

Topics included:

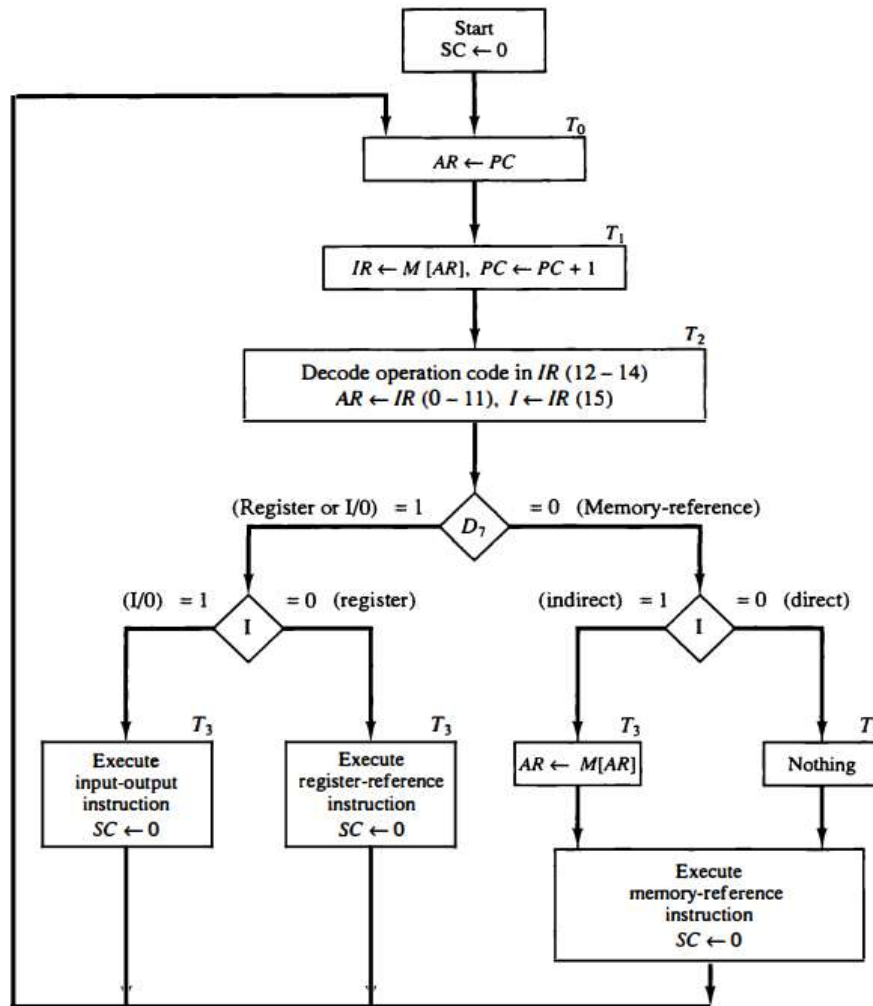
1. Instruction cycle
2. Input-Output and Interrupt
3. Complete Computer Description,

Instruction cycle

- A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of subcycles or phases. In the basic computer each instruction cycle consists of the following 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.
- Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.

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.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statements.
- $T_0: AR \leftarrow PC$
- $T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$
- $T_2: D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), 1 \leftarrow IR(I_5)$



Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T_0 .

The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1 . At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.

At time T_2 , the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR. Note that SC is incremented after each clock pulse to produce the sequence T_0 , T_1 , and T_2 .

Determine the Type of Instruction

- The timing signal that is active after the decoding is T3
- During time T₃, the control unit determines the type of instruction that was just read from memory.

The three possible instruction types available in the basic computer are specified. Decoder output D7 is equal to 1 if the operation code is equal to binary 111. we determine that if D7 = 1, the instruction must be a register-reference or input-output type. If D7 = 0, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction. Control then inspects the value of the first bit of the instruction, which is now available in flip-flop I. If D7 = 0 and I = 1, we have a memory reference instruction with an indirect address. It is then necessary to read the effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement.

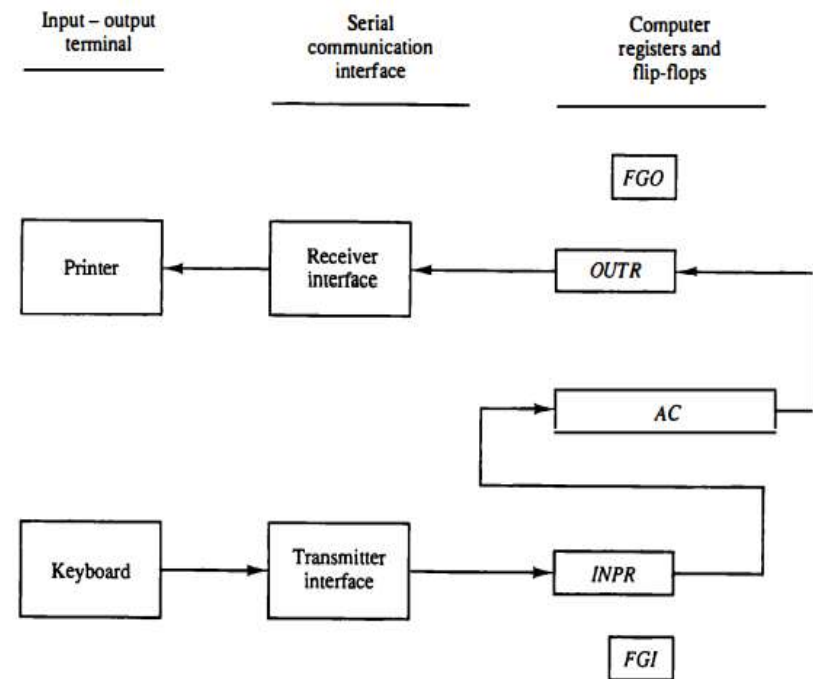
Input-Output and Interrupt

- A computer can serve no useful purpose unless it communicates with the external environment.
- Instructions and data stored in memory must come from some input device.
- Computational results must be transmitted to the user through some output device.
- Commercial computers include many types of input and output devices. To demonstrate the most basic requirements for input and output communication, we will use as an illustration a terminal unit with a keyboard and printer.

- The terminal sends and receives serial information.
- Each quantity of information has eight bits of an alphanumeric code. The serial information from the keyboard is shifted into the input register INPR.
- The serial information for the printer is stored in the output register OUTR.
- These two registers communicate with a communication interface serially and with the AC in parallel.
- The input-output configuration is shown in Figure on the next slide.

Input-Output Configuration

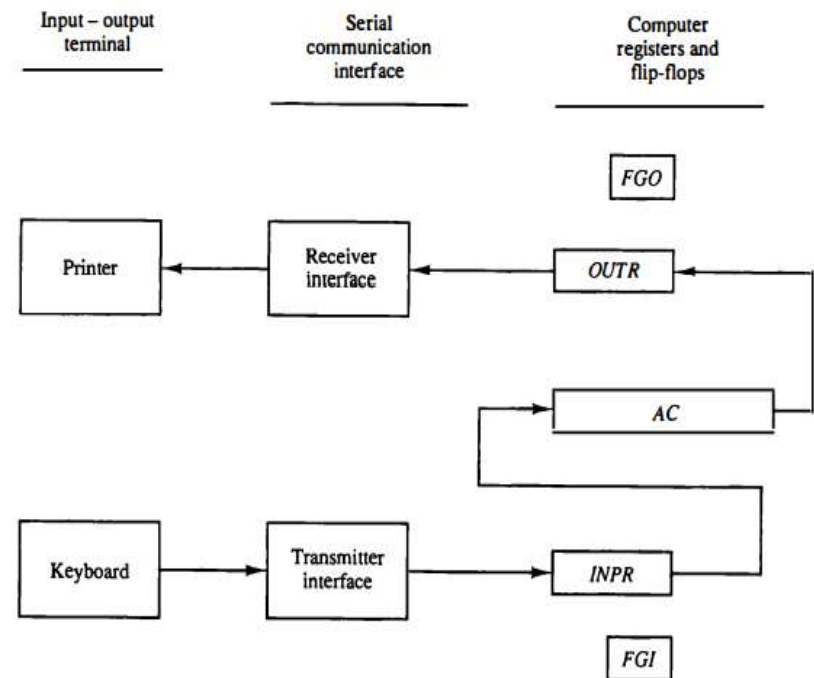
- The input register INPR consists of eight bits and holds an alphanumeric input information. The 1-bit input flag FGI is a control flip-flop. The flag bit is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer.
- The process of information transfer is as follows. Initially, the input flag FGI is cleared to 0. When a key is struck in the keyboard, an 8-bit alphanumeric code is shifted into INPR and the input flag FGI is set to 1. As long as the flag is set, the information in INPR cannot be changed by striking another key. The computer checks the flag bit; if it is 1, the information from INPR is transferred in parallel into AC and FGI is cleared to 0. Once the flag is cleared, new information can be shifted into INPR by striking another key.



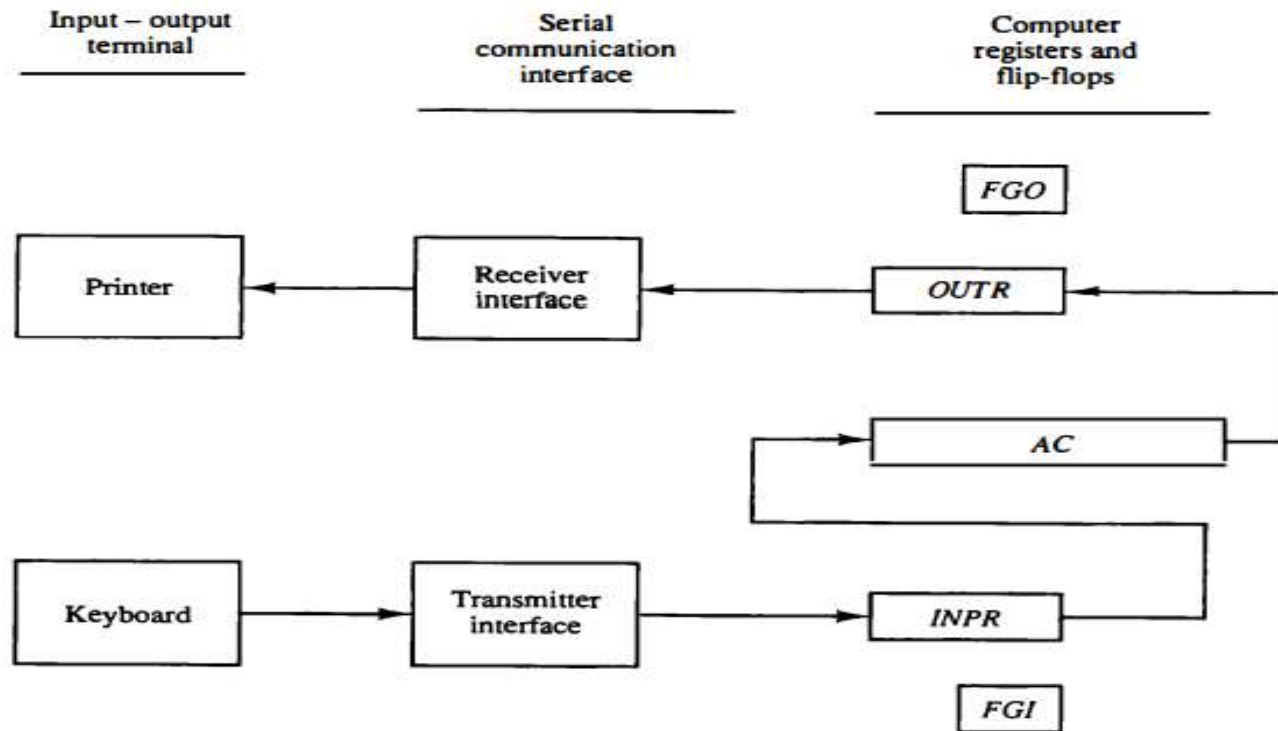
Input-Output Configuration



- The output register OUTF works similarly but the direction of information flow is reversed. Initially, the output flag FGO is set to 1. The computer checks the flag bit; if it is 1, the information from AC is transferred in parallel to OUTF and FGO is cleared to 0. The output device accepts the coded information, prints the corresponding character, and when the operation is completed, it sets FGO to 1. The computer does not load a new character into OUTF when FGO is 0 because this condition indicates that the output device is in the process of printing the character.



Input-Output Configuration block diagram



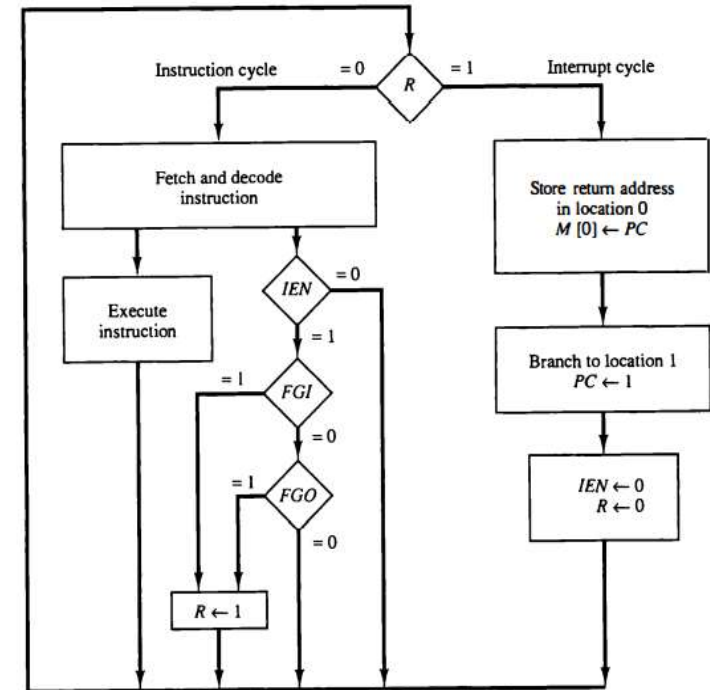
Interrupt

- The computer is wasting time while checking the flag instead of doing some other useful processing task.
- An alternative to the programmed controlled procedure is to let the external device inform the computer when it is ready for the transfer. In the meantime the computer can be busy with other tasks.
- This type of transfer uses the interrupt facility.
- While the computer is running a program, it does not check the flags. However, when a flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that a flag has been set.
- The computer deviates momentarily from what it is doing to take care of the input or output transfer. It then returns to the current program to continue what it was doing before the interrupt

Interrupt Cycle

Instruction cycle(when R=0)

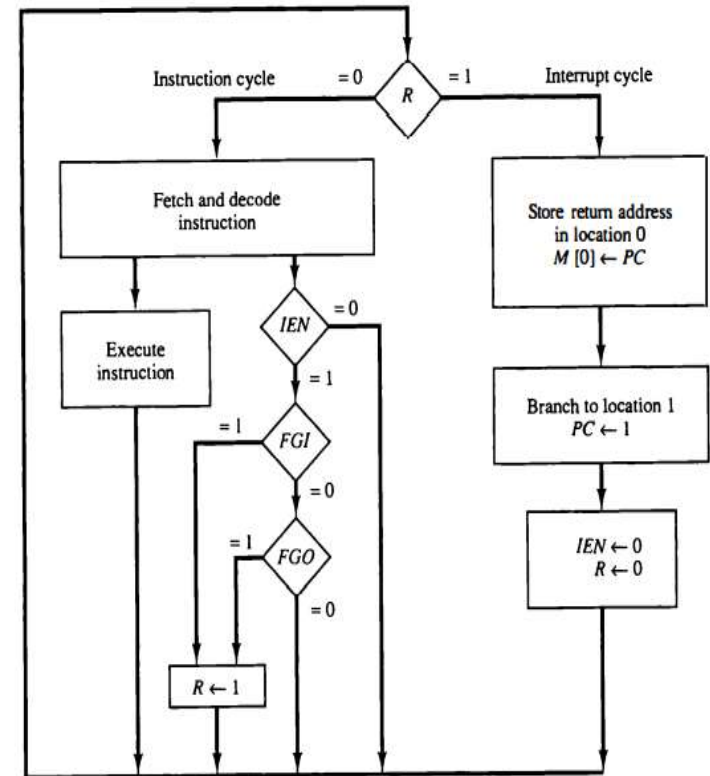
- An interrupt flip-flop R is included in the computer. When $R = 0$, the computer goes through an **instruction cycle**.
- During the execute phase of the instruction cycle IEN is checked by the control. If it is 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle.
- If IEN is 1, control checks the flag bits. If both flags are 0, it indicates that neither the input nor the output registers are ready for transfer of information. In this case, control continues with the next instruction cycle.
- If either flag is set to 1 while $IEN = 1$, flip-flop R is set to 1. At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.



Flowchart for Interrupt Cycle

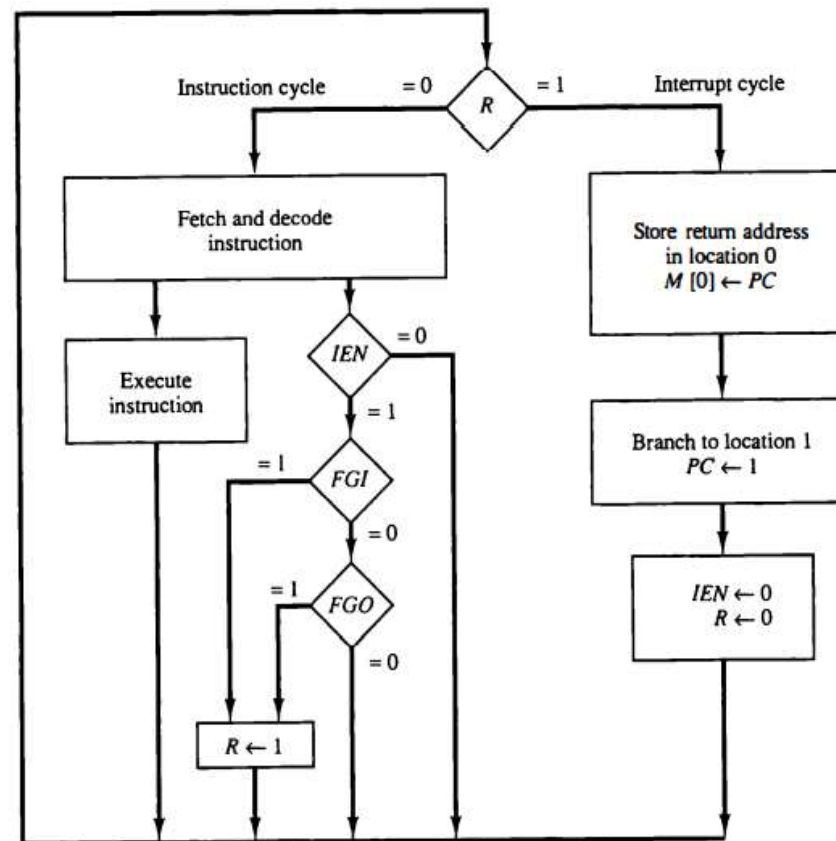
The interrupt cycle(when R=1)

- The interrupt cycle is a hardware implementation of a branch and save return address operation. The return address available in PC is stored in a specific location where it can be found later when the program returns to the instruction at which it was interrupted. This location may be a processor register, a memory stack, or a specific memory location. Here we choose the memory location at address 0 as the place for storing the return address. Control then inserts address 1 into PC and clears IEN and R so that no more interruptions can occur until the interrupt request from the flag has been serviced.



Flowchart for Interrupt Cycle

Flowchart for interrupt cycle



Complete Computer description

