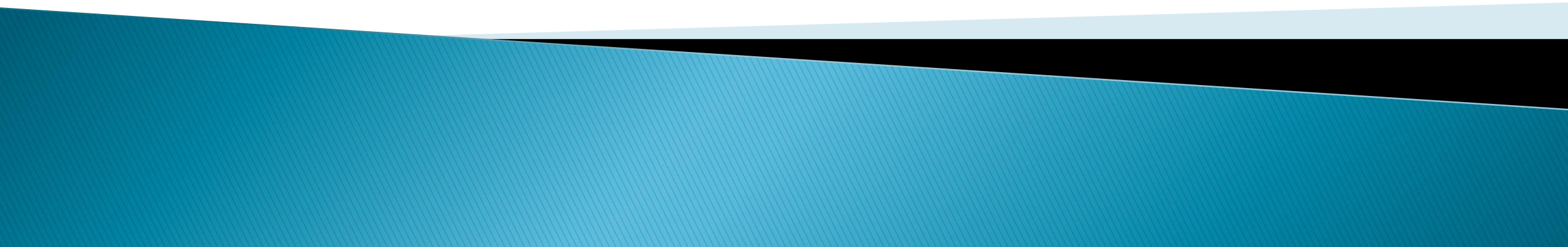


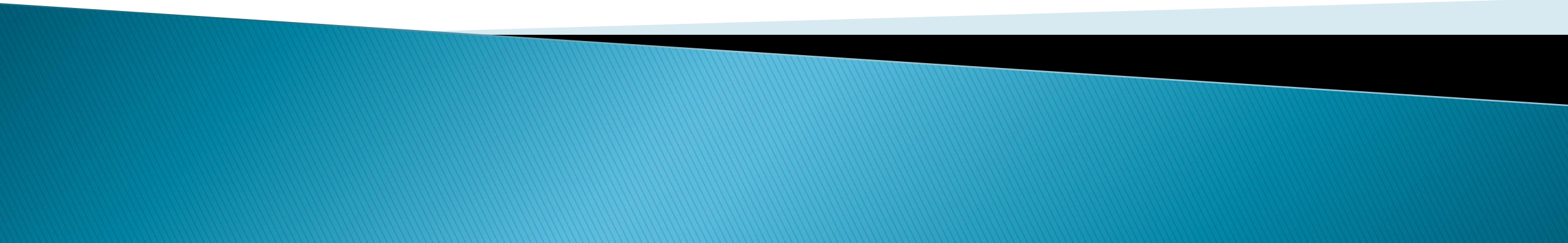
Basic Computer Architecture

Unit 3




- ▶ **Computer Architecture**
 - **Computer Architecture** refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program.
- ▶ **Computer Organization**
 - **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications.


History of Computer Architecture




Computer Architecture

- The design ,arrangement, construction or organization of the different parts of a computer system is known as Computer Architecture
 - Computer architecture defines the way hardware components are connected together to form a computer system
 - Computer architecture deals with high level design issues such as Instruction sets, Addressing modes, Data types, Cache optimization etc.
 - It helps us to understand the functionalities of a system.
 - It acts as an interface between hardware and software
- 


Brief History of Computer Architecture

- First Generation (1945-1958)
 - Vacuum Tubes
 - Machine code
 - CPU was unique to that machine
 - Use of drum or magnetic core memory, programs are loaded using paper tape or punch cards
 - 2 Kb memory, 10 KIPS(thousands instruction per second)
- 


Architecture for a Computing Machine

- Based on the storage and signaling of instruction and data.
 - **Harvard architecture:** uses physically separate storage and signal pathways for instructions and data. (Harvard Mark I). it stored instructions on punched tape and data in relay latches.
 - **Von Neumann architecture:** a single storage structure to hold both set of instructions and data Aka stored program computers.
- 


Second Generation(1958-1964)

- Transistors – small, low-power, low-cost, more reliable than vacuum tube
 - Magnetic core memory
 - Reduced computational time to microseconds
 - High level languages
 - First OS handled one program at a time
- 


Third Generation(1964-1974)

- Integrated circuits, combined thousands of transistors on a single chip using LSI
 - Semiconductor Memory
 - Timesharing, Graphics and Structured programming
 - 2 Mb memory, 5 MIPS
 - Use of Cache memory
- 


Fourth Generation(1974- Present)

- Microprocessor, combined millions of transistors using VLSI and ULSI technology
 - Single Chip processor and the single board computer emerged
 - Creation of the Personal computer(PC)
 - Object oriented programming and Artificial intelligence concept
 - Intel 4004 was the worlds first universal microprocessor.
- 


Self Study !!

- ENIAC
 - Whirlwind computer
 - UNIVAC
 - IBM 7090
 - IBM System/360
 - Intel 4004
- 

Microoperations


- ▶ The operation executed on data stored in registers are called micro operations.
 - ▶ A micro operation is elementary operation performed on the information stored in one or more registers.
 - ▶ The result of operation may replace the previous binary information of register or may be transferred to other register.
 - ▶ Example: shift, count, clear and load.
- 

Instruction Codes

- ▶ Basic computer is smaller in comparison to commercial computer.
 - ▶ Organization of the computer is defined by its internal register, the timing and control structure and set of instructions and its uses.
 - ▶ The general purpose digital computer is capable of executing various **sequence of micro operations**. It performs on data stored in registers
 - ▶ The user of computer can control the process by means of a program which is set of instructions that specify the operations and operand.
- 

Instruction Codes

▶ Computer instruction

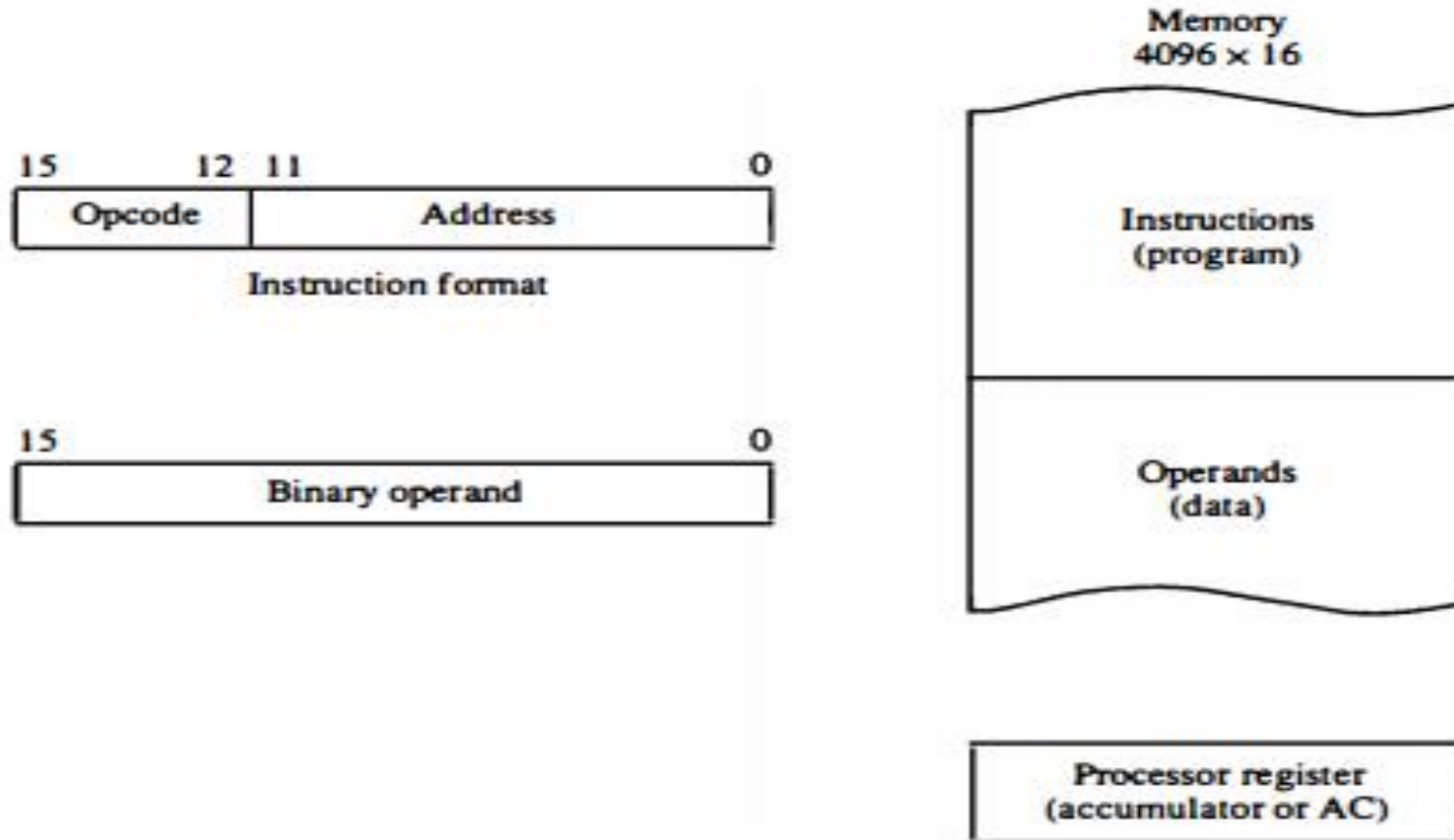
- Is a binary code that **specifies a sequence of micro operations for computer**
 - Instruction code and data are **stored in memory**
 - Computer read each instruction from memory and keep in register
 - Control **interprets** the binary code of instructions and proceed to execute it by issuing a sequence of micro operations.
 - Every computer has its own unique instruction set.
- 

Instruction Codes


- ▶ Instruction code is group of bits that instruct computer to perform a specific operations.
- ▶ It is usually divided into 2 parts:
 - **Operation code**
 - If n bits are in operation code, then there are 2^n distinct operations.
 - Operation can be add, subtract, multiply, shift, complement etc.
 - The control unit receives the instruction from memory and interprets the operation code bits and then issues a control signals to initiate microoperations on internal computer registers.
 - **Operand code**

Stored Program Organization

Figure 5-1 Stored program organization.




Addressing Modes of basic computer (Direct and Indirect addressing mode)

- ▶ When the second part of an instruction code specifies the address of an operand, the instruction is said to have a **Direct address**.
 - ▶ When the second part of an instruction code specifies the address of a memory word in which the address of the operand is available, the instruction is said to have **Indirect address**.
- 

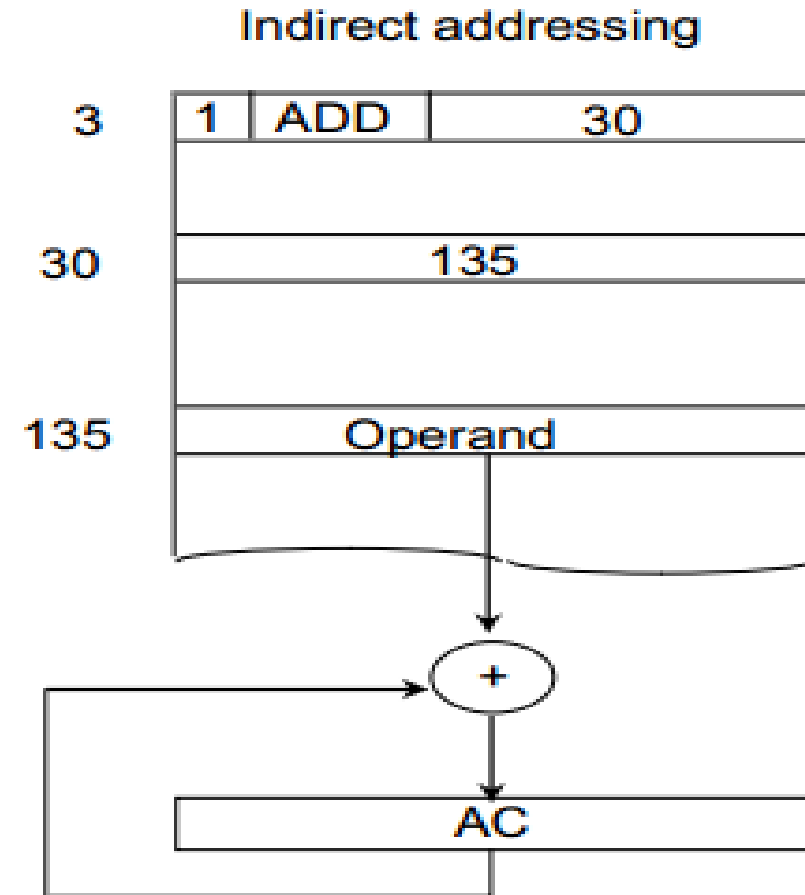
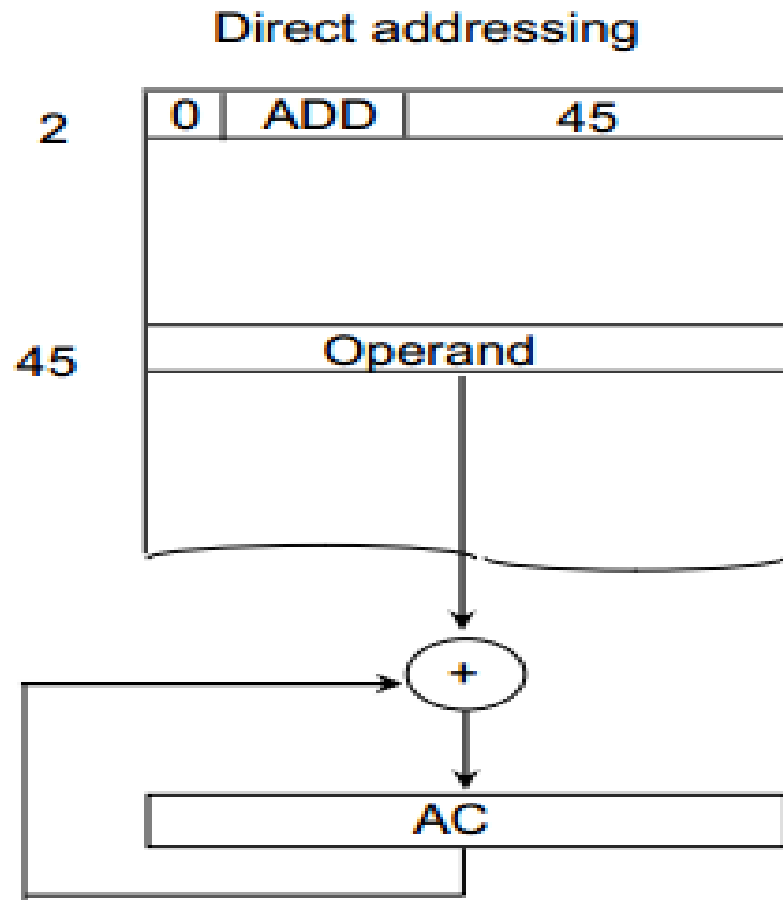
Addressing Modes of Basic computer (Direct and Indirect addressing mode)



Addressing Modes of Basic computer (Direct and Indirect addressing mode)

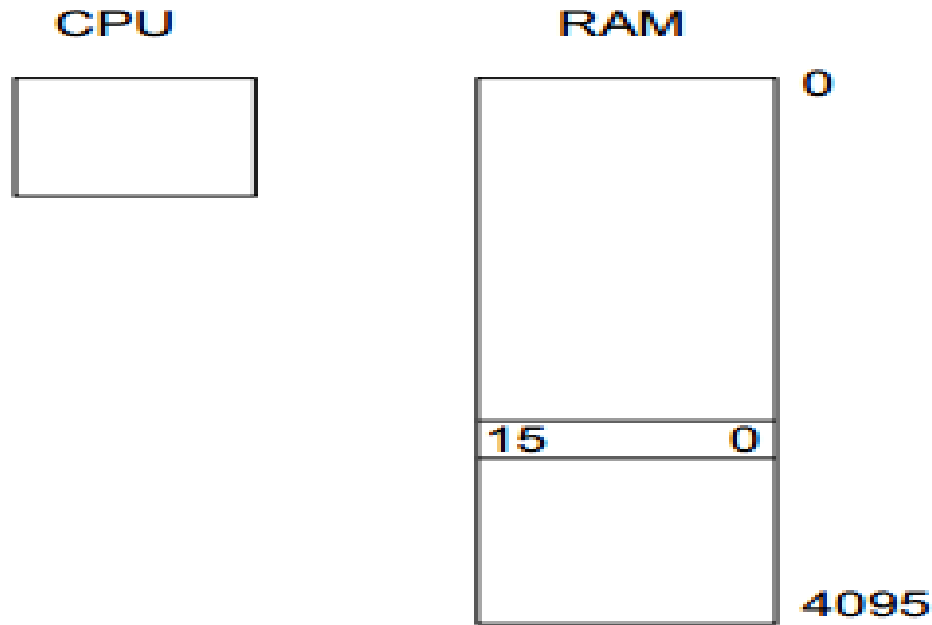
- ▶ The indirect address instruction needs two references to memory to fetch an operand.
 - The first reference is needed to read the address of the operand.
 - Second reference is for the operand itself.
- 

Addressing Modes of basic computer (Direct and Indirect addressing mode)




Computer Registers

- ▶ Basic computer have memory and Processors.
- ▶ Memory have capacity=4096 words= 2^{12}
- ▶ 1 word=16 bits




Computers Registers

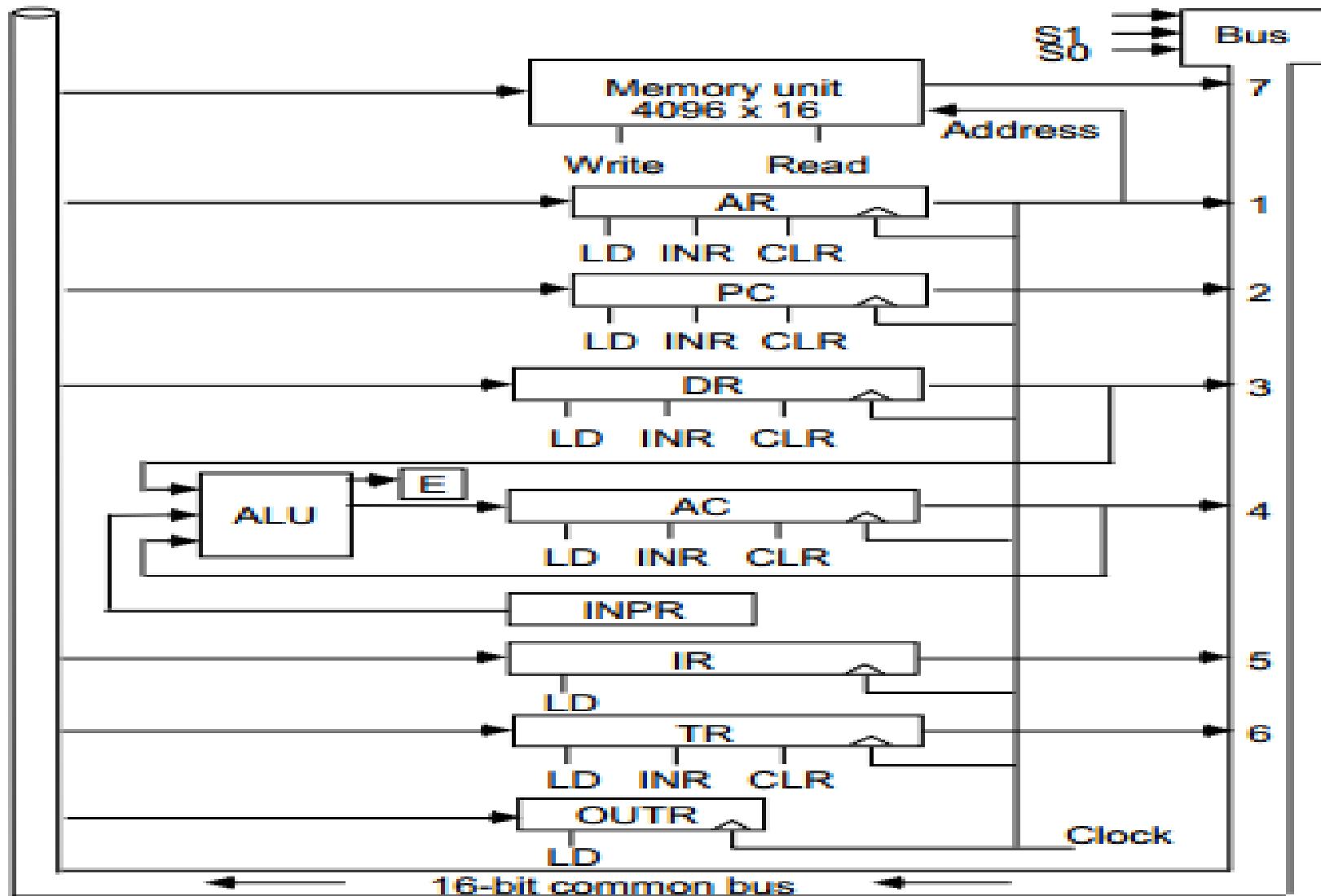
- ▶ DR (Data registers)–To hold operand read from memory. **16 bits**
 - ▶ AC (Accumulator)– General purpose register, used for storing intermediate arithmetic and logic results. **16 bits**
 - ▶ IR(Instruction Register)– Instruction fetched from memory are placed in IR. **16 bits**
 - ▶ TR(Temporary Registers)– used for holding temporary data. **16 bits**
- 

- ▶ AR(Address registers)– holds address for memory. **12 bits**
- ▶ PC(Program counter)– Address of next instruction. **12 bits**
- ▶ INPR(Input Register)–receives 8 bit input from input device. **8 bits**
- ▶ OUTR(Output Registers)–hold 8 bit character for output. **8 bits**


Why common bus system is used?

- ▶ Basic computer has eight registers, a memory unit and a control unit.
 - ▶ Path must be provided to transfer information from one register to another and between memory and register.
 - ▶ The number of wires will be excessive if connections are between the outputs of each register and input of other registers.
 - ▶ So efficient way is to use common bus system.
- 

Common Bus system



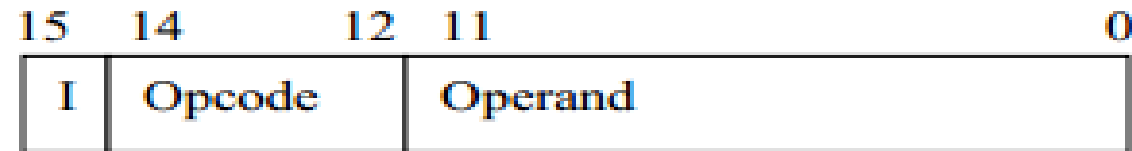
Computer Instructions

- ▶ **Memory reference instructions**—instruction that has operand addresses referring to a location in memory
 - ▶ **Register Reference Instructions**—specifies a operation on AC or a test of the AC
 - ▶ **Input Output instructions**—used to input data / output data
- 

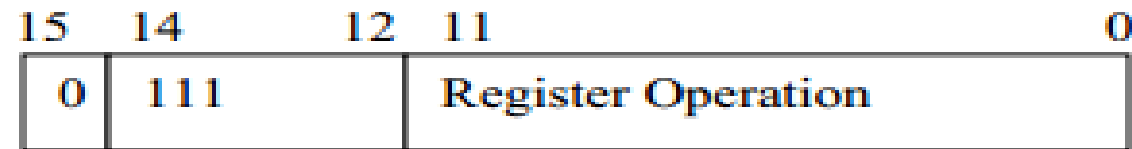
Computer Instruction Format with its types

Instruction Formats of Basic Computer

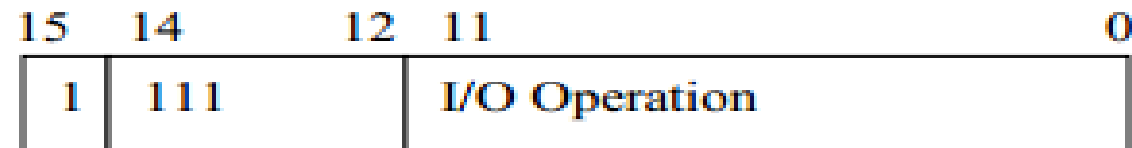
Memory-Reference Instructions (OP-code = 000 ~ 110)



Register-Reference Instructions (OP-code = 111, I = 0)



Input-Output Instructions (OP-code = 111, I = 1)




Basic Computer Instructions

TABLE 5-2 Basic Computer Instructions

Symbol	Hexadecimal code		Description
	<i>I</i> = 0	<i>I</i> = 1	
AND	0xxx	8xxx	AND memory word to <i>AC</i>
ADD	1xxx	9xxx	Add memory word to <i>AC</i>
LDA	2xxx	Axxx	Load memory word to <i>AC</i>
STA	3xxx	Bxxx	Store content of <i>AC</i> in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear <i>AC</i>
CLE	7400		Clear <i>E</i>
CMA	7200		Complement <i>AC</i>
CME	7100		Complement <i>E</i>
CIR	7080		Circulate right <i>AC</i> and <i>E</i>
CIL	7040		Circulate left <i>AC</i> and <i>E</i>
INC	7020		Increment <i>AC</i>
SPA	7010		Skip next instruction if <i>AC</i> positive
SNA	7008		Skip next instruction if <i>AC</i> negative
SZA	7004		Skip next instruction if <i>AC</i> zero
SZE	7002		Skip next instruction if <i>E</i> is 0
HLT	7001		Halt computer
INP	F800		Input character to <i>AC</i>
OUT	F400		Output character from <i>AC</i>
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

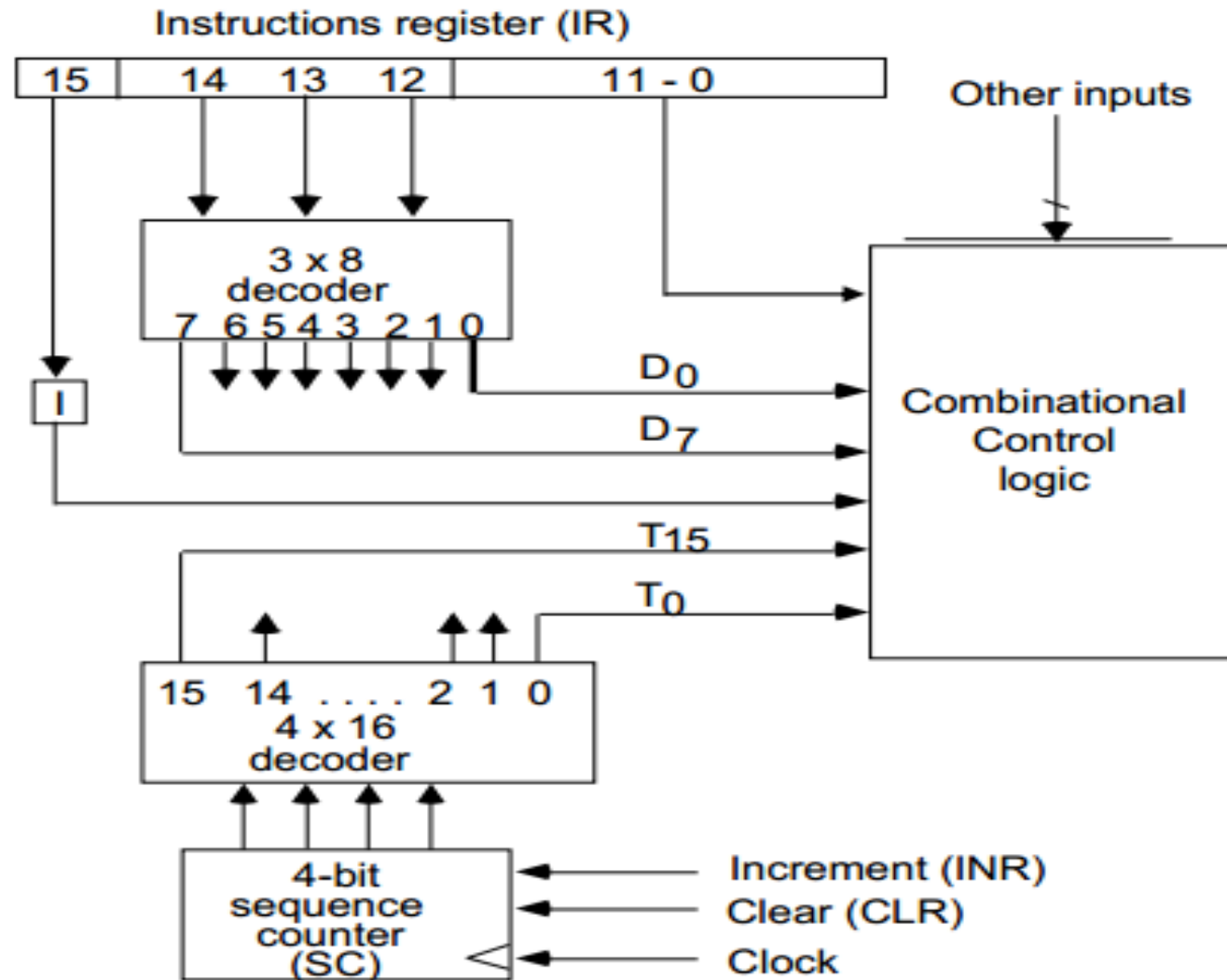
Instruction Set Completeness

- ▶ Instruction set is complete if it contains sufficient set of instructions in each of following categories–
 - Arithmetic, logical and shift instruction
 - Instruction for moving information to and from memory and processor registers.
 - Program control instructions together that check status conditions(E.g Branch)
 - Input and Output Instructions
- 


Control Unit

- ▶ Control Unit is implemented In 2 ways:-
- ▶ ***Hardwired Control***
 - CU is made up of sequential and combinational circuits to generate the control signals.
 - If logic is changed we need to change the whole circuitry
 - Expensive
 - Fast
- ▶ ***Micro programmed Control***
 - A control memory on the processor contains micro-programs that activate the necessary control signals.
 - If logic is changed we only need to change the micro-program.
 - Cheap
 - Slow

Control Unit of Basic Computer



Instructions cycle

- ▶ Program is executed in computer by going through a cycle for each instruction. Each instruction cycle is subdivided into different phases–
 1. Fetch an instruction from memory.
 2. Decode the instruction.
 3. Read the effective address from memory if the instruction has an indirect address.
 4. Execute the instruction.
- 

Fetch and decode

- ▶ Initially PC is loaded with address of first instruction .
- ▶ Sequence counter SC is cleared to 0, providing T_0 . After each clock pulse SC is Incremented and timing sequence go through T_0 , T_1 , T_2 and so on.

Fetch and Decode

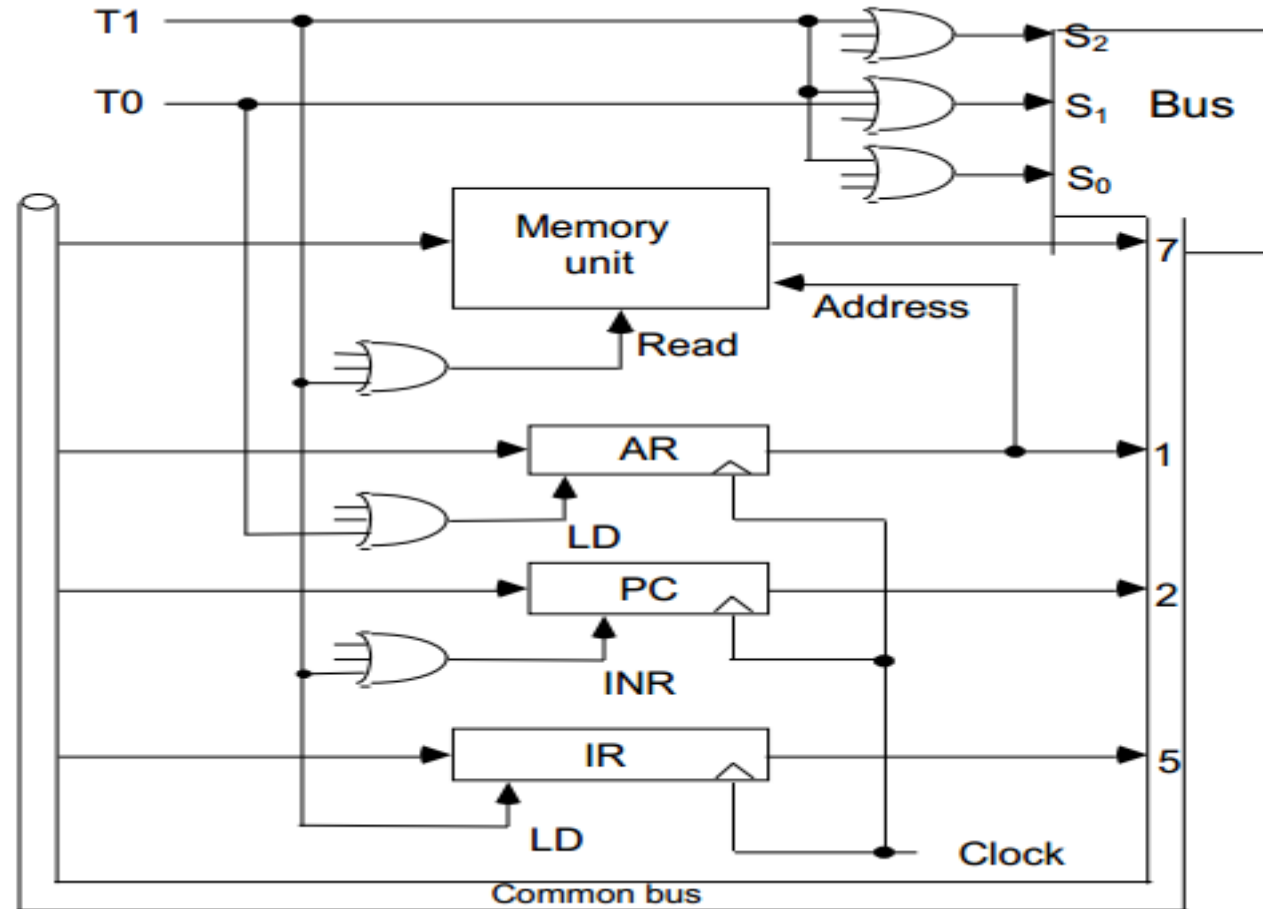
- ▶ The micro operations for the fetch and decode phases can be specified by the following register transfer statements:

T0: $AR \leftarrow PC$ ($S_0S_1S_2=010, T_0=1$)

T1: $IR \leftarrow M[AR], PC \leftarrow PC + 1$ ($S_0S_1S_2=111, T_1=1$)

T2: $D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$

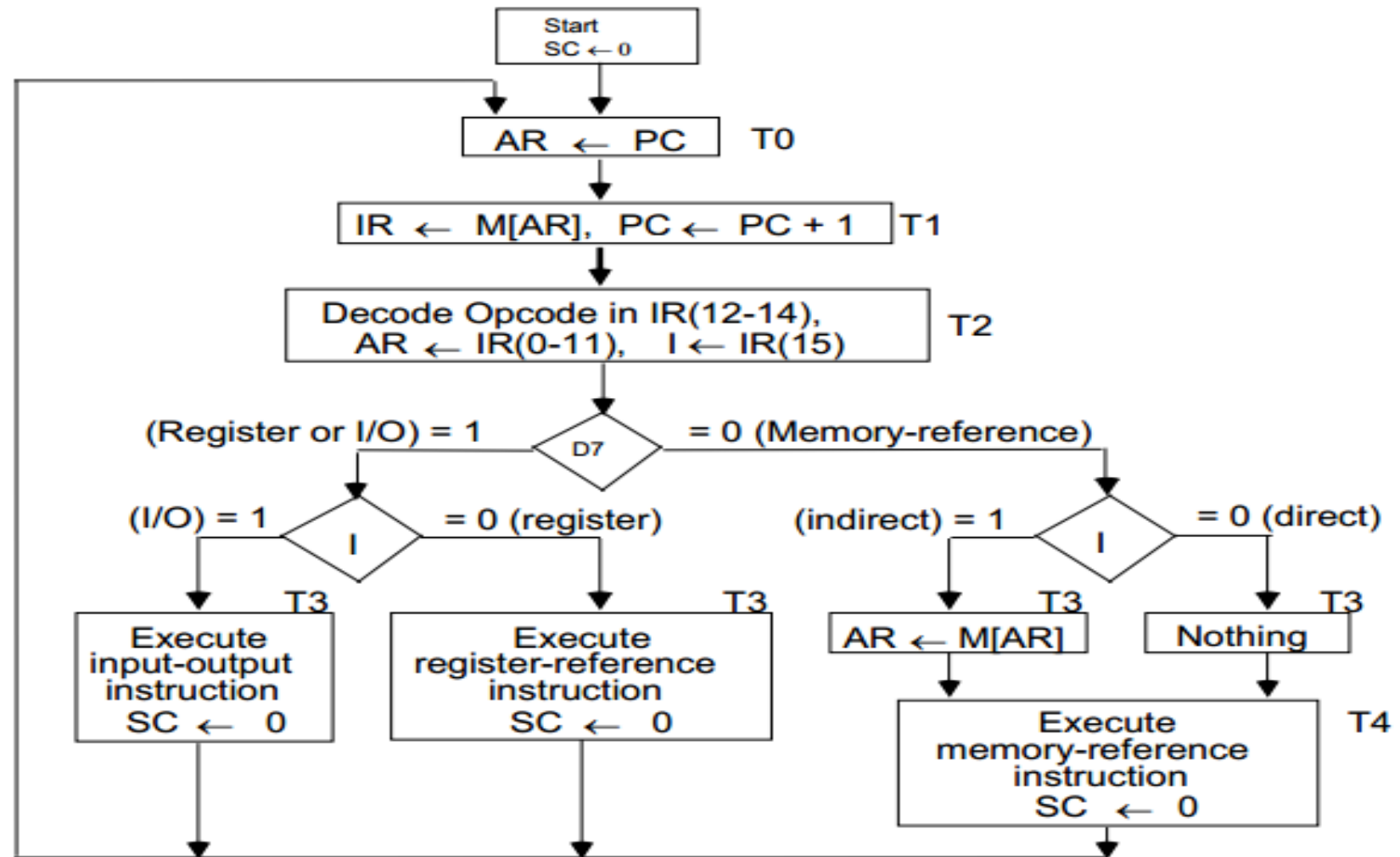
Fetch Phase



Determine the type of instruction

- ▶ After decoding the timing signal is T_3
- ▶ Control unit determine the type of instruction just read from memory.
- ▶ There are 3 type of instruction.
 - $D_7 = 1$, opcode=111
 - Instruction can be register reference or input/output type.
 - $I=0$ register reference
 - $i=1$ I/O reference
 - $D_7 = 0$, opcode=000 to 110
 - Instruction is memory referenced.
 - $I=0$ direct addressing
 - $I=1$ indirect Addressing

Flowchart of instruction cycle



Register –Reference Instructions

- ▶ Register Reference Instructions are identified when
 - $D7 = 1, I = 0$
 - Register Ref. Instr. is specified in b0 ~ b11 of IR
 - Execution starts with timing signal T3

Register –Reference Instructions

CLA	rB11:	$AC \leftarrow 0, SC \leftarrow 0$
CLE	rB10:	$E \leftarrow 0, SC \leftarrow 0$
CMA	rB9:	$AC \leftarrow AC', SC \leftarrow 0$
CME	rB8:	$E \leftarrow E', SC \leftarrow 0$
CIR	rB7:	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0), SC \leftarrow 0$
CIL	rB6:	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	rB5:	$AC \leftarrow AC + 1, SC \leftarrow 0$
SPA	rB4:	if ($AC(15) = 0$) then ($PC \leftarrow PC+1$), $SC \leftarrow 0$
SNA	rB3:	if ($AC(15) = 1$) then ($PC \leftarrow PC+1$), $SC \leftarrow 0$
SZA	rB2:	if ($AC = 0$) then ($PC \leftarrow PC+1$), $SC \leftarrow 0$
SZE	rB1:	if ($E = 0$) then ($PC \leftarrow PC+1$), $SC \leftarrow 0$
HLT	rB0:	$S \leftarrow 0, SC \leftarrow 0$ (S is a start-stop flip-flop)

Memory Reference Instructions

- The effective address of the instruction is in AR and was placed there during timing signal T2 when $I = 0$, or during timing signal T3 when $I = 1$
- Memory cycle is assumed to be short enough to complete in a CPU cycle
 - The execution of memory reference instruction starts with T4

Symbol	Operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1, \text{ if } M[AR] + 1 = 0 \text{ then } PC \leftarrow PC + 1$

MEMORY REFERENCE INSTRUCTIONS

Symbol	Operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1, \text{ if } M[AR] + 1 = 0 \text{ then } PC \leftarrow PC + 1$

- The effective address of the instruction is in AR and was placed there during timing signal T_2 when $I = 0$, or during timing signal T_3 when $I = 1$
- Memory cycle is assumed to be short enough to complete in a CPU cycle
- The execution of MR instruction starts with T_4

AND to AC

D_0T_4 : $DR \leftarrow M[AR]$ Read operand
 D_0T_5 : $AC \leftarrow AC \wedge DR, SC \leftarrow 0$ AND with AC

ADD to AC

D_1T_4 : $DR \leftarrow M[AR]$ Read operand
 D_1T_5 : $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$ Add to AC and store carry in E

MEMORY REFERENCE INSTRUCTIONS

LDA: Load to AC

$D_2T_4: DR \leftarrow M[AR]$

$D_2T_5: AC \leftarrow DR, SC \leftarrow 0$

STA: Store AC

$D_3T_4: M[AR] \leftarrow AC, SC \leftarrow 0$

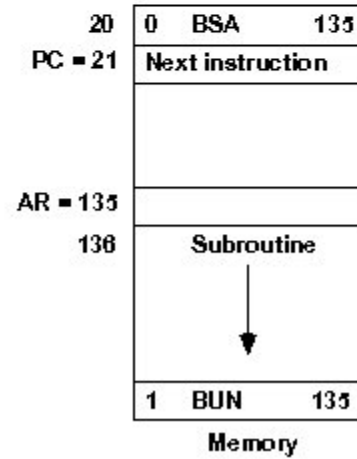
BUN: Branch Unconditionally

$D_4T_4: PC \leftarrow AR, SC \leftarrow 0$

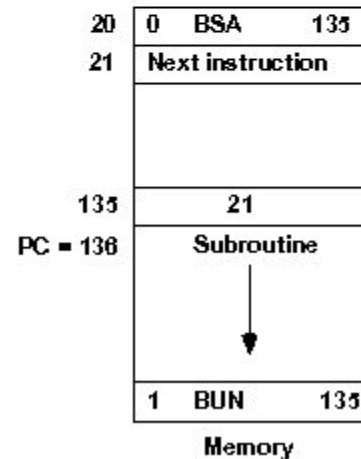
BSA: Branch and Save Return Address

$M[AR] \leftarrow PC, PC \leftarrow AR + 1$

Memory, PC, AR at time T4



Memory, PC after execution



MEMORY REFERENCE INSTRUCTIONS

BSA:

$D_3T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$

$D_3T_5: PC \leftarrow AR, SC \leftarrow 0$

ISZ: Increment and Skip-if-Zero

$D_6T_4: DR \leftarrow M[AR]$

$D_6T_5: DR \leftarrow DR + 1$

$D_6T_6: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$