important machines: John Von Neumann and Alan Turing.
Computer architecture is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible with economic status and as well as the technological constraints. In ancient period, computer architectures were designed and prepared on the paper and then, directly built into the final hardware form. Later, in today`s computer architecture, prototypes were physically built in the form of transistor logic (TTL) computer such as the prototypes of the 6800 and the PA-RISC tested, and tweaked before committing to the final hardware form.

Harvard Architecture

The Harvard architecture is a computer architecture with physically separate storage and signal pathways for instructions and data. The term originated from the Harvard Mark I, which stored instructions on punched tape and data in electro-mechanical counters. These early machines had data storage entirely contained within the central processing unit, and provided no access to the instruction storage as data. It required two memories for their instruction and data. Harvard architecture requires separate bus for instruction and data.



Harvard Model

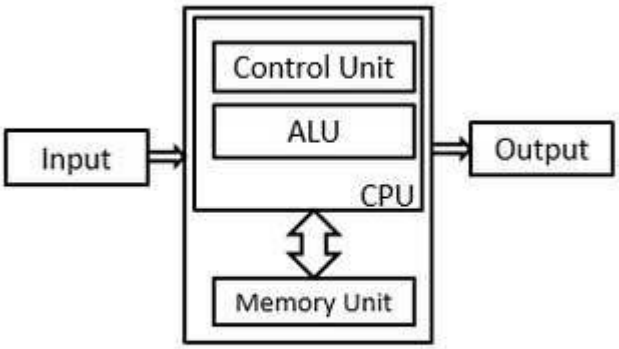
Von Neumann architecture

Von Neumann architecture was first published by John von Neumann in 1945.

His computer architecture design consists of a Control Unit, Arithmetic and Logic Unit (ALU), Memory Unit, Registers and Inputs/Outputs.

Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory. This design is still used in most computers produced today.

The modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same. Von Neumann architecture requires only one bus for instruction and data.



Von Neumann Model

Overview of Computer Organization

Computer organization refers to the operational units and their interconnection that realize the architecture specification. Computer organization deals with physical aspects of computer design, memory and their types and microprocessors design. Computer organization is concerned with the way the hardware components operate and the way they are connected together to form a computer system.

It describes how the computer performs. Ex, circuit design, control signals, memory types and etc.

Computer Organization vs Architecture

1) **Computer Architecture** refers to those attributes of a system that have a direct impact on the logical execution of a program. Examples:
   - the instruction set
   - the number of bits used to represent various data types
   - I/O mechanisms
   - memory addressing techniques

   Computer Organization refers to the operational units and their interconnections that realize the architectural specifications. Examples are things that are transparent to the programmer:
   - control signals
   - interfaces between computer and peripherals
   - the memory technology being used.

2) So, for example, the fact that a multiply instruction is available is a computer architecture issue. How that multiply is implemented is a computer organization issue.

3) **Architecture** is those attributes visible to the programmer
   - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques. e.g. Is there a multiply instruction?

   Organization is how features are implemented
   - Control signals, interfaces, memory technology. e.g. Is there a hardware multiply unit or is it done by repeated addition?

4) **Computer architecture** is concerned with the structure and behavior of computer system as seen by the user.
   Computer organization is concerned with the way the hardware components operate and the way they are connected together to form a computer system.

Memory Hierarchy

The memory unit Is an essential component in any digital computer since It Is needed for storing programs and data. A very small computer with an unlimited application may be able to fulfill its intended task without the use of additional storage capacity. Most general purpose computers would run more efficiently if they were equipped with additional storage beyond the capacity of the main memory. There is just not enough space in one memory unit to accommodate all the programs used in a typical computer. Moreover, most computer users accumulate and continue to accumulate large amounts of data-processing software. Not all accumulated information is needed by the processor at the same time. Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by the CPU.

The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and other backup information. Only programs and data currently needed by the processor reside in main memory. All other information Is stored in auxiliary memory and transferred to main memory when needed.

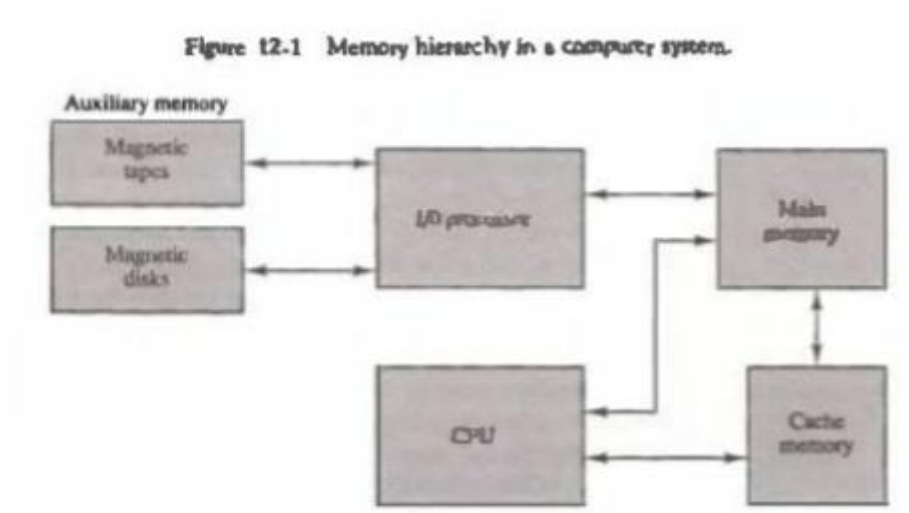


Figure 12-1 Memory hierarchy in a computer system.

The total memory capacity of a computer can be visualized as being a hierarchy of components. The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory accessible to the high-speed processing logic. Above Figure illustrates the components in a typical memory hierarchy. At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files. Next are the magnetic disks used as backup storage. The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor. When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.

Cache Memory

A special very high speed memory called a Cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic. CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory. A technique used to compensate for the mismatch in operating speeds is to employ an extremely fast, small cache between the CPU and main memory whose access time Is dose to processor logic dock cycle time. The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations.

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory. It is placed between the CPU and main memory as illustrated in Fig. below. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

The basic operation of the cache is as follows. When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed. In this manner, some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

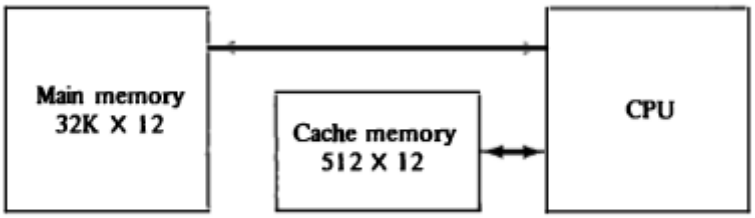


Figure 12-10 Example of cache memory.

The main memory can store 32K words of 12 bits each. The cache is capable of storing 512 of these words at any given time. For every word stored in cache, there is a duplicate copy in main memory. The CPU communicates with both memories. It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

Organization of Hard Disk

A hard disk drive (HDD) is a non-volatile computer storage device containing magnetic disks or platters rotating at high speeds. It is a secondary storage device used to store data permanently. Non-volatile means data is retained when the computer is turned off. A hard disk drive is also known as a hard drive.

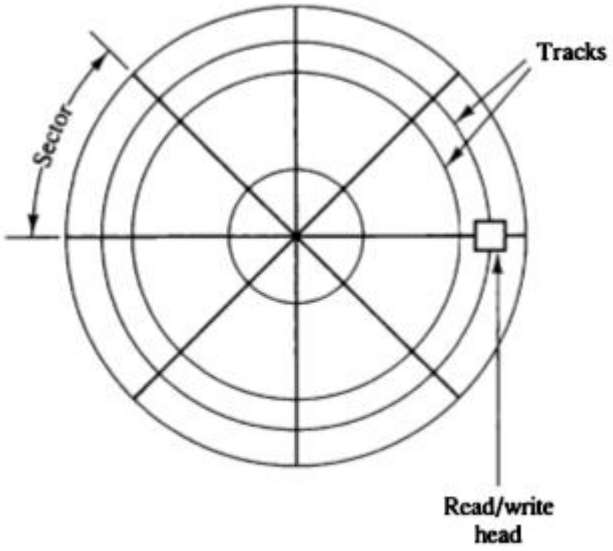


Figure Magnetic disk.

A hard drive consists of the following:

Magnetic platters - Platters are the round plates in the image above. Each platter holds a certain amount of information, so a drive with a lot of storage will have more platters than one with less storage. When information is stored and retrieved from the platters it is done so in concentric circles, called tracks, which are further broken down into segments called sectors.

Arm - The arm is the piece sticking out over the platters. The arms will contain read and write heads which are used to read and store the magnetic information onto the platters. Each platter will have its own arm which is used to read and write data off of it.

Motor - The motor is used to spin the platters from 4,500 to 15,000 rotations per minute (RPM). The faster the RPM of a drive, the better performance you will achieve from it.

When a computer wants to retrieve data off of the hard drive, the motor will spin up the platters and the arm will move itself to the appropriate position above the platter where the data is stored. The heads on the arm will detect the magnetic bits on the platters and convert them into the appropriate data that can be used by the computer. Conversely, when data is sent to the drive, the heads will this time, send magnetic pulses at the platters changing the magnetic properties of the platter, and thus storing your information.

Instruction Code

An instruction code is a group of bits that instruct the computer to perform a specific operation. It is usually divided into parts, each having its own particular interpretation. The most basic part of an instruction code is its operation part. The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement. The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer.

Instruction codes together with data are stored in memory. The computer reads each instruction from memory and places it in a control register. The control then interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of micro operations. Every computer has its own unique instruction set. The ability to store and execute instructions, the stored program concept, is the most important property of a general-purpose computer.

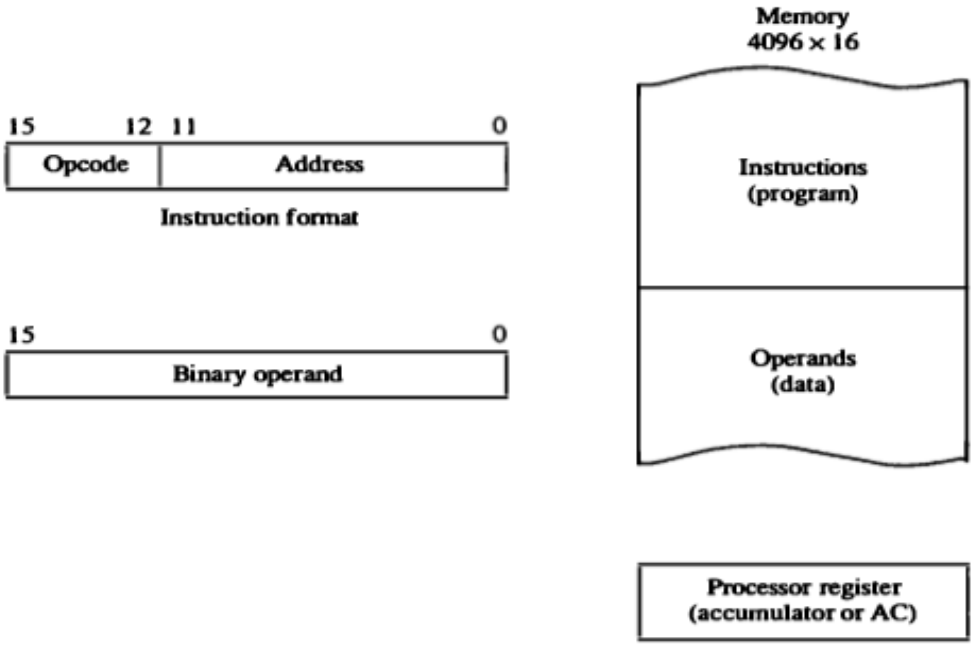
The operation part of an instruction code specifies the operation to be performed. This operation must be performed on some data stored in processor registers or in memory. An instruction code must therefore specify not only the operation but also the registers or the memory words where the operands are to be found, as well as the register or memory word where the result is to be stored.

Stored Program Organization

The simplest way to organize a computer is to have one processor register and an instruction code format with two parts. The first part specifies the operation to be performed and the second specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

Below figure depicts this type of organization. Instructions are stored in one section of memory and data in another. For a memory unit with 4096 words we need 12 bits to specify an address since 212 = 4096. If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code (abbreviated opcode) to specify one out of 16 possible operations, and 12 bits to specify the address of an operand. The control reads a 16-bit instruction from the program portion of memory. It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory. It then executes the operation specified by the operation code.

Figure 5-1 Stored program organization.



Indirect Address

It is sometimes convenient to use the address bits of an instruction code not as an address but as the actual operand. When the second part of an instruction code specifies an operand, the instruction is said to have an immediate operand. When the second part specifies the address of an operand, the instruction is said to have a direct address. This is in contrast to a third possibility called indirect address, where the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found.

In register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.

Example:

MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)

Computer Registers

Registers are the fastest and smallest type of memory elements available to a processor. Registers are normally measured by the number of bits they can hold, for example, an "8-bit register", "32-bit register" or a "64-bit register" (or even with more bits). A processor often contains several kinds of registers, which can be classified according to their content or instructions that operate on them.

TABLE 5-1 List of Registers for the Basic Computer

Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

Common Bus System

The basic computer has eight registers, a memory unit, and a control unit. Paths must be provided to transfer information from one register to another and between memory and registers. The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus.

Four registers, DR, AC, IR, and TR, have 16 bits each. Two registers, AR and PC, have 12 bits each since they hold a memory address. When the contents of AR or PC are applied to the 16-bit common bus, the four most significant bits are set to O's. When AR or PC receive information from the bus, only the 12 least significant bits are transferred into the register.

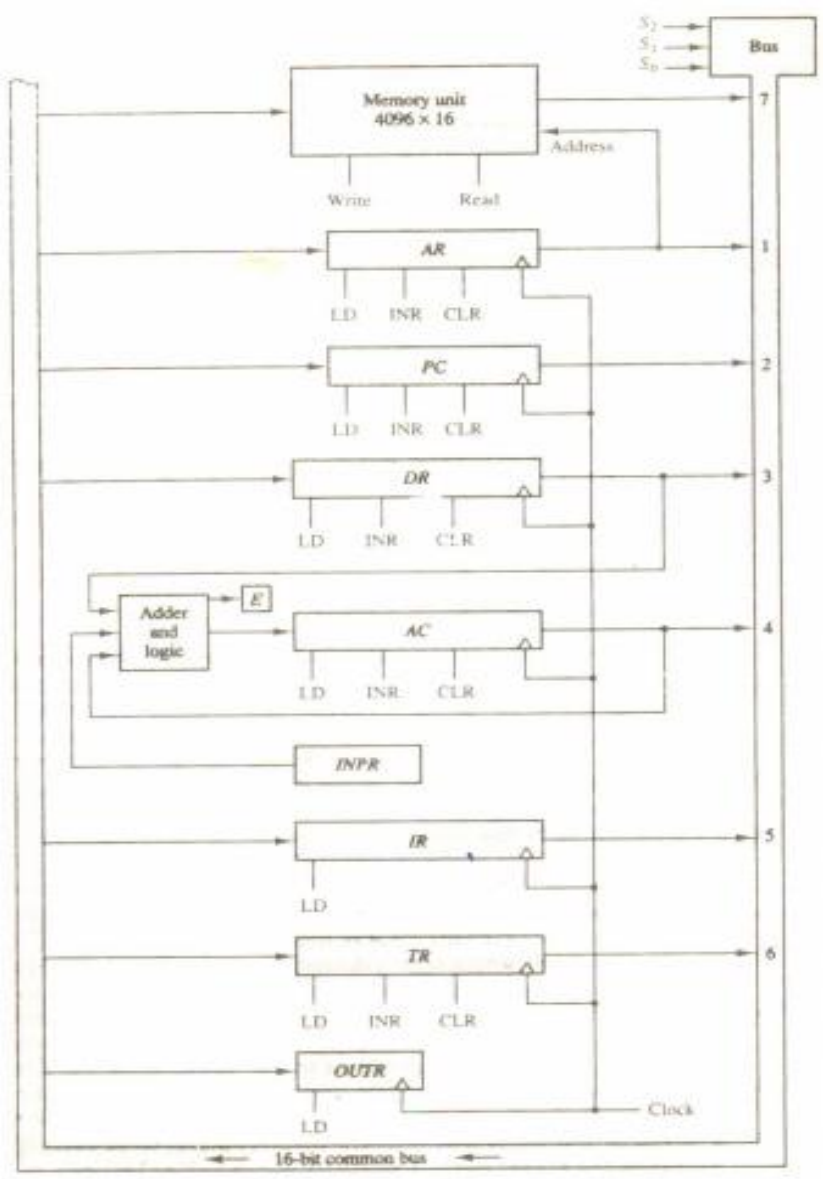


Figure 1.4 Basic registers which are connected to a common bus

The input register INPR and the output register OUTR have 8 bits each and communicate with the eight least significant bits in the bus. INPR is connected to provide information to the bus but OUTR can only receive information from the bus. This is because INPR receives a character from an input device which is then transferred to AC. OUTR receives a character from AC and delivers it to an output device. There is no transfer from OUTR to any of the other registers. The 16lines of the common bus receive information from six registers and the memory unit. The bus lines are connected to the inputs of six registers and the memory. Five registers have three control inputs: LD (load), INR (increment), and CLR (clear).

The input data and output data of the memory are connected to the common bus, but the memory address is connected to AR. Therefore, AR must always be used to specify a memory address. By using a single register for the address, we eliminate the need for an address bus that would have been needed otherwise. The content of any register can be specified for the memory data input during a write operation. Similarly, any register can receive the data from memory after a read operation except AC.

Timing and Control

The timing for all registers in the basic computer is controlled by a master clock generator. The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit. The clock pulses do not change the state of a register unless the register is enabled by a control signal. The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and micro operations for the accumulator.

Instruction Cycle

Time required to execute and fetch an entire instruction is called instruction cycle. A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases. In the basic computer each instruction cycle consists of the following phases:

1. Fetch an instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.