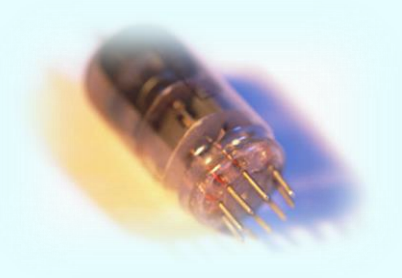
Digital Design and Computer Architecture

Instruction Execution

History of Computers

First Generation: Vacuum Tubes

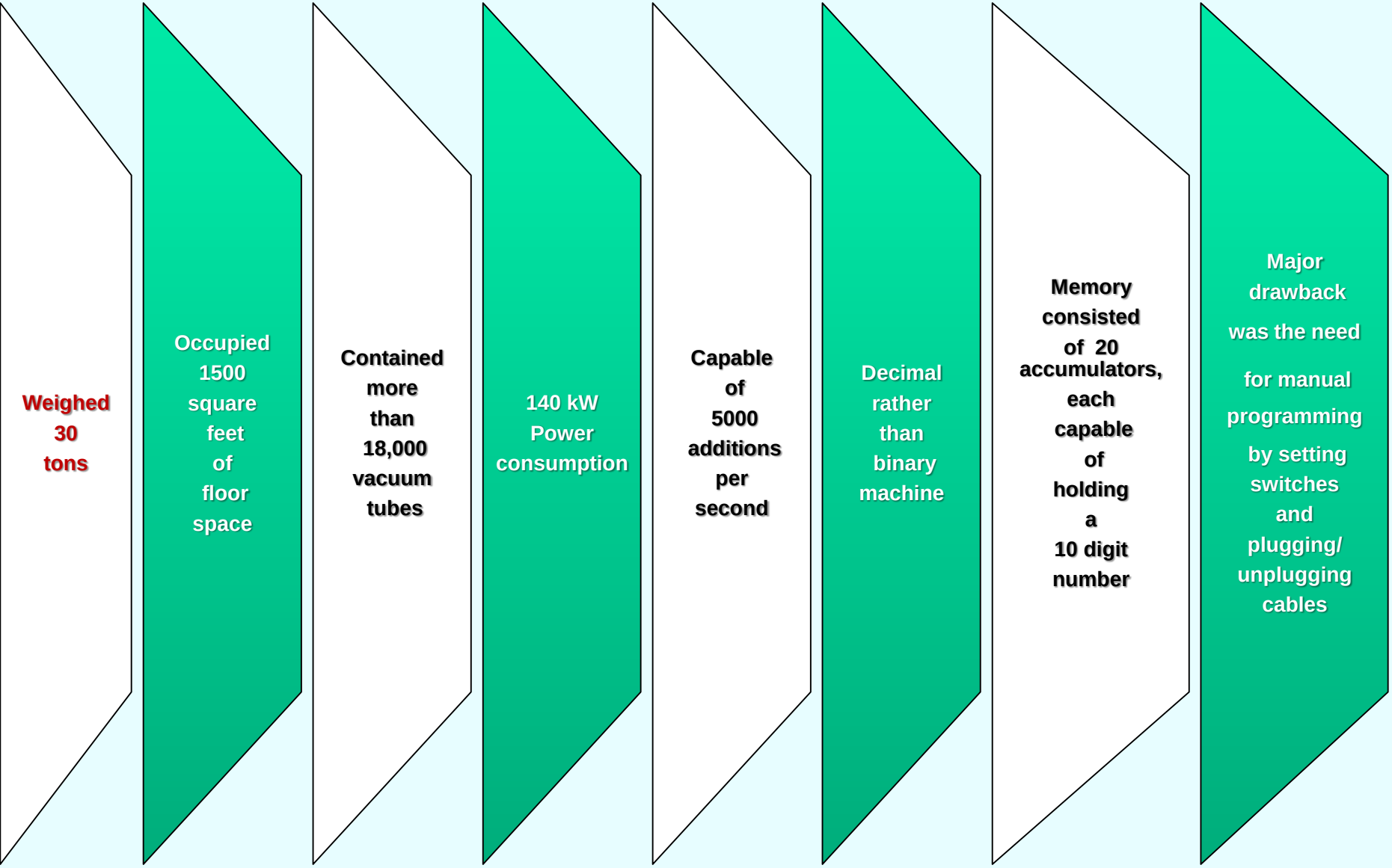


- ENIAC
  - Electronic Numerical Integrator And Computer
- Designed and constructed at the University of Pennsylvania
  - Started in 1943 – completed in 1946
  - By John Mauchly and John Eckert
- World's first general purpose electronic digital computer
  - Army's Ballistics Research Laboratory (BRL) needed a way to supply trajectory tables for new weapons accurately and within a reasonable time frame
  - Was not finished in time to be used in the war effort

– Its first task was to perform a series of calculations that were used to help determine the feasibility of the hydrogen bomb

– Continued to operate under BRL management until 1955 when it was disassembled

ENIAC


Weighed 30 tons
Occupied 1500 square feet of floor space
Contained more than 18,000 vacuum tubes
140 kW Power consumption
Capable of 5000 additions per second
Decimal rather than binary machine
Memory consisted of 20 accumulators, each capable of holding a 10 digit number
Major drawback was the need for manual programming by setting switches and plugging/ unplugging cables

John von Neumann

EDVAC (Electronic Discrete Variable Computer)

- First publication of the idea was in 1945
- Stored program concept
  - Attributed to ENIAC designers, most notably the mathematician John von Neumann
  - Program represented in a form suitable for storing in memory alongside the data

- IAS computer
  - Princeton Institute for Advanced Studies
  - Prototype of all subsequent general-purpose computers
  - Completed in 1952

IAS Memory Formats

- The memory of the IAS consists of 1000 storage locations (called ***words***) of 40 bits each
  - Both data and instructions are stored there
  - Numbers are represented in binary form and each instruction is a binary code

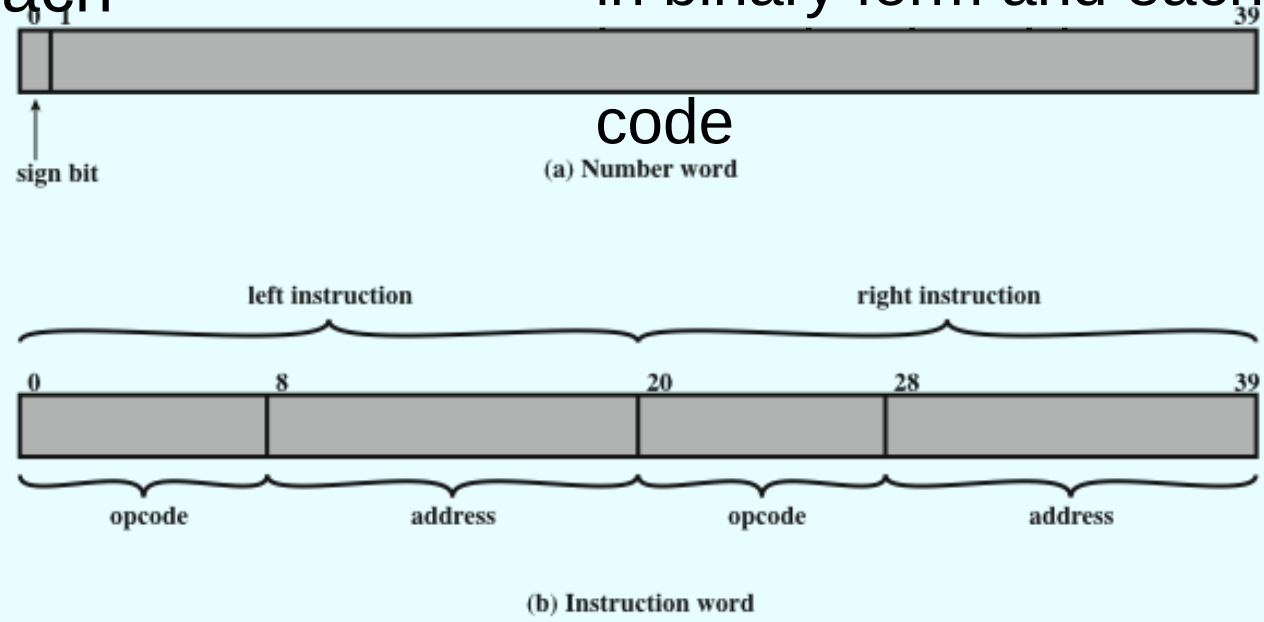


Figure 2.2 IAS Memory Formats

Digital Design and Computer Architecture

- Von Neumann Architecture
  - The 5 component design model
- The Instruction Cycle
  - Basic
  - Exceptions
- Instruction architecture
  - software design
  - hardware circuits

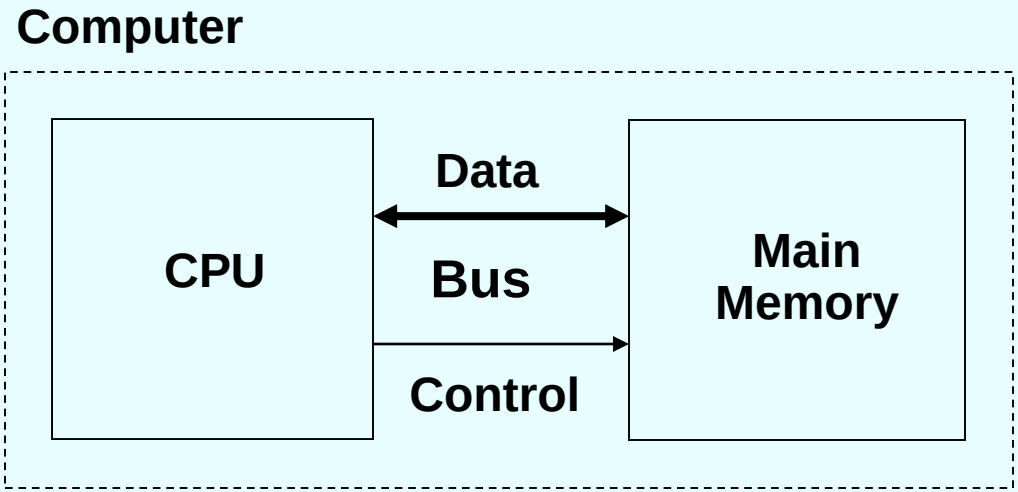
Review Agenda

- Von Neumann Architecture
  - 5 component design of the *stored program digital computer*
  - the *instruction cycle*
    - Basic
    - Exceptions
  - instruction architecture
    - software design
    - hardware circuits

- Digital Design
  - Boolean logic and gates
  - Basic Combinational Circuits
  - Karnaugh maps
  - Advanced Combinational Circuits
  - Sequential Circuits

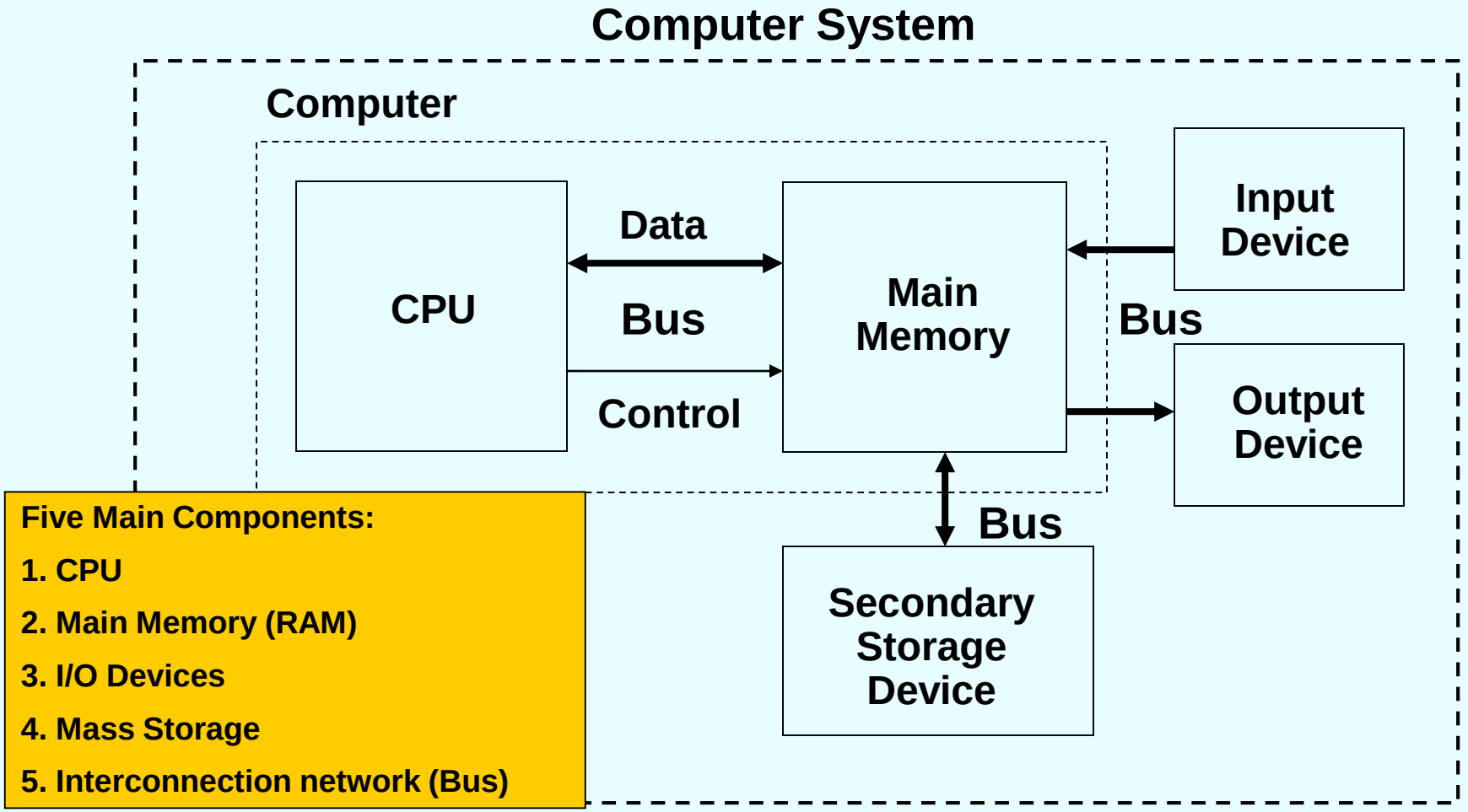
von Neumann Architecture

- Principles
  - Data and instructions are both stored in the main memory(stored program concept)
  - The content of the memory is addressable by location (without regard to what is stored in that location)
  - Instructions are executed sequentially unless the order is explicitly modified
  - The basic architecture of the computer consists of:

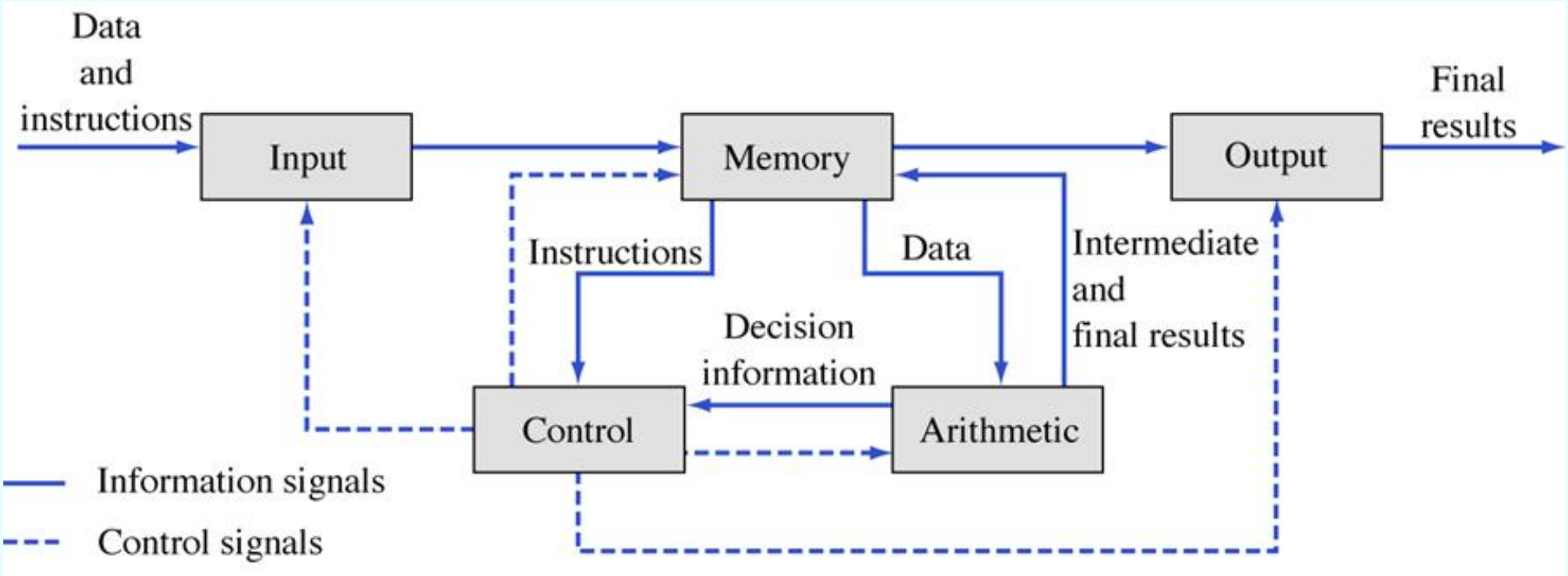


von Neumann Architecture

- A more complete view of the computer *system* architecture that integrates interaction (human or otherwise) consists of:



Another view of a digital computer



Structure of IAS Computer

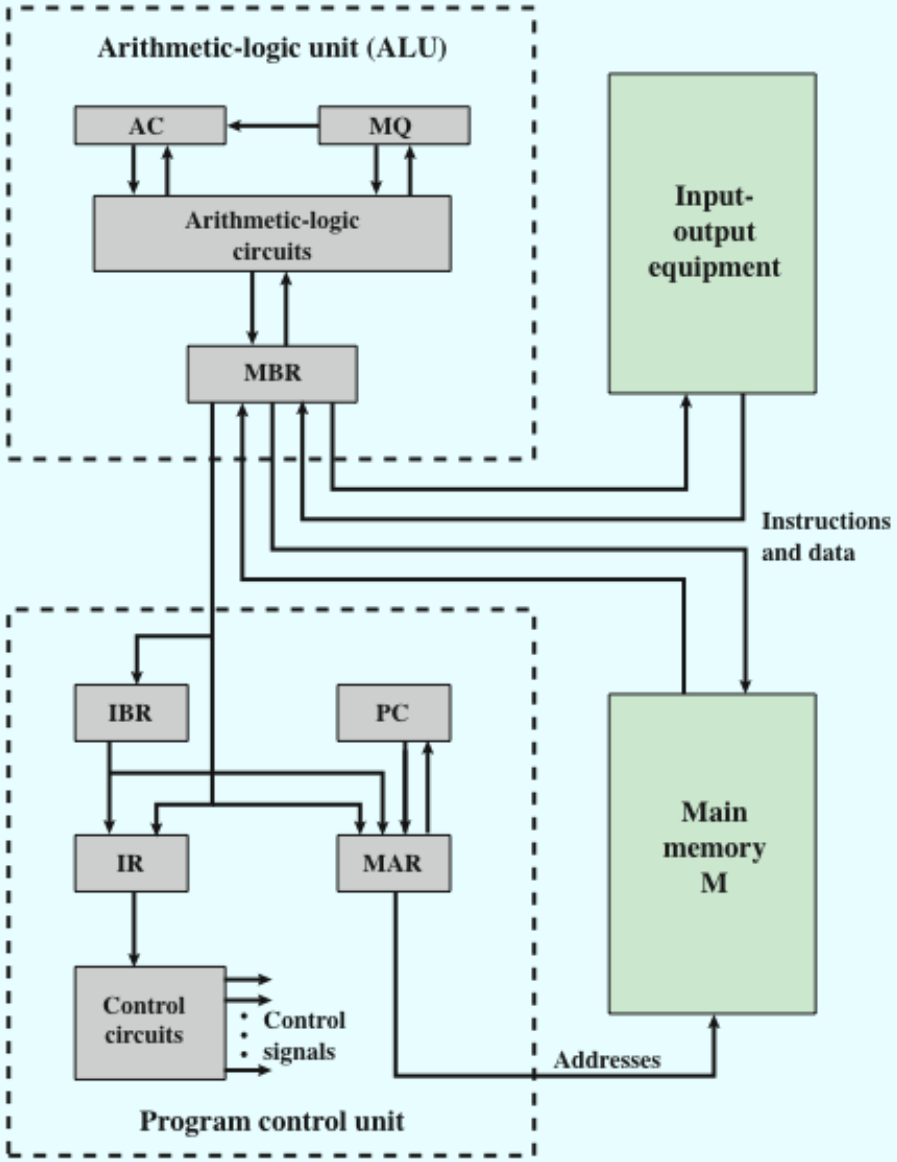


Figure 2.3 Expanded Structure of IAS Computer

Registers

Memory buffer register (MBR)	• Contains a word to be stored in memory or sent to the I/O unit • Or is used to receive a word from memory or from the I/O unit
Memory address register (MAR)	• Specifies the address in memory of the word to be written from or read into the MBR
Instruction register (IR)	• Contains the 8-bit opcode instruction being executed
Instruction buffer register (IBR)	• Employed to temporarily hold the right-hand instruction from a word in memory
Program counter (PC)	• Contains the address of the next instruction pair to be fetched from memory
Accumulator (AC) and multiplier quotient (MQ)	• Employed to temporarily hold operands and results of ALU operations

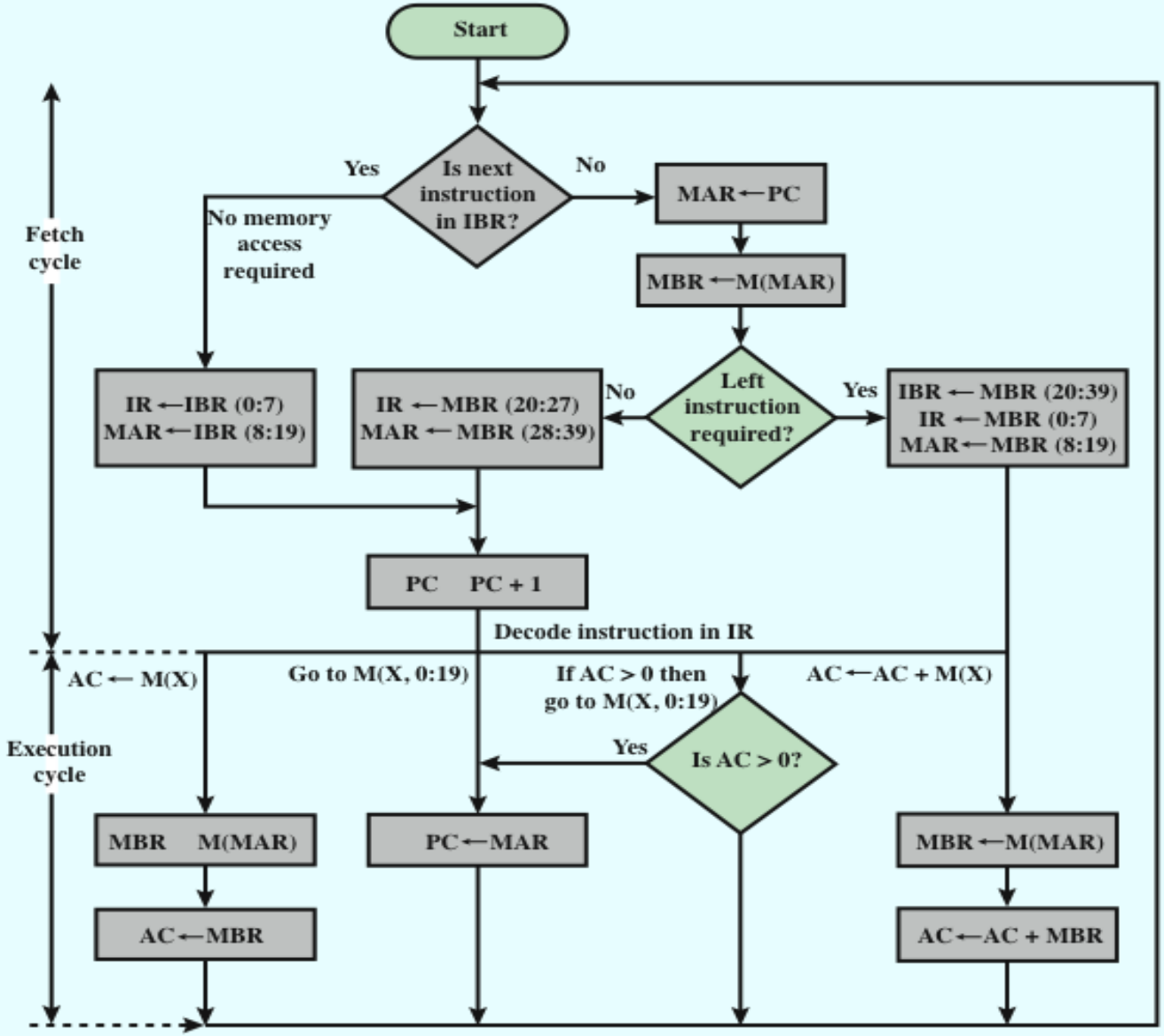


Figure 2.4 Partial Flowchart of IAS Operation

IAS Operations

Instruction Type	Opcode	Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2; i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

Table 2.1

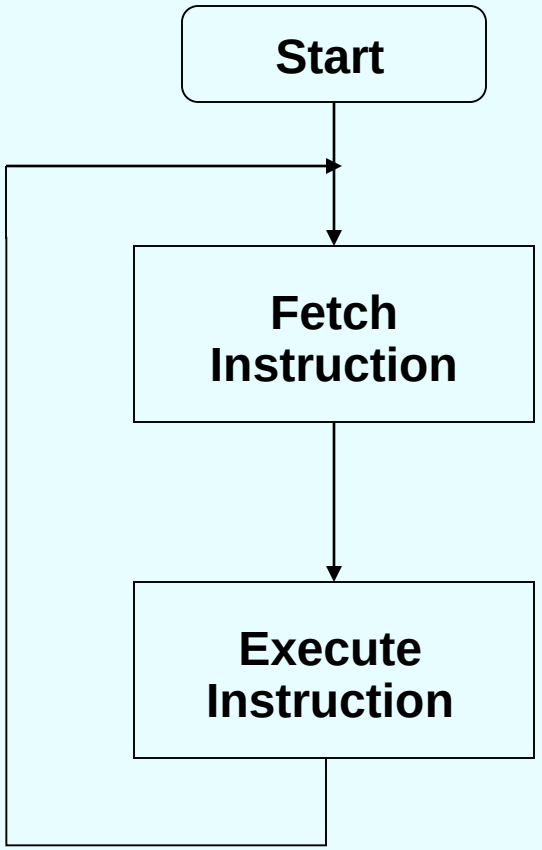
The IAS Instruction Set

The Instruction Cycle

- The Instruction Cycle
  - Basic
  - Intermediate
  - Exceptions

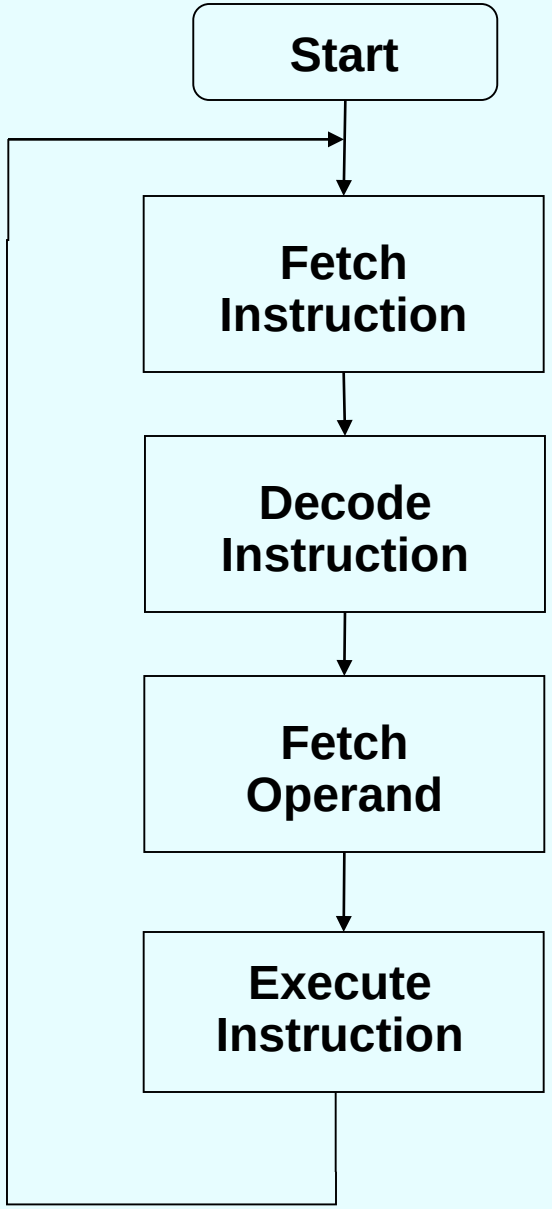
The Instruction Cycle - Basic View

- Once the computer has been started (bootstrapped) it continually executes instructions (until the computer is stopped)
- Different instructions take different amounts of time to execute (typically)
- All instructions and data are contained in main memory



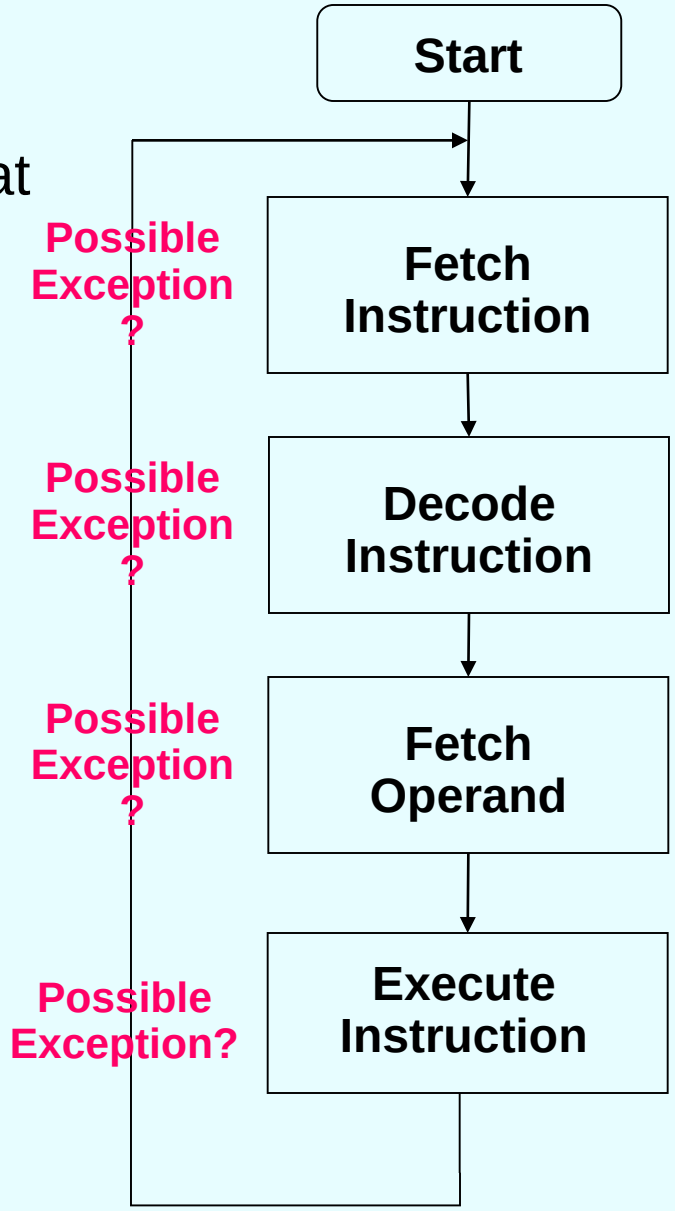
The Instruction Cycle - Intermediate View

- A complete instruction consists of
  - operation code
  - addressing mode
  - zero or more operands
    - immediately available data (embedded within the instruction)
    - the address where the data can be found in main memory



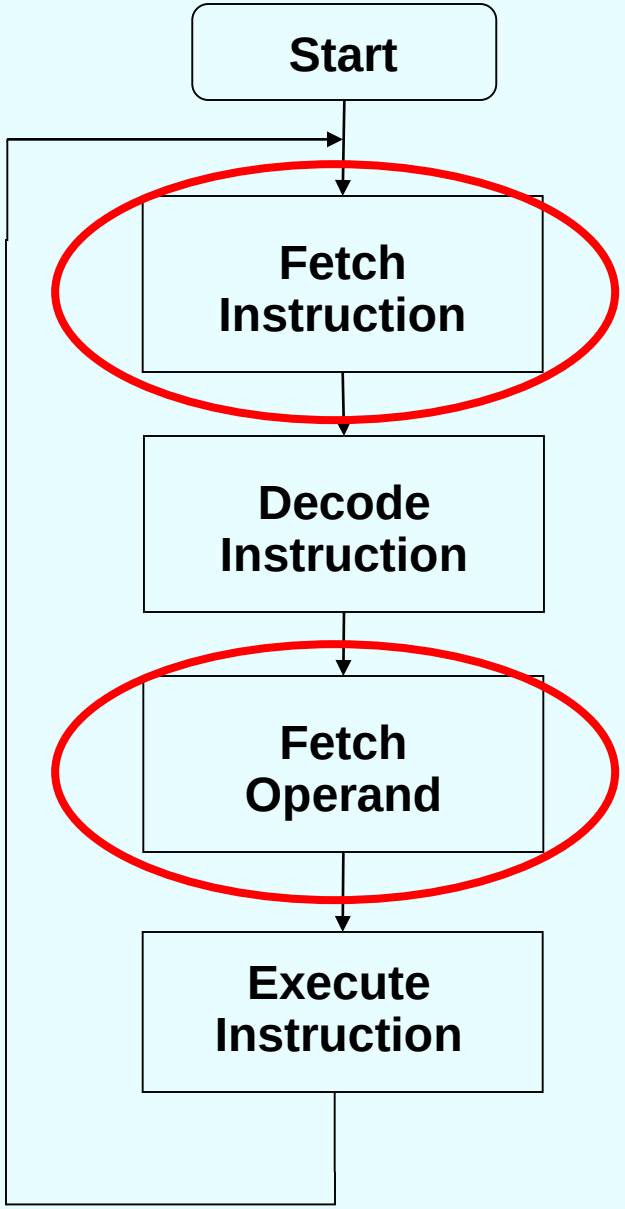
The Instruction Cycle - Exceptions

- Exceptions, or errors, may occur at various points in the instruction cycle, for example:



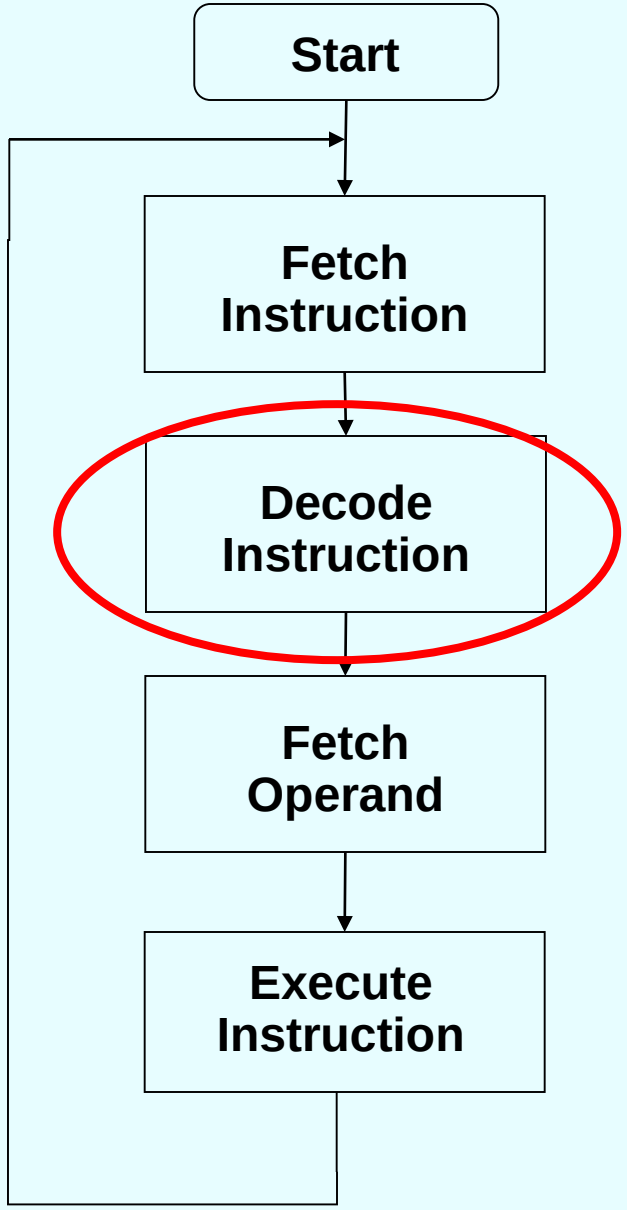
The Instruction Cycle - Exceptions

- Exceptions, or errors, may occur at various points in the instruction cycle, for example:
  - Addressing - the memory does not exist or is inaccessible



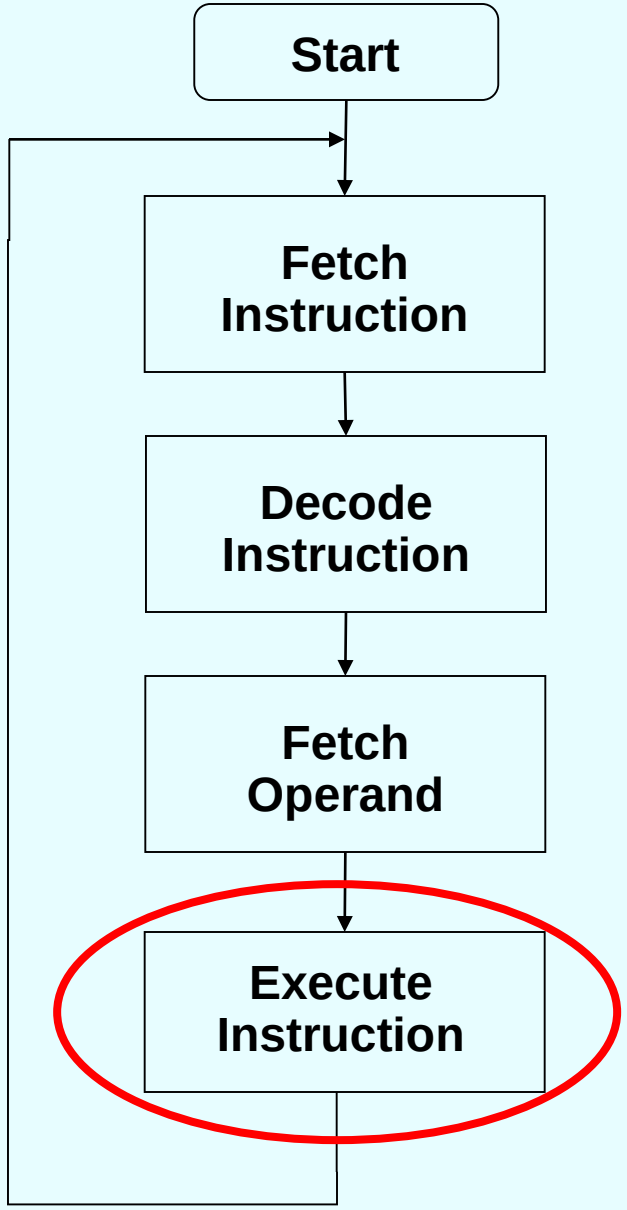
The Instruction Cycle - Exceptions

- Exceptions, or errors, may occur at various points in the instruction cycle, for example:
  - Operation - the operation code does not denote a valid operation



The Instruction Cycle - Exceptions

- Exceptions, or errors, may occur at various points in the instruction cycle, for example:
  - Execution - the instruction logic fails, typically due to the input data
    - divide by zero
    - integer addition/subtraction overflow
    - floating point underflow/overflow

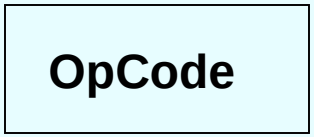


Instruction Architecture

- Software design
- Hardware circuits

Instruction Architecture - Software Design

- Each CPU must be designed to accommodate and understand instructions according to specific formats.
- Examples:
  - All instructions must have an operation code specified
  - NOP no operation
  - TSTST test and set



Instruction Architecture - Software Design

- Each CPU must be designed to accommodate and understand instructions according to specific formats.
- Examples:
  - Most instructions will require one, or more, operands
  - These may be (immediate) data to be used directly
  - or, addresses of memory locations where data will be found (including the address of yet another location)

OpCode	**Operand (Address)**

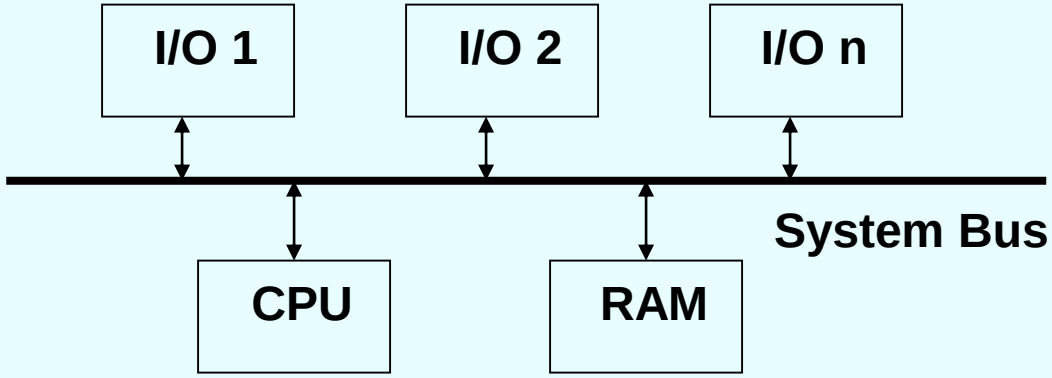
Instruction Architecture - Software Design

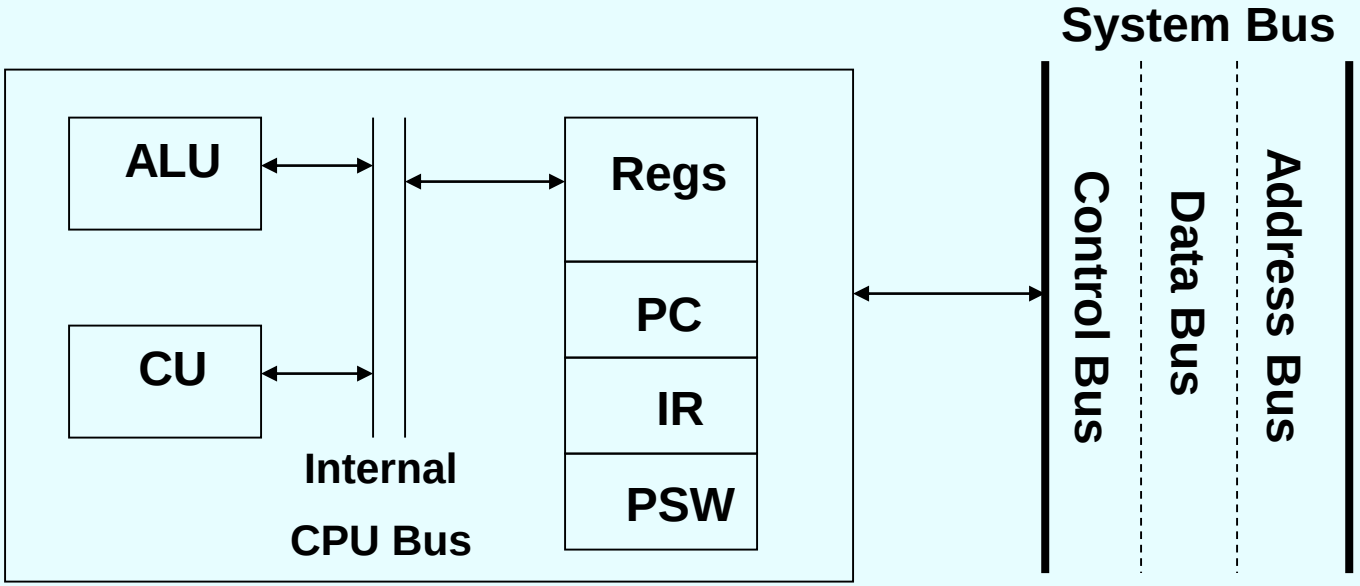
- Sometimes the instruction format requires a code, called the **Mode**, that specifies a particular addressing format to be distinguished from other possible formats
  - direct addressing
  - indirect addressing
  - indexed addressing
  - relative addressing
  - doubly indirect addressing
  - etc.

OpCode	Mode	Op. (Addr.)	Mode	Op. (Addr.)

Instruction Architecture - CPU

- The CPU must be designed to accommodate the instructions and data to be processed





Instruction Architecture - Hardware Circuits

- Everything that the computer can do is the result of designing and building devices to carry out each function – no magic!
- At the most elementary level the devices are called logic gates.
  - Each perform a specific Boolean operation (e.g. AND, OR, NOT, NAND, NOR, XOR, XNOR)
- ALL circuits, hence all functions, are defined in terms of the basic gates.

Instruction Architecture - Hardware Circuits

- Data is represented by various types of “signals”, including electrical, magnetic, optical and so on.
- Data “moves” through the computer along wires that form the various bus networks and which interconnect the gates.

- Combinations of gates are called integrated circuits (IC).

- All computer functions are defined and controlled by IC’s of varying complexity in design. The manufacture of these may be scaled according to size/complexity:
  - LSI / VLSA / ULSA

Instruction Architecture - CU

- The control unit must decode instructions, set up for communication with memory addresses and manage the data stored in the registers and the accumulator.
- Each such operation requires separate circuitry to perform the specialized tasks.
- It is also necessary to know the various data representations to be used on the machine in order to design components that have the desired behaviours.

Instruction Architecture - ALU

- All instructions together are called the instruction set
  - CISC complex instruction set computer
  - RISC reduced instruction set computer

- Each ALU instruction requires a separate circuit, although some instructions may incorporate the circuit logic of other instructions

Goal – Design Circuits!

- After all the conceptualization we must now get down to the most fundamental thing – learning how to design circuits that can implement the logic we intend to impose and use

- Circuit design arises out of a study of Boolean Set Theory and Boolean Algebra

- It works towards higher level abstraction of device components, but starts at an elementary level of concrete behaviors with predefined units called ***gates***.