Module 2

→ Basic Computer Organization & Design
Chapter 4 (Mano's book)

4.1. Register Transfer Language:

- **Digital System:** Interconnection of digital hardware modules that accomplish a specific information processing task.

— Digital System vary in size & complexity:
— Few IC's to a complex, interconnected and interacting digital computers.

⟹ Digital System design invariably uses a modular approach. The modules are constructed from such digital components as registers, decoders, arithmetic elements, and control logic.
— The various modules are interconnected with common data and control paths to form a digital computer system.

- **MICRO OPERATION:**
— Digital modules are best defined by the registers they contain and the operations that are performed on the data stored in them. ①
— **Definition:** The operations executed on data stored in registers are called MICROOPERATIONS.

Definition: A microoperation is an elementary operation performed on the information stored in one or more registers.

— The result of the operation may replace the previous binary information of a register or may be transferred to another register.

Examples: Shift, count, clear, and load.

Chapter 2: Some Digital Components } Registers that implement microoperations
e.g. Counter = Increment & load
eg. Bidirectional shift register: Shift right, Shift left micro...

→ Internal Hardware Organization of a digital Computer: Best defined by specifying:

1. The set of registers it contains & their function
2. The sequence of microoperations performed on the binary information stored in the registers.
3. The control that initiates the sequence of microoperations.

→ It is possible to specify the sequence of microoperations in a computer by explaining every operation in words, but this procedure usually involves a lengthy descriptive explanation. It is more convenient to adopt a suitable symbology to describe the

Sequence of transfers between registers and the various arithmetic and logic microoperations associated with the transfers.

— The use of symbols instead of a narrative explanation provides an organized and concise manner for listing the microperation sequences in registers and the control functions that initiate them.

Definition: The SYMBOLIC NOTATION used to describe the microoperation transfers among registers is called a REGISTER TRANSFER LANGUAGE.

— The term "Register transfer" implies the availability of hardware logic circuits that can perform a stated microoperation and transfer the result of the operation to the same or another register.

— Word "language"... borrowed from programmers, who apply this term to programming languages.

— Programming language: A procedure for writing symbols to a specify a given computational process.

— Natural language e.g. English

— A RTL is a system for expressing in symbolic form the microoperation sequences among the registers of a digital module.

— RTL: Convenient tool for describing the internal organization of digital computers in concise and precise manner

RTL: Can also be used to facilitate design process of digital systems

- RTL adopted here is believed to be as simple as possible, so it should not take very long to memorize
- Define symbols for various types of microoperations and at the same time, describe associated hardware that can implement the stated microoperations.
- Symbolic designation : will be used to specify Register transfers, the microoperations and the control functions that describe the internal hardware organization of digital Computers
- Other symbology in use can easily be learned once this language has become familiar, for most of the differences between RTL's consist of variations in detail rather than in overall purpose.

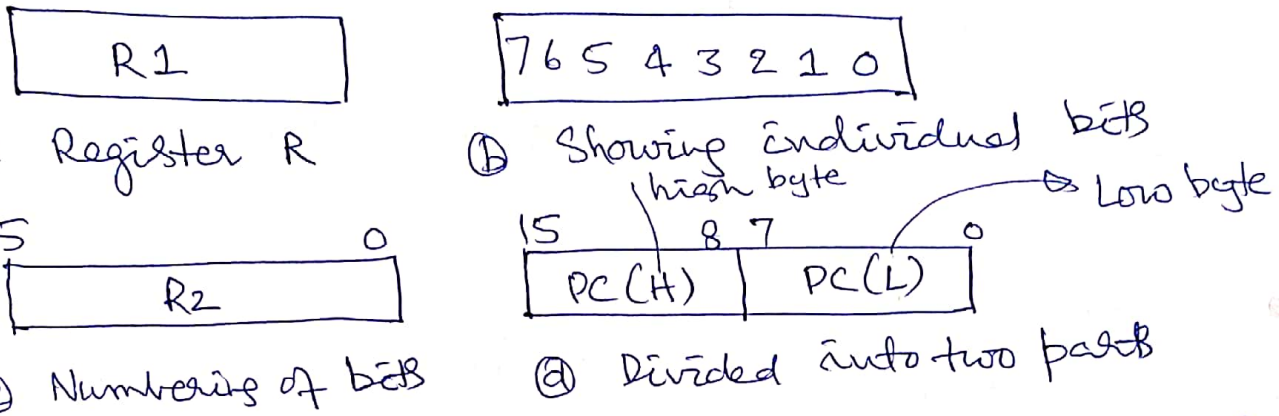
4.2. REGISTER TRANSFER:

→ Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register.

- Memory Address Register (MAR): Register that holds an address for the memory unit.
- PC = Program Counter
  IR = Instruction Register
  R1 = Process Register.

— The individual flip-flops in an n-bit register are numbered in a sequence from '0' through 'n-1', starting from '0' in the rightmost position and increasing the numbers toward the left.

Fig. 4.1. Block Diagram of Register



(a) Register R (b) Showing individual bits

(c) Numbering of bits (d) Divided into two parts

—Common way to represent a register: Rectangular Box with the name of the register inside

- REGISTER TRANSFER: Information transfer from one register to another is designated in symbolic form by means of a replacement operator. The statement

$$R2 \leftarrow R1$$

denotes a transfer of the content of register R1 into register R2. It designates a replacement of the content of R2 by the content of R1. By definition, the content of the source register R1 does not change after the transfer.

——— A statement that specifies a register transfer implies that circuits are available from the outputs of the source register to the inputs of the destination register

and that the destination register has a parallel load capability.

— Normally, we want the transfer to occur only under a predetermined control condition. This can be shown by means of an IF-THEN statement

$$IF\ (P=1),\ THEN\ (R2 \leftarrow R1),$$

where P is a control signal generated in the control section.

- CONTROL FUNCTION:

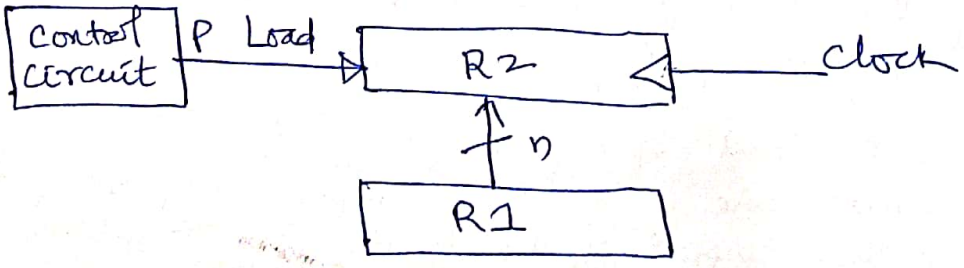
It is sometimes convenient to seperate the control variables from the register transfer operation by specifying a CONTROL FUNCTION. A control function is a Boolean variable that is equal to '1' or '0'. The control function is included in the statement as follows:

$$P:\ R2 \leftarrow R1.$$

The control condition is terminated with a colon. It symbolizes the requirement that the transfer operation be executed by the hardware only if P=1.

→ Every statement written in a register transfer notation implies a hardware construction for implementing the transfer:

Fig: Block Diagram that depicts transfer from R1 to R2



- The 'n' outputs of register R1 are connected to the 'n' inputs of register R2. The letter 'n' will be used to indicate any number of bits for the register. It will be replaced by an actual number when the length of the register is known.

— Register R2 has a load input that is activated by the control variable P. It is assumed that the control variable is synchronized with the same clock as the one applied to the register. As shown in the timing diagram, P is activated in the control section by the rising edge of a clock pulse at time 't'. The next positive transition of the clock at time 't+1' finds the load input active and the data inputs of R2 are then loaded into the register in parallel. P may go back to '0' at time t+1; otherwise, the transfer will occur with every clock pulse transition while P remains active.

Note: Note that the clock is not included as a variable in the register transfer statements. It is assumed that all transfers occur during a clock edge transition. Even though the control condition such as P becomes active just after time t, the actual transfer does not occur until the register is triggered by the next positive transition of the clock at time 't+1'.

→ Table 4-1: Basic Symbols of the Register Transfer Notation

Comma: Used to Specify the following

Comma: Used to Seperate two or more operations that are executed at the same time.

The Statement

T: R2 ← R1, R1 ← R2

denotes an operation that exchanges the contents of two registers during one common clock pulse provided that T=1. This simultaneous operation is possible with registers that have edge-triggered flip flops

4.3. BUS and MEMORY TRANSFERS:

→ A typical digital Computer has many registers, and paths must be provided to transfer information from one register to another.

→ The number of wires will be excessive if Seperate lines are used between each register and all other registers in the System.

→ A MORE EFFICIENT Scheme for transferring information between registers in a multiple-register configuration is a COMMON BUS System.

→ A bus Structure consists of a Set of common lines one for each bit of a register, through which binary information is transferred one at a time.

→ Control Signals determine which register is Selected by the bus during each particular register transfer

→ One way of constructing a common bus system is with multiplexers. The multiplexers select the source register whose binary information is then placed on the bus.

→ In general, a bus system will multiplex 'k' registers of 'n' bits each to produce an 'n' - line common bus.

— The number of multiplexers needed to construct the bus is equal to n, the number of bits in each register.

— The size of each multipler must be k×1 since it multiplexes 'k' data lines.

Example :— Common bus for 8 registers of 16 bits each requires 16 multiplexers, one for each line in the bus.

— Each multiplexer must have 8 data input lines and 3 selection lines to multiplex one significant bit in the 8 registers.

— The transfer of information from a bus into one of many destination register can be accomplished by connecting the bus lines to the inputs of all destination registers and activating the load control of the particular destination register selected.

→ The symbolic statement for a bus transfer may mention the bus or its presence may be implied in the statement. When the bus is included in the statement, the register transfer is symbolized as follows

BUS ← C, R1 ← BUS. (OR) R1 ← C

From this statement the designer knows which control signals must be activated to produce the transfer through the bus.

Note: Bus system can be constructed with 3-state gates instead of multiplexers.

A 3-state gate is a digital circuit that exhibits 3 states (0, 1, High impedance state)

→ MEMORY TRANSFER

- The transfer of information from a memory word to the outside environment is called READ operation
- The transfer of new information to be stored into the memory is called a WRITE operation.
- A memory word will be symbolized by the letter M.
- The particular memory word among the many available is selected by the memory address during the transfer. It is necessary to specify the address of M when writing memory transfer operations. 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. The read operation can be stated as follows:

Read: DR ← M[AR] : transfer to DR from memory word M with AR address

Write: M[AR] ← R1 : transfer from R1 to memory word M selected by the address in AR.

4-4: Types of Microoperations: Arithmetic Microoperations

— A microoperation is an elementary operation performed with the data stored in registers. The microoperations most often encountered in digital computers are classified into 4 categories:

1. Register Transfer Microoperations transfer binary information from one register to another.
2. Arithmetic microopperations perform arithmetic operation on numeric data stored in registers
3. Logic Microoperations perform bit manipulation operations on nonnumeric data stored in registers
4. Shift microoperations perform shift operation on data stored in registers.

Note:— Register transfer microoperation does not change the information content when the binary information moves from the source register to the destination register.

— The other 3 types of microoperations change the information content during the transfer.

⟶ BASIC ARITHMETIC MICROOPERATIONS:

Addition, Subtraction, Increment, Decrement & Shift.

Shifts ⟶ Arithmetic Shifts (later)

Examples:- $R3 \leftarrow R1 + R2$: ADD Microoperation

$R3 \leftarrow R1 - R2$

The arithmetic operations of multiply and divide are not listed in the Table 4.3. These two operations are valid arithmetic operations but are not included in the basic set of microoperations. The only place where these operations can be considered as microoperations is in a digital system, where they are implemented by means of a combinational circuit.

Note:- In most computers, the multiplication operation is implemented with a sequence of add & shift microoperations

- Division is implemented with a sequence of subtract and shift microoperations.

Example Hardware:

- 4 bit binary adder based on full adders

→ Increment/Decrement microoperations are implemented with a combinational circuit or with a binary up-down counter.

4.5. Logic Microoperations:

- Logic Microoperations Specify binary operations for Strings of bits Stored in registers. These operations consider each bit of the register Seperately and treat them as binary variables.

Example: Exclusive-OR microoperation with the contents of two registers R1, R2

$$P: R1 \leftarrow R1 \oplus R2$$

Example:

```
1010   Content of R1
1100   Content of R2
----
0110   Content of R1 after P=1
```

SPECIAL SYMBOLS

→ Special Symbols will be adopted for the logic microoperations OR, AND and COMPLEMENT to distinguish them from the corresponding Symbols used to express Boolean functions.

Reason (I): $\vee$ ... OR ; $\wedge$ ... AND microperation.

↓

COMPLEMENT ... Bar on top of Symbol

- By using different Symbols, it will be possible to differentiate between a logic microoperation and a Control (or Boolean) function.

Reason (I): + ... used for arithmetic plus or logic OR

→ Although the + symbol has two meanings, it will be possible to distinguish between them by noting where the symbol occurs.

— When the symbol + occurs in a microoperation, it will denote an arithmetic plus.

— When it occurs in a control (or Boolean) function, it will denote an OR operation.

— We will never use it to symbolize an OR microoperation.

Example:

P + Q: R1 ← R2 + R3, R4 ← R5 V R6.

— + between P and Q is an OR operation between two binary variables of a control function

— + between R2, R3 ... Add microoperation

— OR logic operation by symbol 'V' between R5, R6.

• SOME APPLICATIONS

— Logic microoperations are very useful for manipulating individual bits or a portion of a word stored in a register.

— They can be used to change bit values, delete a group of bits, or insert new bit values into a register.

4.6. SHIFT MICROOPERATIONS:

- Shift Microoperations
  - Serial transfer of data
  - Also, used in conjunction with arithmetic, logic & other data-processing operations

- Contents of a register can be shifted to the left or right.
- At the same time that the bits are shifted, the first flip-flop receives its binary information from the serial input.
- SHIFT LEFT: During a shift-left operation, the serial input transfers a bit into the rightmost position
- SHIFT RIGHT: During a shift-right operation, the serial input transfers a ~~bit~~ bit into the leftmost position

→ The information transferred through the serial input determines the type of shift.

There are 3 types of shifts: LOGICAL, CIRCULAR and ARITHMETIC.

• LOGICAL SHIFT:

- A LOGICAL SHIFT is one that transfers '0' through the serial input.

Symbols for type of shift: shl ... logical shift-left
shr ... " shift-right

Example: (i) R1 ← Shl R1 } Microoperations
(ii) R2 ← Shr R2 }

(i) Specifies a 1-bit shift to the left of the content of register R1

(ii) Specifies 1-bit shift to the right of the content of register R2.

i.e. Register symbol must be same on both sides of arrow.

- The bit transferred to the end position through the serial input is assumed to be '0' during a logical shift.

- **CIRCULAR SHIFT:** (also known as ROTATE operation)
— It circulates the bits of the register around the two ends without loss of information. This is accomplished by connecting the serial output of the shift register to its serial input.

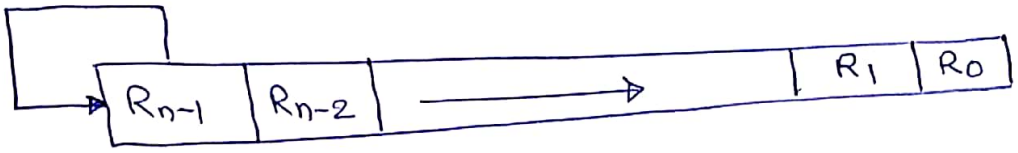
Symbols: cil ... circular shift left
cir ... circular shift right

- **ARITHMETIC SHIFT:** ... It is a microoperation that shifts a signed binary number to the left or right.
— An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2.

— Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiplied or divided by 2.

— The leftmost bit in a register holds the sign bit, and the remaining bits hold the number. The sign bit is '0' for positive and 1 for negative.

Fig. 4.11 Arithmetic Shift Right



— The above figure shows a typical register of 'n' bits. Bit R_{n-1} in the leftmost position holds the sign bit. R_{n-2} is the most significant bit of the number and R_0 is the least significant bit.

— The arithmetic shift-right leaves the sign bit unchanged and shifts the number (including sign bit) to the right. Thus R_{n-1} remains the same, R_{n-2} receives the bit from R_{n-1}, and so on for the other bits in the register. The bit in R_0 is lost.

⟶ The arithmetic shift-left inserts a '0' into R_0, and shifts all other bits to the left.