Course Contents

Unit-01: Introduction to Microprocessor

Unit-02: Intel 8085

Unit-03: Microoperations

Unit-04: Control Unit and Central Processing Unit

Unit-05: Fixed point Computer Arithmetic

Unit-06: Input and Output Organization

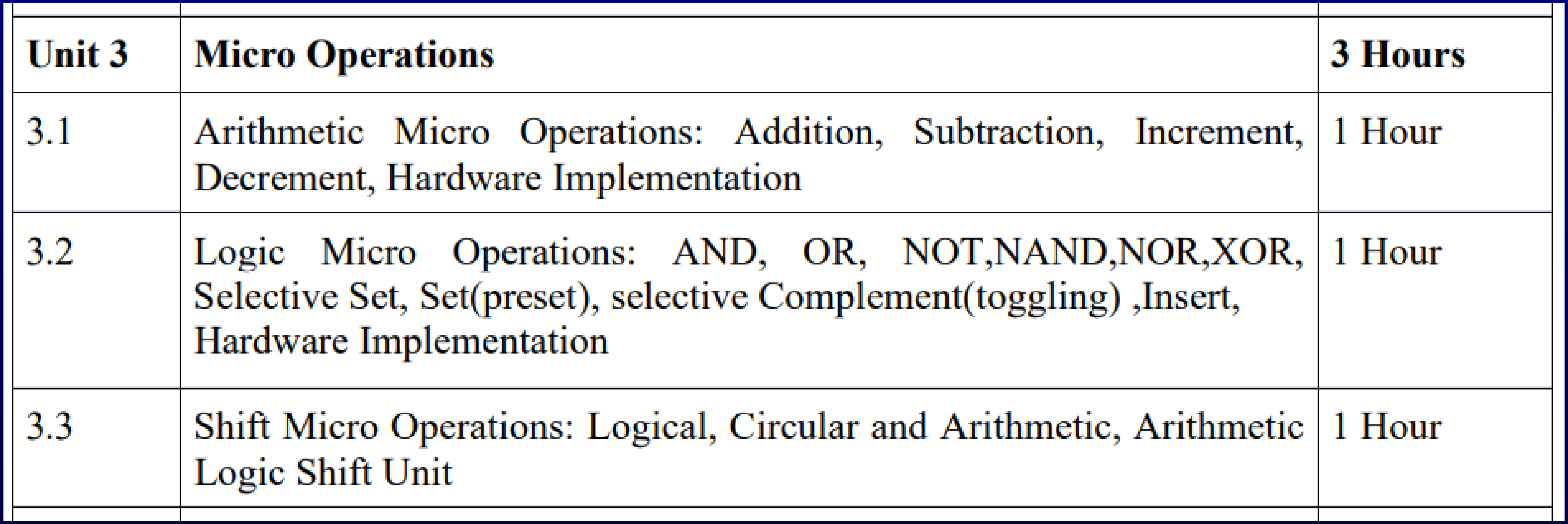
Unit-07: Memory Organization

Unit-08: Pipelining

Course Contents

Unit-03: Microoperations

Arithmetic Microoperations, Logic Microoperations, Shift Microoperations, Arithmetic Logic Shift Unit



Unit 3	**Micro Operations**	**3 Hours**
3.1	Arithmetic Micro Operations: Addition, Subtraction, Increment, Decrement, Hardware Implementation	1 Hour
3.2	Logic Micro Operations: AND, OR, NOT,NAND,NOR,XOR, Selective Set, Set(preset), selective Complement(toggling) ,Insert, Hardware Implementation	1 Hour
3.3	Shift Micro Operations: Logical, Circular and Arithmetic, Arithmetic Logic Shift Unit	1 Hour

Course Contents

Unit-03: Microoperations

- Register Transfer Language: Micro-operation, Register Transfer Language, Register Transfer, Control Function
- Arithmetic Micro Operations: Addition, Subtraction, Increment, Decrement, Hardware Implementation
- Logic Micro Operations: AND, OR, NOT, NAND, NOR, XOR, Selective Set, Set(preset), Selective Complement (toggling), Insert, Hardware Implementation
- Shift Micro Operations: Logical, Circular and Arithmetic, Arithmetic Logic Shift Unit

Register Transfer Language

Microoperation:

- ➢ A digital system is an interconnection of digital hardware modules that accomplish a specific information-processing task.
- Digital systems are modular in nature, with modules containing registers, decoders, arithmetic elements, control logic, etc.
- These digital components are defined by the registers that they contain and the operations performed on their data. These operations are called microoperations.
- Microoperations are elementary operations performed on the information stored in one or more registers.
- The result of the operation may replace the previous binary of a register or may be transferred to another register. Examples of microoperations introduced are shift, count, clear load etc.

Register Transfer Language

Microoperation:

- Examples of microoperations

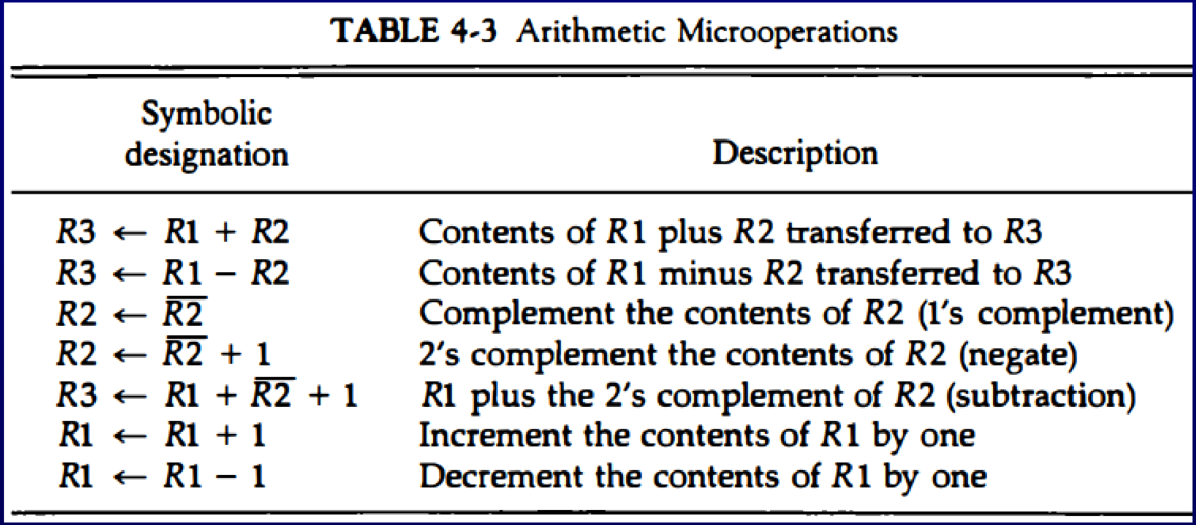


TABLE 4-3 Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

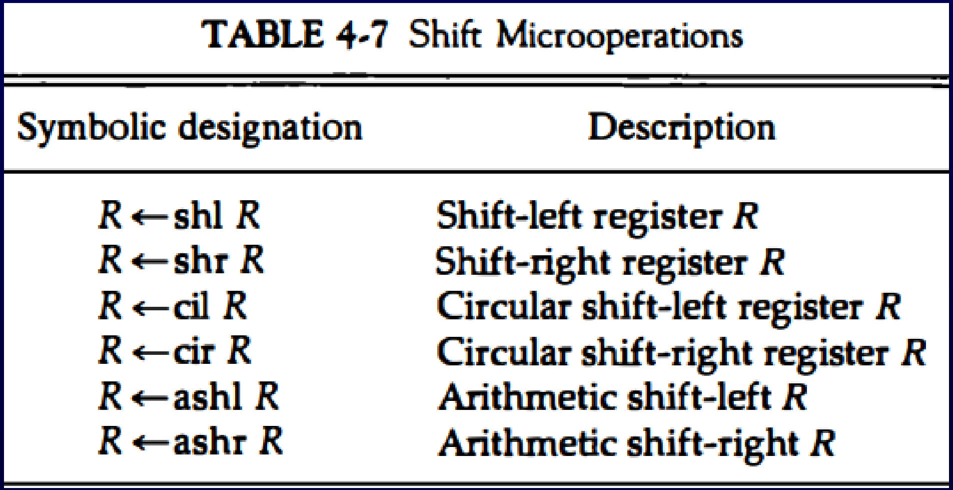
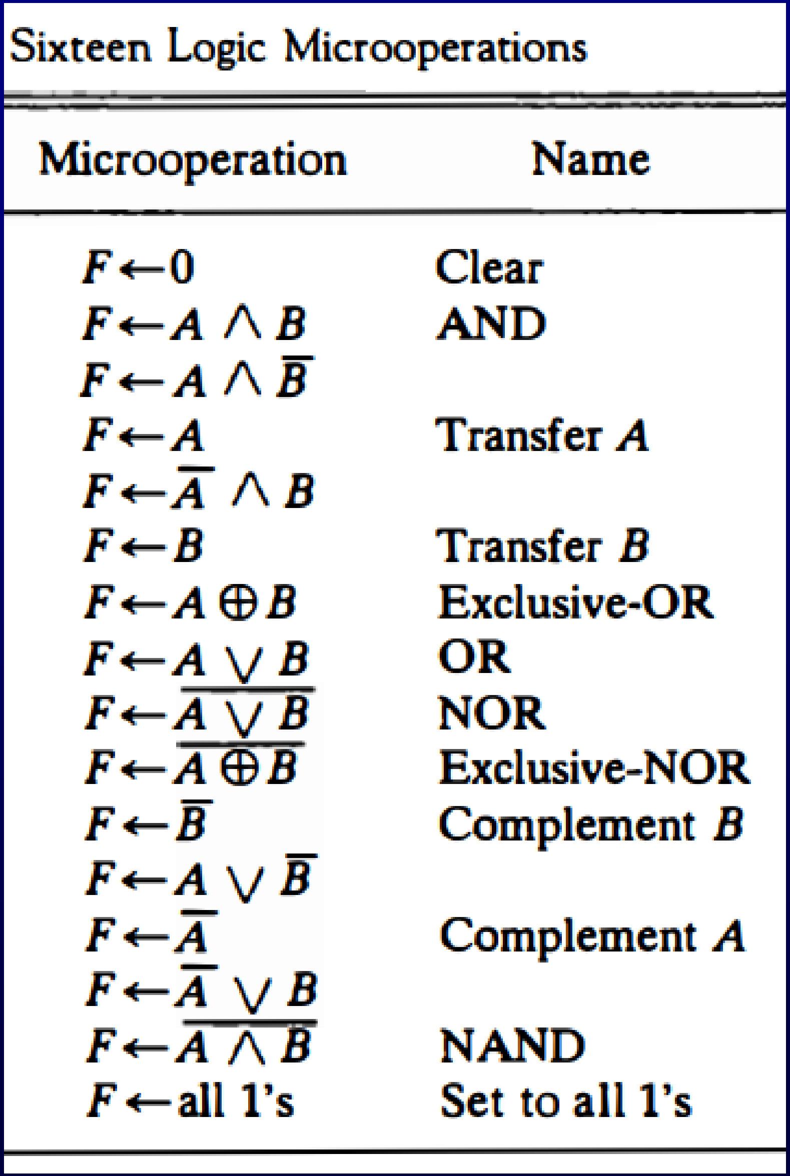


TABLE 4-7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R



Sixteen Logic Microoperations

Microoperation	Name
$F \leftarrow 0$	Clear
$F \leftarrow A \wedge B$	AND
$F \leftarrow A \wedge \overline{B}$	
$F \leftarrow A$	Transfer A
$F \leftarrow \overline{A} \wedge B$	
$F \leftarrow B$	Transfer B
$F \leftarrow A \oplus B$	Exclusive-OR
$F \leftarrow A \vee B$	OR
$F \leftarrow \overline{A \vee B}$	NOR
$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F \leftarrow \overline{B}$	Complement B
$F \leftarrow A \vee \overline{B}$	
$F \leftarrow \overline{A}$	Complement A
$F \leftarrow \overline{A} \vee B$	
$F \leftarrow \overline{A \wedge B}$	NAND
$F \leftarrow$ all 1's	Set to all 1's

Register Transfer Language

Hardware Organization:

The hardware organization of a digital computer is best defined by specifying:

- The set of register that it contains and their function.
- The sequence of microoperations performed on the binary information stored in the registers.
- The control signals that initiate the sequence of microoperations.

Register Transfer Language

Register Transfer Language:

- The Register Transfer Language is the symbolic representation of notations used to specify the sequence of micro-operations.
- In a computer system, data transfer takes place between processor registers and memory and between processor registers and input-output systems.
- 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 microoperation sequences in registers and the control functions that initiate them.

Register Transfer Language

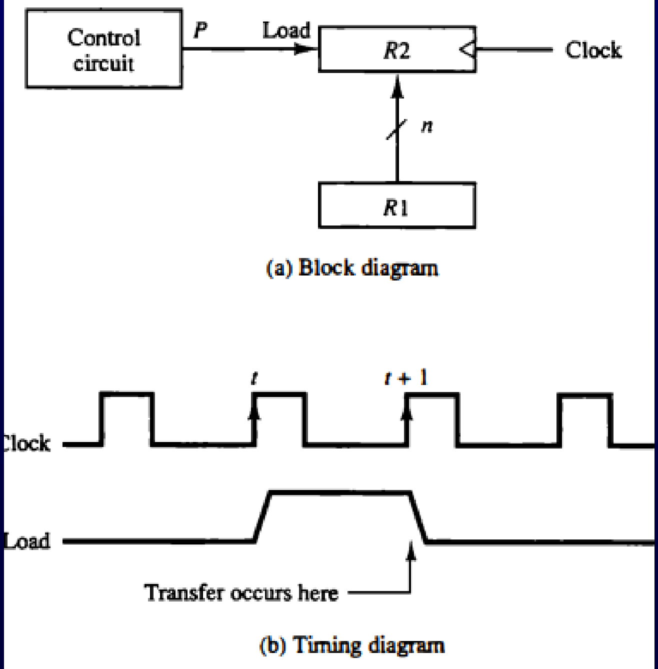
Register Transfer Language:

- ➢ 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.

Example of RTL: Control Function

P: R2 ← R1

Figure 4-2 Transfer from R1 to R2 when P = 1.



Register Transfer Language

Register Transfer:

Registers are denoted by capital letters and are sometimes followed by numerals, e.g.,

- MAR – Memory Address Register (holds addresses for the memory unit)
- PC – Program Counter (holds the next instruction’s address)
- IR – Instruction Register (holds the instruction being executed)
- R1 – Register1 (a CPU register)

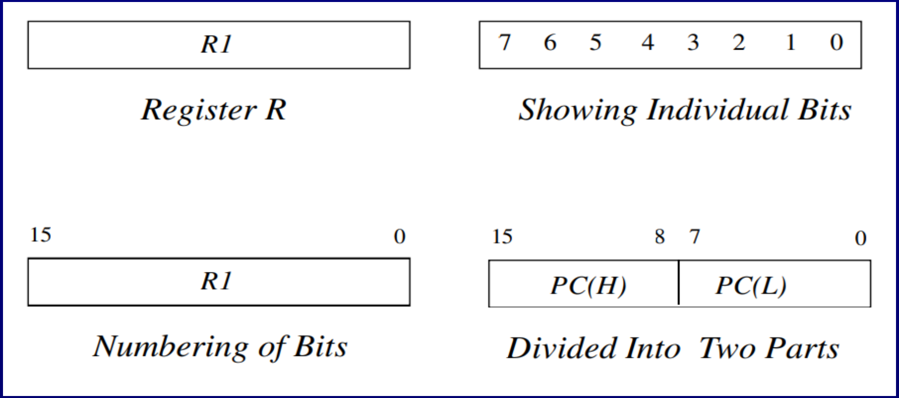
We can indicate individual bits by placing them in parentheses, e.g.,

PC(8-15), R2(5), etc

Register Transfer Language

Register Transfer:

a) Block Diagrams of Registers



b) Register Transfer Language Instructions:

- Register Transfer

  $R2 \leftarrow R1$

- Simultaneous Transfer

  $R2 \leftarrow R1, R1 \leftarrow R2$

- Conditional Transfer (Control Function)

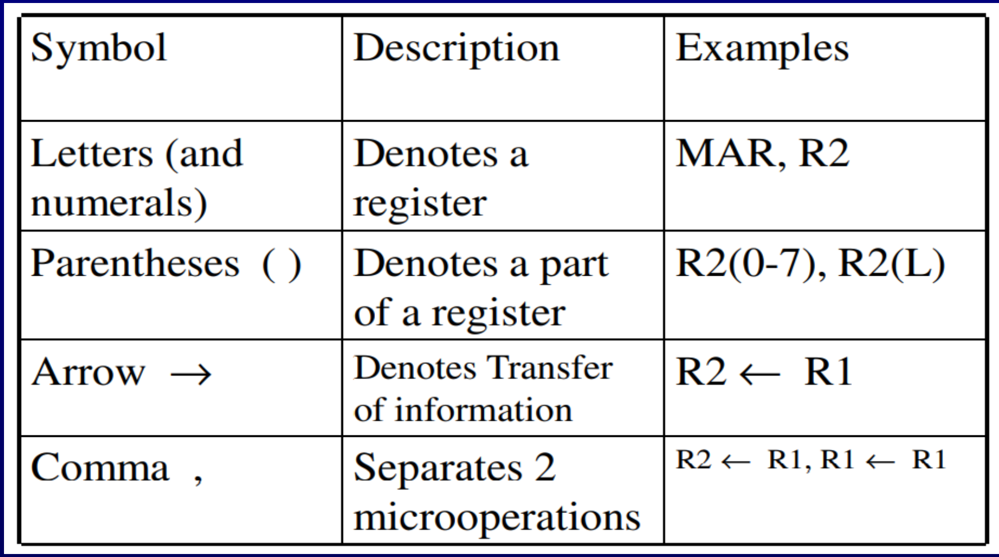
  $P: R2 \leftarrow R1$ or If $(P = 1)$ Then $R2 \leftarrow R1$

- Conditional, Simultaneous Transfer

  $T: R2 \leftarrow R1, R1 \leftarrow R2$

Register Transfer Language

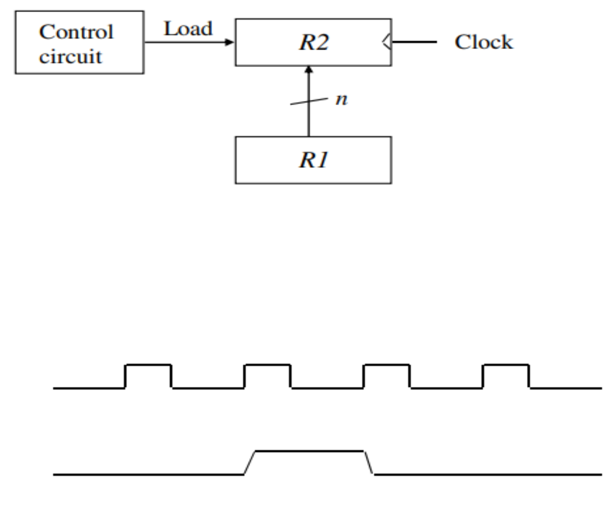
Basic Symbols for Register Transfer:



Symbol	Description	Examples
Letters (and numerals)	Denotes a register	MAR, R2
Parentheses ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow $\rightarrow$	Denotes Transfer of information	R2 $\leftarrow$ R1
Comma ,	Separates 2 microoperations	R2 $\leftarrow$ R1, R1 $\leftarrow$ R1

Register Transfer and Hardware:

Transfer from R1 to R2 when P = 1



Register Transfer Language

Bus and Memory Transfers :

The Bus

- A bus is a set of common wires that carries data between registers.
    - There is a separate wire for every bit in the registers.
    - There are also a set of control signals which determines which register is selected by the bus at a particular time.
- A bus can be constructed using multiplexer which enable a set of registers to share a common bus for data transfer.

Register Transfer Language

Memory Transfer:

- There are two primary operations involving memory:
  - Read – transferring data from memory
  - Write – transferring data into memory
- To indicate in Register Transfer Language that we are moving data from a memory address to the data register, we write:

  Read: $DR \leftarrow M[AR]$

  This causes a transfer of information into DR from the memory word M selected by the address in AR .
- To indicate in RTL that we are moving data from Register1 to a memory location, we write:

  Write: $M[AR] \leftarrow R1$

Register Transfer Language

Register Transfer Language: Micro-operations

Microoperations are classified into four categories:

- Register transfer microoperations (data moves from register to register)
- Arithmetic microoperations (perform arithmetic on data in registers)
- Logic microoperations (perform bit manipulation on data in registers)
- Shift microoperations (perform shift on data in registers)

Arithmetic Microoperations

Arithmetic Microoperations:

2.2 Arithmetic Microoperations: List of Arithmetic Microoperations, Binary Adder, Binary Adder-Subtractor, Binary Incrementer, Arithmetic Circuit

Arithmetic Microoperations

Arithmetic Microoperations:

Unlike register transfer microoperations, arithmetic microoperations change the information content.

The basic arithmetic microoperations are:

- addition
- subtraction
- increment
- decrement
- shift

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

Arithmetic Microoperations

Arithmetic Microoperations:

- Assume that the input data are in register R1.
- $\overline{R2}$ is the symbol for the 1's complement of R2.
- Adding 1 to the 1' s complement produces the 2' s complement.
- Adding the contents of R 1 to the 2' s complement of R2 is equivalent to R1 - R2.

$$R3 \leftarrow R1 + \overline{R2} + 1$$

Arithmetic Microoperations

Arithmetic Microoperations:

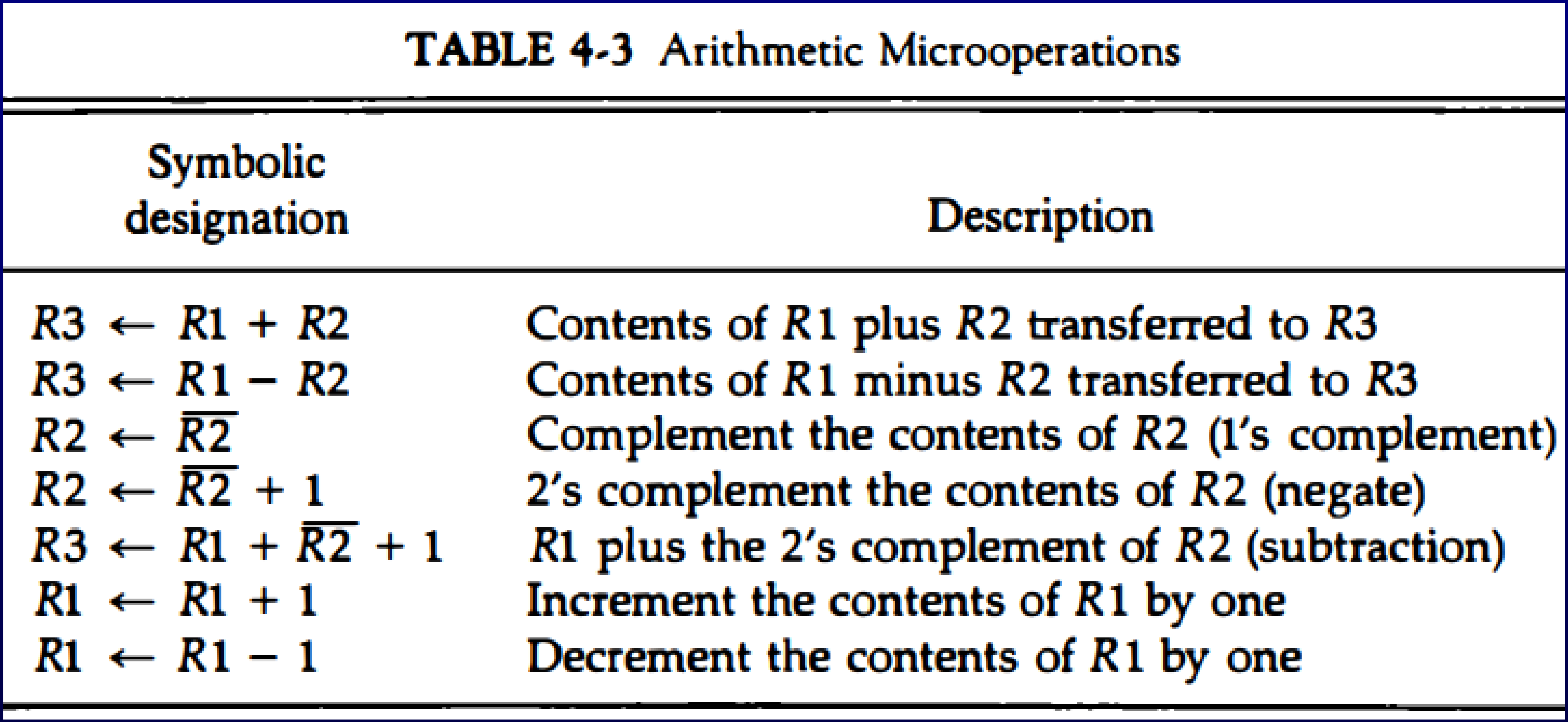


TABLE 4-3 Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

Multiplication and division are not considered microoperations.

- Multiplication is implemented by a sequence of adds and shifts.
- Division is implemented by a sequence of substracts and shifts.

Logic Microoperations

Logic Microoperations:

2.3 Logic Microoperations: List of Logic Microoperations, Hardware Implementation, Applications of Logic Microoperations.

Logic Microoperations

Logic Microoperations:

Logic microoperations are binary operations performed on corresponding bits of two-bit strings. These operations consider each bit of the register separately and treat them as binary variables. For example, the exclusive-OR microoperation with the contents of two registers R1 and R2 is symbolized by the statement

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

  1010 Content of R1

  1100 Content of R2

  0110 Content of R1 after P = 1

- Special Symbols used for logic operations:

  $\wedge$ - AND $\vee$ - OR $\oplus$ - XOR and $\overline{R2}$

This avoids confusing AND with multiplication, OR with addition, etc.

Logic Microoperations

Logic Microoperations: Special symbols

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.

For example, in the statement

$$P + Q: R1 \leftarrow R2 + R3,\quad R4 \leftarrow R5 \vee R6$$

the + between P and Q is an OR operation between two binary variables of a control function. The + between R2 and R3 specifies an add microoperation. The OR microoperation is designated by the symbol V between registers R5 and R6.

Logic Microoperations

Logic Microoperations: List of Logic Microoperations

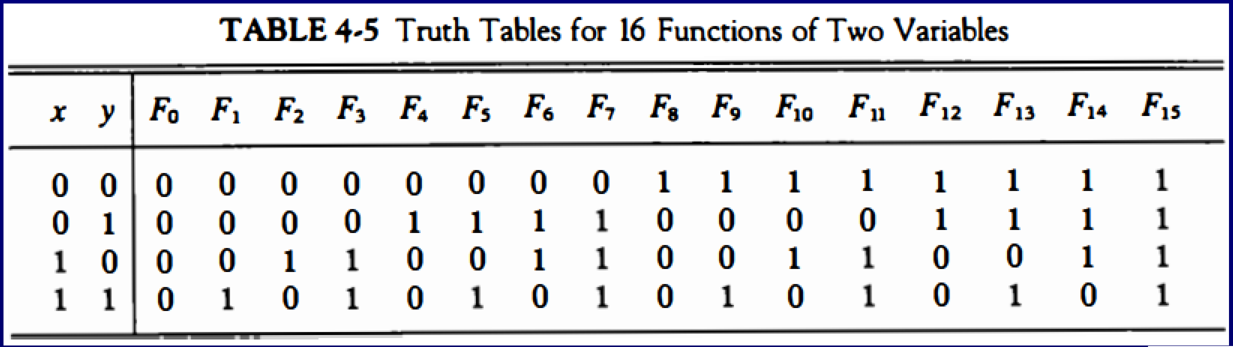


TABLE 4-5 Truth Tables for 16 Functions of Two Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

There are 16 different logic operations that can be performed with two binary variables. They can be determined from all possible truth tables obtained with two binary variables as shown in Table 4-5. The 16 Boolean functions of two variables x and y are expressed in algebraic form in the first column of Table 4-6.

x = 0011

y = 0101

Fx= - - - - (Boolean Function)

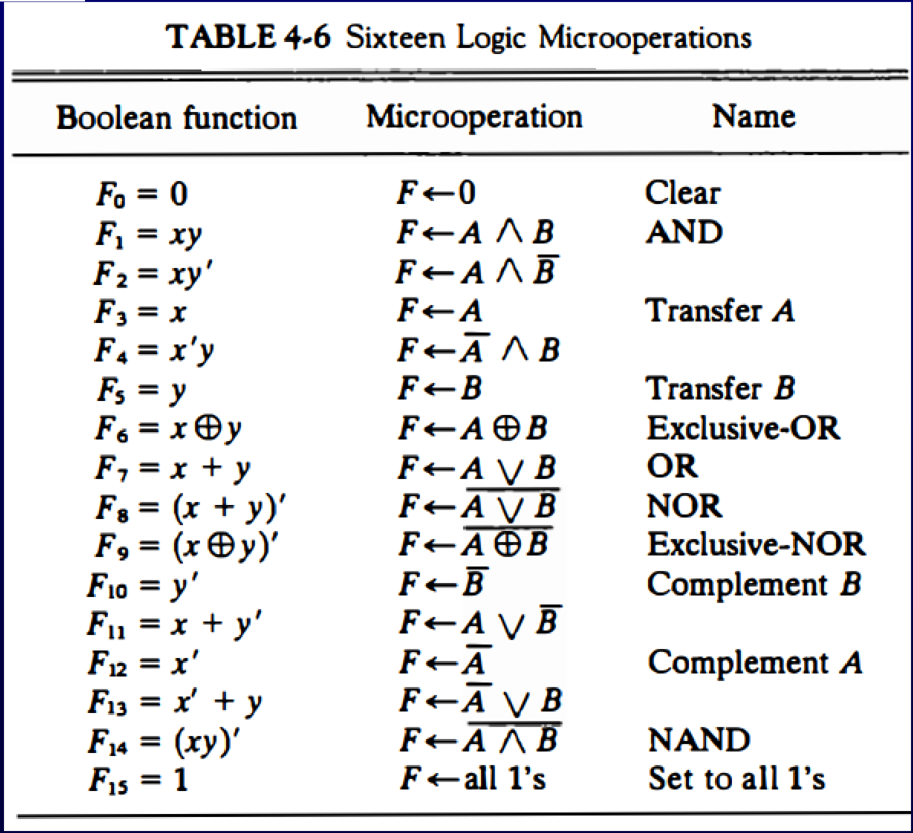


TABLE 4-6 Sixteen Logic Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

Shift Microoperations

Shift Microoperations:

2.4 Shift Microoperations: Logical Shift, Circular shift, Arithmetic Shift, Hardware Implementation of Shifter.

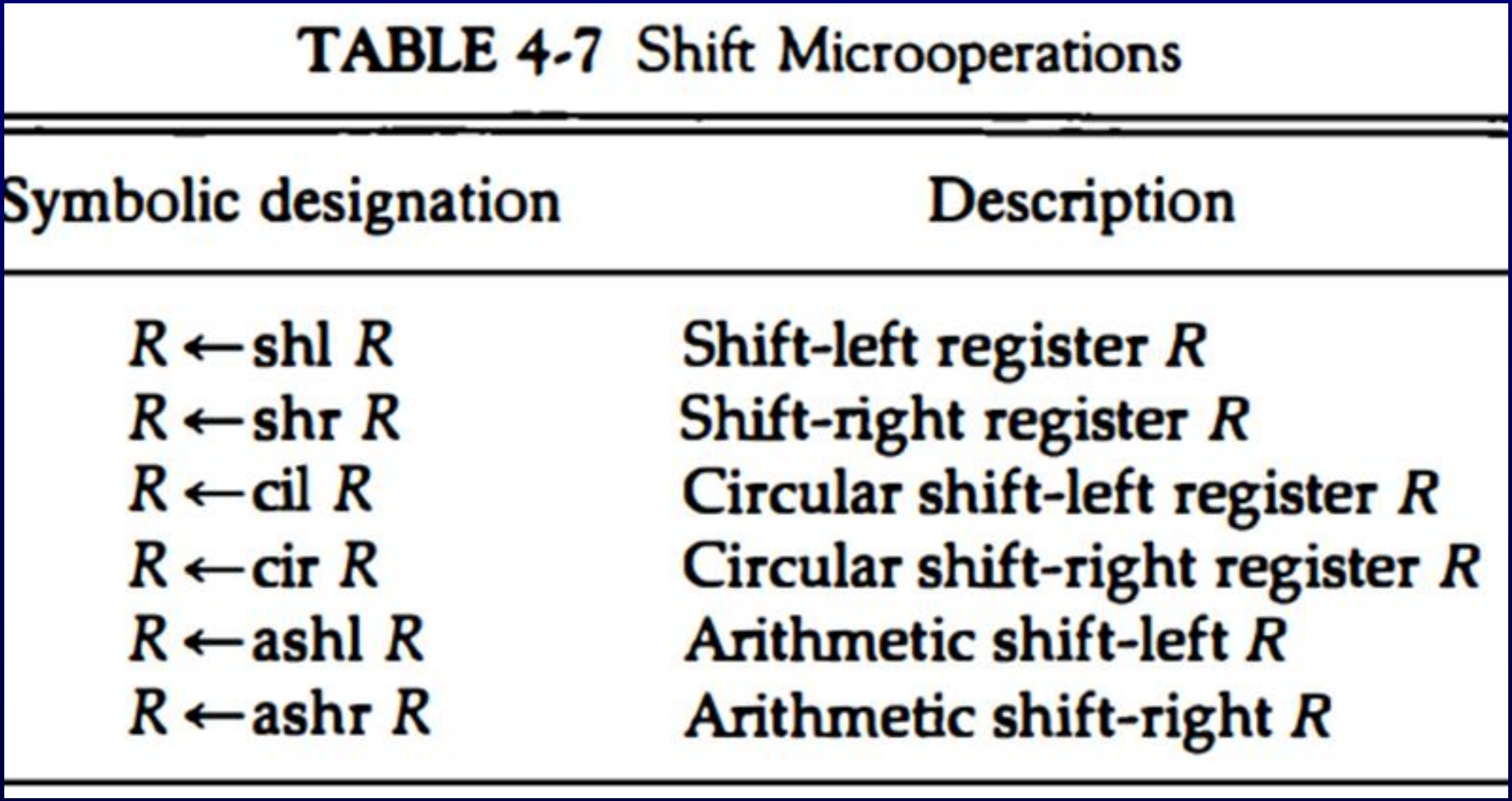


TABLE 4-7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Shift Microoperations

Shift Microoperations:

- Shift microoperations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic, and other data-processing operations.
- The contents of a register can be shifted to the left or the right. At the same time that the bits are shifted, the first flip-flop receives its binary information from the serial input. During a shift-left operation the serial input transfers a bit into the rightmost position. During a shift-right operation the serial input transfers a bit into the leftmost position.
- The information transferred through the serial input determines the type of shift.

There are three types of shifts:

1. logical, 2. circular, 3. arithmetic

Shift Microoperations

Shift Microoperations:

There are three types of shifts:

1. Logical shift microoperations
   a. Logical shift-left
   b. Logical shift-right
2. Circular shift microoperations
   a. Circular shift-left
   b. Circular shift-right
3. Arithmetic shift microoperations
   a. Arithmetic shift-left
   b. Arithmetic shift-right

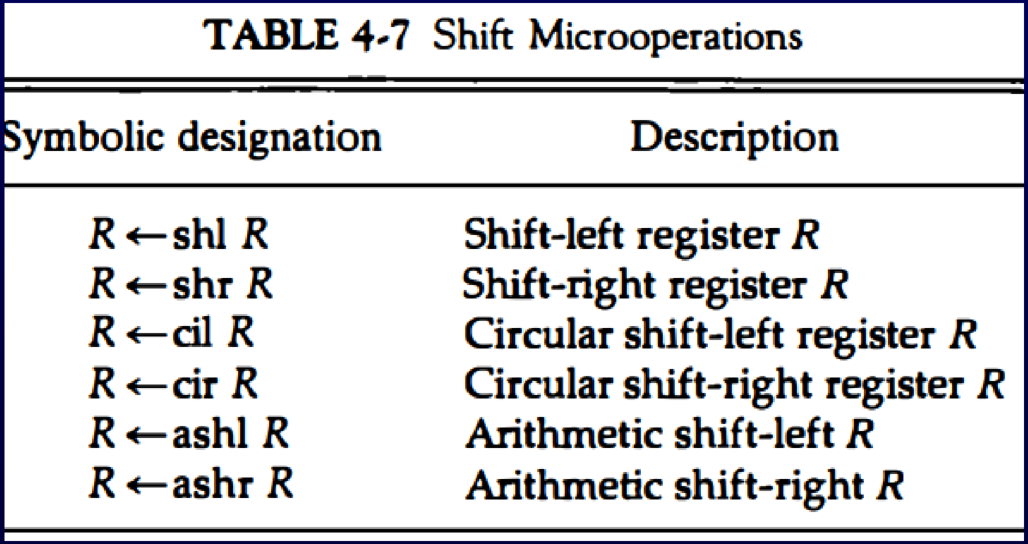


TABLE 4-7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Shift Microoperations

Shift Microoperations:

R1 ← shl R1

R2 ← shr R2

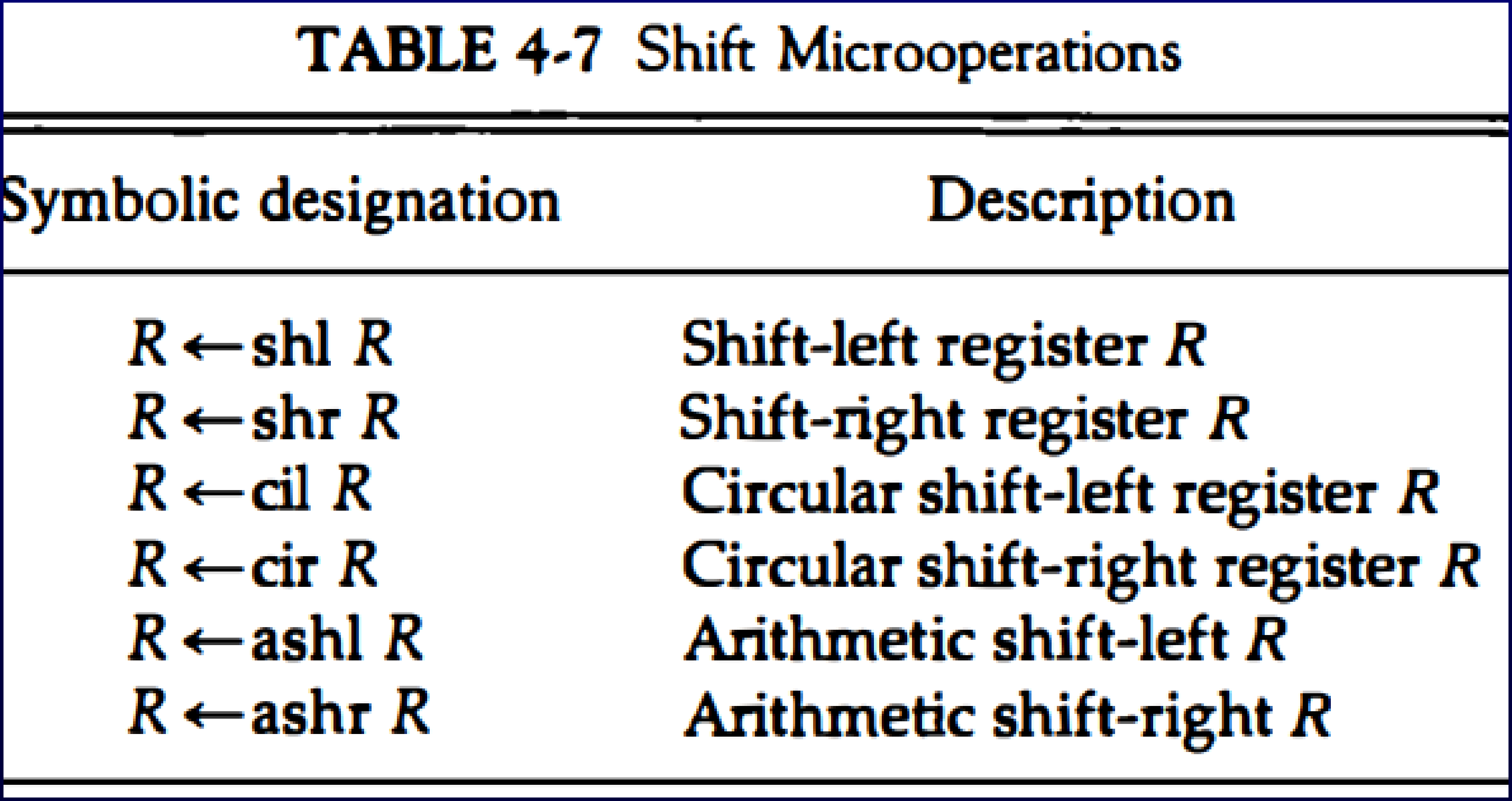
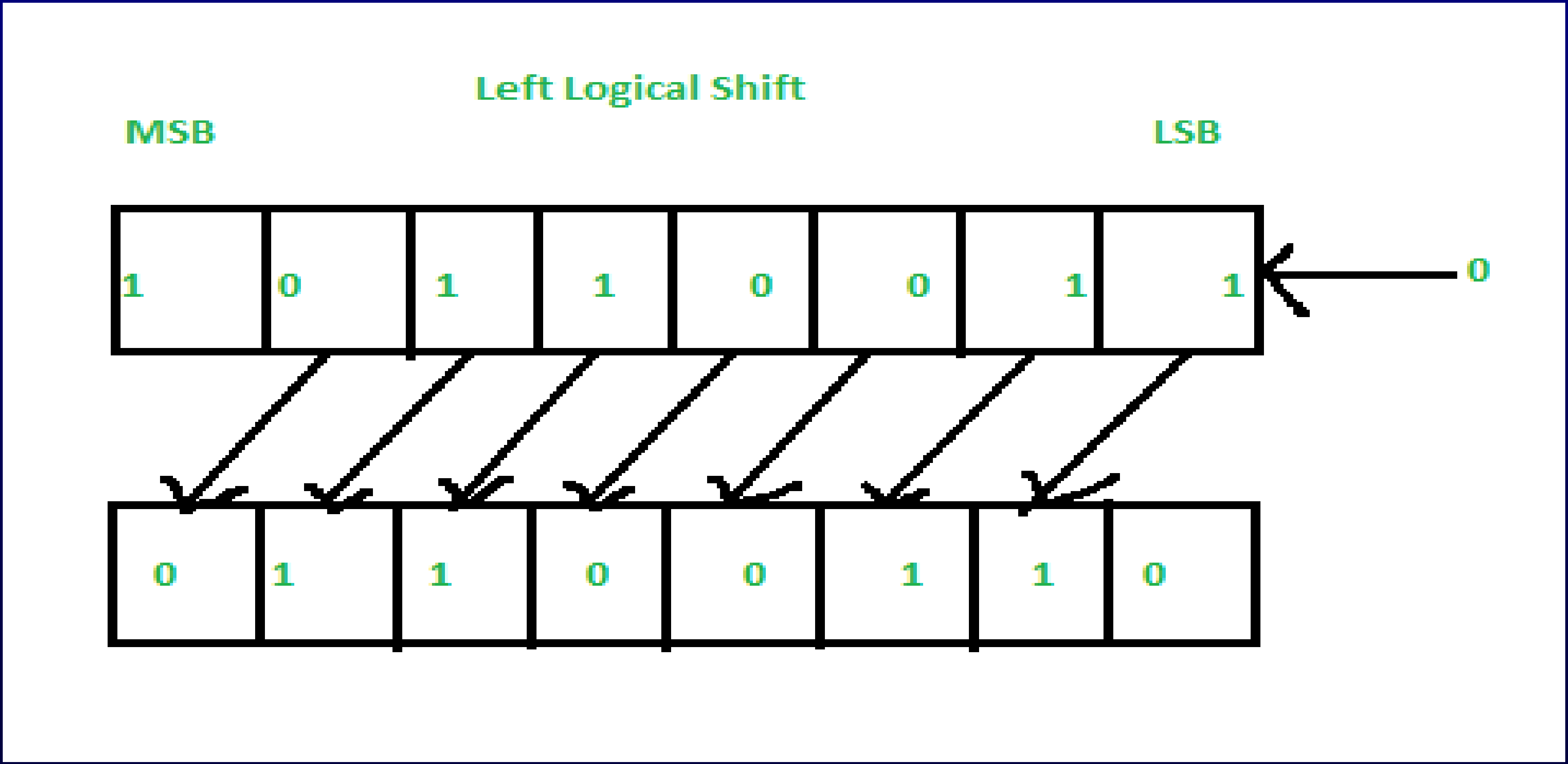


TABLE 4-7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

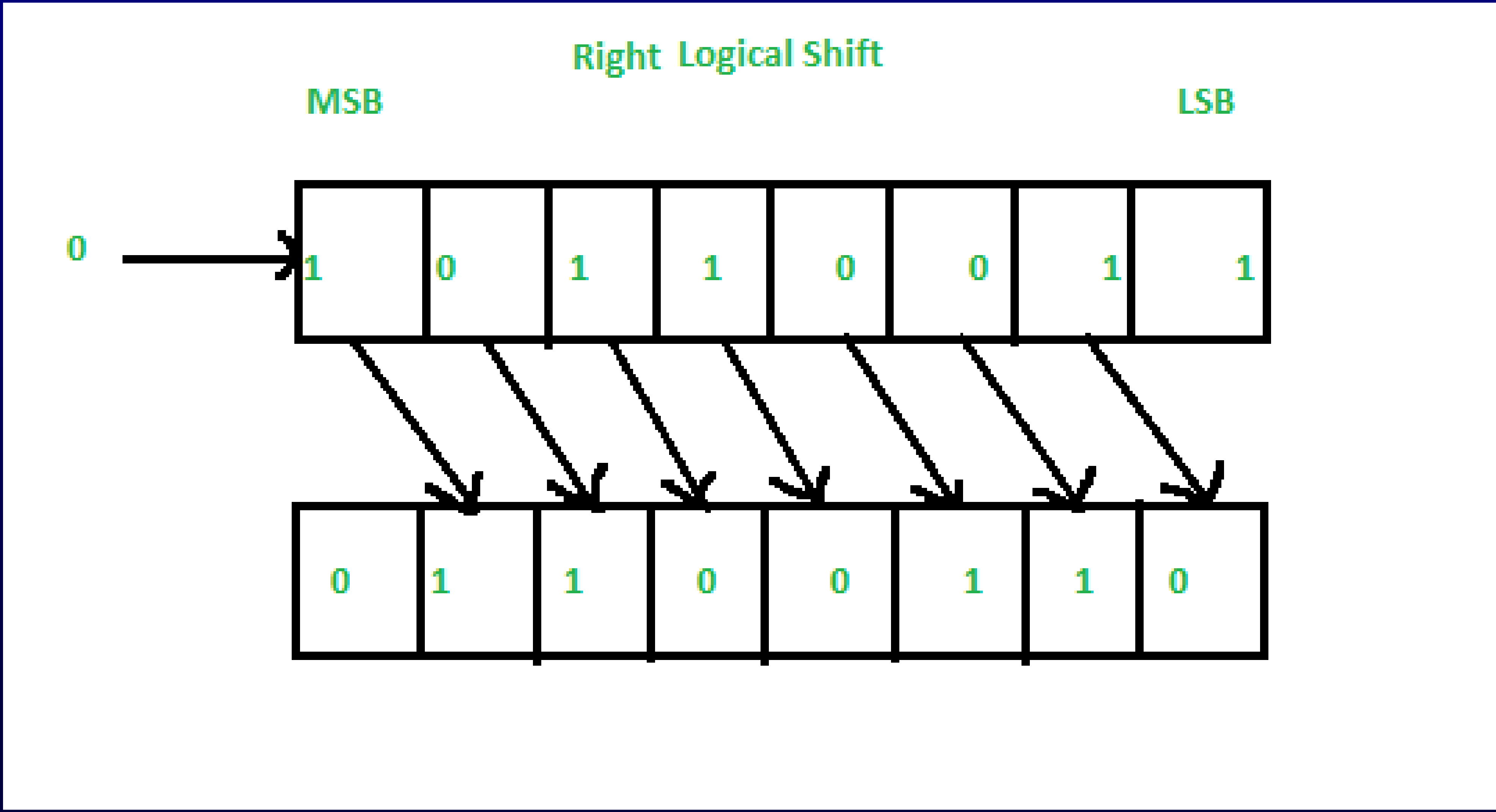
Shift Microoperations

Logical shift-left Microoperations:



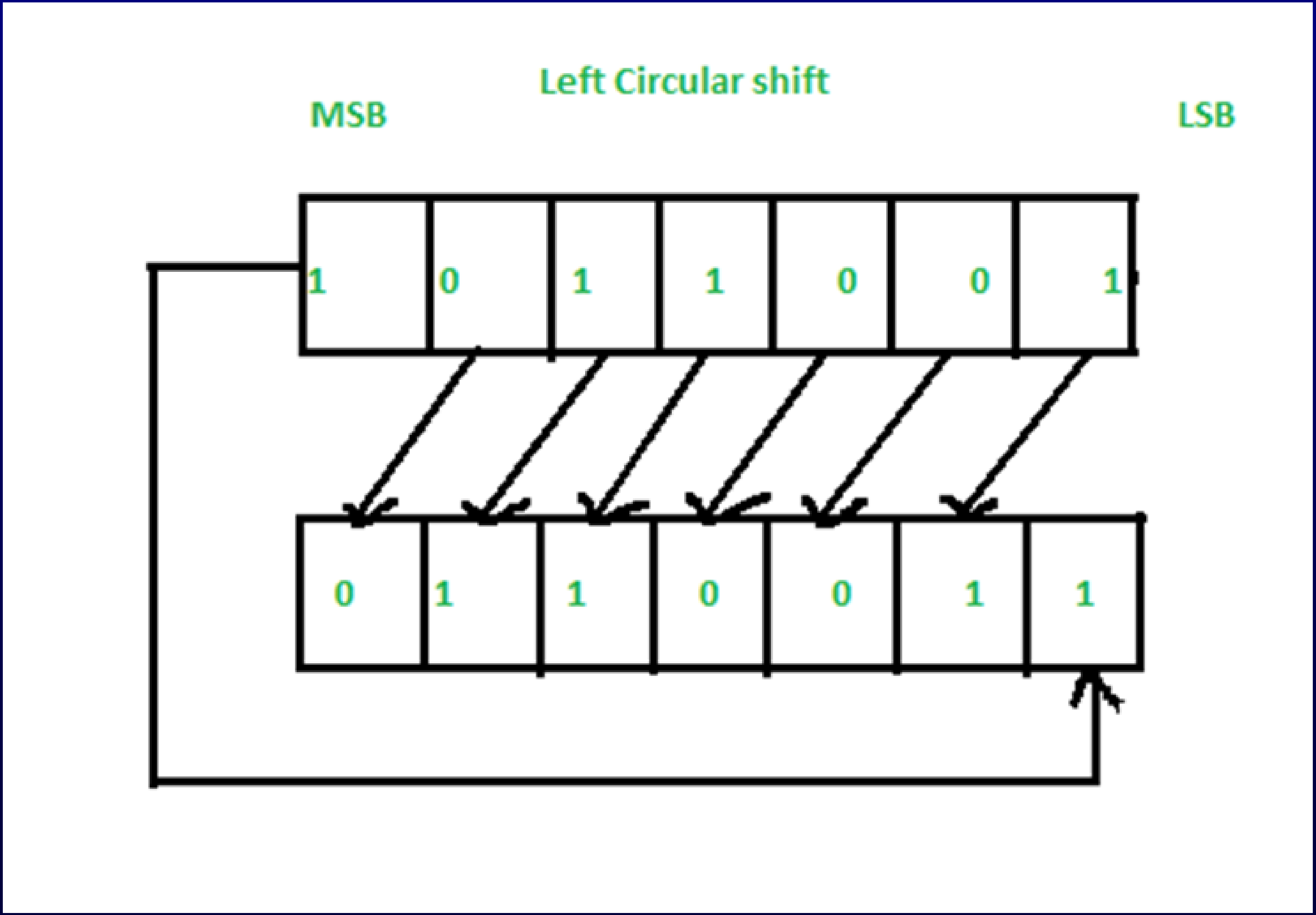
Shift Microoperations

Logical shift-right Microoperations:



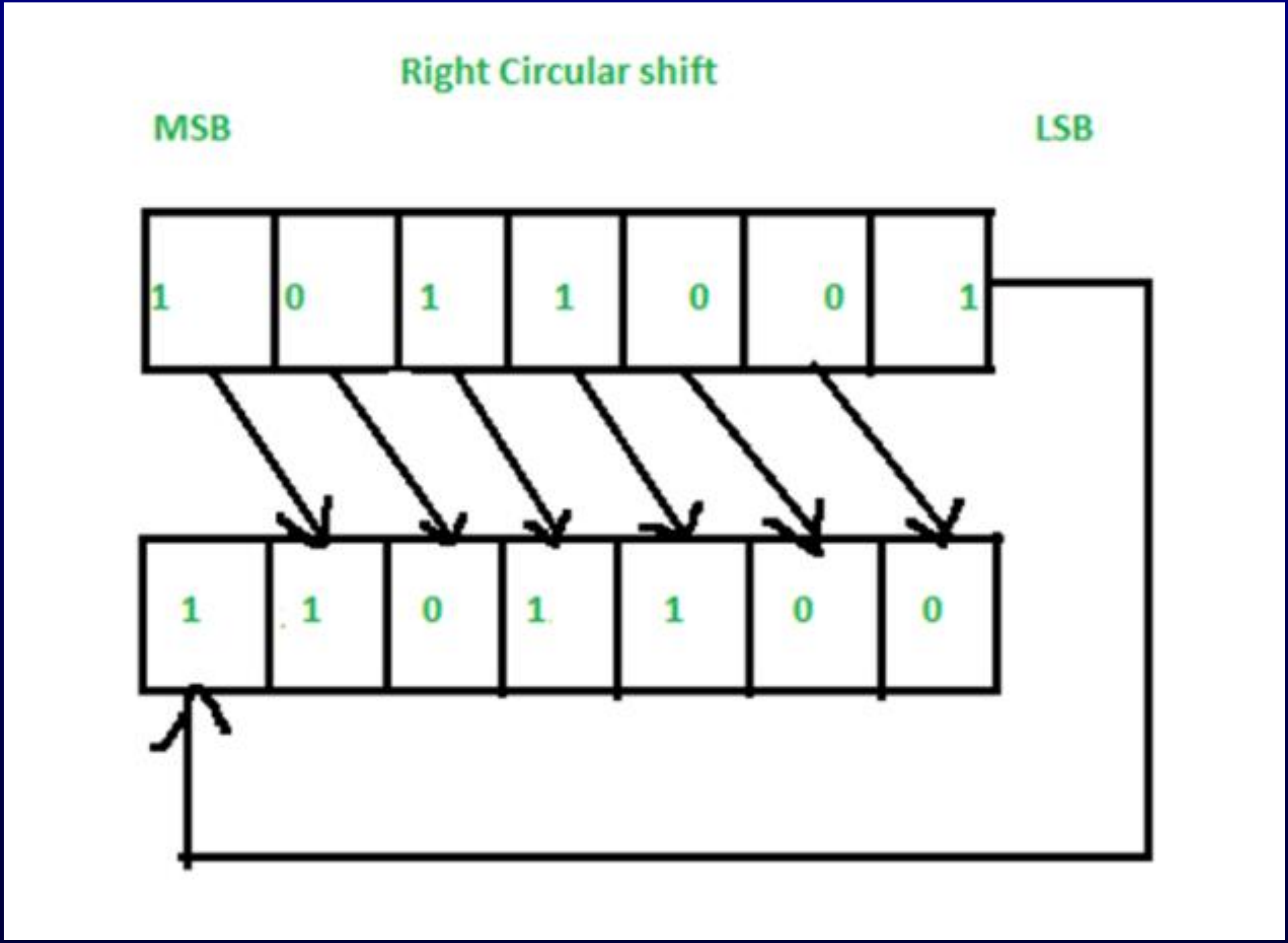
Shift Microoperations

Circular left shift Microoperations:



Shift Microoperations

Circular right shift Microoperations:



Shift Microoperations

Arithmetic shift Microoperations:

An arithmetic shift 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.

Shift Microoperations

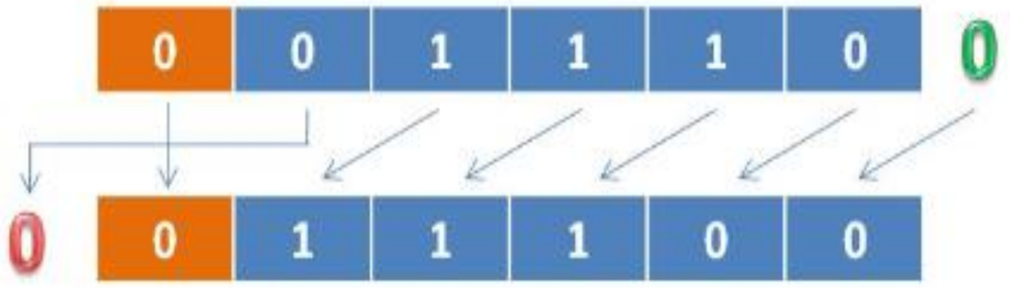
- **Arithmetic Shift Left Microoperations:** Every bit gets shifted towards left. The left most bit (MSB) represents the sign hence it doesn't participate in shifting process normally.

Ex.1-

R1 ← 14

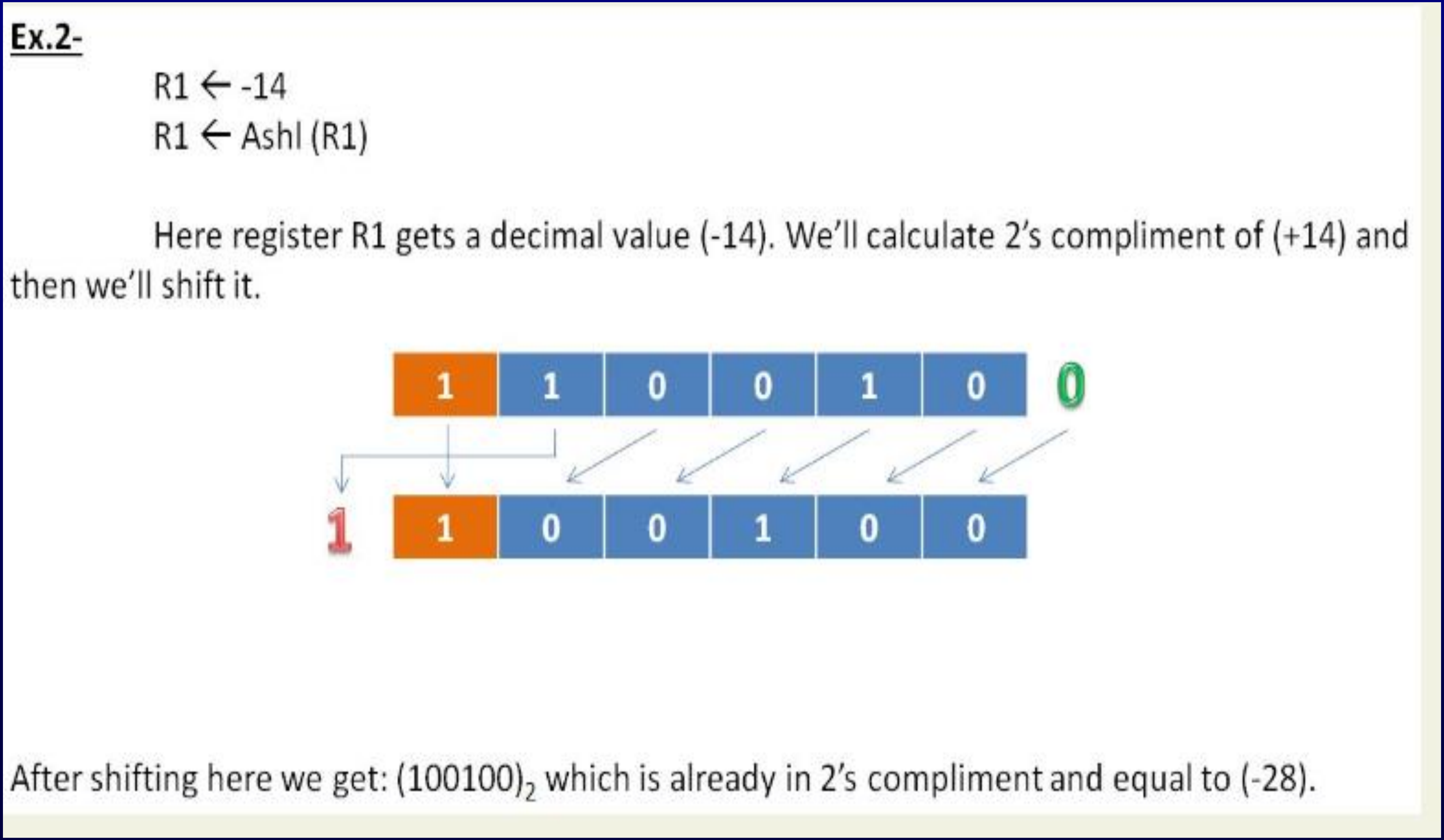
R1 ← Ashl (R1)

Here register R1 gets a decimal value 14. We can express it in not more than 4 bits. But we better is to use a relatively larger register. We'll use a 6-bit register.



After shifting here we get: $(011100)_2$ which is equal to $(28)_{10}$

Shift Microoperation



Ex.2-

R1 $\leftarrow$ -14

R1 $\leftarrow$ Ashl (R1)

Here register R1 gets a decimal value (-14). We'll calculate 2's compliment of (+14) and then we'll shift it.

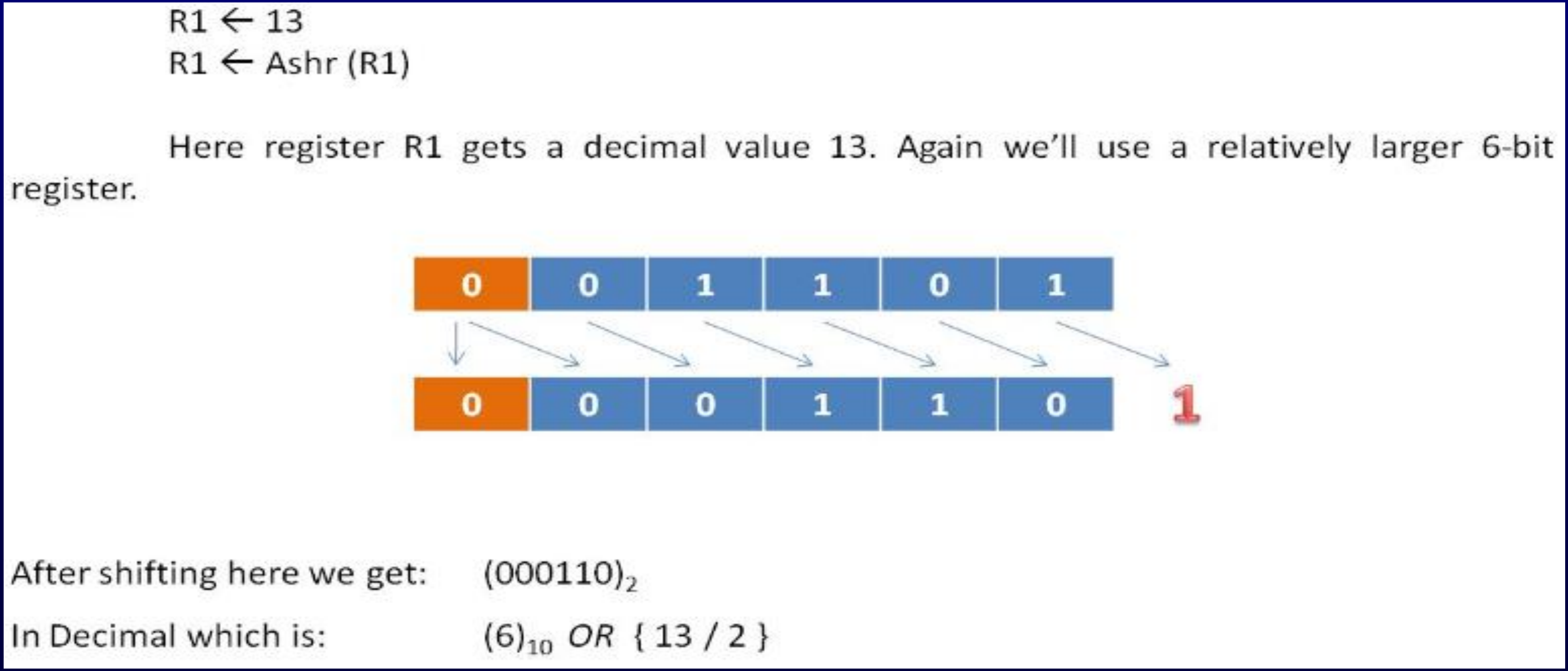
1 1 0 0 1 0
0
1
1 0 0 1 0 0

After shifting here we get: $(100100)_2$ which is already in 2's compliment and equal to (-28).

Shift Microoperation

Arithmetic Shift Right Microoperation

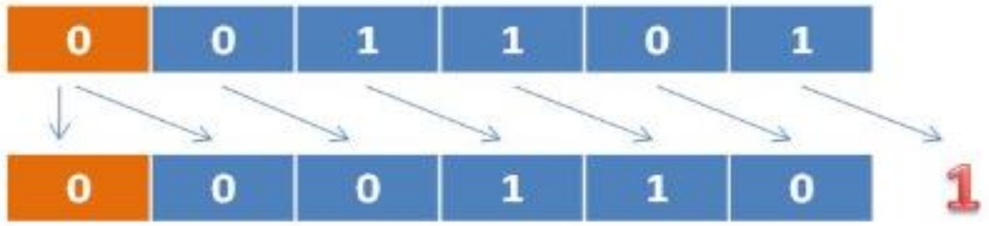
Every bit gets shifted towards right. The right most bit (LSB) gets lost and the sign bit is copied to the MSB-1 position. Again in this case, the sign bit remains the same.



R1 ← 13

R1 ← Ashr (R1)

Here register R1 gets a decimal value 13. Again we'll use a relatively larger 6-bit register.



After shifting here we get: $(000110)_2$

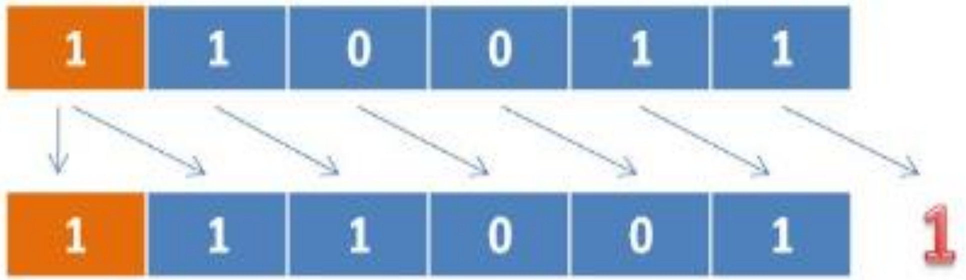
In Decimal which is: $(6)_{10}$ *OR* { 13 / 2 }

Shift Microoperation

R1 ← -13

R1 ← Ashr (R1)

Here register R1 gets a decimal value (-13). We'll calculate 2's compliment of (+13) anc
then we'll shift it.



After shifting here we get: $(111001)_2$ which is already in 2's compliment and equal to (-7).

NOTE: Every negative number is taken already in 2's compliment representation.