Computer Architecture

Lecture-04

Md. Biplob Hosen

Lecturer, IIT-JU.

Email: biplob.hosen@juniv.edu

Reference

- “Computer Organization and Architecture” by William Stallings; 8th Edition (Chapter-03).
  - Any later edition is fine.

Overview

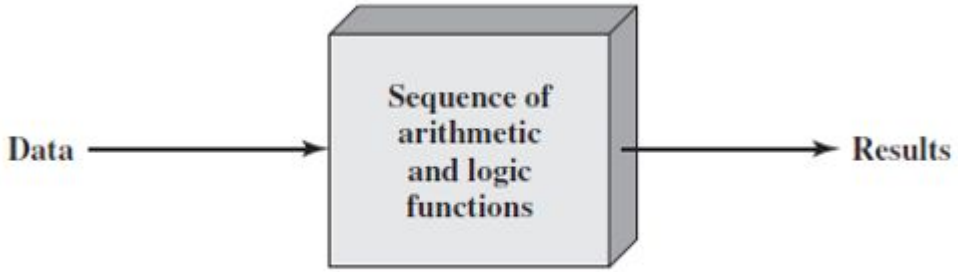
- At a top level, a computer consists of CPU (central processing unit), memory, and I/O components, with one or more modules of each type.
- These components are interconnected in some fashion to achieve the basic function of the computer, which is to execute programs.
- At a top level, we can describe a computer system by:
  - Describing the external behavior of each component—that is, the data and control signals that it exchanges with other components; and
  - Describing the interconnection structure and the controls required to manage the use of the interconnection structure.

Overview

- Virtually all contemporary computer designs are based on concepts developed by John von Neumann.
- Such a design is referred to as the von Neumann architecture and is based on three key concepts:
  - Data and instructions are stored in a single read–write memory.
  - The contents of this memory are addressable by location, without regard to the type of data contained there.
  - Execution normally occurs in a sequential fashion from one instruction to the next.

Types of Programming

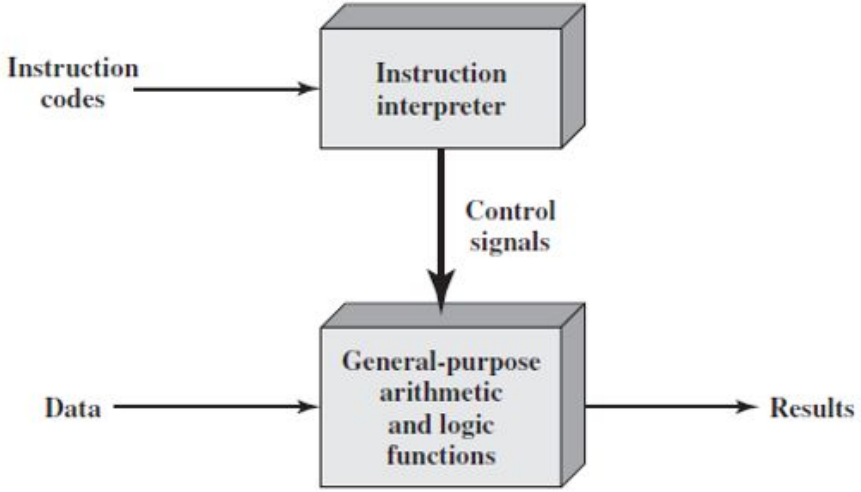
- **Programming in Hardware:** There is a small set of basic logic components that can be combined in various ways to store binary data and to perform arithmetic & logical operations on that data.
- If there is a particular computation to be performed, a configuration of logic components design specially for that computation could be constructed. We can think the process of connecting the various components in the desired configuration as a form of programming. The resulting program is in the form of hardware, and is termed as hardware program.



(a) Programming in hardware

Types of Programming

- **Programming in Software:** Now consider the alternative. Suppose, we construct a general purpose configuration of arithmetic and logic functions. This set of hardware will perform various functions on data depending on control signals applied to the hardware. Now, the system accepts data & control signal, and produce results. Thus, instead of rewiring the hardware for each new program, the programmers now need to supply a new set of control signals.

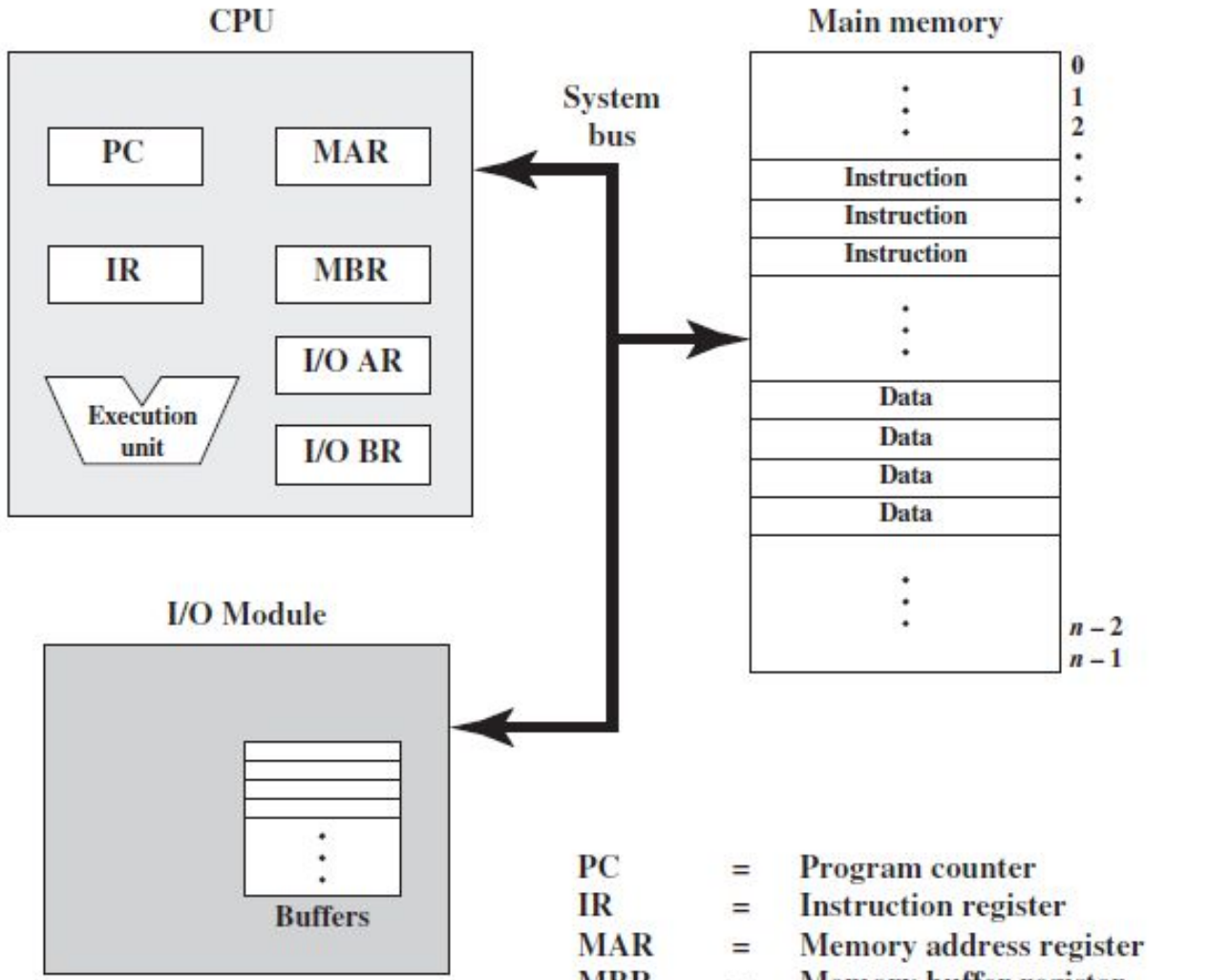


(b) Programming in software

General Purpose Processor

- Figure-b indicates two major components of the system: an instruction interpreter and a module of general-purpose arithmetic and logic functions.
  - These two constitute the CPU.
- Several other components are needed to yield a functioning computer.
- Data and instructions must be put into the system and results must be shown in realizable forms.
  - We need I/O module for that.
- A place is also needed for storing the data and instructions.
  - We need Memory for that.

Continue…


CPU
PC
MAR
IR
MBR
I/O AR
Execution unit
I/O BR
System bus
Main memory
0
1
2
Instruction
Instruction
Instruction
Data
Data
Data
Data
n – 2
n – 1
I/O Module
Buffers
PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

Continue…

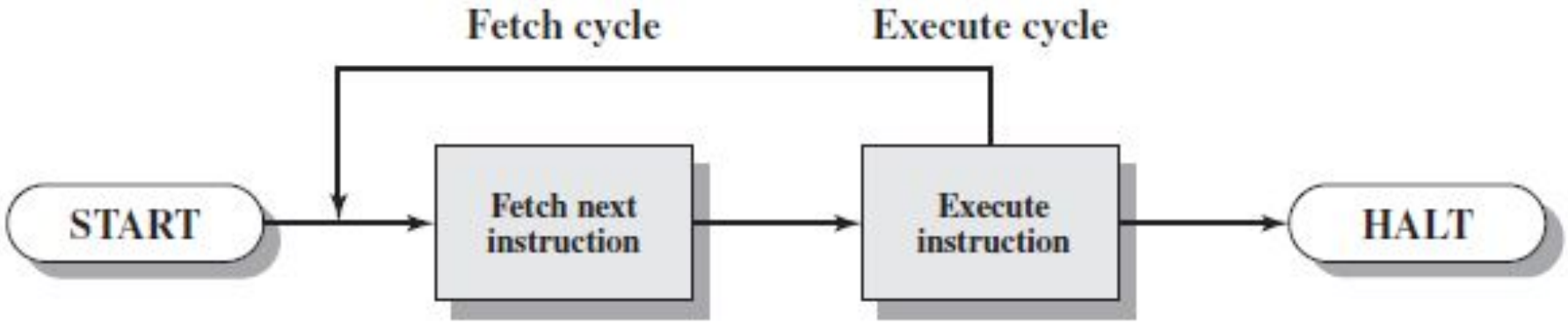
- Figure illustrates these top-level components.
- CPU exchanges data with memory.
- For this purpose, it typically makes use of two internal (to the CPU) registers: a memory address register (MAR), which specifies the address in memory for the next read or write, and a memory buffer register (MBR), which contains the data to be written into memory or receives the data read from memory.
- Similarly, an I/O address register (I/O AR) specifies a particular I/O device.
- An I/O buffer (I/OBR) register is used for the exchange of data between an I/O module and the CPU.

Continue…

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
- In its simplest form, instruction processing consists of two steps:
  - The processor reads (fetches) instructions from memory one at a time and executes each instruction.
  - Program execution consists of repeating the process of instruction fetch and instruction execution.
- The processing required for a single instruction is called an instruction cycle

Instruction Fetch and Execute

- In a typical processor, a register called the program counter (PC) holds the address of the instruction to be fetched next.
- Unless told otherwise, the processor always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence.



Continue…

- The fetched instruction is loaded into a register in the processor known as the instruction register (IR).
- The instruction contains bits that specify the action the processor is to take.
- The processor interprets the instructions and perform the required actions.
- In general, these actions fall into four categories:
  - Processor-memory: Data may be transferred from processor to memory or from memory to processor.
  - Processor-I/O: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
  - Data processing: The processor may perform some arithmetic or logic operation on data.
  - Control: An instruction may specify that the sequence of execution be altered. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction will be from 182. the processor will remember this fact by setting the program counter to 182. thus, on the next fetch cycle, the instruction will be fetched from location 182 rather than 150.

A Hypothetical Processor

- Figure in the next slide illustrates a partial program execution, showing the relevant partitions of memory and processor registers.
- The program fragment shows adding the contents of the memory word at address 940 to the contents of the memory word at address 941.
- All registers are 16 bits. Necessary definitions are give below:

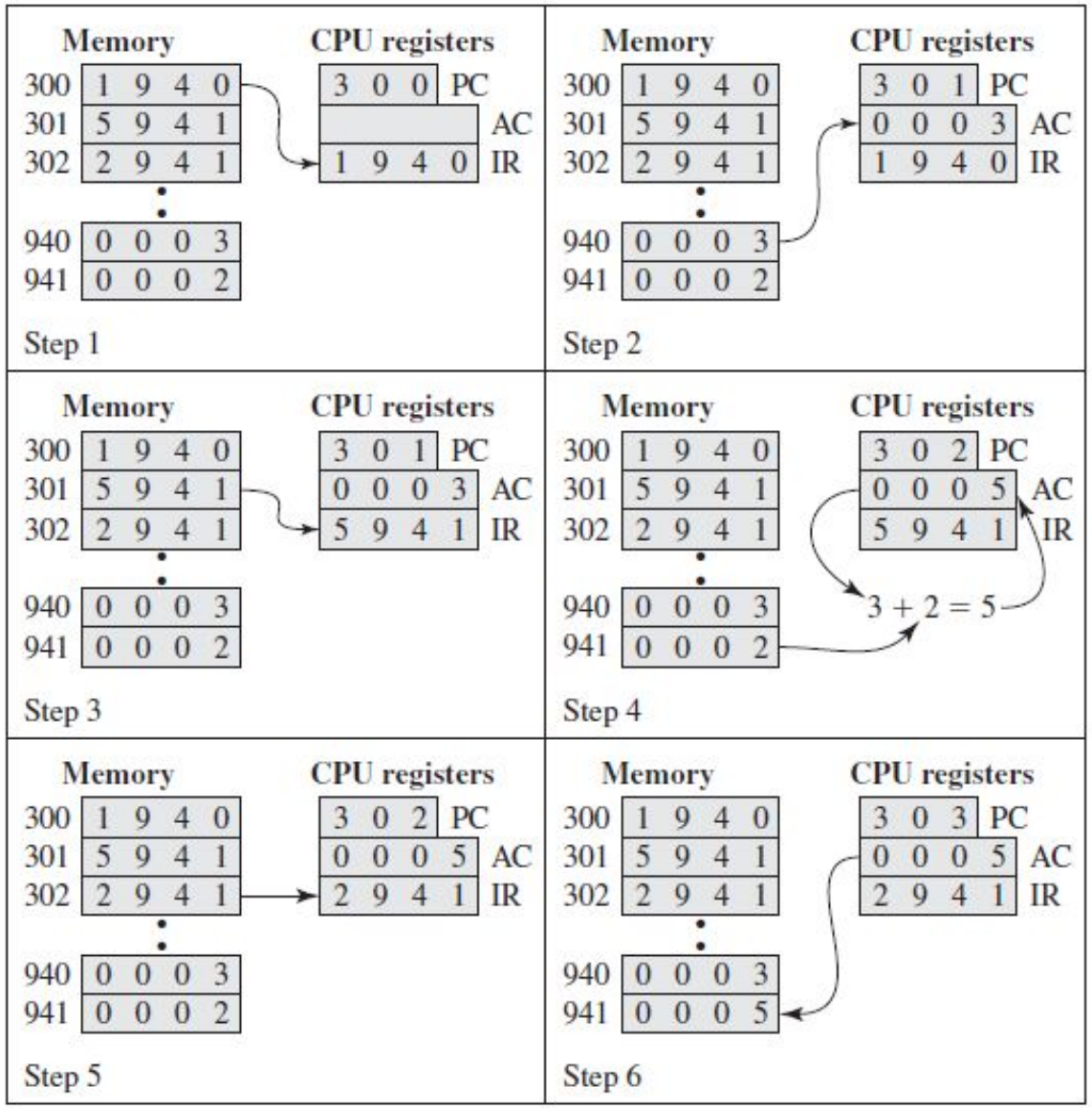
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

Partial list of opcodes

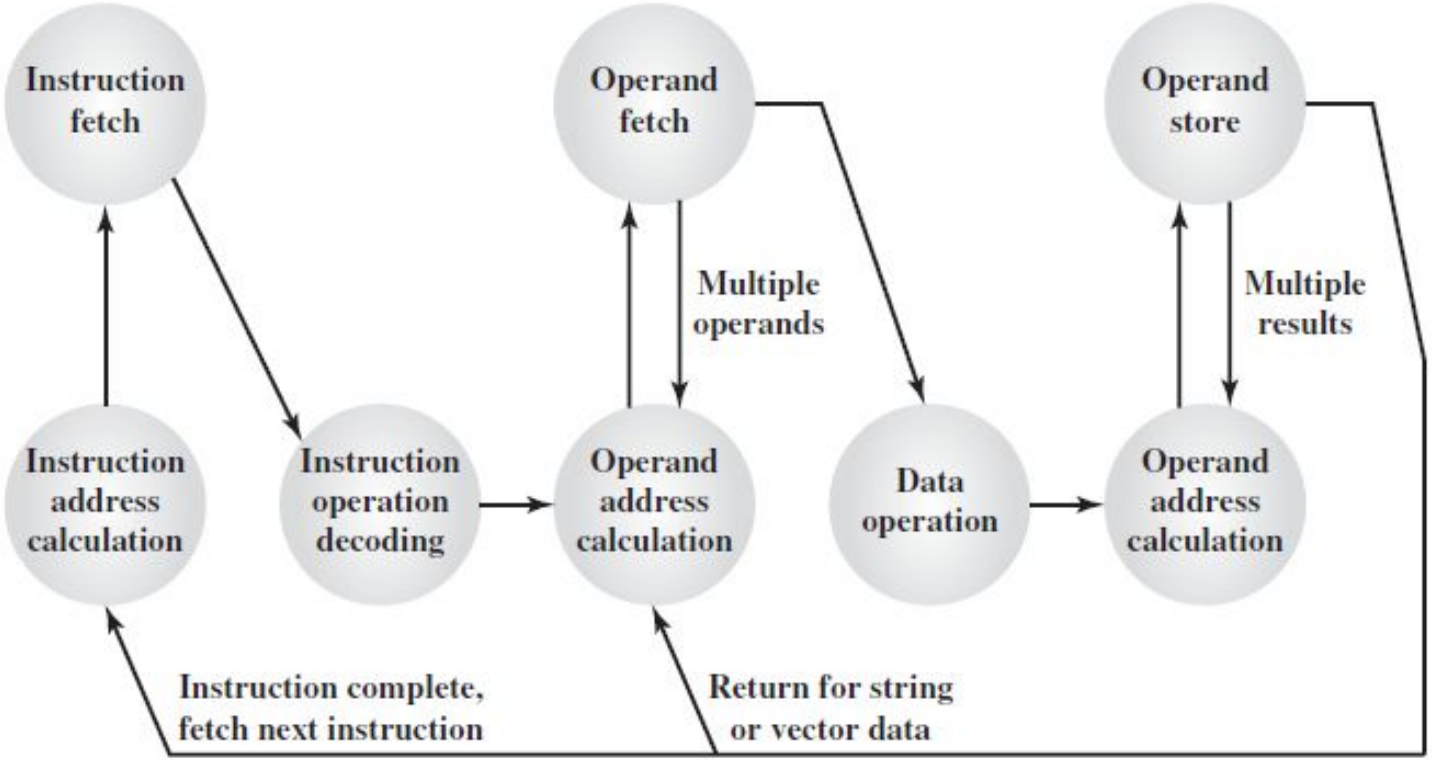
Continue…


Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 0 PC
AC
1 9 4 0 IR
Step 1
3 0 1 PC
0 0 0 3 AC
1 9 4 0 IR
Step 2
5 9 4 1 IR
Step 3
3 0 2 PC
0 0 0 5 AC
3 + 2 = 5
Step 4
2 9 4 1 IR
Step 5
3 0 3 PC
Step 6

Continue…

- The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexa) is loaded into the IR and PC is incremented.
- The first 4bits (first hexa digit) in the IR indicates that the AC is to be loaded. The remaining 12 bits (three hexa digits) specify the address (940) from which data are to be loaded.
- The next instruction (5941) is fetched from location 301 and the PC is incremented.
- The old content of the AC and the content of location 941 are added and the result is stored in the AC.
- The next instruction (2941) is fetched from location 302 and the PC is incremented.
- The content of the AC is stored in location 941.

Instruction Cycle State Diagram



Continue…

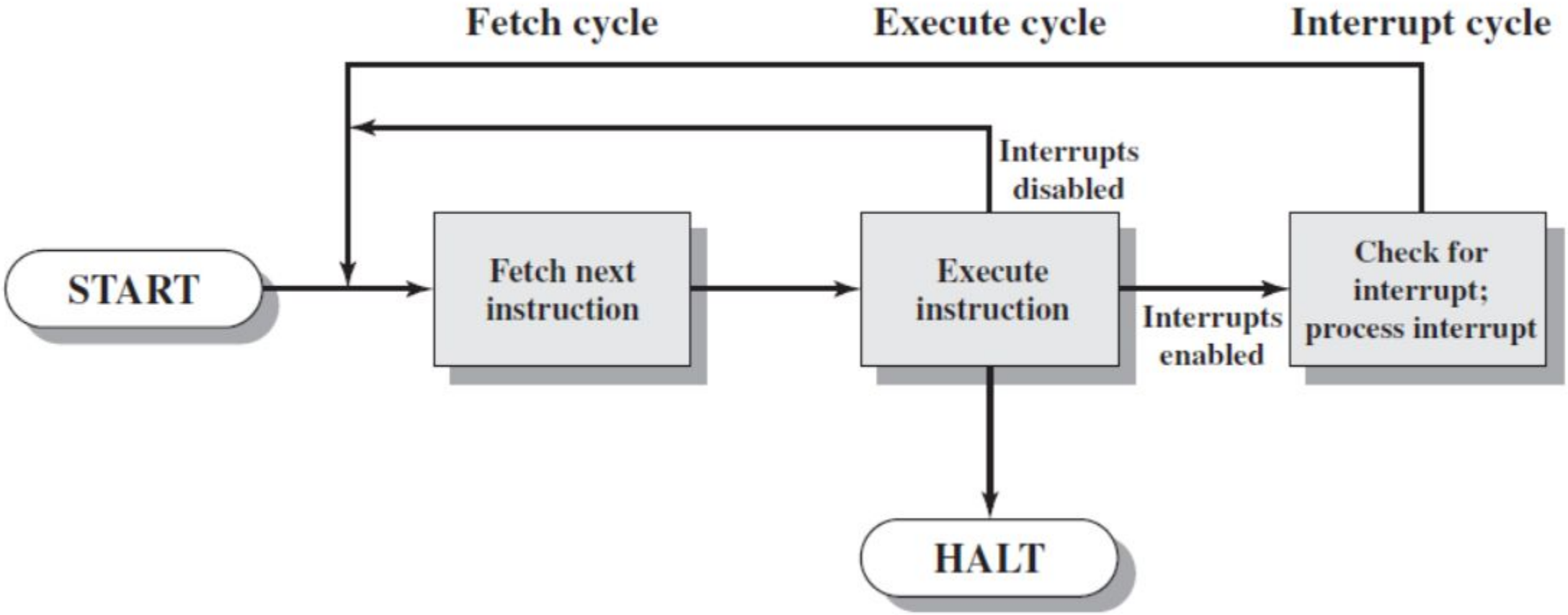
- Instruction address calculation (iac)
- Instruction fetch (if)
- Instruction operation decoding (iod)
- Operand address calculation (oac)
- Operand fetch (of)
- Data operation (do)
- Operand store (os)

Interrupts

- Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.
- Interrupts are provided primarily as a way to improve processing efficiency.
- Classes of Interrupts:

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Instruction Cycle with Interrupts



Thank You