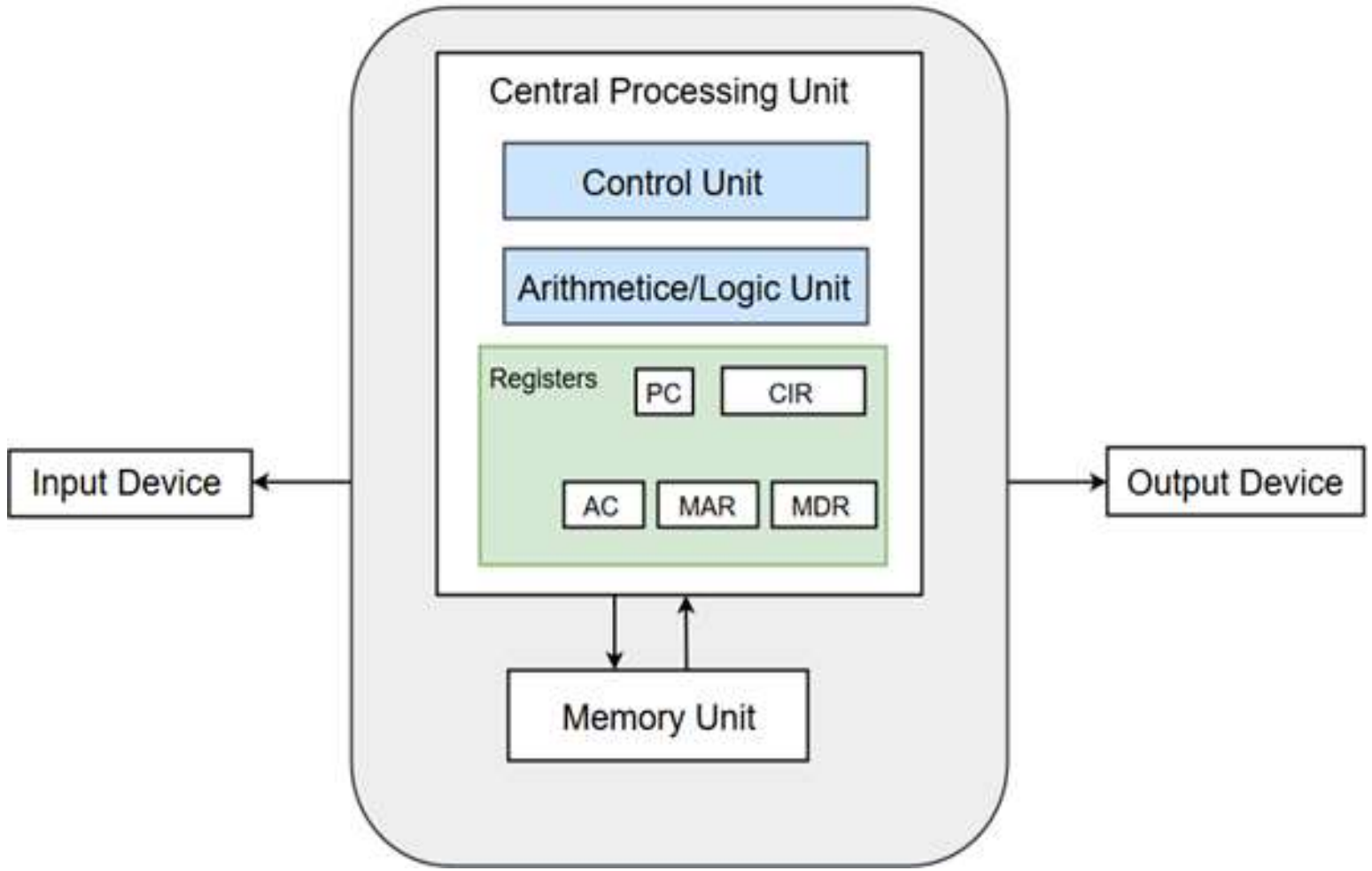
BASICS

Basics

- Computer: An automatic electronic apparatus for making calculations
- Consists of input, output, storage, arithmetic, logic and control units
- Program: sequence of instructions which operates on data to perform certain tasks
- Reads data in binary form
  - Bit
  - Byte
  - Word

Von-Neumann Basic Structure:



Basic Operational Concepts

- The primary function of a computer system is to execute a program, sequence of instructions. These instructions are stored in computer memory.

- These instructions are executed to process data which are already loaded in the computer memory through some input devices.

- After processing the data, the result is either stored in the memory for further reference, or it is sent to the outside world through some output port.

- To perform the execution of an instruction, in addition to the arithmetic logic unit, and control unit, the processor contains a number of registers used for temporary storage of data and some special function registers.

- The special function registers include program counters (PC), instruction registers (IR), memory address registers (MAR) and memory and memory data registers (MDR).

- The Program counter is one of the most critical registers in CPU.

- The Program counter monitors the execution of instructions. It keeps track on which instruction is being executed and what the next instruction will be.

- The instruction register IR is used to hold the instruction that is currently being executed.

- The contents of IR are available to the control unit, which generate the timing signals that control, the various processing elements involved in executing the instruction.

- The two registers MAR and MDR are used to handle the data transfer between the main memory and the processor.

- The MAR holds the address of the main memory to or from which data is to be transferred.

- The MDR contains the data to be written into or read from the addressed word of the main memory.

- The processor communicates in two ways.

- Polling enables the processor software to check each of the input and output devices frequently. During this check, the processor tests to see if any devices needs to communicate or not.

- Interrupt method provides an external asynchronous input that informs the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device.

INSTRUCTION CODES

OVERVIEW

- Internal organization of a digital system is defined by the sequence of micro operations
- User controls the system via programs
- Program: set of instructions that specify operation, operands and sequencing
- Instructions: group of bits
  - 2 parts

Instruction code

Instruction code is a group of bits that instruct the computer to perform a specific operation

Instruction code is divided into parts, each having its own interpretation

- Operation code
- Register or memory word

Operation code

It is a group of bits that define operations such as add, subtract, multiply, shift andcomplemeny

- The no. of bits in the opcode depends on the total no. of operations available in the computer
- It will be at least n bits for a given 2^n (or less) distinct operations

Register or memory word

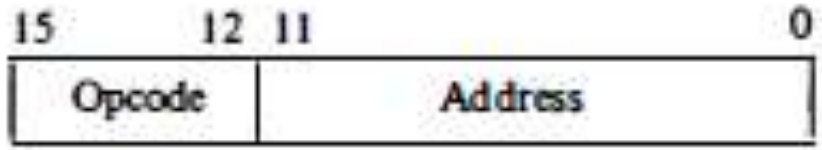
This part specifies where the result of the operations should be stored

- Memory words are specified by their address
- CPU registers are specified by assigning another binary code of k bits that specifies one of 2^k registers.

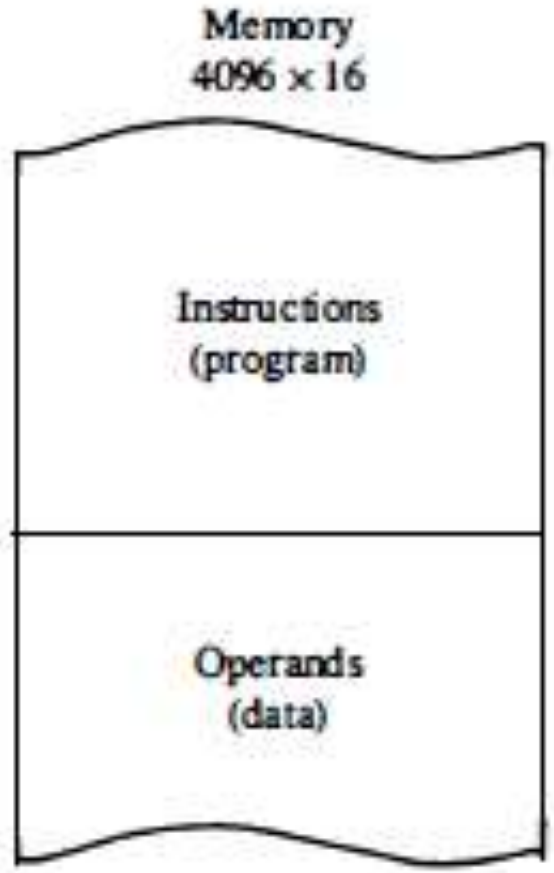
Stored program organization

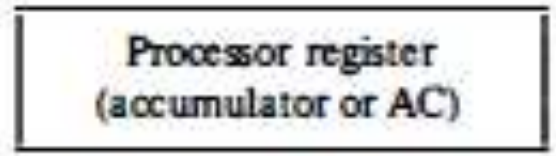
The ability to store and execute instructions is known as stored program concept

- Simplest way to organize a computer is to have one processor register and an instruction code format with two parts operation code and address (memory address to find operand) (shown in fig)



Instruction format

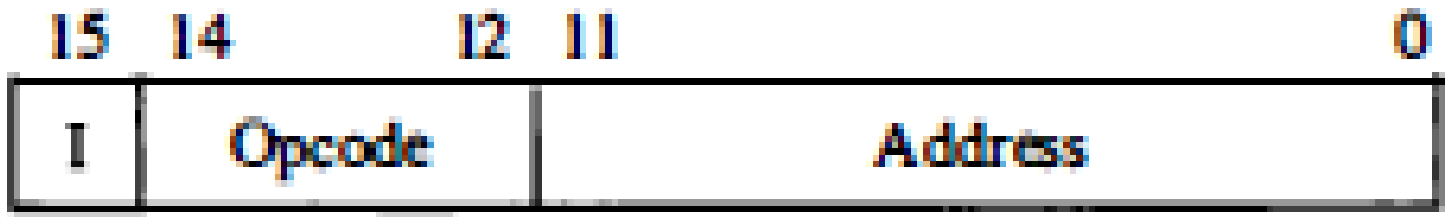




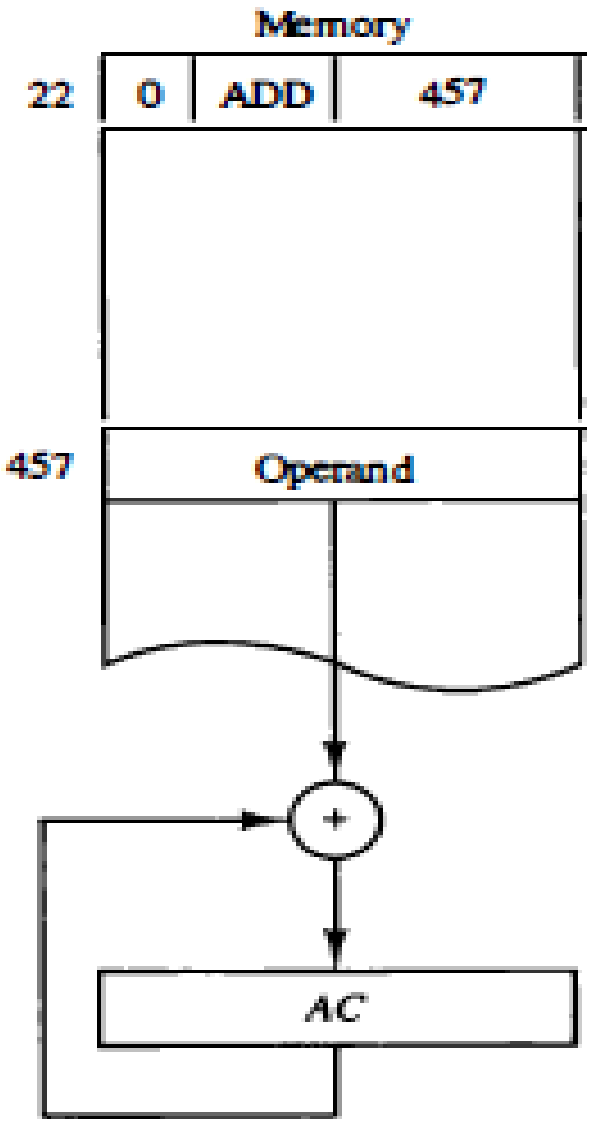
Instruction codes

- 3 types
  - Immediate
  - Direct (addrs of operand)
  - Indirect (addrs of addrs of operand)

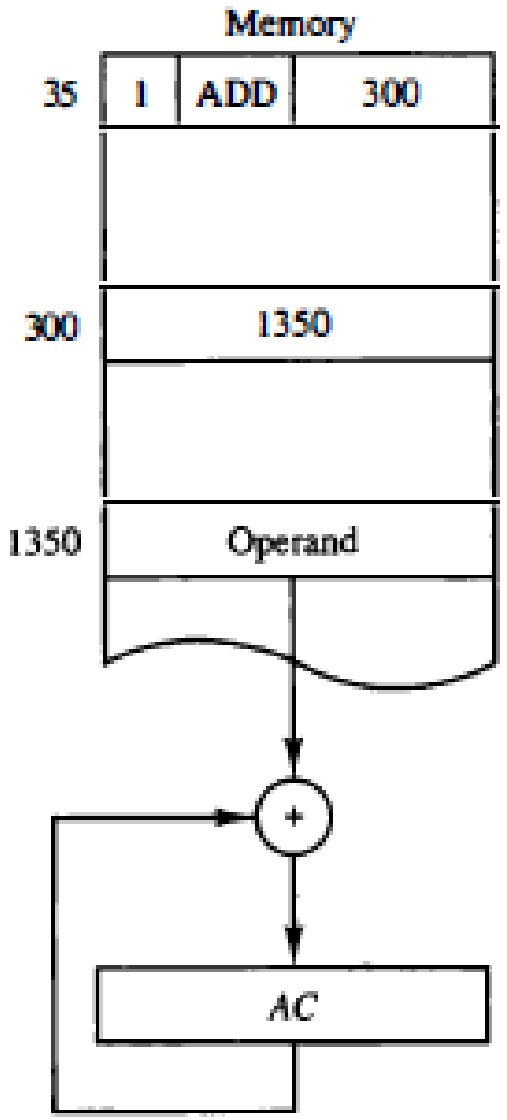
Instruction Format



Direct Address



Indirect Address



COMPUTER REGISTERS

- Computer instructions are normally stored in consecutive memory locations
- The control reads an instruction from a specific address in memory and executes it
- It then continues by reading the next instruction in sequence and executes it, and so on.
- This type of instruction sequencing needs a counter to calculate the address of the next instruction after execution of the current instruction

- It is also necessary to provide a register in the control unit for storing the instruction
- The computer needs processor registers for manipulating data and a register for holding a memory address
- The memory unit has a capacity of 4096 words and each word contains 16 bits.
- 12 bits of an instruction word are needed to specify the address of an operand
- 3 bits for the operation part of the instruction
- 1 bit to specify a direct or indirect address

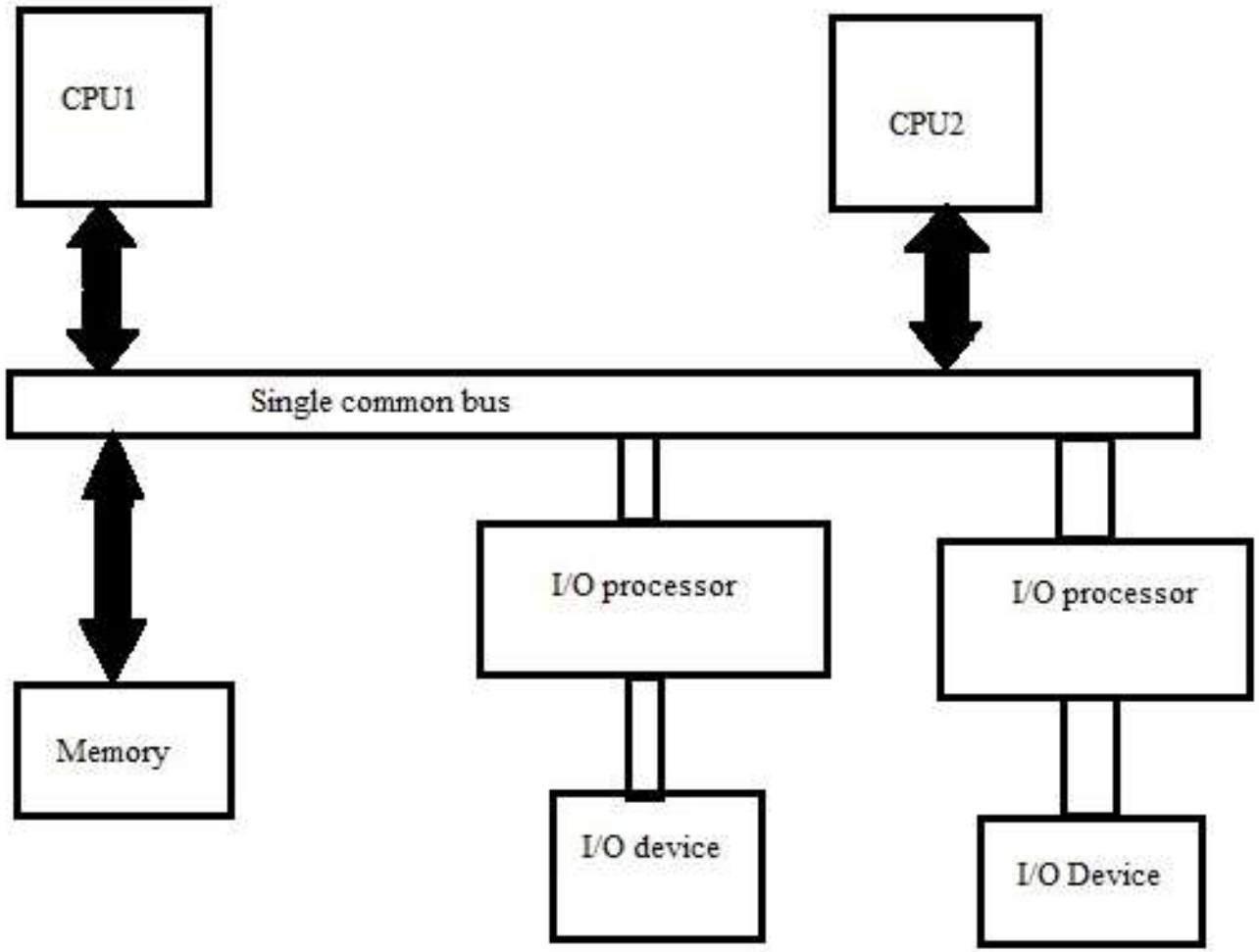
Registers

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
- Efficient scheme for transferring information in a system is to have a common bus.

Single bus system



- A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I.
- I is equal to 0 for direct address and to 1 for indirect address
- The register reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction.
- A register-reference instruction specifies an operation on AC register.
- An operand from memory is not needed
- Therefore, the other 12 bits are used to specify the operation or test to be executed.

- Input-output instruction does not need a reference to memory and is recognized by the operation code Ill with a 1 in the leftmost bit of the instruction

- The remaining 12 bits are used to specify the type of input-output operation or test performed

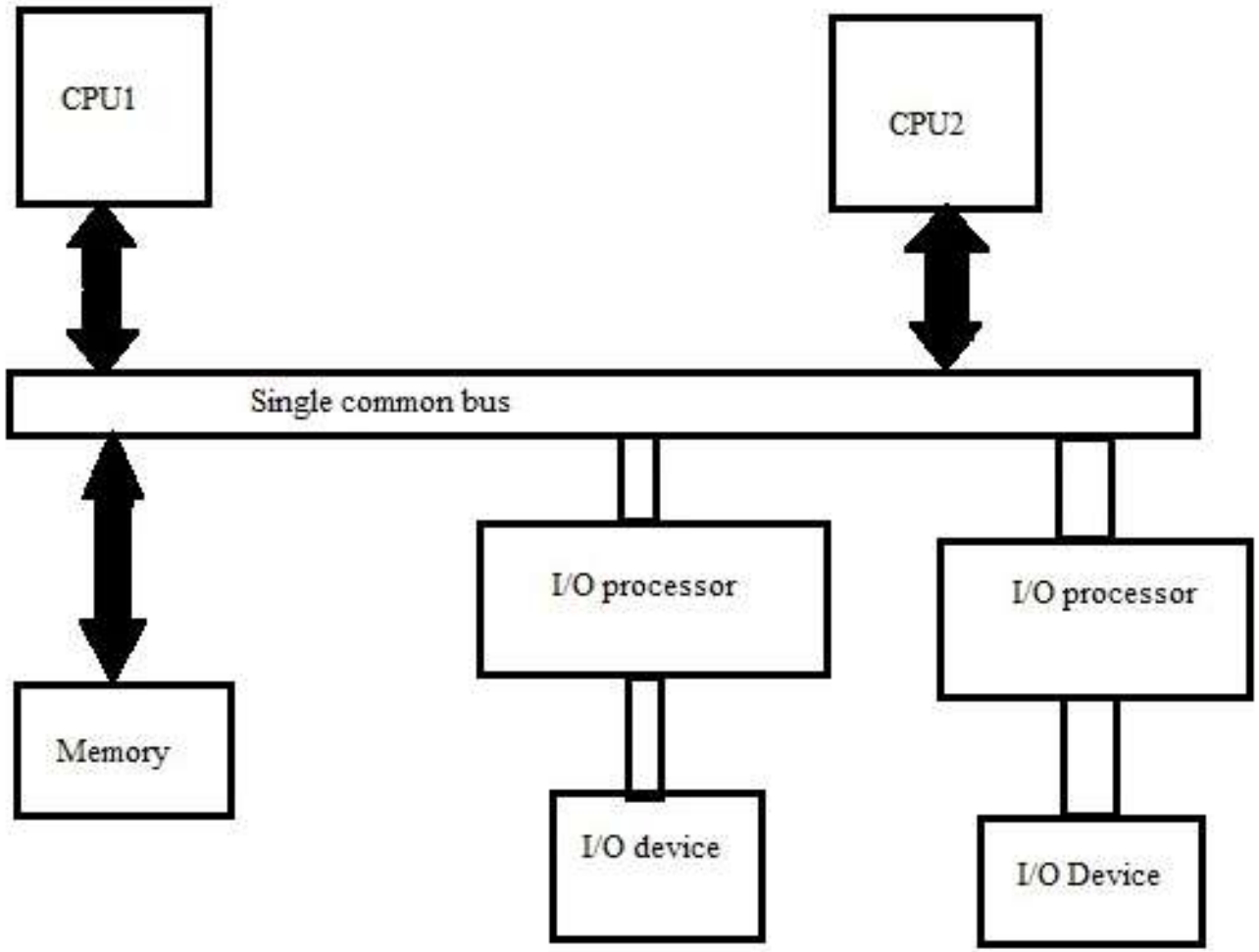
Bus organization

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
- Efficient scheme for transferring information in a system is to have a common bus.

- To transfer data and other information between devices
- Various devices in computer like(Memory, CPU, I/O and Other) are communicate with each other through buses
- Bus is said to be as the communication pathway connecting two or more devices
- It is a shared transmission medium, as multiple devices are attached to it
- A bus consists of multiple communication Pathways or lines which are either in the form of wires or metal lines.
- Each line is capable of transmitting binary 1 and binary 0

Single bus system



Single bus system

- In such a system one processor is allowed to communicate with the memory or another processor at any given time.

- since it is restricted to one transfer at a time,its data transfer rate within the system is limited by the speed of the single shared bus system.

- It has been used in computer system because it can easily be designed and controlled.

DUAL BUS ARCHITECTURE

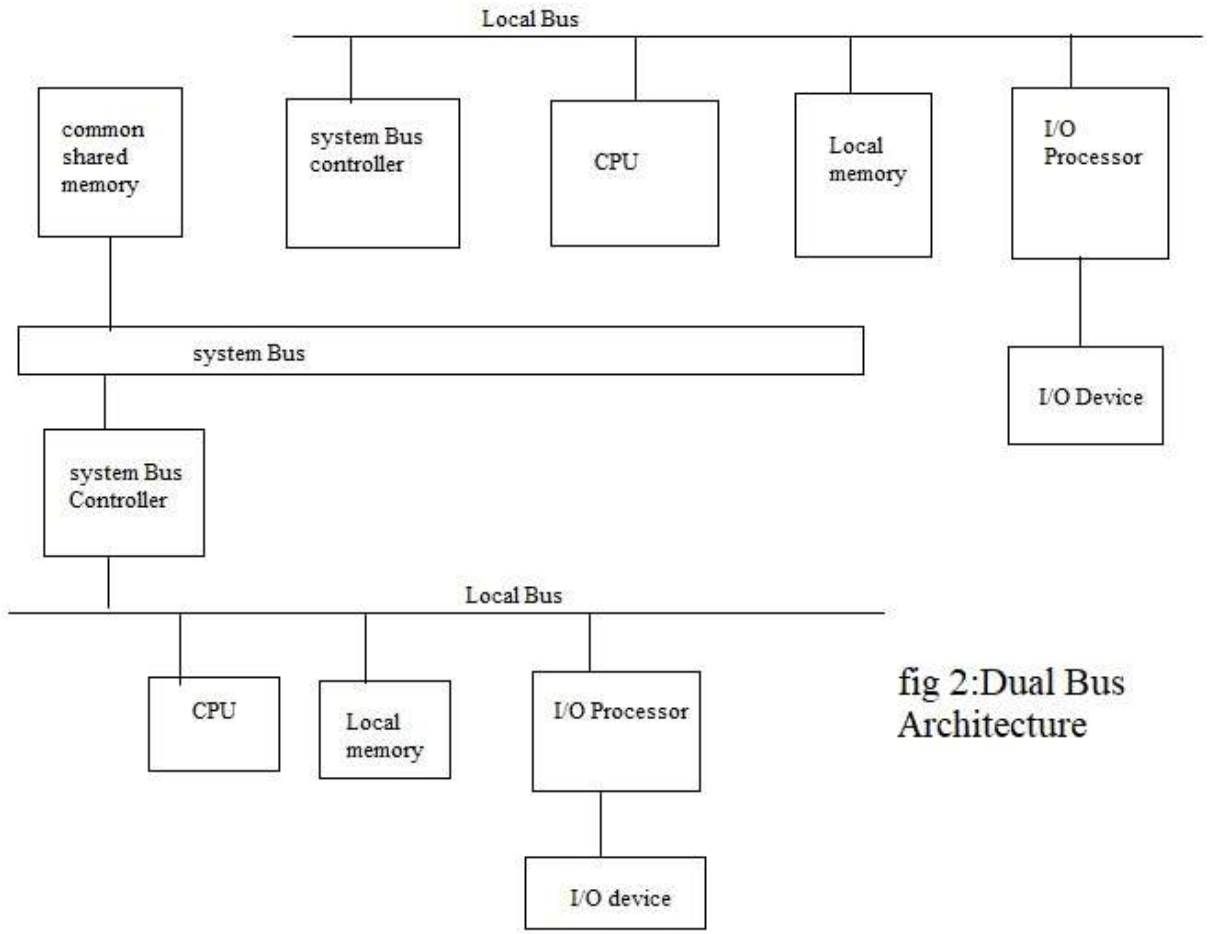
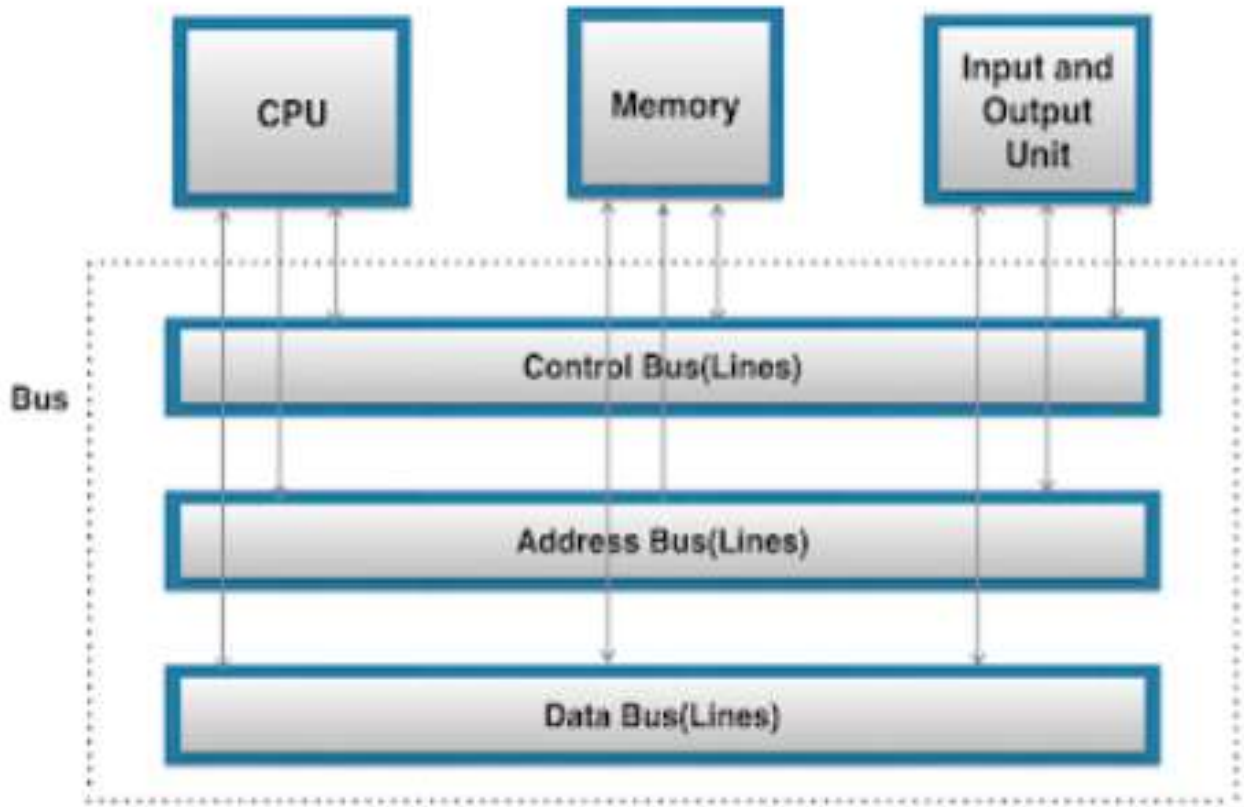


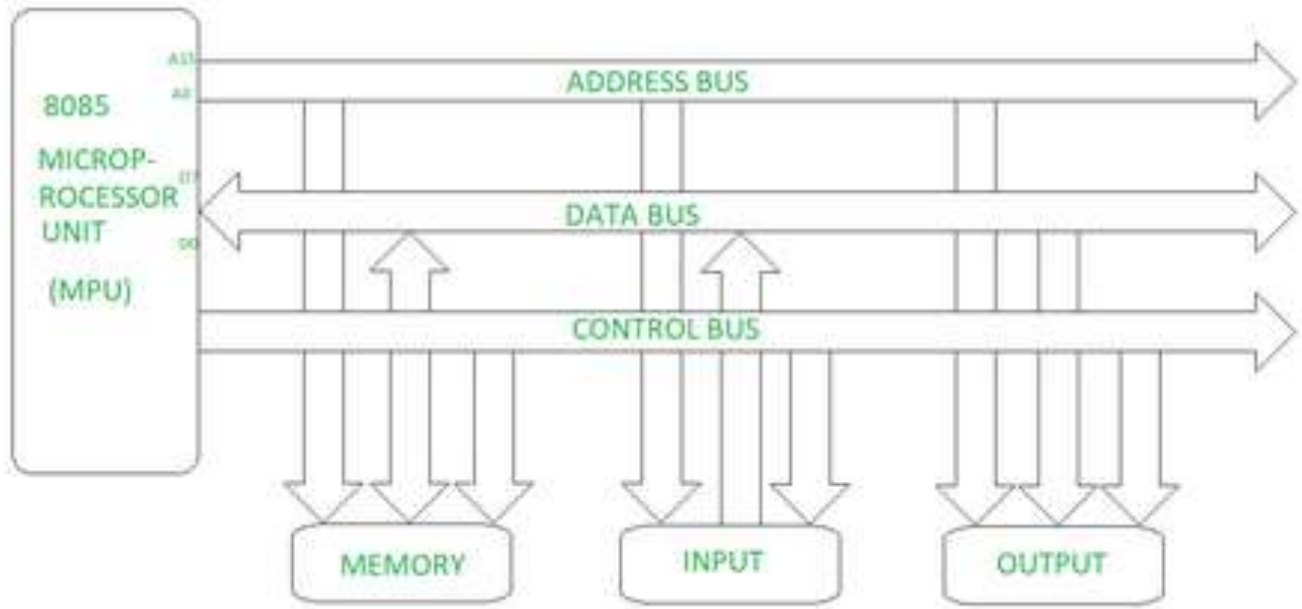
fig 2:Dual Bus Architecture

Dual bus configuration

- In this system local buses are employed.
- Each local bus is connected to a CPU,the local memory and I/O.
- A system bus controller connects each local bus to a common system bus.
- The memory and I/O connected to the common bus can be sheared by all processors.
- In this system only one CPU is allowed to communicate with the shared memory and other common resources through the system bus at any given time.
- The other processors use their local memory and I/O devices.

- It is an improvement over the single shared bus system.
- It has higher data transfer rate.
- But it is a costly and complex bus system.


CPU
Memory
Input and Output Unit
Bus
Control Bus(Lines)
Address Bus(Lines)
Data Bus(Lines)



Bus organization system of 8085 Microprocessor

Data Lines:

- Data Lines provide a path for moving data between system modules.
- It is bidirectional which means data lines are used to transfer data in both directions.
- As an example, CPU can read data on these lines drom memory as well as send data out of these lines to a memory location or to a port.
- And in any bus the no. of lines in data lines are either 8,16,32 or more depending on size of bus.
- These lines , collectively are called as data bus.

Address Lines:

- Address Lines are collectively called as address bus.
- In any bus, the no. of lines in address are usually 16,20,24, or more depending on type and architecture of bus.
- On these lines, CPU sends out the address of memory location on I/O Port that is to be written on or read from.
- In short, it is an internal channel from CPU to Memory across which the address of data(not data) are transmitted.
- Here the communication is one way that is, the address is send from CPU to Memory and I/O Port but not Memory and I/O port send address to CPU on that line and hence these lines are unidirectional.

Control Lines:

- Control Lines are collectively called as Control Bus.
- Control Lines are gateway used to transmit and receives control signals between the microprocessor and various devices attached to it.
- Control Lines are used by CPUs for communicating with other devices within the computer.
- As an example-CPU sends signals on the Control bus to enable the outputs of address memory devices and port devices.
- Typical Control Lines signals are:

  -Memory Read
  -Memory Write
  -I/O Read
  -I/O Write
  -Bus Request
  -Bus Grant, etc.

Operation of Bus

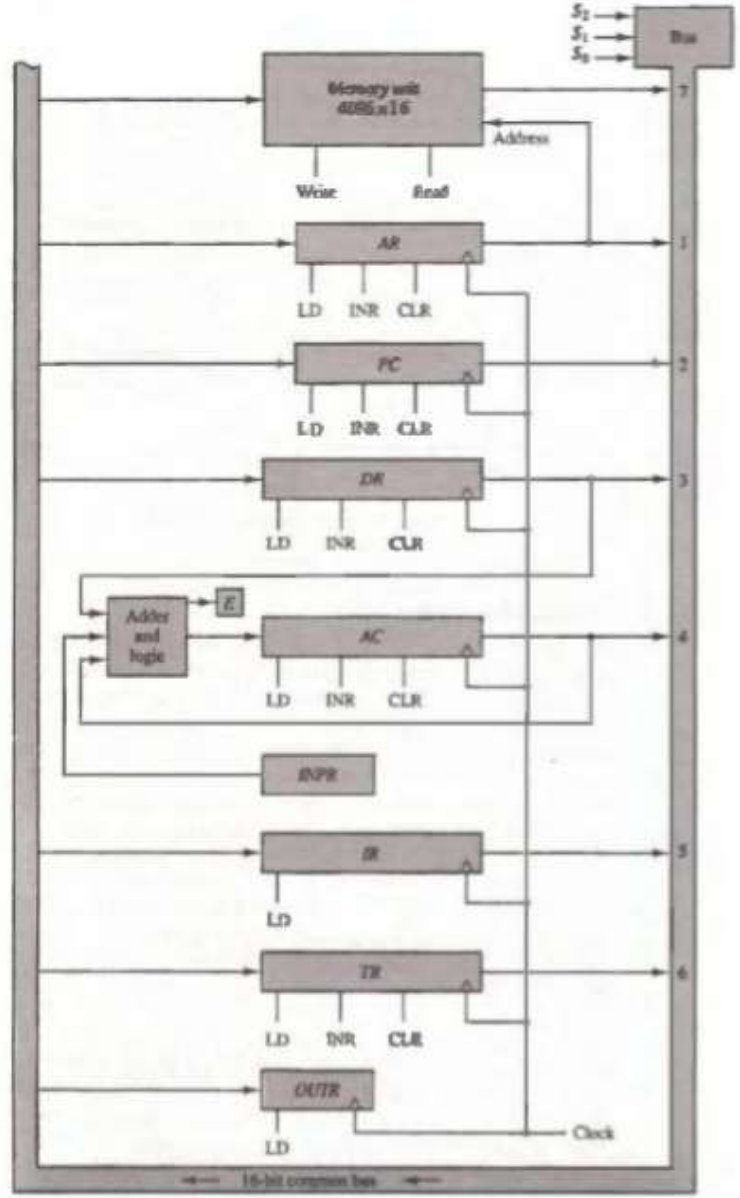
If One module wishes to send data to another, it must do two things:

1. Obtain the use of module Bus
2. Transfer for data to the Bus

If one module wishes to request data from another module, it must:

1. Obtain the use of Bus
2. Transfer a request to other module over the appropriate control and address lines.

It must then wait for that second module to send the data.


Bus
Memory unit
4096 x 16
Address
Write
Read
AR
LD INR CLR
PC
LD INR CLR
DR
LD INR CLR
Adder and logic
E
AC
LD INR CLR
INPR
IR
LD
TR
LD INR CLR
OUTR
LD
Clock
16-bit common bus

- The outputs of seven registers and memory are connected to the common bus.
- The lines from the common bus are connected to the inputs of each register and the data inputs of the memory

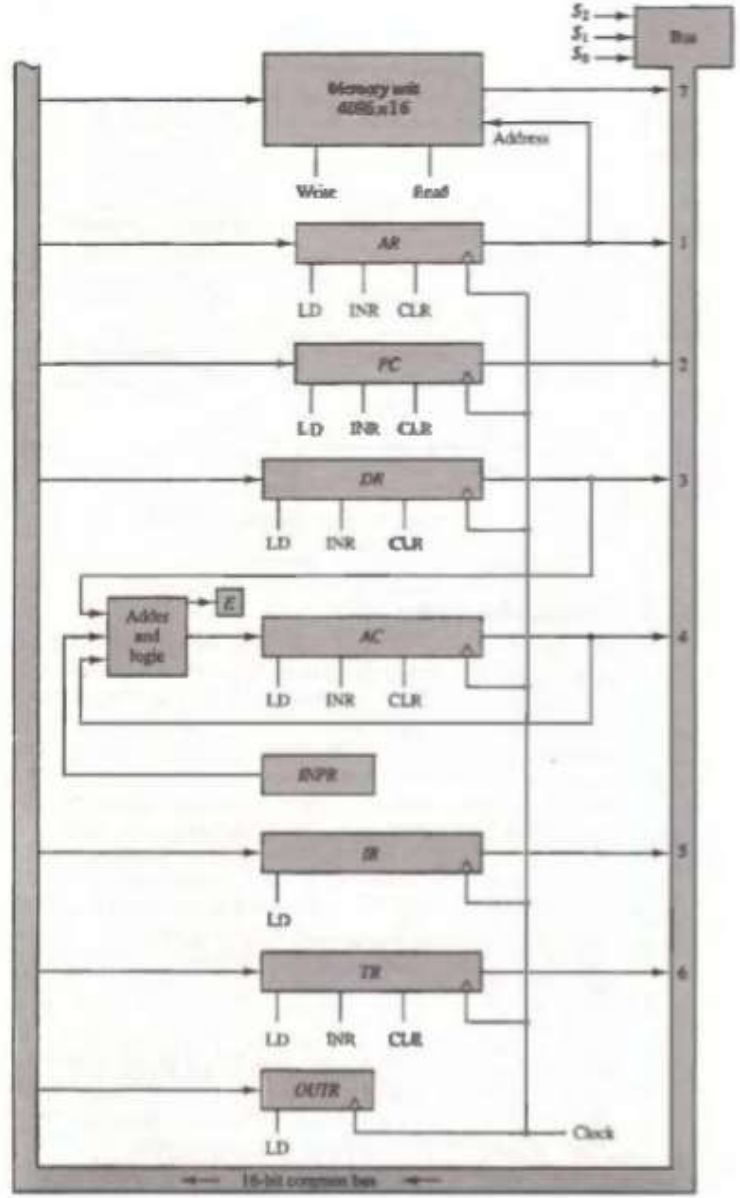
- The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S_2, S_1, and S_0

- The 16-bit outputs of DR are placed on the bus lines when $S_2\ S_1\ S_0 = 011$

- 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 S2S1S0 = 111


Bus
Memory unit
4096 x 16
Address
Write
Read
AR
LD INR CLR
PC
LD INR CLR
DR
LD INR CLR
Adder and logic
E
AC
LD INR CLR
INPR
IR
LD
TR
LD INR CLR
OUTR
LD
Clock
16-bit common bus

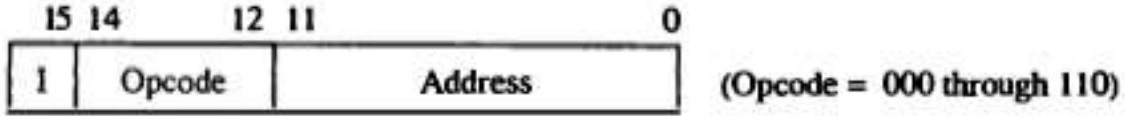
- INPR is connected to provide information to the bus but OUTR can only receive information from the bus
- INPR receives a character from an input device which is then transferred to AC
- OUTR receives a character from AC and delivers it to an output device.
- There is no transfer from OUTR to any of 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
- The memory address is connected to AR.
- 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
- The 16 inputs of AC come from an adder and logic circuit. This circuit has three sets of inputs
- One set of 16-bit inputs come from the outputs of AC. They are used to implement register microperations such as complement AC and shift AC
- Another set of 16-bit inputs come from the data register DR. The inputs from DR andAC are used for arithmetic and logic microperations, such as add DR to AC or AND DR to AC
- A third set of 8-bit inputs come from the input register INPR

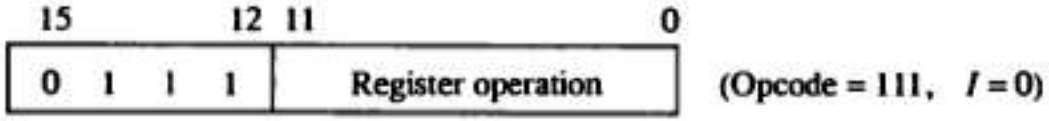
COMPUTER INSTRUCTIONS

- The basic computer has three instruction code formats
  - Memory-reference instruction
  - Register-reference instruction
  - Input-output instruction

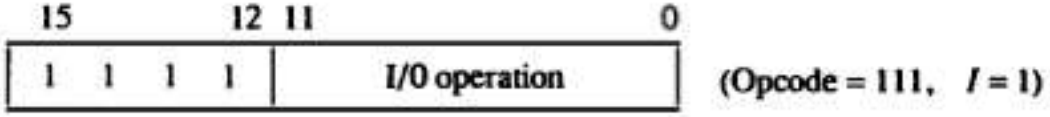
Figure 5-5 Basic computer instruction formats.



(a) Memory – reference instruction



(b) Register – reference instruction



(c) Input – output instruction

- A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I.

- I is equal to 0 for direct address and to 1 for indirect address

- The register-reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction.

- A register-reference instruction specifies an operation on AC register.

- An operand from memory is not needed; therefore, the other 12 bits are used to specify the operation

- An input-output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction.

- The remaining 12 bits are used to specify the type of input-output operation performed

- The type of instruction is recognized by the computer control from the four bits in positions 12 through 15 of the instruction.

- If the three opcode bits in positions 12 though 14 are not equal to 111, the instruction is a memory-reference type and the bit in position 15 is taken as the addressing mode I

- If the 3-bit opcode is equal to Ill, control then inspects the bit in position 15.

- If this bit is 0, the instruction is a register-reference type

- If the bit is 1, the instruction is an input-output type

- Note: the bit in position 15 of the instruction code is designated by the symbol I but is not used as a mode bit when the operation code is equal to 111

TABLE 5-2 Basic Computer Instructions

Symbol	Hexadecimal code $I = 0$	Hexadecimal code $I = 1$	Description
AND	0xxx	8xxx	AND memory word to *AC*
ADD	1xxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
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 instruction if *AC* positive
SNA	7008		Skip next instruction if *AC* negative
SZA	7004		Skip next instruction if *AC* zero
SZE	7002		Skip next instruction if *E* is 0
HLT	7001		Halt computer
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

- The hexadecimal code is equal to the equivalent hexadecimal number of the binary code used for the instruction.

- By using the hexadecimal equivalent we reduced the 16 bits of an instruction code to four digits with each hexadecimal digit being equivalent to 4 bit

- A memory-reference instruction has an address part of 12 bits. The address part is denoted by three x's and stand for the three hexadecimal digits corresponding to the 12-bit address

- The last bit of the instruction is designated by the symbol I. When I = 0, the last four bits of an instruction have a hexadecimal digit equivalent from 0 to 6 since the last bit is 0.

- When I = I, the hexadecimal digit equivalent of the last four bits of the instruction ranges from 8 to E since the last bit is I.

- Register-reference instructions use 16 bits to specify an operation. The leftmost four bits are always 0111, which is equivalent to hexadecimal 7.

- The other three hexadecimal digits give the binary equivalent of the remaining 12 bits.

- The input-output instructions also use all 16 bits to specify an operation. The last four bits are always 1111, equivalent to hexadecimal F.

Instruction Set Completeness

- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

1. Arithmetic, logical, and shift instructions - provide computational capabilities for processing the type of data that the user may wish to employ

2. Instructions for moving information to and from memory and processor registers - The bulk of the binary information in a digital computer is stored in memory, but all computations are done in processor registers

3. Program control instructions together with instructions that check status conditions – Program control instructions such as branch instructions are used to change the sequence in which the program is executed

4. Input and output instructions - needed for communication between the computer and the user. Programs and data must be transferred into memory and results of computations must be transferred back to the user

- One arithmetic instruction, ADD, and two related instructions, complement AC(CMA) and increment AC(INC).
- With these three instructions we can add and subtract binary numbers when negative numbers are in signed-2's complement representation

- The circulate instructions, CIR and CIL, can be used for arithmetic shifts as well as any other type of shifts desired

- Multiplication and division can be performed using addition, subtraction, and shifting

- Logic operation – AND : complement provide a NAND operation. It can be shown that with the NAND operation it is possible to implement all the other logic operations with two variables

- Moving information from memory to AC is accomplished with the load AC(LDA) instruction. Storing information from AC into memory is done with the store AC(STA) instruction

- The branch instructions BUN, BSA, and ISZ, together with the four skip instructions, provide capabilities for program control and checking of status conditions

- The input (lNP) and output (OUT) instructions cause information to be transferred between the computer and external devices.

- An efficient set of instructions will include such instructions as subtract, multiply, OR, and exclusive-OR

Timing and Control

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.

- Two major types of control organization
  - hardwired control
  - Micro-programmed control

- In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits.

- It has the advantage that it can be optimized to produce a fast mode of operation.

- A hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed.

- In the micro-programmed organization, the control information is stored in a control memory.

- The control memory is programmed to initiate the required sequence of micro-operations.

- In the micro-programmed control, any required changes or modifications can be done by updating the microprogram in control memory.

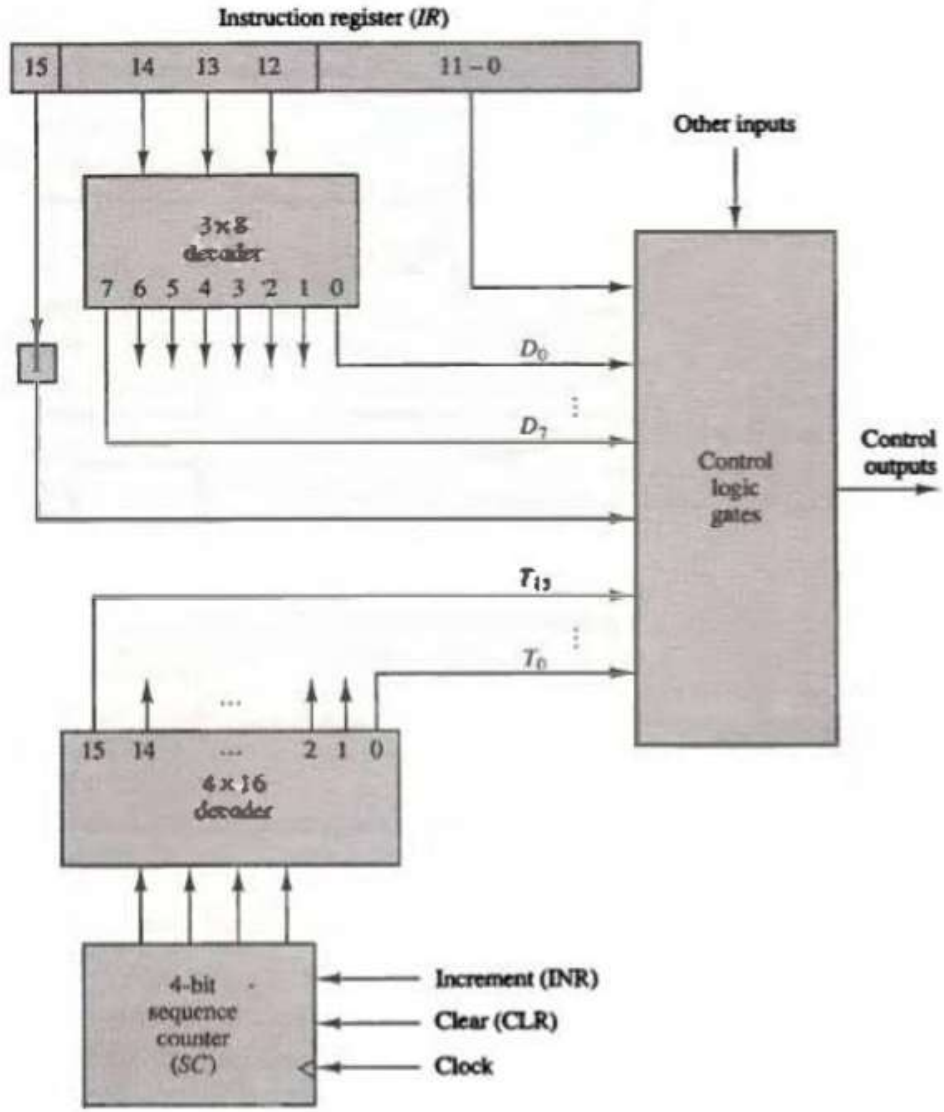


Figure 5-6 Control unit of basic computer.

- It consists of two decoders, a sequence counter, and a number of control logic gates.
- An instruction read from memory is placed in the instruction register (IR).
- The instruction register is divided into three parts: the I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are decoded with a 3 x 8 decoder.
- The eight outputs of the decoder are designated by the symbols D0 through D7
- The subscripted decimal number is equivalent to the binary value of the corresponding operation code.
- Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I.
- Bits 0 through 11 are applied to the control logic gates.
- The 4-bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T0 through T15

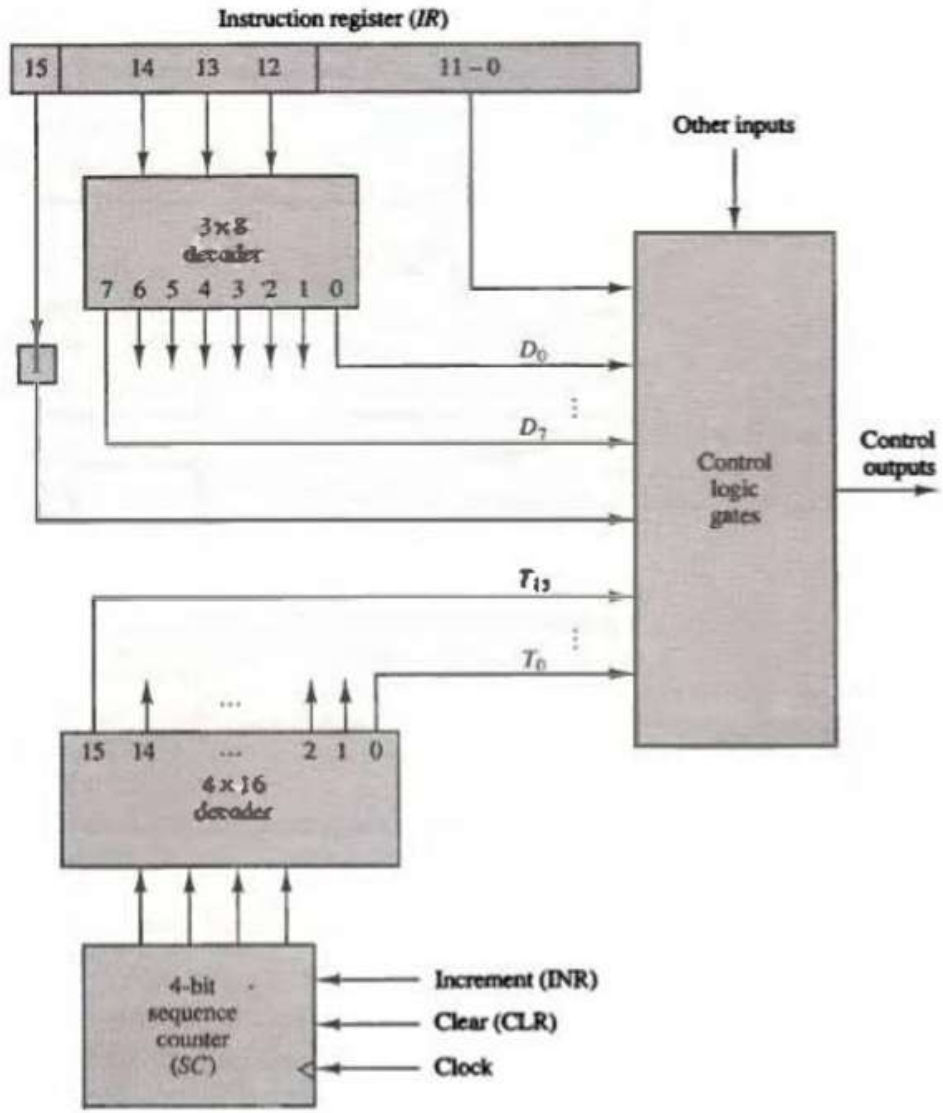


Figure 5-6 Control unit of basic computer.

- The sequence counter SC can be incremented or cleared synchronously.
- Most of the time, the counter is incremented to provide the sequence of timing signals out of the 4 x 16 decoder.
- Once in a while, the counter is cleared to 0, causing the next active timing signal to be To.
- As an example, consider the case where SC is incremented to provide timing signals T0, T1 T2, T3, and T4 in sequence.
- At time T4, SC is cleared to 0 if decoder output D3 is active. This is expressed symbolically by the statement

$$D_3T_4: SC <- 0$$

The time relationship of the control signals

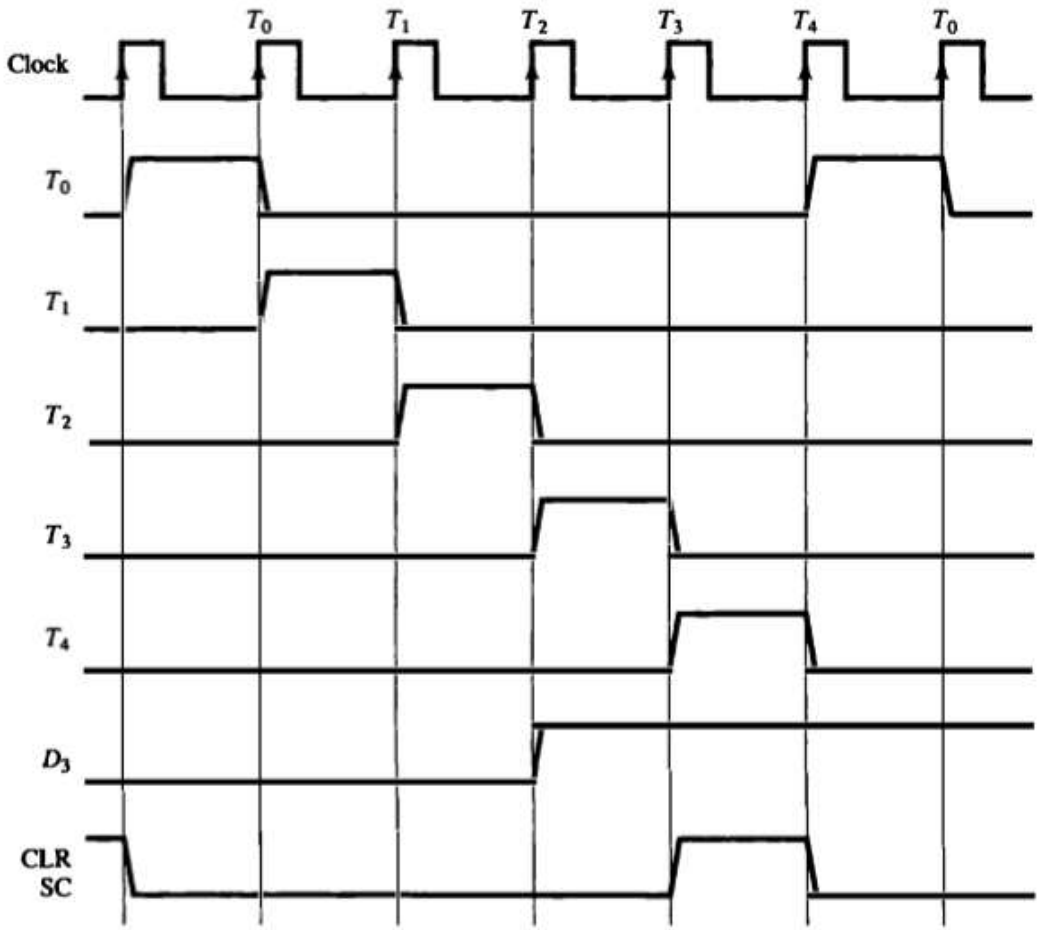


Figure 5-7 Example of control timing signals.

- The sequence counter SC responds to the positive transition of the clock.
- Initially, the CLR input of SC is active

- The first positive transition of the clock clears SC to 0, which in turn activates the timing signal T0 out of the decoder

- T0 is active during one clock cycle

- The positive clock transition labeled T0 in the diagram will trigger only those registers whose control inputs are connected to timing signal To.

- SC is incremented with every positive clock transition, unless its CLlR input is active.

- This produces the sequence of timing signals To. T1, T2, T3, T4, and so on

- If SC is not cleared, the timing signals will continue with T5,T6, up to T15 and back to T0 .

The time relationship of the control signals

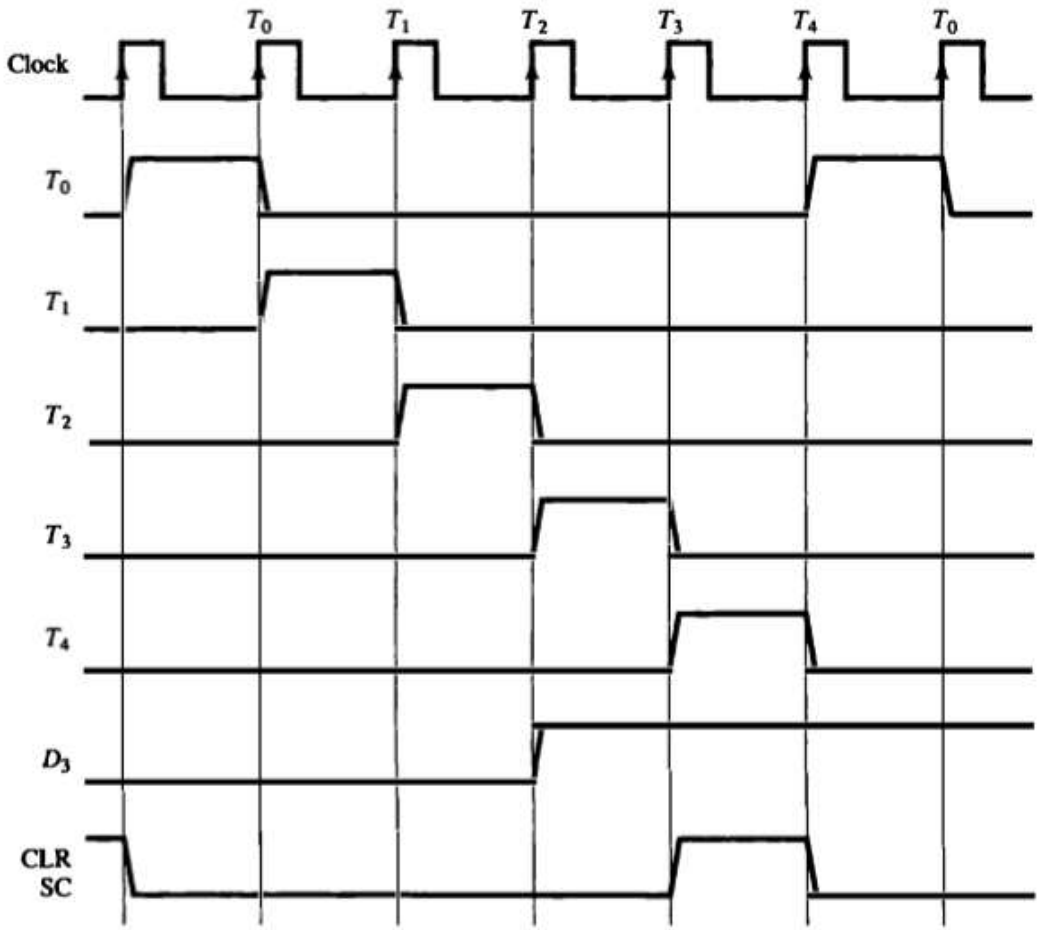


Figure 5-7 Example of control timing signals.

- The last three waveforms show how SC is cleared when D3T4 = I
- Output D3 from the operation decoder becomes active at the end of timing signal T2
- When timing signal T4 becomes active, the output of the AND gate that implements the control function D3T4 becomes active.
- This signal is applied to the CLR input of SC.
- On the next positive clock transition (the one marked T4,) the counter is cleared to 0.
- This causes the timing signal To to become active instead of T5 that would have been active if SC were incremented instead of cleared.

- A memory read or write cycle will be initiated with the rising edge of a timing signal.
- It will be assumed that a memory cycle time is less than the clock cycle time.
- According to this assumption, a memory read or write cycle initiated by a timing signal will be completed by the time the next clock goes through its positive transition.
- The clock transition will then be used to load the memory word into a register.
- This timing relationship is not valid in many computers because the memory cycle time is usually longer than the processor clock cycle.
- In such a case it is necessary to provide wait cycles in the processor until the memory word is available.

- Eg: the register transfer statement

  T0: AR <- PC

- specifies a transfer of the content of PC into AR, if timing signal To is active.

- During this time the content of PC is placed onto the bus (with $S_2S_1S_0$ = 010) and the LD (load) input of AR is enabled.

- The actual transfer does not occur until the end of the clock cycle when the clock goes through a positive transition.

- This same positive clock transition increments the sequence counter SC from 0000 to 0001.

- The next clock cycle has T1 active and T0 inactive.

INSTRUCTION CYCLE

Instruction Cycle

- A program residing in the memory unit of the computer consists of a sequence of instructions

- Each instruction cycle in turn is subdivided into a sequence of subcycles or 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

- This process continues indefinitely unless a HALT instruction is encountered.

Fetch and Decode

- Initially, the program counter PC is loaded with the address of the first instruction in the program.
- The sequence counter SC is cleared to 0, providing a decoded timing signal To.
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T0, T1, T2, and so on.

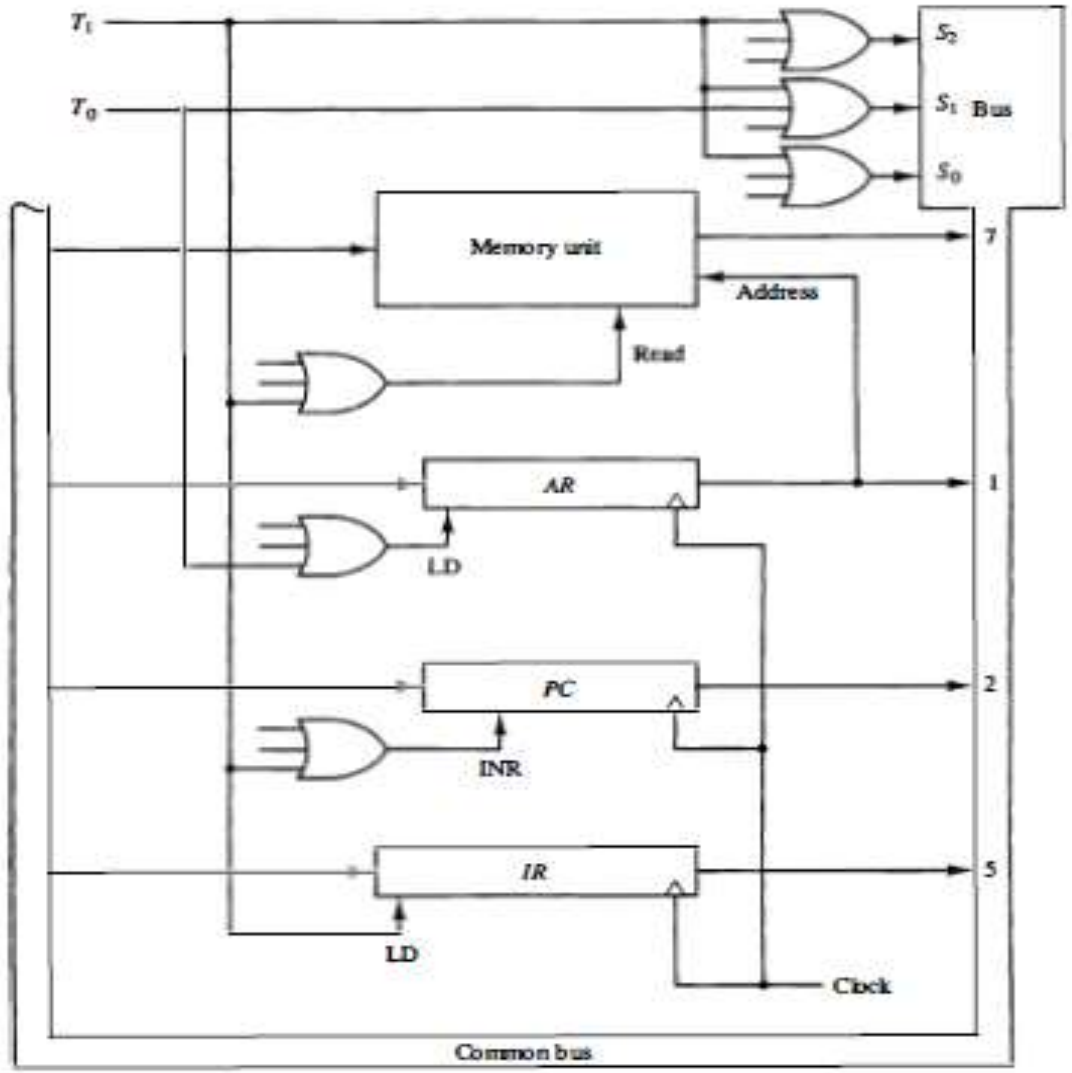


Figure 5-8 Register transfers for the fetch phase.

- The micro-operations for the fetch and decode phases

  T0: AR <- PC

  T1: IR<-M[AR], PC <- PC + 1

  T2: D0, ... , D7 <- Decode IR(12-14), AR <- IR(0-11), I<- IR(l5)

- T0: Only AR is connected to the address inputs of memory.
- it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T0

- T1: The instruction read from memory is then placed in the instruction register IR
- PC is incremented by one

- T2: the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR .

- SC is incremented after each clock pulse to produce the sequence To, T1, and T2.

To provide the data path for the transfer of PC to AR we must apply timing signal T0 to achieve the following connection:

- 1. Place the content of PC onto the bus by making the bus selection inputs S2S1S0 equal to 010.

  2. Transfer the content of the bus to AR by enabling the LD input of AR .

The next clock transition initiates the transfer from PC to AR since T0 = 1.

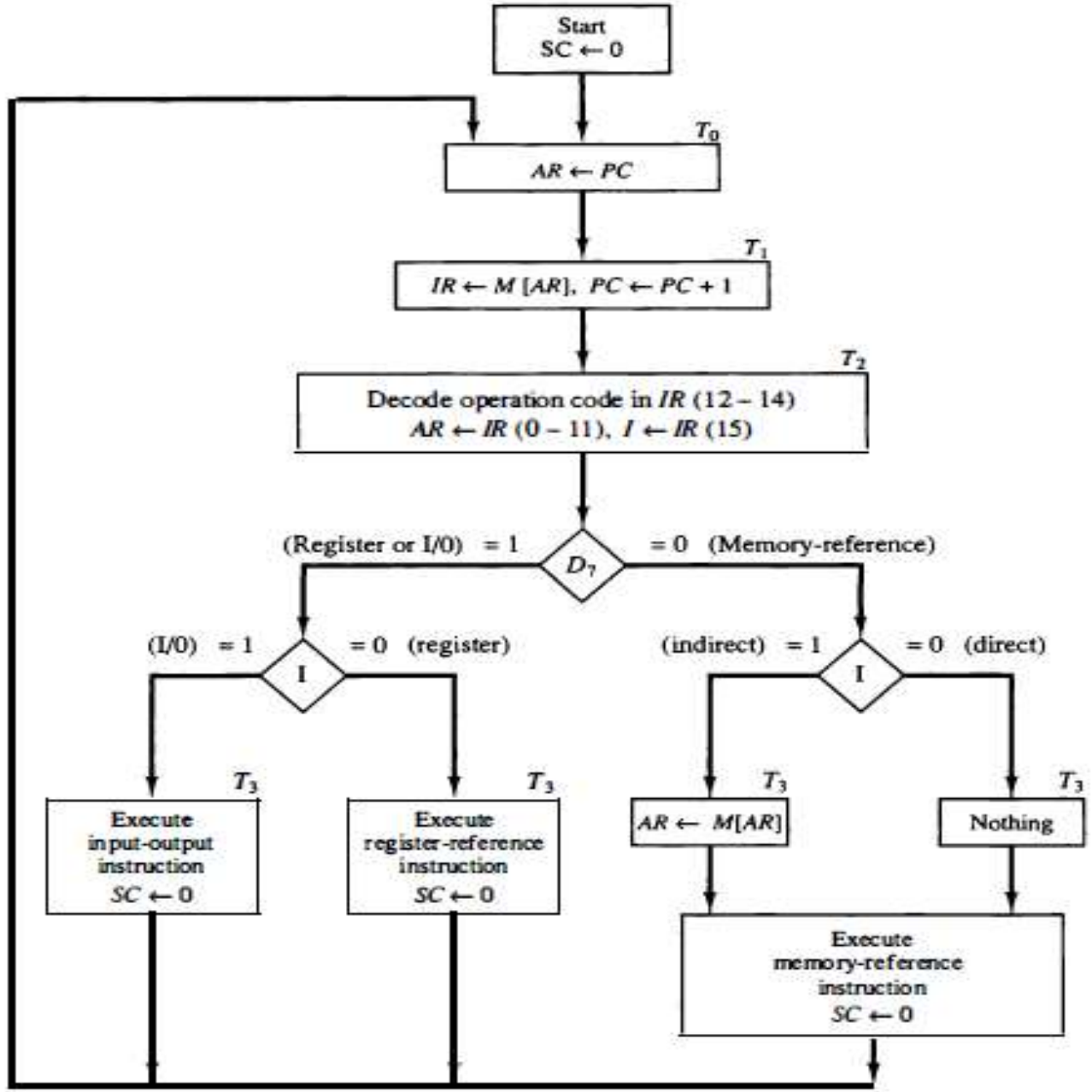


Figure 5-9 Flowchart for instruction cycle (initial configuration).

TABLE 5-3 Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0\text{–}11)$ that specifies the operation]

	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	Increment AC
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	Skip if positive
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	Skip if negative
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1$	Skip if AC zero
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start–stop flip-flop)	Halt computer