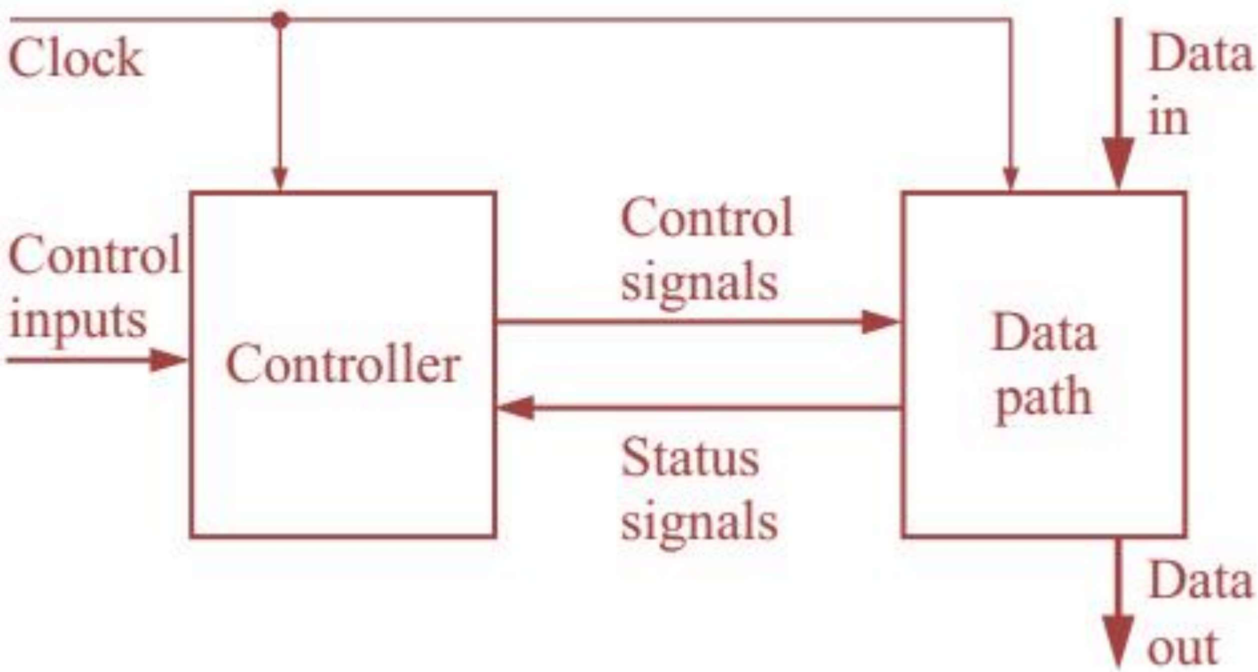
به نام خدا

زبان انتقال ثبات

RTL

Dr. Aref Karimiafshar
A.karimiafshar@iut.ac.ir

CPU



Clock
Data
in
Control
inputs
Controller
Control
signals
Status
signals
Data
path
Data
out

Register Transfer Language (RTL)

- Digital System: An interconnection of hardware modules that do a certain task on the information.
- Registers + Operations performed on the data stored in them = Digital Module
- Modules are interconnected with common data and control paths to form a digital computer system

Register Transfer Language cont.

- Microoperations: operations executed on data stored in one or more registers.
- For any function of the computer, a sequence of microoperations is used to describe it
- The result of the operation may be:
  - replace the previous binary information of a register
  - transferred to another register

101101110011 $\xrightarrow{\text{Shift Right Operation}}$ 010110111001

Register Transfer Language cont.

- The internal hardware organization of a digital computer is defined by specifying:
  - The set of registers it contains and their function
  - The sequence of microoperations performed on the binary information stored in the registers
  - The control that initiates the sequence of microoperations
- Registers + Microoperations Hardware + Control Functions = Digital Computer

Register Transfer Language cont.

- Register Transfer Language (RTL) : a symbolic notation to describe the microoperation transfers among registers

Next steps:

  - Define symbols for various types of microoperations,
  - Describe the hardware that implements these microoperations

Register Transfer (our first microoperation)

- Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register
    - R1: processor register
    - MAR: Memory Address Register (holds an address for a memory unit)
    - PC: Program Counter
    - IR: Instruction Register
    - SR: Status Register

Register Transfer cont.

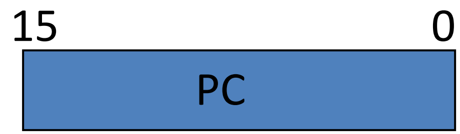
- The individual flip-flops in an n-bit register are numbered in sequence from 0 to n-1 (from the right position toward the left position)

Register R1 Showing individual bits

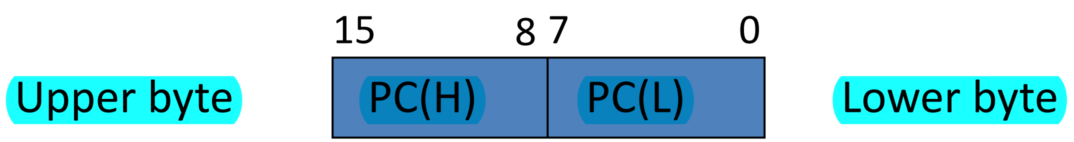
A block diagram of a register

Register Transfer cont.

Other ways of drawing the block diagram of a register:



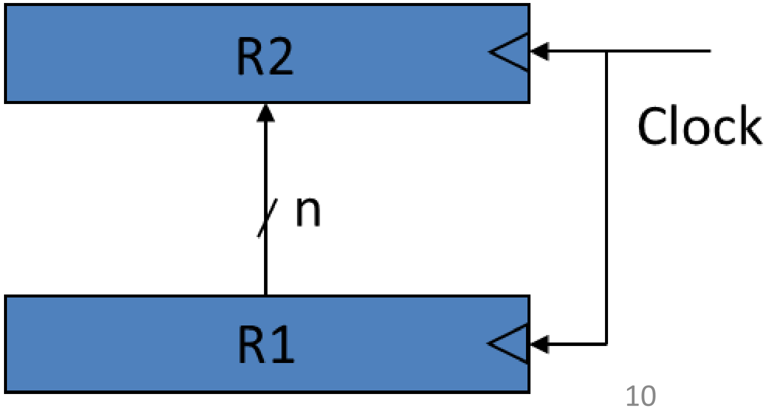
Numbering of bits



Partitioned into two parts

Register Transfer cont.

- Information transfer from one register to another is described by a *replacement operator:* $R2 \leftarrow R1$
  - This statement denotes a transfer of the content of register R1 into register R2
- The transfer happens in one clock cycle
- 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
- We are assuming that the 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



Register Transfer cont.

- Conditional transfer occurs only under a control condition $If\ (P=1)\ then\ (R2 \leftarrow R1)$
- Representation of a (conditional) transfer

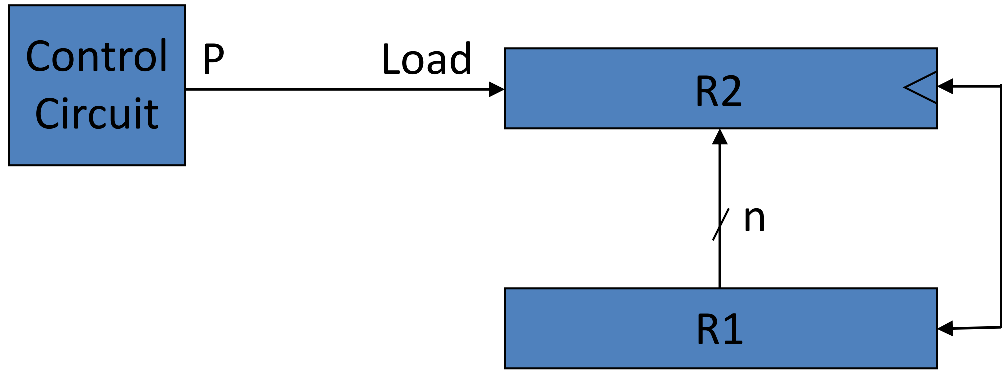
  P: $\qquad R2 \leftarrow R1$
- A binary condition (P equals to 0 or 1) determines when the transfer occurs

- The content of R1 is transferred into R2 only if P is 1

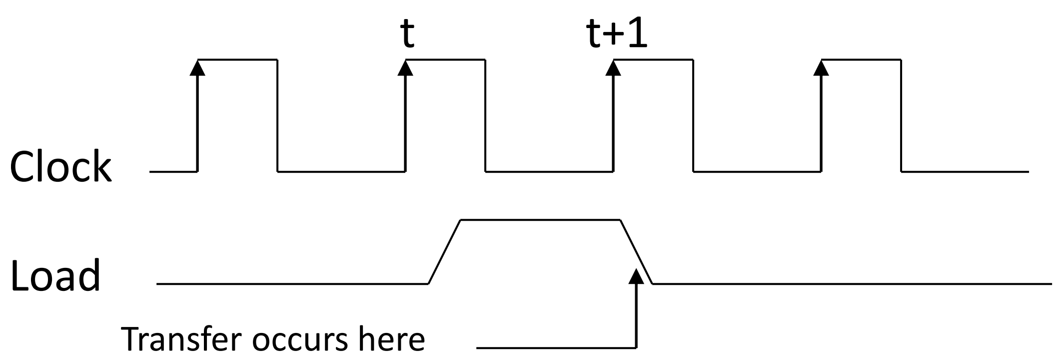
Register Transfer cont.

Hardware implementation of a controlled transfer: $P: R2 \leftarrow R1$

Block diagram:



Timing diagram



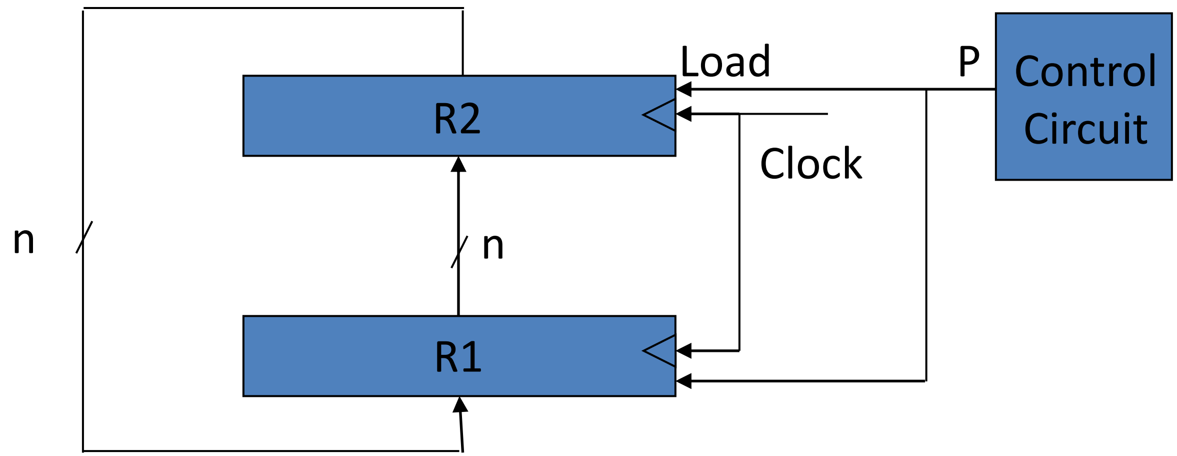
Synchronized with the clock

Register Transfer cont.

Example:

$P: R1 \leftarrow R2, R2 \leftarrow R1$

Block diagram:



Example:

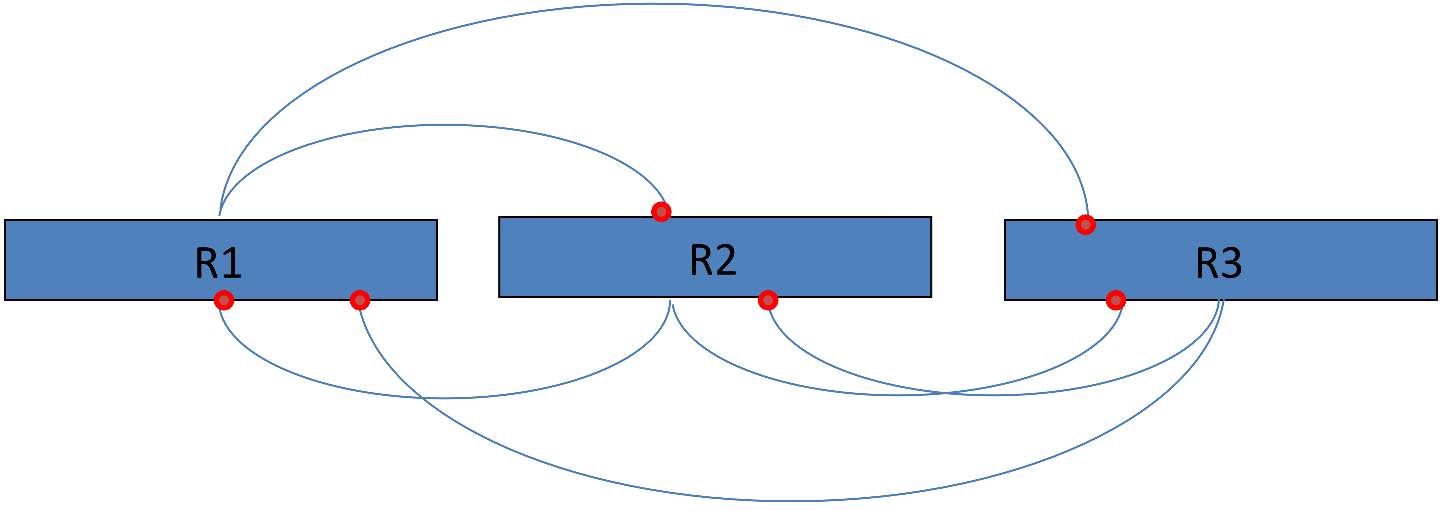
$P: R1 \leftarrow 0, R2 \leftarrow R1$

$P: R1 \leftarrow 0, R1 \leftarrow R2$ ✗

Register Transfer cont.

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

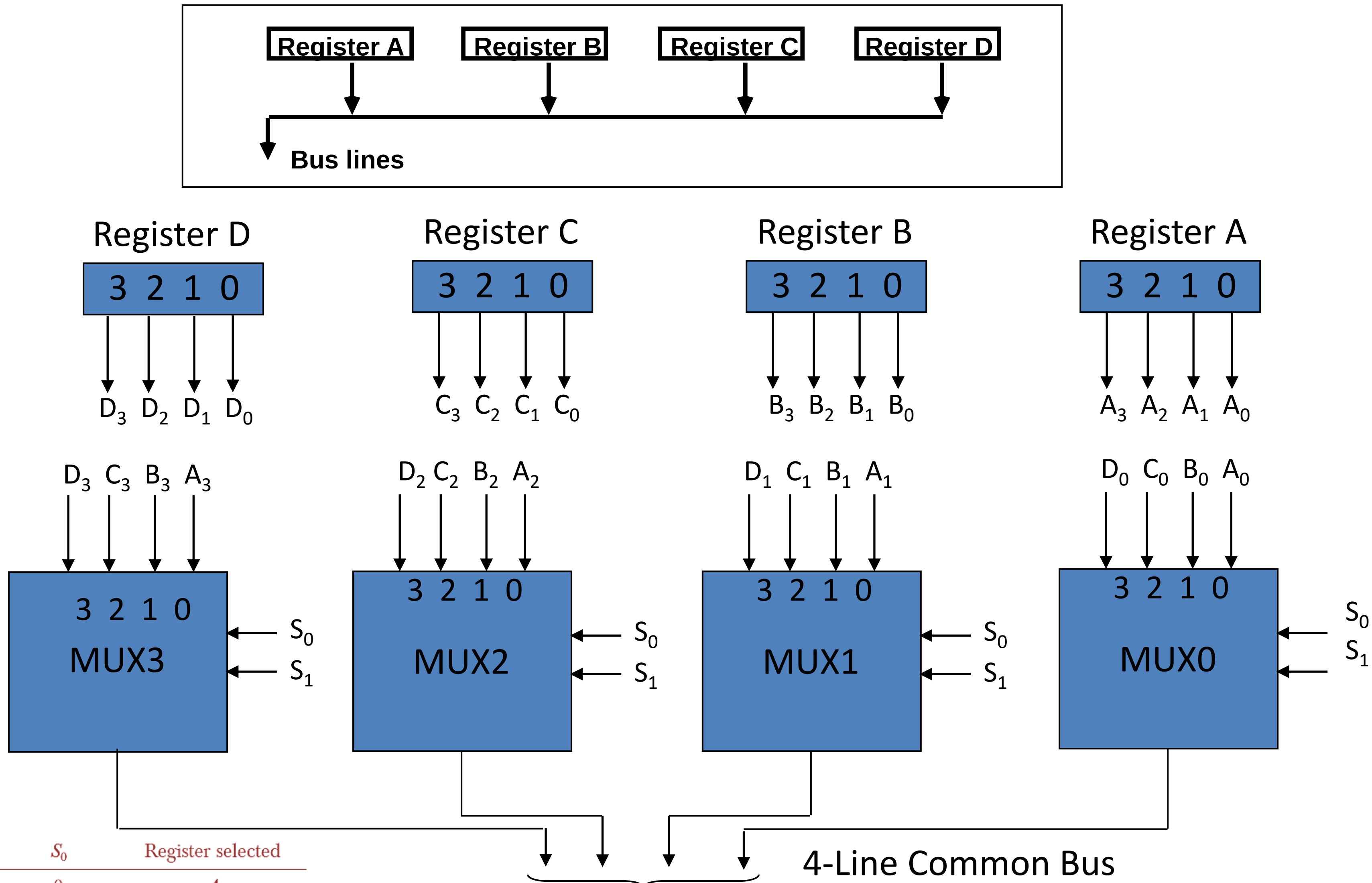


n (n-1)

Bus and Memory Transfers cont.

- 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: 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

Bus and Memory Transfers cont.



S_1	S_0	Register selected
0	0	*A*
0	1	*B*
1	0	*C*
1	1	*D*

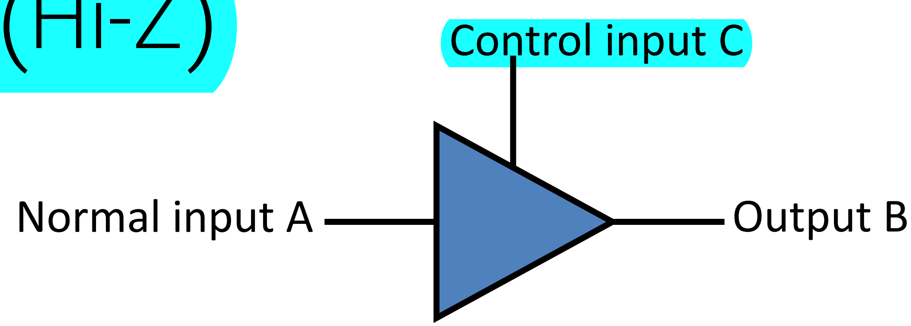
Bus and Memory Transfers cont.

- The transfer of information from a bus into one of many destination registers is done:
  - By connecting the bus lines to the inputs of all destination registers and then:
  - activating the load control of the particular destination register selected
- We write: $R2 \leftarrow C$ to symbolize that the content of register C is *loaded into* the register R2 using the common system bus
- It is equivalent to: $BUS \leftarrow C$, (select C)

  $R2 \leftarrow BUS$ (Load R2)

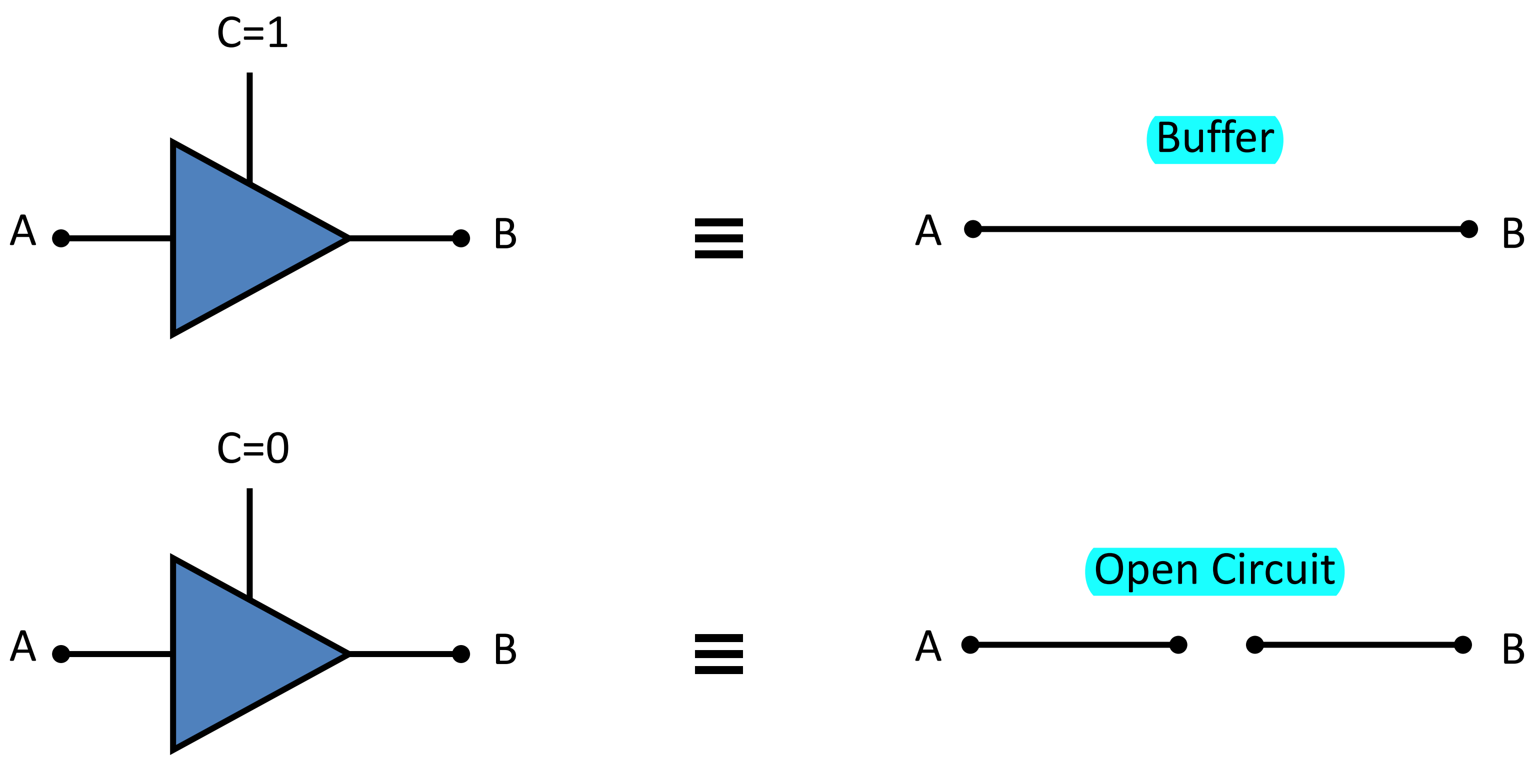
Bus and Memory Transfers: Three-State Bus Buffers

- A bus system can be constructed with three-state buffer gates instead of multiplexers
- A three-state buffer is a digital circuit that exhibits three states: logic-0, logic-1, and high-impedance (Hi-Z)

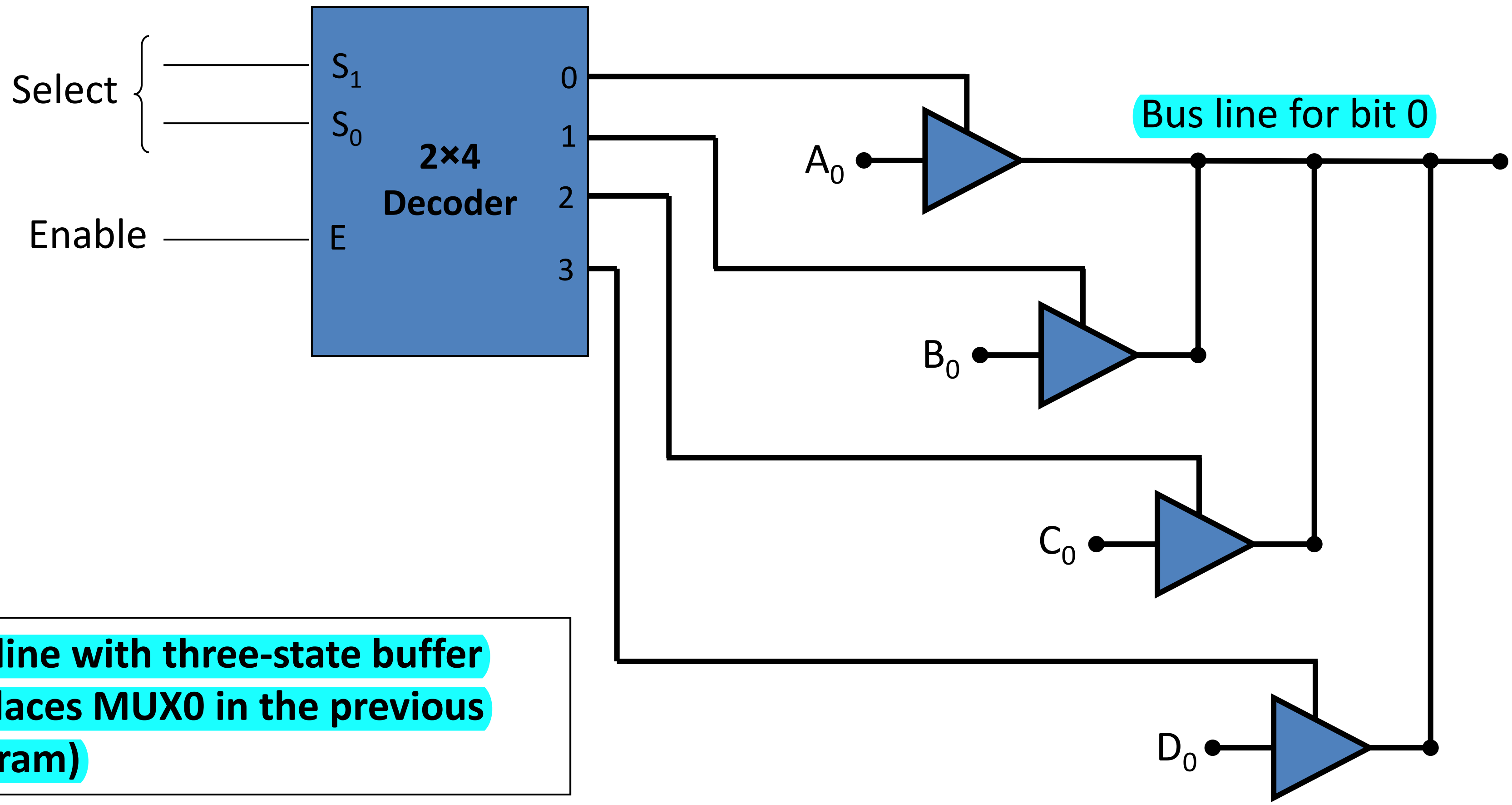


Three-State Buffer

Bus and Memory Transfers: Three-State Bus Buffers cont.



Bus and Memory Transfers: Three-State Bus Buffers cont.



Bus line with three-state buffer (replaces MUX0 in the previous diagram)

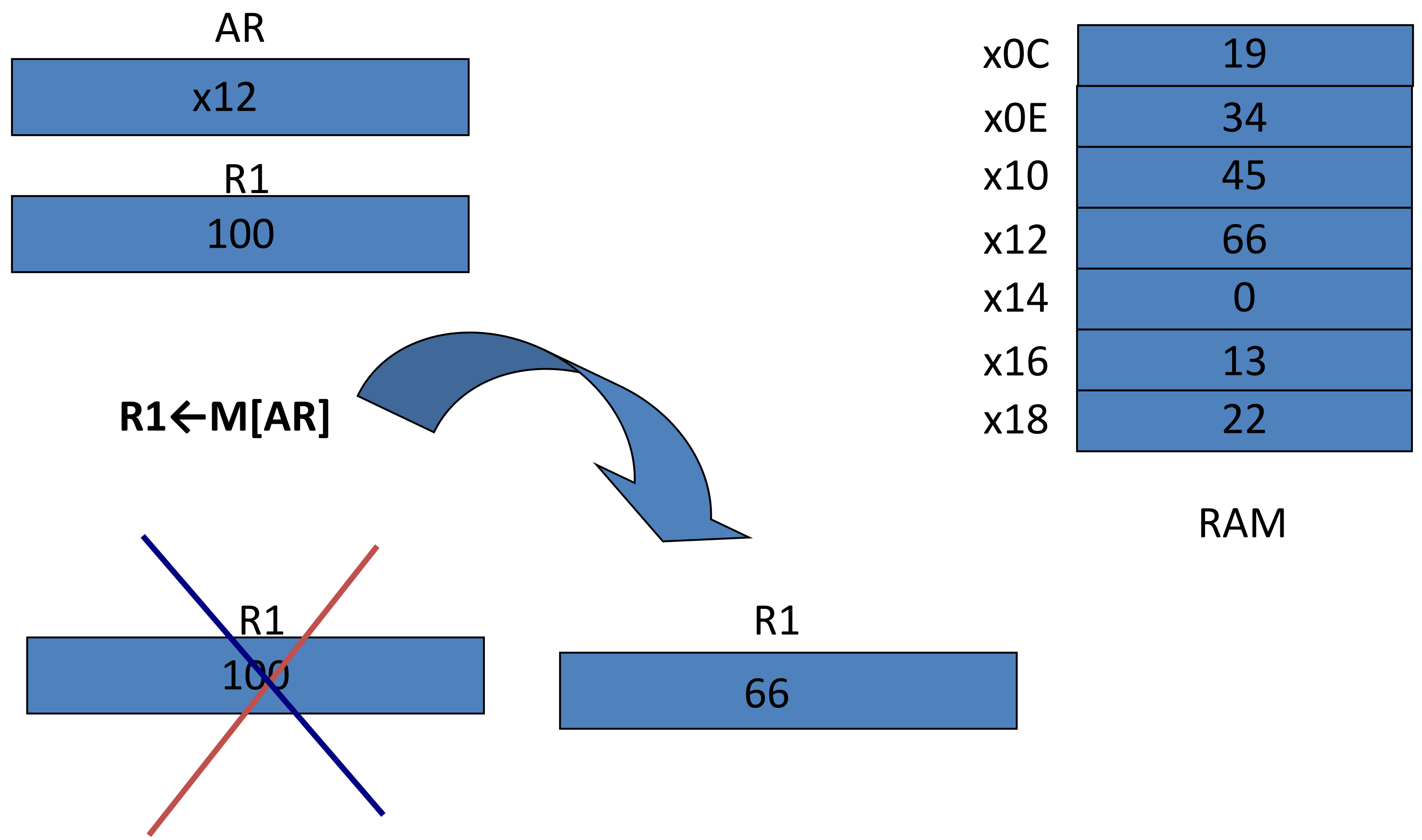
Bus and Memory Transfers

- Memory read : Transfer from memory
- Memory write : Transfer to memory
- Data being read or wrote is called a memory word (called M)
- It is necessary to specify the address of M when writing /reading memory
- This is done by enclosing the address in square brackets following the letter M
- Example: M[0016] : the memory contents at address 0x0016

Bus and Memory Transfers cnt.

- Assume that the address of a memory unit is stored in a register called the Address Register AR
- Lets represent a Data Register with DR, then:
    - Read: DR $\leftarrow$ M[AR]
    - Write: M[AR] $\leftarrow$ DR

Bus and Memory Transfers cnt.



Arithmetic Microoperations

- The microoperations most often encountered in digital computers are classified into four categories:
  - Register transfer microoperations
  - Arithmetic microoperations (on numeric data stored in the registers)
  - Logic microoperations (bit manipulations on non-numeric data)
  - Shift microoperations
- Register transfer microoperation
  - Does not change the information content
  - Other three types of microoperations change the information content

Arithmetic Microoperations cont.

- The basic arithmetic microoperations are: addition, subtraction, increment, decrement, and shift
- Addition Microoperation:

$$R3 \leftarrow R1+R2$$

- Subtraction Microoperation:

$$R3 \leftarrow R1-R2 \text{ or :}$$

1's complement

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

Arithmetic Microoperations cont.

- One's Complement Microoperation:

$$R2 \leftarrow \overline{R2}$$

- Two's Complement Microoperation:

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

- Increment Microoperation:

$$R2 \leftarrow R2+1$$

- Decrement Microoperation:

$$R2 \leftarrow R2-1$$

Arithmetic Microoperations cont.

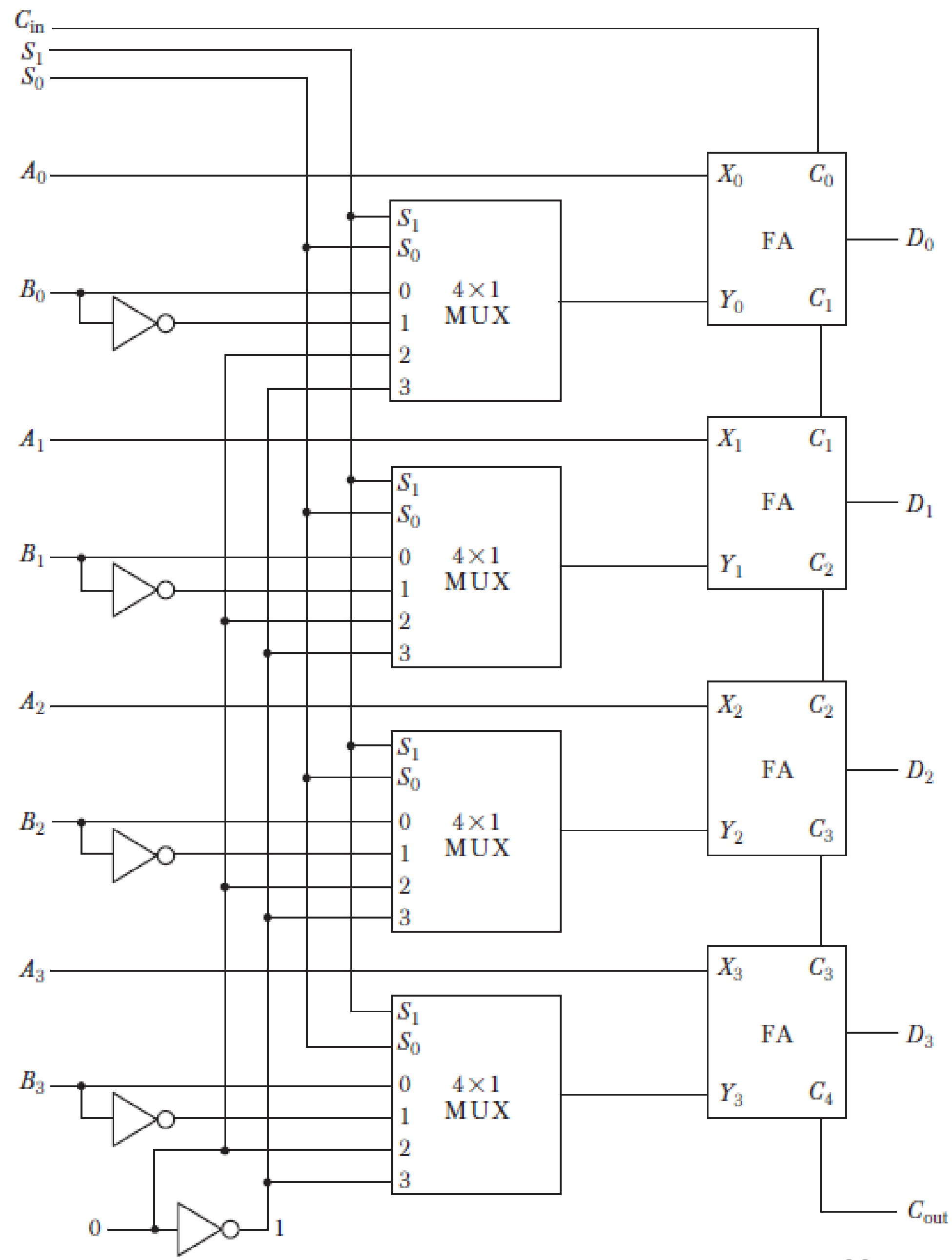
Arithmetic Microoperations

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

Hardware Implementation Arithmetic Microoperations

Arithmetic Circuit Function Table

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A



Logic Microoperations

- Logic microoperations specify binary operations for strings of bits stored in registers
  - These operations consider each bit of the register separately and treat them as binary variables
- Example:
  - Exclusive-or microoperation with the contents of two registers R1 and R2 is symbolized by

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

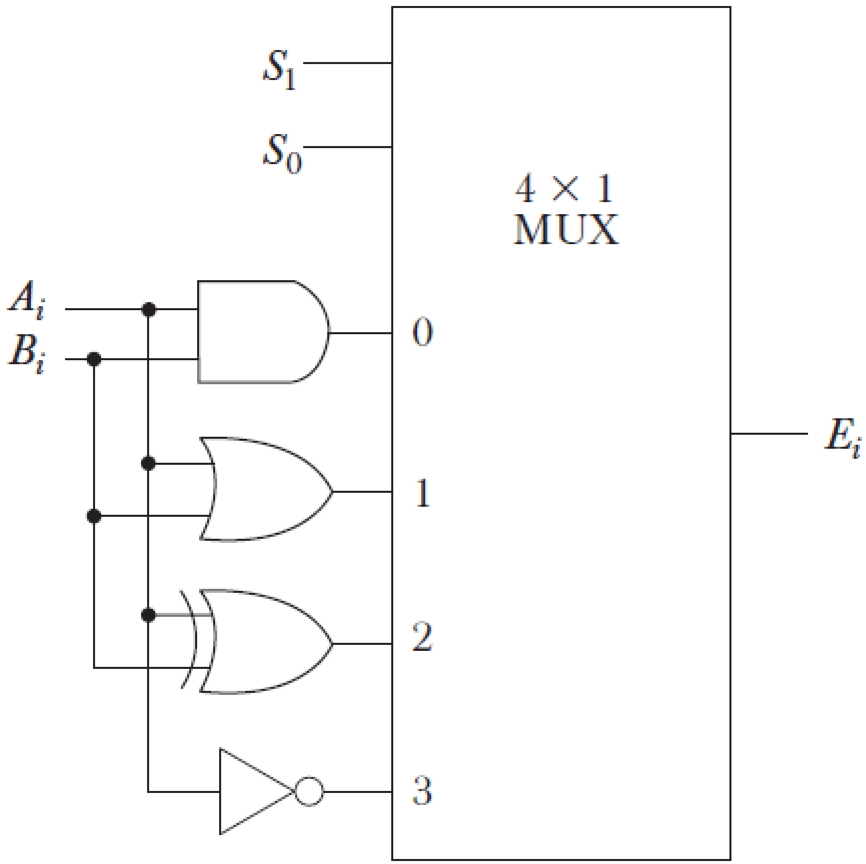
Logic Microoperations

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

Hardware Implementation Logic Microoperations

One stage of logic circuit.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Functional table

Shift Microoperations

- Shift microoperations are used for serial transfer of data
  - Also used in conjunction with arithmetic, logic, and other data-processing operations
- Contents of a register can be shifted to the left or the right
  - At the same time, the first flip-flop receives its binary information from the serial input

$$R1 \leftarrow \text{shl } R1$$

$$R2 \leftarrow \text{shr } R2$$

Shift Microoperations cnt.

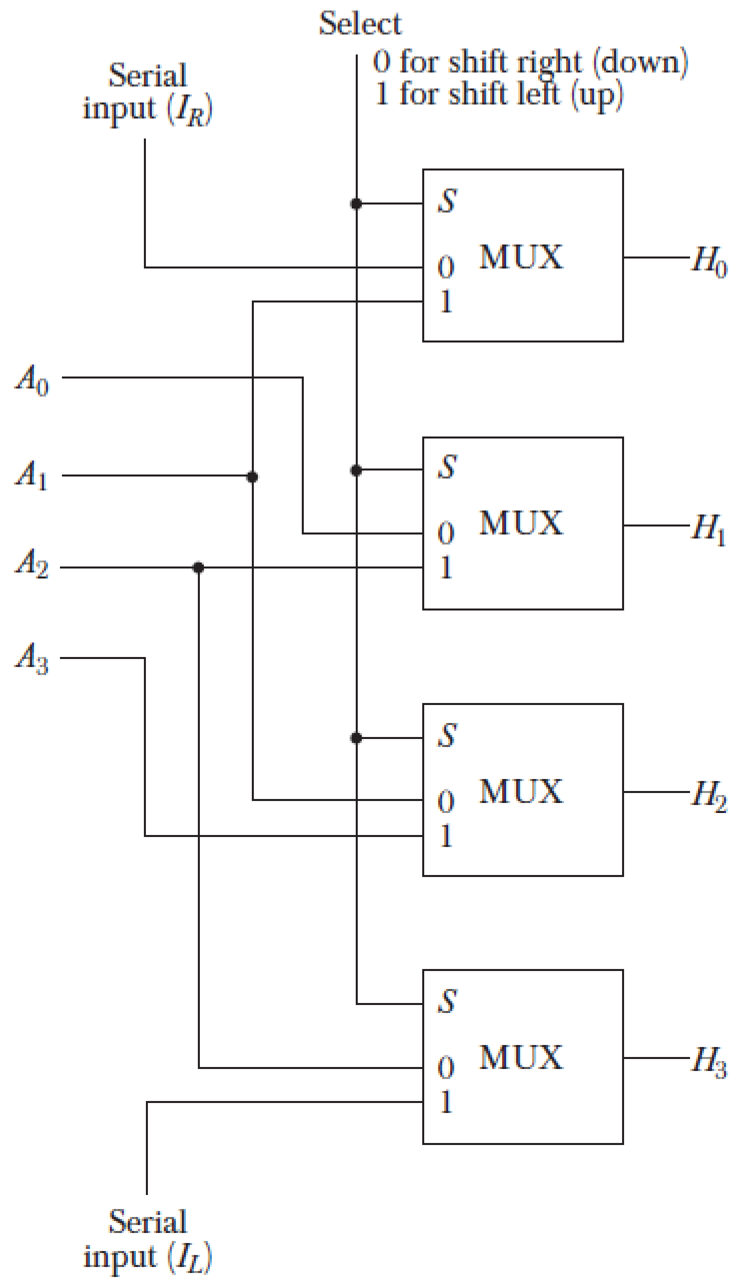
Shift Microoperations

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

Hardware Implementation Shift Microoperations

Functional table

Select	Output			
S	H_0	H_1	H_2	H_2
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L



پایان

موفق و پیروز باشید