Chapter 4:

Basic Computer Organization and Design

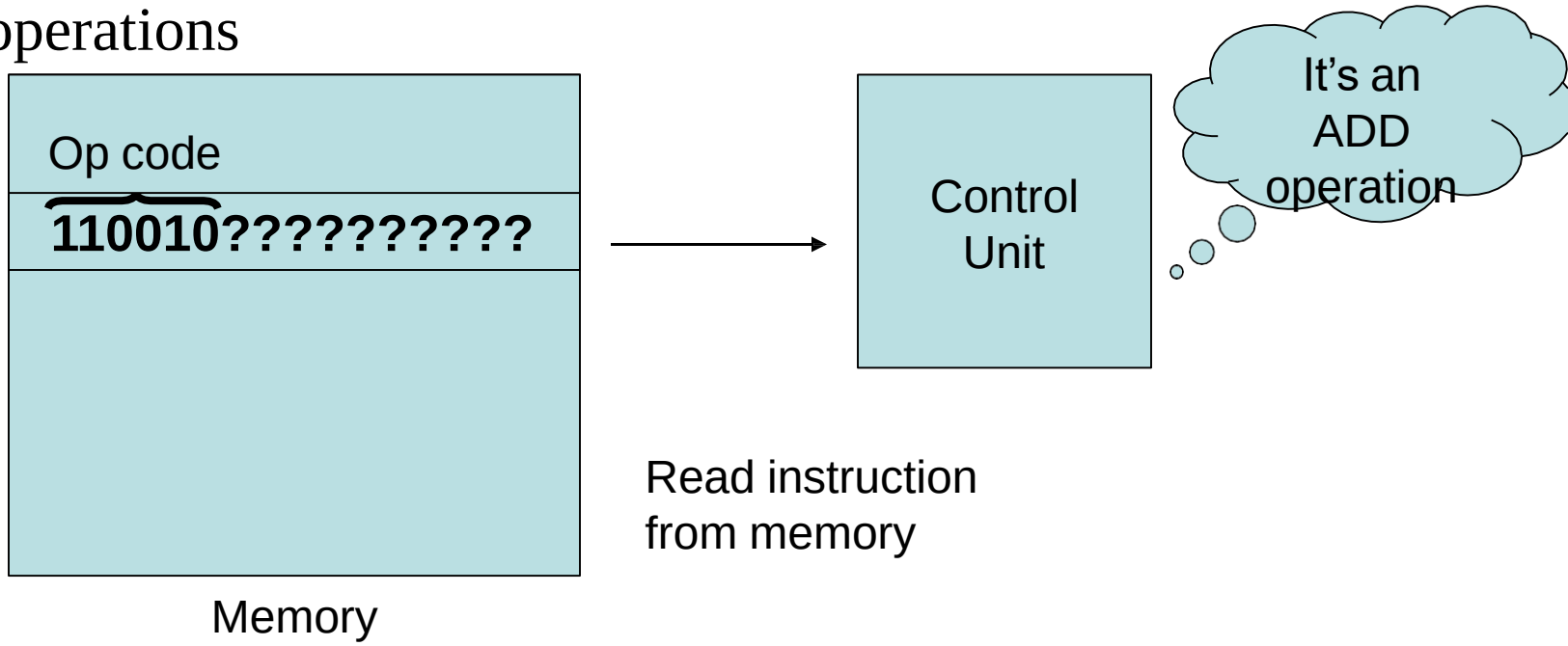
Topics to be covered:

- Instruction Codes
- Addressing modes
- Computer Registers
- Instruction Set
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input –output devices
- Design of Basic Computer

Instruction Codes

- An **instruction** is a binary **code** that specifies a sequence of micro-operations.
- Instructions are encoded as binary *instruction codes*.
- **Instruction codes and data are stored in memory**
- An **instruction code** is a group of bits that tells the computer to perform a specific operation part.
- The computer reads each instruction from memory and places it in a control register
  Instruction codes and data are stored in memory
- The control unit interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of micro-operations
- An instruction is divided into 2 parts (basic part is the **operation part**)
  - The **operation code (op code)** of an instruction is a group of bits that define operations such as add, subtract, multiply, shift and compliment.

- The number of bits required for the operation code depends on the total number of operations available in the computer
- n bit operation code -> 2^n (or little less) distinct operations
  - It must consist of at least **n bits** for a given **2^n** distinct operations

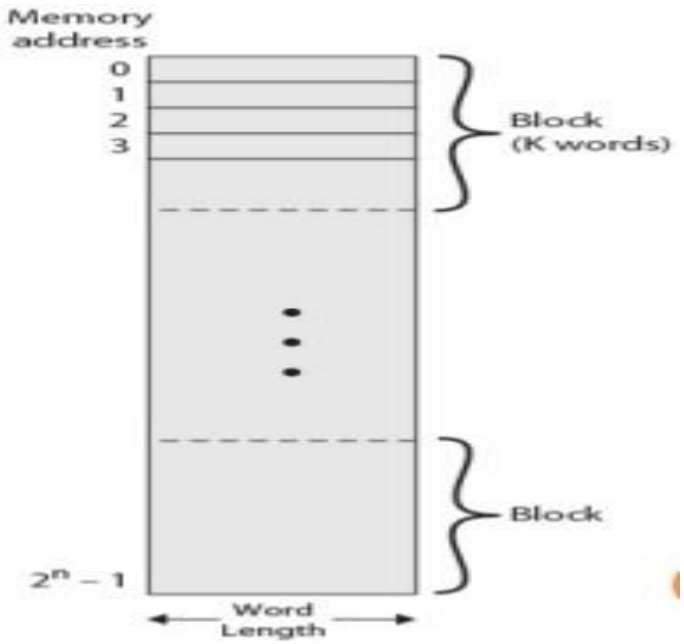


- Suppose we are having **64 operation** then the length of OpCode will be 6. (since 2^6=64)
- The **control unit** decode the OpCode and determines the **operations**

Instruction Codes cont.

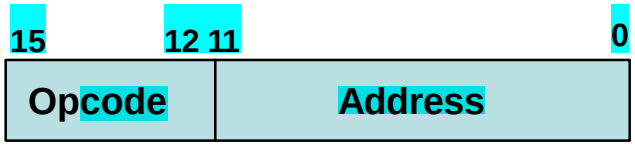
- An operation must be performed on some data stored in processor registers or in memory
- An instruction code must therefore specify not only the operation, but also the location of the operands (in registers or in the memory), and where the result will be stored (registers/memory)

•Memory words can be specified in instruction codes by their address



Instruction Codes cont.

- Every Computer has its own particular instruction code format.
  - Instruction code formats are conceived by computer designers who specify the architecture of the computer
- The simplest way to organize a computer is to have **Processor Register** and **instruction code with two parts**.
- The first part specifies the operation to be performed and the second specifies an address



Instruction Format

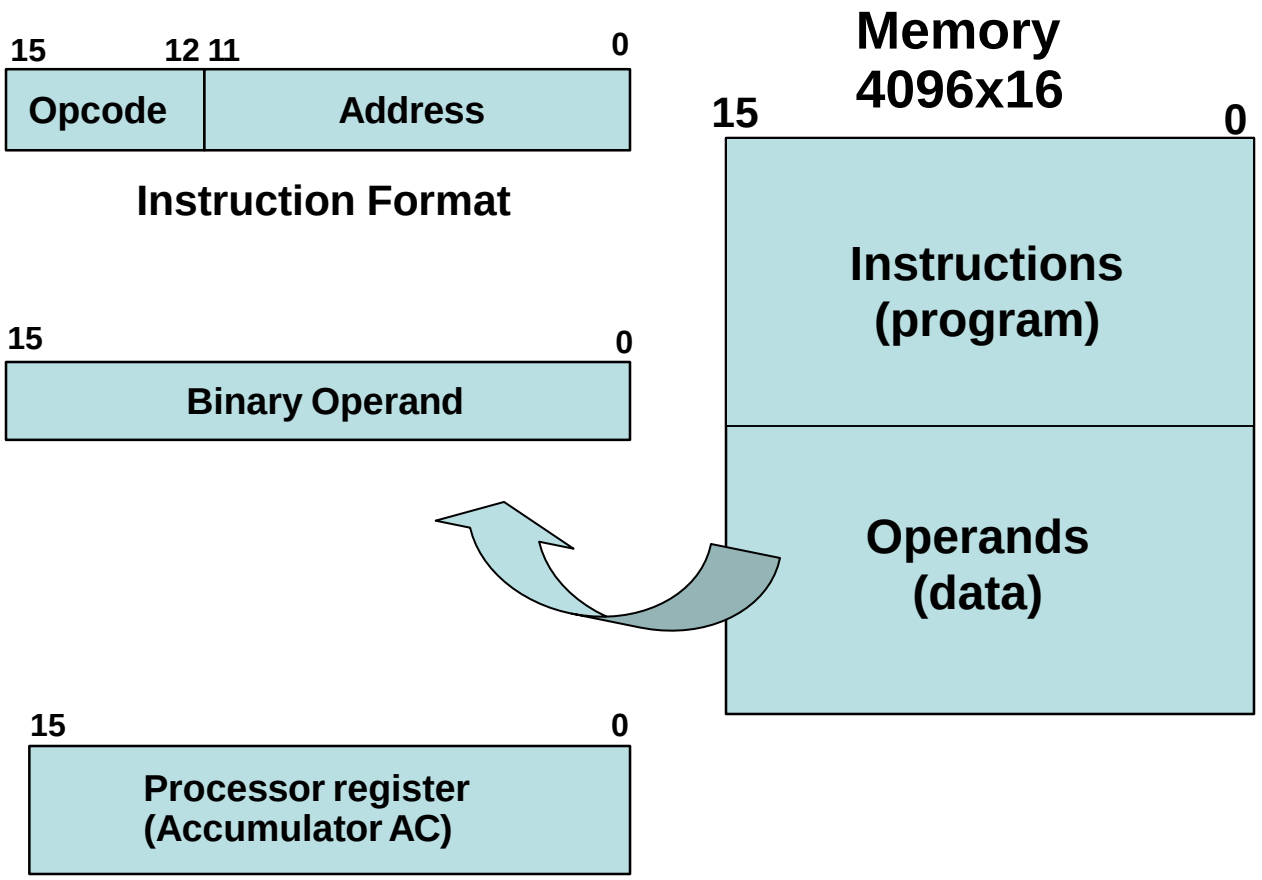
What to do: Opcode

Where: memory address

- The first part specifies the operation to be performed and second specifies an address.
- The memory address tells where the operand in memory will be found.

Instruction Codes

- 2^{12}=4096 word of RAM , 12 bits are required for addressing
- 16 bit RAM (word length)



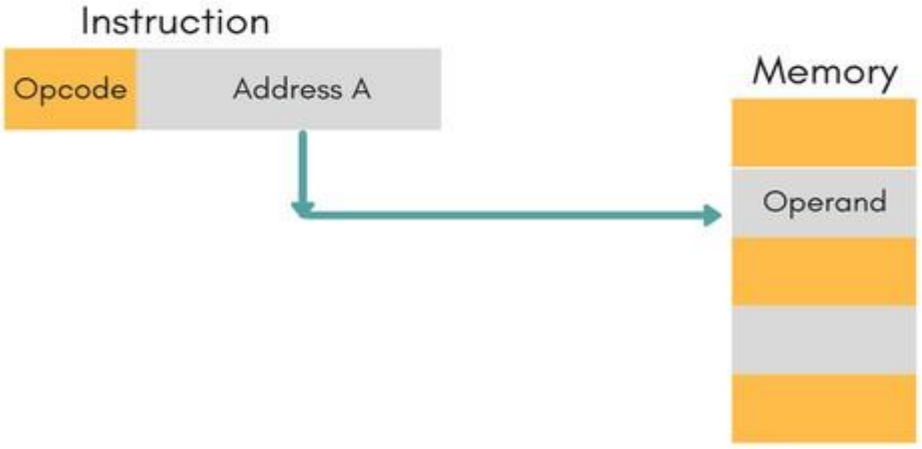
Addressing modes

- The operation field of an instruction specifies the operation to be performed.
- This operation will be executed on some data which is stored in computer registers or the main memory.
- **The way any operand is selected** during the program execution is dependent on the addressing mode of the instruction.
  - ***Addressing mode*** refers to the way in which the operand of an instruction is specified.

- There are different **Addressing Modes** used for address portion of the instruction code. Some of these addressing modes are
  - Immediate addressing mode
  - Direct addressing mode
  - Indirect addressing mode

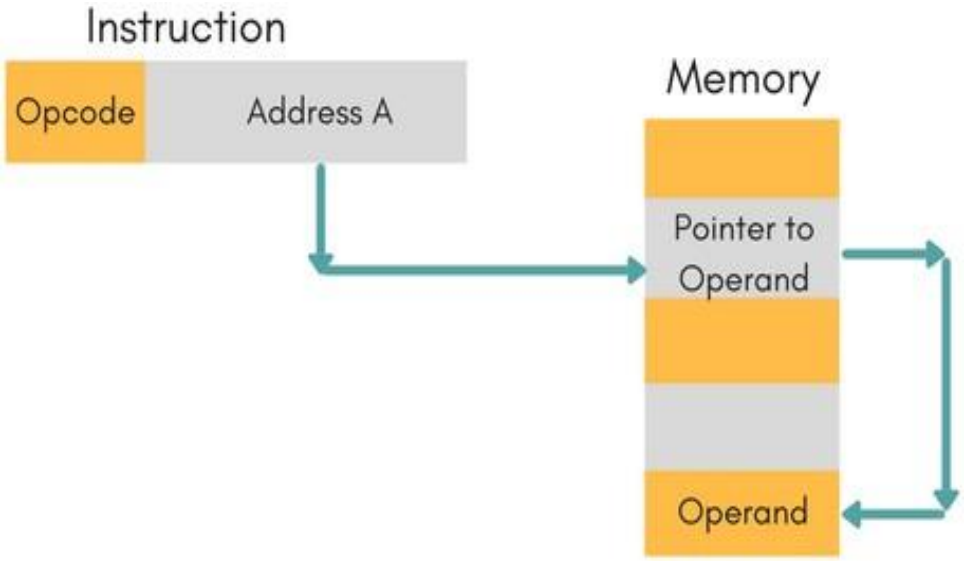
Addressing modes

- **Immediate**: the operand is given in the address portion (constant)
  - For example: **ADD 7**
    - which says Add 7 to contents of accumulator. 7 is the operand here.
- **Direct**: the address points to the operand stored in the memory
  - effective address of operand is present in instruction itself.
  - Single memory reference to access data.
  - No additional calculations to find the effective address of the operand.



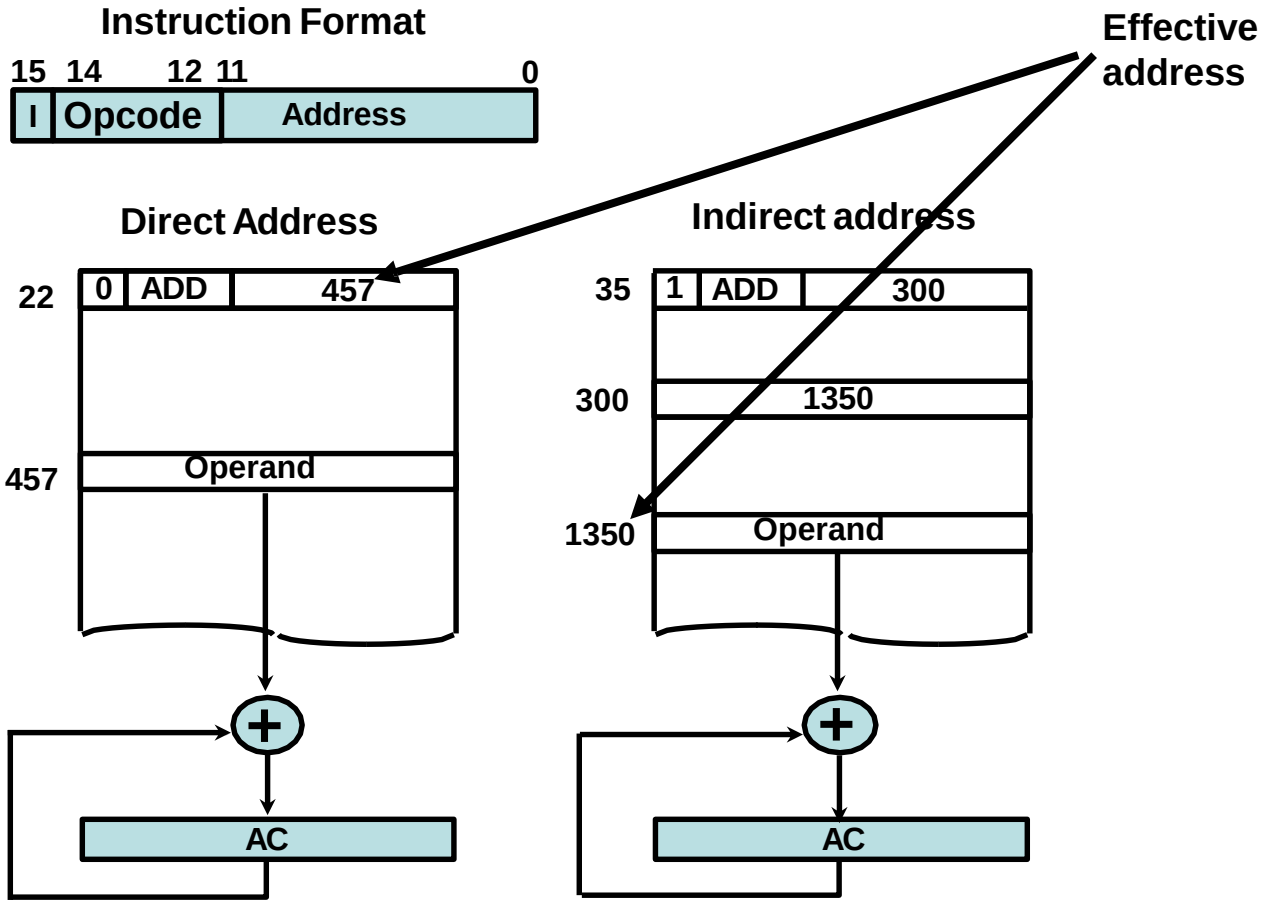
Addressing modes

- **Indirect**: the address points to the pointer (another address) stored in the memory that references the operand in memory
  - This slows down the execution, as this includes multiple memory lookups to find the operand.
- One bit of the instruction code can be used to distinguish between direct & indirect addresses



Addressing modes

- The addressing mod (I) is 0 for direct addressing and 1 for indirect mode in the following instruction format.



Effective address: the address of the operand

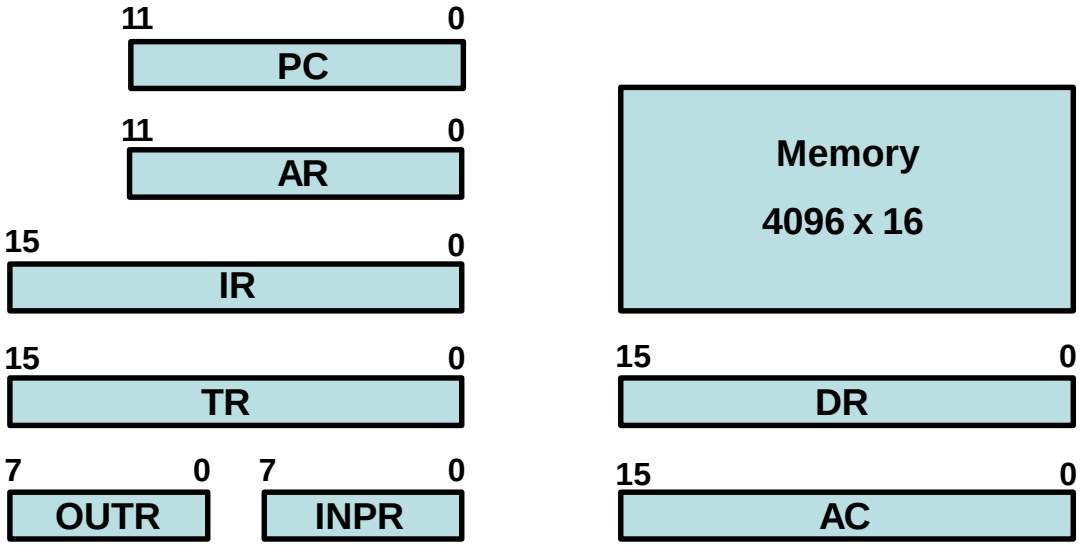
In order to cover the basic concepts behind designing a computer, a model (an imaginary system) will be presented to you throughout this chapter. This model will be called the "Basic Computer"

- The basic computer has 8 registers, a memory unit and a control unit.
- A **Common Bus System is used** for transferring data in a system.
- The basic computer has three instruction code formats.

Computer Registers

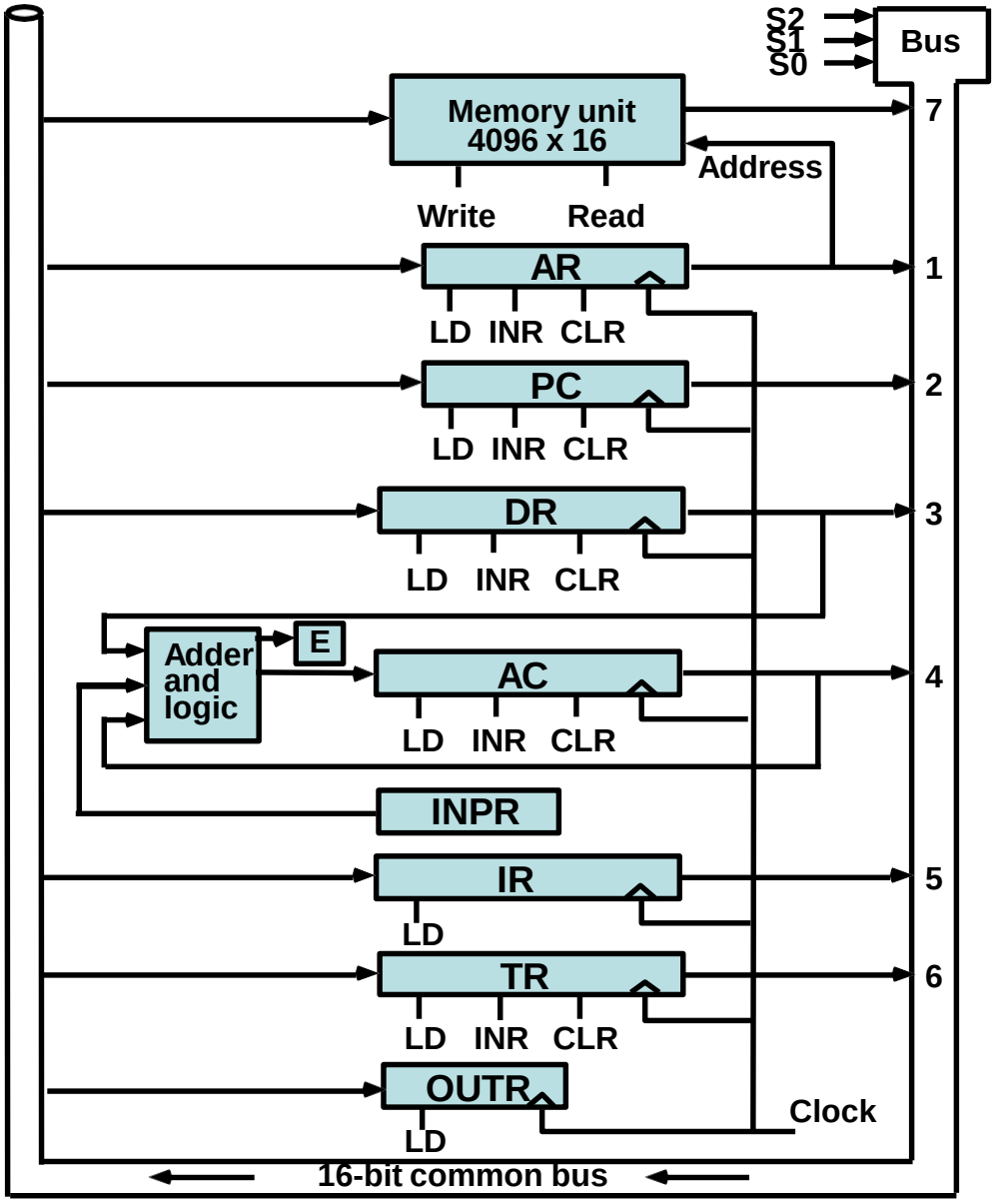
- Computer instructions are normally stored in consecutive memory locations and executed sequentially one at a time
- The control reads an instruction from a specific address in memory and executes it, and so on
- This type of sequencing needs a counter to calculate the address of the next instruction after execution of the current instruction is completed
- It is also necessary to provide a register in the control unit for storing the instruction code after it is read from memory
- The computer needs processor registers for manipulating data and a register for holding a memory address

Registers in the Basic Computer



List of BC Registers

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



Computer Registers Common Bus System

Bus is a set of wires in a computer along which information can be sent to and from other parts of the computer

Register → Memory: Write operation

$$M[AR] \leftarrow DR$$

Memory → Register: Read operation

$$DR \leftarrow M[AR]$$

$$IR \leftarrow M[AR]$$

(note that AC cannot directly read from memory!!)

Computer Registers
Common Bus System cont.

- **$S_2S_1S_0$**: Selects the register/memory that would use the bus
- **LD** (load): When enabled, the **particular register** receives the data from the bus during the next clock pulse transition
- **E** (extended AC bit): flip-flop holds the carry
- **DR, AC, IR, and TR**: have 16 bits each
- **AR and PC**: have 12 bits each since they hold a memory address
- When the contents of AR or PC are applied to the **16-bit** common bus, the **four** most significant bits are set to **zeros**
- When **AR** or **PC receives** information from the bus, only the **12 least** significant bits are transferred into the register.
- **INPR and OUTR**: communicate with the eight least significant bits in the bus

Computer Registers
Common Bus System cont.

- **INPR**: Receives a character from the input device (keyboard,…etc) which is then transferred to AC
- **OUTR**: Receives a character from AC and delivers it to an output device (say a Monitor)
- **Five** registers have three control inputs: **LD** (load), **INR** (increment), and **CLR** (clear)
- The input data and output data of the memory are connected to the **common bus**
- But the memory address is connected to **AR**
- Therefore, **AR** must always be used to specify a memory address

Computer Registers cont.

- The transition at the end of the cycle transfers the content of the bus into the destination register, and the output of the adder and logic circuit into the AC
- For example, the two microoperations

  DR←AC and AC←DR (Exchange)

  can be executed at the same time
- This is done by:

Computer Registers cont.

- 1- place the contents of AC on the bus ($S_2S_1S_0$=100)
- 2- enabling the LD (load) input of DR
- 3- Transferring the contents of the DR through the adder and logic circuit into AC
- 4- enabling the LD (load) input of AC
- All during the same clock cycle
- The two transfers occur upon the arrival of the clock pulse transition at the end of the clock cycle

Instruction Set

- Computer instructions are a set of machine language instructions that a particular processor understands and executes.
- A computer performs tasks on the basis of the instruction provided.
- What is **instruction set**?
  - An **instruction set** is a list of commands ready to be executed directly by CPU.
  - The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
  - It is the complete set of all the instructions in machine code that can be recognized and executed by a central processing unit.
- NB: **Each computer has its unique instruction set**
  - Eg., Intel Microprocessor 8085 has 246 instructions

Instruction Set

Instruction Set Completeness

- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:
  - Arithmetic, logical, and shift instructions
  - Instructions for moving information to and from memory and processor registers
  - Program control instructions together with instructions that check status conditions
  - Input & output instructions

Instruction Classification

- Computer provides an extensive set of instructions to give the user the flexibility to carryout various computational task.
- Most computer instruction can be classified into three categories.
    1. Data transfer instruction
    2. Data manipulation instruction
    3. Program control instruction

1. Data Transfer Instruction

- used to move data from one place in the computer to another **without changing the data content.**
- The most common transfers are: between memory and processes registers, between processes register & input or output, and between processes register themselves

Instruction Classification

Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

2. Data Manipulation Instruction

- It performs operations on data and provides the computational capabilities for the computer.
- The data manipulation instructions in a typical computer are usually divided into three basic types.
  - Arithmetic Instruction
  - Logical bit manipulation Instruction
  - Shift Instruction.

Instruction Classification

Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate(2's Complement)	NEG

Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thru carry	RORC
Rotate left thru carry	ROLC

Instruction Classification

3. Program Control Instructions

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test (by AND)	TST

Examples of Program Control Instructions

- Program control instructions specify conditions for altering the content of the program counter, while data transfer and manipulation instructions specify conditions for data processing operations.

  BR 100 : the program counter go to address 100.

Computer Instructions

- The basic computer has three Instruction code formats
- Each format has 16 bits

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

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

Computer Instructions

Instruction Set Completeness

- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:
  - Arithmetic, logical, and shift instructions
  - Instructions for moving information to and from memory and processor registers
  - Program control instructions together with instructions that check status conditions
  - Input & output instructions

Timing & Control

timing and control signals to all operations in the computer.

-It **controls the flow of data between the processor and memory and peripherals**.

- The timing for all registers in the basic computer is controlled by a master clock generator
- The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit
- The clock pulses do not change the state of a register unless the register is enabled by a control signal (i.e., Load)

Timing & Control cont.

- The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and microoperations for the accumulator
- There are two major types of control organization:
  - Hardwired control
  - Microprogrammed control

1. **Hardwired control unit :** the control unit will be design the help of hardware component.

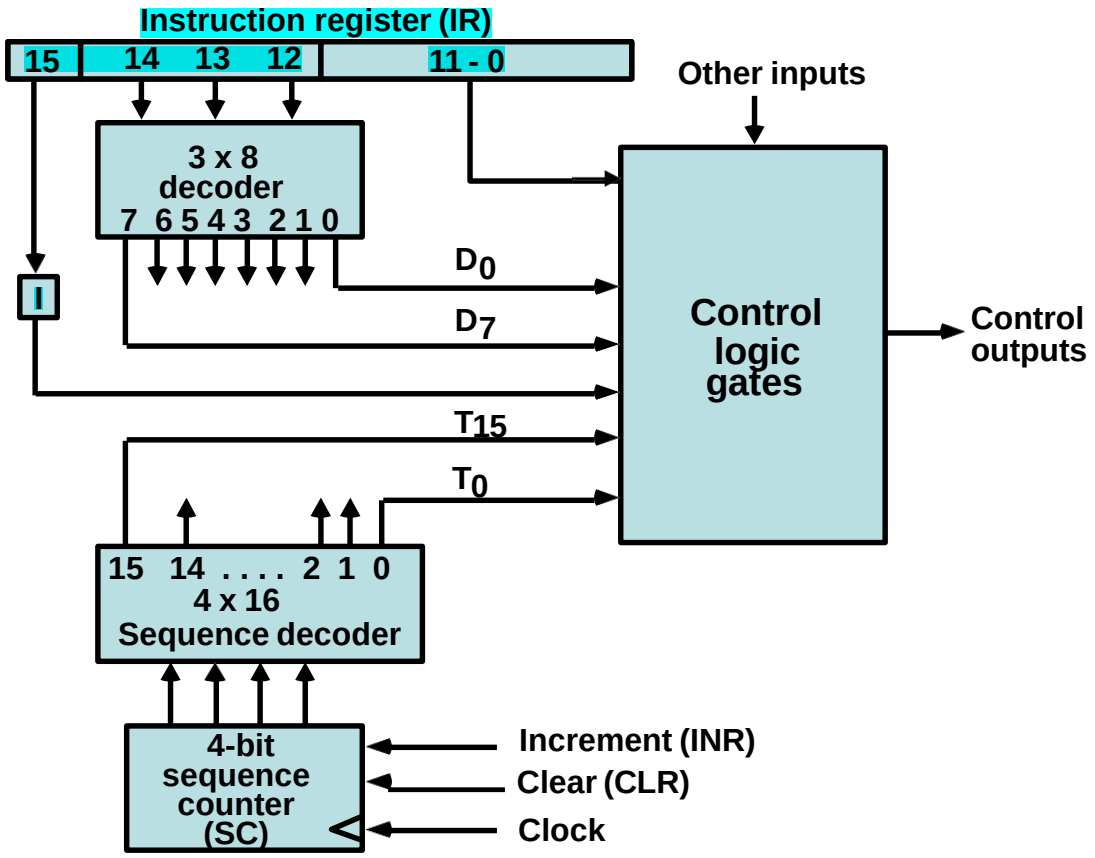
- In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits.
- It is extremely faster but it is difficult to modification.

Timing & Control cont.

2. Microprogrammed control unit: the control unit design with the help of microprogram , control memory.

- In the microprogrammed organization, the control information is stored in a control memory (if the design is modified, the microprogram in control memory has to be updated)
- Microprogrammed control unit
  - Is less complex and simpler to implement than hardwired control unit
  - Hence, it is cheaper and less error prone than hardwired control unit
  - But it is slower than hardwired control unit

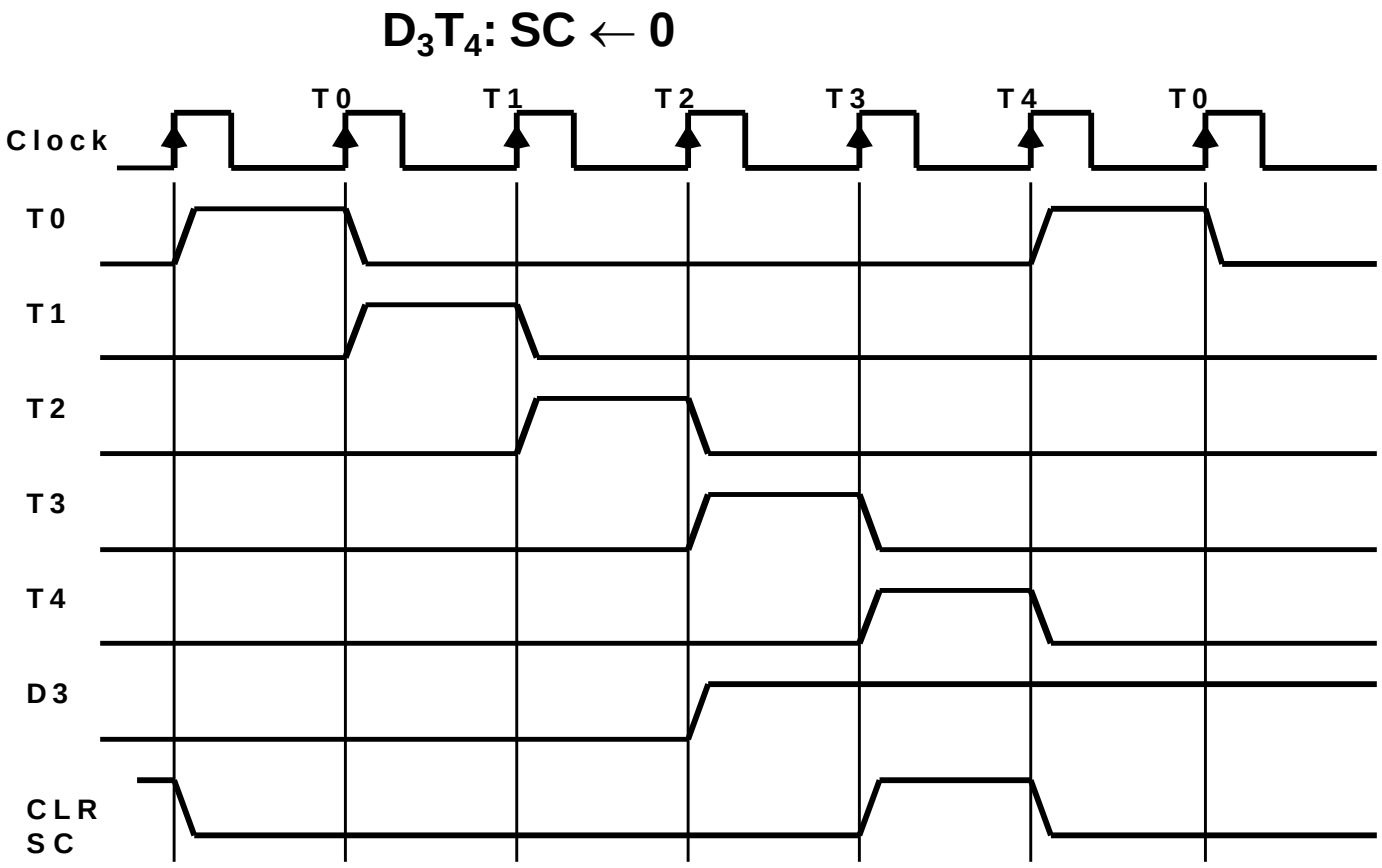
The Control Unit for the basic computer



Hardwired Control Organization

- **Timing signals Generated by 4-bit sequence counter and 4x16 decoder**
- **The SC can be incremented or cleared.**
- **Example: $T_0, T_1, T_2, T_3, T_4, T_0, T_1, \ldots$**

  Assume: At time T_4, SC is cleared to 0 if decoder output D3 is active.



Timing & Control cont.

- T_0: AR←PC
  - Transfers the content of PC into AR if timing signal T_0 is active
  - T_0 is active during an entire clock cycle interval
  - During this time, the content of PC is placed onto the bus (with $S_2S_1S_0$=010) and the LD (load) input of AR is enabled
  - The actual transfer does not occur until the end of the clock cycle when the clock goes through a positive transition
  - This same positive clock transition increments the sequence counter SC from 0000 to 0001
  - The next clock cycle has T_1 active and T_0 inactive

Instruction Cycle

- A program is a sequence of instructions stored in memory
- The program is executed in the computer by going through a cycle for each instruction (in most cases)
- Each instruction in turn is subdivided into a sequence of sub-cycles or phases

Instruction Cycle cont.

- Instruction Cycle Phases:
  - 1- Fetch an instruction from memory
  - 2- Decode the instruction:which operation in the corresponding instruction will be performed.
  - 3- Read the effective address from memory
  - 4- Execute the instruction
- This cycle repeats indefinitely unless a HALT instruction is encountered

Instruction Cycle
Fetch and Decode

- Initially, the Program Counter (PC) is loaded with the address of the first instruction in the program
- The sequence counter SC is cleared to 0, providing a decoded timing signal T_0
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on

Instruction Cycle
Fetch and Decode cont.

Fetch

– T_0: AR←PC (this is essential!!)

The address of the instruction is moved to AR.

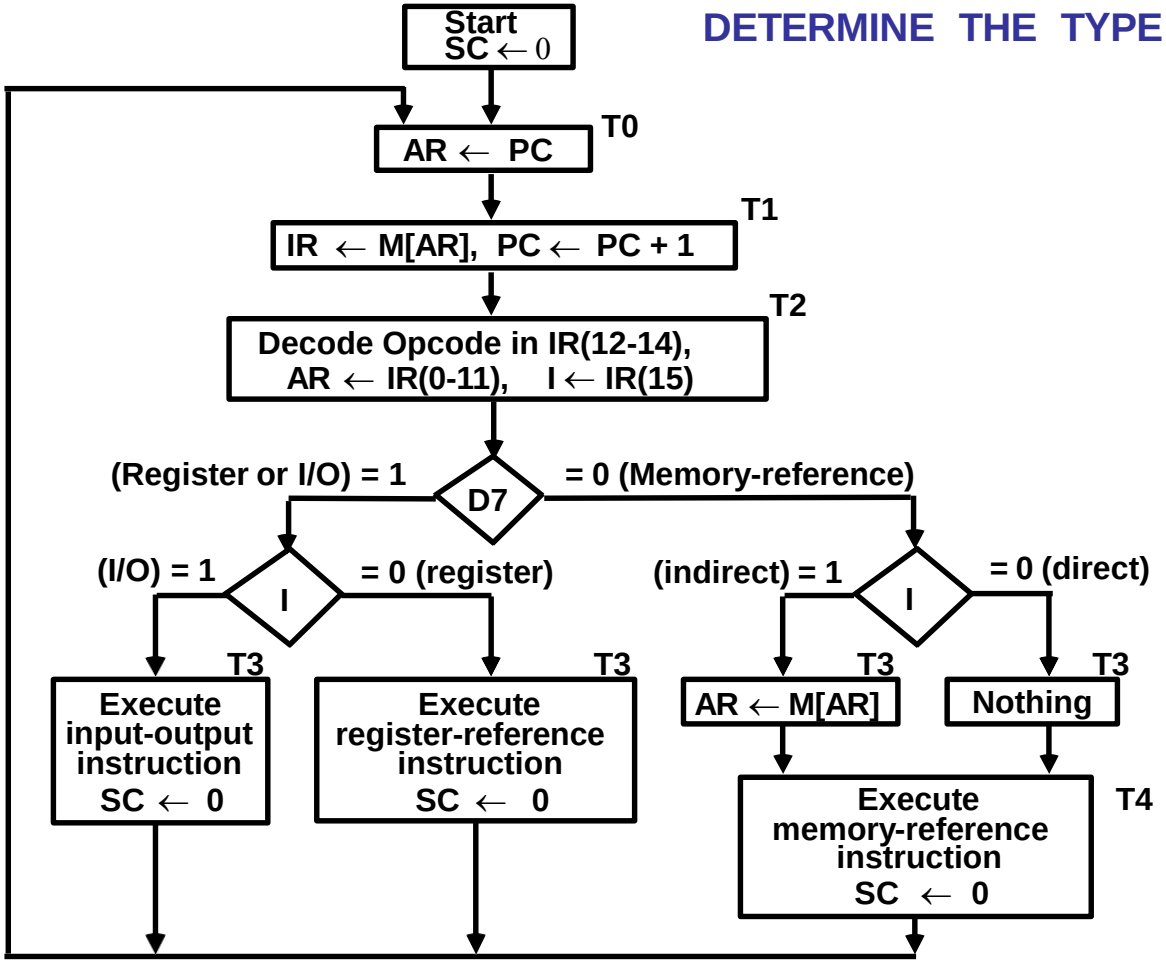
– T_1: IR←M[AR], PC←PC+1

The instruction is fetched from the memory to IR ,and the PC is incremented.

Decode

– T_2: D_0,…, D_7←Decode IR(12-14), AR←IR(0-11), I←IR(15)

DETERMINE THE TYPE OF INSTRUCTION



D'_7IT_3: $AR \leftarrow M[AR]$

$D'_7I'T_3$: Nothing

$D_7I'T_3$: Execute a register-reference instr.

D_7IT_3: Execute an input-output instr.

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	$M[AR] \leftarrow M[AR] + 1$, if $M[AR] + 1 = 0$ 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 T3 when I = 1
- 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

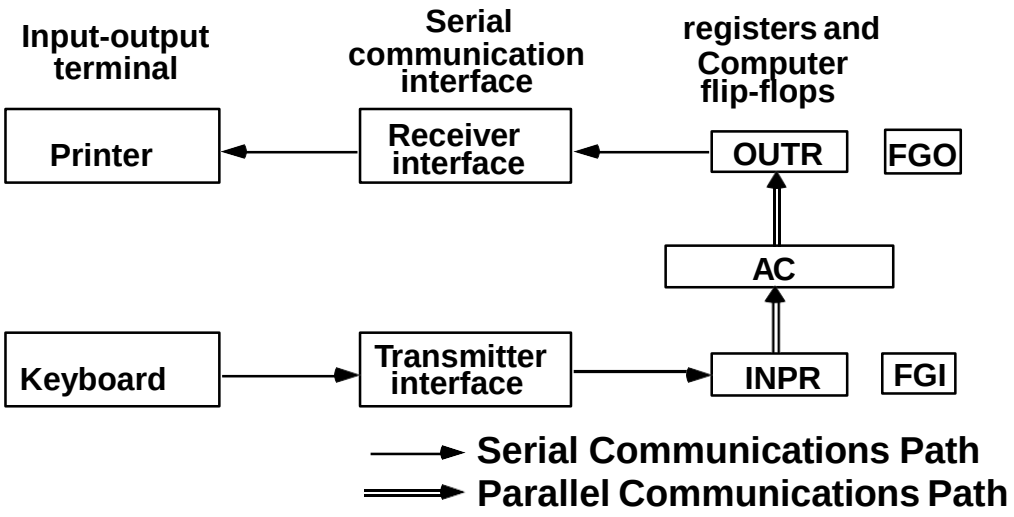
D_1T_4: $DR \leftarrow M[AR]$ — Read operand

D_1T_5: $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$ — Add to AC and store carry in E

Input-Output Devices

- Instructions and data stored in memory must come from some input devices
- Computational results must be transmitted to the user through some output device
- For the system to communicate with an input device, serial information is shifted into the input register INPR
- To output information, it is stored in the output register OUTR

Input-Output Devices



Input-Output Devices cont.

- INPR and OUTR communicate with a communication interface serially and with the AC in parallel. They hold an 8-bit alphanumeric information
- I/O devices are slower than a computer system → we need to synchronize the timing rate difference between the input/output device and the computer.
- FGI: 1-bit input flag (Flip-Flop) aimed to control the input operation

Input-Output Devices cont.

- FGI is set to 1 when a new information is available in the input device and is cleared to 0 when the information is accepted by the computer
- FGO: 1-bit output flag used as a control flip-flop to control the output operation
- If FGO is set to 1, then this means that the computer can send out the information from AC. If it is 0, then the output device is busy and the computer has to wait!

Design of Basic Computer

1. **A memory unit: 4096 x 16.**
2. **Registers: AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC**
3. **Flip-Flops (Status): I, E, FGI, and FGO**
4. **Decoders:**
    1. **a 3x8 Opcode decoder**
    2. **a 4x16 timing decoder**
5. **Common bus: 16 bits**
6. **Control logic gates**
7. **Adder and Logic circuit: Connected to AC**

Design of Basic Computer cont.

- The control logic gates are used to control:
  - Inputs of the nine registers
  - Read and Write inputs of memory
  - Set, Clear, or Complement inputs of the flip-flops
  - S2, S1, S0 that select a register for the bus
  - AC Adder and Logic circuit