

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

Computer Architecture & Organization (6CS4-04)

Computer Science and Engineering Department

Vision of the Department

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

Mission of the Department

- To impart outcome based education for emerging technologies in the field of computer science and engineering.
- To provide opportunities for interaction between academia and industry.
- To provide platform for lifelong learning by accepting the change in technologies.
- To develop aptitude of fulfilling social responsibilities.

Program Outcomes (PO):

- **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

- **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEO):

PEO1: To provide students with the fundamentals of Engineering Sciences with more emphasis in computer science and engineering by way of analyzing and exploiting engineering challenges.

PEO2: To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real life problems.

PEO3: To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.

PEO4: To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.

PEO5: To prepare students to excel in Industry and Higher education by educating Students along with High moral values and Knowledge.

Program Specific Outcome (PSO):

PSO: Ability to interpret and analyze network specific and cyber security issues, automation in real word environment.

PSO2: Ability to Design and Develop Mobile and Web-based applications under realistic constraints.

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

COURSE OUTCOMES: After Completion of the course, Students will be able to:

CO1: Identification of registers, micro-operations and basic computer organizations & design

CO2: Identification of computer architecture and processing

CO3: Introduction and applications of computer arithmetic operations

CO4 : Knowledge of computer Memory organization

Mapping Between CO and PO

CO-PO Mapping												
Computer Architecture 6CS4-04												
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
Co1: Ability to understand the functional units of the processor and various micro operations.	2	2	2	1	1	1	1	1	1	2	1	3
Co2: Analyze different architectural and organizational design issues that can affect the performance of a computer.	3	3	3	2	2	1	1	1	1	2	1	3
Co3. Examine the airthmetic problems and principles of computer design.	3	2	2	2	2	1	1	1	1	2	1	3
Co4. Describe and examine the concept of cache memory, Virtual memory and I/O organization.	3	2	2	2	1	1	1	1	1	1	1	3

Mapping Between CO and PSO:

CO-PSO Mapping		
Computer Architecture 6CS4-04		
	PSO1	PSO2
Co1: Ability to understand the functional units of the processor and various micro operations.	1	1
Co2: Analyze different architectural and organizational design issues that can affect the performance of a computer.	2	1
Co3. Examine the airthmetic problems and principles of computer design.	2	1
Co4. Describe and examine the concept of cache memory, Virtual memory and I/O organization.	1	1

Jaipur Engineering College and Research Centre
Computer Architecture and Organization (6CS4-04)
Session 2020-21



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-VI Semester: B.Tech. Computer Science and Engineering

6CS4-04: Computer Architecture and Organization

Credit: 3
3L+0T+0P

Max. Marks: 150(IA:30, ETE:120)

End Term Exam: 3 Hours

SN	Contents	Hours
1	Introduction: Objective, scope and outcome of the course.	01
2	Computer Data Representation: Basic computer data types, Complements, Fixed point representation, Register Transfer and Micro-operations: Floating point representation, Register Transfer language, Register Transfer, Bus and Memory Transfers (Tree-State Bus Buffers, Memory Transfer), Arithmetic Micro-Operations, Logic Micro-Operations, Shift Micro-Operations, Arithmetic logical shift unit. Basic Computer Organization and Design Instruction codes, Computer registers, computer instructions, Timing and Control, Instruction cycle, Memory-Reference Instructions, Input-output and interrupt, Complete computer description, Design of Basic computer, design of Accumulator Unit.	10
3	Programming The Basic Computer: Introduction, Machine Language, Assembly Language, assembler, Program loops, Programming Arithmetic and logic operations, subroutines, I-O Programming. Micro programmed Control: Control Memory, Address sequencing, Micro program Example, design of control Unit	7
4	Central Processing Unit: Introduction, General Register Organization, Stack Organization, Instruction format, Addressing Modes, data transfer and manipulation, Program Control, Reduced Instruction Set Computer (RISC) Pipeline And Vector Processing, Flynn's taxonomy, Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction, Pipeline, RISC Pipeline, Vector Processing, Array Processors	8
5	Computer Arithmetic: Introduction, Addition and subtraction, Multiplication Algorithms (Booth Multiplication Algorithm), Division Algorithms, Floating Point Arithmetic operations, Decimal Arithmetic Unit. Input-Output Organization, Input-Output Interface, Asynchronous Data Transfer, Modes Of Transfer, Priority Interrupt, DMA, Input-Output Processor (IOP), CPU IOP Communication, Serial communication.	8
6	Memory Organization: Memory Hierarchy, Main Memory, Auxiliary Memory, Associative Memory, Cache Memory, Virtual Memory. Multiprocessors: Characteristics of Multiprocessors, Interconnection Structures, Inter-processor Arbitration, Inter-processor Communication and Synchronization, Cache Coherence, Shared Memory Multiprocessors.	8
Total		42

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

Computer Architecture and Organization (6CS4-04)

LECTURE PLAN:

Unit No./ Total Lecture Reqd.	Topics	Lect. Reqd.	Lect. No.
Unit-I (10)	1. Objective, scope and outcome of the course. Basic computer data types, Complements, Fixed point representation, Register Transfer	1	1
	2. Micro-operations:		
	2a. Floating point representation, Register Transfer language	1	2
	2b. Register Transfer, Bus and Memory Transfers (Tree-State Bus Buffers, Memory Transfer)	1	3
	2c. Arithmetic Micro-Operations, Logic Micro-Operations	1	4
	2d. Shift Micro-Operations, Arithmetic logical shift unit	1	5
	3. Basic Computer Organization and Design Instruction codes	1	6
	4. Computer registers, computer instructions	1	7
	5. Timing and Control, Instruction cycle,	1	8
	6. Memory-Reference Instructions, Input-output and interrupt	1	9
	7. Complete computer description, Design of Basic computer, design of Accumulator Unit	1	10
BC-1	Von Neuman Architecture	1	11
Unit-II (7)	1. Introduction, machine language, Assembly language	1	12
	2. Assembler, Program loops	1	13
	3. Programming Arithmetic and logic operations	1	14
	4. Subroutines and I-O Programming	1	15
	5. Control Memory	1	16
	6. Address Sequencing	1	17
	7. Micro program Example	1	18
	8. Design of control unit	1	19
UNIT III (8)	1. Introduction General Register Organization	1	20
	2. Stack Organization, Instruction Format	1	21
	3. Addressing Modes, Data transfer and manipulation	1	22
	4. Program Control, Reduced Instruction set computer (RISC) pipeline	1	23
	5. Vector Processing and Flynn's taxonomy	1	24
	6. Parallel processing, pipeline	1	25
	7. Arithmetic pipeline, Instruction pipeline	1	26
	8. RISC pipeline, Vector Processing and Array Processing	1	27
Unit-IV (8)	1. Introduction, Addition and subtraction	1	28
	2. Multiplication Algorithms (Booth Multiplication Algorithm)	1	29
	3. Division Algorithms, Floating Point Arithmetic operations	1	30
	4. Decimal Arithmetic Unit, Input-Output Organization	1	31
	5. Input-Output Interface, Asynchronous Data Transfer	1	32
	6. Modes of Transfer, Priority Interrupt	1	33
	7. DMA, Input-Output Processor (IOP)	1	34

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

	8. CPU-IOP Communication, Serial communication	1	35
BC-2	Interaction of computer with hardware	1	36
Unit- V (8)	1. Memory Hierarchy	1	37
	2. Main Memory, Auxiliary memory	1	38
	3. Associative memory,	1	39
	4. Cache Memory, Virtual Memory	1	40
	5. Microprocessor: Characteristics of microprocessor	1	41
	6. Interconnection Structure	1	42
	7. Inter-processor Arbitration, Intercrosses communication and synchronization	1	43
	8. Cache Coherence, Shared Memory Multiprocessor	1	44
BC-3	Different Operating System Architectures and their relation to Computer System Architectures	1	45

This schedule is tentative and is subject to minimal changes during teaching.

Jaipur Engineering College and Research Centre
Computer Architecture and Organization (6CS4-04)
Session 2020-21

Unit 1: Computer Data Representation

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

Objectives

1.1 1's and 2's Complement

In computers, binary number system is used for storing information as 0's and 1's.

The basic unit of computer storage is a bit, which can be either 0 (off/positive) or
Did u know?

1 (on/negative).

1's Complement

1's complement is a method of representation of negative numbers in computers. A binary number's 1's complement is obtained by inverting all 0s to 1s and all 1s to 0s.



Example: 0 in 8-bits is represented as 00000000 and -0 is represented in 1's complement as 11111111.

2's Complement

2's complement is a method of representation of negative binary numbers in computers. A binary number's 2's complement is obtained by adding 1 to that number's 1's complement.



Example: We can represent -28 in 2's complement as follows:

1. Write 28 in binary form. 00011100
2. Invert the digits by converting 0's to 1's and 1's to 0's. 11100011
3. Add 1 to 11100011

The result is 11100100, which represents -28.

Computer circuits perform different operation on binary number system. Binary operations are the key to perform all basic arithmetic operations, such as addition, subtraction, multiplication, and division. In these operations, the Most Significant Bit (MSB) is reserved to indicate the sign (+ or -). MSB is **0** for positive numbers and **1** for negative numbers. The rules for the binary additions are depicted in the table 1.1.

Jaipur Engineering College and Research Centre
Computer Architecture and Organization (6CS4-04)
Session 2020-21

As per the table 1.1, when two positive numbers are to be added, bit pairs are added, starting from lower-order (right end) bits, going up to the higher-order bits. While adding 1+1, a carry is generated.

Table 1.1: Binary Operation (Addition)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Just like binary additions, binary subtractions also have some rules to be followed. These rules are depicted in table 1.2.


As per table 1.2, when two positive numbers are to be subtracted, bit pairs are subtracted, starting from lower-order (right end) bits, going up to the higher-order bits. The subtraction of binary numbers is similar to the subtraction of decimal numbers. Just as decimal subtraction has the concept of “borrow”, subtraction of binary numbers also has the concept of “carry”. If you have to subtract a one from a zero, you need to “carry” from the left, just as in decimal subtraction.

Table 1.2: Binary Operation (Subtraction)

A	B	Difference	Carry
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

1's Complement Addition

In 1's complement addition, when two numbers are added, the two binary numbers may be added including the sign bit. If there is a carry after the MSB position, it is called carry-out. In case of signed number addition, it is called end-around-carry. In 1's complement, the representation of positive numbers is identical to sign magnitude system. To represent a negative number, the convention is different and is done by bit- complementing method, meaning replacing 0 by 1 and 1 by 0.


E  xample: Add $(1110)_2$ and $(1010)_2$.

$\begin{array}{r} 1 \\ 1110 \\ + 1010 \\ \hline 11000 \end{array}$	Carry	$\begin{array}{r} \text{Decimal } 18 \\ + \text{Decimal } 10 \\ \hline 0 \\ \hline \text{Decimal } 24 \end{array}$
---	-------	---

1's Complement Subtraction


1's complement has two cases of subtraction. These cases are:

1. Subtraction of a smaller number from a larger number. The steps followed for subtraction are:
 - (a) Determine the 1's complement of the smaller number.
 - (b) Add 1's complement to the larger number.
 - (c) Add the carry that is obtained, to the result.

 Example: Subtract $(101011)_2$ from $(111001)_2$.

$\begin{array}{r} 111001 \\ + 01010 \\ \hline 0 \\ \hline 1\ 001101 \end{array}$	1's complement of 101011
$\begin{array}{r} 001101 \\ + 1 \\ \hline 001110 \end{array}$	Add end-round carry Answer

2. Subtraction of a larger number from a smaller number. The steps followed for this type of subtraction are:
 - (a) Determine the 1's complement of the larger number.
 - (b) Add 1's complement to the smaller number.
 - (c) As the result is in 1's complement, to obtain the required result, the answer is converted to 1's complement with a negative sign.

 Example: Subtract $(111001)_2$ with $(101011)_2$.

$\begin{array}{r} 101011 \\ + 000110 \\ \hline 110001 \end{array}$	→ 1's complement of 111001
$\begin{array}{r} 110001 \\ - 001110 \\ \hline 001110 \end{array}$	Answer in 1's complement
	Answer in negative

Jaipur Engineering College and Research Centre

Computer Architecture and Organization (6CS4-04)

Session 2020-21

2's Complement Subtraction

The following steps explain subtraction using 2's complement.

If A and B are the numbers that are to be subtracted, A is called as minuend and B is called the subtrahend.

1. Represent both numbers in signed-2's complement format.
2. Obtain 2's complement of the subtrahend B (which may be in complement form already if it is negative).
3. Add it to A.



Example: Subtract 12 from 7.

$$\begin{array}{r} +7 \text{ } (-12) \\ 0000 \ 0111 \ (+7) \\ + \ 0100 \ (-) \\ \hline 1111 \ 1011 \ (-5) \end{array}$$

The result is automatically in signed-2's complement form.

2's complement has two cases of subtraction. These cases are:

1. Subtraction of a smaller number from a larger number. The subtraction steps are:
 - (a) The smaller number of 2's complement is determined.
 - (b) The bigger number is added with 2's complement.
 - (c) The carry is discarded.

The magnitude of the signed binary numbers can be represented using three approaches. They are:

1. Sign and magnitude representation.
2. Signed 1's complement representation.
3. Signed 2's complement representation.

The following section deals with a brief explanation of these three approaches.

Sign and Magnitude Representation

In this approach, the leftmost bit in the number is used for indicating the sign; 0 indicates a positive integer, and 1 indicates a negative integer. The remaining bits in the number give the magnitude of the number.

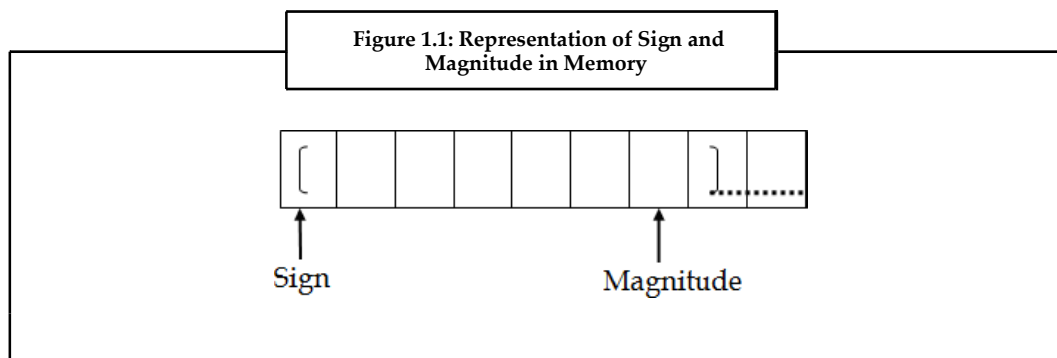


Example: -2410 is represented as:

1001 1000

In this example, the leftmost bit 1 means negative, and the magnitude is 24.

The magnitude for both positive and negative value is same, but they differ only with their signs. The range of values for the sign and magnitude representation is from -127 to 127.



Example: 1-sign bit and 7-bit magnitude.

01001001 = +73 0 at the LHS indicates the positive

value 10010010 = -73 1 at the LHS indicates the

negative value

Signed 1's Complement Representation

In signed 1's complement representation, a negative value is obtained by taking the 1's complement of the corresponding positive number. Also, a signed 1's complement method produces end carry during arithmetic operation that cannot be discarded.

The range of values for the signed 1's complement representation is from -127 to 128.



Example: Consider 8-bit numbers for 1's complement.

$(29)_{10} = (00011101)_2 = 000011101$ 1's complement for positive value

$-(29)_{10} = -(00011101)_2 = 111100010$ 1's complement for negative value

Notes

Signed 2's Complement Representation

In signed 2's complement representation, the 2's complement of a number is found by first taking the 1's complement of that number, then incrementing the result by 1.

The range of values for the signed 2's complement representation is from -128 to 127.



Example:

Consider 8-bit numbers for 2's complement.

$(29)_{10} = (00011100)_2 = (000011100)_{2s}$ 2's complement for positive value

$-(29)_{10} = -(00011100)_2 = (11110010)_{2s}$ 2's complement for negative value

1.2.1 Floating-Point Representation

The floating-point representation is used to perform operations for high range values. The scientific calculations are carried out using floating-point values. To make calculations simple, scientific numbers are represented as follows:

The number 5,600,000 can be represented as $0.56 * 10^7$

Here, 0.56 is the mantissa and 7 is the value of the exponent.

Similar to the above example, binary numbers can also be represented in the exponential form. The representation of binary numbers in the exponential form is known as floating-point representation. The floating-point representation divides the number in two parts, the left hand side is a signed, fixed-point number called **mantissa** and the right hand side of the number is called the **exponent**. The floating-point values are also assigned with a sign; **0** indicating the positive value and **1** indicating the negative value.

General form of floating-point representation of a binary

$$\begin{aligned} \text{number: } x &= (x_0 * 2^0 + x_1 * 2^1 + x_2 * 2^2 + \dots \\ &+ b_{-(n-1)} * 2^{-(n-1)}) \\ &\text{mantissa} * 2^{\text{exponent}} \end{aligned}$$

In the above syntax, the decimal point is moved left for negative exponents of two and right for positive exponents of two. Both the mantissa and the exponent are signed values allowing for negative numbers and for negative exponents respectively.



Example: Convert 111101.1000110 into floating-point value.

$111101.1000110 = 1.111011000110 * 2^5$ converted to floating-point value.

_____ - indicates the negative sign value

In this example, the integer value is converted to floating-point value by shifting the radix point next to the signed integer and scaling up the number to the exponential form by multiplying the value with the base 2. The value remains unchanged and this procedure is called the normalized method.

The floating-point numbers are of two types, which are:

1. Normalized and Un-normalized
2. Single precision and Double precision

Normalized and Un-Normalized Floating-Point Numbers

In normalized floating-point representation, the most significant digit of the mantissa is non-zero. Thus, the number is normalized only if its leftmost digit is non-zero. Normalized floating-point numbers provide the maximum number of possible precisions for the floating-point number.



Example: The number 450 is normalized, but the number 000045 is not normalized.

$0.0035 * 10^5$ is un-normalized, whereas $0.35000 * 10^3$ is normalized.

Eight bit numbers are not normalized because of leading 0s. These numbers can be normalized by shifting three places to the left to obtain 10010000 in a number.

Normalized Version representation is shown below:

Value represented = $+1.0110... * 2^6$ is a normalized value

Un-normalized version representation is shown below:

Value represented = $+0.0010110... * 2^9$ is an unsigned value. There is no implicit point to the left of the binary point.

Single Precision and Double Precision

As depicted in the figure 3.2, the 23 bit, which represents the **mantissa** whose most significant bit is always equal to 1, is normalized. This bit is immediate to the left of the binary point. Hence, the 23 bits stored in the **M** field represent the fractional part of the **mantissa** and this 32 bit representation is called the single precision because it occupies a single 32 bit word.

Double precision floating point numbers are used to improve the accuracy and range of floating - point numbers. The excess -1023 exponent **E'** has the range $1 \leq E' \leq 2064$ for normal values. Thus, the 53-bit **mantissa** provides a precision equivalent to 16 decimal digits.

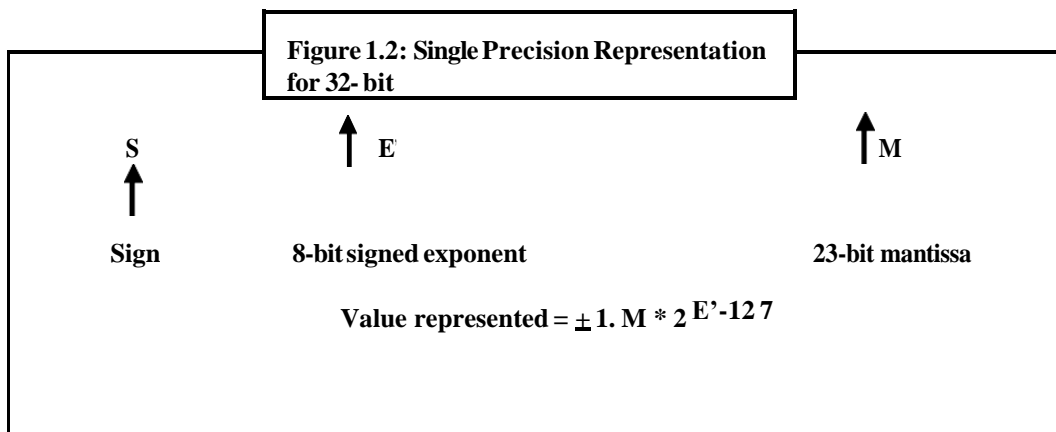
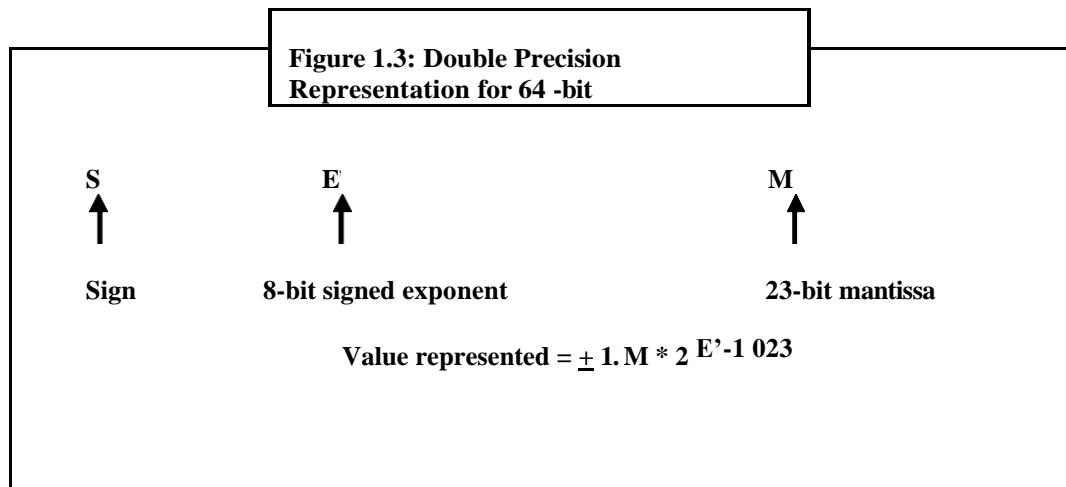
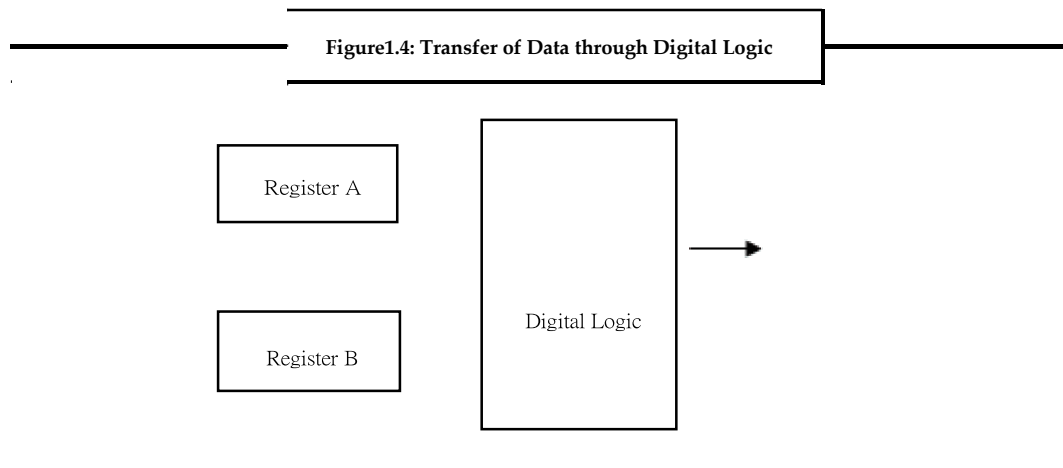


Figure 1.3 depicts double precision representation for 64-bit.



Register Transfer

Registers refer to the storage space that holds the data and instructions. To transmit data and instructions from one register to another register, memory to register and memory to memory, the register transfer method is used. This register helps in the transfer of data and instructions between memory and processors to perform the specified tasks. The transfer of data is more concise, precise, and is provided in an organized manner. Digital logic is used to process the data. Figure depicts the transfer of data through digital logic.



There are two types of register transfers. They are:

1. Bus transfer
2. Memory transfer

Bus Transfer

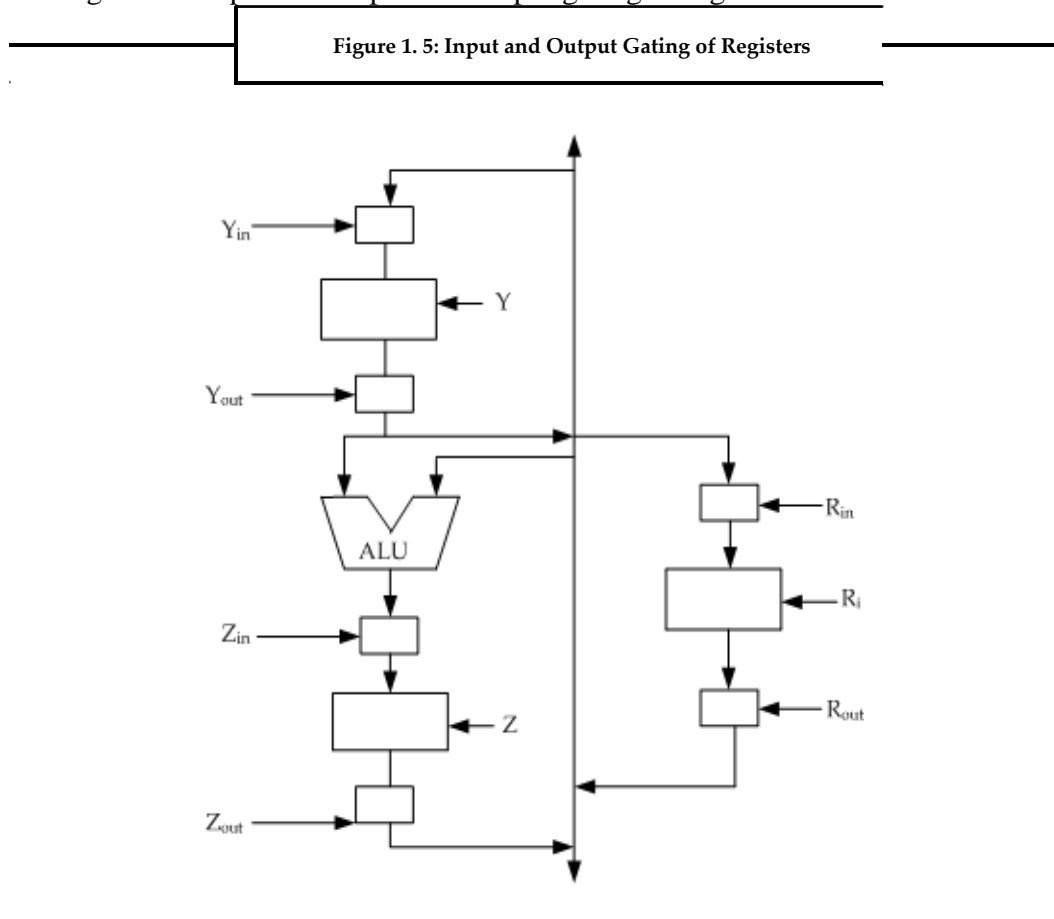
The most efficient way to transfer data is by using a common bus system, which is configured using common bus registers in a multiple register. The structure of the bus consists of a set of lines. These lines are registers of one bit each that transfer only one data at a time. The data transfer is controlled by the control signals. Control signals determine which register is to be selected during each register transfer. To construct a common bus system, two methods are used:

1. Using multiplexer
2. Using three states bus buffers

Using Multiplexer

A common bus can be constructed using a multiplexer. Multiplexer helps in selecting the source register to place the binary information on the bus. The bus register has input and output gating controlled by control signals.

The figure 1.5 depicts the input and output gating of registers.



In figure 1.5:

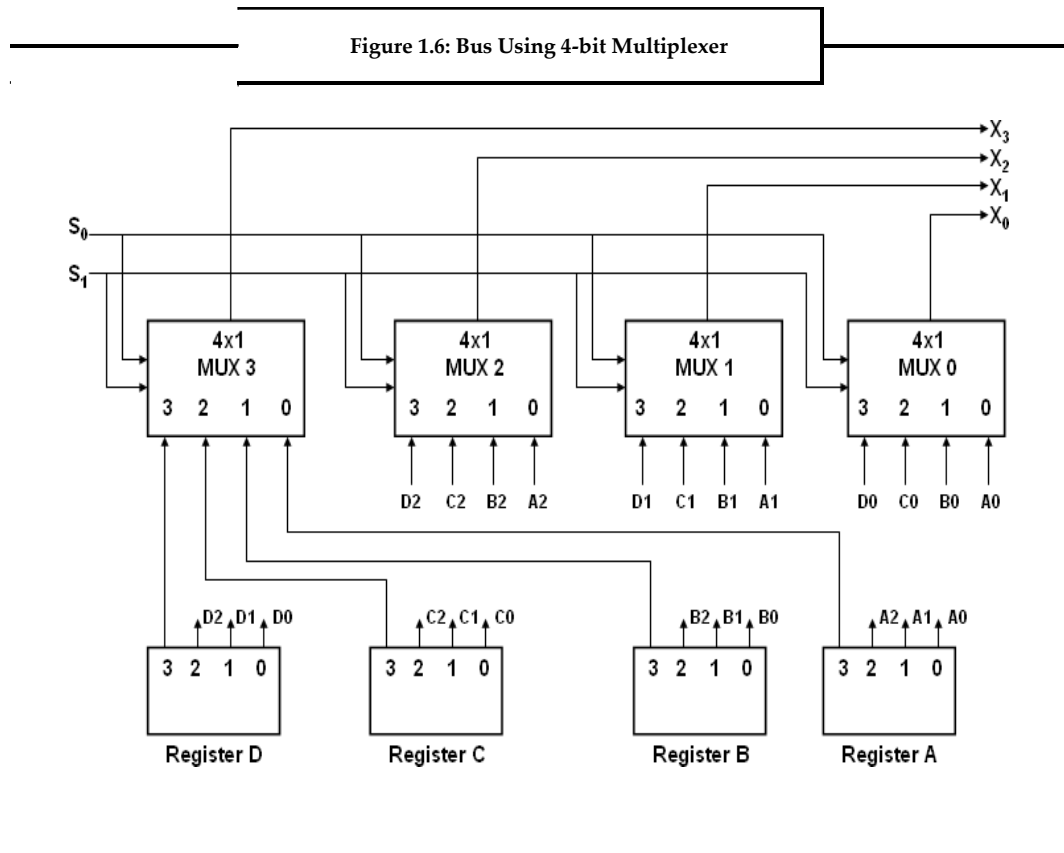
1. Ri is the register and Rin and Rout are the input and output gating signals of Ri.
2. Z is the register and Zin and Zout are the input and output gating signal of register Z.
3. Y is the register and Yin and Yout are the input and output signals of Y.

The figure 3.5 illustrates input and output gating, that is, the switches controlled by control signals. In the figure, **Rin** and **Rout** are the input and output gating of the register **Ri**. When the signal is **ON**, **Ri** is set to **1** and when the signal is **OFF**, **Ri** is set to **0**.

When the input gating Rin is set to 1, the data is loaded into the register bus Ri available on the common bus. Similarly, when Rout is set to 1, the contents of the register Ri are placed on the data bus. Hence, they are known as input enabled and output enabled signals. The operation that takes place within the processor is in sync with the clock pulse.

Notes

The figure 1.6 depicts a bus that uses a 4-bit multiplexer.



Source: Computer organization and architecture by Krishnananda.

In the figure 1.6, the multiplexer with four bit register is illustrated. Each register is of four bits from 0 to 3 and the bus carries 4*1 multiplexers with four data inputs from 0 to 3 through X0 and X3. Here, S1 and S0 are the selection inputs for all the four multiplexers. The connection is made from the output of the registers through the input of the multiplexers.

In the figure 1.6, the output 1 of the register A is connected to the input 0 of multiplexer 1. Thus, the selected data is loaded into the registers and placed on the data bus to perform the operations by using the register bus transfer. Table 1.3 depicts the register that is selected on the basis of the two switches S0 and S1.

Table 1.3: Register Selected On the Basis of the Two Switches S0 and S1

S ₀	S ₁	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

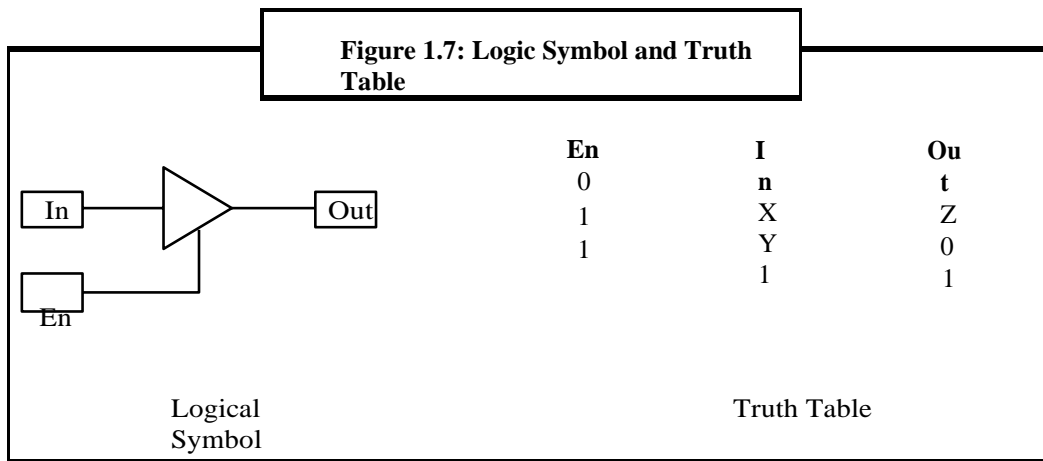
Three-State Buffers

Three-state buffers can also be used to construct a common bus. The buffer is a region of the memory, which is inserted in between the other devices to prevent multiple interactions and to match the impedance. The buffers supply additional drive and relay capabilities to the bus registers.

The three-state buffers are based on the three states, 1, 0, and the open circuit. These three states describe that:

1. The logic 0 and 1 are the two signals equivalent to the ones in conventional gate.
2. The high impedance state means that it does not have the logic significance and the output is disconnected. It also behaves like an open circuit.
3. These three-state gates can perform any conventional logic AND or NAND, OR or NOR, but the most commonly used gate is the buffer gate in the bus register system.

Figure 1.7 depicts the logic symbols and the associated truth table.



According to figure 1.7:

1. When the output is enabled and the control input is equal to 1, the logic gate acts like buffer with the output equal to the input.
2. When the input given is 0, the gate goes to high impedance state Z and the output is disabled.
3. The impedance in three-state buffers connects all the outputs with a wire to form a common bus line and does not endanger the loading effect.
4. The truth table explains that when some input is given and the gate is disabled, it results in high impedance.
5. When the gate is enabled with some input given, then the output results are not in disabled mode.
6. When the gate is enabled with input as 1, the output is equal to 1.

Memory Transfer

The read operation in memory transfer is defined as the transfer of data from the address register (AR) with the selected word M for the memory into the memory buffer register (MBR).

[AR]M MBR=Read Operation

The control signal of the **read** operation initiates the read operation. The read operation statement causes the information transfer from the selected memory register M into the MBR.

The memory transfer in write operation is defined as the transfer of data from the memory buffer register (MBR) to the address register (AR) with the selected word M for the memory.

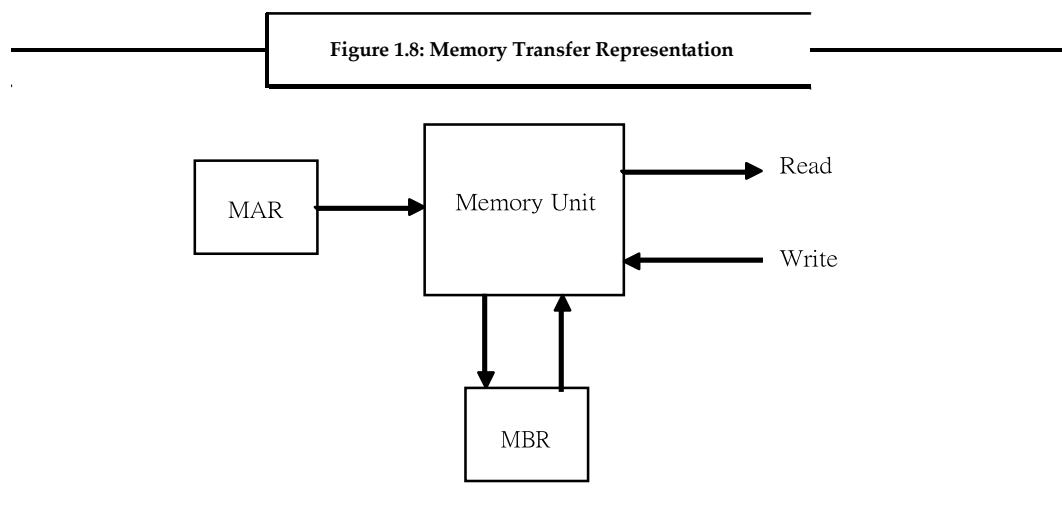
MBR M[AR]=Write Operation.

The control signal of the write operation initiates the write operation. The write

operation statement causes the information transfer from the MBR into the selected memory register by the address present in the memory M[AR].

To perform either read or write operation, first the memory register (M) must be selected by a specified address.

The figure 1.8 depicts the memory transfer representation.



The figure 1.8 shows that the memory unit is used to transfer the information from the memory address register and memory buffer register to perform read and write operations in the memory transfer.

1.3 Microoperation

Microoperations specify the transfer of data to storage. The following sections deals with logic microoperation, shift microoperation and arithmetic logic shift unit.

Logic Microoperation

The logic microoperations for the string of bits stored in registers specify the binary operations. This logic microoperation considers each bit of register separately and treats them as binary variables.

The two registers with exclusive-or microoperation are symbolized as: A : R1 R1 R2.

The control variable A =1, specifies that the microoperations are to be executed on the individual bits.



Example: Consider registers R1 and R2 each having 4-bits binary numbers.

$$\begin{array}{r}
 1010 \text{ R1} \\
 \underline{1100 \text{ R2}} \\
 0110 \text{ A=1}
 \end{array}$$

In this example, 1010 is the content of register R1, 1100 is the content of R2. After the execution of the microoperation, the content of R1 is equal to the bit-by-bit XOR operation on pairs of bits in R2 and the previous value of R1.

Logical symbols like OR, AND, and their complements are adopted to differentiate them from the corresponding symbols that are used to represent Boolean functions.

The symbol \wedge is used to denote **AND** microoperation, the symbol \vee is used to denote **OR** microoperation, and to denote the complement of **AND** and **OR**, **1's complement** is taken with a **bar** on top of the symbol that denotes the register name.

There are some applications of logic microoperations. Some of the applications of microoperations are:

1. It manipulates a part of the memory word or individual bits stored in the registers.
2. It is used to change, delete, and insert new bit values within the memory.

3. It operates on Selective Set,

$$A \leftarrow A \vee B,$$



Exempl	101	A
e:	0	
	110	B
	0	
	111	A
	0	

As shown in the above example, the microoperations set a group of bit values.

In the bit register, the microoperation performs the following function:

1. It changes the values in register A to correspond to the values in register B, that is, in places where the register B holds the value 1, the corresponding bits of 'A' are also changed to 1.
2. It operates on Selective • Complement,

$$A \leftarrow A \oplus B,$$



Example:	1010	A
	1100	B
	0110	A

In this example, microoperation complements the 1 in bit register A, where there are corresponding 1 bits in register B. It does not affect the 0 bits in the register.

3. It operates on Selective Clear,

$$A \leftarrow A \wedge B,$$



Example:	1010	A
	1100	B
	0010	A

In this example, microoperation clears the 0 in bit register A, where there are corresponding 1 bits in register B. It affects the 0 bits in the register.

4. Microoperation operates on Selective • Mask,

$$A \leftarrow A \wedge B,$$



Example:	1010	A
	1100	B
	0010	A

It masks the group of bit values. It is same as that of selective clear, but the bits of 'A' are cleared only when there are corresponding 0 bits in 'B'.

Notes

1.4.2 Shift Microoperation

The transfer of data is done using the shift operation. Shift Microoperation is used in logic, arithmetic and other data processing conventions in conjunction. The content of bits from the registers is shifted to left and right. After the bits are shifted, the flip-flops receive the information from the serial input.

During the left shift operation, the bits are shifted to the rightmost position by the serial input operations. During the right shift operation, the bits are shifted to the leftmost position by the serial input operations.

Many arithmetic operations like multiplications and divisions require shifting of the operands. For general operands, logical shift is used, which preserves the sign of the number. The table 1.4 shows the different shift operations.

Table 1.4: Different Shift Operations

Register	Description
R	Lshl R Shift left register
R	Lshr R Shift right register
R	cil R Circular left shift register
R	cir R Circular right shift register
R	ashl R Arithmetic left shift register
R	ashr R Arithmetic right shift register

There are three different types of shift operations. They are:

1. Logical shift operation
2. Circular shift operation
3. Arithmetic shift operation

The following section deals with the explanation of each of these shift operations.

Logical Shift Operation

There are two logical shift operations **Lshl** for the logical left shift register and **Lshr** for the logical right shift register. Logical shift instructions shift an operand by a number of bit positions specified in the logical shift instruction.

General form of logical left shift

operation Lshl count, R

In this form of logical shift operation, the count is the immediate operand or it may be given in a processor register. When bits are shifted left, the rightmost bit is filled with zeros. The bits shifted from the MSB position are passed through a carry flag **C** and are lost. **R** is the processor register.

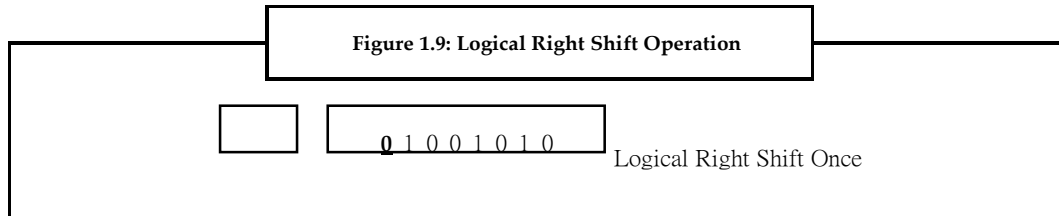
General form of logical right shift

operation Lshr count, R

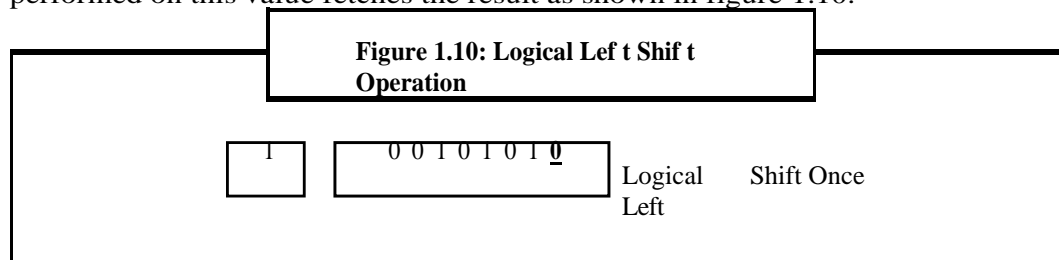
In this form of logical shift operation, the leftmost positions are filled with zeros when bits are shifted to the right. The bits that are shifted from the MSB position and passed through a carry flag **C** are lost. Involving carry flag in the shift operations helps in performing operations on large numbers.

Jaipur Engineering College and Research Centre
 Computer Architecture and Organization (6CS4-04)
 Session 2020-21

Suppose the register holds the value 10010100, the logical right shift operation performed on this value fetches the result as shown in figure 1.9.



Suppose the register holds the value 01010101, the logical left shift operation performed on this value fetches the result as shown in figure 1.10.



The figure 1.9 depicts a logical right shift by 1 bit and the figure 1.10 depicts a logical left shift by 1 bit.

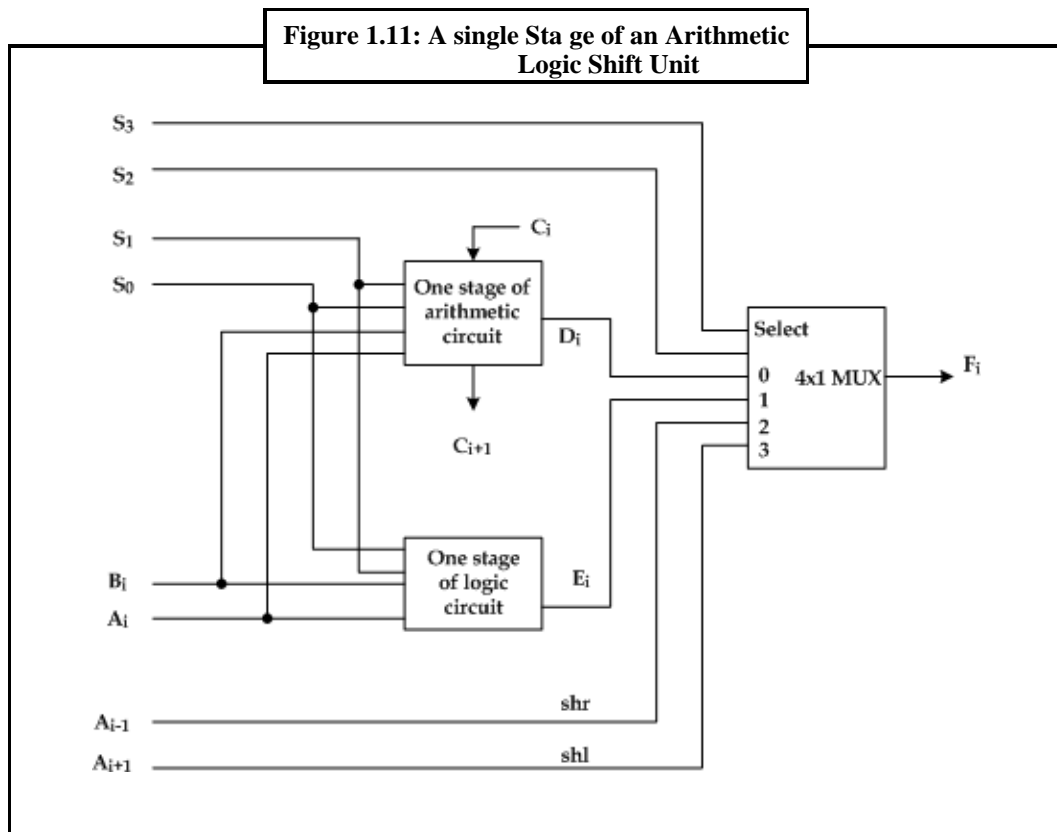
1.4.3 Arithmetic Logic Shift Unit

Computer systems make use of a number of storage registers that are connected to common operational unit. This unit is called as an arithmetic and logic unit (ALU). The contents of the specified source register form the inputs of the ALU. The ALU performs an operation and transfers the result to the destination register. The ALU comprises combination of circuits that transfers the content from source register to destination register in one clock pulse period. Usually, the shift microoperation is performed in a separate unit. However, sometimes the shift unit becomes part of the overall unit. The arithmetic, logic, and shift circuits can be incorporated into one ALU with common selection variable.

The figure 1.11 shows a single stage of an arithmetic logic shift unit. The subscript 'i' indicates a typical stage. A_i and B_i are the two inputs assigned to arithmetic as well as logic units. A microoperation is selected with inputs S_1 and S_0 . E_i is the arithmetic output and D_i is the logic output. Multiplexer 4x1 chooses between arithmetic output and logic output. S_3 and S_2 are the inputs used to select data in the multiplexer. The other inputs to the multiplexer are A_{i-1} for the shift-right operation and A_{i+1} for the shift-left operation.

Notes

Figure 1.11 depicts a single stage of the arithmetic logic shift operation.



This circuit is repeated 'n' number of times to obtain an 'n' bit ALU. The output carry denoted by C_{i+1} of one stage is connected to the input carry denoted by C_i of the next stage. The connection between the output carry and the input carry happens in a sequential manner. The input carry of the first stage is denoted by C_i and helps in selecting the variable for the arithmetic operation.

Jaipur Engineering College and Research Centre
 Computer Architecture and Organization (6CS4-04)
 Session 2020-21

Notes

The table 1.5 gives the 14 operation of the ALU.

Table 1.5: ALU Function Table

Select Operation	S ₃	S ₂	S ₁	C _i	Operation Performed	Description
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
1	0	0	0	0	$F = A + B$	Addition
1	0	0	0	1	$F = A + B + 1$	Add with carry
0	1	0	0	0	$F = A - B$	Subtract with borrow
0	1	0	0	1	$F = A - B - 1$	Subtraction
1	1	0	0	0	$F = A - 1$	Decrement A
1	1	0	0	1	$F = A$	Transfer A
0	0	1	0	x	$F = A \wedge B$	AND
1	0	1	0	x	$F = A \vee B$	OR
0	1	1	0	x	$F = A \oplus B$	XOR
1	1	1	0	x	$F = \overline{A}$	Complement A
x	X	0	1	x	$F = shrA$	Shift right A into F
x	X	1	1	x	$F = shlA$	Shift left A into F

The first eight operations relate to arithmetic operations and are selected based on the variables S₃, S₂ = 00. The next four operations relate to logical operations and are selected based on variables S₃, S₂ = 01.