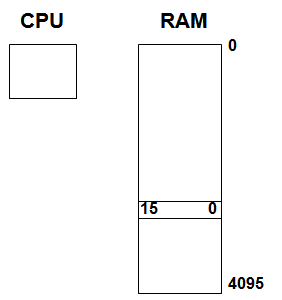
**Instruction Codes**

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.

The Basic Computer has two components, a processor and memory. The memory has 4096 words in it

4096 = 212, so it takes 12 bits to select a word in memory.Each word is 16 bits long.

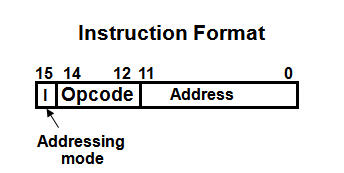


Program-A sequence of (machine) instructions

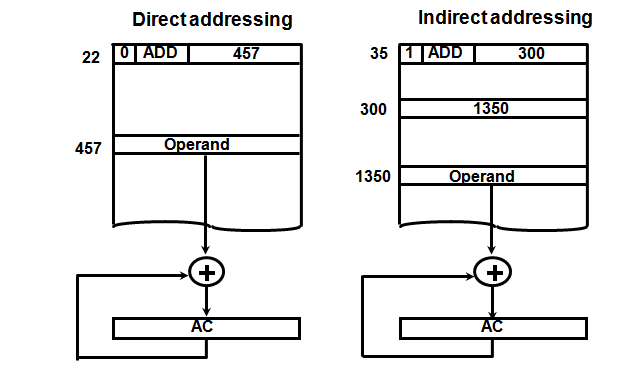
(Machine) Instruction-A group of bits that tell the computer to perform a specific operation (a sequence of micro-operation) .The instructions of a program, along with any needed data are stored in memory

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.

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 (= 212) words, we needs 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.



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.



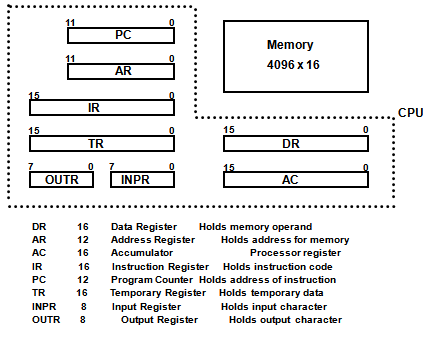
Effective Address (EA)-The address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch-type instruction.

**PROCESSOR REGISTERS**

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).

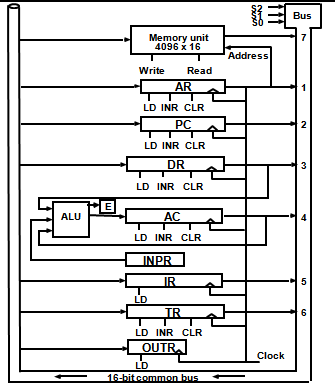
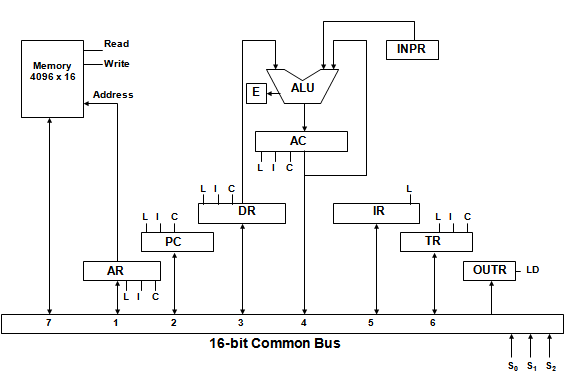
The significance of a general purpose register is that it can be referred to in instructions e.g. load AC with the contents of a specific memory location; store the contents of AC into a specified memory location

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.

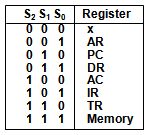


**COMMON BUS SYSTEM**

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

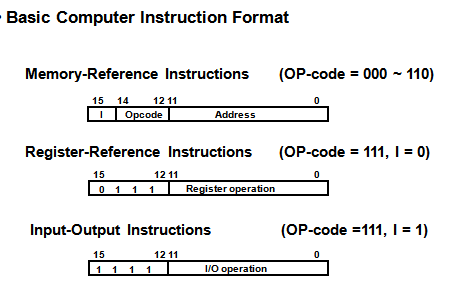


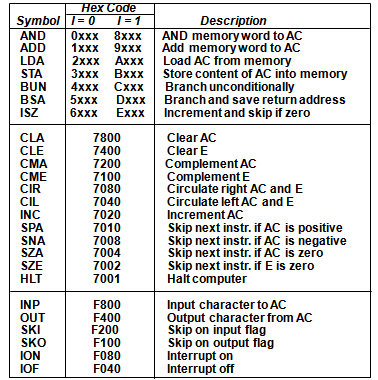
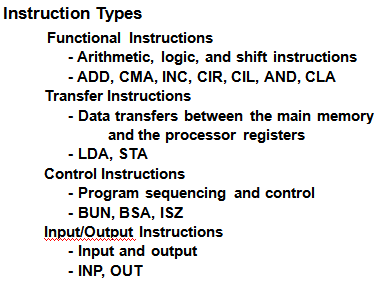
Three control lines, S2, S1, and S0 control which register the bus selects as its input



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.

**BASIC COMPUTER INSTRUCTIONS**





**CONTROL UNIT**

Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them.

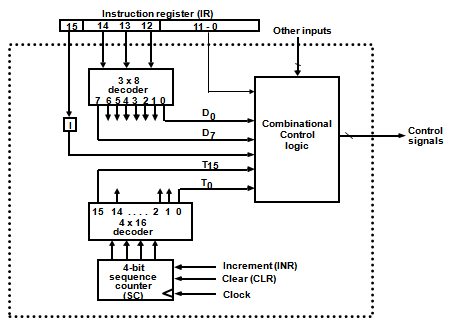
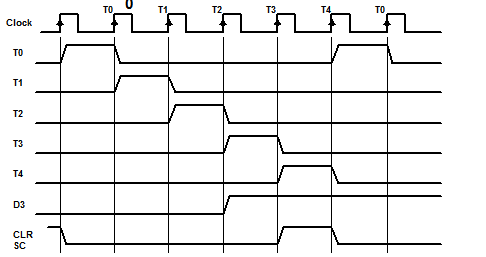
Control units are implemented in one of two ways

* 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.



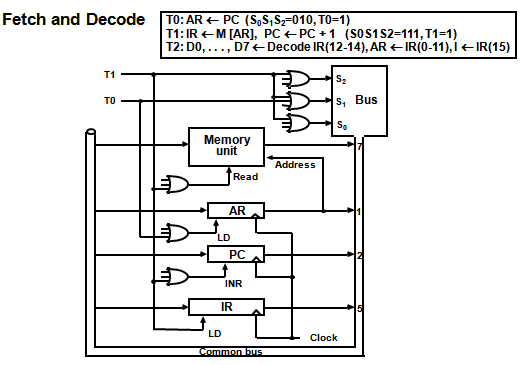
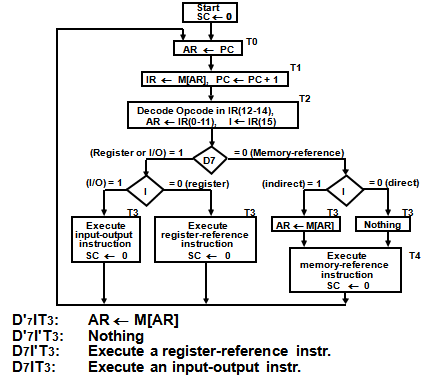
Generated by 4-bit sequence counter and 416 decoder and The SC can be incremented or cleared.

Example: T0, T1, T2, T3, T4, T0, T1, . . . Assume: At time T4, SC is cleared to 0 if decoder output D3 is active.

**INSTRUCTION CYCLE**

* In Basic Computer, a machine instruction is executed in the following cycle:
  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
  5. After an instruction is executed, the cycle starts again at step 1, for the next instruction

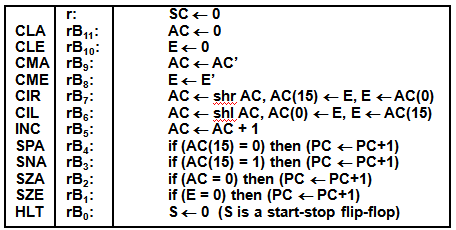
Please note that every different processor has its own (different) instruction cycle .



**REGISTER REFERENCE INSTRUCTIONS**

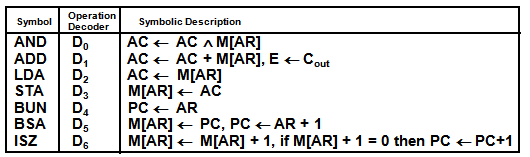
Register Reference Instructions are identified when D7 = 1, I = 0, Register Ref. Instr. is specified in b0 ~ b11 of IR, Execution starts with timing signal T3and r = D7 I′T3 => Register Reference Instruction

Bi = IR(i) , i=0,1,2,...,11

****

**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 and The execution of MR instruction starts with T4

****

AND to AC

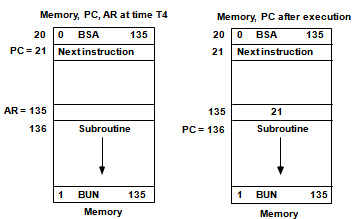
D0T4: DR ← M[AR] Read operand

D0T5: AC ← AC ∧ DR, SC ← 0 AND with AC

ADD to AC

D1T4: DR ← M[AR] Read operand

D1T5: AC ← AC + DR, E ← Cout, SC ← 0 Add to AC and store carry in E

LDA: Load to AC

D2T4: DR ← M[AR]

D2T5: AC ← DR, SC ← 0

STA: Store AC

D3T4: M[AR] ← AC, SC ← 0

BUN: Branch Unconditionally

D4T4: PC ← AR, SC ← 0

BSA: Branch and Save Return Address

M[AR] ← PC, PC ← AR + 1

BSA:

D5T4: M[AR] ← PC, AR ← AR + 1

D5T5: PC ← AR, SC ← 0

ISZ: Increment and Skip-if-Zero

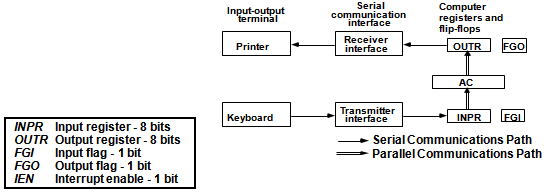
D6T4: DR ← M[AR]

D6T5: DR ← DR + 1

D6T4: M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0

**INPUT-OUTPUT AND INTERRUPT**

Input-Output Configuration



The terminal sends and receives serial information

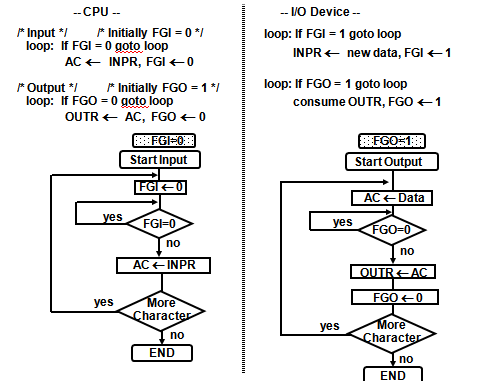
- The serial info. from the keyboard is shifted into INPR

- 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

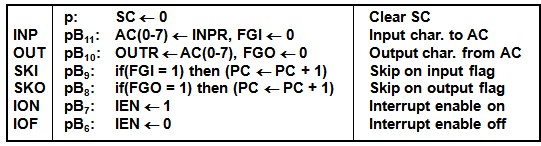
PROGRAM CONTROLLED DATA TRANSFER



**INPUT-OUTPUT INSTRUCTIONS**

D7IT3 = p

IR(i) = Bi, i = 6, …, 11



**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 founds 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.

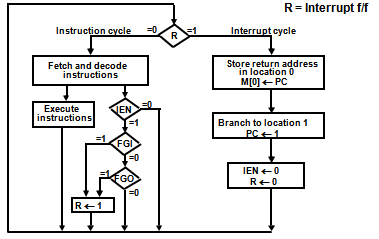
- IEN (Interrupt-enable flip-flop)-can be set and cleared by instructions and when cleared, the computer cannot be interrupted.

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

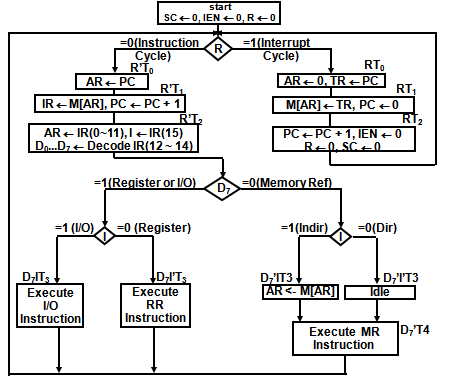
- At the beginning of the next instruction cycle, the instruction that is read from memory is in address 1.

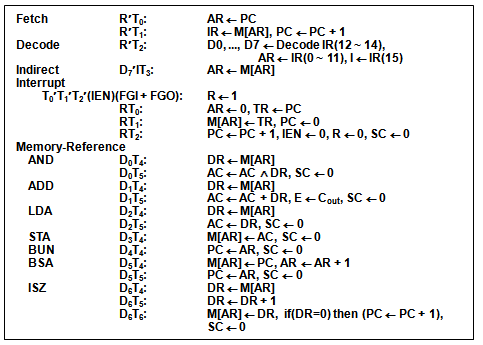
- At memory address 1, the programmer must store a branch instruction that sends the control to an interrupt service routine.

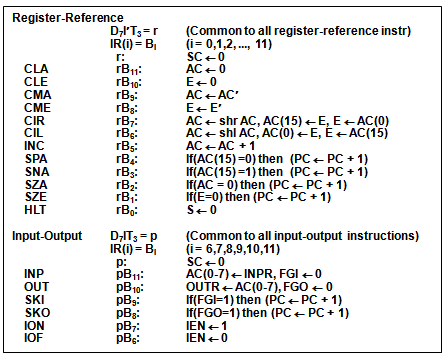
- The instruction that returns the control to the original program is "indirect BUN 0"



**COMPLETE COMPUTER DESCRIPTION**





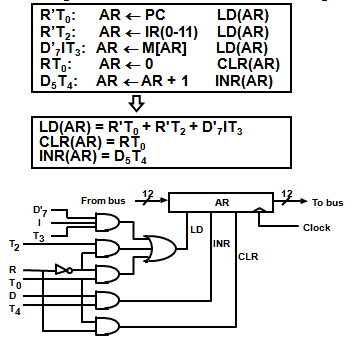


**DESIGN OF BASIC COMPUTER**

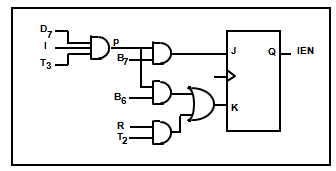
Hardware Components-A memory unit: 4096 x 16,Registers: AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC,Flip-Flops(Status): I, S, E, R, IEN, FGI, and FGO ,Decoders: a 3x8 Opcode decoder,a 4x16 timing decoder,Common bus: 16 bits,Control logic gates:Adder and Logic circuit: Connected to AC.

Control Logic Gates-Input Controls of the nine registers;- Read and Write Controls of memory; Set, Clear, or Complement Controls of the flip-flops; S2, S1, S0 Controls to select a register for the bus; AC, and Adder and Logic circuit.

Scan all of the register transfer statements that change the content of AR:



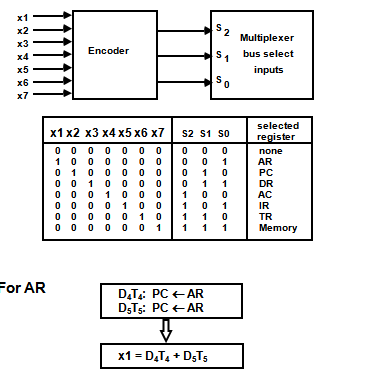
pB7: IEN <- 1 (I/O Instruction)

pB6: IEN <- 0 (I/O Instruction)

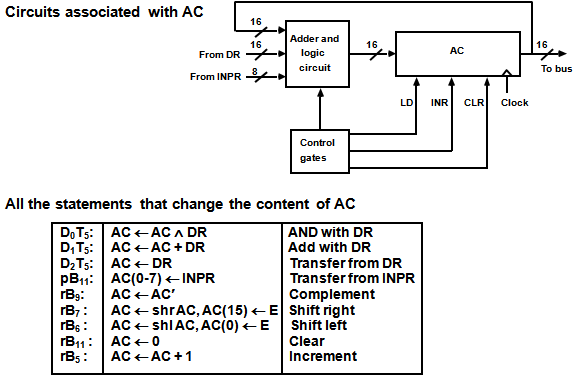
RT2: IEN <- 0 (Interrupt)

p = D7IT3 (Input/Output Instruction)

Common bus



ACCUMULATOR LOGIC



ALU

