University Of Diyala
College Of Engineering
Computer Engineering Department

COMPUTER ARCHITECTURE I

PART 2: REGISTER TRANSFER AND MICRO-OPERATIONS

Asst. Prof. Ahmed Salah Hameed

Second stage

2022-2023

CONTENTS

4-1 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

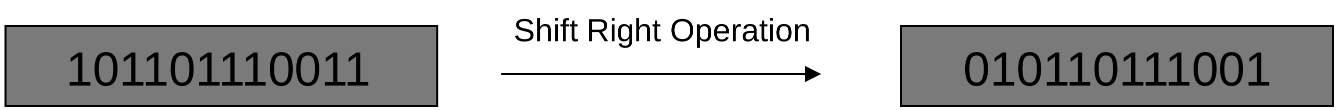
4-1 REGISTER TRANSFER LANGUAGE CONT.

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 or
- transferred to another register



4-1 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

4-1 REGISTER TRANSFER LANGUAGE CONT.

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

Next steps:

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

4-2 REGISTER TRANSFER
(OUR FIRST MICRO-OPERATION)

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

4-2 REGISTER TRANSFER CONT.

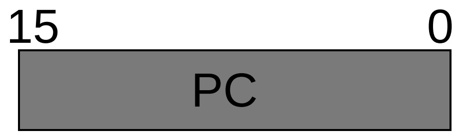
The individual flip-flops in an n-bit register are numbered in sequence from 0 to n-1 (from the right position toward the left position)

Register R1 — Showing individual bits

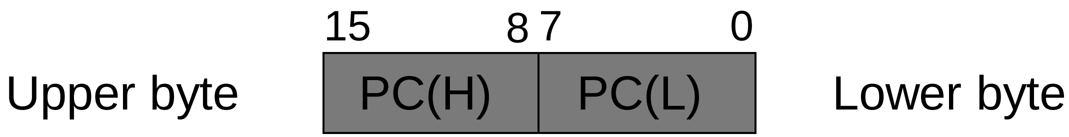
A block diagram of a register

4-2 REGISTER TRANSFER CONT.

Other ways of drawing the block diagram of a register:



Numbering of bits



Partitioned into two parts

4-2 REGISTER TRANSFER CONT.

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 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

4-2 REGISTER TRANSFER CONT.

Conditional transfer occurs only under a control condition

Representation of a (conditional) transfer

P: $R2 \leftarrow R1$

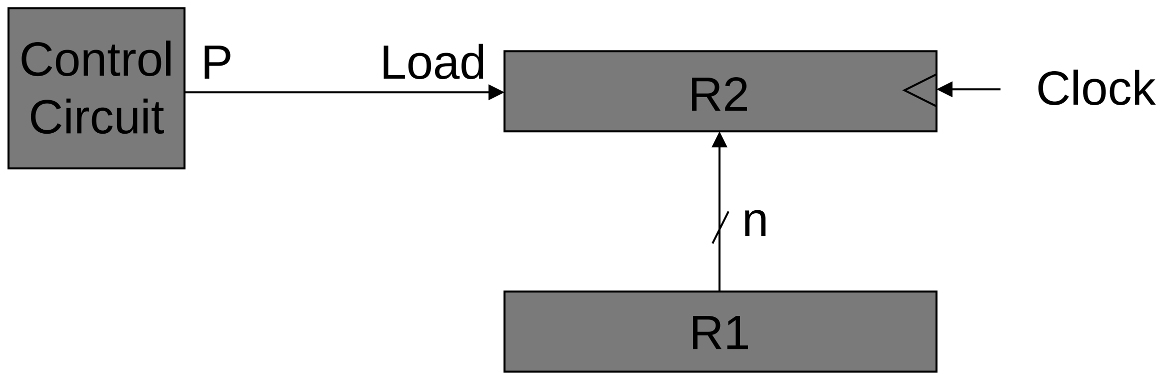
A binary condition (P equals to 0 or 1) determines when the transfer occurs

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

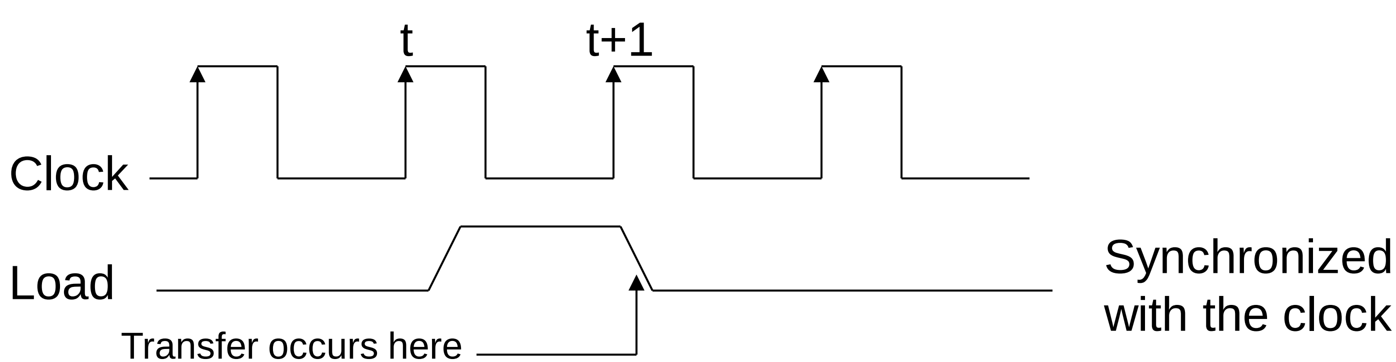
4-2 REGISTER TRANSFER CONT.

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

Block diagram:



Timing diagram



4-2 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

4-3 BUS AND MEMORY TRANSFERS

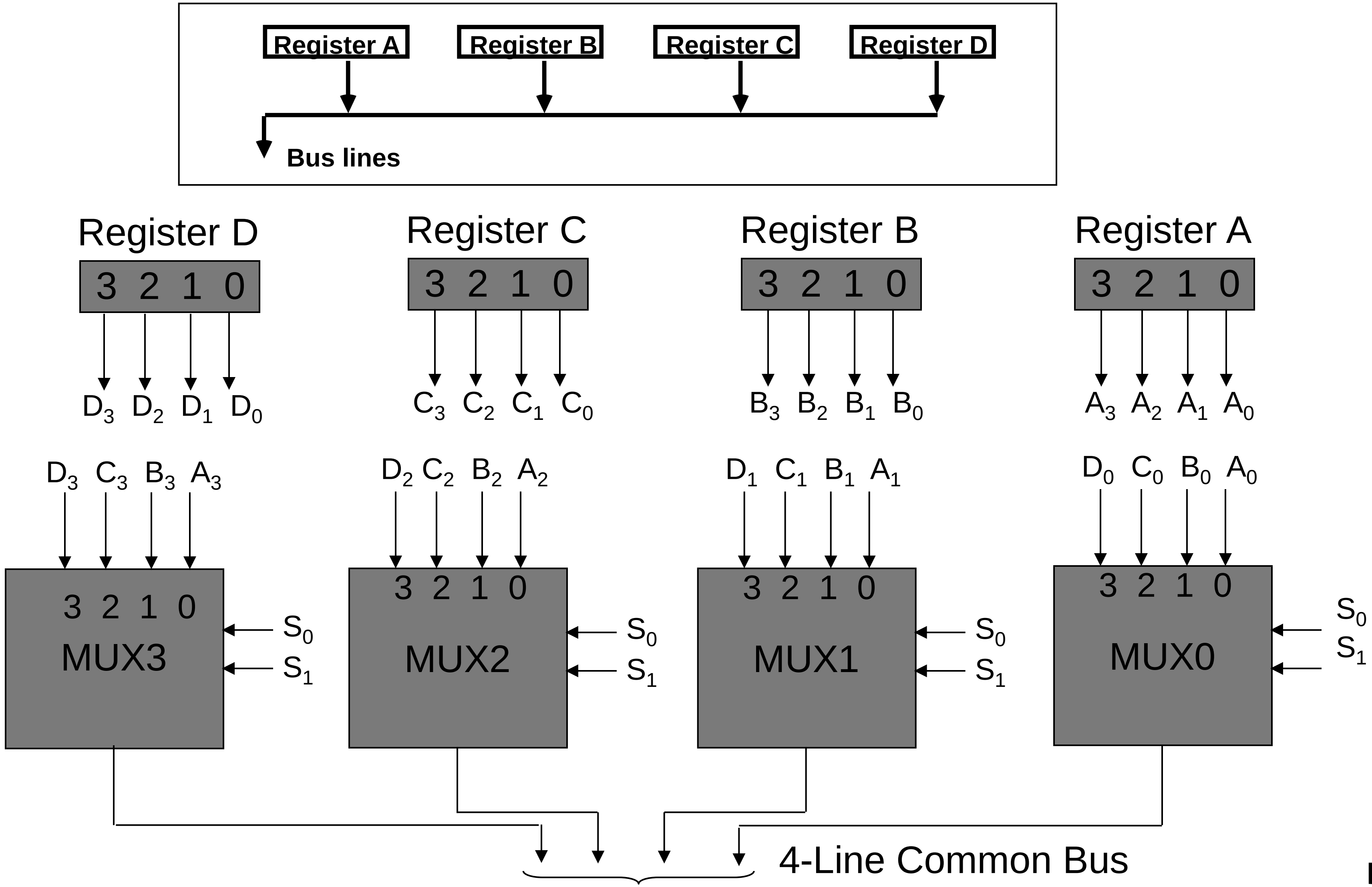
Paths must be provided to transfer information from one register to another

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

4-3 BUS AND MEMORY TRANSFERS



4-3 BUS AND MEMORY TRANSFERS

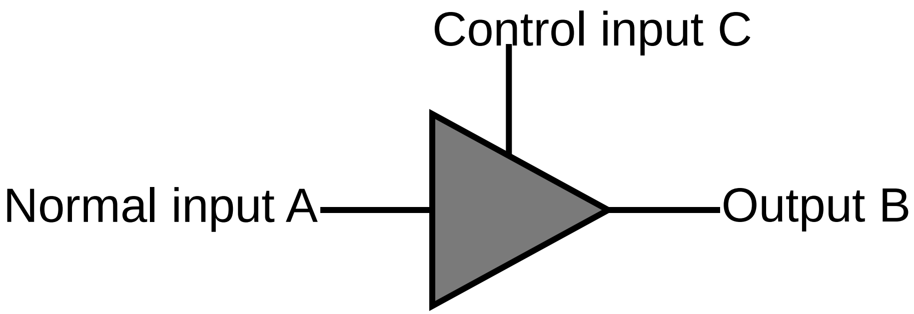
- **The transfer of information from a bus into one of many destination registers is done:**
  - By connecting the bus lines to the inputs of all destination registers and then:
  - activating the load control of the particular destination register selected
- **We write: R2 $\leftarrow$ C to symbolize 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)

4-3 BUS AND MEMORY TRANSFERS:
THREE-STATE BUS BUFFERS

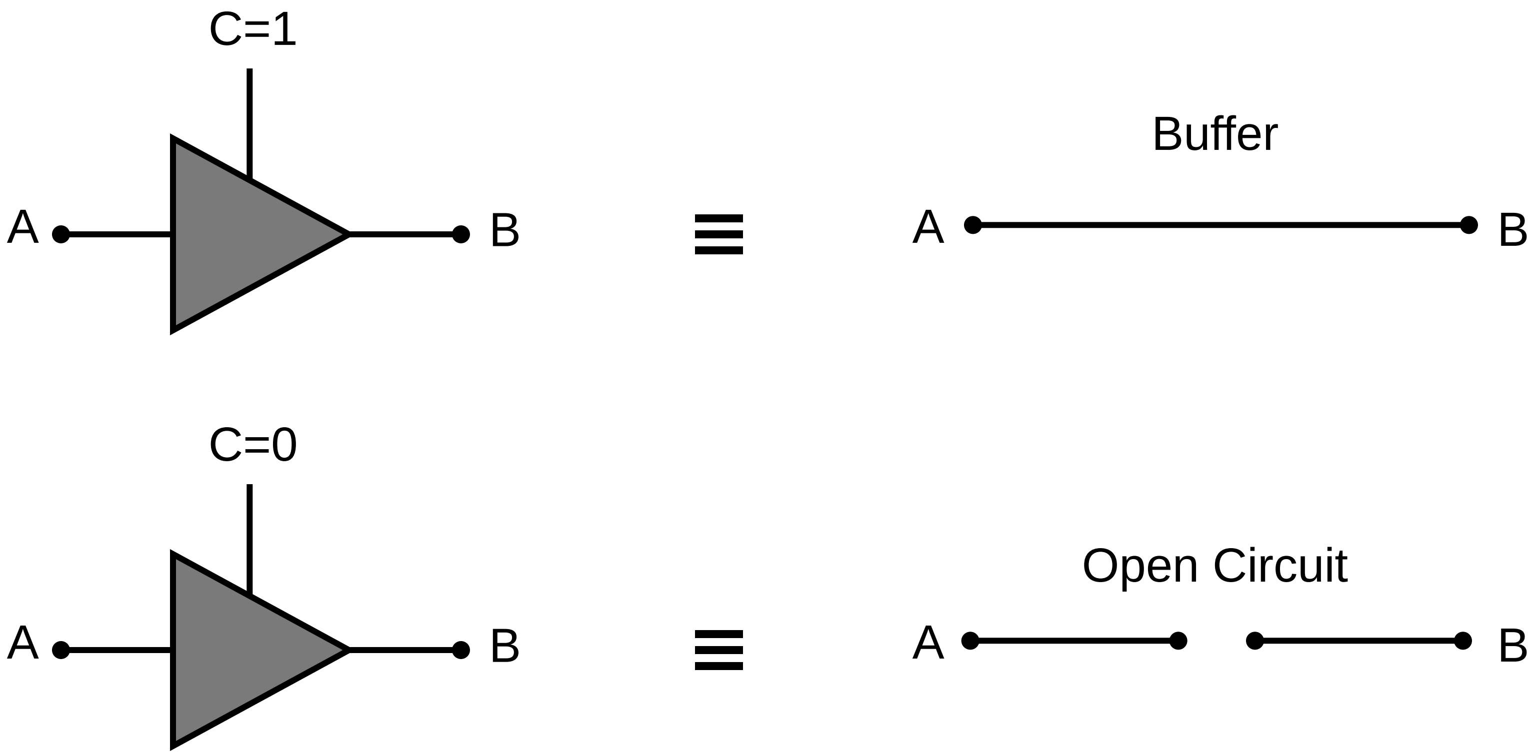
A bus system can be constructed with three-state buffer gates instead of multiplexers

A three-state buffer is a digital circuit that exhibits three states: logic-0, logic-1, and high-impedance (Hi-Z)

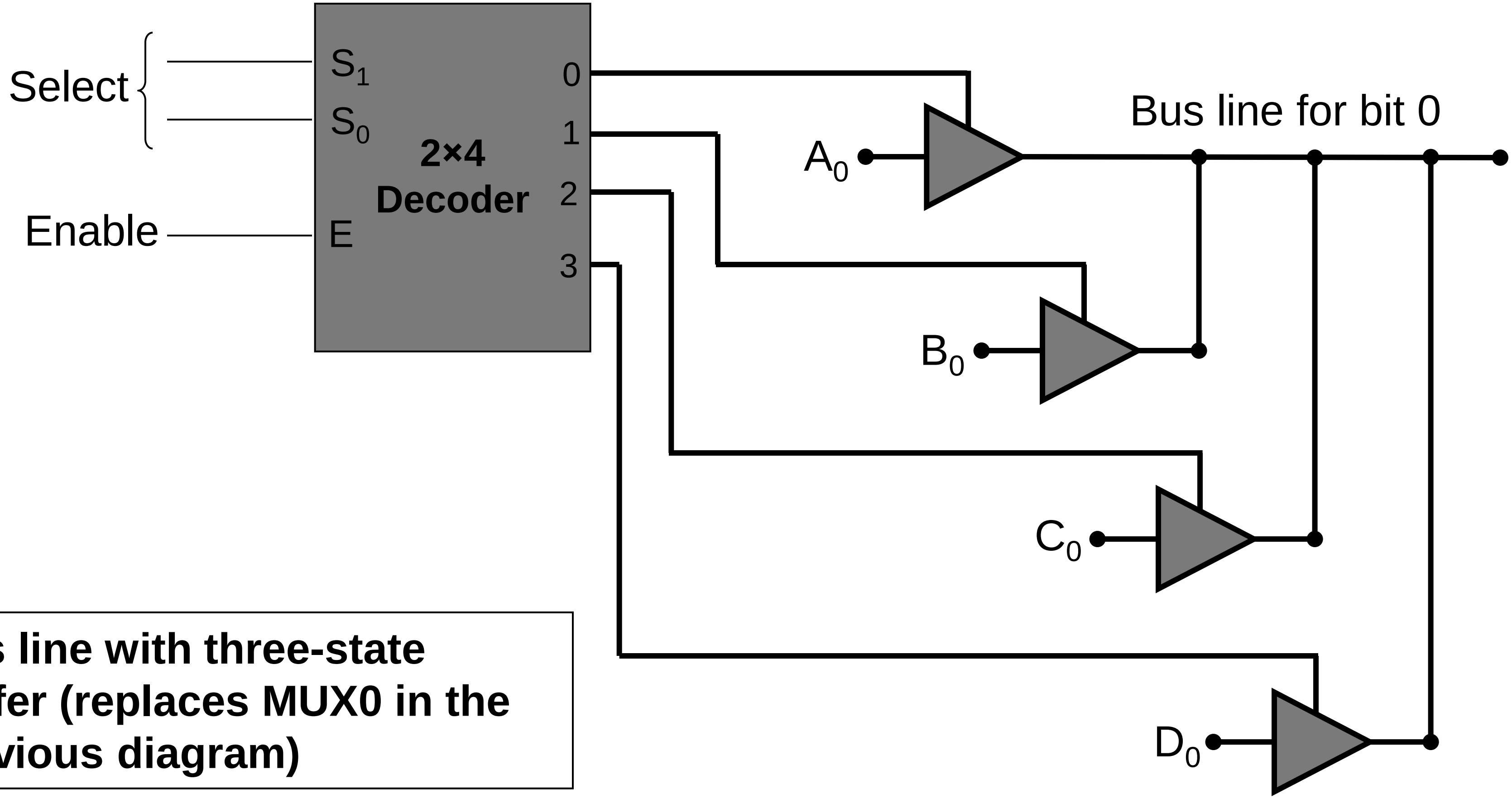


Three-State Buffer

4-3 BUS AND MEMORY TRANSFERS:
THREE-STATE BUS BUFFERS CONT.



4-3 BUS AND MEMORY TRANSFERS:
THREE-STATE BUS BUFFERS CONT.



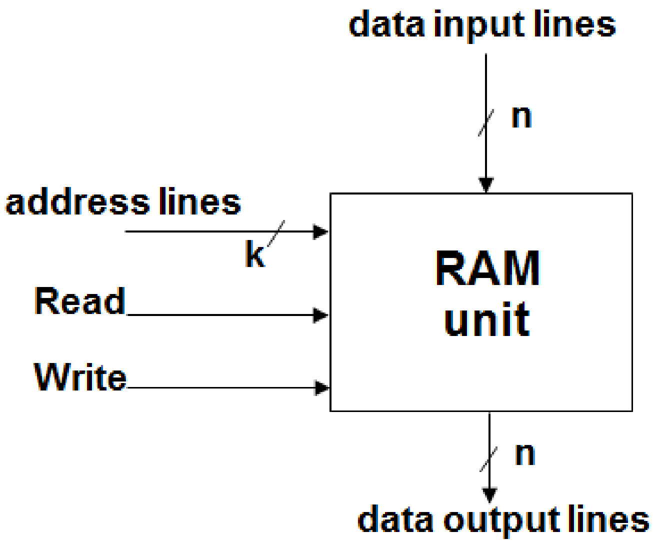
Bus line with three-state buffer (replaces MUX0 in the previous diagram)

4-3 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).
- 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

4-3 BUS AND MEMORY TRANSFERS: MEMORY TRANSFER CONT.

- Each register (word) can hold n bits of data
- Assume the RAM contains r = 2^k words. It needs the following
  - n data input lines
  - n data output lines
  - k address lines
  - A Read control line
  - A Write control line



4-3 BUS AND MEMORY TRANSFERS: MEMORY TRANSFER CONT.

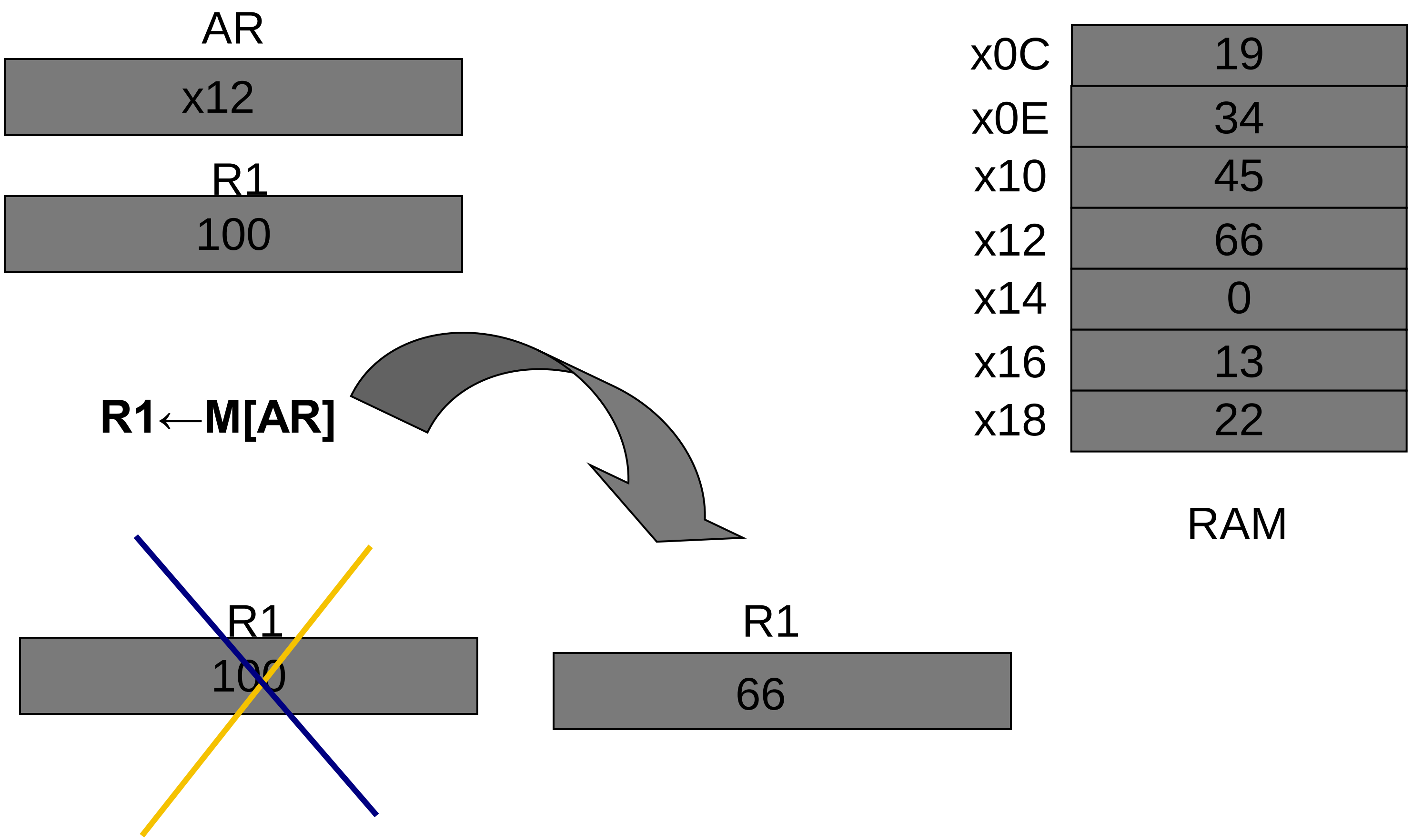
Assume that the address of a memory unit is stored in a register called the Address Register AR

Lets represent a Data Register with DR, then:

Read: DR ← M[AR]

Write: M[AR] ← DR

4-3 BUS AND MEMORY TRANSFERS: MEMORY TRANSFER CONT.



SUMMARY OF REGISTER TRANSFER MICROOPERATIONS

A ← B	Transfer content of reg. B into reg. A
AR ← DR(AD)	Transfer content of AD portion of reg. DR into reg. AR
A ← constant	Transfer a binary constant into reg. A
ABUS ← R1,	Transfer content of R1 into bus A and, at the same time,
R2 ← ABUS	transfer content of bus A into R2
AR	Address register
DR	Data register
M[R]	Memory word specified by reg. R
M	Equivalent to M[AR]
DR ← M	Memory *read* operation: transfers content of memory word specified by AR into DR
M ← DR	Memory *write* operation: transfers content of DR into memory word specified by AR

4-4 ARITHMETIC MICRO-OPERATIONS

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

4-4 ARITHMETIC MICRO-OPERATIONS CONT.

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

$$R3 \leftarrow R1 + R2$$

- Subtraction Microoperation:

$$R3 \leftarrow R1 - R2$$

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

4-4 ARITHMETIC MICRO-OPERATIONS 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:

$$R2 \leftarrow R2-1$$

SUMMARY OF TYPICAL ARITHMETIC MICRO-OPERATIONS

R3 ← R1 + R2	Contents of R1 plus R2 transferred to R3
R3 ← R1 - R2	Contents of R1 minus R2 transferred to R3
R2 ← R2’	Complement the contents of R2
R2 ← R2’+ 1	2's complement the contents of R2 (negate)
R3 ← R1 + R2’+ 1	subtraction
R1 ← R1 + 1	Increment
R1 ← R1 - 1	Decrement

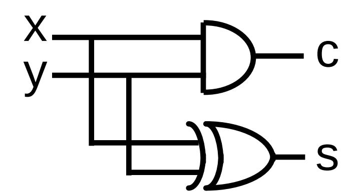
HALF ADDER/FULL ADDER

Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

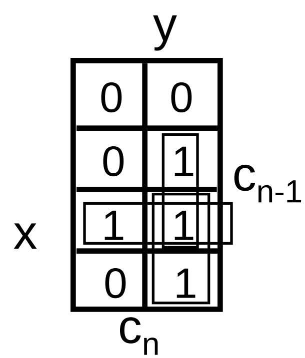
$c = xy$

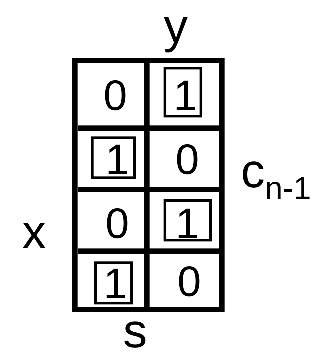
$s = xy' + x'y$
$= x \oplus y$



Full Adder

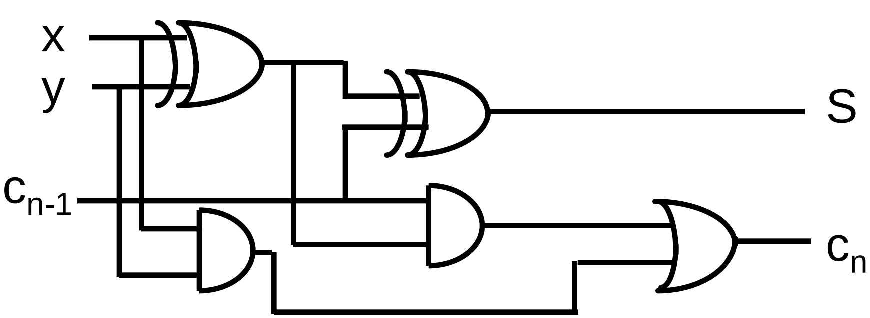
x	y	c_{n-1}	c_n	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



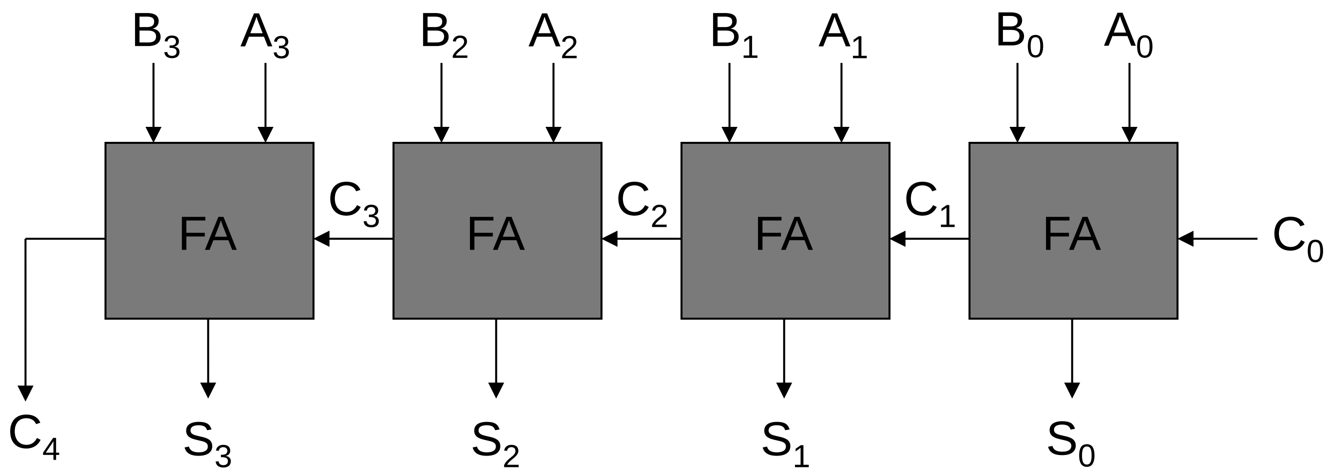


$c_n = xy + xc_{n-1} + yc_{n-1}$
$= xy + (x \oplus y)c_{n-1}$

$s = x'y'c_{n-1} + x'yc'_{n-1} + xy'c'_{n-1} + xyc_{n-1}$
$= x \oplus y \oplus c_{n-1} = (x \oplus y) \oplus c_{n-1}$

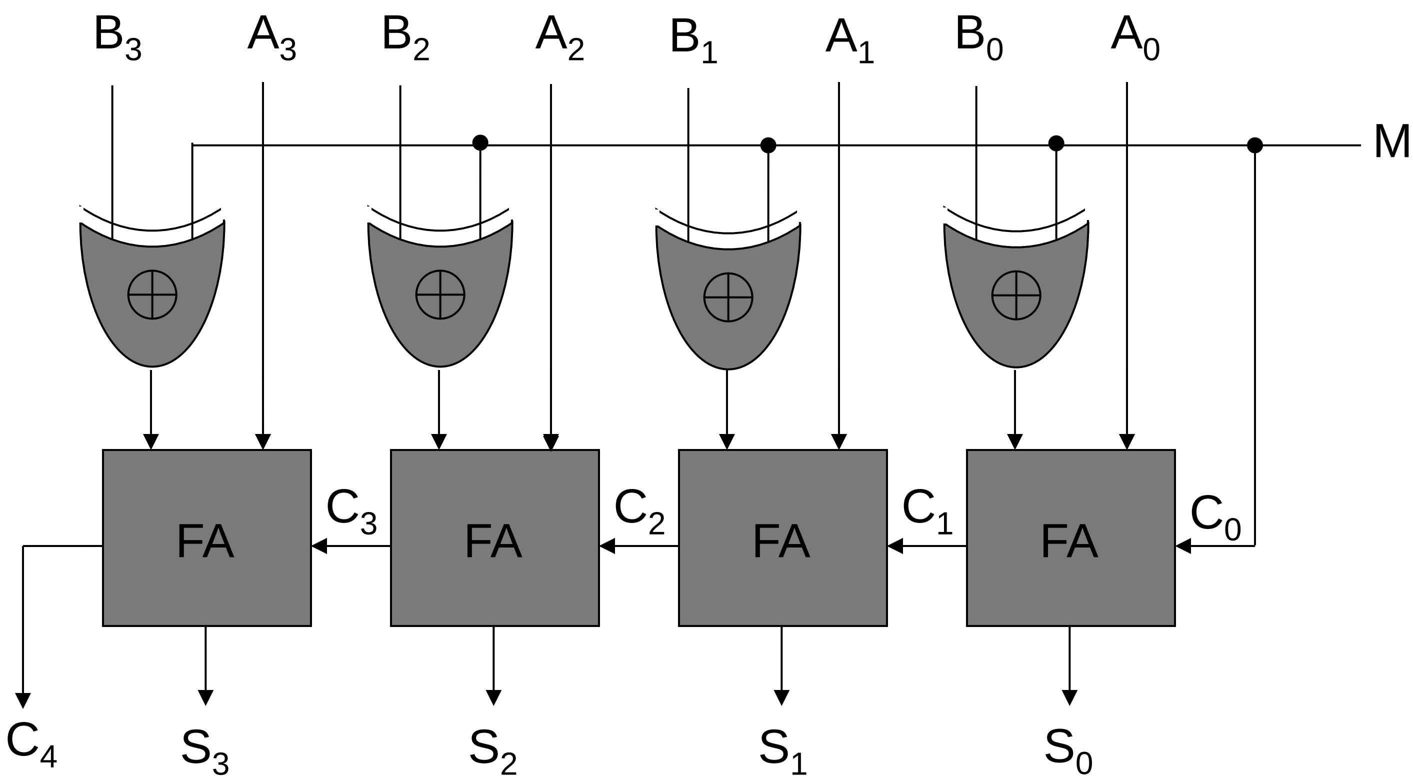


4-4 ARITHMETIC MICRO-OPERATIONS
BINARY ADDER



4-bit binary adder
(connection of FAs)

4-4 ARITHMETIC MICRO-OPERATIONS BINARY ADDER-SUBTRACTOR



4-bit adder-subtractor

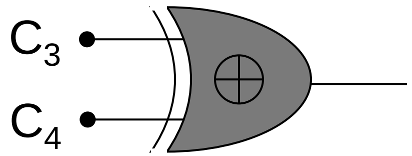
4-4 ARITHMETIC MICRO-OPERATIONS BINARY ADDER-SUBTRACTOR

- For unsigned numbers, this gives A – B if A≥B or the 2's complement of (B – A) if A < B

(example: 3 – 5 = -2= 1110)

- For signed numbers, the result is A – B provided that there is no overflow. (example : -3 – 5= -8)

```
 1101
 1011 +
------
 1000
```



$$V = \begin{cases} 1, \text{ if overflow} \\ 0, \text{ if no overflow} \end{cases}$$

Overflow detector for signed numbers

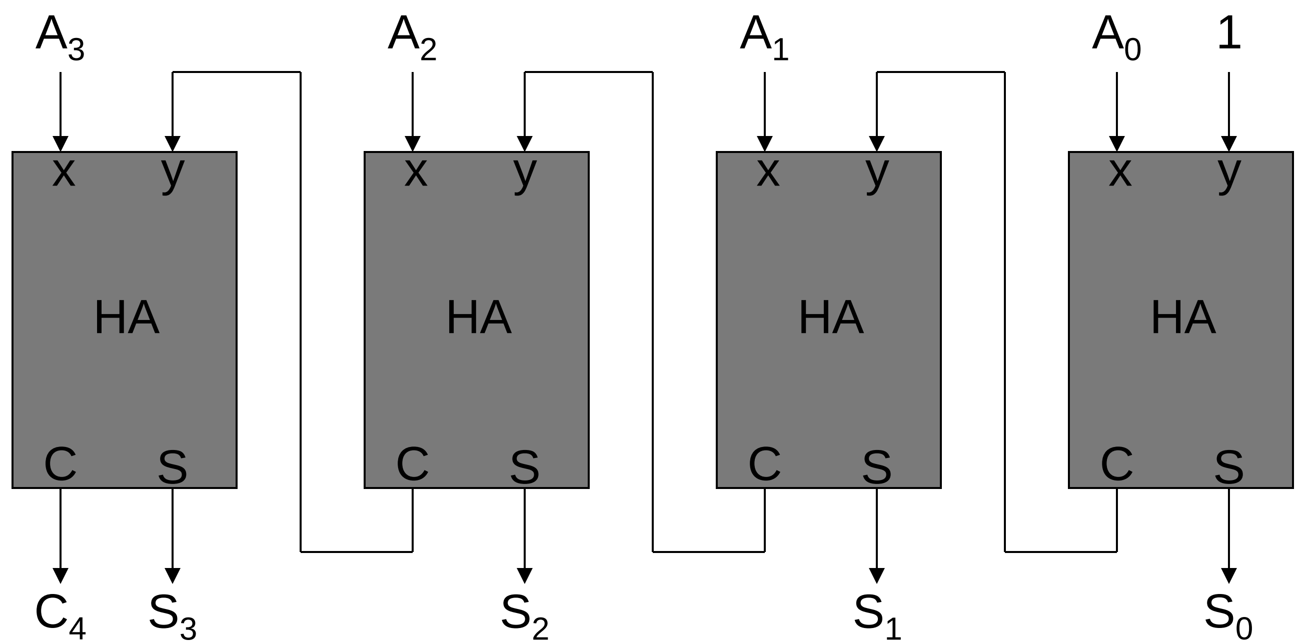
4-4 ARITHMETIC MICRO-OPERATIONS BINARY ADDER-SUBTRACTOR

What is the range of unsigned numbers that can be represented in 4 bits?

What is the range of signed numbers that can be represented in 4 bits?

Repeat for n-bit?!

4-4 ARITHMETIC MICRO-OPERATIONS
BINARY INCREMENTER



4-bit Binary Incrementer

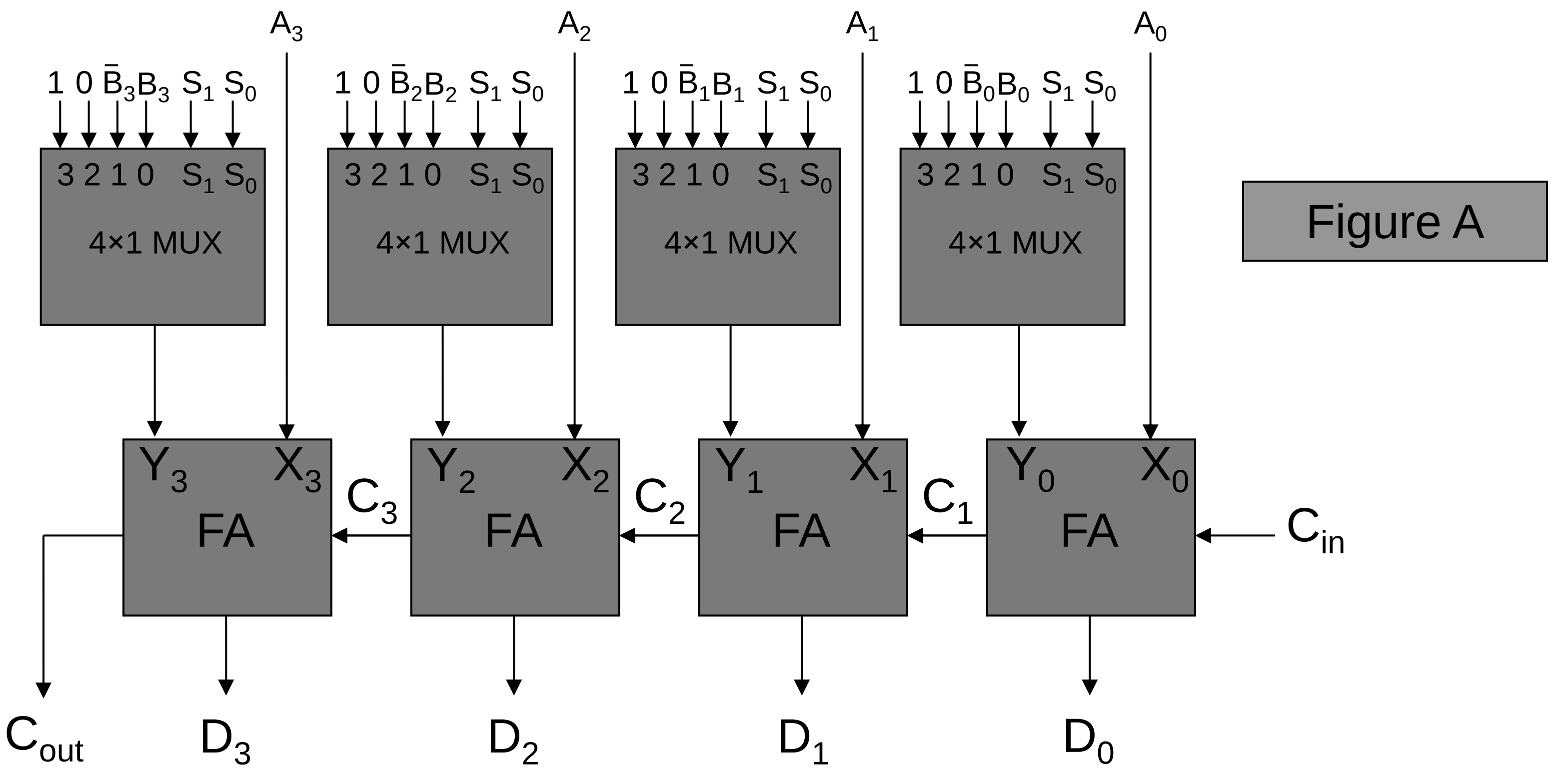
- **Binary Incrementer can also be implemented using a counter**
- **A binary decrementer can be implemented by adding 1111 to the desired register each time!**

4-4 ARITHMETIC MICRO-OPERATIONS ARITHMETIC CIRCUIT

- This circuit performs seven distinct arithmetic operations and the basic component of it is the parallel adder

- The output of the binary adder is calculated from the following arithmetic sum:
  - $D = A + Y + C_{in}$

4-4 ARITHMETIC MICRO-OPERATIONS
ARITHMETIC CIRCUIT CONT.



4-bit Arithmetic Circuit