UNIT-II

BASIC COMPUTER ORGANIZATION AND DESIGN

Instruction Codes

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.

Every different processor type has its own design

M. Morris Mano introduced a simple processor model ,called the **Basic Computer**.

Program: A set of instructions that specifies operation, operands, and sequence of processing has to occur.

The instructions of a program, along with any needed data are stored in memory.

The CPU reads the next instruction from memory and places it in an Instruction Register(IR).

Control circuitry in control unit then translates the instruction into the sequence of micro-operations necessary to implement it .

Computer Instruction: A binary code that specifies a sequence of micro-operations for the computer.

Every computer has its own unique instruction set.

Instruction code: A group of bits instructing the computer to perform a specific operation.

It is divided into parts, each with particular meaning.

Operation code: This is the most important part of instruction code .

It defines the type operation like add, subtract, multiply, shift, and complement.

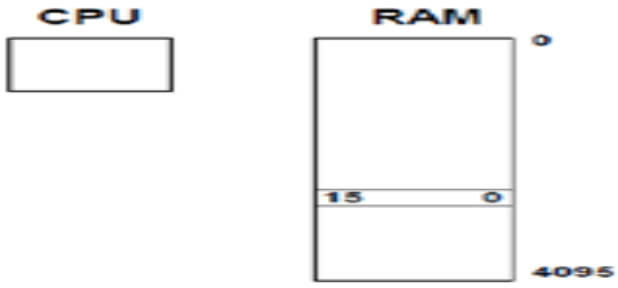
Stored Program Organization

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



A computer instruction is often divided into two parts

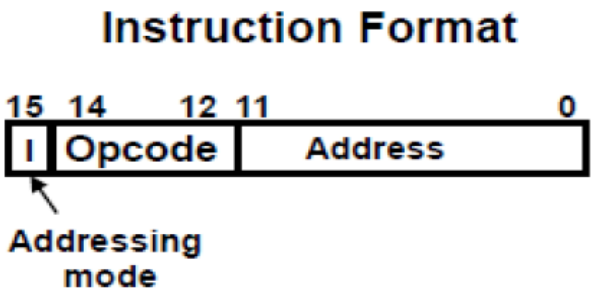
An op-code (Operation Code) that specifies the operation for that instruction

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, so 12 bits 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 op-code .



Addressing Modes

three addressing modes

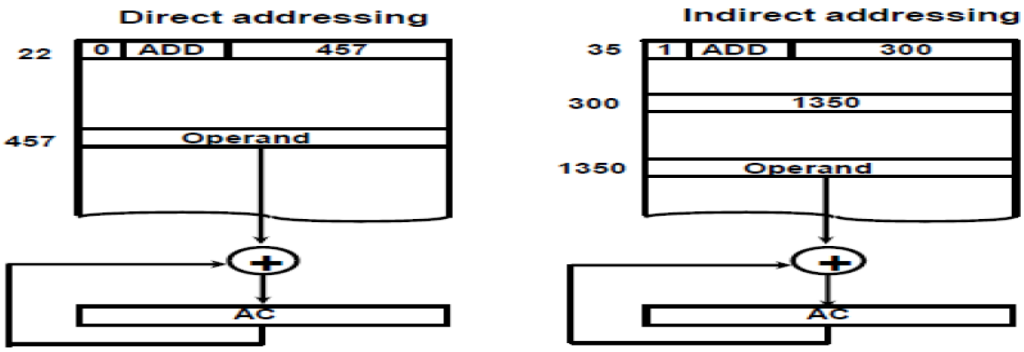
1. **Immediate operand**: when the second part of the instruction code specifies an operand.
2. **Direct address**: when the second part of the instruction code specifies the address of the operand.
3. **Indirect address**: when this part specifies an address in a memory where we can find the true address of the operand.

The (I) bit of the instruction code specifies direct addressing If I=0 or Indirect addressing if I=1.

Accumulator Register (AC): exist in single register processors (AC) and all operations are performed with memory operand and this register.

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



The previous figure shows an example of addressing mode. In location 22 there is an ADD instruction that adds the AC with operand in location 457 as an indication for direct addressing mode as I=0.

On the other hand, the second part of the figure shows in location 35 and ADD instruction between AC and the address of operand found in location 300. Location 300 contains the operand address of 1350. In 1350 the operand will be read and added to the AC register, as I=1 shows.

Computer Registers

A processor has many registers to hold instructions after it has been fetched from memory, addresses of operands need to be accessed data manipulated with accumulator, general purpose register, and others.

Program Counter (PC): used is to hold the memory address of the next instruction to be executed. Size: 12 bits. :
The **Address Register** (AR) is used to keep track of what locations in memory processor is addressing. Size: 12 bit
This AR register is always connected to address pins of memory unit.
Data Register (DR): When an operand is found, using either direct or indirect addressing, it is placed in DR.
The processor then uses this value as data for its operation.
DR is connected to data pins of memory unit.
Accumulator (AC):

The Basic Computer has a single general purpose register–the Accumulator (AC) 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.
AC is always one side in any data computation.

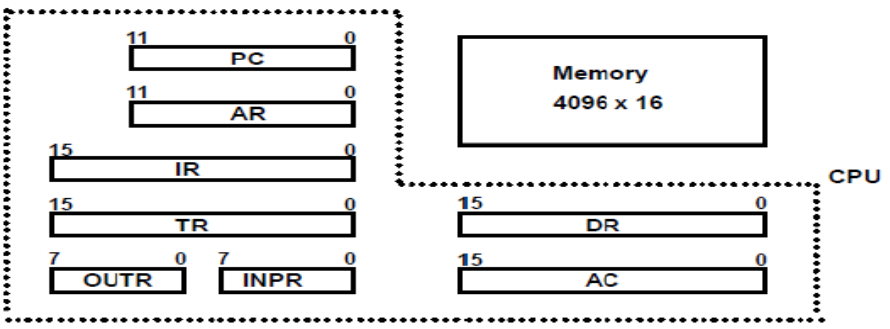
Temporary Register (TR):

a scratch register used to store intermediate results or other temporary data.

-For instruction reading, a PC is used to hold address of next instruction to be fetched and executed.
- First the PC value is copied into AR to start an instruction reading cycle, then the 16 bit instruction is fetched and placed in Instruction Register (IR).
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.
And 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.

Registers in the Basic Computer



List of BC Registers

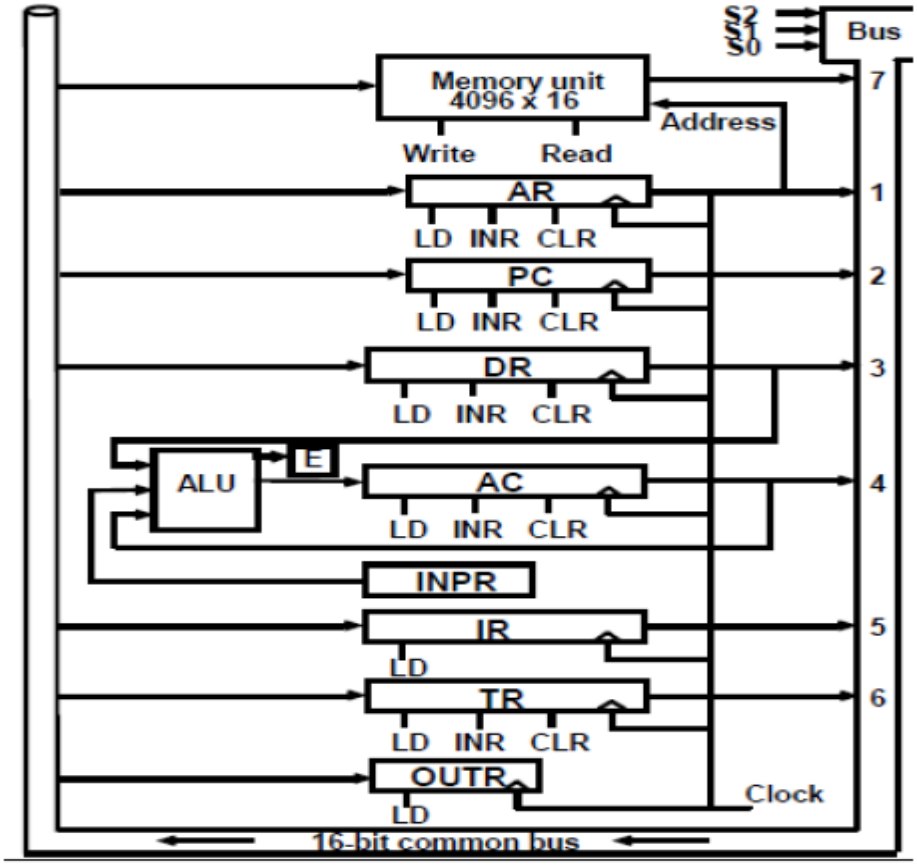
DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds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

Common Bus System

A path needed to transfer data between 8 registers beside memory unit and registers.

So a common bus will be the answer for that problem. The next figure shows the answer consisting of a multiplexer or 3 state buffers with decoder. This gives a savings in circuitry

over complete connections between registers.



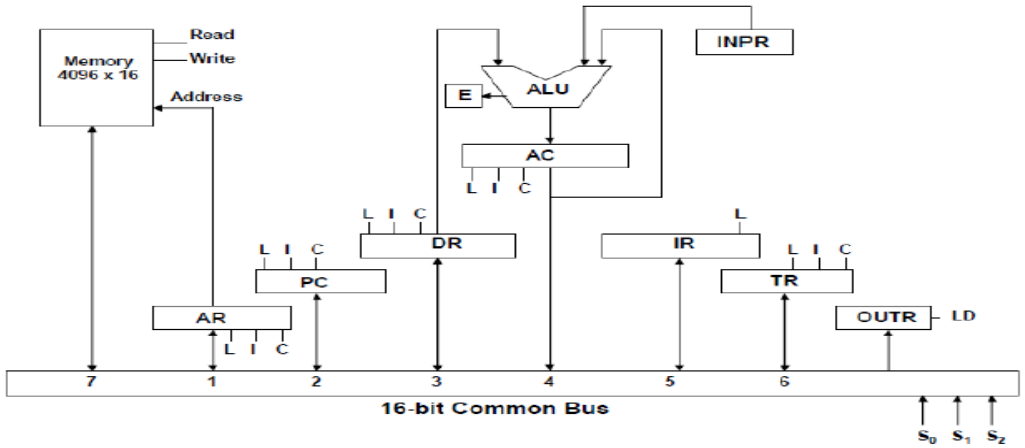
Three control lines, S2, S1, and S0 control which register the bus selects as its input and so the selected register will issue its output to the bus.
The lines from the common bus are connected to the input of each register.
Either one of the registers will have its load signal activated, or the memory will have its read signal activated.
And this will determine where the data from the bus gets loaded to during next clock transition.
The memory will put its content to the bus when S2S1S0 =111 and its read control signal is activated.
In the same manner, the memory will save the content of the bus,when its write control signal is activated.
AR register is always used to hold address of data accessed from memory.
4 registers of 16-bit each, DR, AC, IR, and TR. Also we have two registers PC and AR are 12 bits each. The 12-bit registers, AR and PC, have 0's loaded onto the bus in the high order 4 bit positions.

OUTR and INPR are 8-bit each and are connected to the lower 8 bits of the bus.

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

Five registers have 3 control inputs, LD, INC, CLR; those are AC, AR, TR, PC, and DR.

two registers, OUTR, and IR will only have a LD input.
The data to the AC register comes from ALU unit which accepts operands from AC register, INPR register, or DR register.



Any and only one source of those register can be selected to apply its content to the bus and during the same clock cycle the bus content could be directed to one or many destinations of registers or memory unit. For example we can do the next micro-operation:
DR ← AC and AC ← DR
S2S1S0 = 100
LD of DR is enabled
Transferring DR through ALU to AC
LD of AC is enabled

Computer Instructions

The Basic Computer has 3 instruction code formats

. Each format has 16 bits.

-The op-code of the instruction contains 3 bits and the meaning of the remaining 13 bits depends on the operation code encountered.

- Memory reference instruction uses 12 bits to specify the operand address and one bit for indirect address.

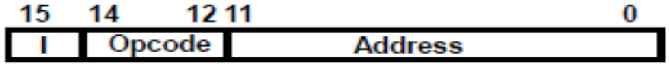
- The register reference instruction are recognized by op-code 111 with 0 in left most bit (Bit 15) of the instruction.

The 12 bits are used to specify the operation done with AC register.

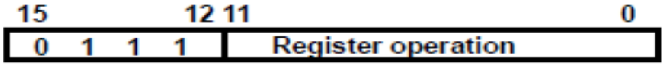
- Input-Output instruction is recognized by op-code 111 and with 1 in bit 15.

The remaining 12 bits are used to specify type of Input-Output instruction type.

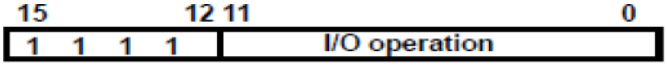
Memory-Reference Instructions (OP-code = 000 ~ 110)



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



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



Functional Instructions includes

Only 3 bits of the instruction are used for the operation code, op-code, and since register reference instructions and IO instructions use the other unneeded 12 bits, so the total number of instruction coded in this computer is 25.

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 will be:

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

INP, OUT

Symbol	Hex Code I = 0	Hex Code I = 1	Description
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

Timing & Control

The timing for all registers is controlled by a master clock.
The clock pulses generated do not change the state of a register unless it is enabled by a control signal.
Control unit (CU) of a processor translates machine instructions to the control signals for the micro-operations that implement them. The control signals are generated in the control unit and provides control inputs to

All register
Multiplexers
Common bus
micro-operation indicators

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

Micro-programmed Control

A control memory on the processor contains micro-programs that activate the necessary control signals

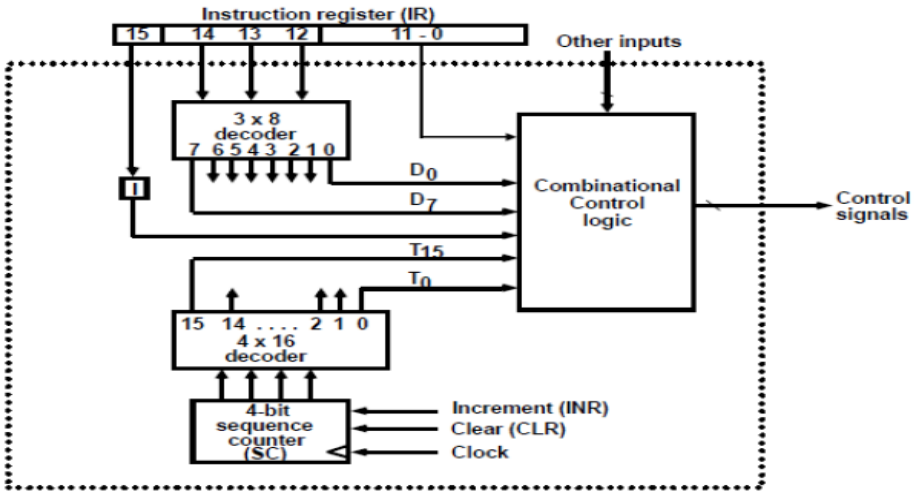
Hardwired control unit of the basic computer:

It consists of 2 decoders, a sequence counter, and a number control logic gates.

The instruction register holds the instruction fetched from memory.

It is divided into 3 parts: I bit, op-code, and 12 bits.
Op-codes is divided by 3 by 8 decoder into 8 different outputs; D0 to D7
Bit 15 is transferred to I flip flop
Bit 0 to Bit 11 (B0 to B11) are applied to control logic gates.

The 4-bit sequence counter is connected to 4 by 16 decoder giving control inputs T0 to T15.



Sequence counter is used to synchronize action with those 16 different time intervals.

Sequence counter can be incremented and cleared synchronously.

Most of the time it is incremented to generate sequence of timing signals

Once in a while it is cleared by specific condition causing next timing sequence to go back to T0.

Example on Timing Control:

Consider a case where SC is incremented to provide timing sequence T0, T1, T2, T3, and T4, then T0 again.

At time T4 Sc is cleared based on a condition D3 is true.

Expressed in symbolic RTL form: D3T4: SC ← 0

The next figure shows its timing diagram.

When D3T4 is true then at first positive clock transition SC is cleared.

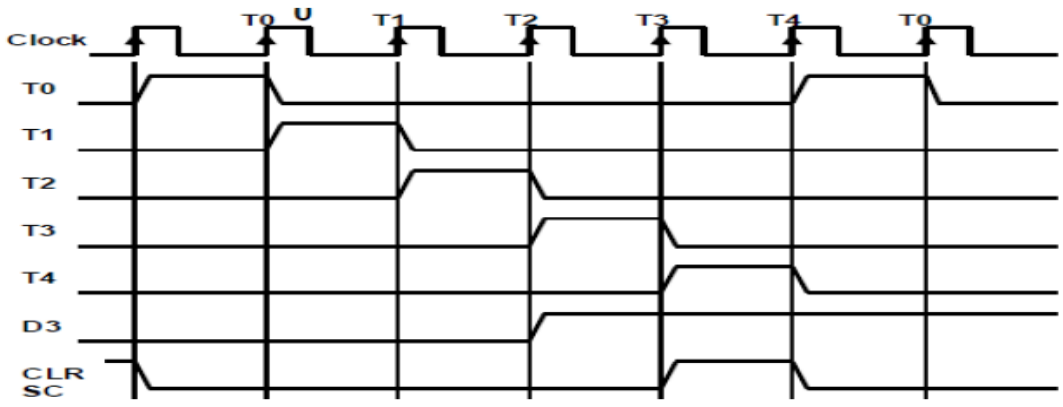
If SC is not cleared then it will continue its counting from T5 to T15 then it rolls over to T0 again.

For a memory read operation, it must be clear that between 2 rising edge of the clock, the data should be read and applied to the bus, so that at next rising edge the data can be saved in destination register.

T0: AR ← PC

That specifies transfer of data from PC to AR register in one clock pulse T0.

The content PC is put on the bus (S2S1S0=010) and LD of AR register is enabled during T0 cycle only.



Instruction Cycle

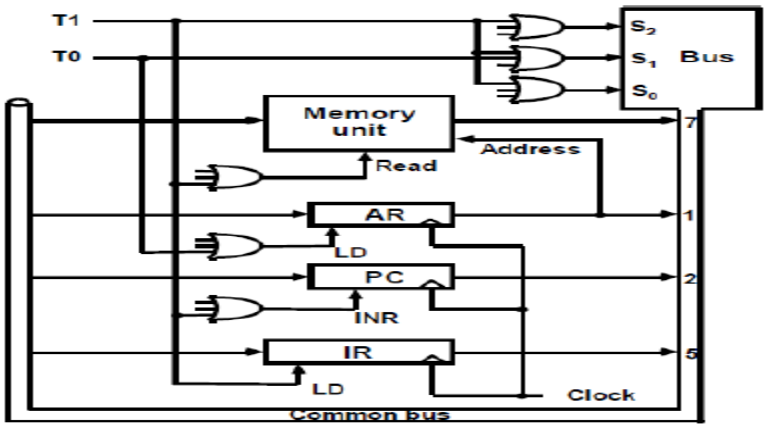
In Basic Computer, a machine instruction is executed in the following cycle:

- Fetch an instruction from memory
- Decode the instruction
- Read the effective address from memory if the instruction has an indirect address
- Execute the instruction

After an instruction is executed, the cycle starts again at step 1, for the next instruction
Fetch and Decode:

Initially the PC is loaded with address with first instruction in the program then SC is cleared giving time instance T0.
After each clock pulse SC is incremented resulting of T1, T2, and so on.

T0: AR ← PC ($S_0S_1S_2$=010, T0=1)
T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)
T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)



A portion of bus system that shows hardware implementation of micro-operations taking place during T0 and T1

During T0

S2S1S0=010 which means PC apply its output to the bus

AR register LD = 1 which means what in the bus goes into AR register

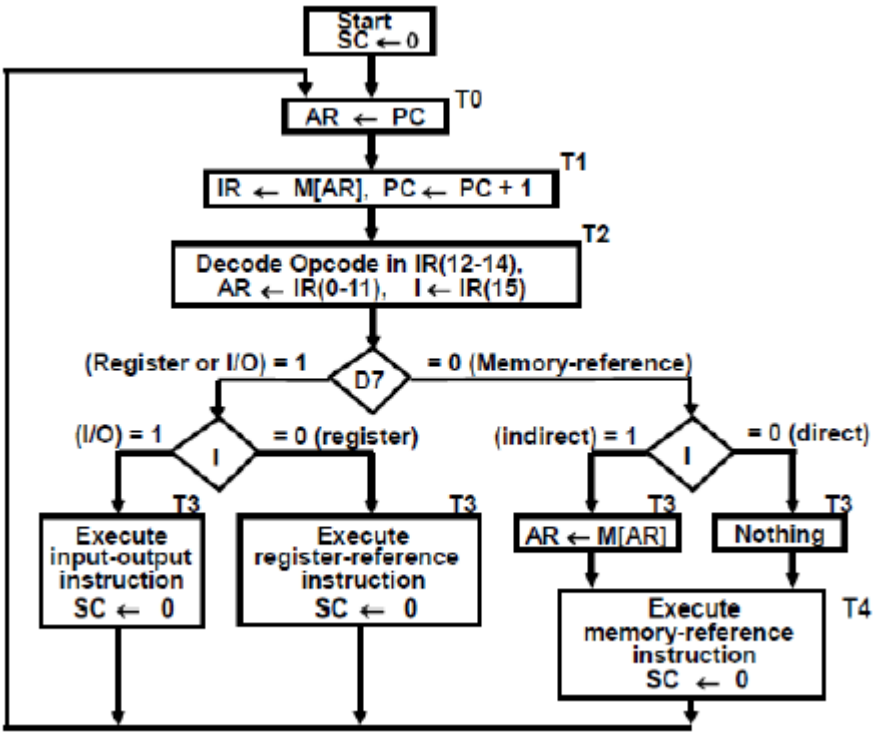
During T1

Memory RD=1 which means memory unit will get its data out as AR register indicates.

S2S1S0=111 which means what is read from memory goes to the bus

IR register LD = 1 which means what on the bus will go into IR register .

PC register INR = 1 which means increment PC register.



Determine the Type of Instruction

During T3 the type of instruction will be decoded in steps.

It shows how control determines the instruction type after the decoding.

If D7=1 then the instruction must be register reference or Input-Output instruction

D7.I' = Register reference Instruction

D7.I = IO Instruction

If D7=0 then it may be 000 to 110 values = D0 to D6 which specifies memory reference instructions.

If I=1 then it will be indirect memory reference instruction otherwise if I=0 then it will be direct memory reference instruction.

D7'.I' = memory reference direct addressing mode

D7'.I = memory reference Indirect addressing mode

AR ← M[AR]

During T3 one of 4 different paths

D7'IT3 = AR □M[AR]

D7'I'T3 = Nothing

D7I'T3 = execute register reference instruction

D7IT3 = execute input-output instruction

In timing T4 memory reference instruction will execute

SC is either incremented to enable computer to go to next timing sequence, or set to zero to indicate the termination of instruction execution and start new instruction fetch cycle.

Register Reference Instructions

Register reference instruction will be recognized by control unit if

D7= 1 and I = 0

Register Ref. Instr. is specified in b0~ b11of IR,

Bi= IR(i) , i=0,1,2,...,11 is the bit which indicate each instruction.

Its condition will be summarized by **r = D7.I'.T3**

One in each bit from 0 to 11 specifies a different register reference instruction.

Execution of register reference starts with timing signal T3 and completed here.

Also Sc is cleared to indicate the end of execution and return to fetch new instruction with T0.

Register Reference Instructions :

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

the first seven instructions will be carried on accumulator or carry bit, E bit.

the next 4 skip instructions will add one to PC register only if their condition is met. Those conditions will be

Accumulator is positive, A15=0

Accumulator is negative, A15=1

Accumulator is zero, A0 to A15 are zero.

E is 0, carry bit is zero.

Memory Reference Instructions :

The bits in IR 12 to 14 determines the memory reference instruction type if the bits were from 000 to 110.

Used here a 3 by 8 decoder is used to decode the 3 bits to 8 lines D0 to D7, although D7 not used here.

The effective address of the instruction is in AR and was placed there during timing signal T2 when I = 0, or during timing signal T3when I = 1.

Data from memory must be accessed to processor during T4.

The execution of MR instruction starts with T4. The complete execution of this type of instructions will require a sequence of micro-operations during T4 and T5 and likely T6.

List of all memory reference Instructions:

AND to AC

D_0T_4: DR ← M[AR] Read operand

D_0T_5: AC ← AC ∧ DR, SC ← 0 AND with AC

ADD to AC

D_1T_4: DR ← M[AR] Read operand

D_1T_5: AC ← AC + DR, E ← C_{out}, SC ← 0 Add to AC and store carry in

LDA: Load to AC

D_2T_4: DR ← M[AR]

D_2T_5: AC ← DR, SC ← 0

STA: Store AC

D_3T_4: M[AR] ← AC, SC ← 0

BUN: Branch Unconditionally

D_4T_4: PC ← AR, SC ← 0

BSA: Branch and Save Return Address

M[AR] ← PC, PC ← AR + 1

BSA:

D_5T_4: M[AR] ← PC, AR ← AR + 1

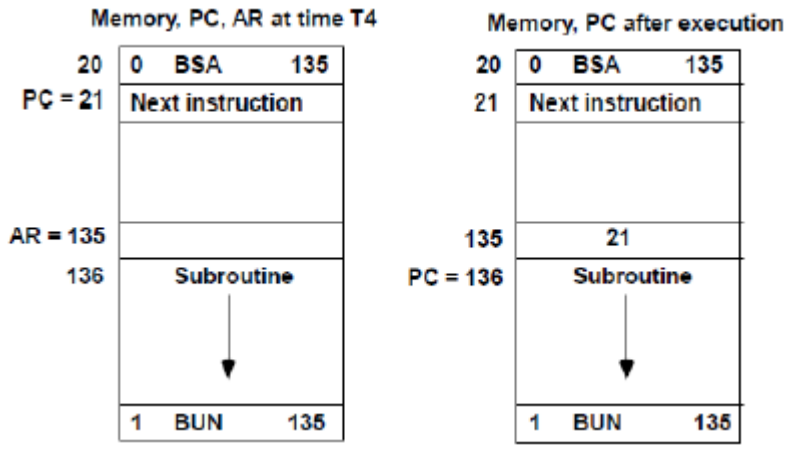
D_5T_5: PC ← AR, SC ← 0

ISZ: Increment and Skip-if-Zero

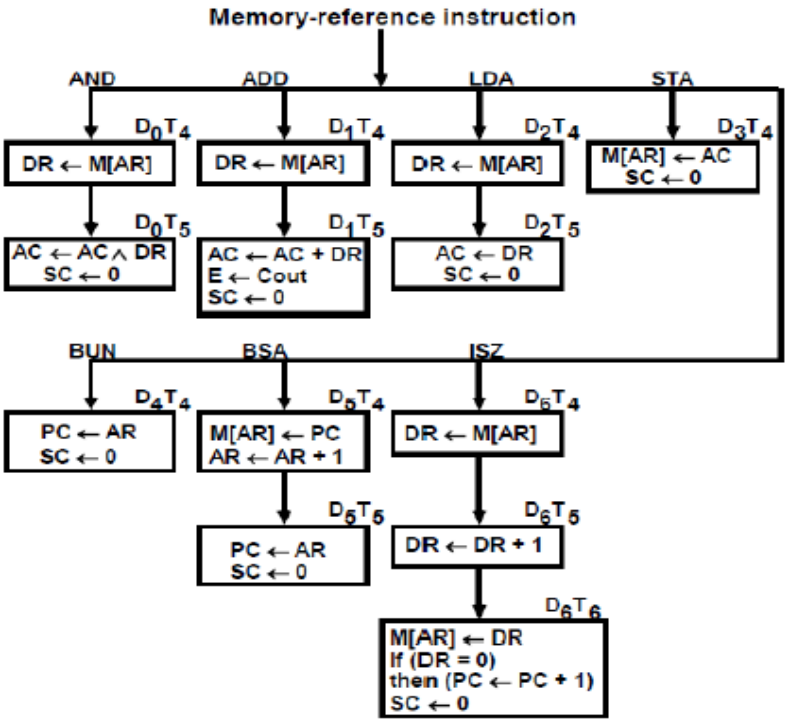
D_6T_4: DR ← M[AR]

D_6T_5: DR ← DR + 1

D_6T_4: M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0



Control flow chart for memory reference instructions:



Input Output & Interrupts

- Computer must communicate with external device to receive and send data with it.
- Instructions and data must come to computer from external input device.

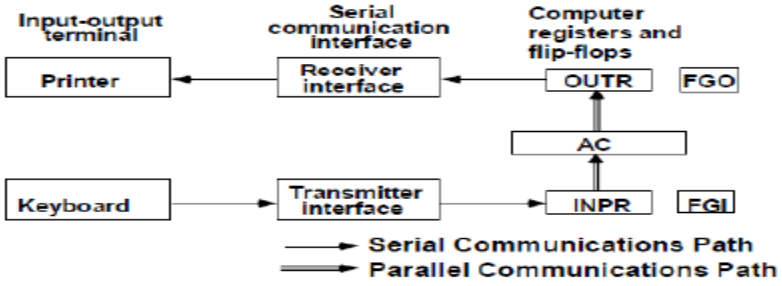
Computational result must be transmitted to user through an output device.

Input Output Configuration

The terminal sends and receives 8 bit data converted to serial information and receives serial information and convert it back to parallel 8 bits.

The serial info from the keyboard is received serially and shifted into INPR.

The serial info for the printer is stored in the OUTR and converted to serial and sends to the printer.



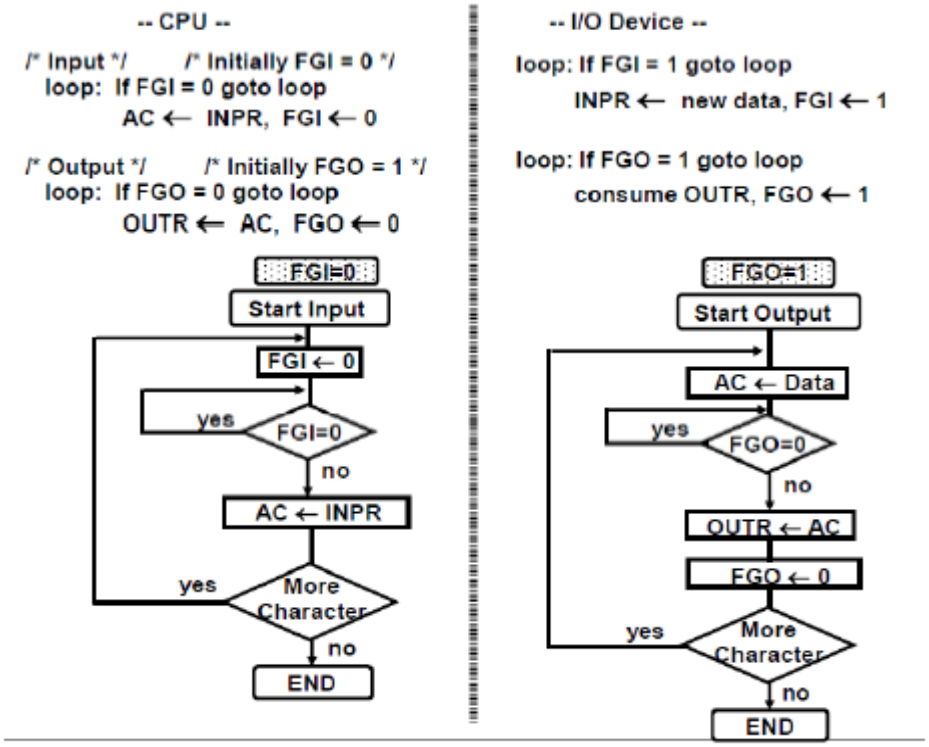
INPR	Input register - 8 bits
OUTR	Output register - 8 bits
FGI	Input flag - 1 bit
FGO	Output flag - 1 bit
IEN	Interrupt enable - 1 bit

The 1 bit FGI is a control flip flop that sets to 1 when new data is available in input device and cleared to m0 (by processor) when computer receives it (needed to synchronize time difference between processor and input device).

When key is pressed in keyboard its code is shifted to serial and shifted to INPR and FGI is set (by the device). This insures that data in INPR will be untouched by another key pressed till it's cleared by the processor.

First the FGO is set to 1 (usually by the device) and processor scans that flag. If FGO is 1 then it will transfer AC to OUTR and clears FGO to 0. The output device accepts data in OUTR and sets FGO to 1 indicating it ready for another transfer.

the program controlled data transfer using flags and INPR and OUTR.



Input Output Instructions

Needed to transfer to and from AC register and output device.

Input Output instructions have op-code of 1111. Recognized when D7=1 and I=1.

how different IO instructions will be excuted when control signal p=D7.I.T3 occurs

$D_7IT_3 = p$
$IR(i) = B_i$, i = 6, ..., 11

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

INP instruction transfer data from INPR register to AC0 to AC7 and clears FGI=0

OUT instruction transfers AC0 to AC7 to OUTR and clears FGO=0

The next 2 instructions scans flags and skip next instruction if flag=1 (usually designed for branching to different locations in program based on value of FGI or FGI flags).

The last 2 instructions sets and clear Interrupt Enable flip flop (see next section).

Interrupt Initiated IO and Interrupt Cycle

Program controlled transfer described earlier keeps scanning flag bits and when set it initiate data transfer.

Inefficient since keeps processor doing nothing except scanning IO flags. Also.

For slow device transfer it can be considered wasting a lot of time (different data transfer rate between IO and processor).

The solution will be the device can interrupt and tell processor when it wants to be served.

The I/O interface, instead of the CPU, monitors the I/O device.

When the interface founds that the I/O device is ready for data transfer, it generates an interrupt request to the CPU. CPU does not check the flags except when it is interrupted by the device.

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.

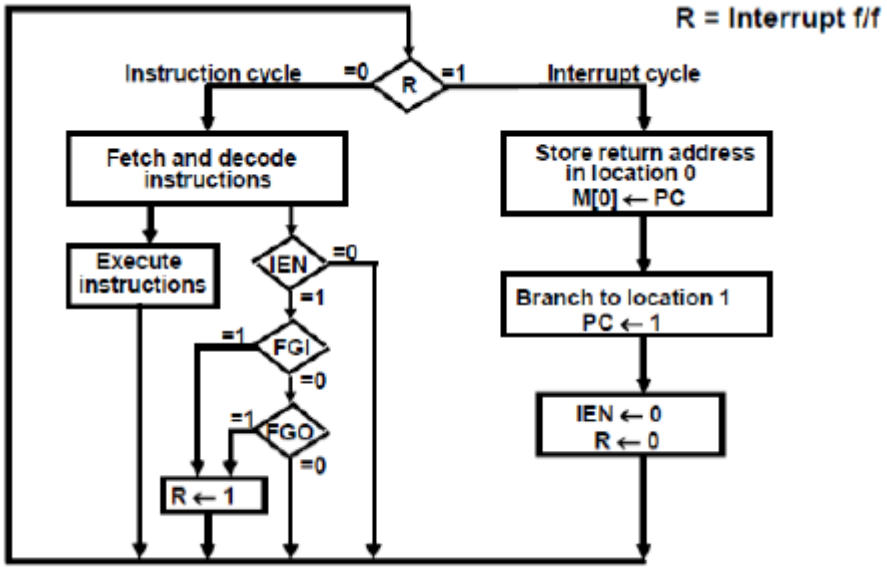
IEN (Interrupt-enable flip-flop) will be introduced here to help in allowing or not allowing devices to interrupt processor through FGI and FGO flags.

Can be set and cleared by instructions like ION and IOF.

When cleared, the computer cannot be interrupted.

□

Flowchart of interrupt cycle.



During execute phase of current instruction cycle, IEN is checked.

If IEN=0 means that user switched off the interrupt so it finishes its current instruction cycle

But if IEN=1 the it will check FGI and FGO; if none of them is set then it will continue normally with current instruction

But if one of the flags is set then flip flop R=1 is set indicating next cycle will be an interrupt cycle.

Interrupt Cycle

The interrupt cycle is a HW implementation of a branch and save return address operation.

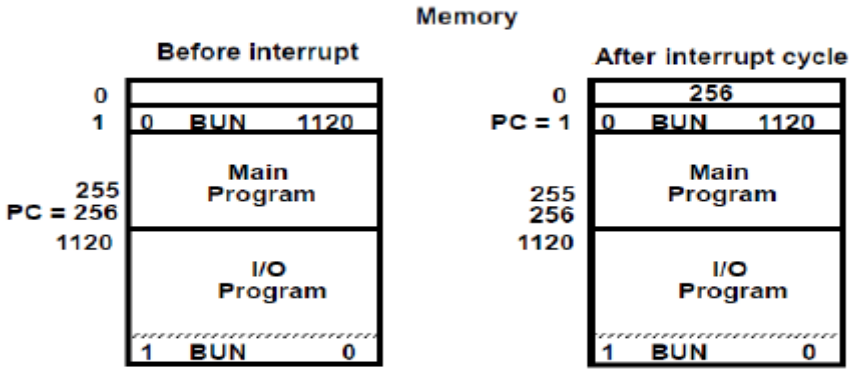
The return address is stored in PC and it will be stored in location 0 of memory (RAM).

PC will be updated to 1 which means it will execute the instruction held at address = 1 and

Clears both IEN=0 and R=0.

At end of interrupt routine BUN to 0 is inserted with I=1. Means go back to where it was interrupted from by "indirect BUN 0".

An example of interrupt cycle



Register transfer statements involved in an interrupt cycle.

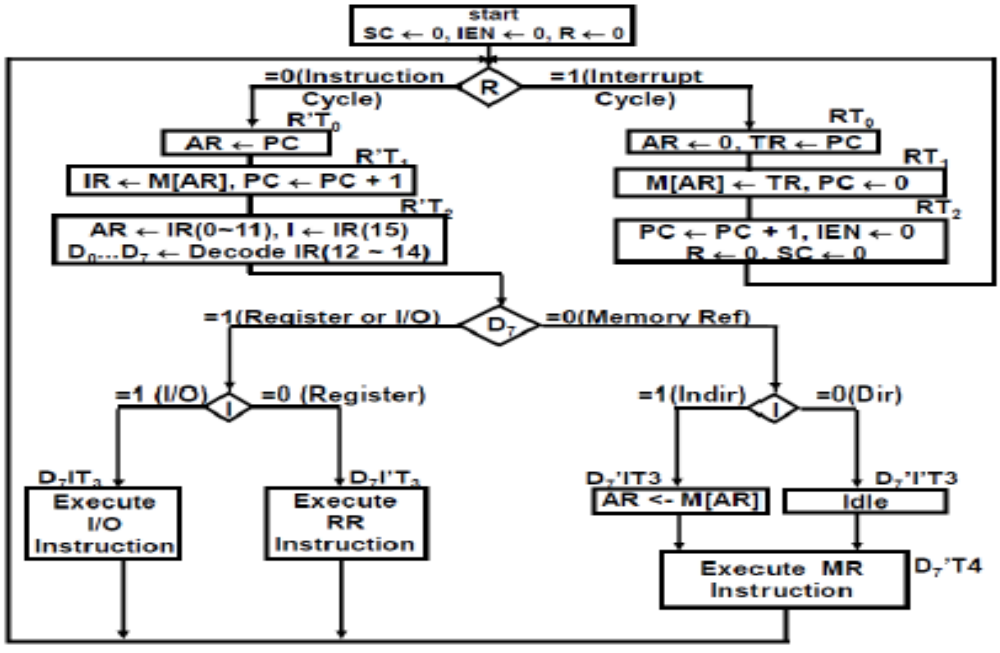
Interrupt cycle is initiated after the last execute phase if R=1. This only can happen in any clock transition except when in times T0, T1, or T2.

The condition set of R flip flop will be

$$T_0'T_1'T_2'(IEN)(\overline{FGI} + \overline{FGO}): \quad R \leftarrow 1$$

Fetch and decode phase will be modified in order to service the interrupt operations involved. Need modified T0, T1, and T2 phases only for interrupt cycle

$$RT_0: \quad AR \leftarrow 0,\ TR \leftarrow PC$$
$$RT_1: \quad M[AR] \leftarrow TR,\ PC \leftarrow 0$$
$$RT_2: \quad PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$$



Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D0, \ldots, D7 \leftarrow$ Decode $IR(12 - 14)$, $AR \leftarrow IR(0 - 11)$, $I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt		
	$T_0'T_1'T_2'(IEN)(FGI + FGO)$:	$R \leftarrow 1$
	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:	$M[AR] \leftarrow DR$, if(DR=0) then $(PC \leftarrow PC + 1)$, $SC \leftarrow 0$
Register-Reference		
	$D_7I'T_3 = r$	(Common to all register-reference instr)
	$IR(i) = B_i$	(i = 0,1,2, ..., 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	$D_7IT_3 = p$	(Common to all input-output instructions)
	$IR(i) = B_i$	(i = 6,7,8,9,10,11)
	p:	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0\text{-}7) \leftarrow INPR$, $FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{-}7)$, $FGO \leftarrow 0$
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:	$IEN \leftarrow 0$