COA Assignment U1

1. a) Explain the logical Micro operations.
   b) Discuss about Arithmetic Logic Shift Unit.

1. a) Logic Micro-operations

- Logic micro-operations specify binary operations for bit strings in registers.
- Each bit in the register is treated as a binary variable.
- Example: Exclusive-OR micro-operation between registers R1 and R2 when the control variable P = 1.

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

List of Logic Microoperations

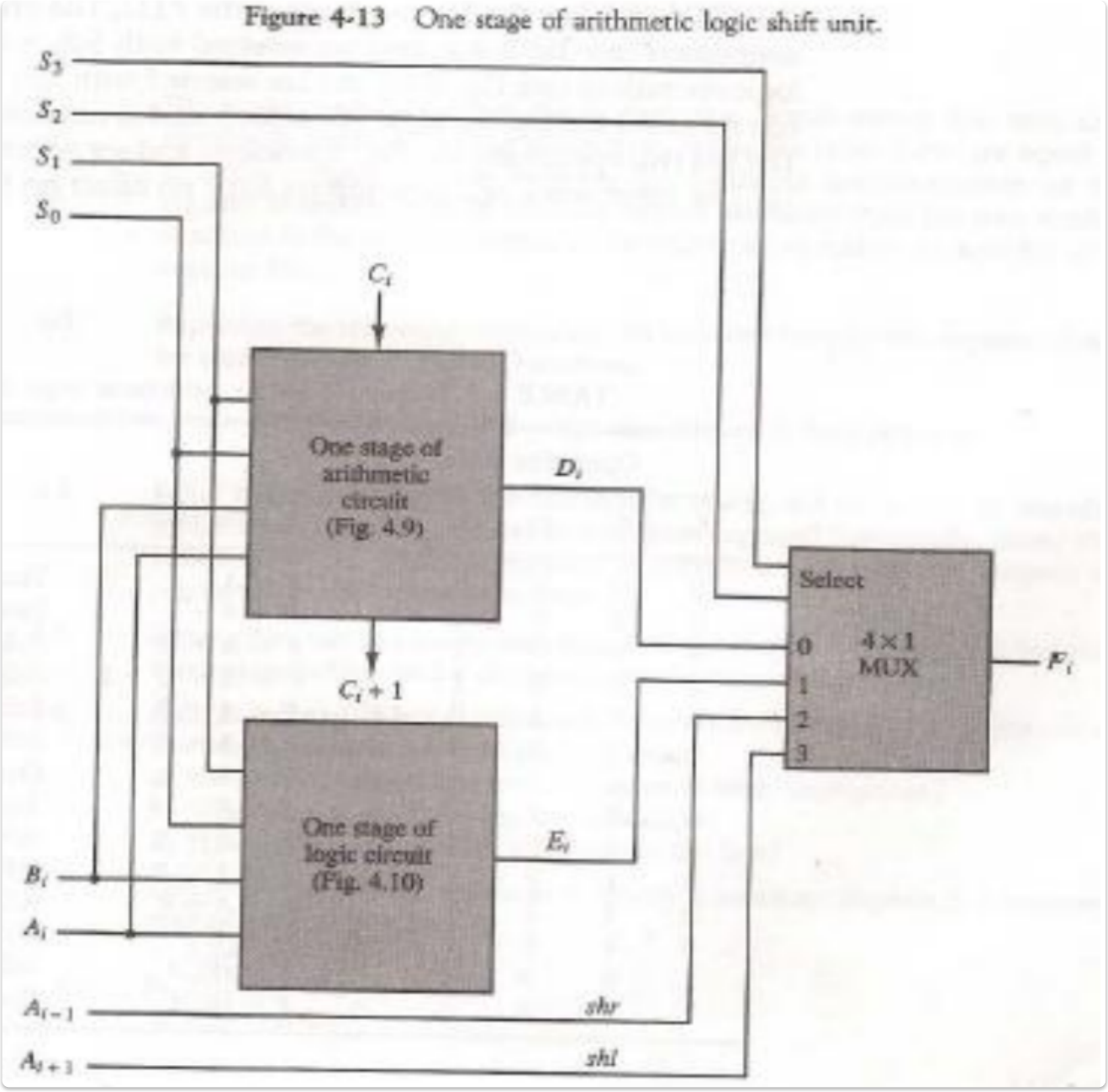
- 16 different logic operations for two binary variables.
- Derived from truth tables with two binary variables (see Table 4-5).
- Expressions in Table 4-6 show Boolean functions for variables x and y.
- Logic micro-operations derived by replacing x with content of register A and y with content of register B.

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

1. b) Arithmetic Logic Shift Unit

Figure 4-13 One stage of arithmetic logic shift unit.



- Computer systems use an Arithmetic Logic Unit (ALU) for microoperations.
- ALU is a combinational circuit connected to storage registers.
- Shift microoperations can be part of the ALU.
- One stage of an arithmetic logic shift unit shown in Fig. 4-13.
- Selection variables S1, S0, S3, S2, and Cin determine the operation.
- Table 4-8 lists 14 operations of the ALU (8 arithmetic, 4 logic, 2 shift).

TABLE 4-8 Function Table for Arithmetic Logic Shift Unit

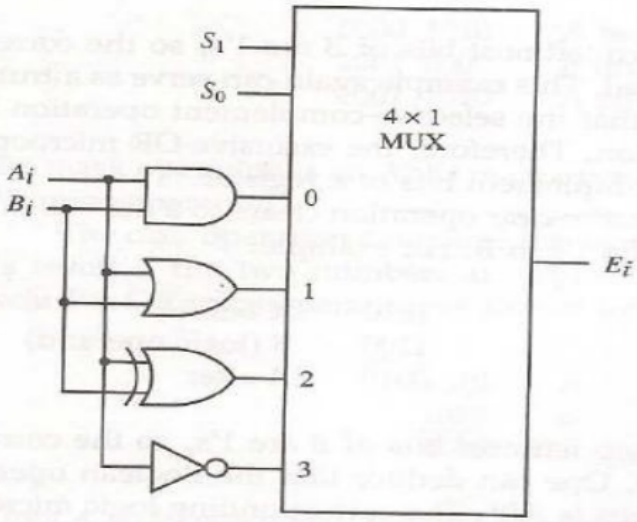
Operation select						
S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	Complement A
1	0	×	×	×	$F =$ shr A	Shift right A into F
1	1	×	×	×	$F =$ shl A	Shift left A into F

2. a) Explain the hardware implementation of Logic Micro operations.
   b) Draw the instruction code Format.

2. a) Hardware Implementation

- Logic microoperations hardware implementation uses logic gates for each bit or pair of bits in registers.
- Most computers use four basic logic microoperations: AND, OR, XOR (exclusive-OR), and complement.
- Circuit example for generating basic logic microoperations using gates and a multiplexer.

Figure 4-10 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) Function table

Some Applications

- Logic micro-operations useful for manipulating individual bits or portions of a word in a register.
- Examples of applications:
  - Selective set
  - Selective complement
  - Selective clear
  - Mask
  - Insert
  - Clear

Selective Set

- Sets to 1 the bits in register A where there are corresponding 1's in register B.
- OR microoperation used for selective setting.

Selective Complement

- Complements bits in A where there are corresponding 1's in B.
- Exclusive-OR microoperation used for selective complementing.

Selective Clear

- Clears to 0 the bits in A where there are corresponding 1's in B.
- Corresponding logic microoperation is AND.

Mask

- Similar to selective clear, but clears bits in A only where there are corresponding 0's in B.
- Mask operation is an AND microoperation.

Insert

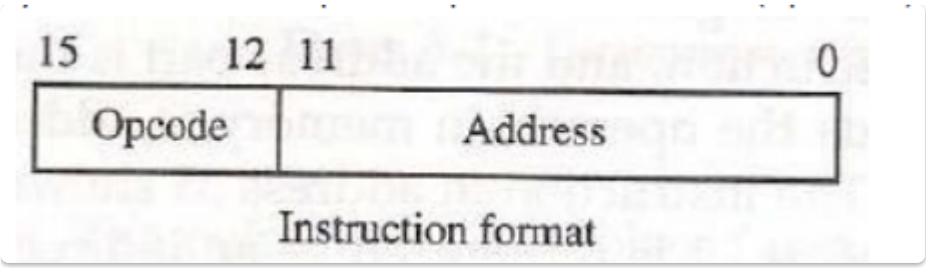
- Inserts a new value into a group of bits by masking unwanted bits and ORing with the required value.
- Example provided for inserting a value into a register.

Clear

- Compares words in A and B, produces all 0's result if the two numbers are equal.
- Achieved by an exclusive-OR microoperation.

2. b) Instruction Codes

> ## Instruction Codes
>
> - The organization of the computer is defined by its internal registers, timing and control structure, and the set of instructions.
> - Instructions are stored in memory and executed based on their opcode and operand.
>
> ### Instruction Code Format

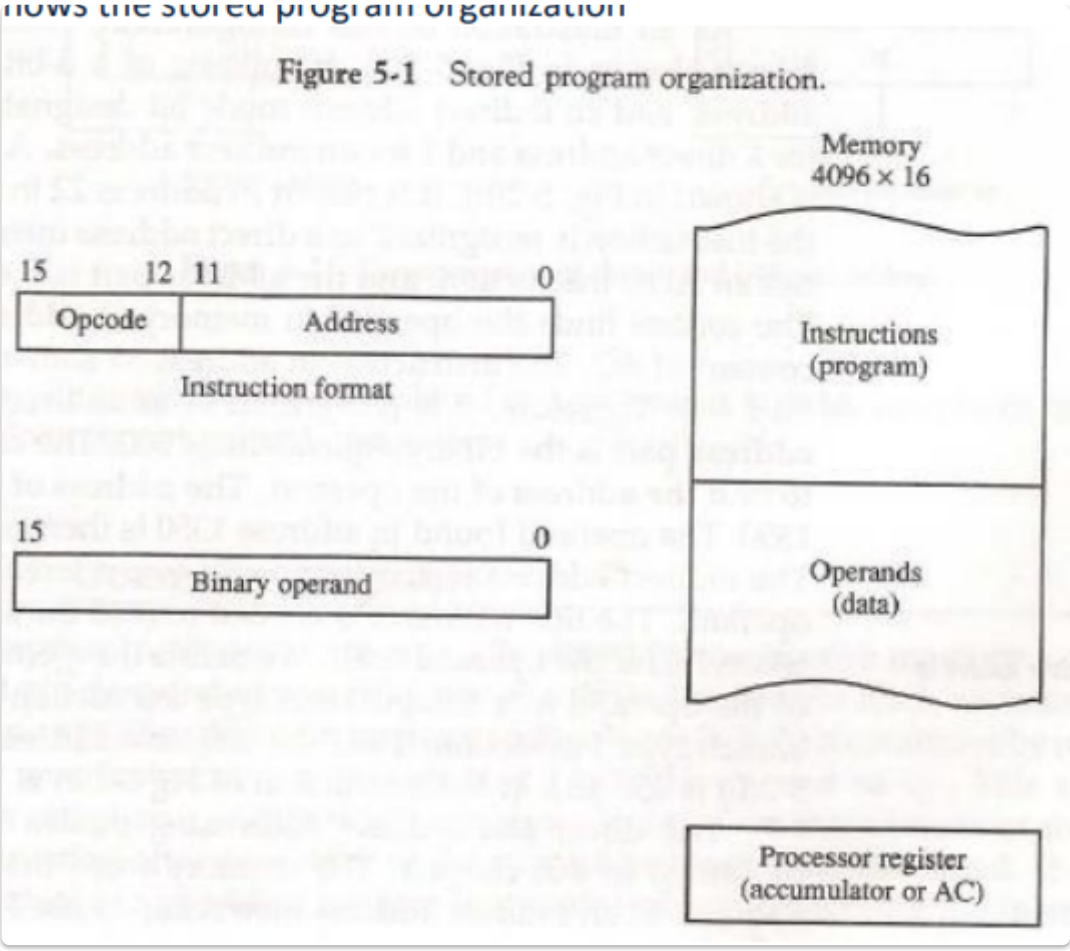


Instruction format

- Instruction codes are divided into two parts: Opcode and Address (Operand).
- **Operation Code (Opcode):**
  - Specifies the operation (add, subtract, etc.).
  - Number of bits required depends on the number of operations.
- **Address (Operand):**
  - Specifies the location of operands (registers or memory words).

Stored Program Organization

- Stored program organization allows storing instructions in memory and executing them sequentially.
- Example instruction format: 16-bit word with 3-bit opcode and 12-bit address.



Addressing of Operand

- Immediate Operand: Operand is part of the instruction.
- Direct Address: Operand is specified by the address part of the instruction.
- Indirect Address: Address of the operand is found in another memory location.
- Addressing modes are determined by a mode bit (0 for direct, 1 for indirect).

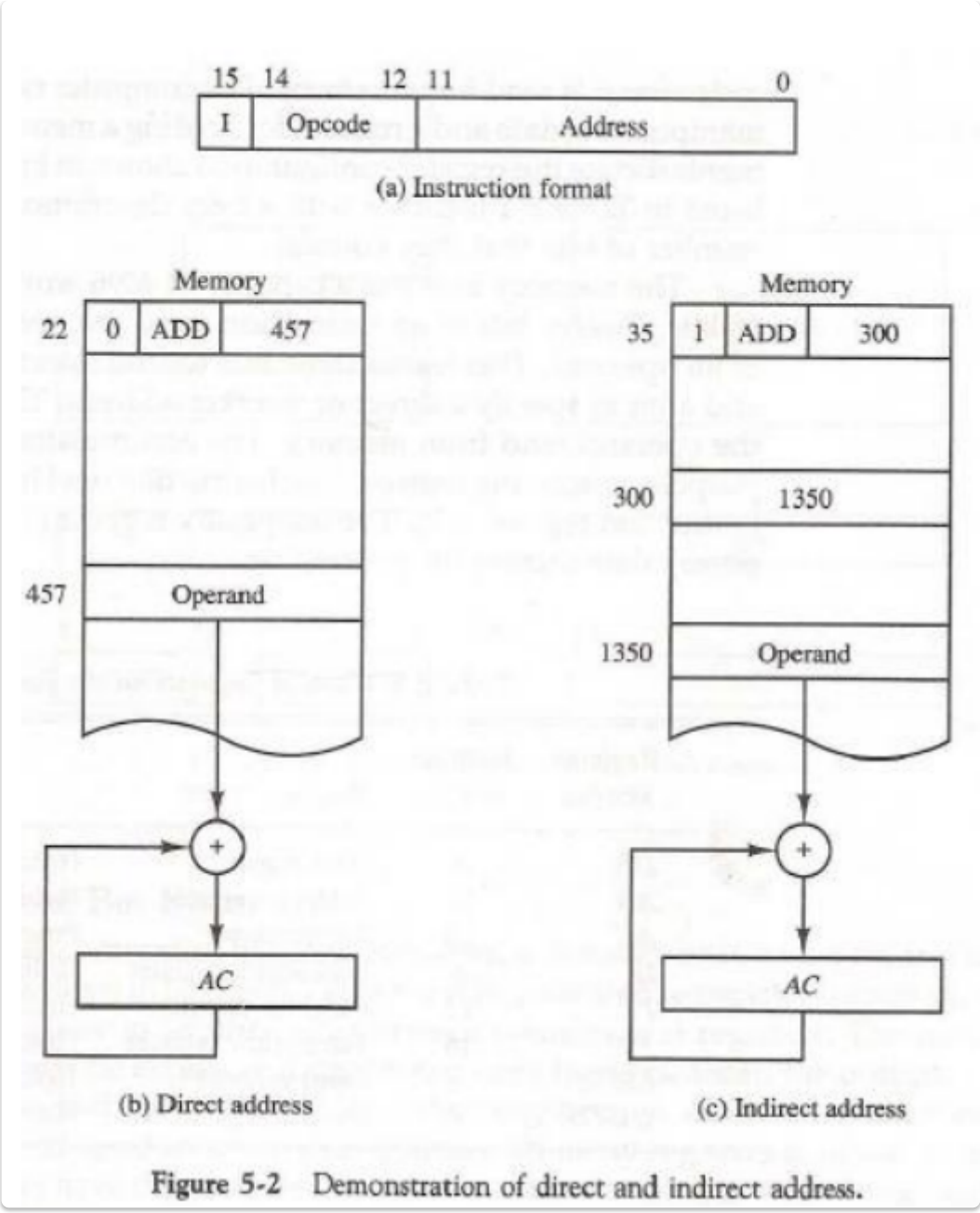


Figure 5-2 Demonstration of direct and indirect address.

3. a) Explain the computer registers.
b) Explain the timing and control signal with help of timing diagram

3. a)

Computer Registers

- Registers play a crucial role in instruction sequencing and data manipulation.

- Common registers include Accumulator (AC), Instruction Register (IR), Program Counter (PC), etc.

TABLE 5-1 List of Registers for the Basic Computer

Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

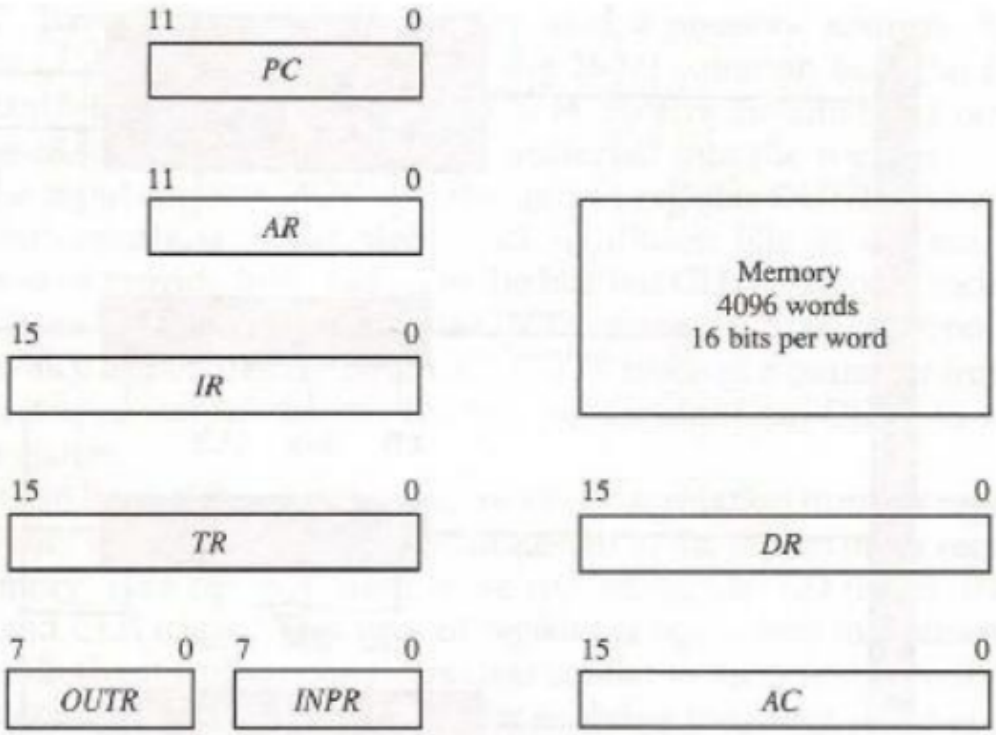


Figure 5-3 Basic computer registers and memory.

Common Bus System

- A common bus system efficiently transfers information between registers and memory.
- Selection variables determine which register's output is connected to the bus.

- Load (LD), Increment (INR), and Clear (CLR) control inputs are used for register operations.

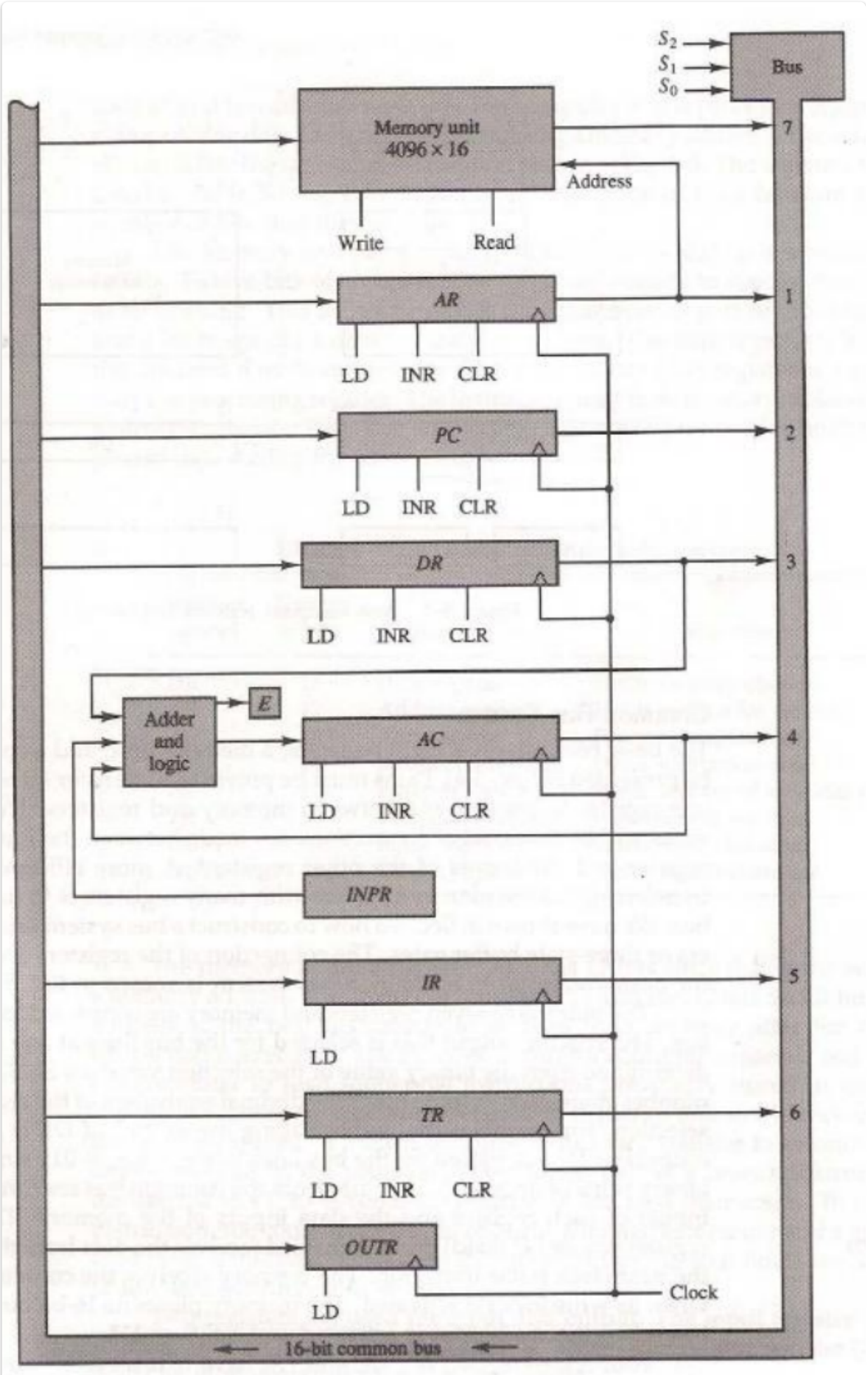


Figure 5-4 Basic computer registers connected to a common bus.

Example Registers

- Example registers include Data Register (DR), Temporary Register (TR), Memory Address Register (AR), etc.
- Input and output registers (INPR and OUTR) communicate with external devices.

Adder and Logic Circuit

- The Adder and Logic Circuit perform arithmetic operations and logical functions.
- Multiple inputs, including content from registers, enable versatile operations.
- Operations can be executed in parallel during the same clock cycle.

3. b)

Timing and Control

- The timing for all registers in the basic computer is controlled by a master clock generator.
- Clock pulses are applied to all flip-flops and registers in the system when enabled by a control signal.
- Control signals, generated in the control unit, control multiplexers, processor registers, and microoperations.

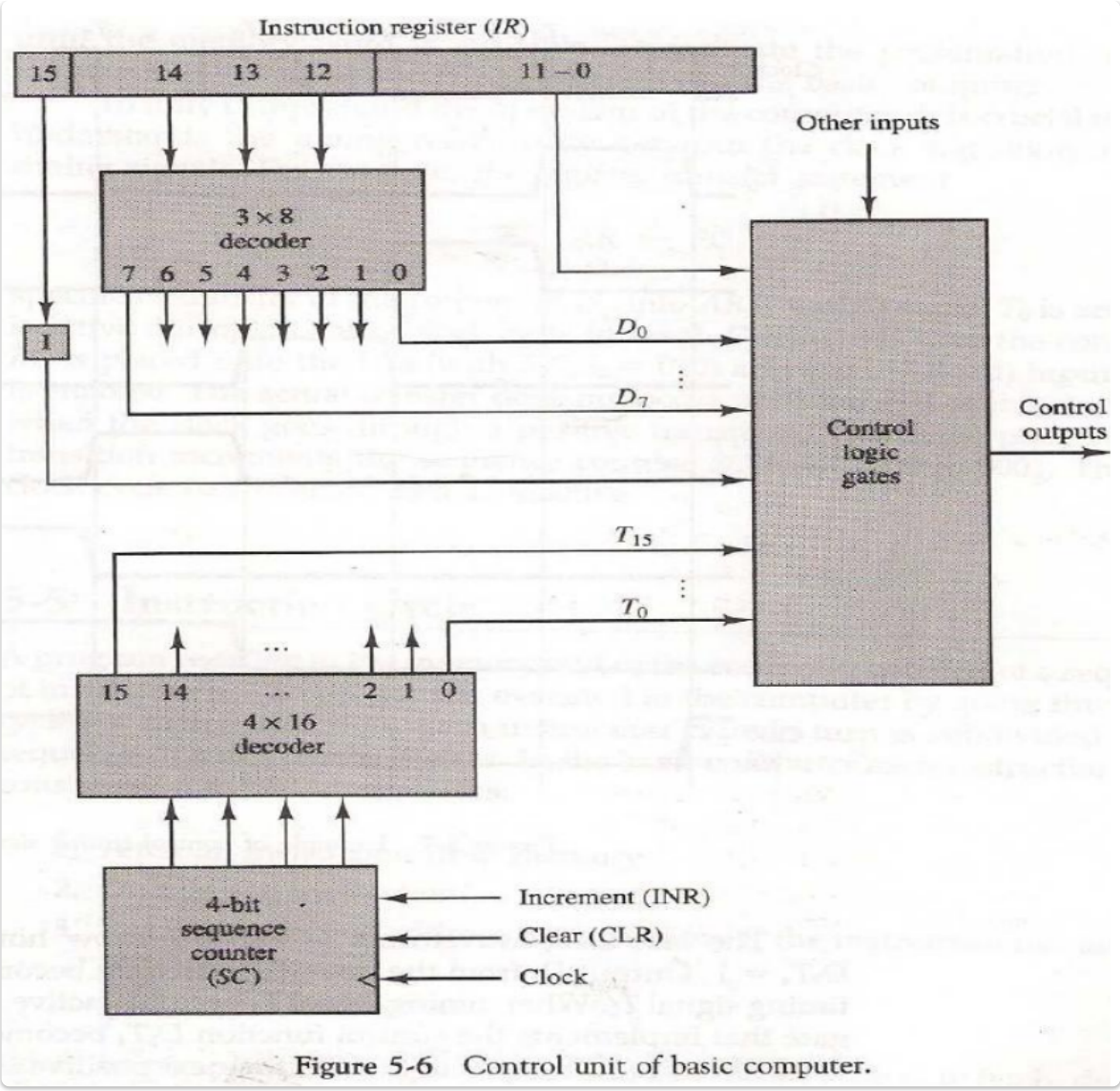


Figure 5-6 Control unit of basic computer.

Control Organizations

Hardwired Control

- Implemented with gates, flip-flops, decoders, and other digital circuits.
- Control information is stored in the control memory, programmed for the required sequence of microoperations.
- Advantages include optimization for faster operation.
- Changes in design require modifications in wiring among components.

Microprogrammed Control

- Control logic is stored in control memory.

- The control memory is programmed to initiate the required sequence of microoperations.
- Offers flexibility for modifications by updating the microprogram.

Hardwired Control Unit

- Block diagram includes two decoders, a sequence counter, and control logic gates.
- Instruction from memory is placed in the instruction register (IR), divided into I bit, operation code, and bits 0-11.
- Operation code (bits 12-14) is decoded with a 3x8 decoder, producing outputs D0 through D7.
- Bit 15 of the instruction is transferred to a flip-flop (I).
- Bits 0-11 are applied to control logic gates.
- Sequence counter (SC) can count in binary from 0 through 15.
- Outputs of SC are decoded into 16 timing signals (T0-T15).
- SC can be incremented or cleared synchronously.
- Timing diagram shows the time relationship of control signals.

Timing Diagram

- SC responds to the positive transition of the clock.
- Initially, the CLR input of SC is active.
- SC is incremented with every positive clock transition unless CLR input is active.
- The timing diagram illustrates the sequence of timing signals and the clearing of SC based on control signals.

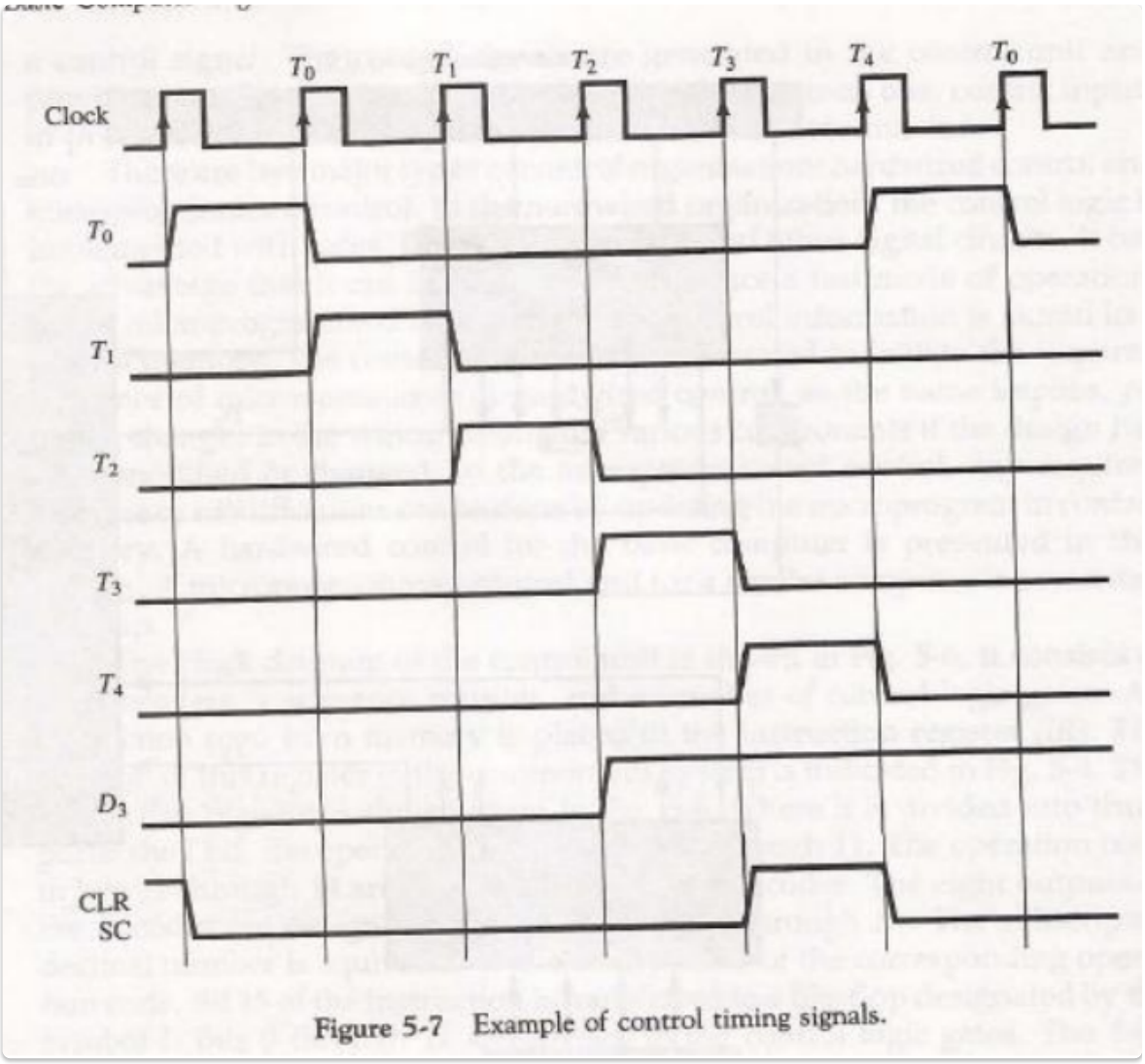


Figure 5-7 Example of control timing signals.