CHAPTER 3

- **Introduction:** History of computer architecture, Overview of computer organization, Memory hierarchy and Cache, Organization of hard disk
- **Instruction Codes**: Stored Program Organization, Indirect address, Computer Registers, Common Bus systems, Instruction set, Timing and Control, Instruction Cycle.

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. A very good example of computer architecture is von Neumann architecture, which is still used by most types of computers today.

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.

The Top Level Computer Structure

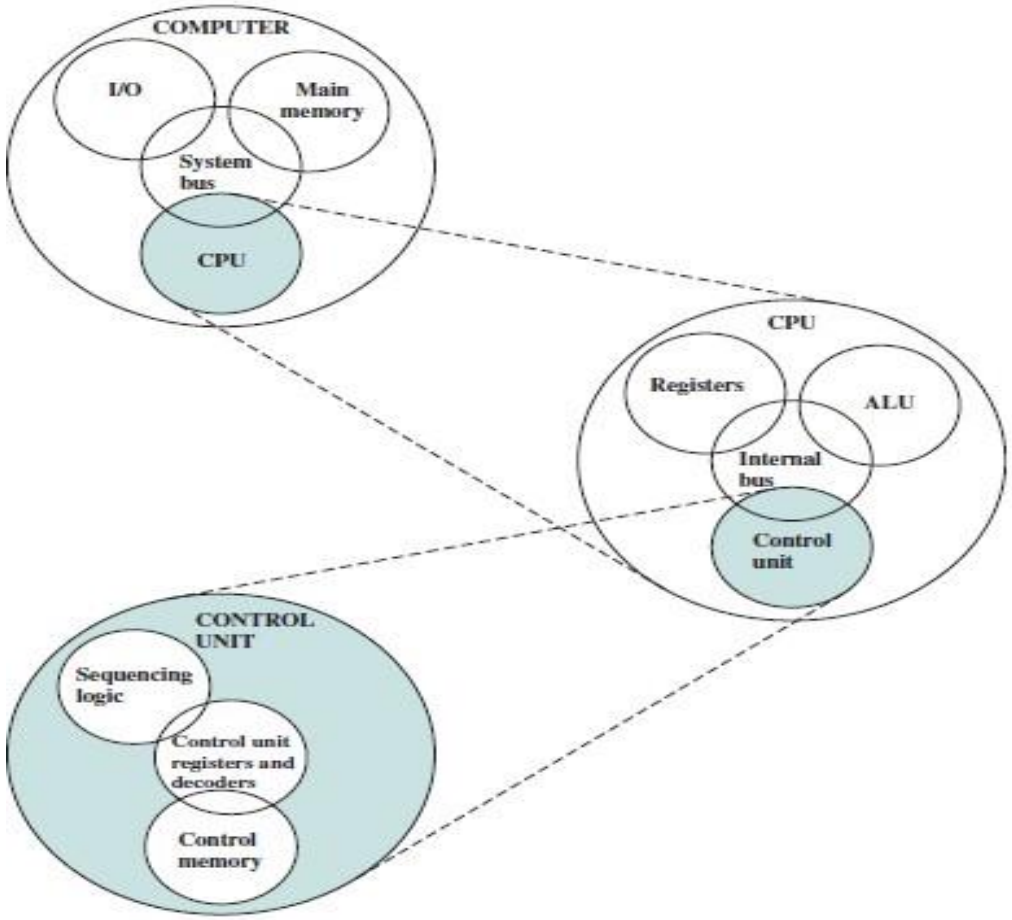


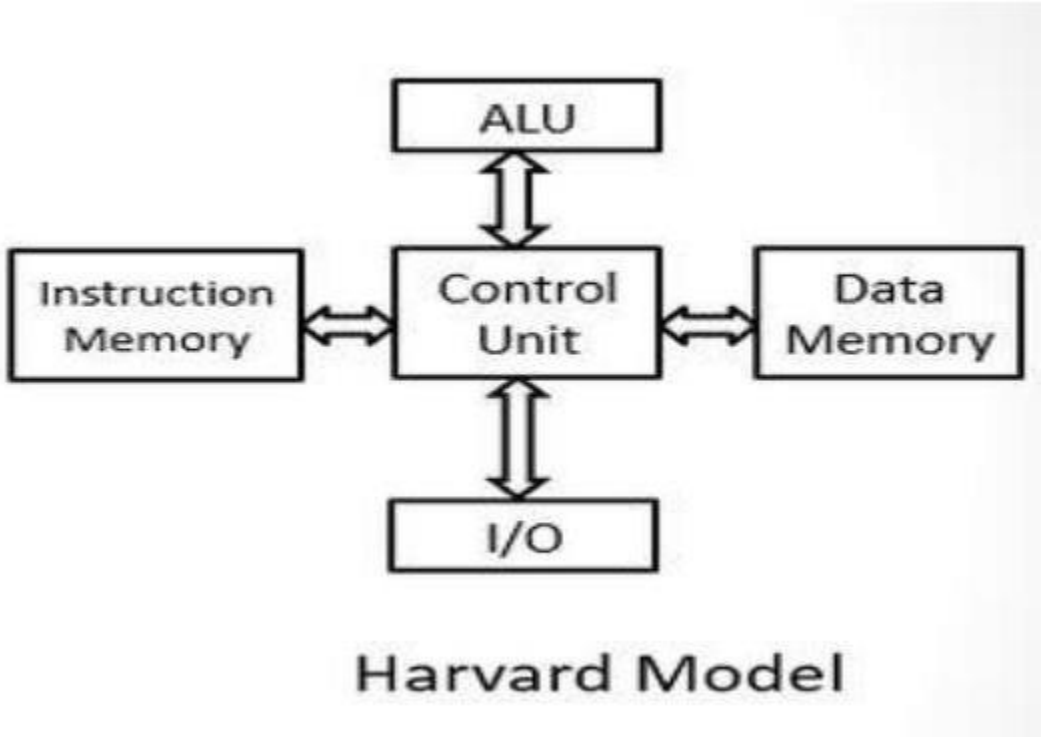
Figure 1.4 The Computer: Top-Level Structure

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.

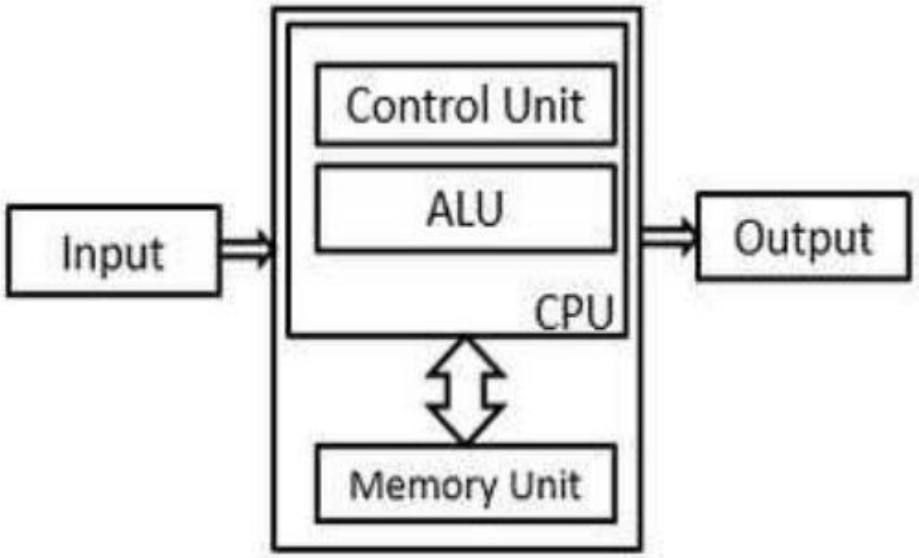
Harvard Architecture

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.



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. Eg: 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.

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.

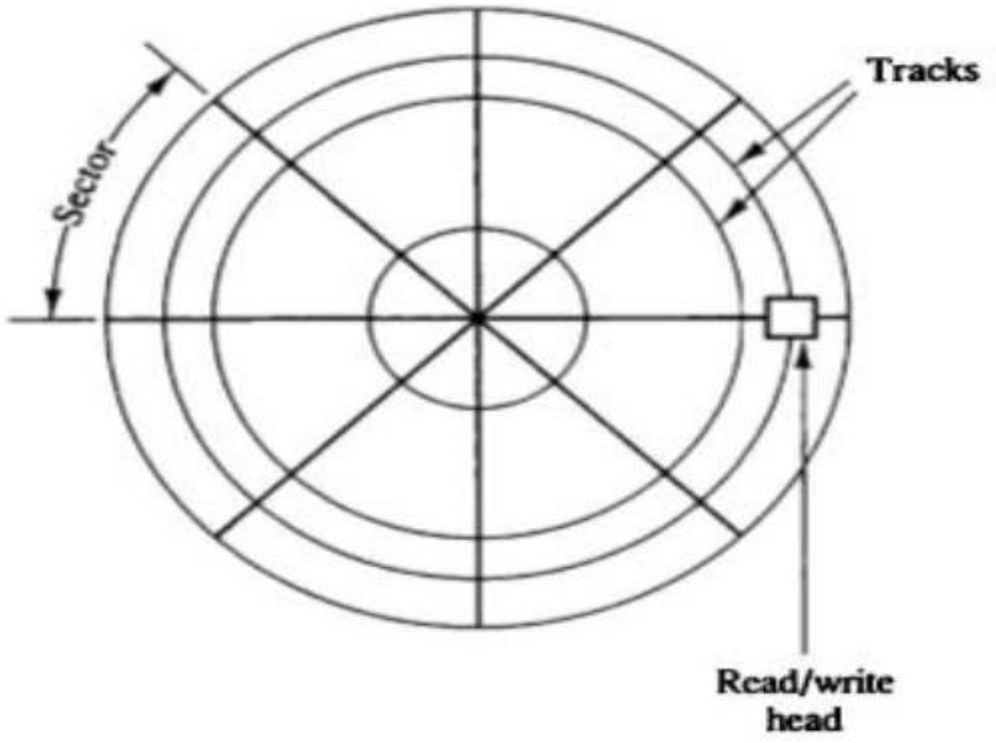


Figure Magnetic disk.

- 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.

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.

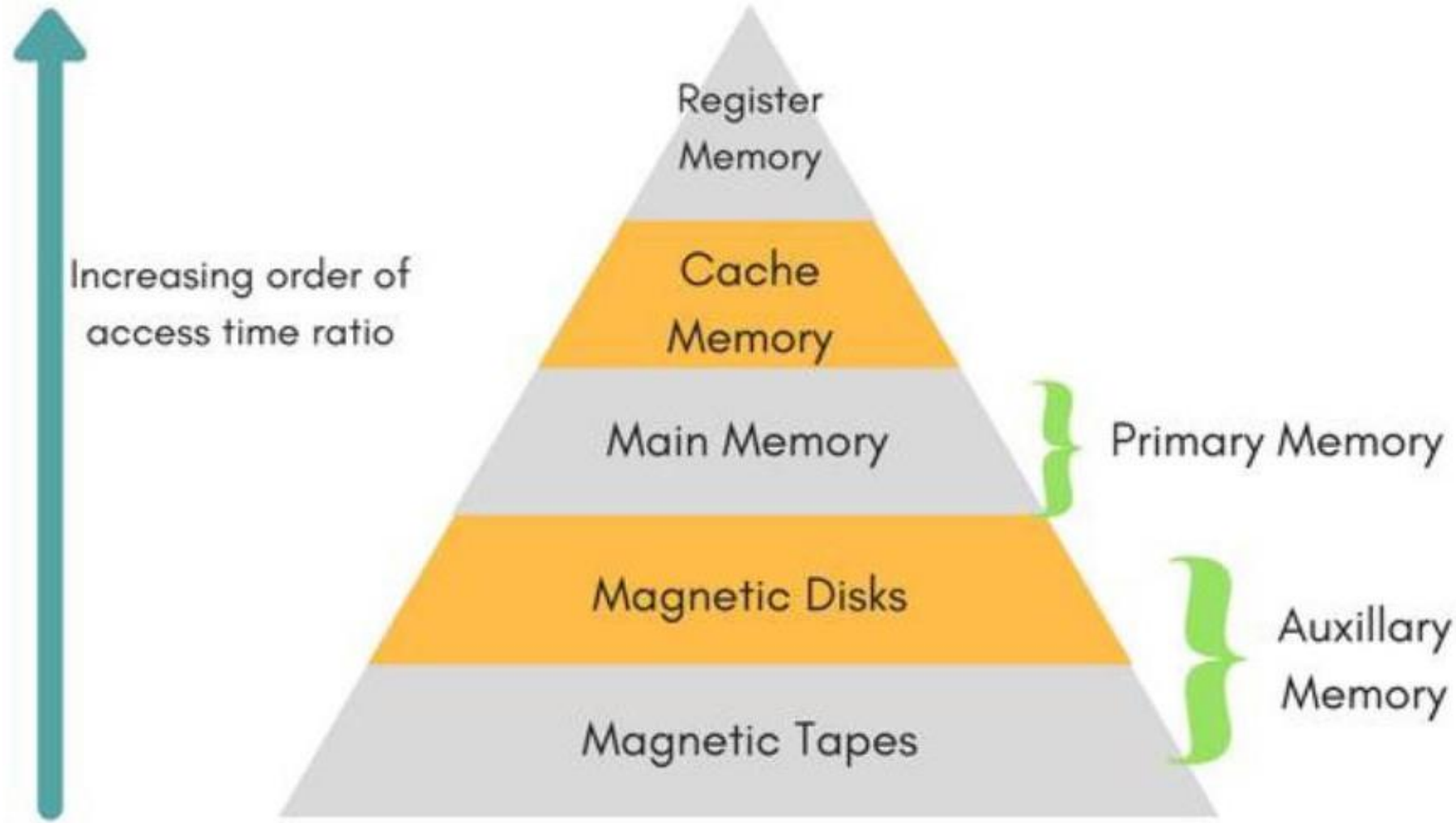
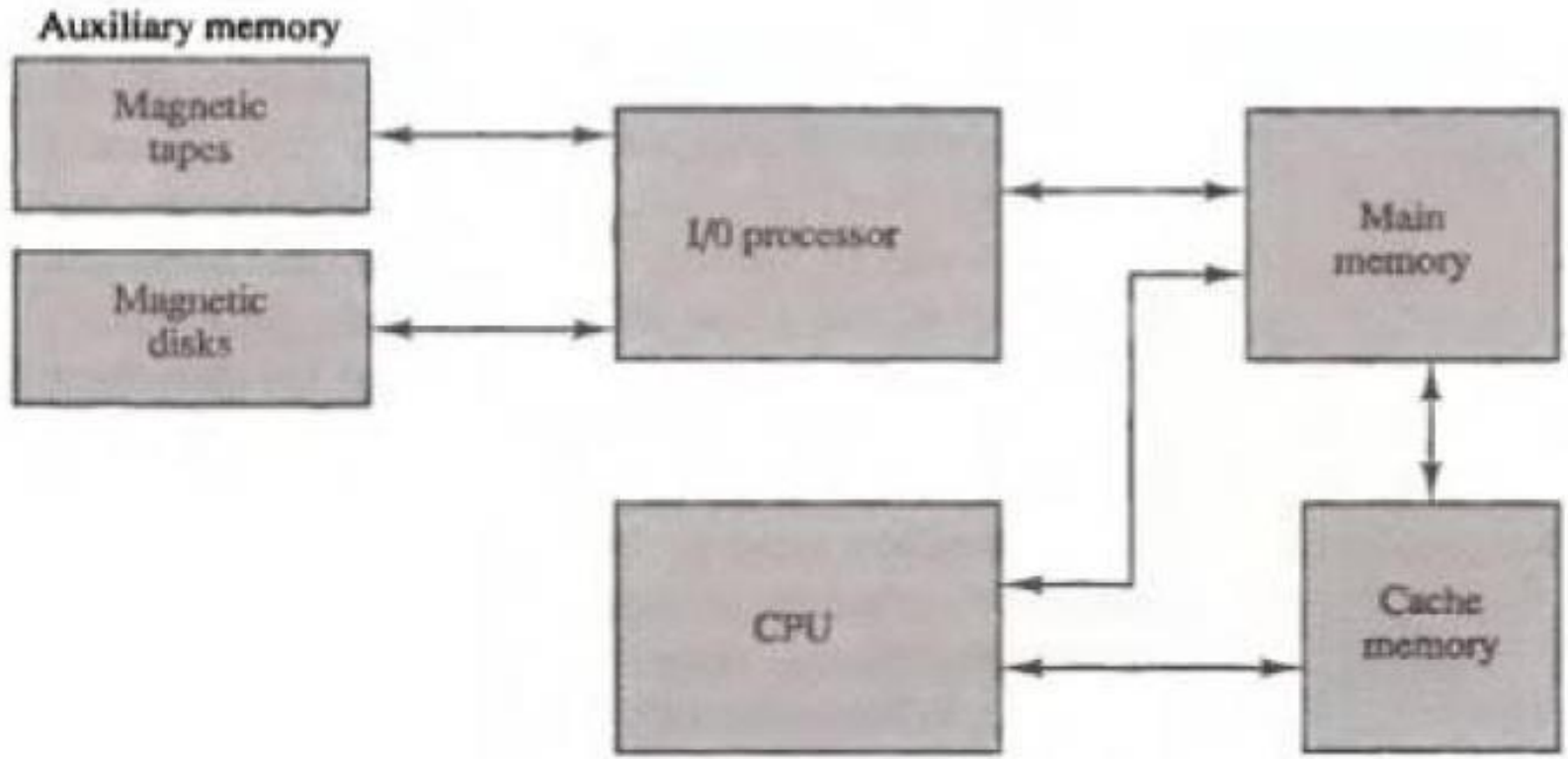

Increasing order of access time ratio
Register Memory
Cache Memory
Main Memory
Magnetic Disks
Magnetic Tapes
Primary Memory
Auxillary Memory

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 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.
- 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.

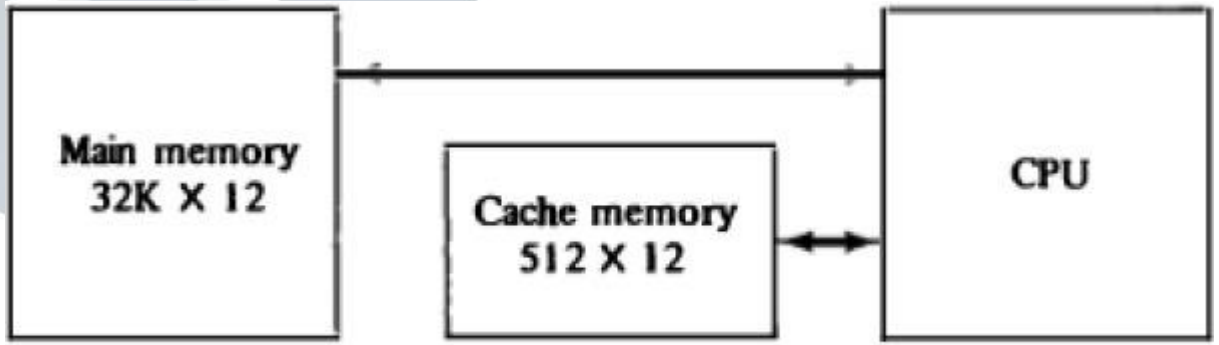


Figure 12-10 Example of cache memory.

Elements of Cache Design

There are a few basic design elements that serve to classify and differentiate cache architectures. They are listed down:

- Cache Addresses
- Cache Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Line Size
- Number of caches

1. CACHE ADDRESSES

• **Logical Address**: When virtual addresses are used, the cache can be placed between the processor and the MMU or between the MMU (memory management unit) and main memory. A logical cache, also known as a virtual cache, stores data using virtual addresses. The processor accesses the cache directly, without going through the MMU.

• **Physical Address:** A physical cache stores data using main memory physical addresses. One advantage of the logical cache is that cache access speed is faster than for a physical cache, because the cache can respond before the MMU performs an address translation.

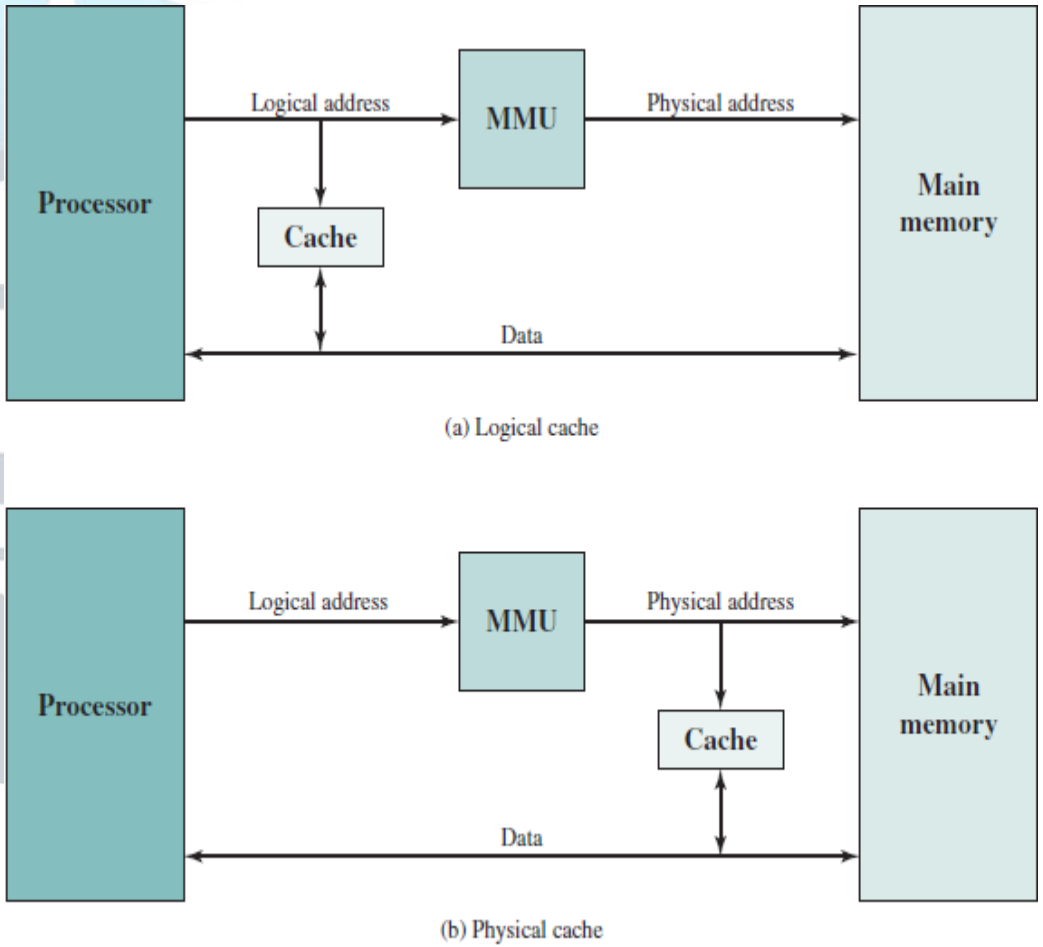


Figure 4.7 Logical and Physical Caches

2. CACHE SIZE:

The size of the cache should be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

3. MAPPING FUNCTION

As there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. Further, a means is needed for determining which main memory block currently occupies a cache line. The choice of the mapping function dictates how the cache is organized. Three techniques can be used: direct, associative, and set associative.

- **DIRECT MAPPING:** The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.
- **ASSOCIATIVE MAPPING:** Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache.
- **SET-ASSOCIATIVE MAPPING:** Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches. With set-associative mapping, block can be mapped into any of the lines of set j.

4. REPLACEMENT ALGORITHM’S

Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced. For direct mapping, there is only one possible line for any particular block, and no choice is possible. For the associative and set associative techniques, a replacement algorithm is needed. To achieve high speed, such an algorithm must be implemented in hardware.

- Least Recently Used (LRU): use for replacing recently activities.
- Least Frequently Used(LFU): use for replacing frequently activities.
- First In First Out (FIFO): use for replacing long term activities.

.

5. **WRITE POLICY**

• **Write Through:** The information is written to both block in cache and the block in lower-level memory.

• **Write Back:** The information is written only to block in cache. The modified cache block is written to main memory only when it is replaced.

• **Dirty Bit:** It is indicate whether the block is dirty (modified while in cache) .

• **Cache Hit**: A memory reference where the required data is found in cache.

• **Cache Miss:** A memory reference where the required data is not found in cache

6. LINE SIZE

- Another design element is the line size. When a block of data is retrieved and placed in the cache, not only the desired word but also some number of adjacent words is retrieved. Basically, as the block size increases, more useful data are brought into the cache. The hit ratio will begin to decrease, however, as the block becomes even bigger and the probability of using the newly fetched information becomes less than the probability of reusing the information that has to be replaced.
- The relationship between block size and hit ratio is complex, depending on the locality characteristics of a particular program, and no definitive optimum value is found as of yet.

7. NUMBER OF CACHES

When caches were originally introduced, the typical system had a single cache. More recently, the use of multiple caches has become an important aspect. There are two design issues surrounding number of caches.

MULTILEVEL CACHES: Most contemporary designs include both on-chip and external caches. The simplest such organization is known as a two-level cache, with the internal cache designated as level 1 (L1) and the external cache designated as level 2 (L2). There can also be 3 or more levels of cache. This helps in reducing main memory accesses.

UNIFIED VERSUS SPLIT CACHES: Earlier on-chip cache designs consisted of a single cache used to store references to both data and instructions. This is the unified approach. More recently, it has become common to split the cache into two: one dedicated to instructions and one dedicated to data. These two caches both exist at the same level. This is the split cache. Using a unified cache or a split cache is another design issue.

INTRODUCTION: Description of Basic Computer

- We introduce here a basic computer whose operation can be specified by the register transfer statements. Internal organization of the computer is defined by the sequence of microoperations it performs on data stored in its registers. Every different processor type has its own design (different registers, buses, microoperations, machine instructions, etc). Modern processor is a very complex device. It contains:

– Many registers

– Multiple arithmetic units, for both integer and floating point calculations

– The ability to pipeline several consecutive instructions for execution speedup

However, to understand how processors work, we will start with a simplified processor model. M. Morris Mano introduces a simple processor model; he calls it a ―Basic Computer. The Basic Computer has two components, a processor and memory.

The memory has 4096 words in it

–4096 = 2^12, so it takes 12 bits to select an address in memory

Each word is 16 bits long

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 2^12 = 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.

- The program (instruction) as well as data (operand) is stored in the same memory. If the instruction needs data, the data is found in the same memory and accessed. This feature is called stored program organization.

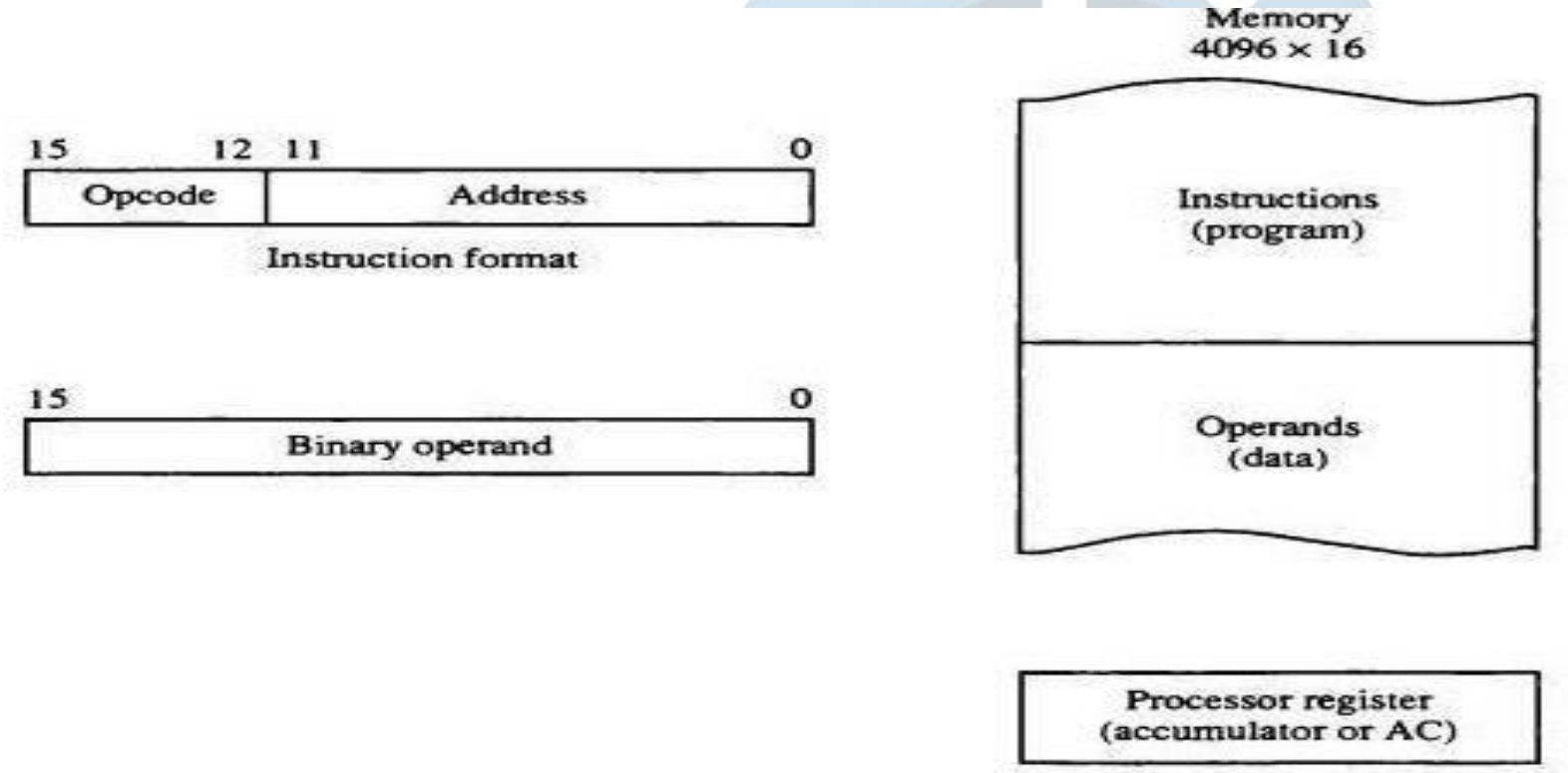


Figure: Stored Program Organisation

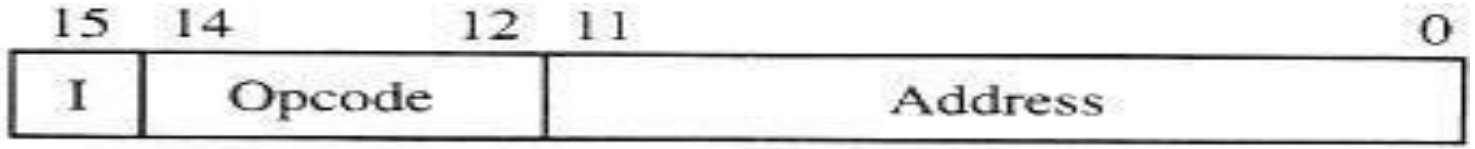
Instruction Formats

Instruction Format of Basic Computer:

A computer instruction is often divided into two parts

- An *opcode* (Operation Code) that specifies the operation for that instruction
- An *address* that specifies the registers and/or locations in memory to use for that operation.

In the Basic Computer, since the memory contains 4096 (= 2^{12}) words, we needs 12 bit to specify the memory address that is used by this instruction. In the Basic Computer, bit 15 of the instruction specifies the *addressing mode* (0: direct addressing, 1: indirect addressing). Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode.



(a) Instruction format

Addressing Modes:

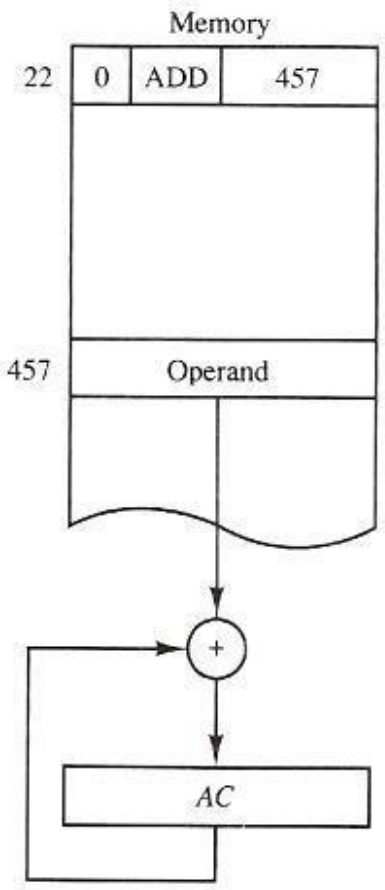
Addressing Modes:

The address field of an instruction can be represented in two ways

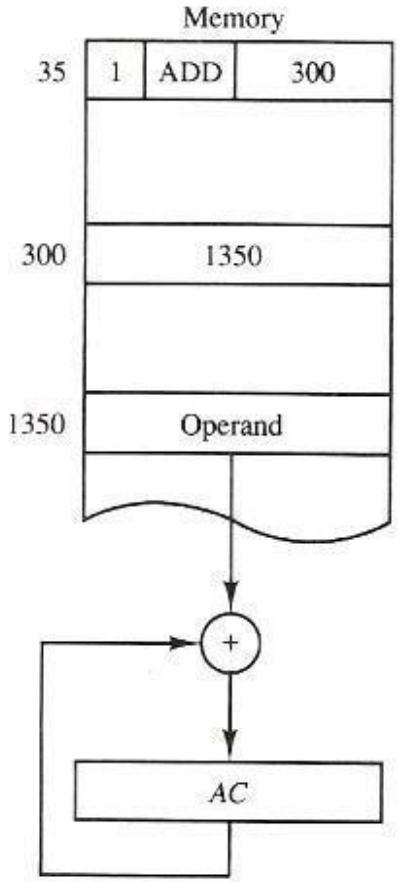
– **Direct address**: the address operand field is effective address (the address of the operand)

– **Indirect address:** the address in operand field contains the memory address where effective address resides.

Note: **Effective Address (EA):** The address, where actual data resides is called effective address.



(b) Direct address



(c) Indirect address

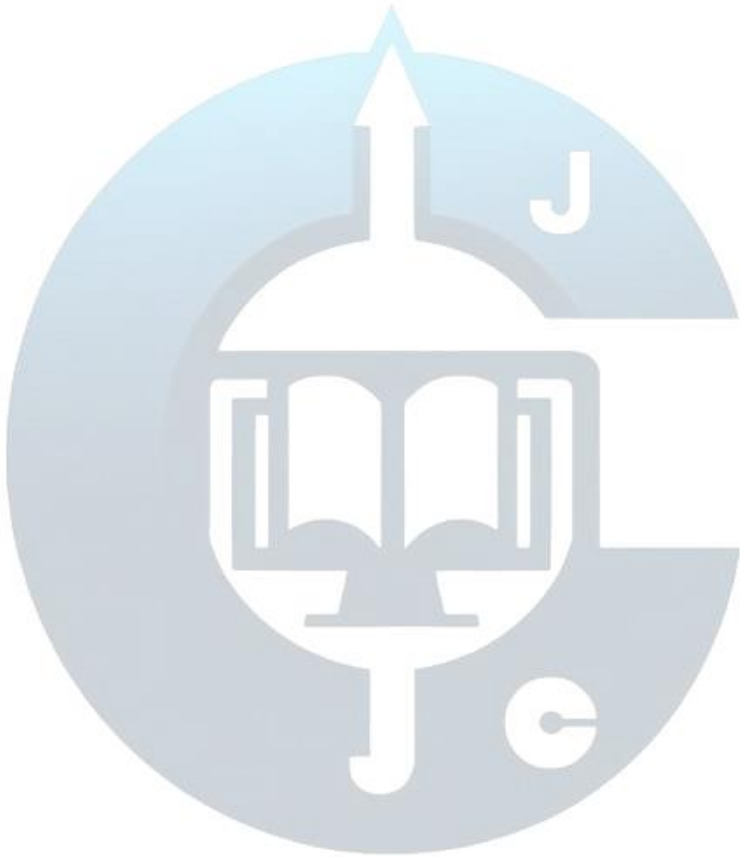
- A direct address instruction is shown in figure b. It is placed in address 22 in memory. The I bit is 0, so the instruction is recognized as a direct address instruction. The opcode specifies an ADD instruction, and the address part is the binary equivalent of 457. The control finds the operand in the memory at address 457 and adds it to the content of AC.
- The instruction in address 35 shown in figure c has a mode bit I=1. Therefore, it is recognized as indirect address instruction. The address part is the binary equivalent of 300. The control goes to address 300 to find the address of an operand. The address of an operand in this case is 1350. The operand found in address 1350 is then added to the content of AC. The indirect address instruction needs two references to memory to fetch an operand. The first reference is needed to read the address of the operand and second is for the operand itself. The effective address of direct addressing mode is 457 and that of indirect addressing mode is 1350.

Computer Registers

- Computer instructions are normally stored in the consecutive memory locations and are executed sequentially one at a time. Thus computer needs processor registers for manipulating data and holding memory address which are shown in the following table:

Symbol	Size	Register Name	Description
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



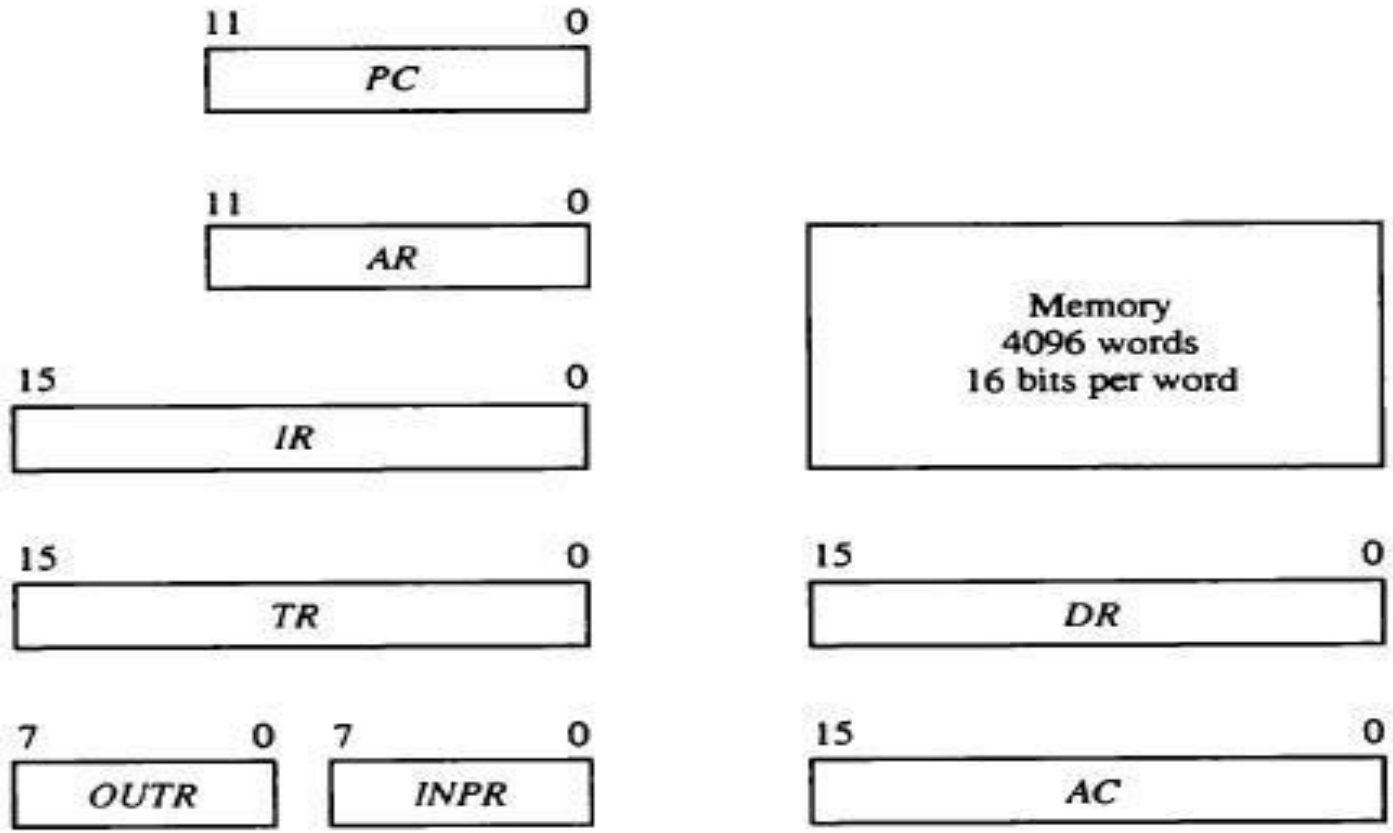


Figure 5-3 Basic computer registers and memory.

- Since the memory in the Basic Computer only has 4096 (=2^{12}) locations, PC and AR only needs 12 bits. Since the word size of Basic Computer only has 16 bit, the DR, AC, IR and TR needs 16 bits. The Basic Computer uses a very simple model of input/output (I/O) operations.

–Input devices are considered to send 8 bits of character data to the processor

–The processor can send 8 bits of character data to output devices

The Input Register (INPR) holds an 8-bit character gotten from an input device and the Output Register (OUTR) holds an 8-bit character to be sent to an output device.

Q) Why the address bus (AR) and PC of basic computer is 12 bit?

Q) What is the use of PC? – it holds the address of the next instruction to be executed.

Common Bus System

- The basic computer has eight registers, a memory unit, and a control unit. These registers, memory and control unit are connected using a path (bus) so that information can be transferred to each other. If separate buses are used for connecting each registers, it will cost high. The cost and use of extra buses can be reduced using a special scheme in which many registers use a common bus, called *common bus system*
- The registers in the Basic Computer are connected using a bus. This gives a savings in circuitry over complete connections between registers. Three control lines, S2, S1, and S0 control which register the bus selects as its input.

S_2	S_1	S_0	Register
0	0	0	X (nothing)
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

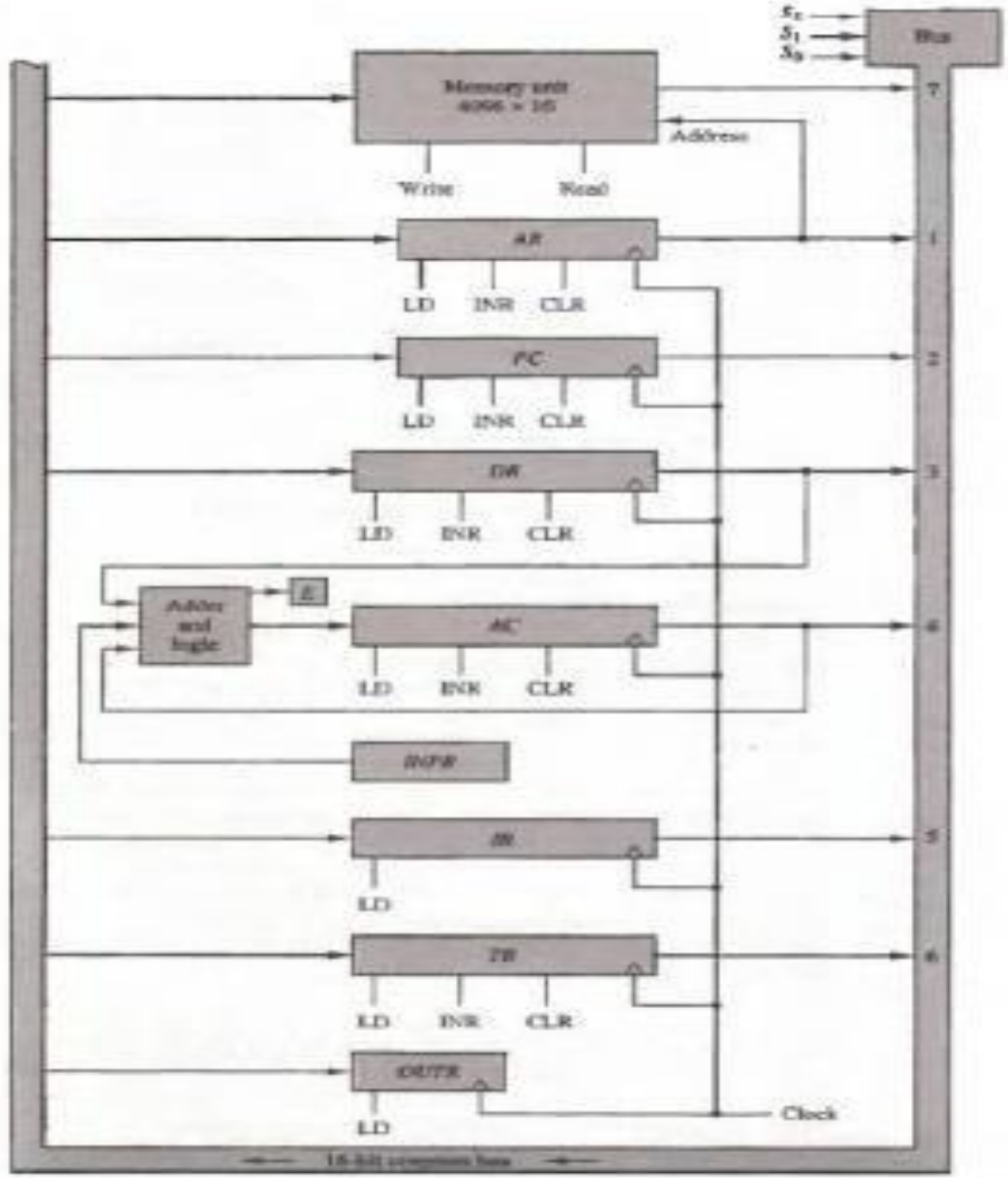


Figure 5-4 Basic computer registers connected to a common bus.

- The particular register whose LD (Load) input is enabled receives the data from the bus during the next clock pulse transition. The memory receives the contents of the bus when its write input is activated.
- The memory places its 16 bit output onto the bus when the read input is activated and the value of S2, S1, and S0 is 111.
- Four registers DR, AC, IR and TR have 16 bits each and two registers AR and PC have 12 bits each since they hold a memory address.
- When the content of AR and PC are applied to the 16-bit common bus, the four MSBs are set to 0’s.
- When AR and PC receive information from the bus, only the 12 least significant bits are transferred into the register.
- The input register INPR and the output register OUTR have 8 bits each and communicate with the eight least significant bits in the bus.

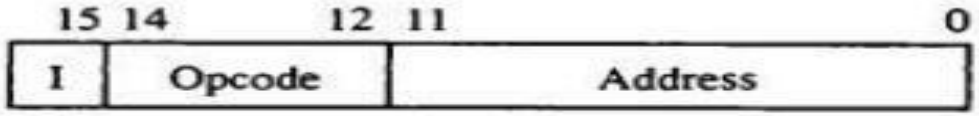
- INPR is connected to provide information to the bus but OUTR can only receive information from the bus.
- Five registers have three control inputs: LD (load), INR (increment) and CLR(Clear).
- Two registers have only a LD input.
- AR must always be used to specify memory address; therefore memory address is connected to AR. The 16 inputs of AC come from an Adder and Logic Circuit. This circuit has three inputs.
- Set of 16- bit inputs come from the outputs of AC.
- Set of 16-bit inputs come from the data register DR.
- Set of 8-bit inputs come from the input register INPR.
- The result of an addition is transferred to AC and the end carry out of the addition is transferred to flip flop E (extended AC-bit).
- The clock transition at the end of the cycle transfers the content of the bus into the designated destination register and the output of the adder and logic circuit into AC.

Computer Instructions

The basic computer has 3 instruction code formats. Type of the instruction is recognized by the computer control from 4-bit positions 12 through 15 of the instruction.

- **Memory-Reference Instructions**
- **Register-Reference Instructions**
- **Input-Output Instructions**

1. **Memory-Reference Instructions:**

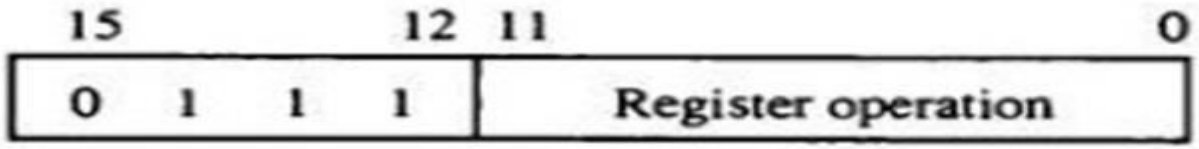


(Opcode = 000 through 110)

(a) Memory – reference instruction

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero

2. **Register-Reference Instructions**

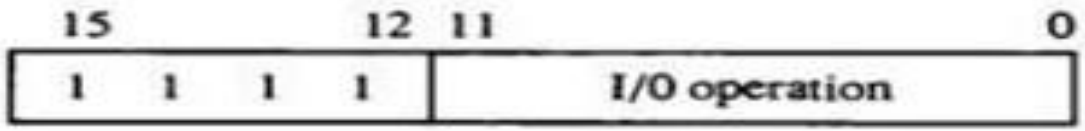


(Opcode = 111, $I = 0$)

(b) Register – reference instruction

CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right AC and E
CIL	7040	Circulate left AC and E
INC	7020	Increment AC
SPA	7010	Skip next instr. if AC is positive
SNA	7008	Skip next instr. if AC is negative
SZA	7004	Skip next instr. if AC is zero
SZE	7002	Skip next instr. if E is zero
HLT	7001	Halt computer

3. **Input-Output Instructions**



(Opcode = 111, $I = 1$)

(c) Input – output instruction

INP	F800	Input character to AC
OUT	F400	Output character from AC
SKI	F200	Skip on input flag
SKO	F100	Skip on output flag
ION	F080	Interrupt on
IOF	F040	Interrupt off

Instruction Set Completeness

An instruction set is said to be complete if it contains sufficient instructions to perform operations in following categories:

Functional Instructions

- Arithmetic, logic, and shift instructions
- Examples: ADD, CMA, INC, CIR, CIL, AND, CLA

Transfer Instructions

- Data transfers between the main memory and the processor registers
- Examples: LDA, STA

Control Instructions

- Program sequencing and control
- Examples: BUN, BSA, ISZ

Input/output Instructions

- Input and output
- Examples: INP, OUT

Instruction set of Basic computer is complete because:

- ADD, CMA (complement), INC can be used to perform addition and subtraction and CIR (circular right shift), CIL (circular left shift) instructions can be used to achieve any kind of shift operations. Addition subtraction and shifting can be used together to achieve multiplication and division. AND, CMA and CLA (clear accumulator) can be used to achieve any logical operations.
- LDA instruction moves data from memory to register and STA instruction moves data from register to memory.
- The branch instructions BUN, BSA and ISZ together with skip instruction provide the mechanism of program control and sequencing.
- INP instruction is used to read data from input device and OUT instruction is used to send data from processor to output device.

Timing and Control Unit

Control Unit

Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them. There are two types of control organization:

a) Hardwired Control

b) Microprogrammed Control

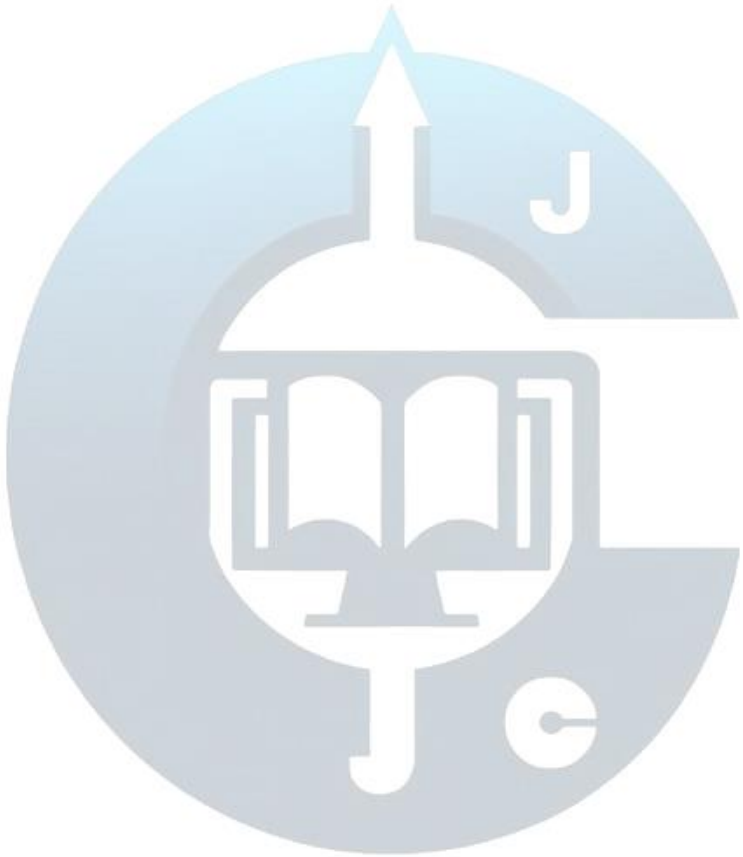
a) Hardwired Control

- CU is made up of sequential and combinational circuits to generate the control signals.
- If logic is changed, we need to change the whole circuit.
- Expensive
- Fast

b) Microprogrammed Control

- A control memory on the processor contains microprograms that activate the necessary control signals.
- If logic is changed, we only need to change the microprogram.
- Cheap

- Slow



Control Unit of a Basic Computer (Hardwired Control)

The block diagram of a hardwired control unit is shown below. It consists of two decoders, a sequence counter, and a number of control logic gates.

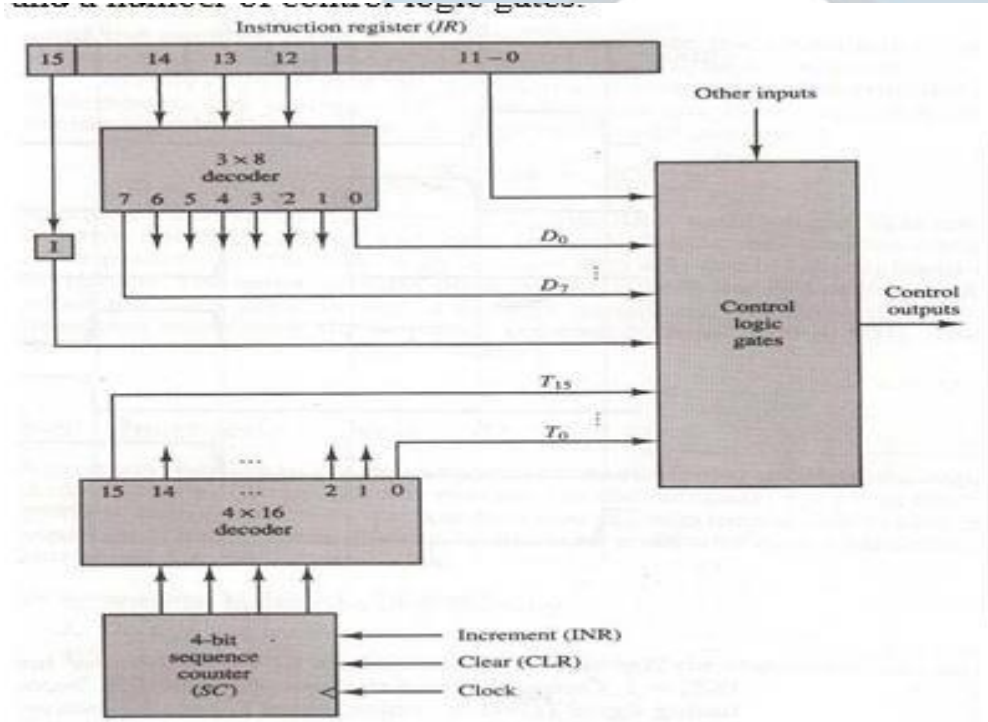


Fig: Control unit of a basic computer

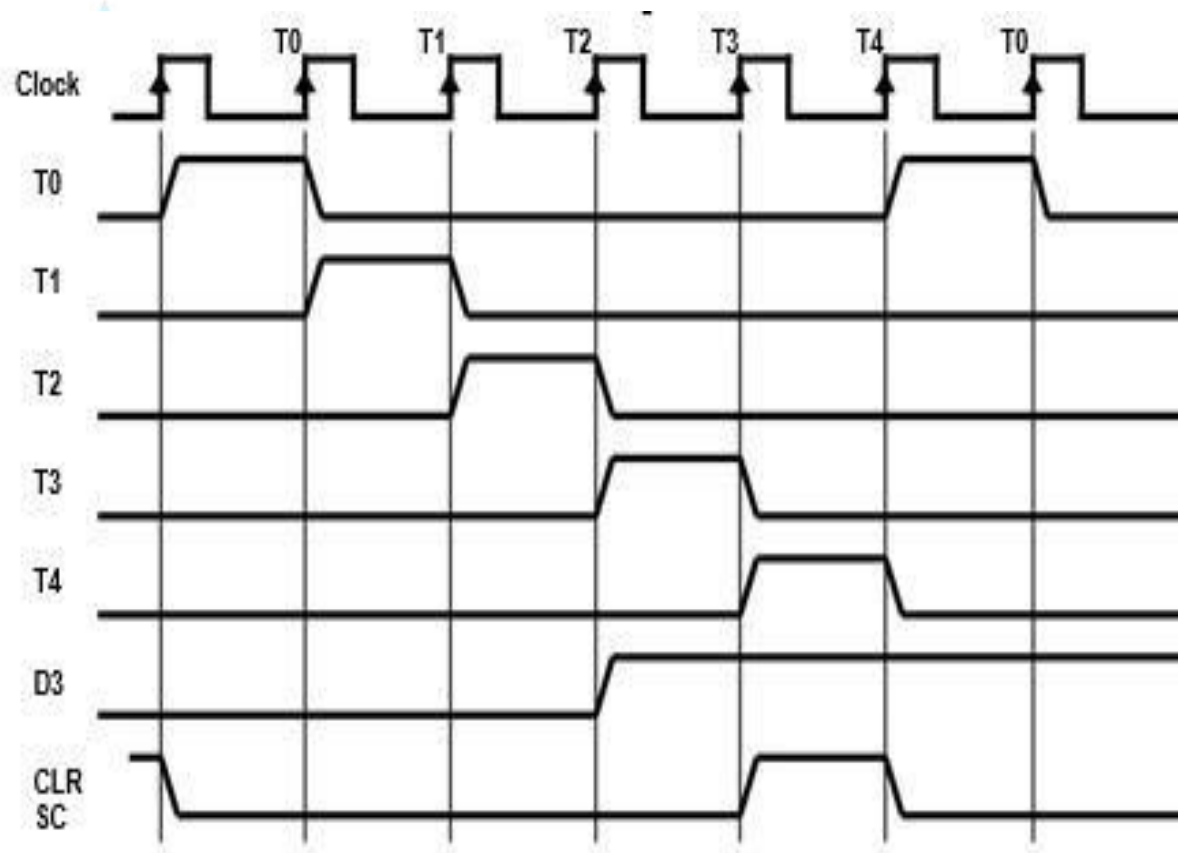
Mechanism:

- An instruction read from memory is placed in the instruction resister (IR) where it is decoded into three parts: **I** bit, **operation code** and bits **0 through 11**.
- The operation code bit is decoded with 3 x 8 decoder producing 8 outputs D_0 through D_7.
- Bit 15 of the instruction is transferred to a flip-flop I.
- And operand bits are applied to control logic gates.
- The 16 outputs of 4-bit sequence counter (SC) are decoded into 16 timing signals T_0 through T_{15}.

This means instruction cycle of basic computer cannot take more than 16.

Timing Signal

- Generated by 4-bit sequence counter and 4x16 decoder.
- The SC can be incremented or cleared.
- Example: T_0, T_1, T_2, T_3, T_4, T_0, T_1 . . .
- Assume: At time T4, SC is cleared to 0 if decoder output D3 is active:
- D_3T_4: SC← 0



Instruction Cycle

Processing required for complete execution of an instruction is called instruction cycle.

In Basic Computer, a machine instruction is executed in the following cycle:

- Fetch an instruction from memory
- Decode the instruction
- Read the effective address from memory if the instruction has an indirect address
- Execute the instruction
- Upon the completion of step 4, control goes back to step 1 to fetch, decode and execute the next instruction. This process is continued indefinitely until HALT instruction is encountered.

Fetch and Decode

Sequence of steps required for fetching instruction from memory to CPU internal register is known as fetch cycle. The microoperations for the fetch and decode phases can be specified by the following resister transfer statements: (first two steps for fetch and third step for decode)

```
T0: AR ← PC  (S0S1S2=010, T0=1)
T1: IR ← M [AR],  PC ← PC + 1   (S0S1S2=111, T1=1)
T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)
```

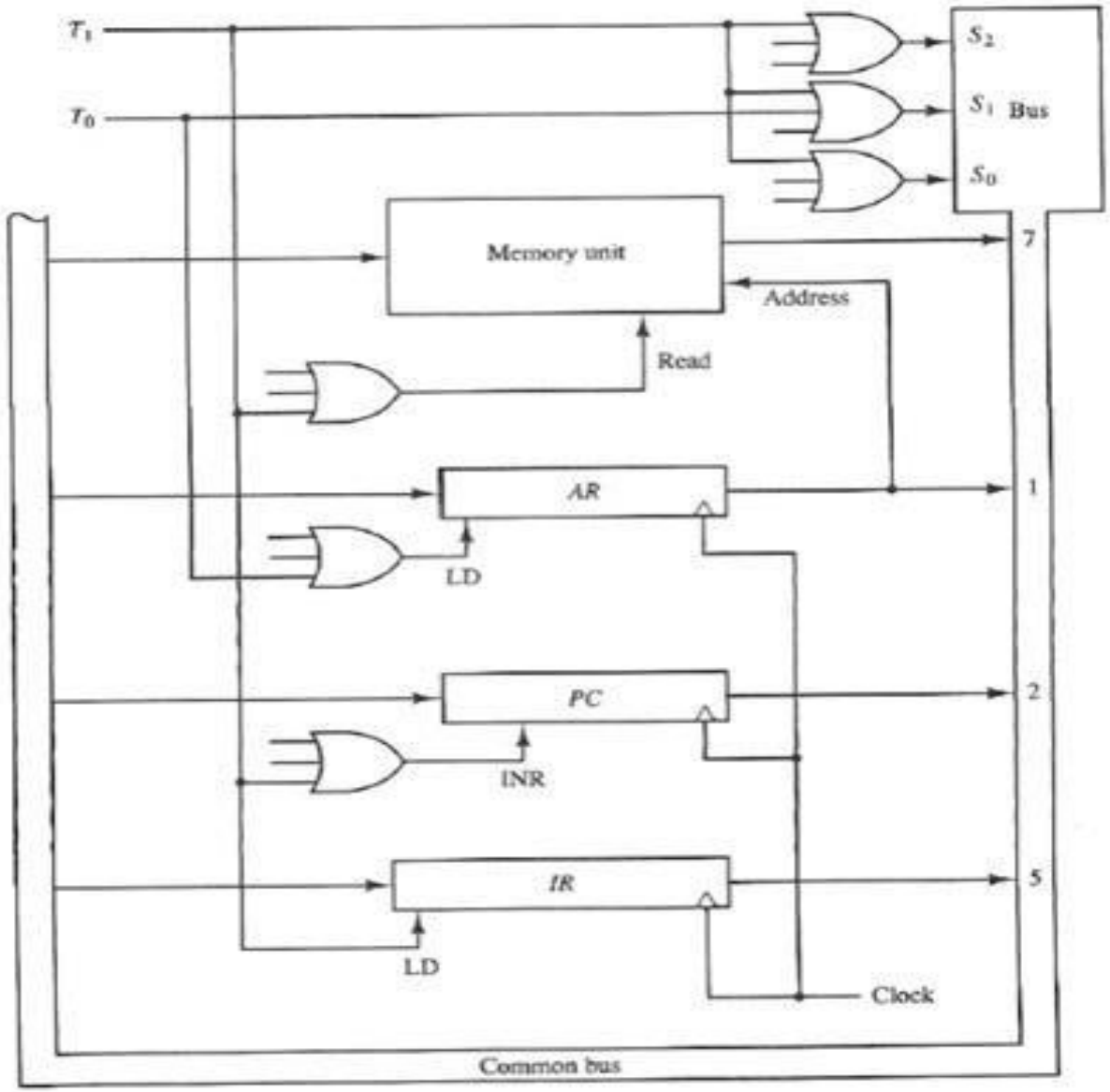


Fig: Resister transfers for the fetch phase

It is necessary to transfer the address from PC to AR during clock transition associated with the timing signal T_0. The instruction read from memory is then placed in IR with clock transition associated with the timing signal T_1. At the same time, PC is incremented by one to prepare for the next instruction in the program. At time T_2, the opcode in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.

NOTE: SC is incremented after each clock pulse to produce the sequence T_0, T_1 and T_2.

- **Determine the type of the instruction**

The timing signal that is active after decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory. Following flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after decoding.

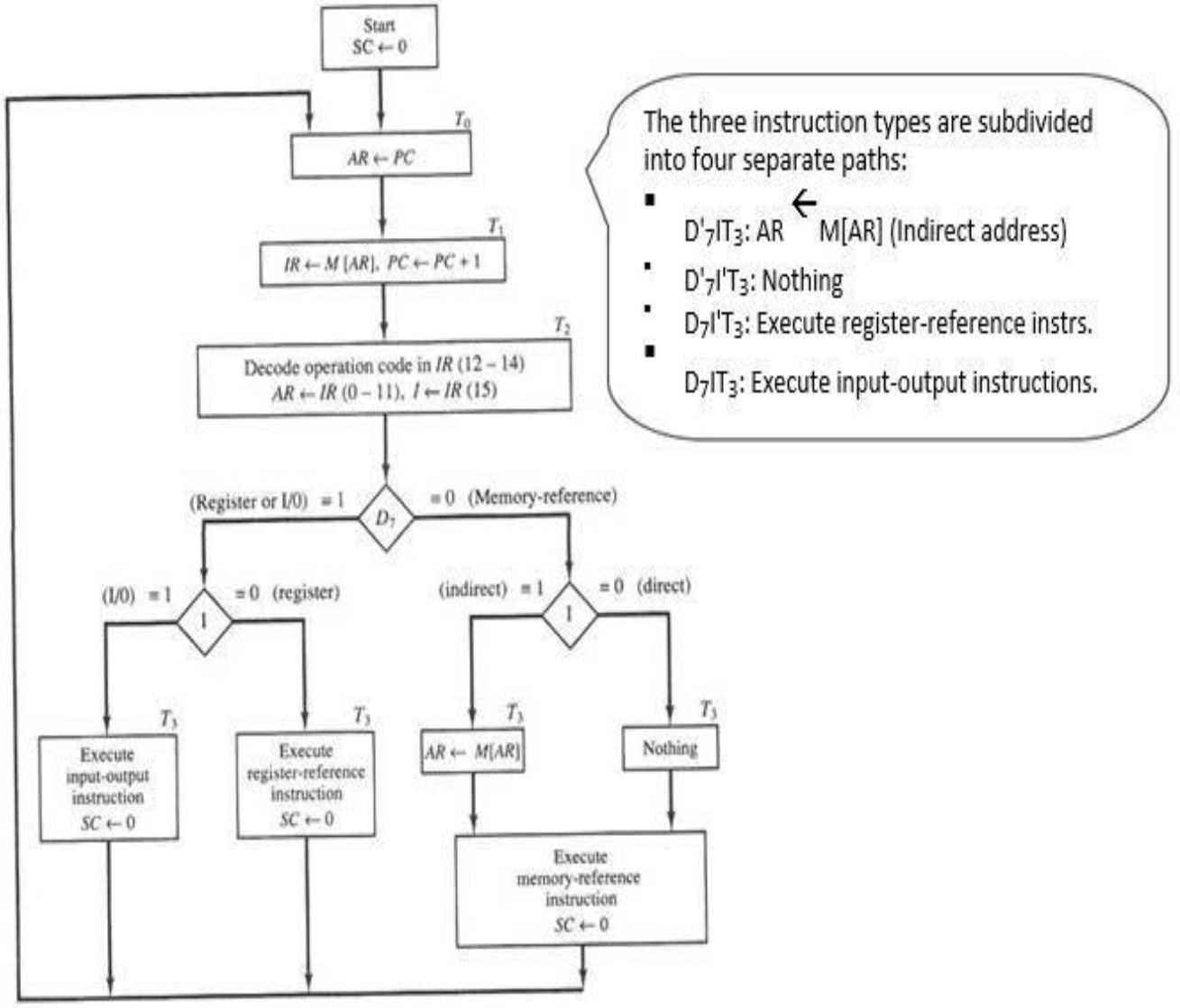


Fig: Flowchart for instruction cycle (Initial configuration)

Then, among decoded, D7 determines which type of instruction.

1) If D7 = 1, it will be either register-reference or input-output instruction.

- If I = 1, input-output instruction is executed during T3.
- If I = 0, register-reference instruction is executed during T3.

2) If D7 = 0, it will be memory-reference instruction.

- If I = 1, indirect addressing mode instruction during T3.
- If I = 0, direct addressing mode instruction during T3.

The SC is reset after executing each instruction.