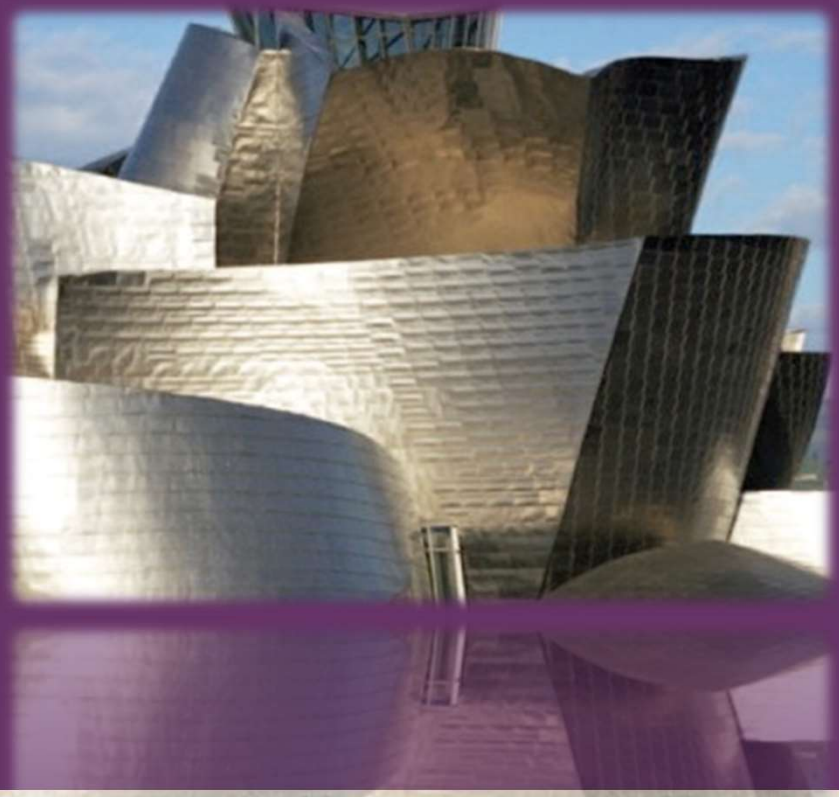
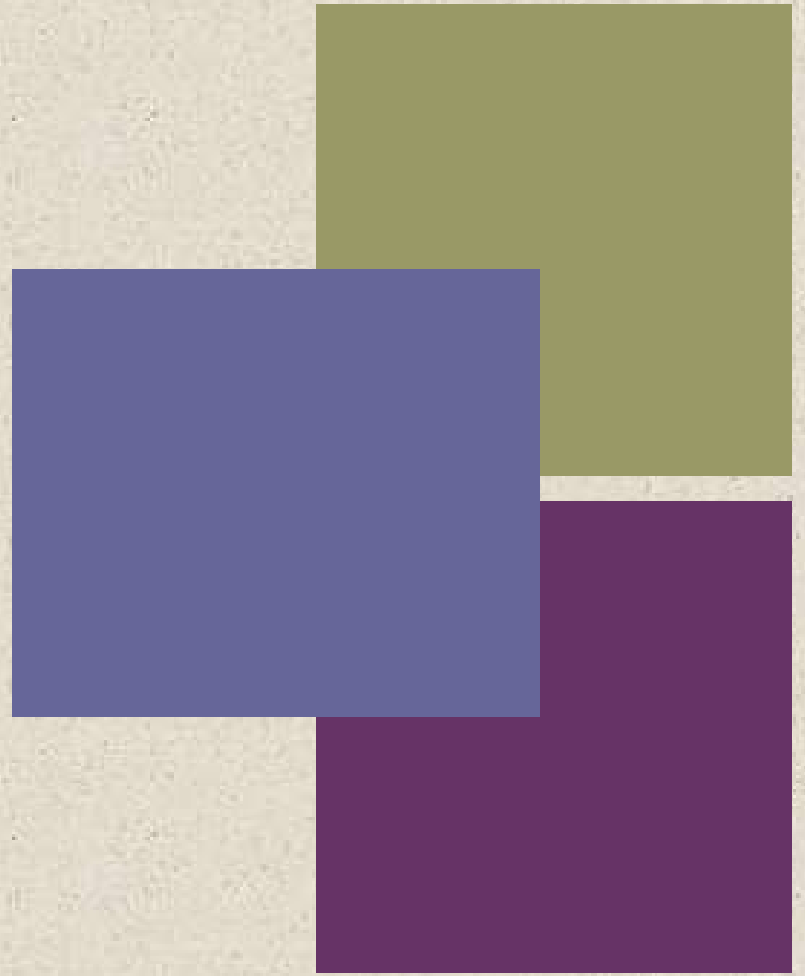


William Stallings Computer Organization and Architecture 10th Edition



Chapter 3

A Top-Level View of Computer Function and Interconnection

Recap

- Designing for Performance
- Performance Balance
- Improvement in chip organization and architecture
- Diminishing Returns

Overview

- Mainly three topics discussed
  - Instruction cycle, program execution
  - Interrupts
  - Bus system
    - Hierarchy of buses
    - Key design elements of buses

Top Level View of Computer

- At top level computer consists of …?
- These components are interconnected in some fashion to execute a program
- Thus at top level we describe computer system by:
  - Describing the external behaviour of each component. i.e. the data and the control signals it exchanges with other component
  - Describing the interconnection structure and the controls required to manage its use

Computer Components

- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- 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 occurs in a sequential fashion (unless explicitly modified) from one instruction to the next

Hard Wired Program

- Various hardware components can be connected to gather to perform a specific computation:
  - Comparison of two numbers
  - Addition of two Numbers
  - Multiplication of two Numbers
- We can think of connecting these components as a form of programing:
  - The resultant program is hardwired program (Customized Hardware)
  - For every different function, there will be different structure

General Purpose Configuration of Hardware

- In hardwired program system accepted data and produced output
- We construct a general purpose configuration of Arithmetic and Logic functions
- Performs various functions on data depending on control signals applied to the hardware:
  - Accepts Data like hardwired
  - Accepts control signals unlike hardwired programs
  - Thus, instead of rewiring apply new set of control signals

How to apply Control Signals?

- Program is a sequence of steps
- At each step some arithmetic or logic function is performed
- For each step a new control signal is required (may be same)
- We provide a unique code for each set of control signals
  - We add a segment into General purpose Hardware configuration
    - Accept a code and generate control signals

How to apply Control Signals?

- Each code is in form a instruction
  - Interpreted by hardware to generate new set of control signals
- So this new method of programming where sequence of codes or instructions are used is called software

Hardware and Software Approaches

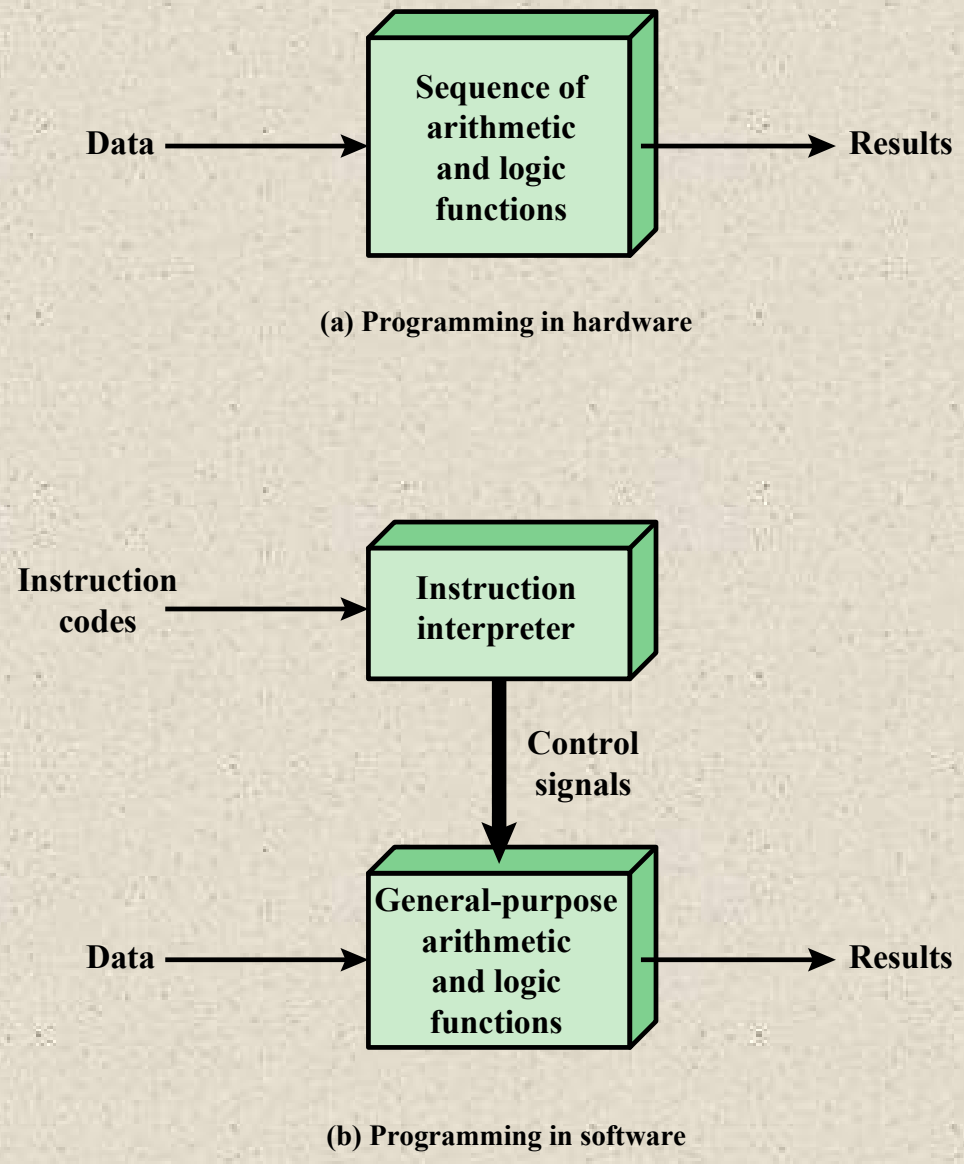
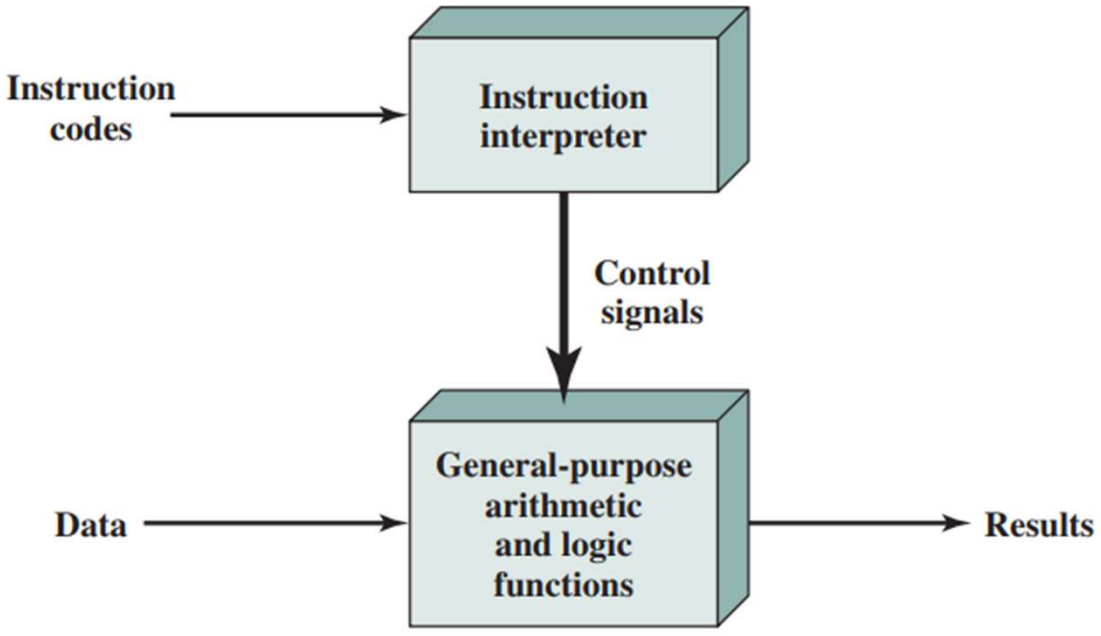


Figure 3.1 Hardware and Software Approaches

Figure 3.1 b

- Indicates the CPU



(b) Programming in software

Components contd…

- Several other components are needed in order to yield a function in computer:
  - Data and instructions must be put into the system (Input Module):
    - Accepting data and converting it into a form understandable by the computer
  - A means for reporting results is needed (Output Module):
    - Takes signals and converts them in a form understandable by the user of the system
  - A place to temporarily hold instructions and data:
    - Memory or Main Memory

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

Major components:

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results

Software

I/O Components

Communication of CPU with Memory

- CPU exchanges data with memory
- It makes use of two registers:
  - MAR:
    - Hold the address of current read/write in memory
  - MBR:
    - Data to be written into the memory or to be accessed from the memory

Communication of CPU with I/O

- I/O Module vs I/O Device?
- CPU Communicates using two registers:
- I/OAR (I/O Address Register):
  - Specifies a particular I/O Device
- I/OBR(I/O Buffer Register):
  - Exchange of data between an I/O module and CPU
- I/O Buffers are also present to hold the data temporarily

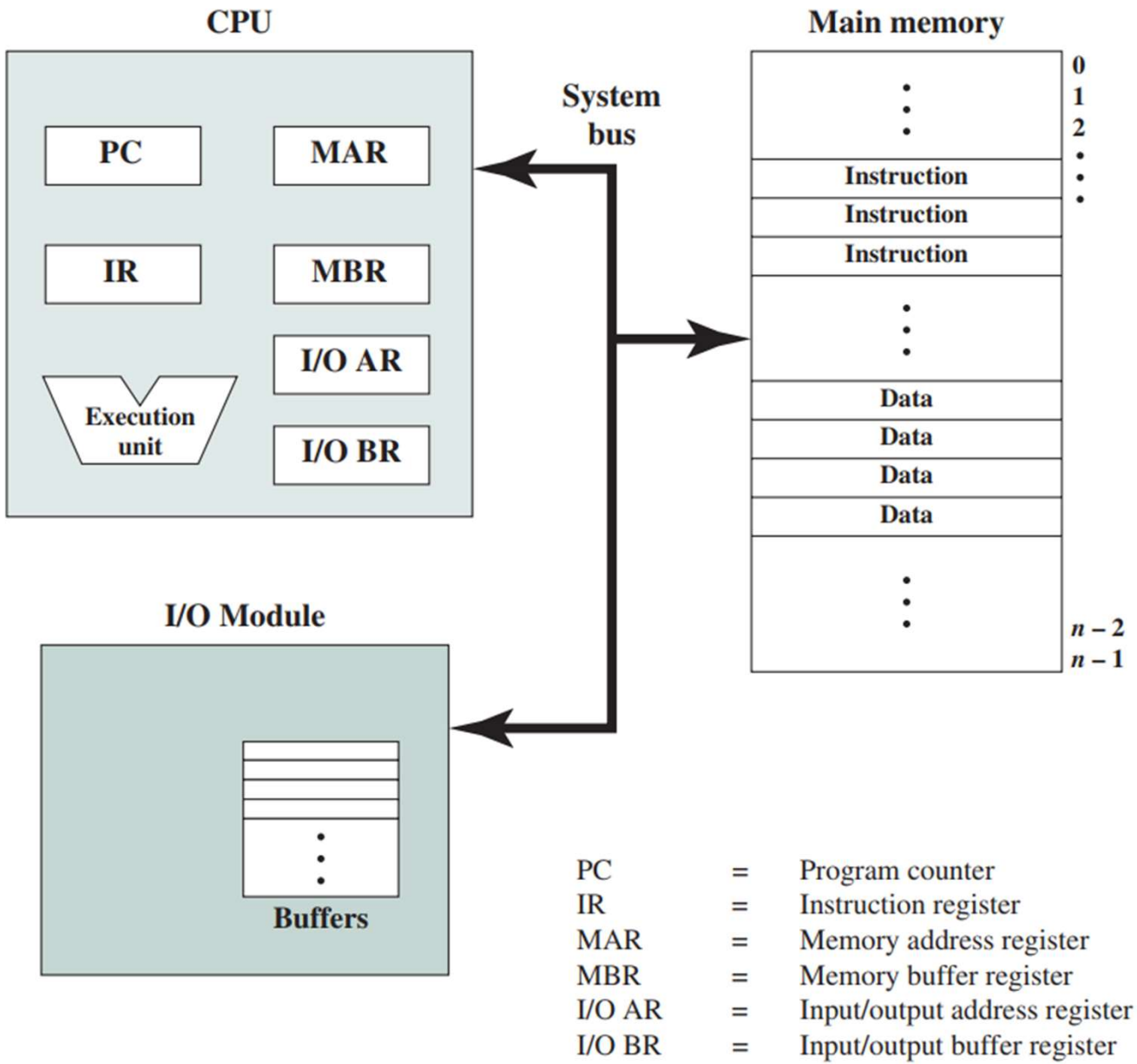
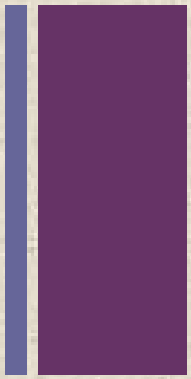


Figure 3.2 Computer Components: Top-Level View

Now



- We turn to an overview of how these components function together to execute programs

Computer Function

- What is the basic function?
  - Execution of Programs
    - Set of Instructions stored in memory
- At simplest Instruction processing is:
  - Fetch
  - Execute
  - Repeatedly until turns off, error occurs, or halt instruction is executed
- Instruction Cycle:
  - Processing required for a single instruction

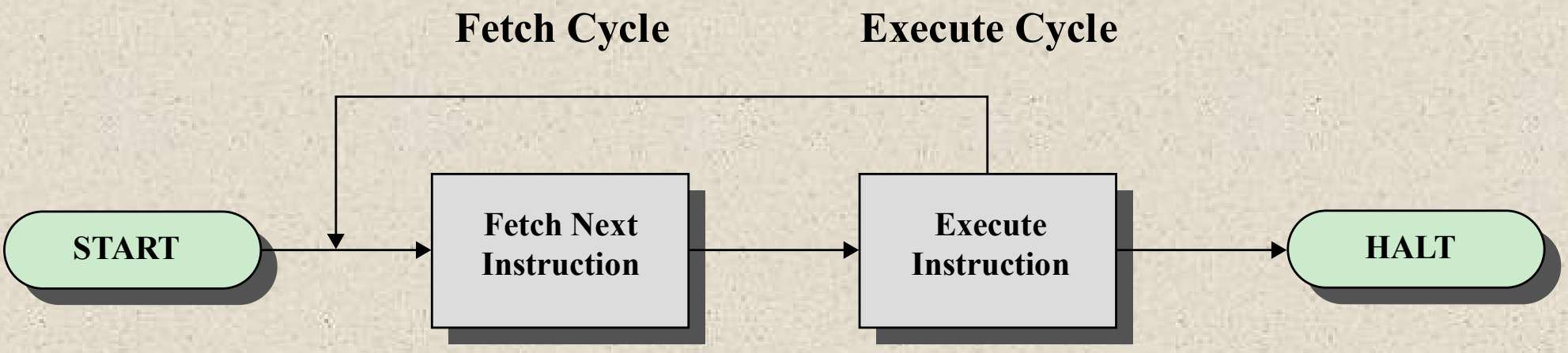
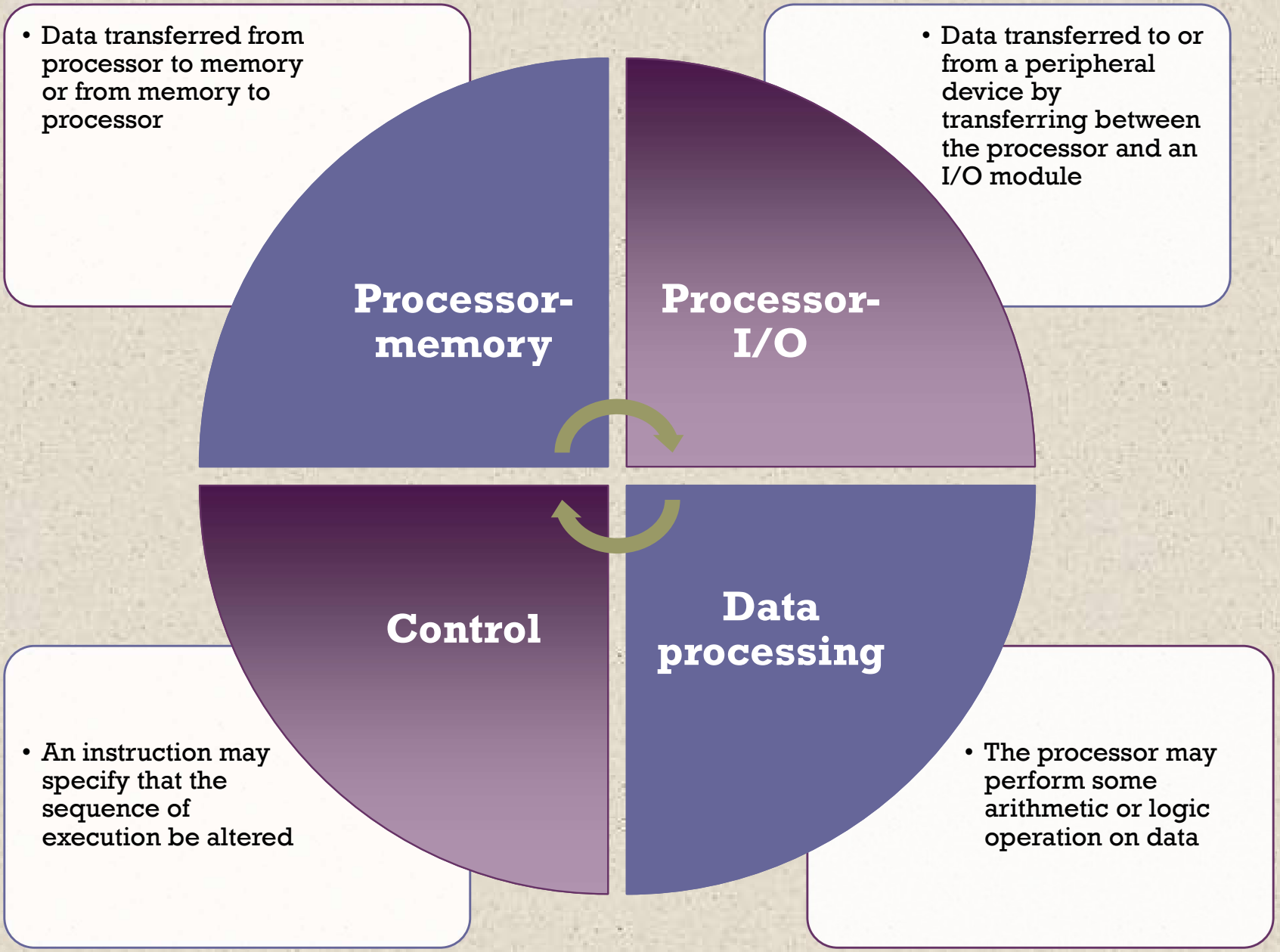


Figure 3.3 Basic Instruction Cycle

Fetch Cycle

- At the beginning of each instruction cycle the processor fetches an instruction from memory
- The program counter (PC) holds the address of the instruction to be fetched next
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
- The fetched instruction is loaded into the instruction register (IR)
- The processor interprets the instruction and performs the required action

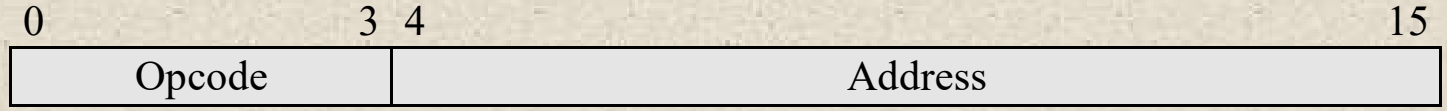
Action Categories



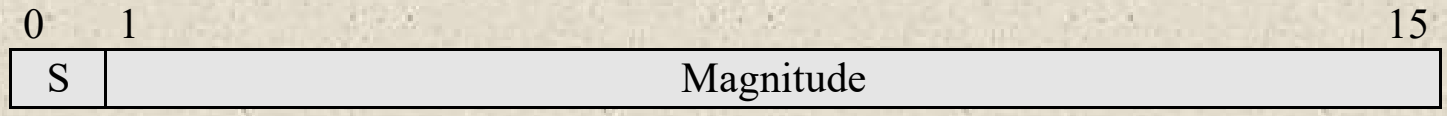
Explaining Control

- Processor fetched an Instruction from memory location 149
- Instruction specifies that, next instruction will be fetched from 182
- Thus on next fetch cycle:
  - Instruction will be fetched from 182 instead of 150

- Figure in next slide show:
  - Both data and Instructions are 16 bits
  - 4 bits for Opcode: 2^4 opcodes
  - 2^{12} words of memory are directly addressable (4096 locations)



(a) Instruction format



(b) Integer format

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

(c) Internal CPU registers

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

(d) Partial list of opcodes

Figure 3.4 Characteristics of a Hypothetical Machine

- Figure in next slide show:
  - Addition of contents at memory location 940 with 941 location, and storing the results back at 941 location

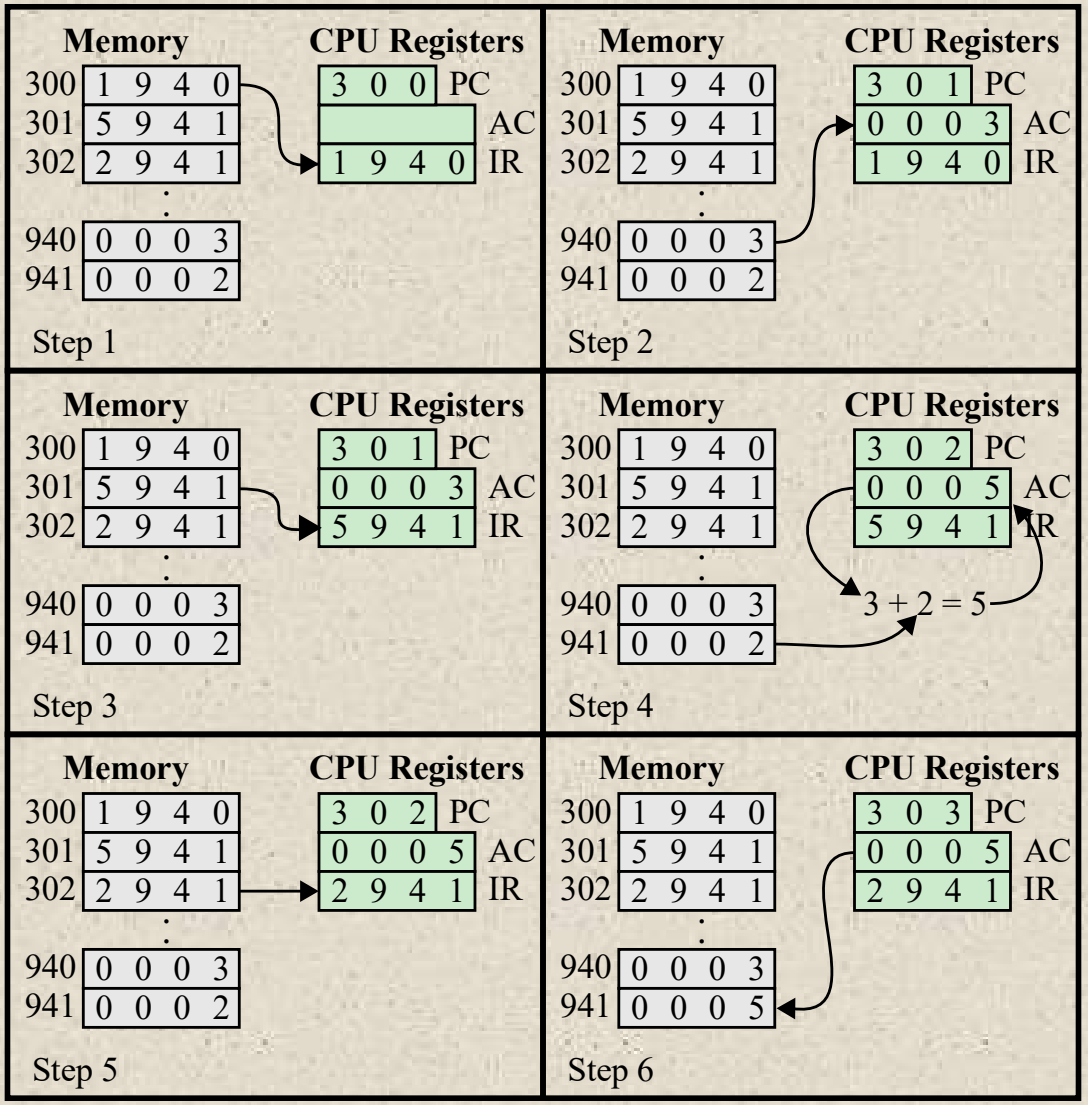


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

1. The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexadecimal) is loaded into the instruction register IR and the PC is incremented. Note that this process involves the use of a memory address register (MAR) and a memory buffer register (MBR). For simplicity, these intermediate registers are ignored.
2. The first 4 bits (first hexadecimal digit) in the IR indicate that the AC is to be loaded. The remaining 12 bits (three hexadecimal digits) specify the address (940) from which data are to be loaded.
3. The next instruction (5941) is fetched from location 301 and the PC is incremented.
4. The old contents of the AC and the contents of location 941 are added and the result is stored in the AC.
5. The next instruction (2941) is fetched from location 302 and the PC is incremented.
6. The contents of the AC are stored in location 941.

Do it yourself ☺

- Action to perform: SUB contents of memory location 940 to the contents of memory location 941 and store the result back to location 940
  - Load 940
  - SUB 941 to AC
  - Store AC to 940
  - 0110 IS FOR SUBTRACTION