RTL & Micro-Operations

Mano Ch. 4

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
- 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

Register Transfer Language $^{\text{cont.}}$

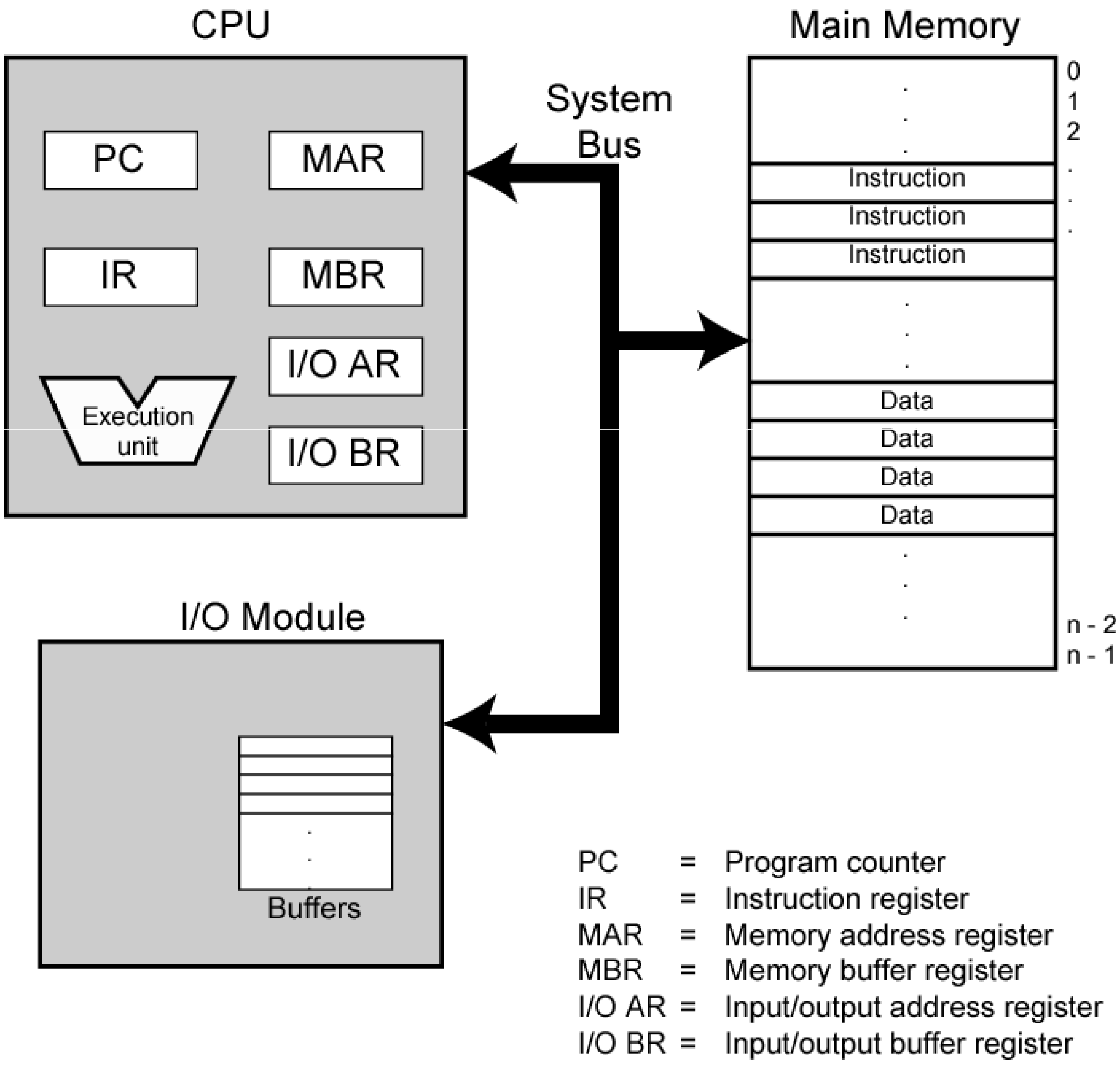
- The internal hardware organization of a digital computer is defined by specifying:
  - The set of registers it contains and their function
  - The sequence of microoperations performed on the binary information stored in the registers
  - The control that initiates the sequence of microoperations

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

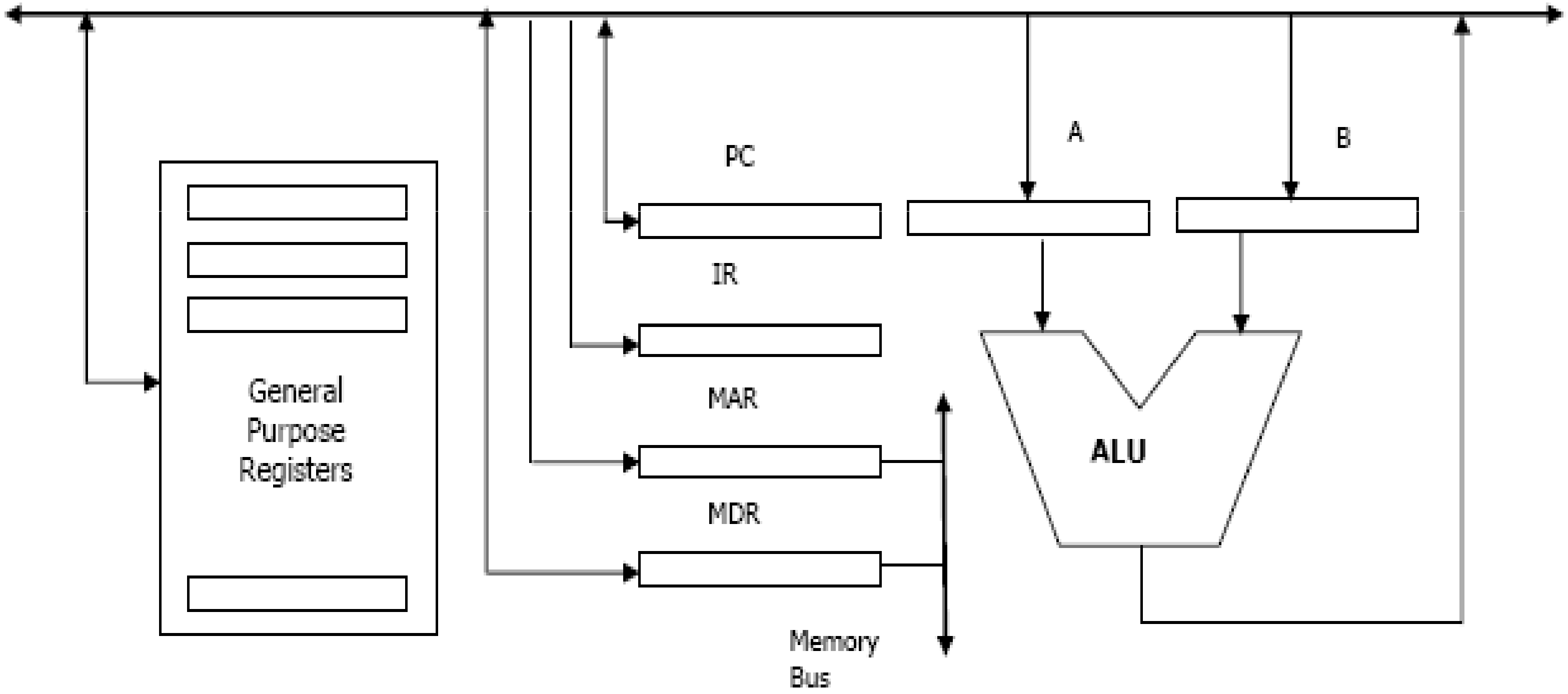
Computer Components: Top Level View



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

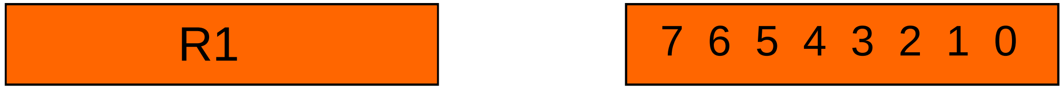
CPU local Bus Organization

- One-Bus Organization



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)



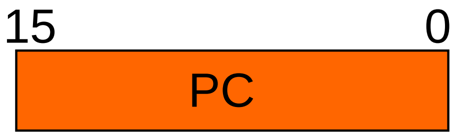
Register R1

Showing individual bits

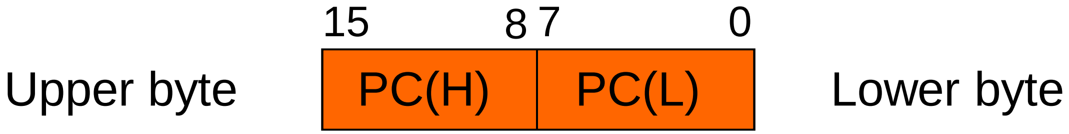
A block diagram of a register

Register Transfer cont.

Other ways of drawing the block diagram of a register:



Numbering of bits



Partitioned into two parts

Register Transfer $^{\text{cont.}}$

- Information transfer from one register to another is described by a *replacement operator:*

$$\mathbf{R2} \leftarrow \mathbf{R1}$$

  - 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 $^{\text{cont.}}$

- Conditional transfer occurs only under a control condition
- Representation of a (conditional) transfer

$$P: \qquad R2 \leftarrow R1$$

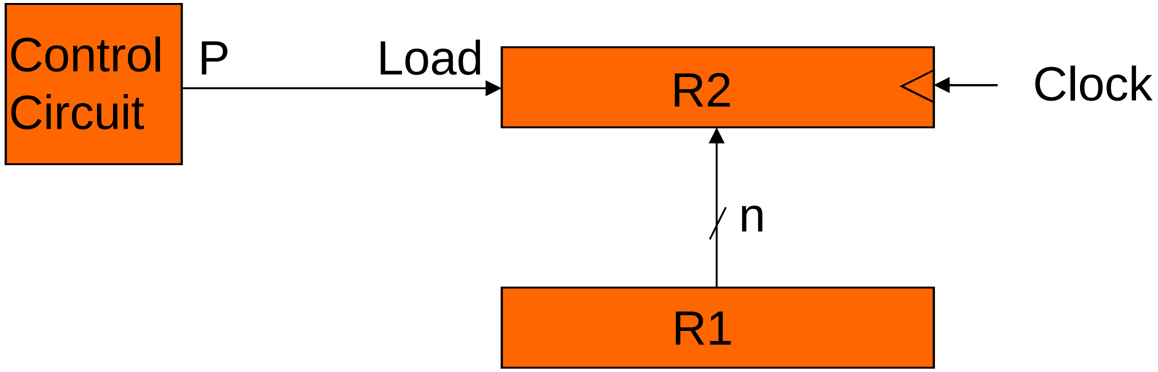
- A binary condition (P equals to 0 or 1) determines when the transfer occurs
- The content of R1 is transferred into R2 only if P is 1

Register Transfer cont.

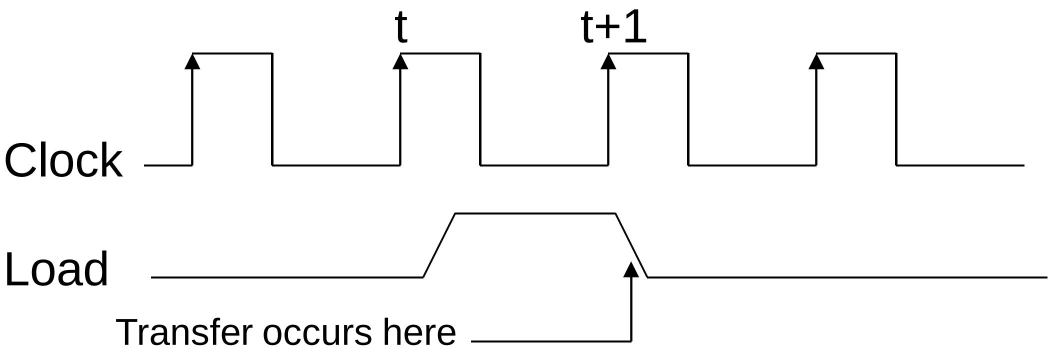
Hardware implementation of a controlled transfer:

P: R2 ← R1

Block diagram:



Timing diagram



Synchronized with the clock

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

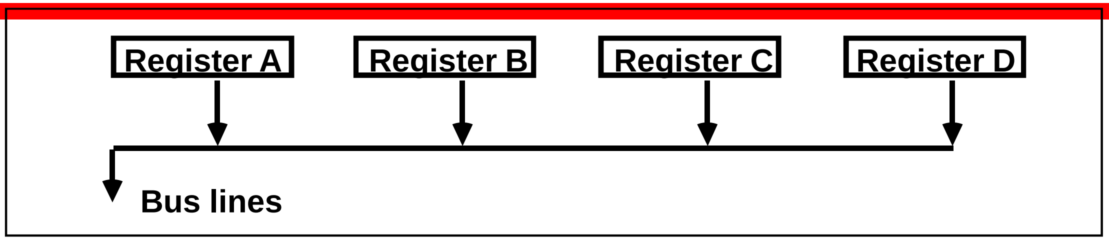
- Paths must be provided to transfer information from one register to another

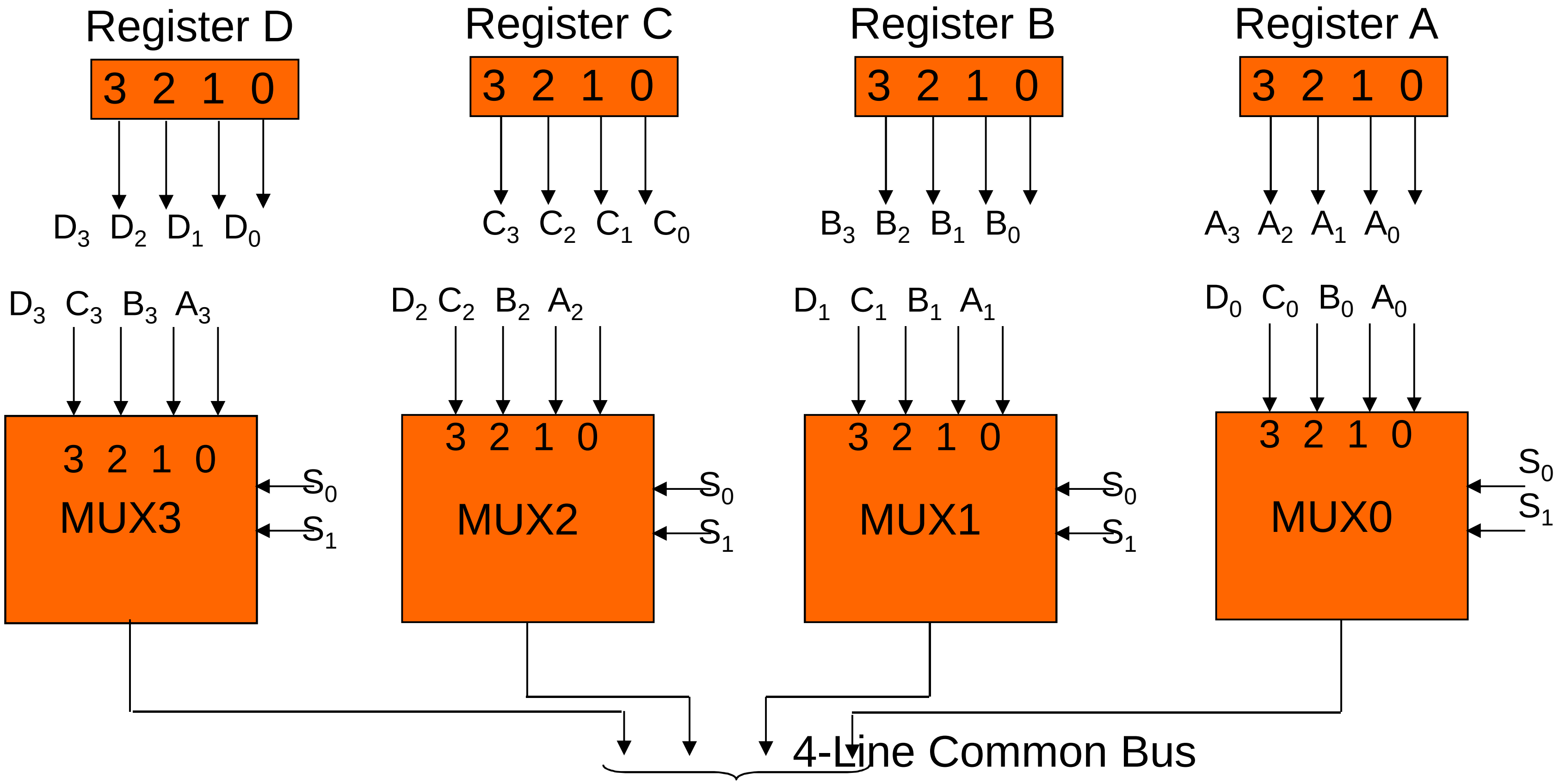
- A *Common Bus System* is a scheme for transferring information between registers in a multiple-register configuration

- 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





Bus and Memory Transfers

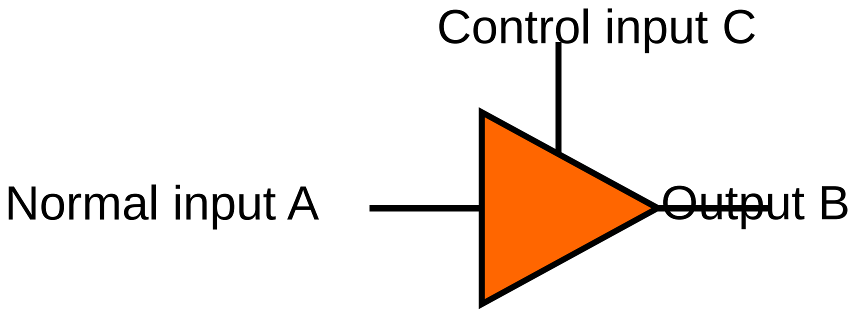
- The transfer of information from a bus into one of many destination registers is done:
  - By connecting the bus lines to the inputs of all destination registers and then:
  - activating the load control of the particular destination register selected

- We write: R2 ← 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 ←C, (select C)

R2 ←BUS (Load R2)

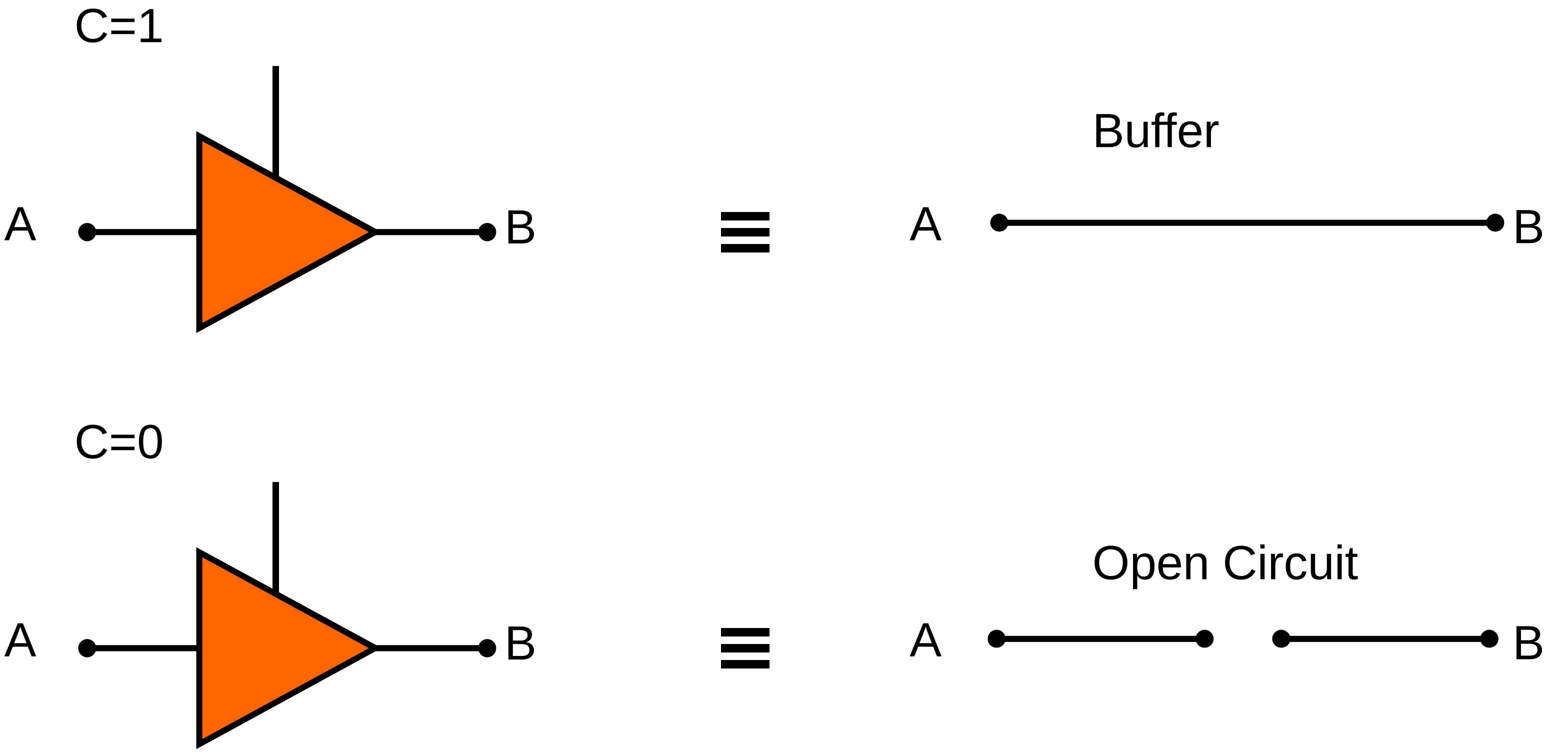
Bus and Memory Transfers: Three-State Bus Buffers

- 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)

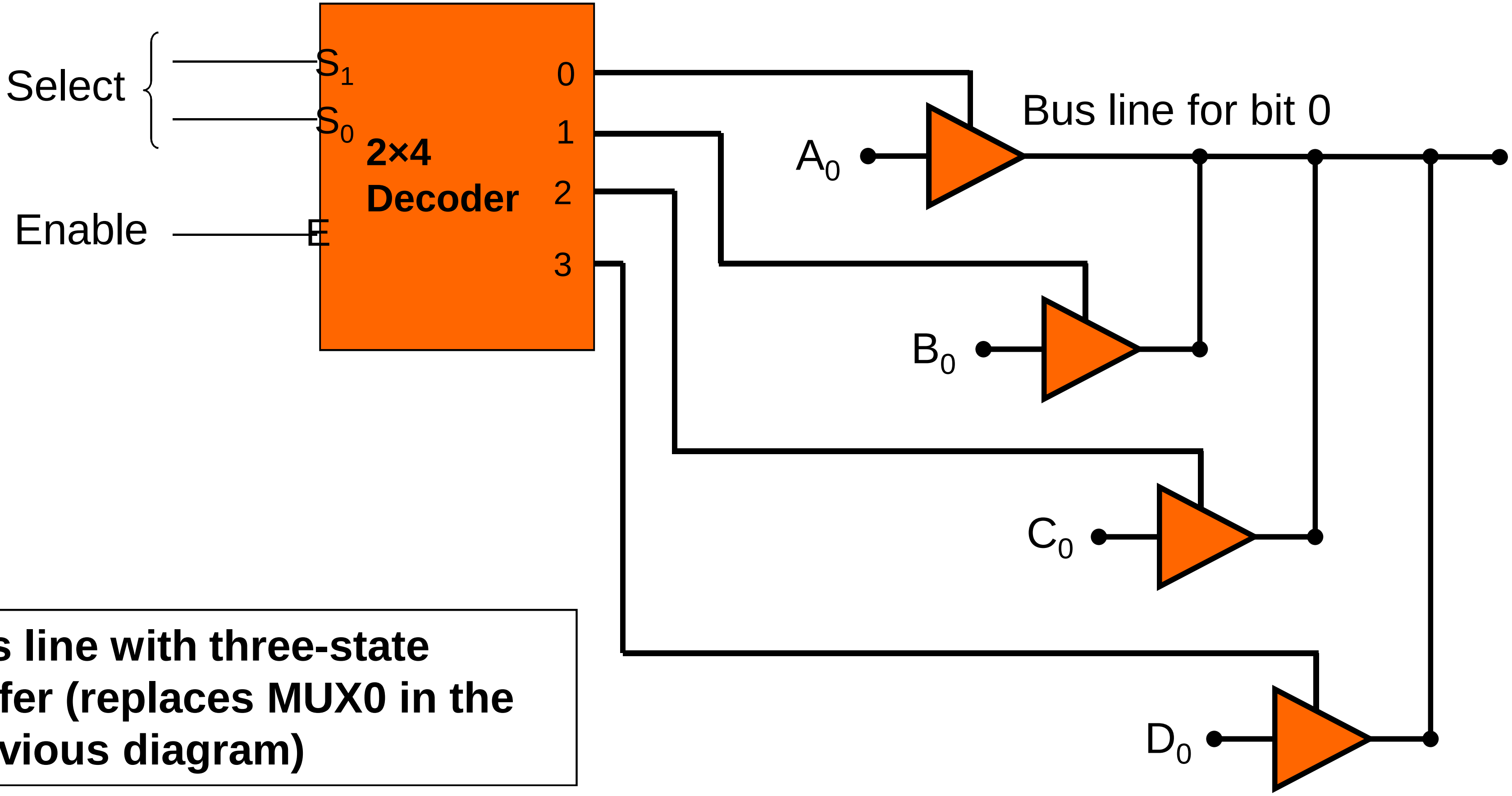


Three-State Buffer

Bus and Memory Transfers: Three-State Bus Buffers $^{\text{cont.}}$



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



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

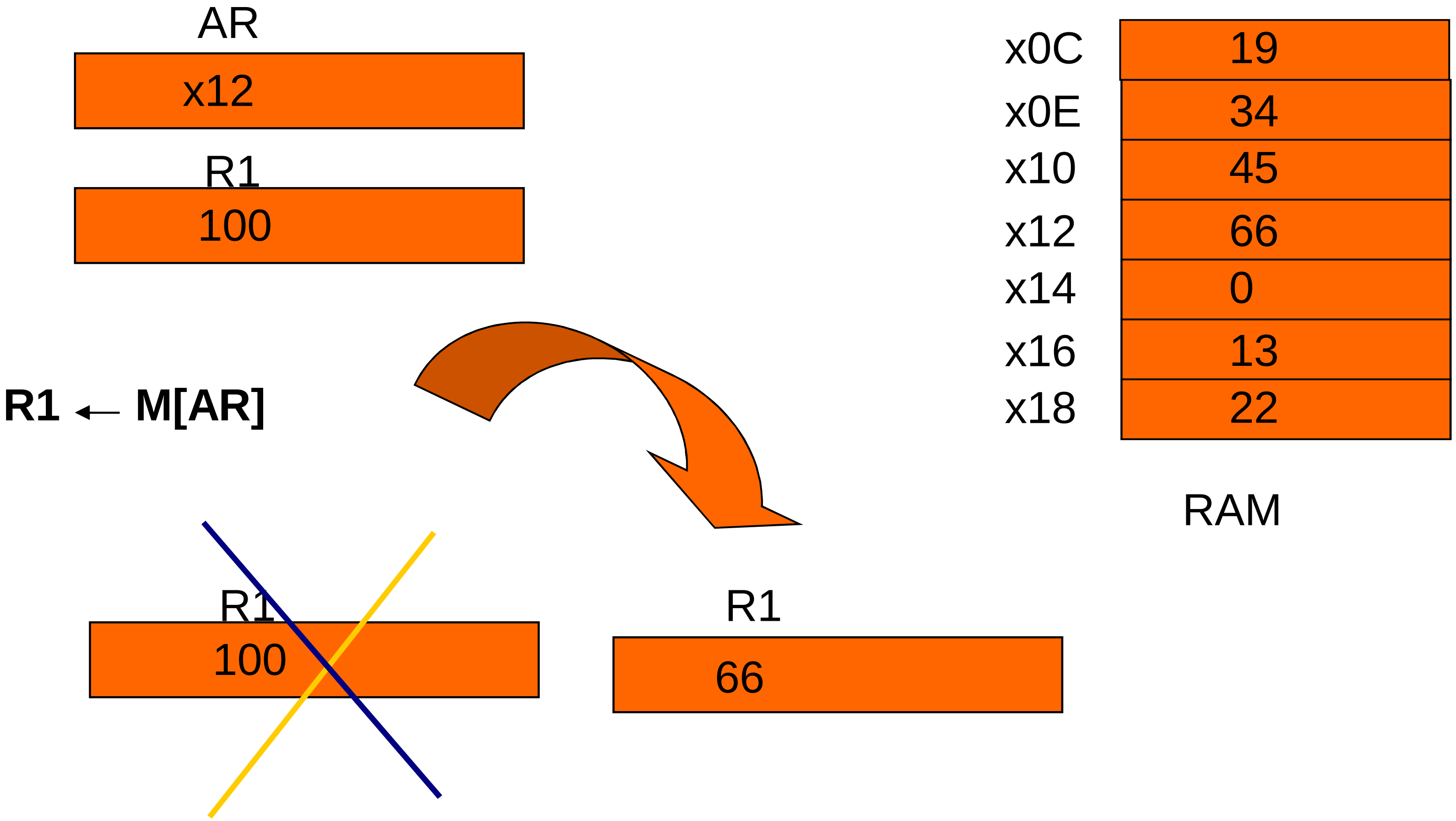
Bus and Memory Transfers: Memory Transfer

- Memory read : Transfer from memory
- Memory write : Transfer to memory
- Data being read or wrote is called a memory word (called M)- (refer to section 2-7)
- 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: Memory Transfer $^{\text{cont.}}$

- 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 ← M[AR]
  - Write: M[AR] ← DR

Bus and Memory Transfers: Memory Transfer cont.



AR

x12

R1

100

R1 ← M[AR]

R1

100

R1

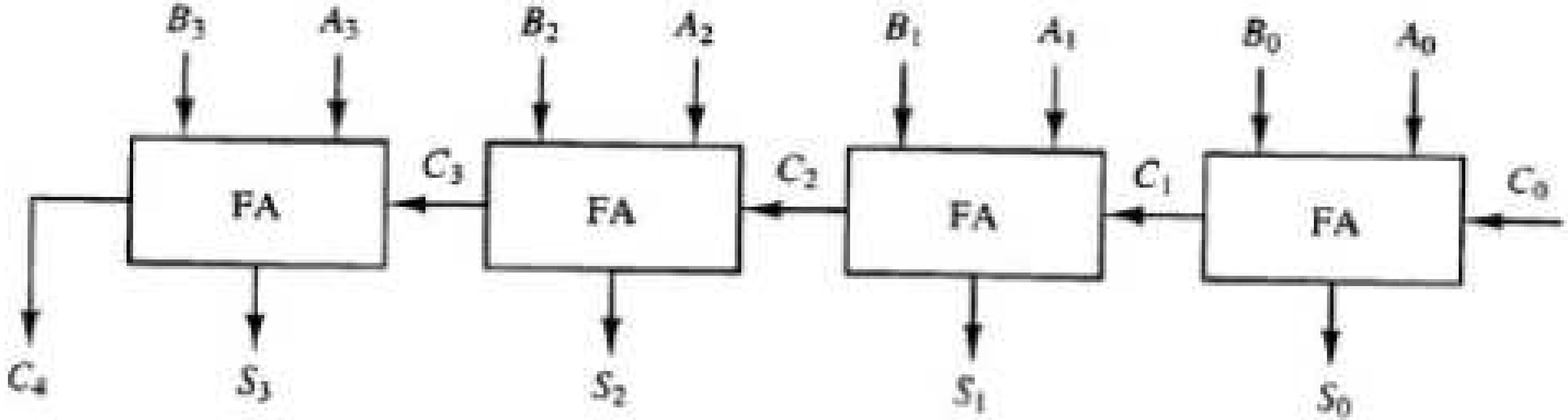
66

x0C	19
x0E	34
x10	45
x12	66
x14	0
x16	13
x18	22

RAM

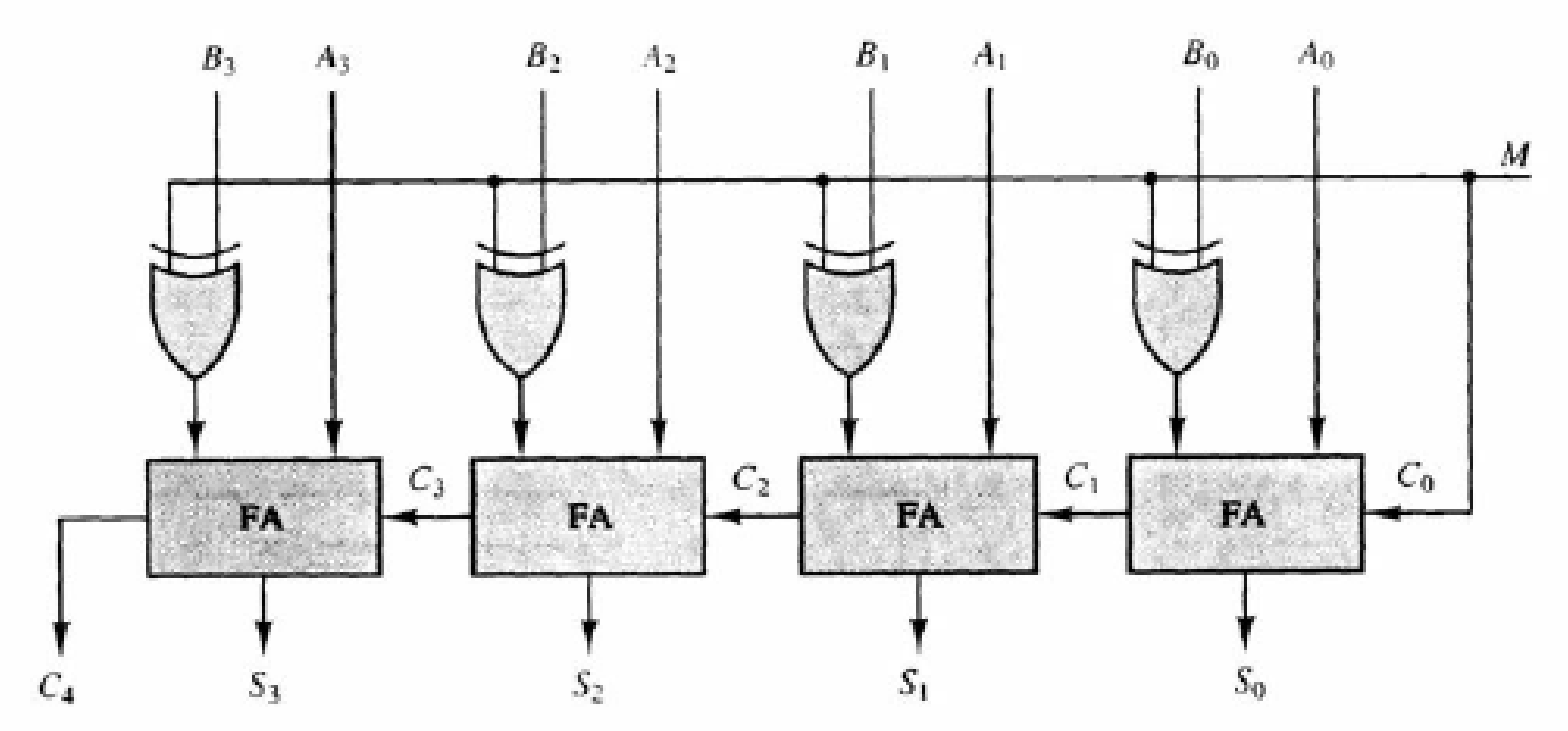
Arithmetic Micro-operations

- Binary Adder

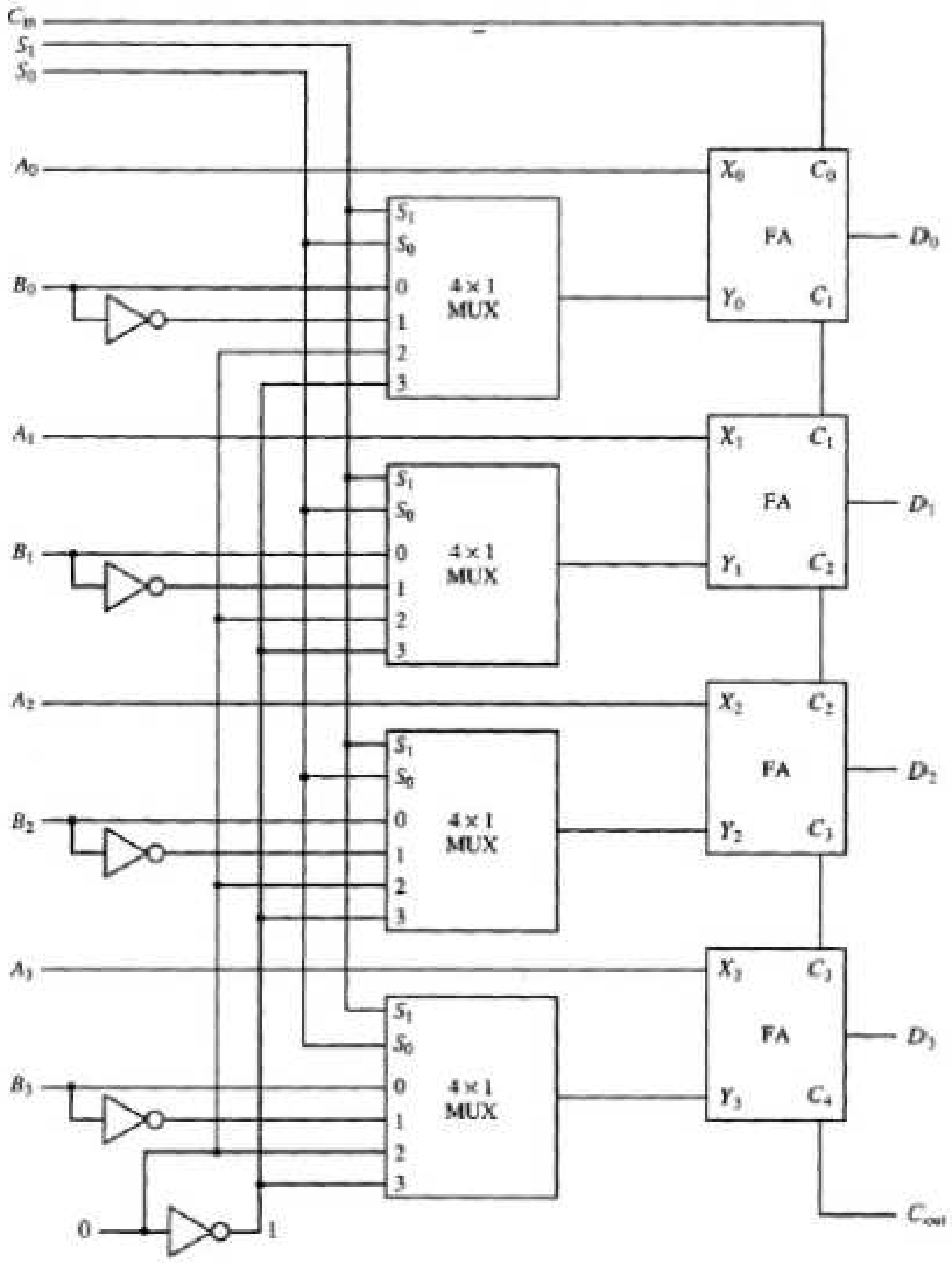


Arithmetic Micro-operations

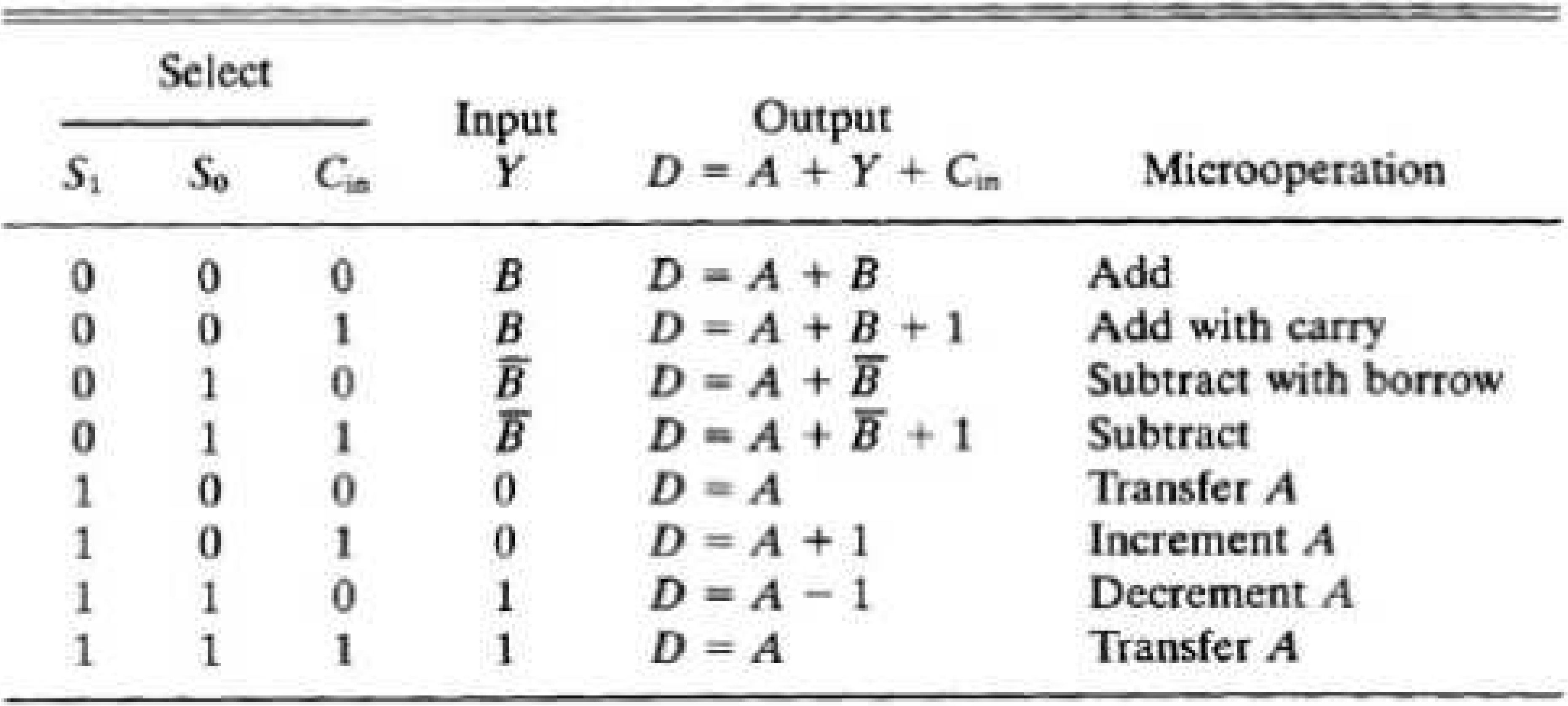
- Binary Adder/Subtractor



Arithmetic Circuit

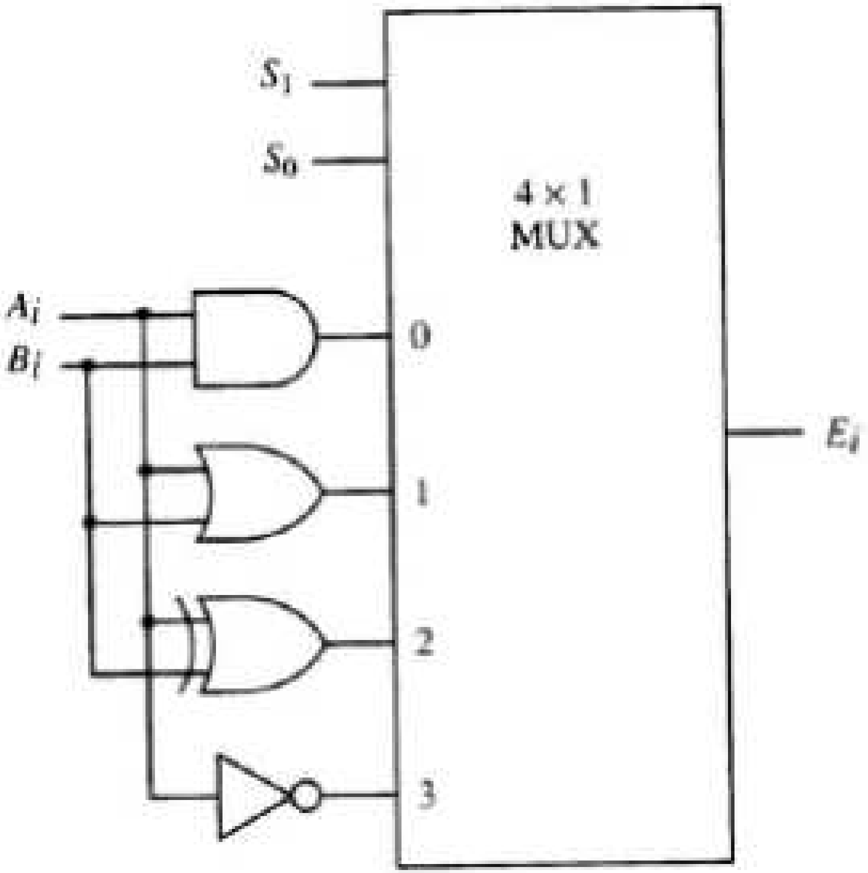


Arithmetic Circuit



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

Logical 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	$E = \bar{A}$	Complement

(b) Function table