UNIT 1 BASIC COMPUTER ORANIZATION AND DESIGN

Computer organization is concerned with the structure and behaviour of the as seen by the user. It deals with components and connections in the system. Computer Organization tells us how exactly all the units in the system are arranged and interconnected.

Computer Architecture is concerned with the way hardware components are connected together to form a computer system. It acts as an interface between hardware and software. It help us the functionalities of the system.

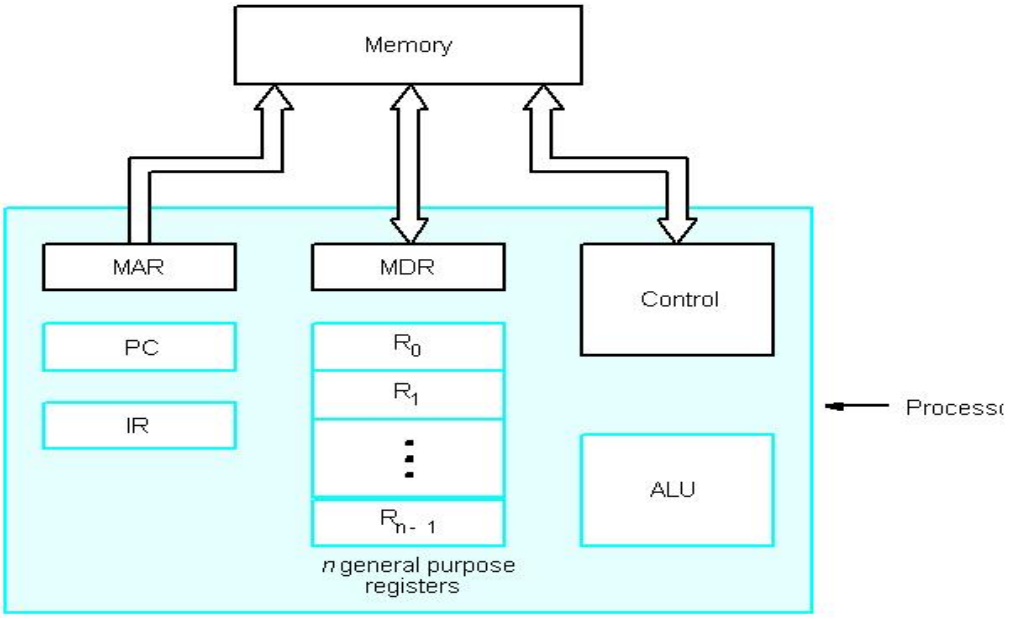
Basic operational Concepts

To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory.

Examples: - Add LOCA, R0

This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register. This instruction requires the performance of several steps,

1. First the instruction is fetched from the memory into the processor.
2. The operand at LOCA is fetched and added to the contents of R0
3. Finally the resulting sum is stored in the register R0



The above figure shows how te memory and processor can be connected. I addition to ALU and control unit the processor contains the no. of registers used for several purposes.

The PC (**Program Counter)** contains the memory address of the instruction to be executed. During execution, the contents of the PC are updated to point to the next instruction. Every time that an instruction is to be executed, the program counter releases its contents to the internal bus and sends it to the memory address register.

The registers MAR and MDR facilitates communication with the memory. The MAR (**Memory Address Register)** holds the address of the location to or from which data are to be transferred. As can be seen from the figure above, the connection of the MAR to the main memory is one-way or unidirectional.

The MDR (**Memory Data Register)** contains the data to be written or read out of the addressed location. During the fetch operation, the MDR contains the instruction to be executed or data needed during execution. In write operation, MDR the data to be written into the main memory.

The IR (**Instruction Register)** contains the instruction that is being executed. Before the IR executes the instruction it needs to be decoded first. As soon as the content of the MDR is transferred to the IR, the decoding process commences. After decoding, execution of the instruction will take place.

The n- general purpose registers **R_0 to R n_{-1}** is used for storing temporary values

Let us consider some operating steps.

1. Programs reside in the memory & usually get these through the I/P unit.

2. Execution of the program starts when the PC is set to point at the first instruction of the program. 3. Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.

4. After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.

5. Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.

6. If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.

7. An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.

8. When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.

9. After one or two such repeated cycles, the ALU can perform the desired operation.

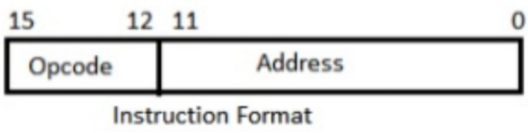
10. If the result of this operation is to be stored in the memory, the result is sent to MDR.

11. Address of location where the result is stored is sent to MAR & a write cycle is initiated.

12. The contents of PC are incremented so that PC points to the next instruction that is to be executed.

Instruction Codes

An instruction code is a group of bits that instruct the computer to perform a specific operation. A computer instruction is a binary code that determines the micro -operations in a sequence for a computer. They are saved in the memory along with the information. Each computer has its specific group of instructions. It is usually divided in to two parts:



Instruction Format

It consists of 12 bits of memory that are required to define the address as the

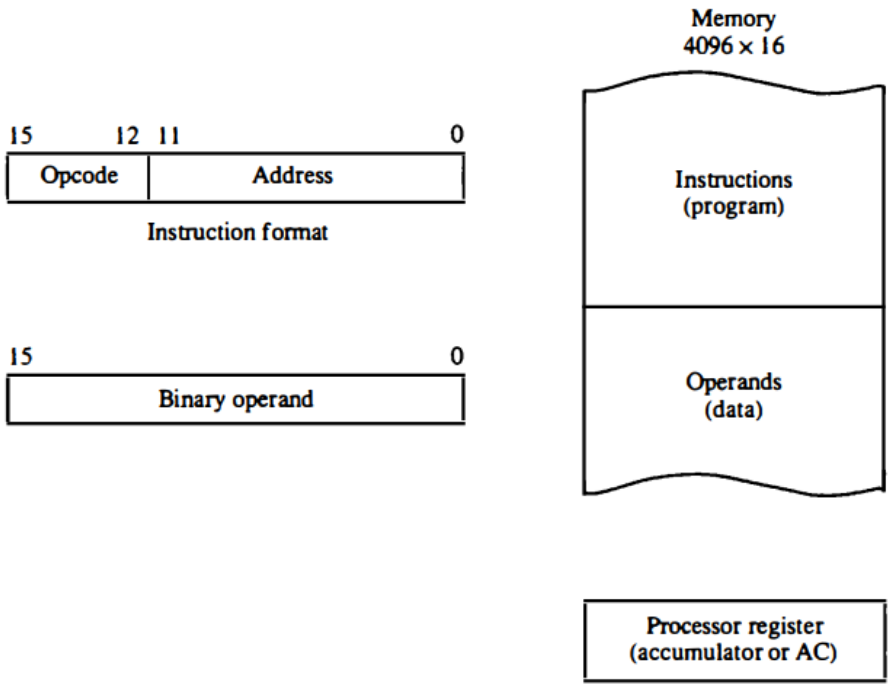
memory includes 4096 words. The 15th bit of the instruction determines the addressing mode (where direct addressing corresponds to 0, indirect addressing corresponds to 1). Therefore, the instruction format includes 12 bits of address and 1 bit for the addressing mode, 3 bits are left for Opcodes.

The number of bits required for the operation code depends upon the total number of operations available on the computer. The operation code must consist of at least **n bits** for a given **2^n** operations. The operation part of an instruction code specifies the operation to be performed.

Stored Program Organisation

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.

Figure Stored program organization.



The above 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.

Computers with a single processor register is known as **Accumulator (AC)**. The operation is performed with the memory operand and the content of AC.

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. One bit of the instruction code can be used to distinguish between a direct and an indirect address.

As an illustration of this configuration, consider the instruction code format shown in Fig. below part (a). It consists of a 3-bit operation code, a 12-bit address, and an indirect address mode bit designated by I. The mode bit is 0 for a direct address and 1 for an indirect address. A direct address instruction is shown in Fig. below part (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 memory at address 457 and adds it to the content of AC. The instruction in address 35 shown in Fig. below part (c) has a mode bit I = 1. Therefore, it is recognized as an indirect address instruction. The address part is the binary equivalent of 300. The control goes to address 300 to find the address of the operand. The address of the 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; the second is for the operand itself. We define the effective address to be the

address of the operand in a computation-type instruction or the target address in a branch-type instruction. Thus the effective address in the instruction of Fig. below part (b) is 457 and in the instruction of Fig below part (c) is 1350. The memory word that holds the address of the operand in an indirect address instruction is used as a pointer to an array of data. The pointer could be placed in a processor register instead of memory as done in commercial computers.

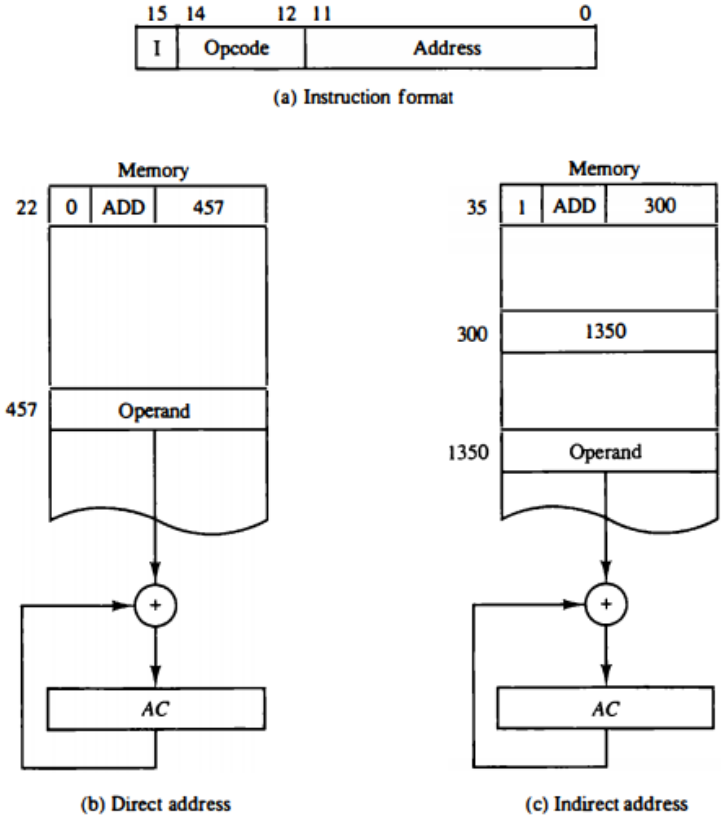


Figure Demonstration of direct and indirect address.

Computer Registers

The registers are also listed in Table below together with a brief description of their function and the number of bits that they contain. The memory unit has a capacity of 4096 words and each word contains 16 bits. Twelve bits of an instruction word are needed to specify the address of an operand. This leaves three bits for the operation part of the instruction and a bit to specify a direct or indirect address. The data register (DR) holds the operand read from memory.

The accumulator (AC) register is a general purpose processing register. The instruction read from memory is placed in the instruction register (IR). The

temporary register (TR) is used for holding temporary data during the processing.

The memory address register (AR) has 12 bits since this is the width of a memory address. The program counter (PC) also has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed.

The PC goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory. Instruction words are read and executed in sequence unless a branch instruction is encountered. A branch instruction calls for a transfer to a non-consecutive instruction in the program. The address part of a branch instruction is transferred to PC to become the address of the next instruction. To read an instruction, the content of PC is taken as the address for memory and a memory read cycle is initiated. PC is then incremented by one, so it holds the address of the next instruction in sequence.

Two registers are used for input and output. The input register (INPR) receives an 8-bit character from an input device. The output register (OUTR) holds an 8-bit character for an output device.

TABLE 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

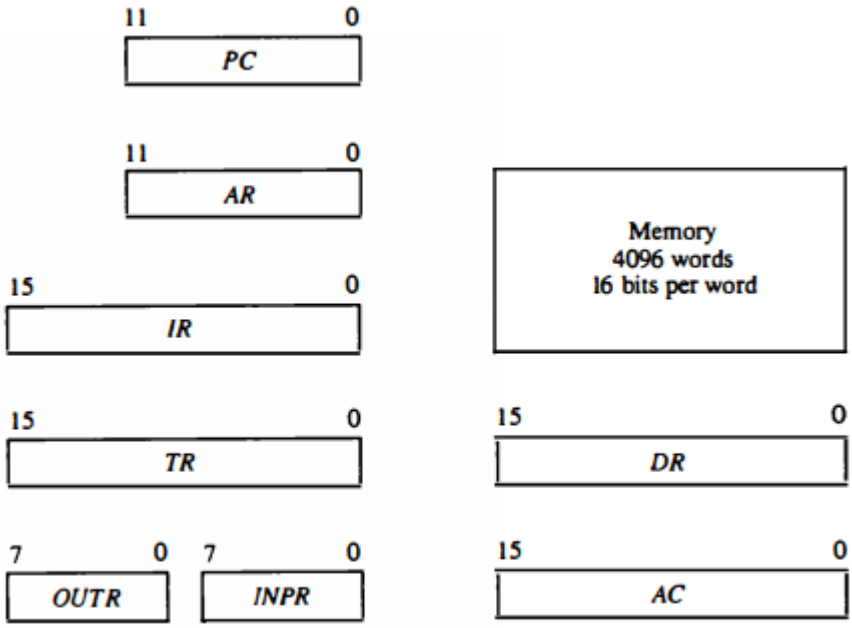


Figure Basic computer registers and memory.

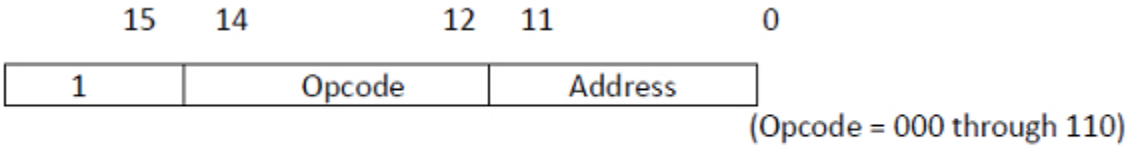
Computer Instructions

A basic computer has three instruction code formats which are:

1. Memory - reference instruction
2. Register - reference instruction
3. Input-Output instruction

Memory - reference instruction

A memory-reference instruction uses 12 bits to specify an address and one bit to determine the addressing mode I. I is the same as 0 for direct address and to 1 for indirect address.

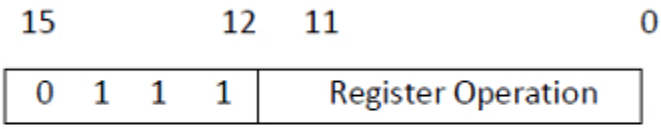


(a) Memory Reference Instruction

Register reference instructions

The register reference instructions are identified by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction. It determines an operation on or a test of the AC register. An operand from memory is not required because the additional

12 bits are used to determine the operation or test to be implemented.

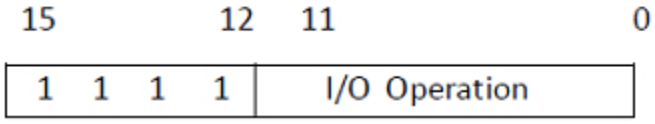


(Opcode = 111, I = 0)

(b) Register Reference Instruction

Input-output instruction

An input-output instruction does not require a reference to memory and is identified by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits can determine the type of input-output operation or test implemented.



(Opcode = 111, I = 1)

(c) Input-Output Instruction

The type of instruction is identified by the computer control from the four bits in positions 12 through 15 of the instruction. If the three opcode bits in positions 12 through 14 are not similar 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 similar to 111, the control then examines 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.

Instruction set completeness

A set of instructions is said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

- Arithmetic, logical and shift instructions
- A set of instructions for moving information to and from memory and processor registers.
- Instructions which controls the program together with instructions that check status conditions.
- Input and Output instructions

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

A huge amount of binary information is stored in the memory unit, but all computations are done in processor registers. Therefore, one must possess the capability of moving information between these two units.

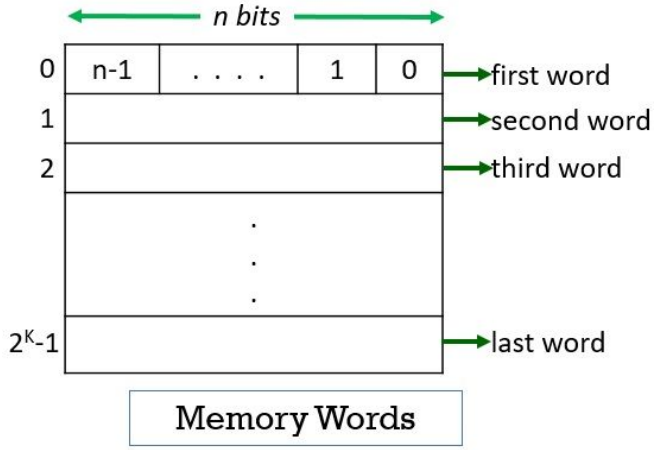
Program control instructions such as branch instructions are used change the sequence in which the program is executed.

Input and Output instructions act as an interface between the computer and the user. Programs and data must be transferred into memory, and the results of computations must be transferred back to the user.

Memory locations and addresses

Memory consists of many millions of storage cells (flip-flops). Each cell can store a bit of information i.e. 0 or 1 .Each group of n bits is referred to as a word of information, and n is called the word length. The word length can vary from 8 to 64 bits. A unit of 8 bits is called a byte. Accessing the memory to store or retrieve a single item of information (word/byte) requires distinct addresses for each item location. (It is customary to use numbers from 0 through 2ˆk-1 as the addresses of successive-locations in the memory).If 2ˆk = no. of addressable locations; then 2ˆk addresses constitute the address-space of the computer. For example, a 24-bit address generates an address-space of 2ˆ24 locations (16 MB).

The group of n bit is termed as **word** where n is termed as the **word length**. The word length of the computer has evolved from 8, 16, 24, 32 to 64 bits. General-purpose computers nowadays have 32 to 64 bits. The group of 8 bit is called a byte.



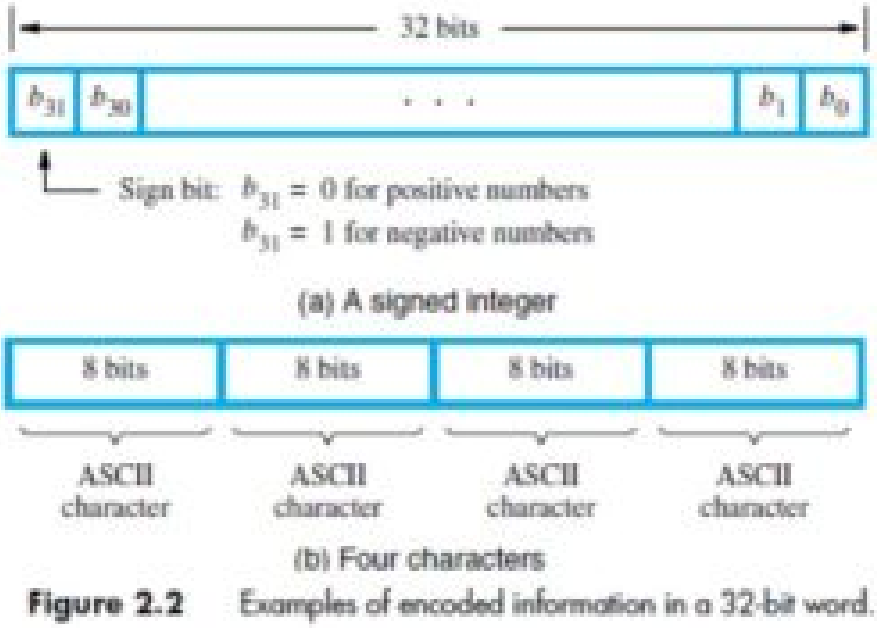


Figure 2.2 Examples of encoded information in a 32-bit word.

The figure 2.2 (a) shows a 32 pattern that can be used to represent signed numbers. The left most bit is called sign bit – 0 for positive and 1 for negative numbers. The magnitude of the number is determined from bits b_{30} to $b_{0\ by}$ the formula

Magnitude = $b_{30} \times 2^{30} + b_{29} \times 2^{29} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots b_1 x 2^1 + b_0 \times 2^0$

The above encoding format is called sign and magnitude representation.

If the word length of a computer is 32 , a single word can store a 32 bit binary number or 4 ASCII character each occupying 8 bits as shown in the figure 2.2 (b)

Instruction Cycle

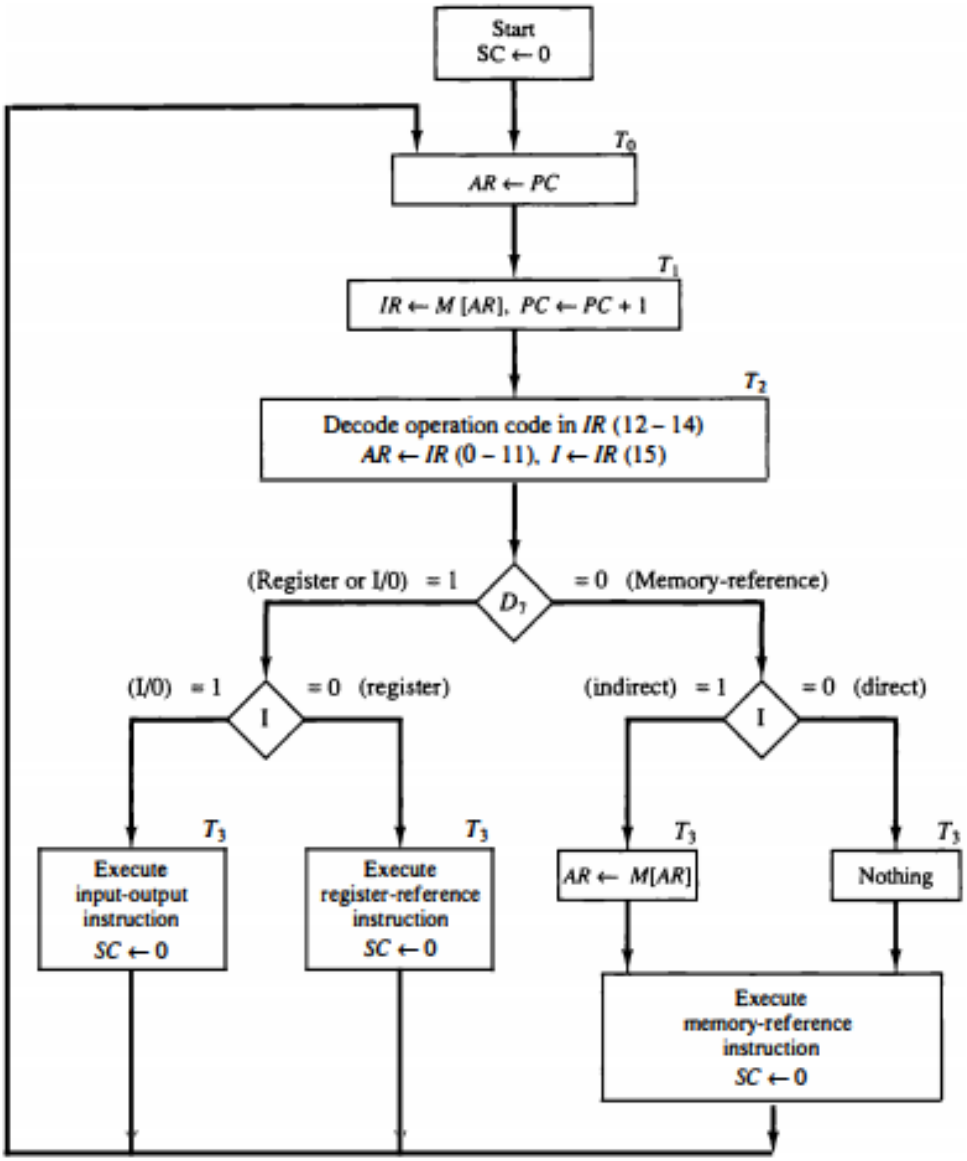


Figure Flowchart for instruction cycle (initial configuration).

A program residing in the memory unit of a computer consists of a sequence of instructions. These instructions are executed by the processor by going through a cycle for each instruction.

In a basic computer, each instruction cycle consists of the following phases:

1. Fetch instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory.
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.

The above flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction after decoding.

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 T0. After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T0, T1, T2, and so on. The rnicro-operations for the fetch and decode phases can be specified by the following register transfer statements.

1. T0: AR ← PC
2. T1: IR ← M[AR], PC ← PC + 1
3. T2: D0, ... , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)

Since 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. The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T1. At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program. At time 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. Note that SC is incremented after each clock pulse to produce the sequence T0, T1, and T2

Determine the type of instruction

The timing signal that is active after the decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory.

Decoder output D_7 is equal to 1 if the operation code is equal to binary 111. From Fig. on basic computer formats we determine that if $D_7 = 1$, the instruction must be a register-reference or input-output type. If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction. Control then inspects the value of the first bit of the instruction, which is now

available in flip-flop I. If D7 = 0 and I = 1, we have a memory reference instruction with an indirect address. It is then necessary to read the effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement: AR ← M[AR] Initially, AR holds the address part of the instruction. This address is used during the memory read operation. The word at the address given by AR is read from memory and placed on the common bus. The LD input of AR is then enabled to receive the indirect address that resided in the 12 least significant bits of the memory word. When a memory-reference instruction with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR. However, the sequence counter SC must be incremented when D'7T3 = 1, so that the execution of the memory-reference instruction can be continued with timing variable T4. A register-reference or input-output instruction can be executed with the clock associated with timing signal T3. After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with T0 = 1.Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition.

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.

There are two major types of control organization:

1. Hardwired control and
2. 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. 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. 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 control**, any required changes or modifications can be done by updating the micro-program in control memory.

The block diagram of the control unit is shown below

It consists of two decoders,

1. A sequence counter, and
2. A number of control logic gates.

An instruction read from memory is placed in the instruction register (IR).position of this register in the common bus system is indicated in Fig The instruction register is shown again in Fig, where it is divided into three parts:

1. the 1 bit,
2. the operation code, and
3. Bits 0 through 11.

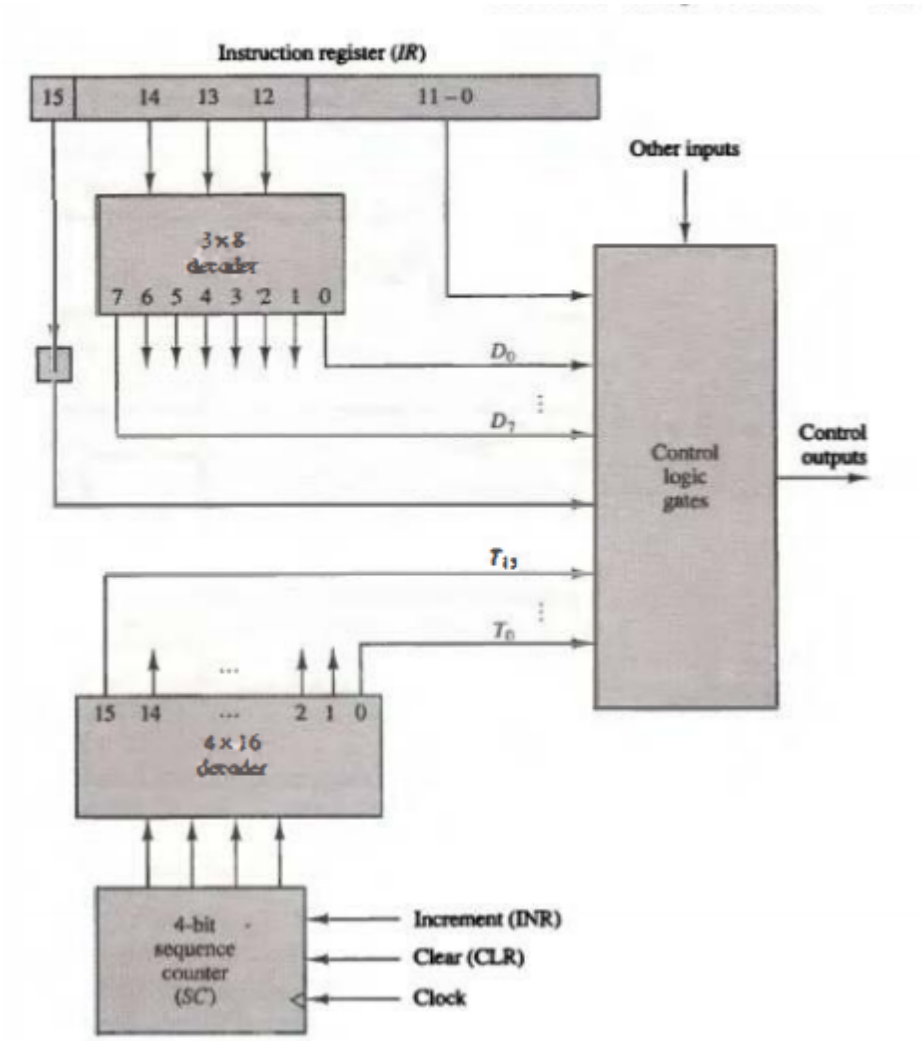


Figure 5-6 Control unit of basic computer.

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 D_0 through D_7. 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 T_0 through T_{15}.

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 T_0.

As an example, consider the case where SC is incremented to provide timing

signals T_0, T_1, T_2, T_3, and T_4 in sequence. At time T_4, SC is cleared to 0 if decoder output D_3 is active. This is expressed symbolically by the statement

$$D_3T_4: SC \leftarrow 0$$

The timing diagram of Fig. 5-7 shows the time relationship of the control signals.

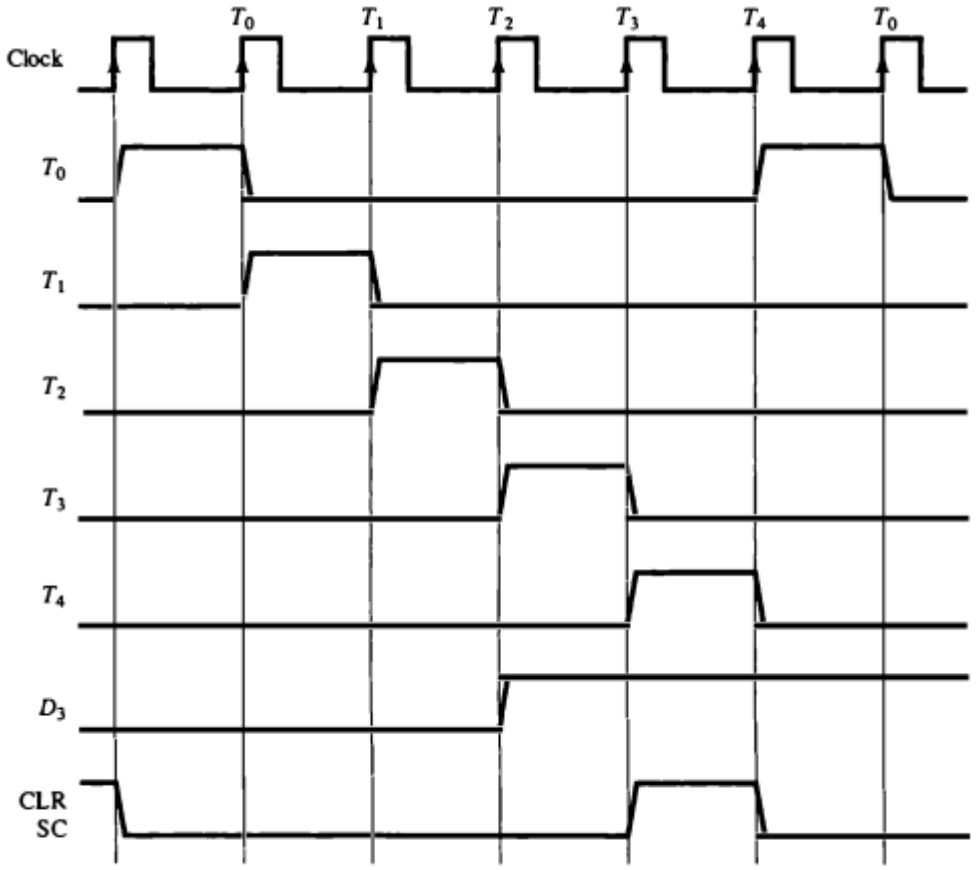


Figure 5-7 Example of control timing signals.

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 tum activates the timing signal T_0 out of the decoder. T_0 is active during one clock cycle. The positive clock transition labelled T_0 in the diagram will trigger only those registers whose control inputs are transition, to timing signal T_0. SC is incremented with every positive clock transition unless its CLR input is active. This produces the sequence of timing signals T_0, T_1, T_2, T_3, and T4 and so on, as shown in the diagram. If SC is not cleared, the timing signals will continue with T5, T6 up to T_{15} and back to T_0

Bus Organization

To achieve a reasonable speed of operation, a computer must be organized so that

all its units can handle one full word of data at a given time. When a word of data is transferred between units, all its bits are transferred in parallel, that is, the bits are transferred simultaneously over many wires, or lines, one bit per line. A group of lines that serves as a connecting path for several devices is called a bus. In addition to the lines that carry the data, the bus must have lines for address and control purposes.

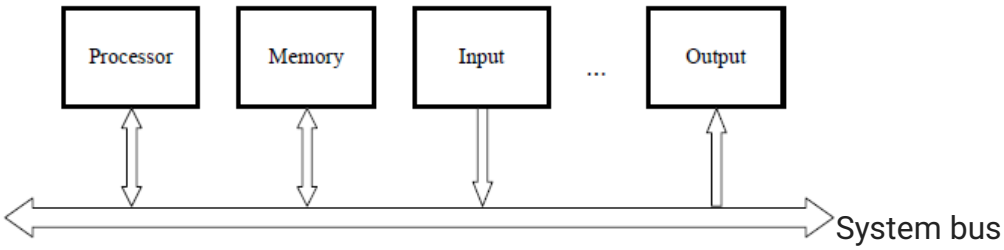
The different buses are

(i) Address bus: Address bus is used to carry the address. It is unidirectional bus. The address is sent to from CPU to memory and I/O port and hence unidirectional.

(ii)Data bus: Data bus is used to carry or transfer data to and from memory and I/O ports. They are bidirectional. The processor can read on data lines from memory and I/O port and as well as it can write data to memory.

(iii)Control bus: Control bus is used to carry control signals in order to regulate the control activities. They are bidirectional. The CPU sends control signals on the control bus to enable the outputs of addressed memory devices or port devices.

Single bus structure:



The simplest way to interconnect functional units is to use a single bus, as shown in Figure. All units are connected to this bus. Because the bus can be used for only one transfer at a time, only two units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of the bus. The main virtue of the single-bus structure is its low cost and is flexibility for attaching peripheral" devices. Systems that contain multiple buses achieve more concurrency in operations by allowing two or more transfers to be carried out at the same time. This leads to better performance but at an increased cost.

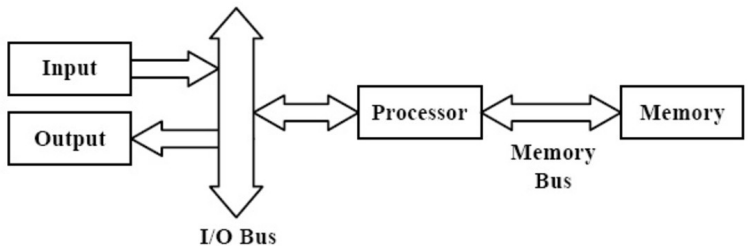
The devices connected to a bus vary widely in their speed of operation. Some electromechanical devices, such as keyboards and printers are relatively slow. Other devises like magnetic or optical disks, are considerably faster. Memory and

processor units operate at electronic speeds, making them the fastest parts of a computer. Because all these devices must communicate with each other over a bus, an efficient transfer mechanism that is not constrained by the slow devices and that can be used to smooth out the differences in timing among processors, memories, and external devices is necessary.

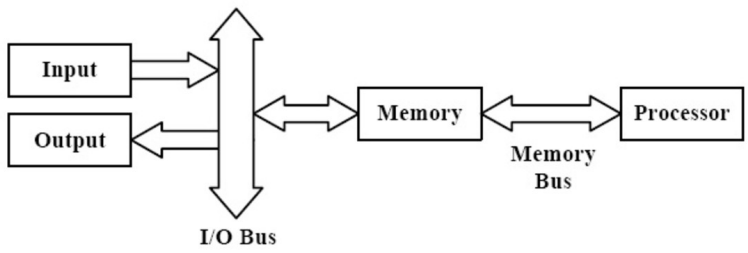
A common approach is to include buffer registers with the devices to hold the information during transfers. To illustrate this technique, consider the transfer of an encoded character from a processor to a character printer. The processor sends the character over the bus to the printer buffer. Since the buffer is an electronic register, this transfer requires relatively little time. Once the buffer is loaded, the printer can start printing without further intervention by the processor. The bus and the processor are no longer needed and can be released for other activity. The printer continues printing the character in its buffer and is not available for further transfers until this process is completed. Thus, buffer registers smooth out timing differences among processors, memories, and I/O devices. They prevent a high-speed processor from being locked to a slow I/O device during a sequence of data transfers. This allows the processor to switch rapidly from one device to another, interweaving its processing activity with data transfers involving several I/O devices.

Two bus Structures

Configuration 1



Configuration 2



In two bus structure, one bus is used to fetch instruction while other is used to fetch data, required for execution. It is to overcome the bottleneck of single bus structure.Various units are connected through two independent buses.

In configuration 1, I/O units are connected to the processor though an I/O bus and Memory is connected to the processor through the memory bus. I/O bus consists of address, data and control bus Memory bus also consists of address, data and control bus. In this type of arrangements processor completely supervises the transfer of information to and from I/O units. All the information is first taken to processor and from there to the memory. Such kind of transfers are called as program controlled transfer.

In configuration 2, I/O units are directly connected to the memory and not to the processor

The I/O units are connected to special interface logic known as Direct Memory Access (DMA) or an I/O channel. This is also called as Peripheral Processor Unit (PPU)

In this the data from the I/O device is directly sent to memory bypassing the processor.

Representation of Signed Binary Numbers:

There are three types of representations for signed binary numbers

Sign-Magnitude form:

For n bit binary number, 1 bit is reserved for sign symbol. If the value of sign bit is 0, then the given number will be positive, else if the value of sign bit is 1, then the given number will be negative. Remaining (n-1) bits represent magnitude of the number. Since magnitude of number zero (0) is always 0, so there can be two representation of number zero (0), positive (+0) and negative (-0), which depends on value of sign bit.

1’s complement form:

Since, 1’s complement of a number is obtained by inverting each bit of given number. So, we represent positive numbers in binary form and negative numbers in 1’s complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and you can directly represent it in simple binary form, but if value of sign bit 1, then number is negative and you have to take 1’s complement of given binary number.

2’s complement form:

Since, 2’s complement of a number is obtained by inverting each bit of given number plus 1 to least significant bit (LSB). So, we represent positive numbers in binary form and negative numbers in 2’s complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and you can directly represent it in simple binary form, but if value of sign bit 1, then number is negative and you have to take 2’s complement of given binary number