UNIT-III
REGISTER TRANSFER LANGUAGE AND DESIGN OF CONTROL UNIT

UNIT-III
REGISTER TRANSFER LANGUAGE AND DESIGN OF CONTROL UNIT

- **Register Transfer:** Register Transfer Language - Register Transfer - Bus and Memory Transfers - Arithmetic Micro operations - Logic Micro operations - Shift Micro Operations.
- **Control Unit:** Control Memory - Address Sequencing – Micro program Example - Design of Control Unit.

Register Transfer Language

- Digital systems are composed of modules that are constructed from digital components, such as registers, decoders, arithmetic elements, and control logic.
- The modules are interconnected with common data and control paths to form a digital computer system.
- The operations executed on data stored in registers are called *microoperations*
- A microoperation is an elementary operation performed on the information stored in one or more registers.
- Examples are shift, clear and load.

Register Transfer Language

- The internal hardware organization of a digital computer is best defined by specifying
  - The **set of registers** it contains and their functions.
  - The **sequence of microoperations** performed on the binary information stored.
  - The **control that initiates** the sequence of microoperations.

Register Transfer Language

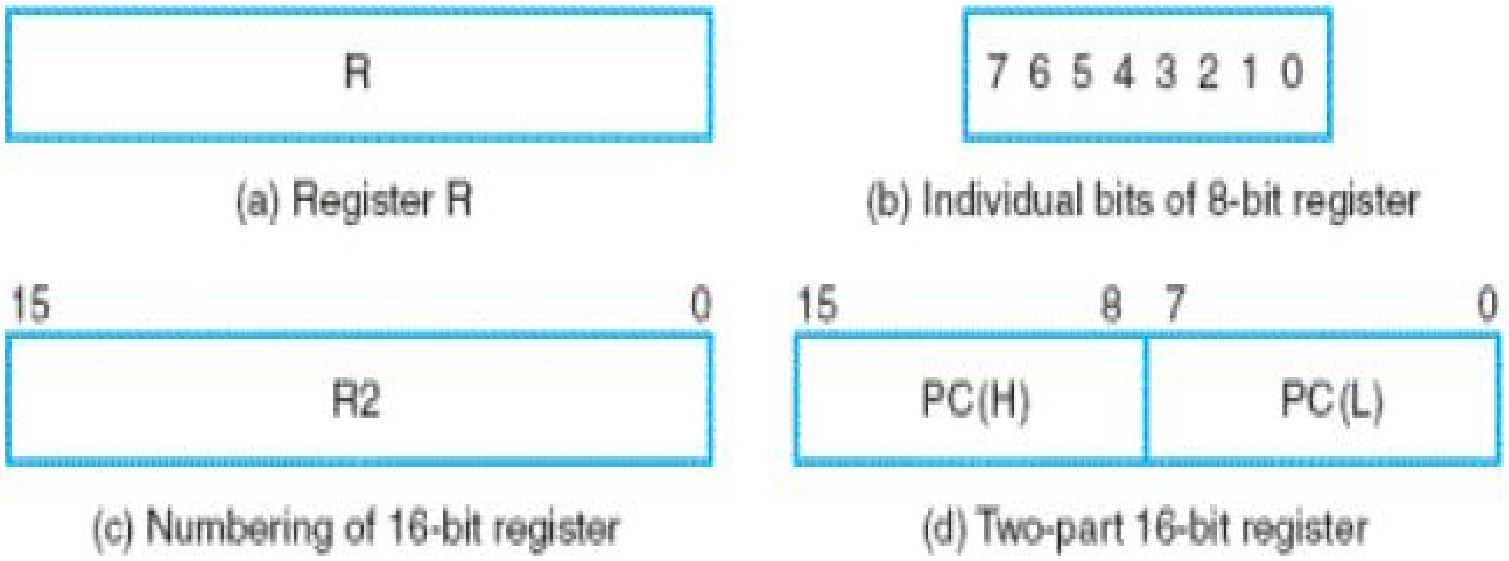
- The symbolic notation used to describe the micro operation transfers among register is called a register transfer language.
- A programming language is a procedure for writing symbols to specify a given computational process.
- A register transfer language is a system for expressing in symbolic form the micro operation sequences among the register of a digital module

Register Transfer

- Computer Registers are designated by capital letters to denote its function.
- The register that holds an address for the memory unit is called as MAR.
- The program counter register that holds the address of next instruction is called as PC .
- IR is the instruction register and R1 is a processor register.

Register Transfer

REPRESENTATION OF REGISTER



(a) Register R

(b) Individual bits of 8-bit register

(c) Numbering of 16-bit register

(d) Two-part 16-bit register

Register Transfer

- The individual flip-flops in an *n*-bit register are numbered from 0 to *n*-1 (from the right position toward the left position)

- The most common way to represent a register is by a rectangular box with the name of the register inside, as in Fig(a).

- The individual bits can be distinguished as in Fig(b).

- The numbering of bits in a 16-bit register can be marked on top of the box as shown in Fig(c).

Register Transfer

- A 16-bit register is partitioned into two parts in Fig (d).
- Bits 0 through 7 are assigned the symbol L (for low order byte) and bits 8 through 15 are assigned the symbol H (for high order byte).
- The name of the 16-bit register is PC. The symbol PC(O—7) or PC(L) refers to the low-order byte and PC(8—15) or PC(H) to the high-order byte.

Register Transfer

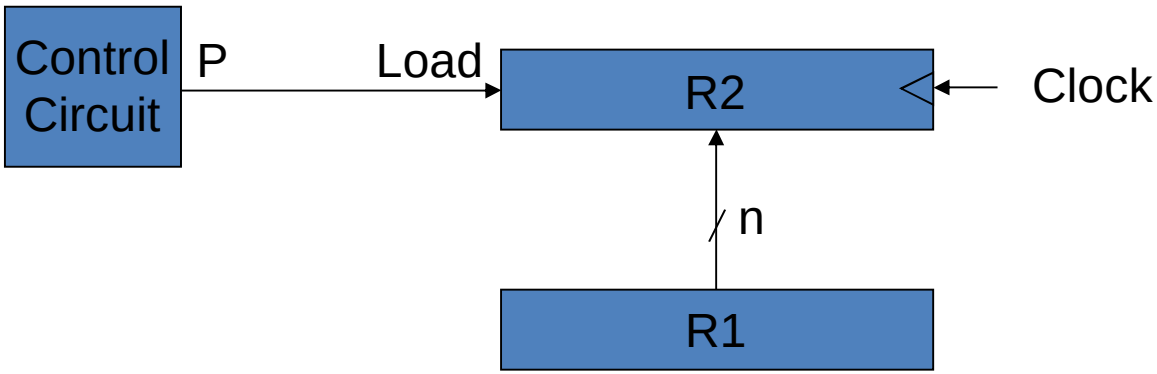
- Information transfer from one register to another is described by a *replacement operator:* **R2 ← R1**
- This statement denotes a transfer of the content of register R1 into register R2
- The content of the R1 (source) does not change
- The content of the R2 (destination) will be lost and replaced by the new data transferred from R1
- If the transfer is to occur only under a predetermined control condition, designate it by

  If (P = 1) *then* (R2 ← R1) or P: R2 ← R1,

  where P is a control function that can be either 0 or 1

Register Transfer

- Hardware implementation of a controlled transfer: P: R2 ← R1



Register Transfer

Basic Symbols for Register Transfers		
Symbol	Description	Examples
Letters & numerals	Denotes a register	MAR, R2
Parenthesis ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two microoperations	R2 ← R1, R1 ← R2

Bus and Memory Transfers

- Paths must be provided to transfer information from one register to another.
- A *Common Bus System* is a scheme for transferring information between registers in a multiple-register configuration.
- A bus is a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signals determines which register is selected by the bus during each particular register transfer

COMMON BUS SYSTEM CONFIGURATION

- Constructing a common bus system is done by

  a. Using multiplexers

  b. Using three state buffers.

Using Multiplexers:

- The multiplexers select the source register whose binary information is then placed on the bus.
- Each register has four bits, numbered 0 through 3.

Using Multiplexers

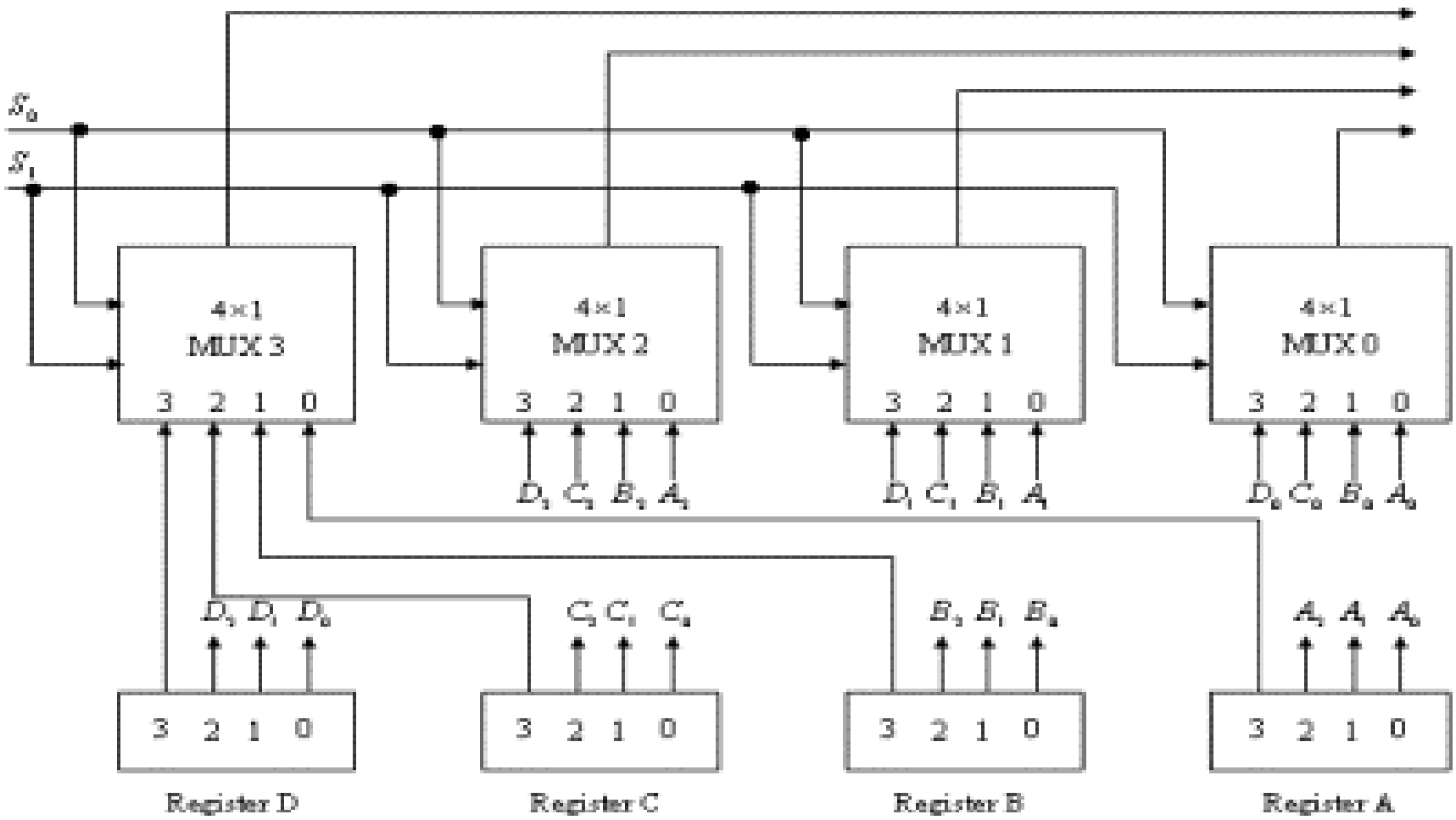
- The bus consists of four 4 x 1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1 and S0.
- we use labels to show the connections from the outputs of the registers to the inputs of the multiplexers.
- For example, output 1 of register A is connected to input 0 of MUX 1 because this input is labeled A1.

Using Multiplexers

- The two selection lines S0 and S1 are connected to the selection inputs of all four multiplexers.
- The following table shows the register that is selected by the bus for each of the four possible binary values of the selection lines.

S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

Using Multiplexers



Using Multiplexers

- The number of multiplexers needed to construct the bus is equal to n.
- The size of each multiplexer must be k x 1 since it multiplexes k data lines.

For example:

- A common bus for eight registers of 16 bits each requires 16 multiplexers, one for each line in the bus.
- So Each multiplexer must have eight data input lines and three selection lines to multiplex one significant bit in the eight registers.

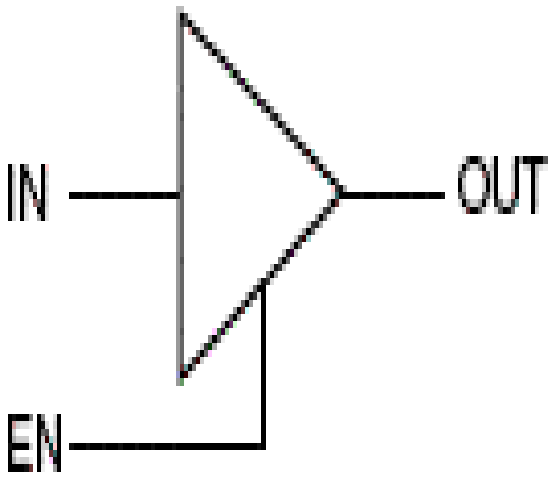
Three-state gate

- Three state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate and the third state is a high impedance state.
- The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have logic significance.

Three-state gate

- Three-state gates may perform any conventional logic, such as AND or NAND.
- However, the one most commonly used in the design of a bus system is the buffer gate.
- It is distinguished from a normal buffer by having both a normal input and a control input.

Three-state gate



(a) Logic symbol

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

(b) Truth table

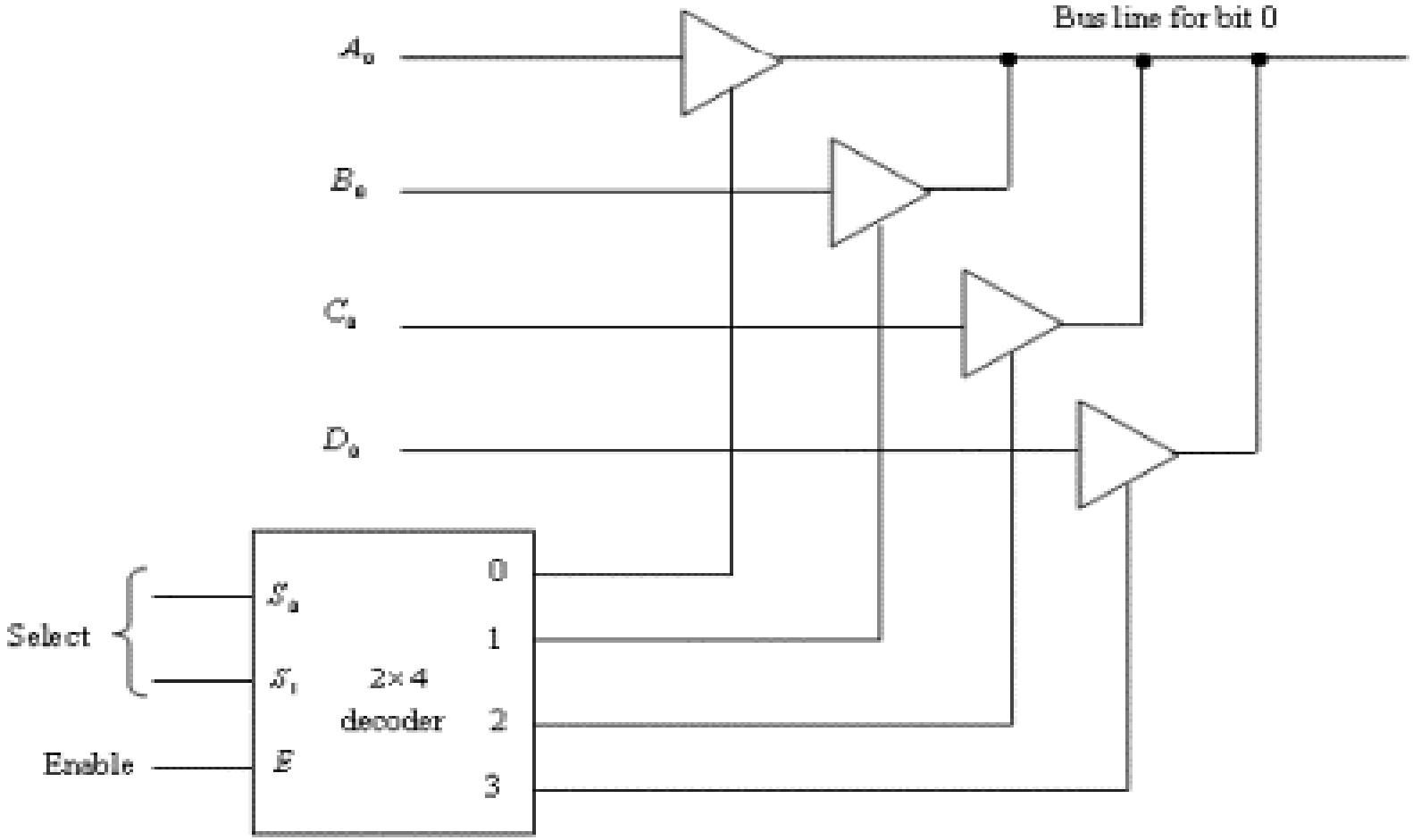
Three-state gate

- When the control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer, with the output equal to the normal input.

- When the control input is 0, the output is disabled and the gate goes to a high-impedance state, regardless of the value in the normal input.

- The high-impedance state of a three-state gate provides a special feature not available in other gates.

Three-state gate



Three-state gate

- The outputs of four buffers are connected together to form a single bus line.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- No more than one buffer may be in the active state at any given time.
- The connected buffers must be controlled so that only one three-state buffer has access to the bus line while all other buffers are maintained in a high-impedance state.

Three-state gate

- One way to ensure that no more than one control input is active at any given time is to use a decoder, as shown in the diagram.
- To construct a common bus for four registers of n bits each using three state buffer, we need n circuit with four buffer receives one significant bit from the four registers.
- Each common output produces one of the lines for the common bus for a total of n lines .
- Only one decoder is necessary to select between the four registers.

MEMORY TRANSFER

- A memory word will be symbolized by the letter **M.**
- The particular memory word among the many available word is selected by the memory address during the transfer.
- This will be done by enclosing the address in square brackets following the letter M.
- Consider a memory unit that receives the address from a register, called the address register, symbolized by **AR.**
- **The data are transferred to** another register, called the data register, symbolized by **DR**

MEMORY TRANSFER

READ OPERATION:

- The transfer of information from a memory word to the outside environment.

Read: DR ← M[AR]

- This causes a transfer of information into DR from the memory word M selected by the address in AR.

MEMORY TRANSFER

WRITE OPERATION:

- The write operation transfers the content of a data register to a memory word M selected by the address.

Write: M[AR] ← DR

- This causes a transfer of information from DR into the memory word M selected by the address in AR.