

Overview

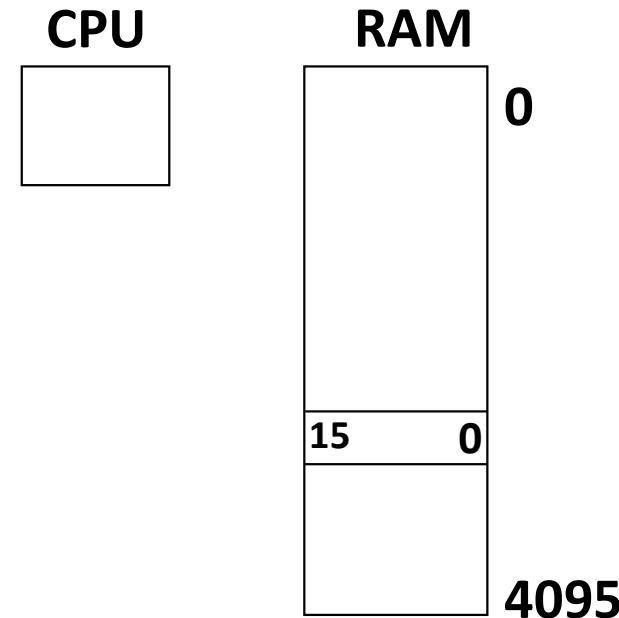
- **Instruction Codes**
- **Computer Registers**
- Computer Instructions
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description

Introduction

- Every different processor type has its own design (different registers, buses, microoperations, machine instructions, etc)
- Modern processor is a very complex device
- It contains
 - Many registers
 - Multiple arithmetic units, for both integer and floating point calculations
 - The ability to pipeline several consecutive instructions to speed execution
 - Etc.
- However, to understand how processors work, we will start with a simplified processor model

Basic Computer

- The Basic Computer has two components, a processor and memory
- The memory has 4096 words in it
 - $4096 = 2^{12}$, so it takes 12 bits to select a word in memory
- Each word is 16 bits long



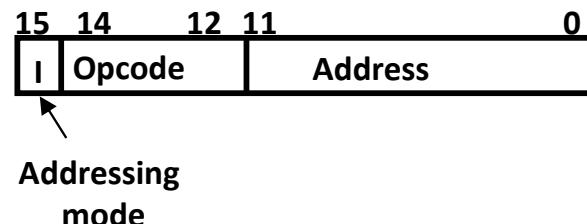
Instruction

- Program
 - A sequence of (machine) instructions
- (Machine) Instruction
 - A group of bits that tell the computer to *perform a specific operation* (a sequence of micro-operation)
- The instructions of a program, along with any needed data are stored in memory
- The CPU reads the next instruction from memory
- It is placed in an Instruction Register (IR)
- Control circuitry in control unit then translates the instruction into the sequence of microoperations necessary to implement it

Instruction Format

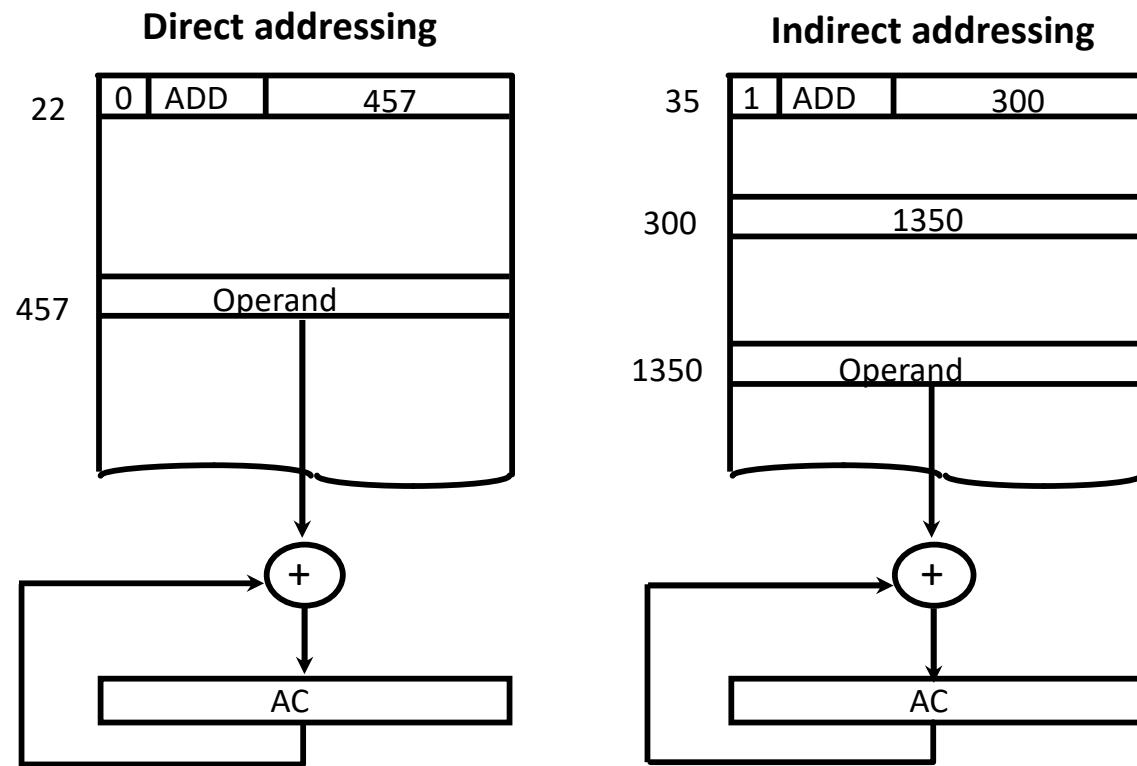
- A computer instruction is often divided into two parts
 - An opcode (Operation Code) that specifies the operation for that instruction
 - An address that specifies the registers and/or locations in memory to use for that operation
- In the Basic Computer, since the memory contains 4096 ($= 2^{12}$) words, we need 12 bit to specify which memory address this instruction will use
- In the Basic Computer, bit 15 of the instruction specifies the addressing mode (0: direct addressing, 1: indirect addressing)
- Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode

Instruction Format



Addressing Mode

- The address field of an instruction can represent either
 - Direct address: the address in memory of the data to use (the address of the operand), or
 - Indirect address: the address in memory of the address in memory of the data to use



- Effective Address (EA)**
 - The address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch-type instruction

Processor Register

- A processor has many registers to hold instructions, addresses, data, etc
- The processor has a register, the Program Counter (PC) that holds the memory address of the next instruction to get
 - Since the memory in the Basic Computer only has 4096 locations, the PC only needs 12 bits
- In a direct or indirect addressing, the processor needs to keep track of what locations in memory it is addressing: The Address Register (AR) is used for this
 - The AR is a 12 bit register in the Basic Computer
- When an operand is found, using either direct or indirect addressing, it is placed in the Data Register (DR). The processor then uses this value as data for its operation
- The Basic Computer has a single general purpose register – the Accumulator (AC)

Processor Register

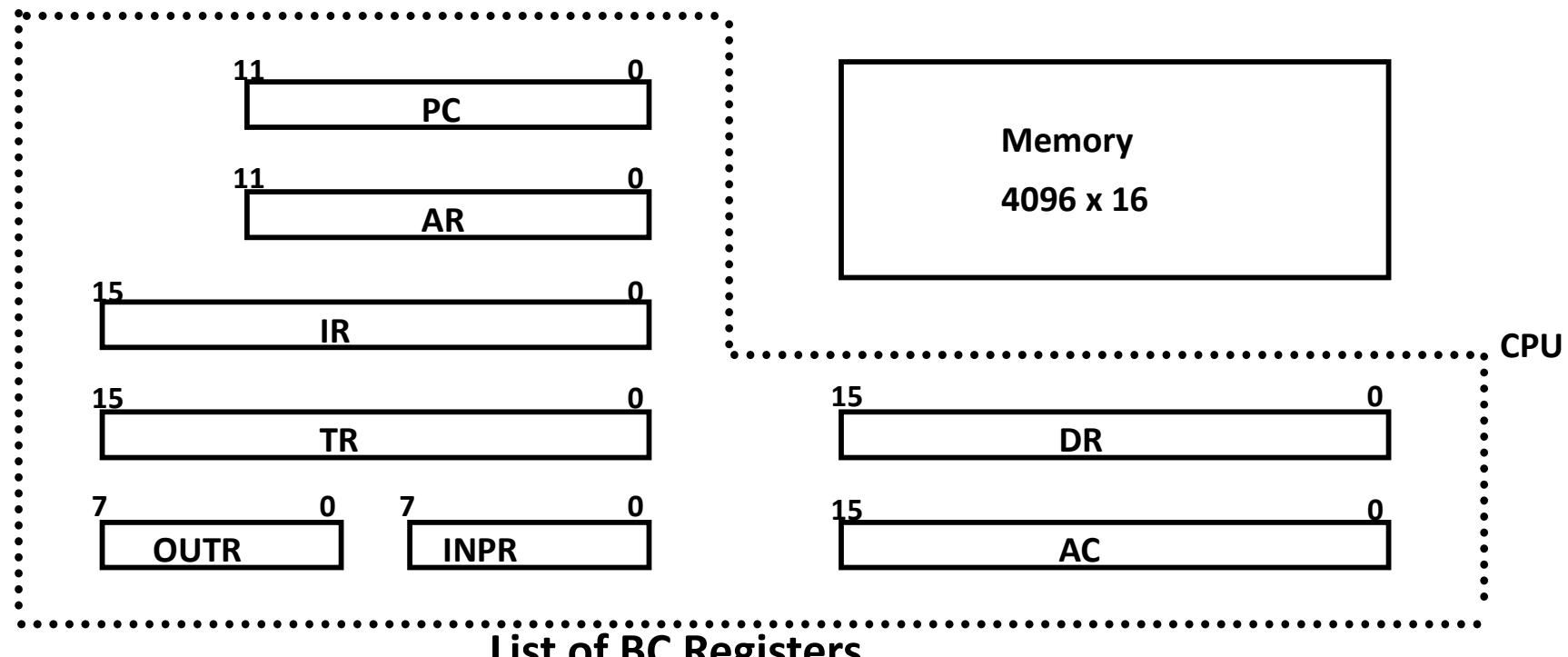
- The significance of a general purpose register is that it can be referred to in instructions
 - e.g. load AC with the contents of a specific memory location; store the contents of AC into a specified memory location
- Often a processor will need a scratch register to store intermediate results or other temporary data; in the Basic Computer this is the Temporary Register (TR)
- The Basic Computer uses a very simple model of input/output (I/O) operations
 - Input devices are considered to send 8 bits of character data to the processor
 - The processor can send 8 bits of character data to output devices
- The Input Register (INPR) holds an 8 bit character gotten from an input device
- The Output Register (OUTR) holds an 8 bit character to be send to an output device

Overview

- Instruction Codes
- **Computer Registers**
- Computer Instructions
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description

Processor Register

Registers in the Basic Computer



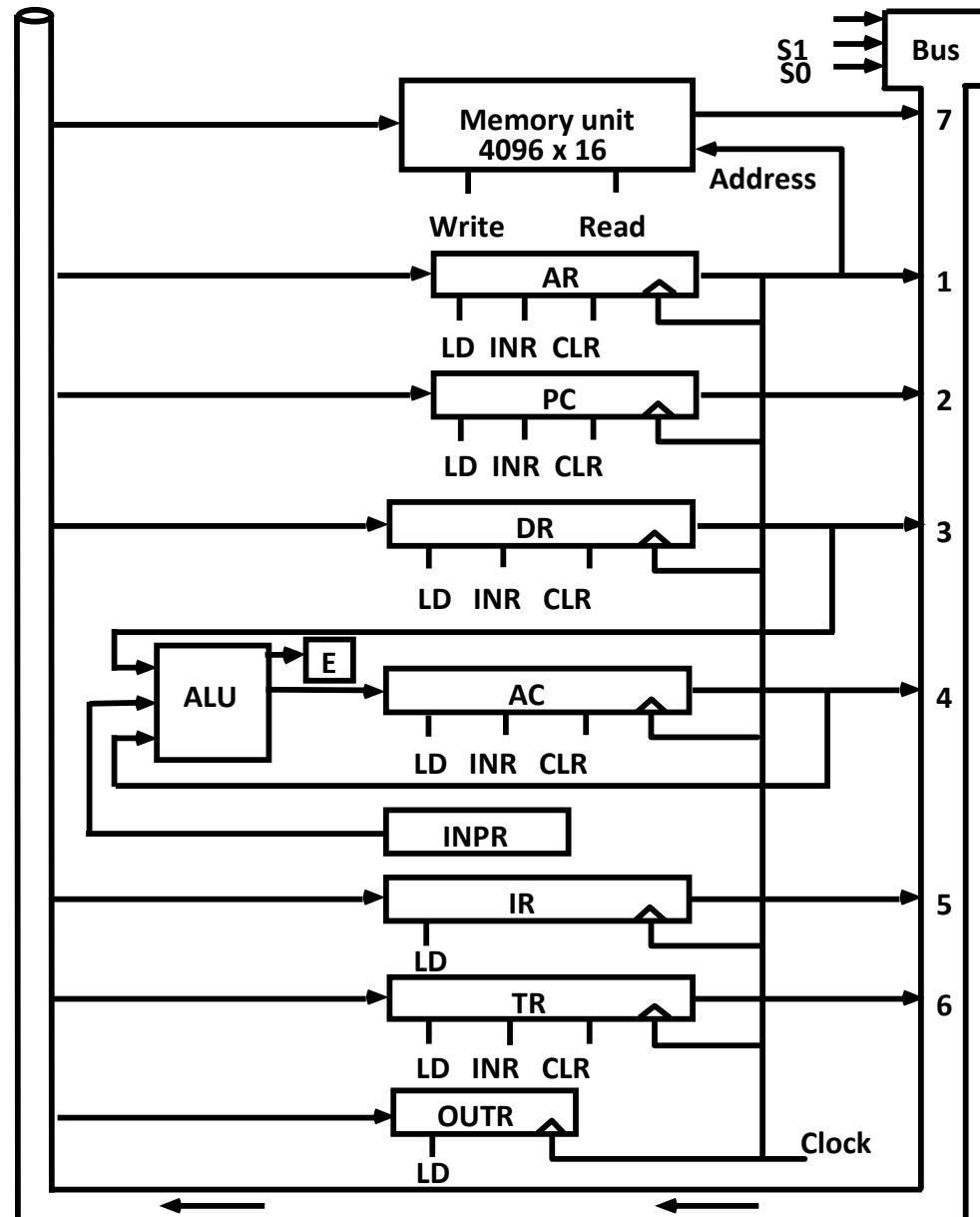
List of BC Registers

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

Common Bus System

- The registers in the Basic Computer are connected using a bus
- This gives a savings in circuitry over complete connections between registers

Common Bus System



Common Bus System

- Three control lines, S_2 , S_1 , and S_0 control which register the bus selects as its input

$S_2\ S_1\ S_0$	Register
0 0 0	x
0 0 1	AR
0 1 0	PC
0 1 1	DR
1 0 0	AC
1 0 1	IR
1 1 0	TR
1 1 1	Memory

- Either one of the registers will have its load signal activated, or the memory will have its read signal activated
 - Will determine where the data from the bus gets loaded
- The 12-bit registers, AR and PC, have 0's loaded onto the bus in the high order 4 bit positions
- When the 8-bit register OUTR is loaded from the bus, the data comes from the low order 8 bits on the bus

Overview

- Instruction Codes
- Computer Registers
- **Computer Instructions**
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description

Basic Computer Instructions

Basic Computer Instruction Format

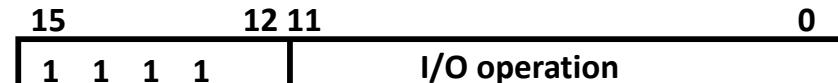
1. Memory-Reference Instructions (OP-code = 000 ~ 110)



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



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



Basic Computer Instructions

<i>Symbol</i>	<i>Hex Code</i>		<i>Description</i>
	<i>I = 0</i>	<i>I = 1</i>	
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
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
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

Instruction Set Completeness

A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable.

Instruction Types

➤ Functional Instructions

- Arithmetic, logic, and shift instructions
- ADD, CMA, INC, CIR, CIL, AND, CLA

➤ Transfer Instructions

- Data transfers between the main memory and the processor registers
- LDA, STA

➤ Control Instructions

- Program sequencing and control
- BUN, BSA, ISZ

➤ Input/output Instructions

- Input and output
- INP, OUT

Overview

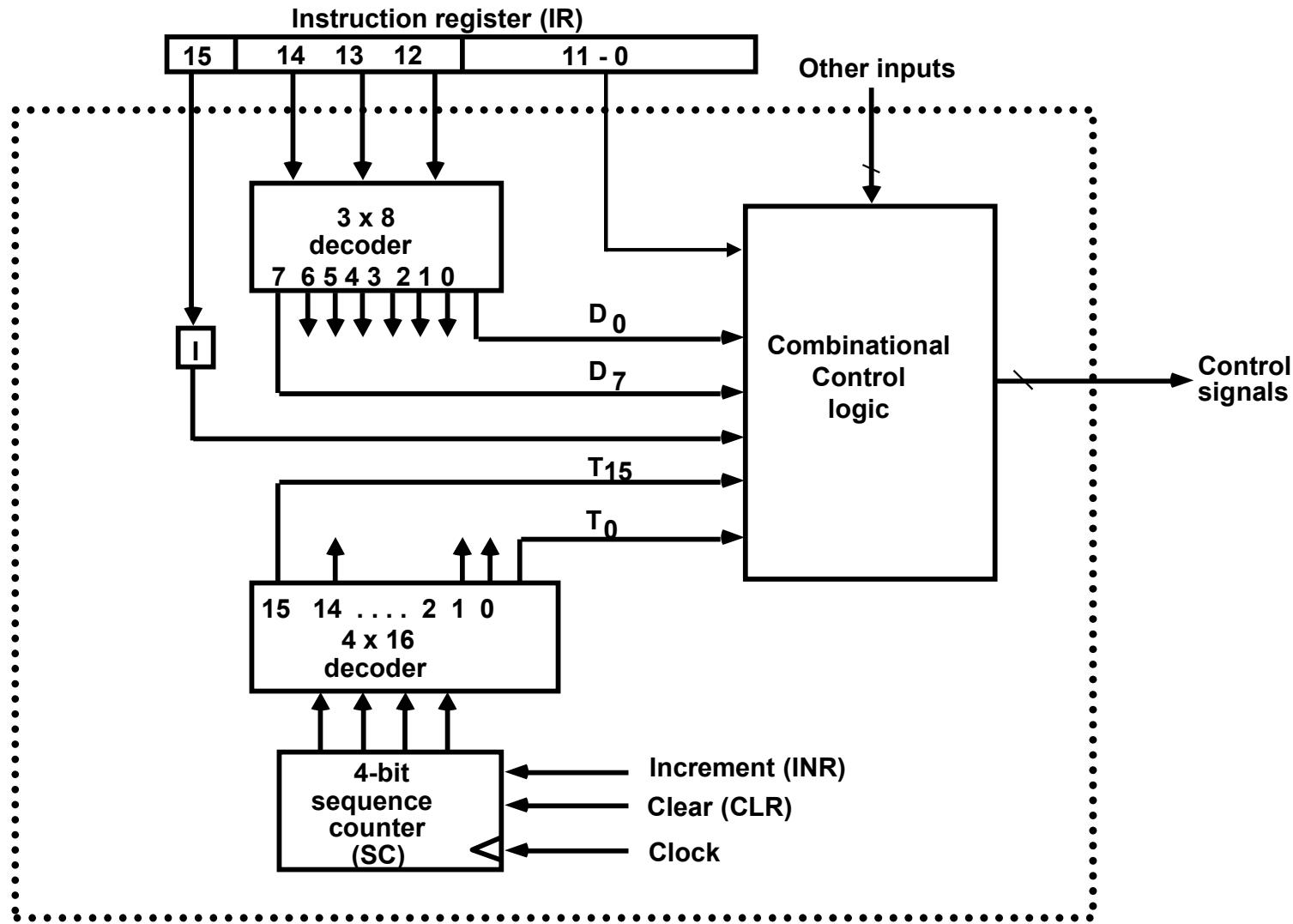
- Instruction Codes
- Computer Registers
- Computer Instructions
- **Timing and Control**
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description

Control Unit

- **Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them**
- **Control units are implemented in one of two ways**
 - Hardwired Control**
CU is made up of sequential and combinational circuits to generate the control signals
 - Microprogrammed Control**
A control memory on the processor contains microprograms that activate the necessary control signals
- **We will consider a hardwired implementation of the control unit for the Basic Computer**

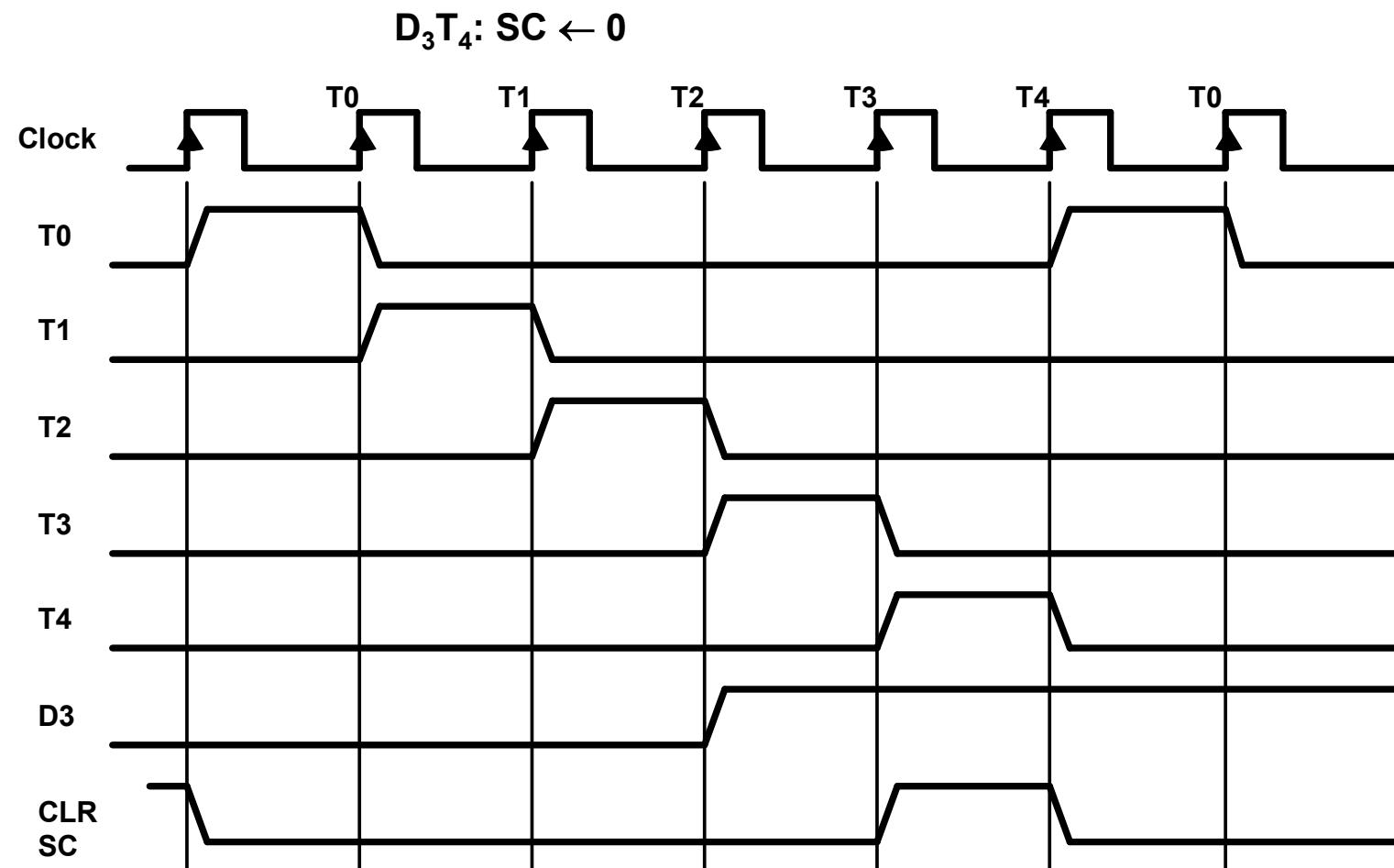
Timing and Control

Control unit of Basic Computer



Timing Signals

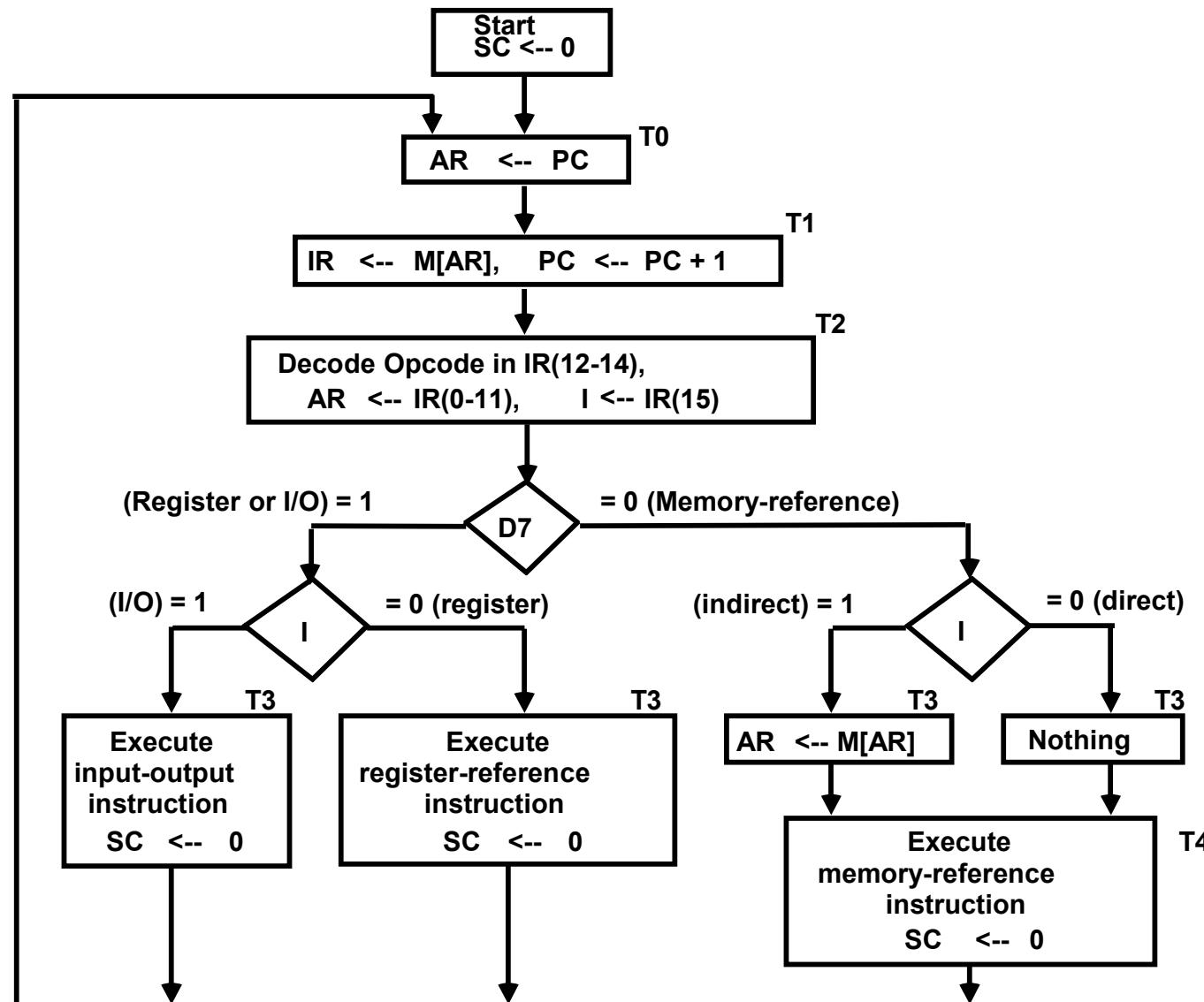
- Generated by 4-bit sequence counter and 4×16 decoder
- The SC can be incremented or cleared.
- Example: $T_0, T_1, T_2, T_3, T_4, T_0, T_1, \dots$
Assume: At time T_4 , SC is cleared to 0 if decoder output D3 is active.



Overview

- Instruction Codes
- Computer Registers
- Computer Instructions
- Timing and Control
- **Instruction Cycle**
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description

Flow Chart (Instruction Cycle)



Determining Type of Instruction

- D'7IT3: $AR \leftarrow M[AR]$
- D'7I'T3: Nothing
- D7I'T3: Execute a register-reference instr.
- D7IT3: Execute an input-output instr.

Register Reference Instruction

Register Reference Instructions are identified when

- $D_7 = 1, I = 0$
- Register Ref. Instr. is specified in $b_0 \sim b_{11}$ of IR
- Execution starts with timing signal T_3

$r = D_7 I' T_3 \Rightarrow$ Register Reference Instruction

$B_i = IR(i), i=0,1,2,\dots,11$

	$r:$	$SC \leftarrow 0$
CLA	$rB_{11}:$	$AC \leftarrow 0$
CLE	$rB_{10}:$	$E \leftarrow 0$
CMA	$rB_9:$	$AC \leftarrow AC'$
CME	$rB_8:$	$E \leftarrow E'$
CIR	$rB_7:$	$AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6:$	$AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_5:$	$AC \leftarrow AC + 1$
SPA	$rB_4:$	if $(AC(15) = 0)$ then $(PC \leftarrow PC+1)$
SNA	$rB_3:$	if $(AC(15) = 1)$ then $(PC \leftarrow PC+1)$
SZA	$rB_2:$	if $(AC = 0)$ then $(PC \leftarrow PC+1)$
SZE	$rB_1:$	if $(E = 0)$ then $(PC \leftarrow PC+1)$
HLT	$rB_0:$	$S \leftarrow 0$ (S is a start-stop flip-flop)

Overview

- Instruction Codes
- Computer Registers
- Computer Instructions
- Timing and Control
- Instruction Cycle
- **Memory Reference Instructions**
- Input-Output and Interrupt
- Complete Computer Description

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 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 MR instruction starts with T₄

AND to AC

D₀T₄: $DR \leftarrow M[AR]$

Read operand

D₀T₅: $AC \leftarrow AC \wedge DR, SC \leftarrow 0$

AND with AC

ADD to AC

D₁T₄: $DR \leftarrow M[AR]$

Read operand

D₁T₅: $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$

Add to AC and store carry in E

Memory Reference Instructions

LDA: Load to AC

$D_2 T_4: DR \leftarrow M[AR]$

$D_2 T_5: AC \leftarrow DR, SC \leftarrow 0$

STA: Store AC

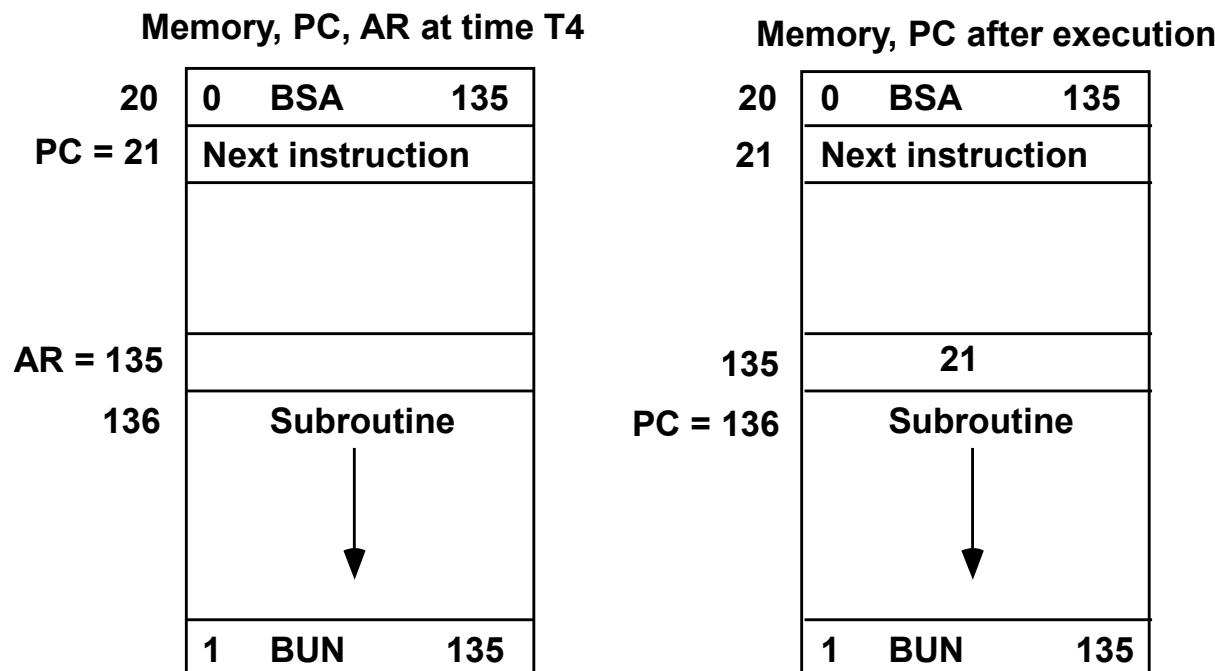
$D_3 T_4: M[AR] \leftarrow AC, SC \leftarrow 0$

BUN: Branch Unconditionally

$D_4 T_4: PC \leftarrow AR, SC \leftarrow 0$

BSA: Branch and Save Return Address

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



Memory Reference Instructions

BSA:

$D_5 T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$

$D_5 T_5: PC \leftarrow AR, SC \leftarrow 0$

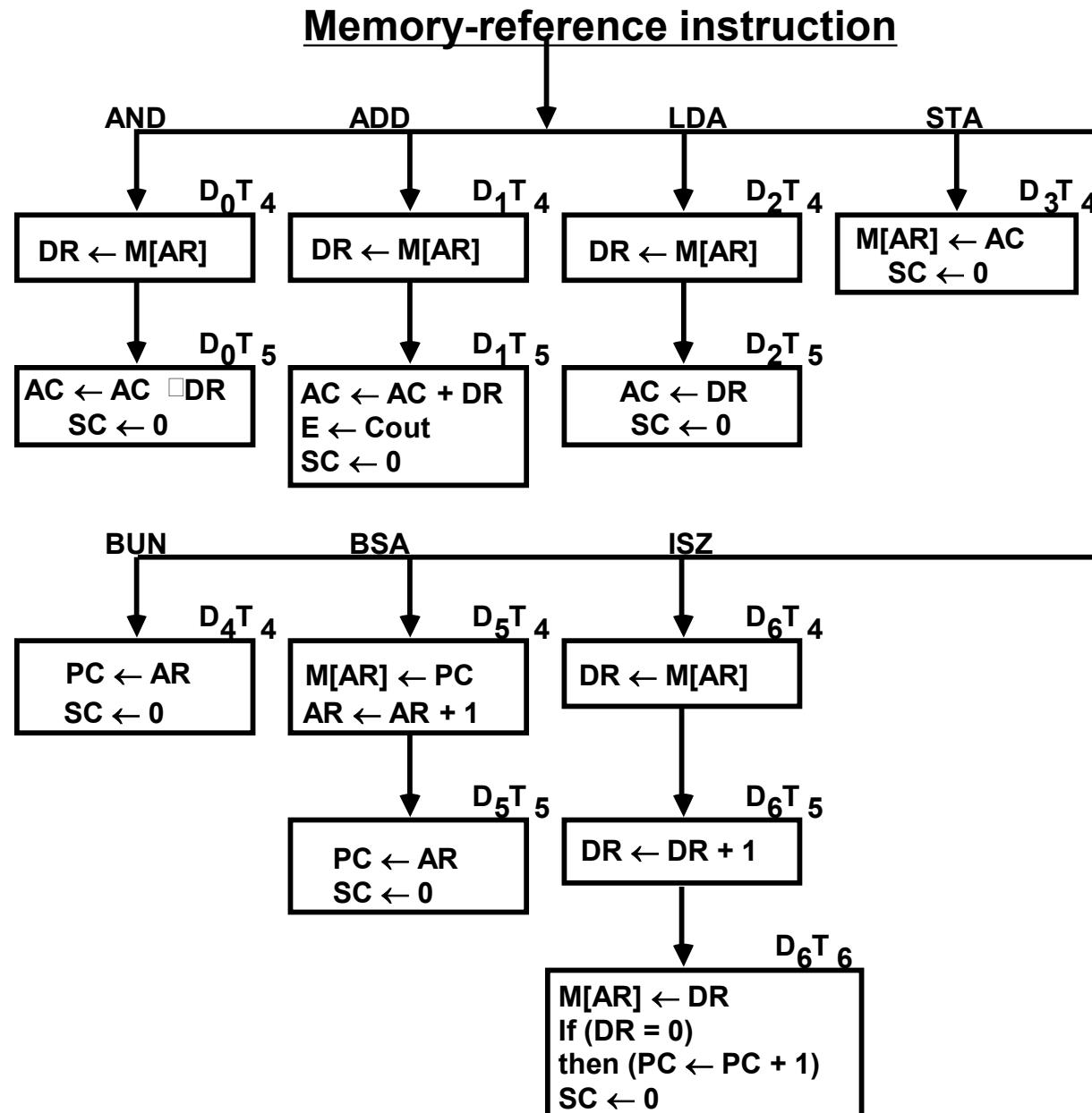
ISZ: Increment and Skip-if-Zero

$D_6 T_4: DR \leftarrow M[AR]$

$D_6 T_5: DR \leftarrow DR + 1$

$D_6 T_4: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$

Flow Chart - Memory Reference Instructions



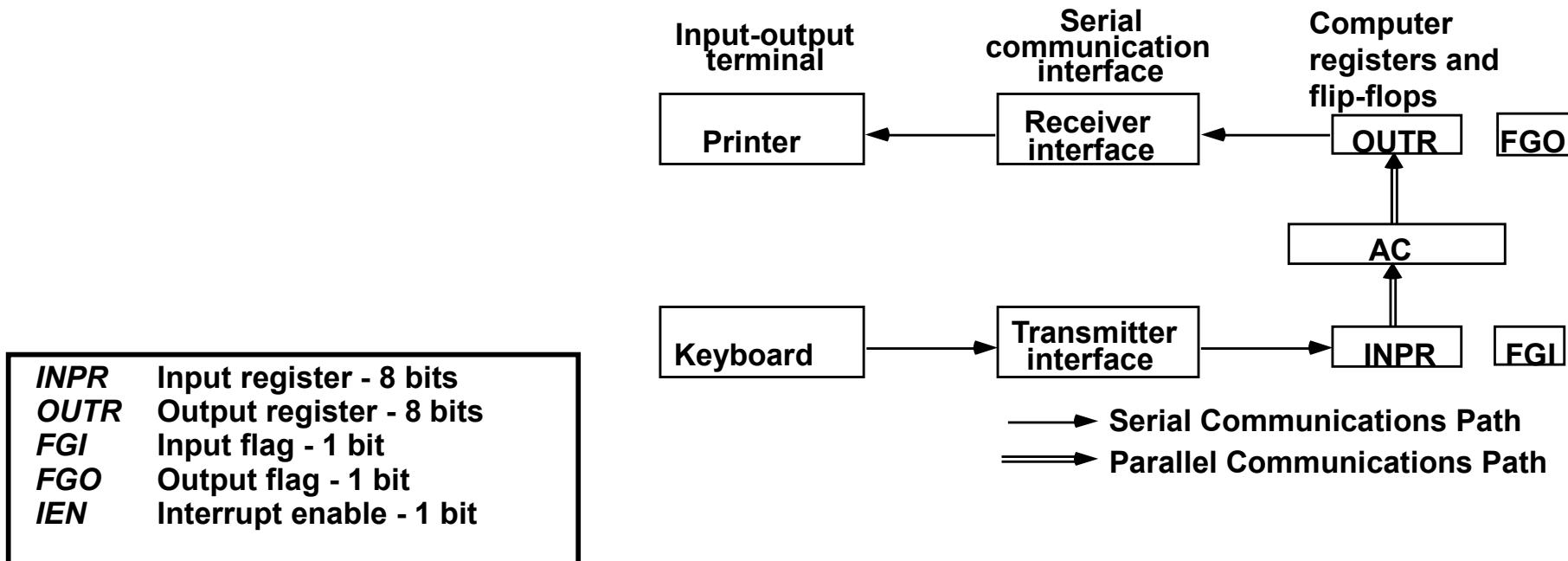
Overview

- Instruction Codes
- Computer Registers
- Computer Instructions
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- **Input-Output and Interrupt**
- Complete Computer Description

Input/Output and Interrupt

A Terminal with a keyboard and a Printer

Input-Output Configuration



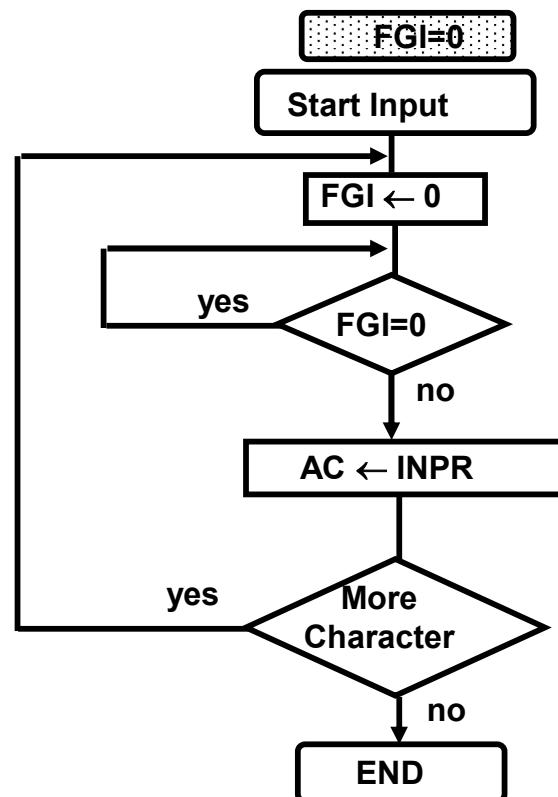
- The terminal sends and receives serial information
- The serial info. from the keyboard is shifted into INPR
- The serial info. for the printer is stored in the OUTR
- INPR and OUTR communicate with the terminal serially and with the AC in parallel.
- The flags are needed to synchronize the timing difference between I/O device and the computer

Programmed Controlled Data Transfer

-- CPU --

```
/* Input */      /* Initially FGI = 0 */
loop: If FGI = 0 goto loop
      AC ← INPR, FGI ← 0
```

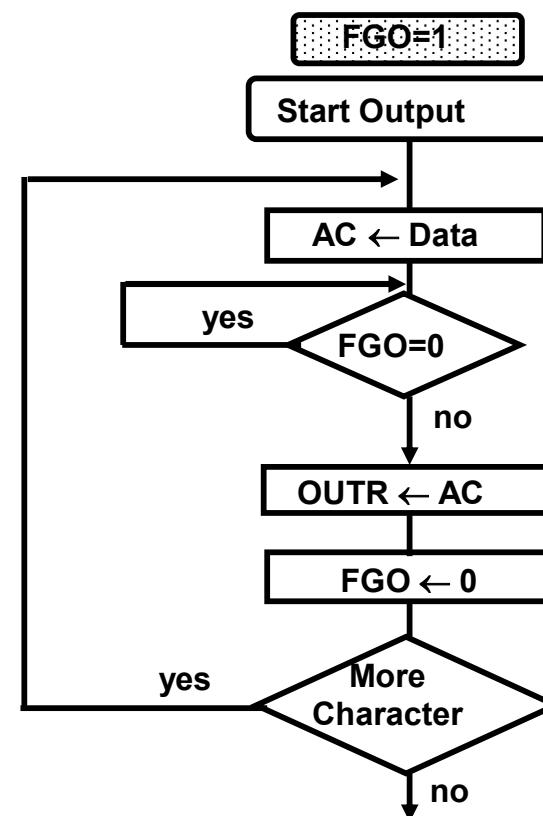
```
/* Output */     /* Initially FGO = 1 */
loop: If FGO = 0 goto loop
      OUTR ← AC, FGO ← 0
```



-- I/O Device --

```
loop: If FGI = 1 goto loop
      INPR ← new data, FGI ← 1
```

```
loop: If FGO = 1 goto loop
      consume OUTR, FGO ← 1
```



Input/Output Instructions

$$D_7IT_3 = p$$

$$IR(i) = B_i, i = 6, \dots, 11$$

	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11} :	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input char. to AC
OUT	pB_{10} :	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	Output char. from AC
SKI	pB_9 :	$if(FGI = 1) then (PC \leftarrow PC + 1)$	Skip on input flag
SKO	pB_8 :	$if(FGO = 1) then (PC \leftarrow PC + 1)$	Skip on output flag
ION	pB_7 :	$IEN \leftarrow 1$	Interrupt enable on
IOF	pB_6 :	$IEN \leftarrow 0$	Interrupt enable off

Program controlled Input/Output

- Program-controlled I/O
 - Continuous CPU involvement
 - I/O takes valuable CPU time
 - CPU slowed down to I/O speed
 - Simple
 - Least hardware

Input

LOOP,	SKI	DEV
	BUN	LOOP
	INP	DEV

Output

LOOP,	LDA	DATA
LOP,	SKO	DEV
	BUN	LOP
	OUT	DEV

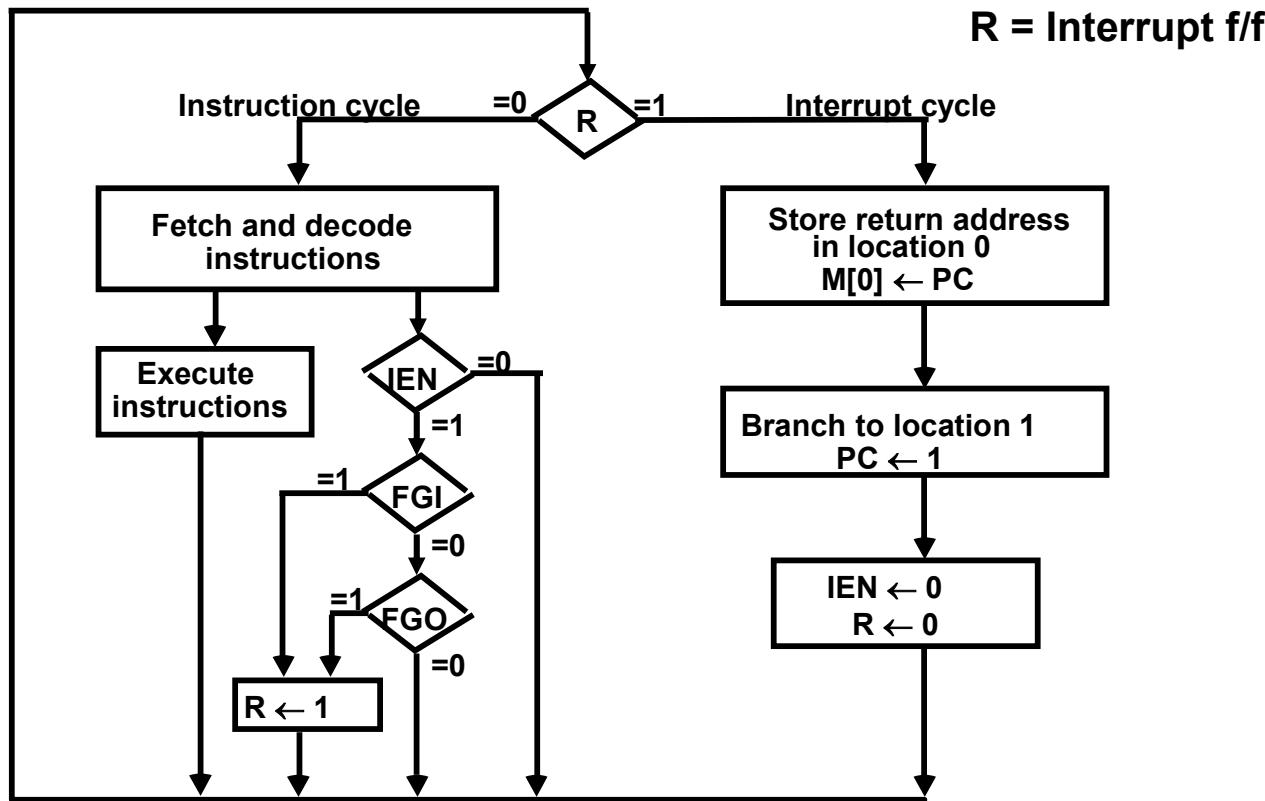
Interrupt Initiated Input/Output

- Open communication only when some data has to be passed --> *interrupt*.
- The I/O interface, instead of the CPU, monitors the I/O device.
- When the interface finds that the I/O device is ready for data transfer, it generates an interrupt request to the CPU
- Upon detecting an interrupt, the CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing.

IEN (Interrupt-enable flip-flop)

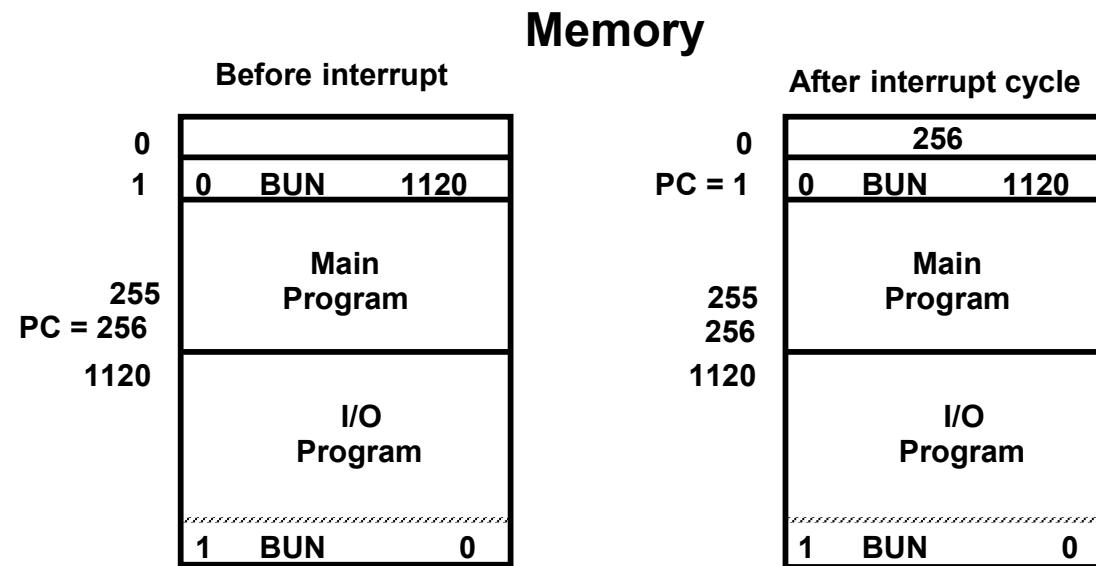
- can be set and cleared by instructions
- when cleared, the computer cannot be interrupted

Flow Chart of Interrupt Cycle



- The interrupt cycle is a HW implementation of a branch and save return address operation.
- At the beginning of the next instruction cycle, the instruction that is read from memory is in address 1.
- At memory address 1, the programmer must store a branch instruction that sends the control to an interrupt service routine
- The instruction that returns the control to the original program is "indirect BUN 0"

Register Transfer Operations in Interrupt Cycle



- The fetch and decode phases of the instruction cycle must be modified → Replace T_0, T_1, T_2 with $R'T_0, R'T_1, R'T_2$

- The interrupt cycle :

$RT_0: AR \leftarrow 0, TR \leftarrow PC$

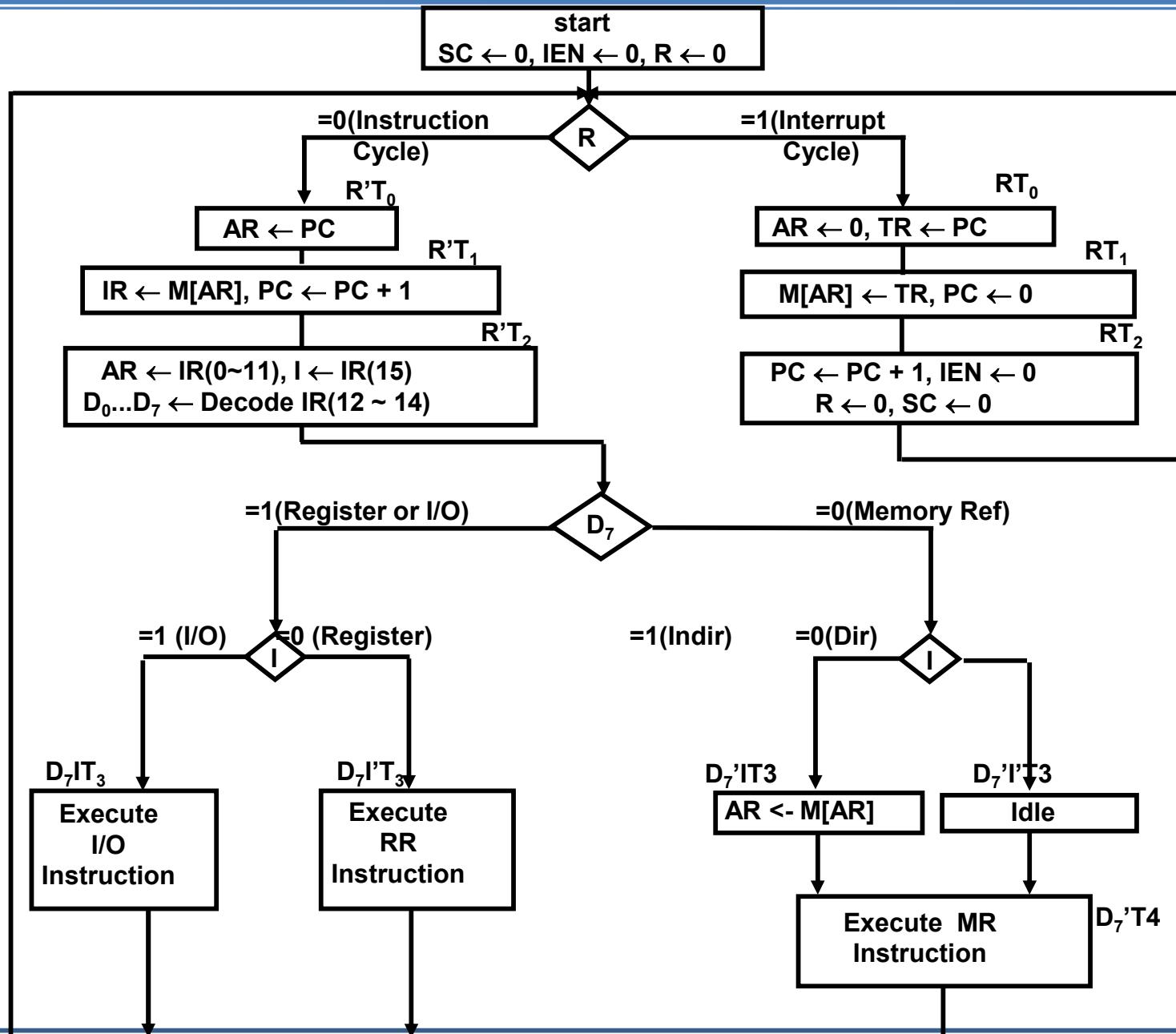
$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$

$RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$

Overview

- Instruction Codes
- Computer Registers
- Computer Instructions
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- **Complete Computer Description**

Complete Computer Design



Complete Computer Design

Fetch	$R'T_0:$	$AR \leftarrow PC$
	$R'T_1:$	$IR \leftarrow M[AR], PC \leftarrow PC + 1$
Decode	$R'T_2:$	$D_0, \dots, D_7 \leftarrow \text{Decode } IR(12 \sim 14),$ $AR \leftarrow IR(0 \sim 11), I \leftarrow IR(15)$
Indirect Interrupt	$D_7'IT_3:$	$AR \leftarrow M[AR]$
		$R \leftarrow 1$
	$RT_0:$	$AR \leftarrow 0, TR \leftarrow PC$
	$RT_1:$	$M[AR] \leftarrow TR, PC \leftarrow 0$
	$RT_2:$	$PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
Memory-Reference		
AND	$D_0T_4:$	$DR \leftarrow M[AR]$
	$D_0T_5:$	$AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	$D_1T_4:$	$DR \leftarrow M[AR]$
	$D_1T_5:$	$AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$
LDA	$D_2T_4:$	$DR \leftarrow M[AR]$
	$D_2T_5:$	$AC \leftarrow DR, SC \leftarrow 0$
STA	$D_3T_4:$	$M[AR] \leftarrow AC, SC \leftarrow 0$
BUN	$D_4T_4:$	$PC \leftarrow AR, SC \leftarrow 0$
BSA	$D_5T_4:$	$M[AR] \leftarrow PC, AR \leftarrow AR + 1$
	$D_5T_5:$	$PC \leftarrow AR, SC \leftarrow 0$
ISZ	$D_6T_4:$	$DR \leftarrow M[AR]$
	$D_6T_5:$	$DR \leftarrow DR + 1$
	$D_6T_6:$	$M[AR] \leftarrow DR, \text{ if}(DR=0) \text{ then } (PC \leftarrow PC + 1),$ $SC \leftarrow 0$

Complete Computer Design

Register-Reference

	$D_7 I' T_3 = r$	(Common to all register-reference instr)
	$IR(i) = B_i$	($i = 0, 1, 2, \dots, 11$)
	r:	$SC \leftarrow 0$
CLA	$rB_{11}:$	$AC \leftarrow 0$
CLE	$rB_{10}:$	$E \leftarrow 0$
CMA	$rB_9:$	$AC \leftarrow AC'$
CME	$rB_8:$	$E \leftarrow E'$
CIR	$rB_7:$	$AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6:$	$AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_5:$	$AC \leftarrow AC + 1$
SPA	$rB_4:$	If($AC(15) = 0$) then ($PC \leftarrow PC + 1$)
SNA	$rB_3:$	If($AC(15) = 1$) then ($PC \leftarrow PC + 1$)
SZA	$rB_2:$	If($AC = 0$) then ($PC \leftarrow PC + 1$)
SZE	$rB_1:$	If($E=0$) then ($PC \leftarrow PC + 1$)
HLT	$rB_0:$	$S \leftarrow 0$

Input-Output

	$D_7 IT_3 = p$	(Common to all input-output instructions)
	$IR(i) = B_i$	($i = 6, 7, 8, 9, 10, 11$)
	p:	$SC \leftarrow 0$
INP	$pB_{11}:$	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_{10}:$	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_9:$	If($FGI=1$) then ($PC \leftarrow PC + 1$)
SKO	$pB_8:$	If($FGO=1$) then ($PC \leftarrow PC + 1$)
ION	$pB_7:$	$IEN \leftarrow 1$
IOF	$pB_6:$	$IEN \leftarrow 0$

Design of a Basic Computer(BC)

Hardware Components of BC

A memory unit: 4096 x 16.

Registers:

AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC

Flip-Flops(Status):

I, S, E, R, IEN, FGI, and FGO

Decoders: a 3x8 Opcode decoder

a 4x16 timing decoder

Common bus: 16 bits

Control logic gates:

Adder and Logic circuit: Connected to AC

Control Logic Gates

- Input Controls of the nine registers
- Read and Write Controls of memory
- Set, Clear, or Complement Controls of the flip-flops
- S_2, S_1, S_0 Controls to select a register for the bus
- AC, and Adder and Logic circuit