+

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

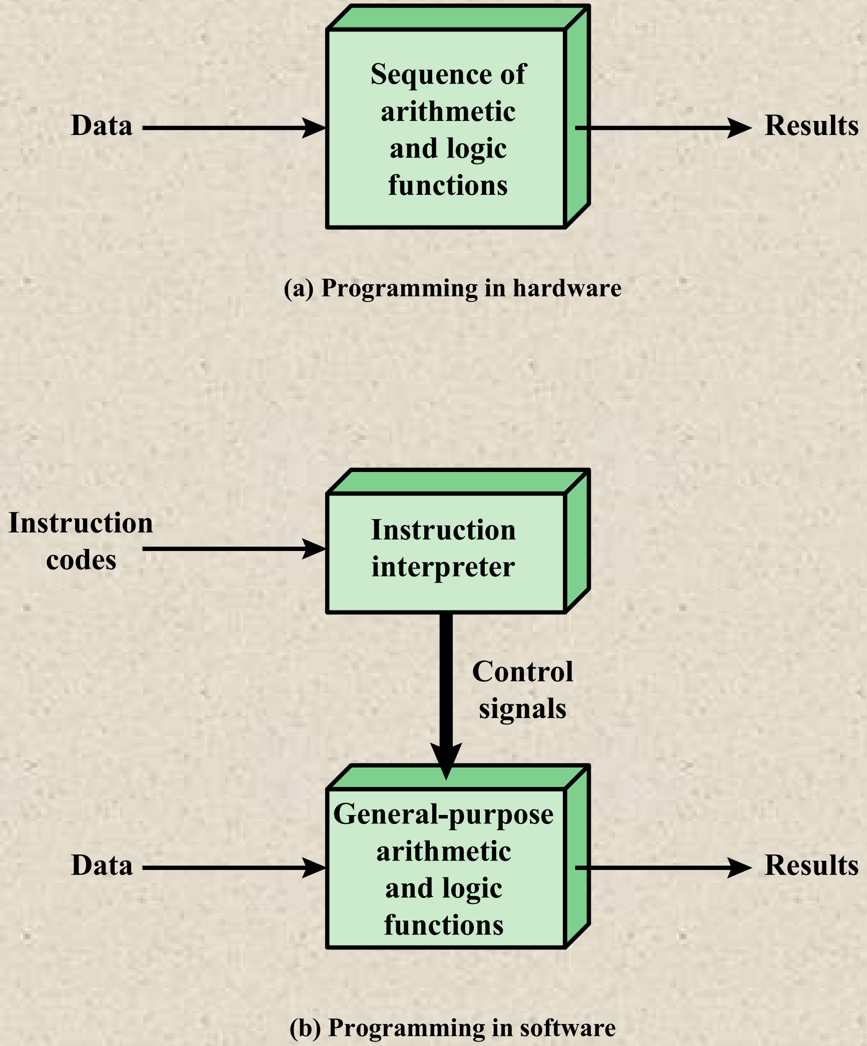
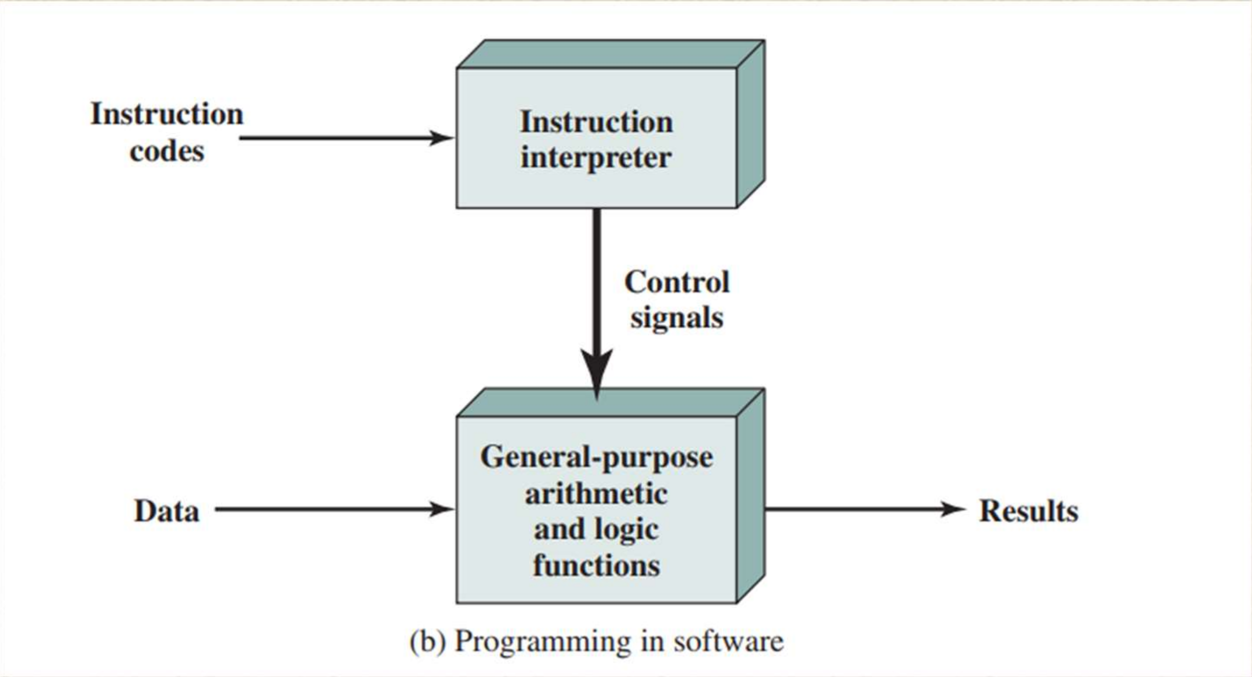

Data
Sequence of arithmetic and logic functions
Results
(a) Programming in hardware
Instruction codes
Instruction interpreter
Control signals
Data
General-purpose arithmetic and logic functions
Results
(b) Programming in software

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

+

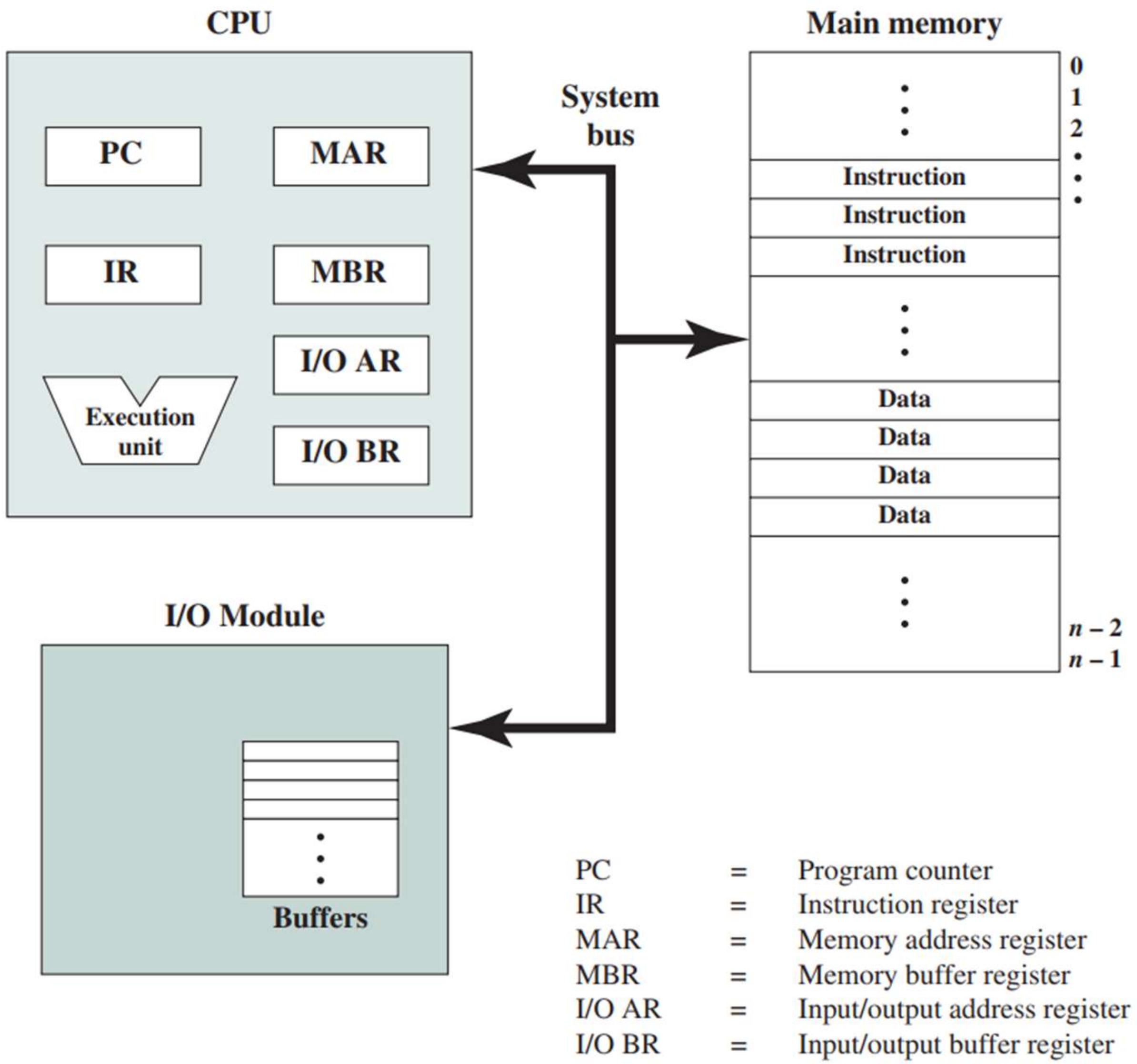
Communication of CPU with Memory

- 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



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

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

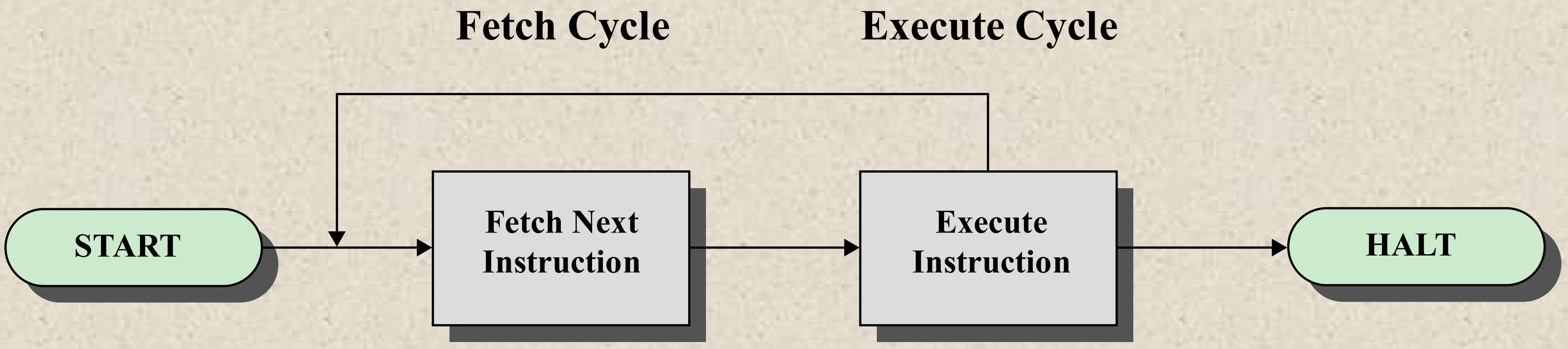


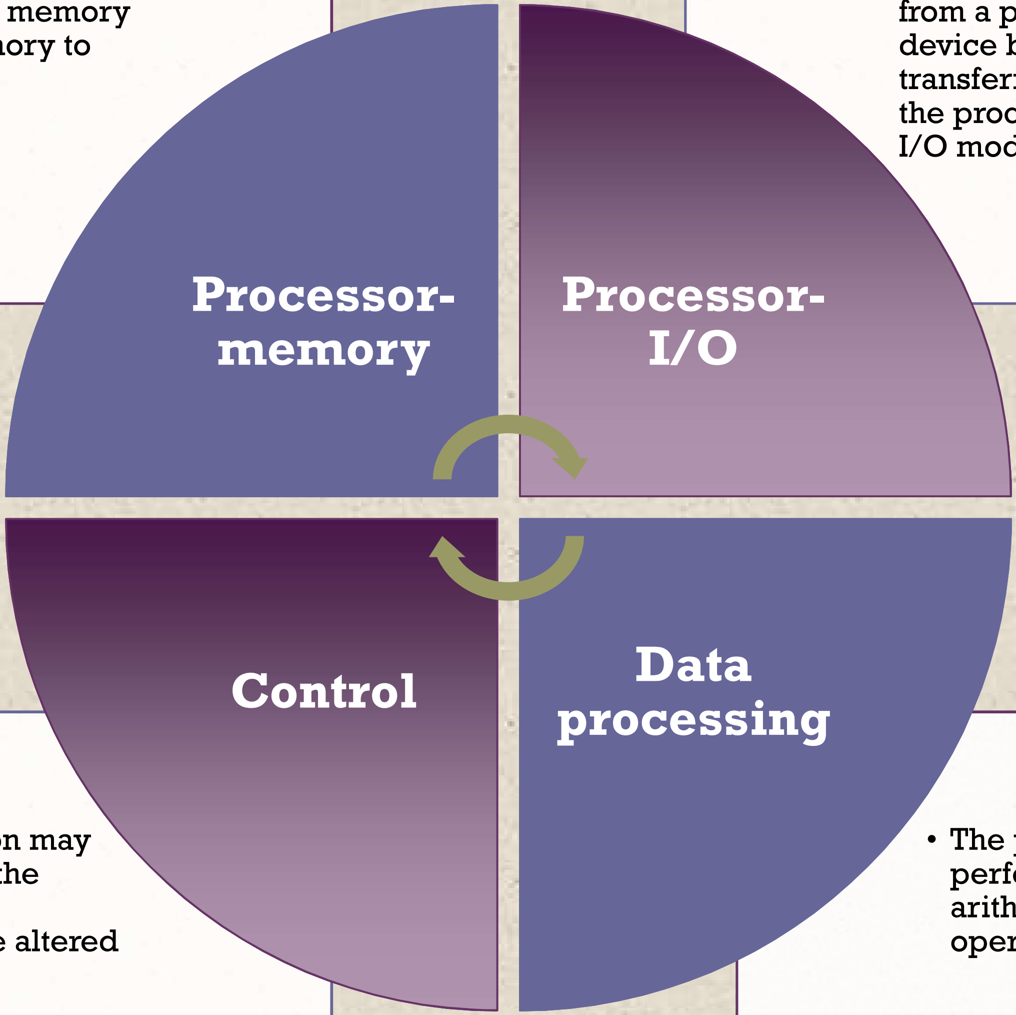
Figure 3.3 Basic Instruction Cycle

+

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



Processor-memory

- Data transferred from processor to memory or from memory to processor

Processor-I/O

- Data 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

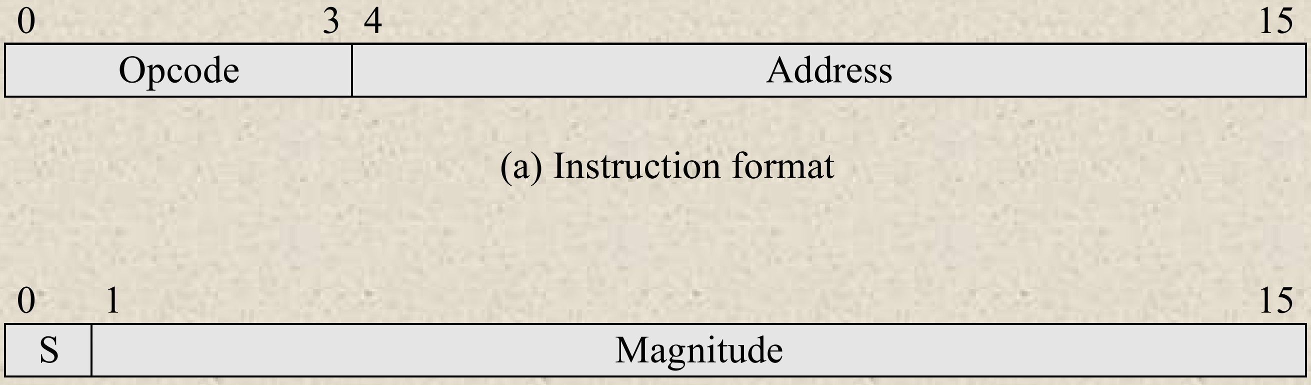
Control

- An instruction may specify that the sequence of execution be altered

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

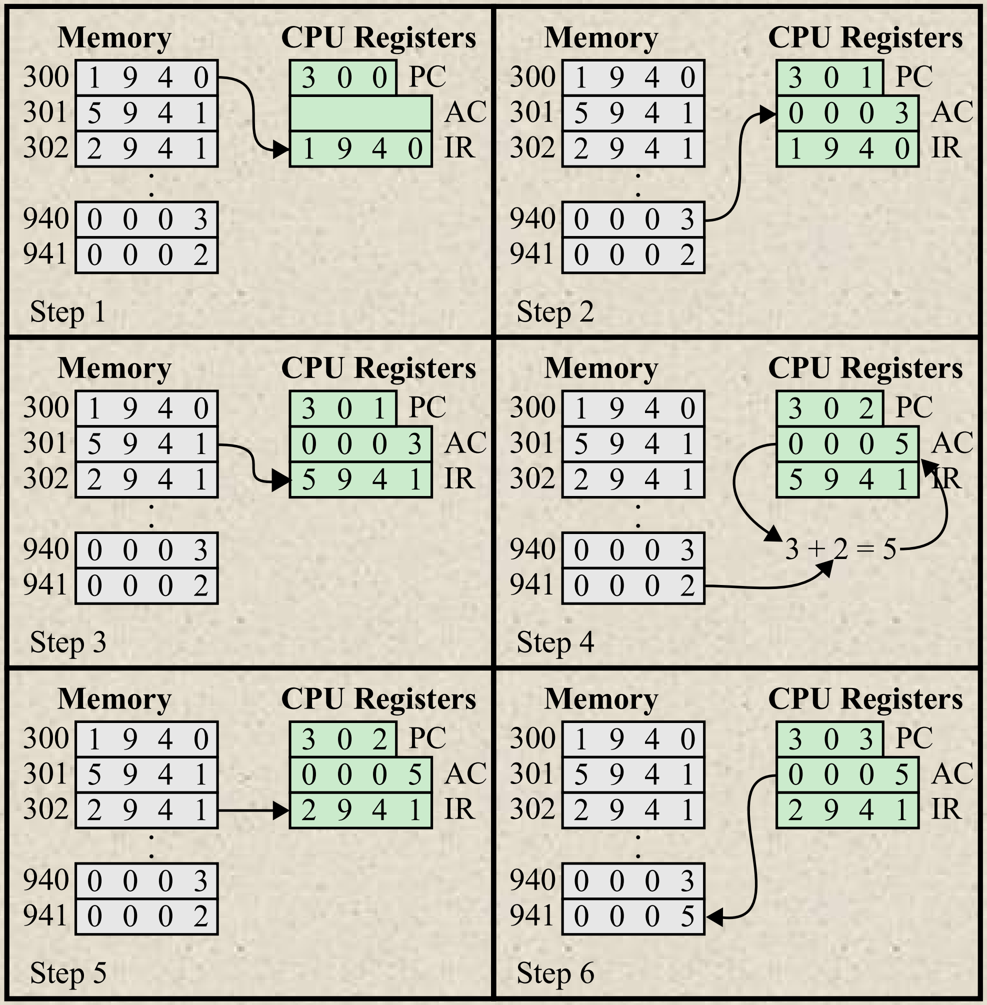
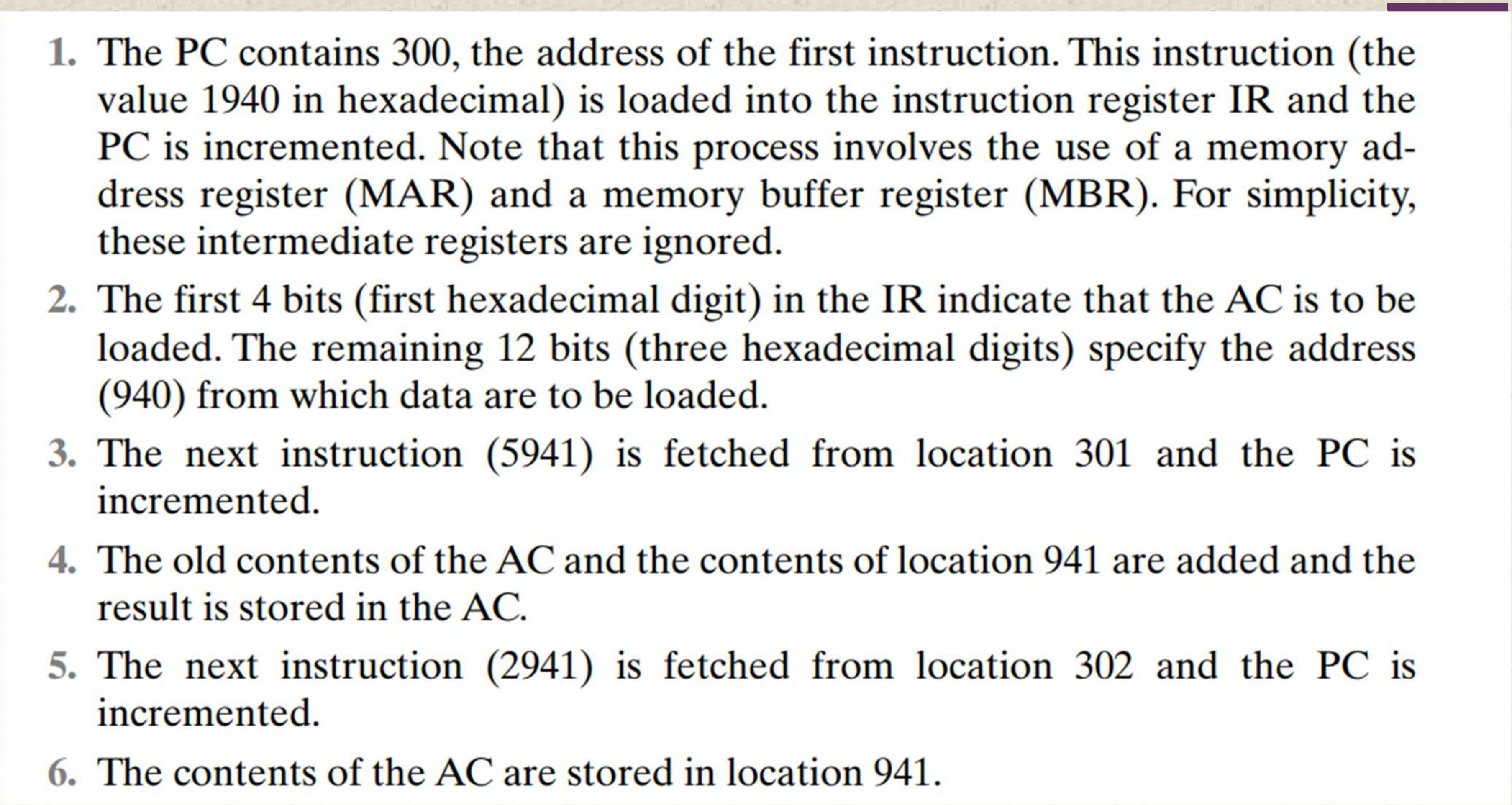


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)



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

+
More Detailed look of an Instruction Cycle

- Figure 3.6

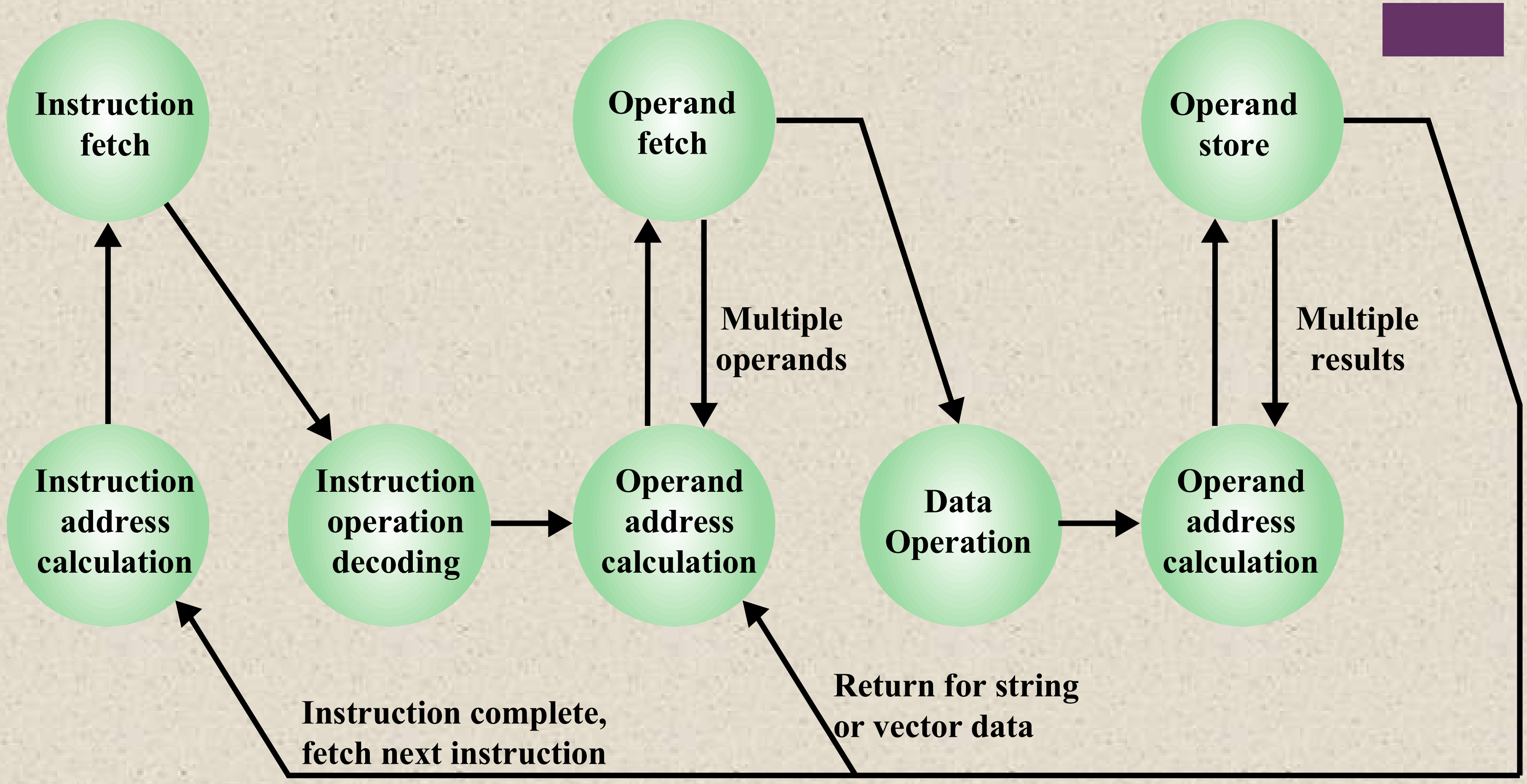

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Return for string or vector data
Instruction complete, fetch next instruction

Figure 3.6 Instruction Cycle State Diagram

+

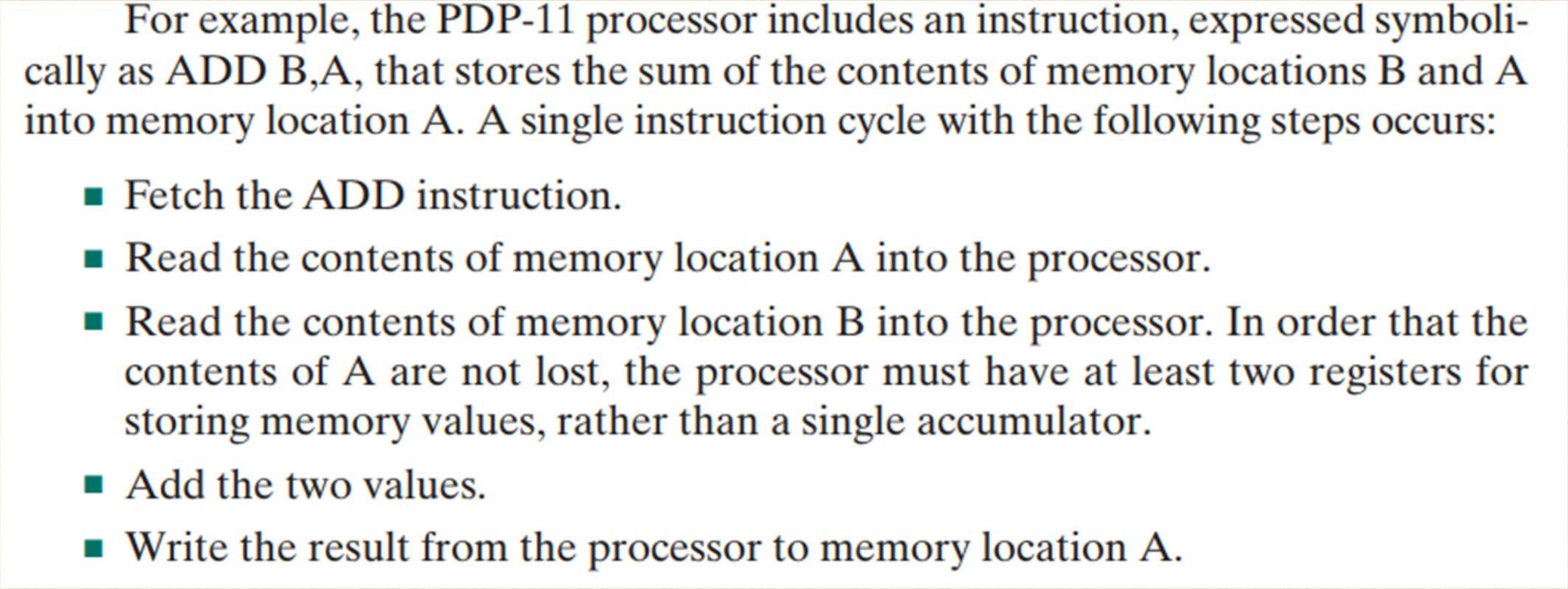
More Detailed look of an Instruction Cycle

- IAC: Calculate the address of next instruction to be executed
- IF: Read instruction from memory location to processor
- IOD: Decode to determine type of operation and operands to be used
- OAC: Provide reference to an operand if it is in memory or available via I/O
- OF: Fetch operand from memory or from I/O
- DO: Perform operation as indicated in instruction
- Operand Store (OS): Write the result in memory or out to I/O

Task

- Map PDP-11 instruction ADD A,B to the state diagram, Instruction Resides at memory location 940, Operand A resides at 903, Operand B resides at 904

+



For example, the PDP-11 processor includes an instruction, expressed symbolically as ADD B,A, that stores the sum of the contents of memory locations B and A into memory location A. A single instruction cycle with the following steps occurs:

- Fetch the ADD instruction.
- Read the contents of memory location A into the processor.
- Read the contents of memory location B into the processor. In order that the contents of A are not lost, the processor must have at least two registers for storing memory values, rather than a single accumulator.
- Add the two values.
- Write the result from the processor to memory location A.

Interrupt

- Mechanism:
  - By which other modules (I/O, Memory) may interrupt the normal processing of the processor
- Primarily used to improve processor efficiency:
  - External devices are slower than processor
  - Exp: Processor writing to a printer (pauses of thousand instruction cycles)
  - After each write operation processor has to wait for printer to catch-up

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, request service from the processor, or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Table 3.1

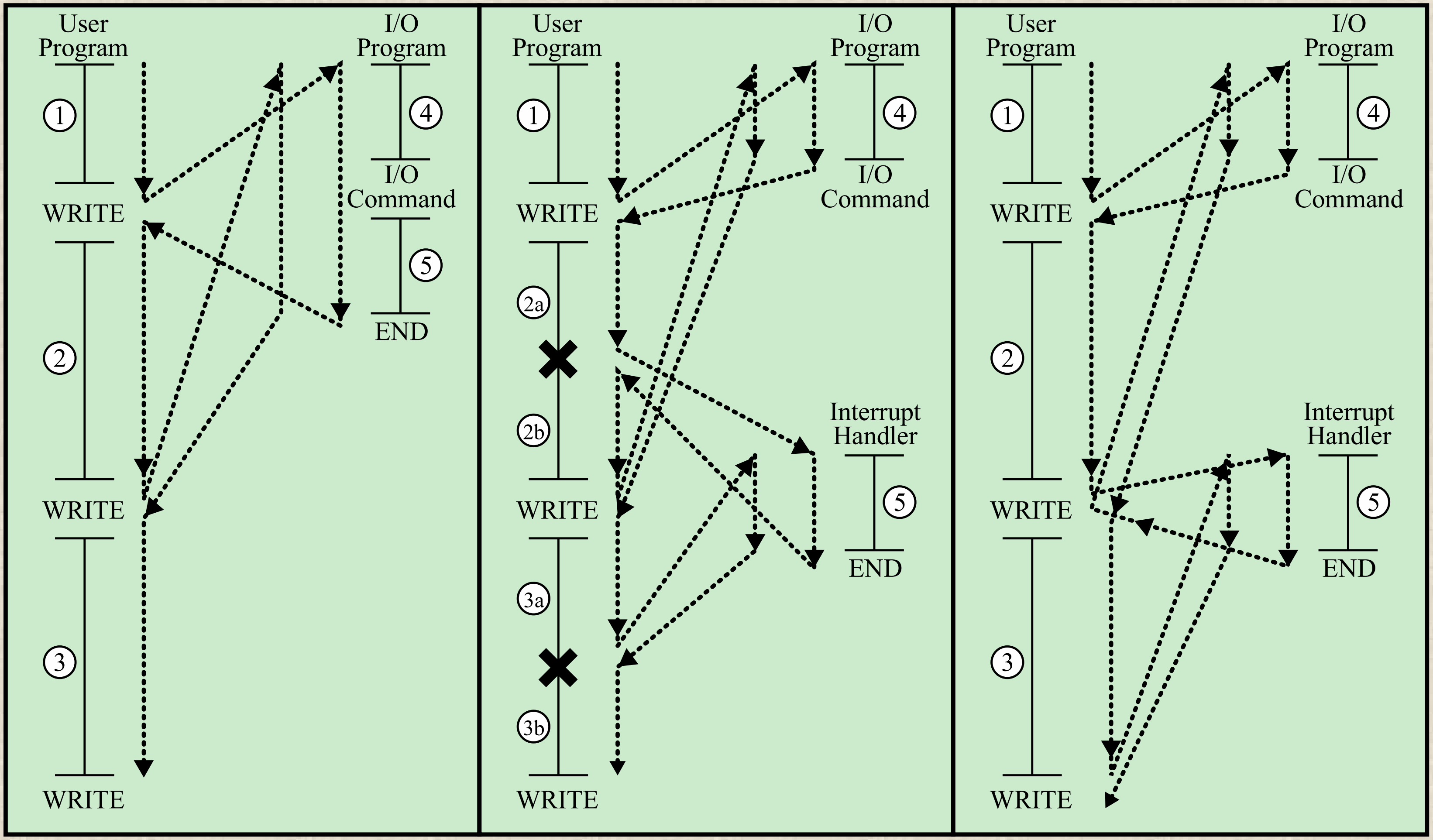
Classes of Interrupts

+
No Interrupts

- User program contains set of Instructions labelled from 1 to 3
- Write calls are to I/O Program
  - Processor waits until these calls get executed
- I/O Program consists of:
  - Preparation for actual I/O Labelled 4
  - Actual I/O Command to perform action
  - Sequence of Instructions labelled 5 to indicate the status through flags:
    - Complete/Failur

With Interrupts

- Processor can perform executions even when I/O is in progress



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

✖ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

Interrupts and The Instruction Cycle

- Processor is engaged while I/O is in progress
- When device is ready to accept more data from processor:
  - I/O Module sends an interrupt signal
  - Current state of program is saved
  - Control is transferred to Interrupt handler
  - Resume after servicing the Interrupt

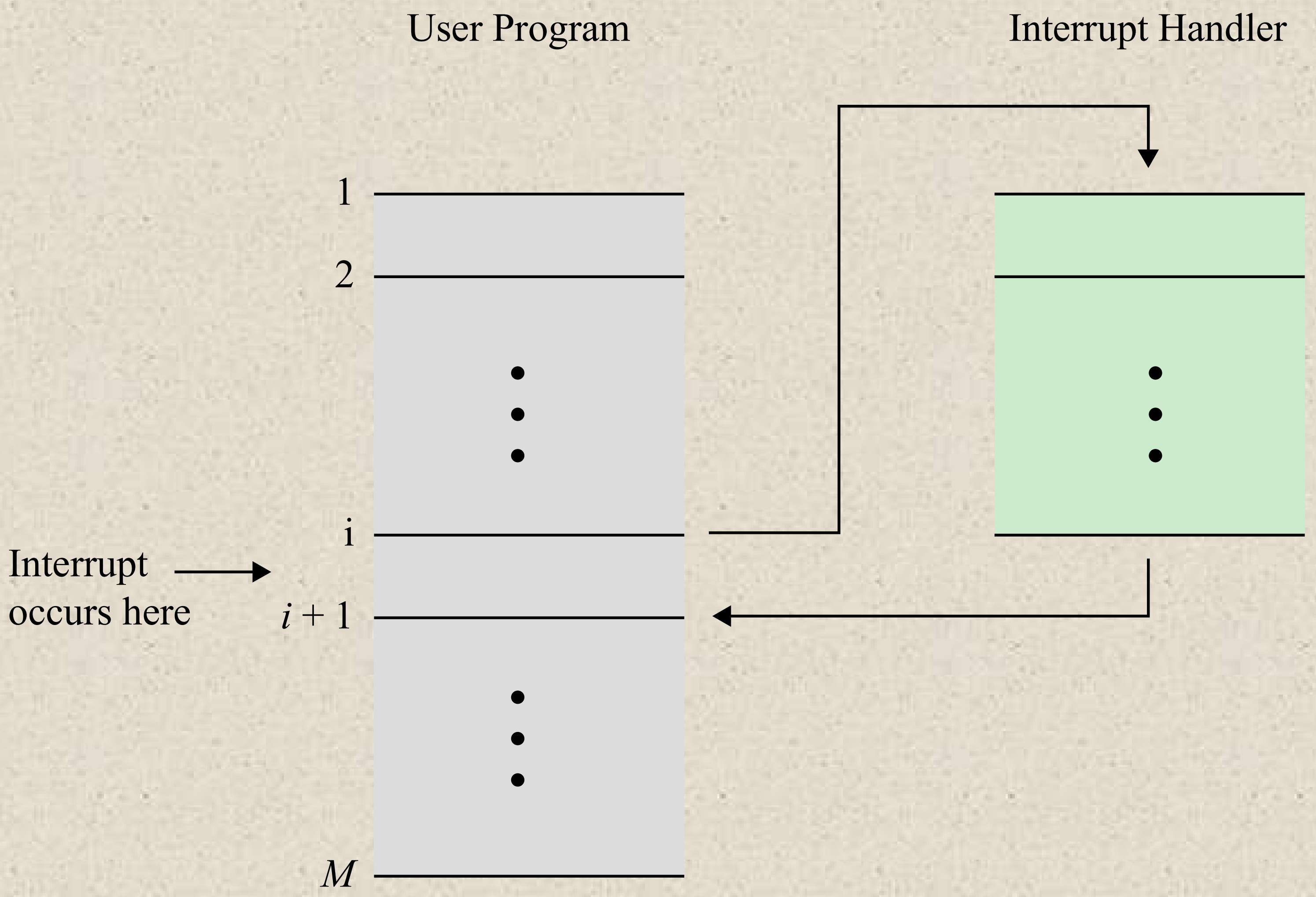


Figure 3.8 Transfer of Control via Interrupts

+

Addition of Interrupt cycle to Instruction cycle

- Interrupt cycle is added to check for interrupts
- If there is no interrupt:
  - Proceed to fetch next instruction of current program
- But if Interrupt is pending:
  - Processor saves the context of current program
  - Sets the program counter to starting address of Interrupt handler routine and interrupt is serviced
- After servicing interrupts the current program resumes and PC is set to the next instruction

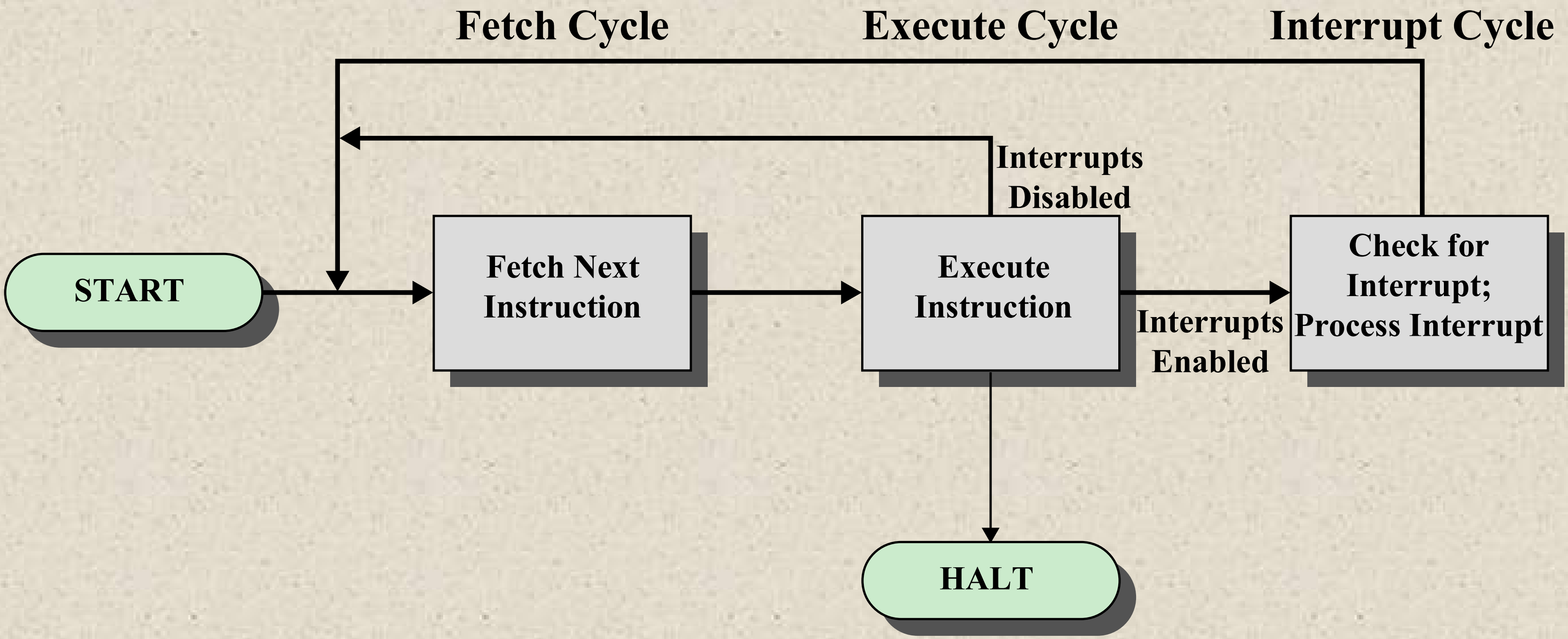


Figure 3.9 Instruction Cycle with Interrupts

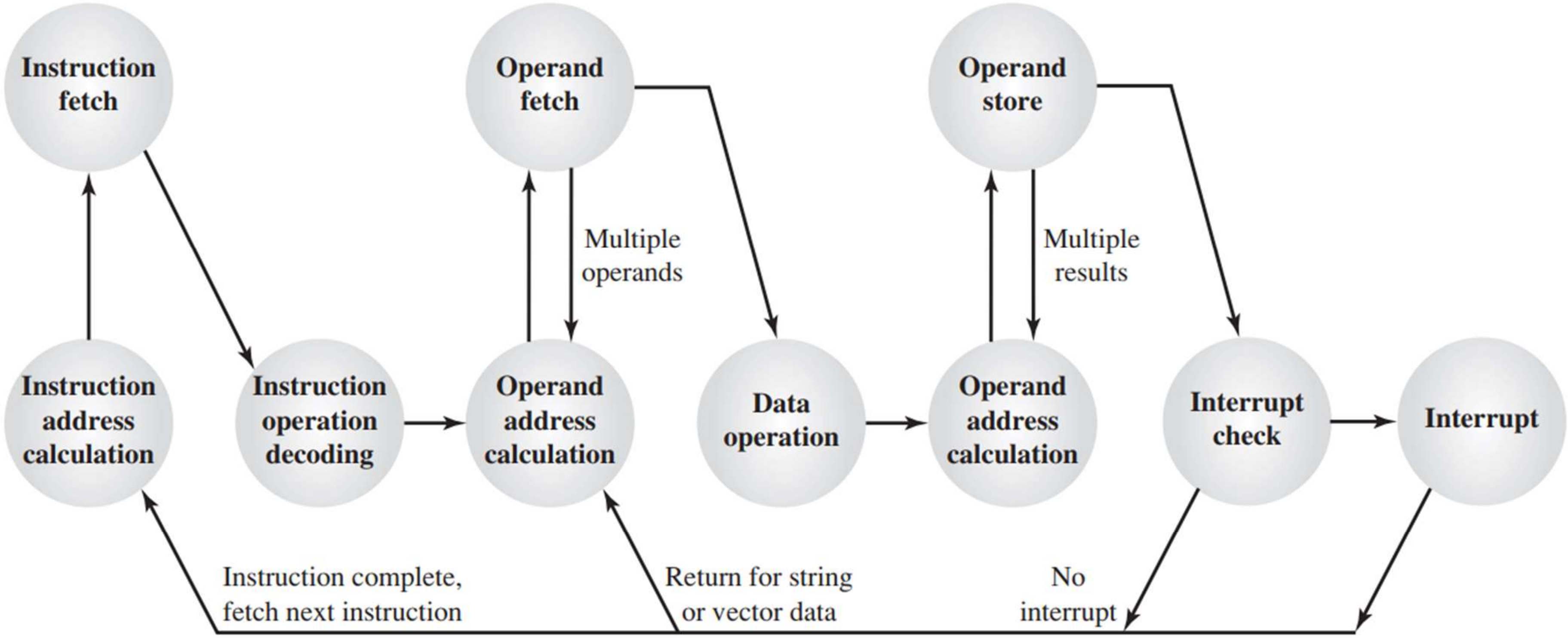

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string or vector data
No interrupt

Figure 3.12 Instruction Cycle State Diagram, with Interrupts

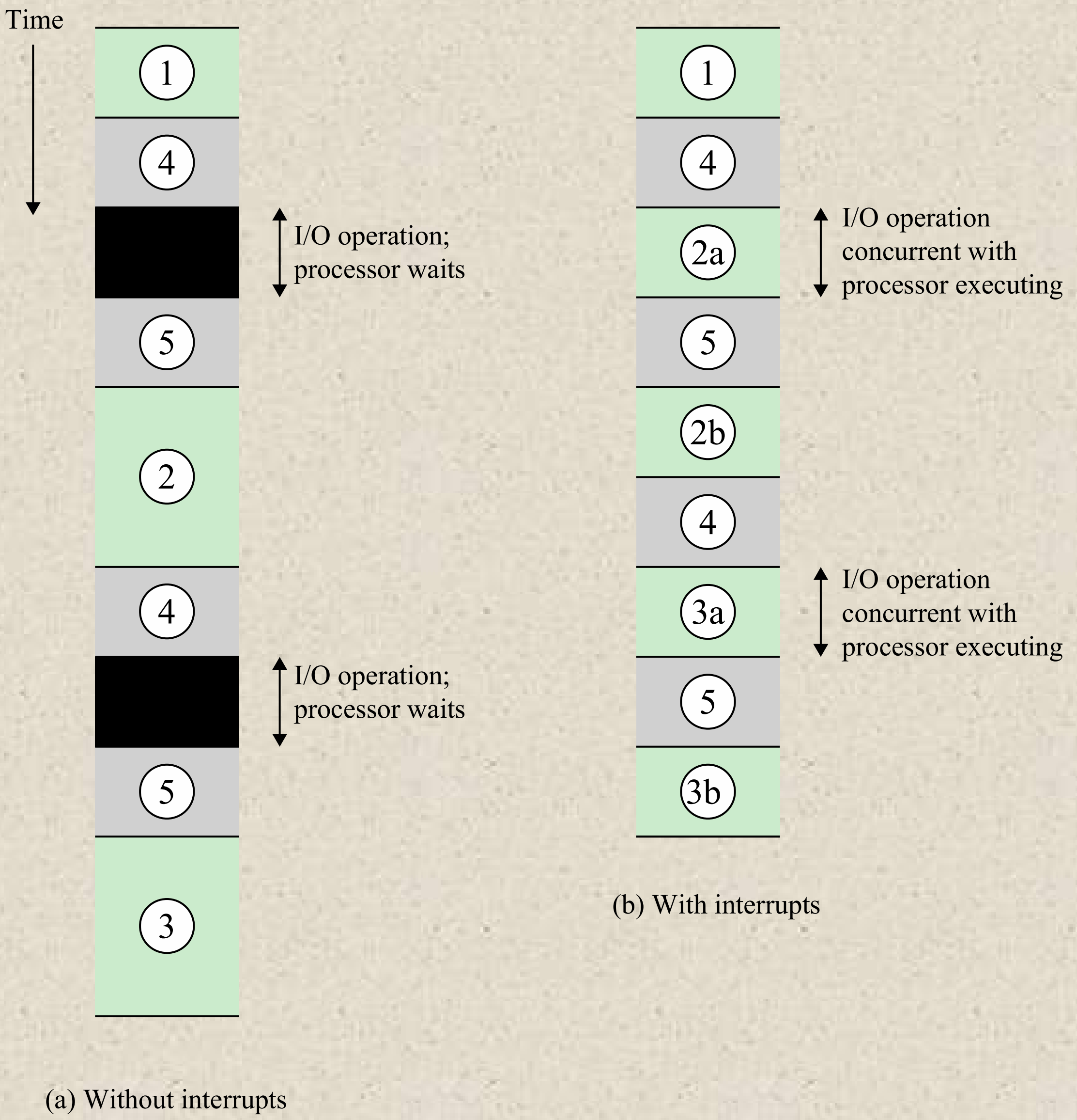

Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2a
I/O operation concurrent with processor executing
5
2b
4
3a
I/O operation concurrent with processor executing
5
3b
(b) With interrupts

Figure 3.10 Program Timing: Short I/O Wait

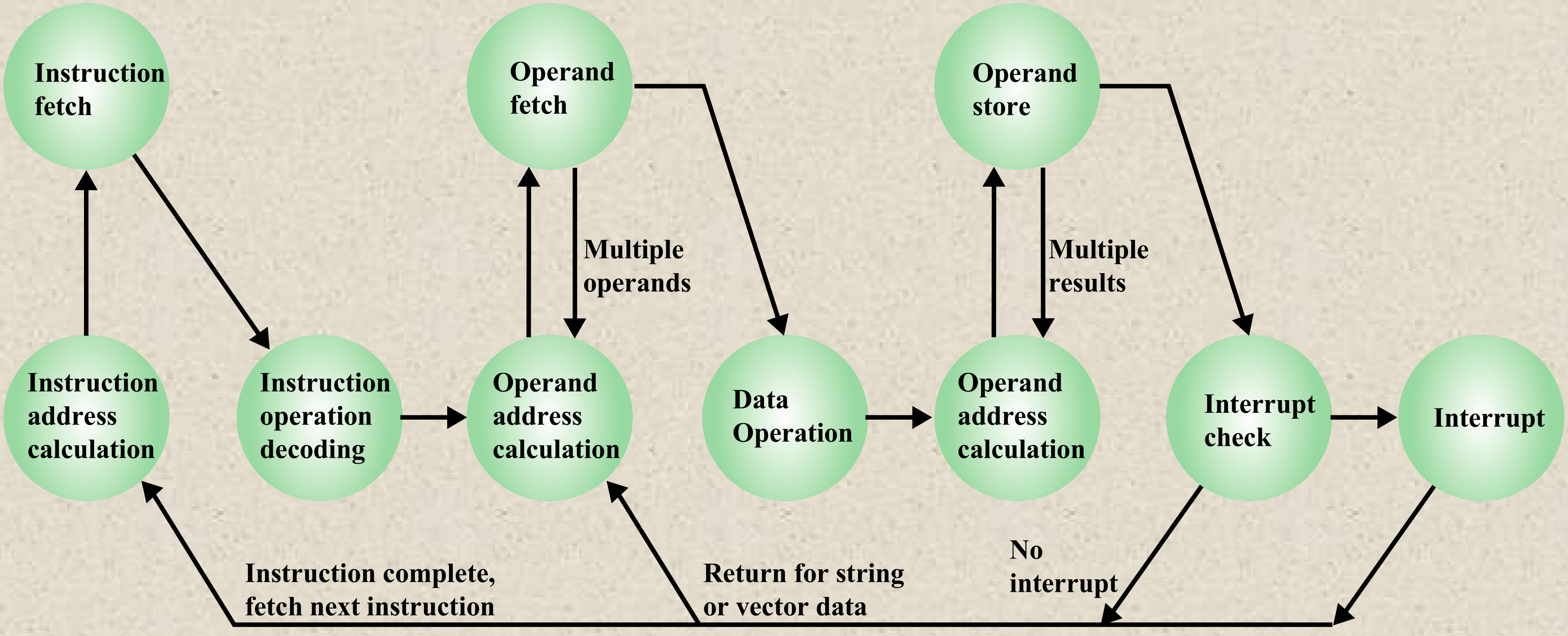

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
No interrupt
Instruction complete, fetch next instruction
Return for string or vector data

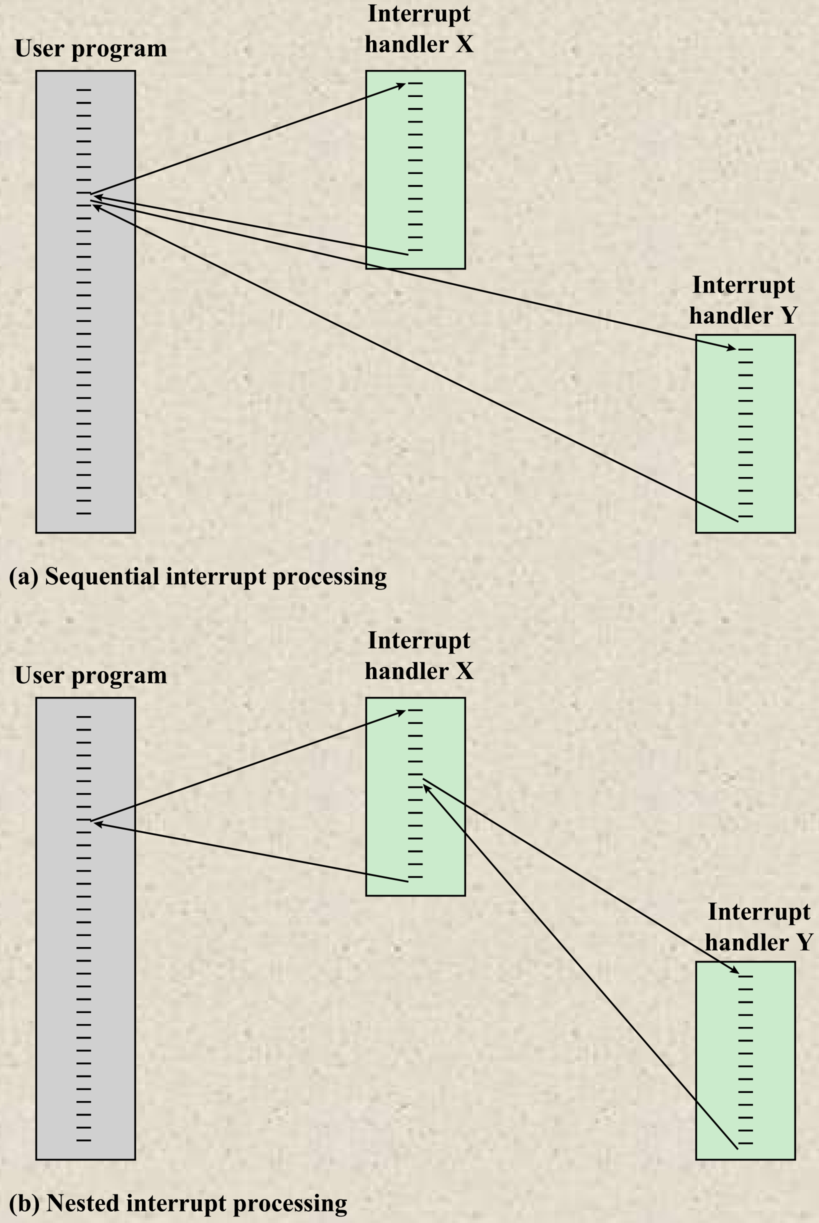
Figure 3.12 Instruction Cycle State Diagram, With Interrupts

Multiple Interrupts

- Possibility of more than one interrupt:
  - Communication line and printer data
- Two approaches to deal with multiple interrupts:
  - **1.Disabled Interrupts:**
    - After occurrence of first interrupt, interrupts are disabled
    - Processor ignores all incoming interrupts
    - Once interrupt is handled all interrupts are enabled
    - Drawback:
      - Not suitable for time critical applications

Multiple Interrupts

- Two approaches to deal with multiple interrupts:
  - **2. Priority Interrupts:**
    - Higher priority interrupts are given preferences
    - Printer=2, communication=4
    - At time t=1, printer interrupt occurred and serviced via Interrupt handler
    - At time t=2 communication interrupt occurred, printer interrupt service routine pushed onto the stack, and communication interrupt is serviced



(a) Sequential interrupt processing

(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

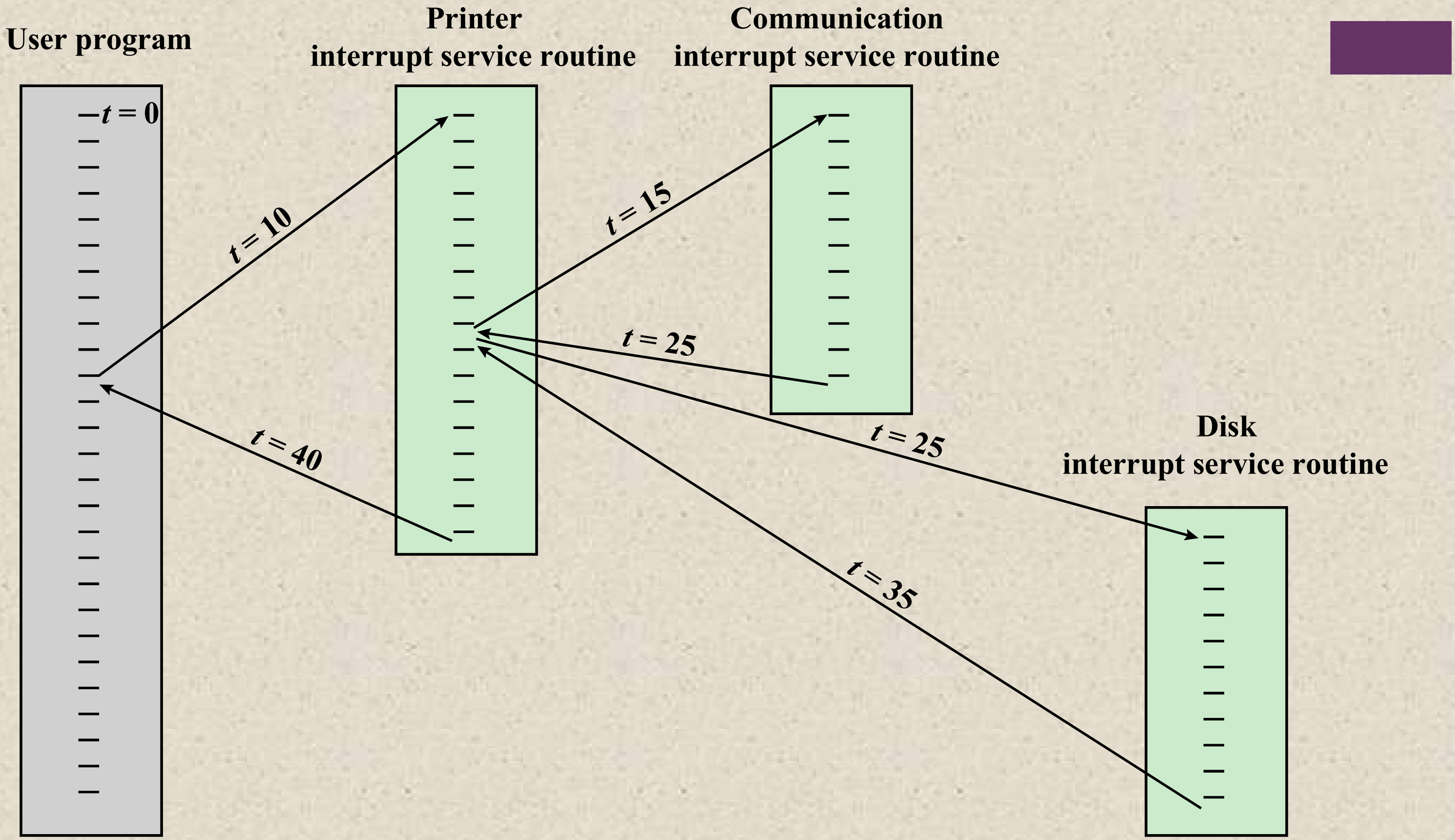


Figure 3.14 Example Time Sequence of Multiple Interrupts

Surprise Quiz!

- How does Computer Communicates with Memory and I/O?
- What is hardwired program, give two examples
- Write Assembly Language code:
  - Which takes two numbers as input
  - Subtracts the larger one from smaller one

Interconnection Structure

- Computer is network of three modules:
  - Processor
  - Memory
  - I/O
- Path for connecting these three modules is called interconnection structure

+

Interconnection Structure

- Memory:
  - N words from 0 to N-1
  - Read/Write
  - Location is specified through address

- I/O:
  - Read/Write
  - I/O module controls more than one device, by assigning each device a unique port address
  - Can Send Interrupt signals to the processor

Interconnection Structure

- Processor:
  - Reads data, processes and writes data

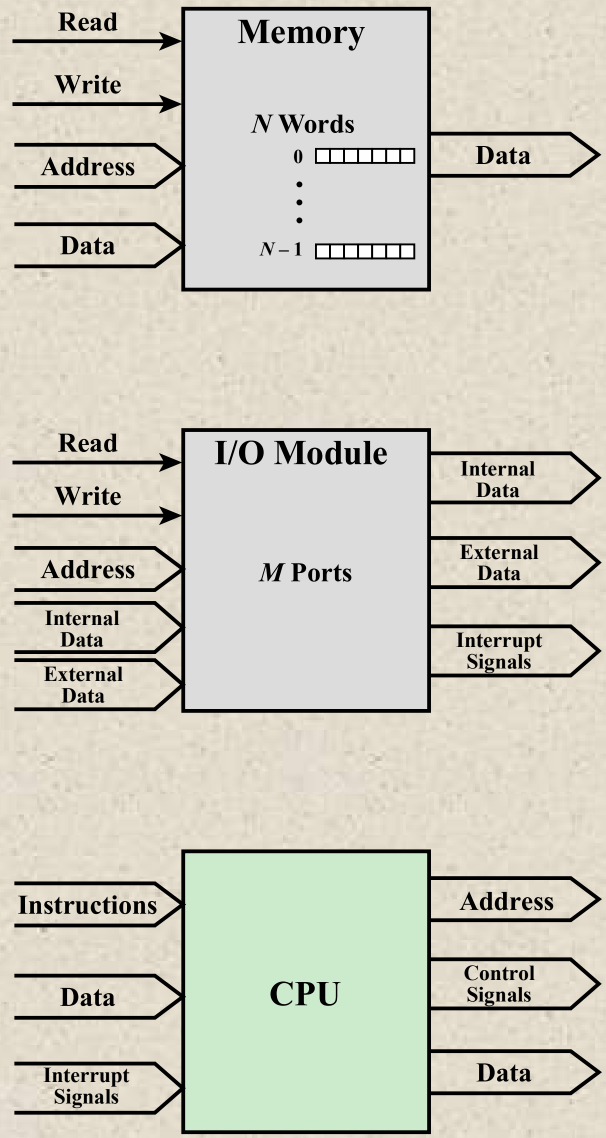


Figure 3.15 Computer Modules

The interconnection structure must support the following types of transfers:

Memory to processor

Processor reads an instruction or a unit of data from memory

Processor to memory

Processor writes a unit of data to memory

I/O to processor

Processor reads data from an I/O device via an I/O module

Processor to I/O

Processor sends data to the I/O device

I/O to or from memory

An I/O module is allowed to exchange data directly with memory without going through the processor using direct memory access

Bus Interconnection

A communication pathway connecting two or more devices

- Key characteristic is that it is a shared transmission medium

Signals transmitted by any one device are available for reception by all other devices attached to the bus

- If two devices transmit during the same time period their signals will overlap and become garbled

Typically consists of multiple communication lines

- Each line is capable of transmitting signals representing binary 1 and binary 0

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy

System bus

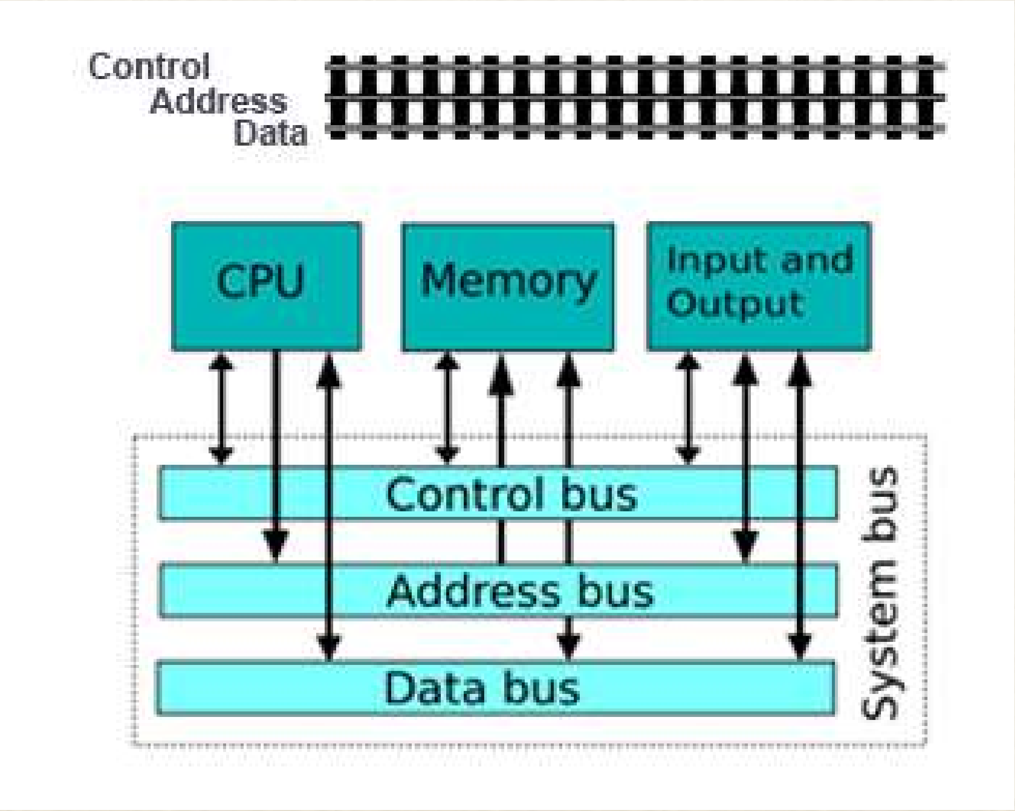
- A bus that connects major computer components (processor, memory, I/O)

The most common computer interconnection structures are based on the use of one or more system buses

Bus Structure

- System Bus:
  - 50 to hundred separate lines
  - Each line dedicated for a particular function from following functional groups:
    - Data lines
    - Control lines
    - Address lines

Bus Structure



Data Bus

- Data lines provide a path for:
  - Moving data among system modules
  - Data line collectively called Data Bus
- Width of Data Bus:
  - Number of lines in data bus
  - Each line carries one or more bit
  - 32, 64, 128 or even more
  - Number of Lines (Width) determine the amount of data which can be transferred.

Data Bus

- Width of Data Bus is key component for system performance:
  - Instruction 64 bits
  - Data bus 32 bits
  - During Instruction fetch memory visited twice.

+

Address Bus

- Address lines:
  - Determine the source/destination of data over data bus
  - Collectively called address Bus
- Processor wishes to read a word of 16, 32 or 64 bit from memory:
  - Puts the address of word on Address Bus
  - Width Determines maximum possible memory capacity of system

Address Bus

- Also used to address I/O Ports/Memory Locations:
  - 8 bit address bus
  - 01111111, higher order bit refers to the memory module 0 and 128 locations in memory module zero
  - 10000000, 1 refers to I/O module

Control Bus

- Data and address lines are shared by all the components
- So their use must be monitored:
  - Monitored through control lines
- Control lines given different commands shown in next slide

Typical control lines include:

- • **Memory write:** causes data on the bus to be written into the addressed location
- • **Memory read:** causes data from the addressed location to be placed on the bus
- • **I/O write:** causes data on the bus to be output to the addressed I/O port
- • **I/O read:** causes data from the addressed I/O port to be placed on the bus
- • **Transfer ACK**: indicates that data have been accepted from or placed on the bus
- • **Bus request:** indicates that a module needs to gain control of the bus
- • **Bus grant:** indicates that a requesting module has been granted control of the bus

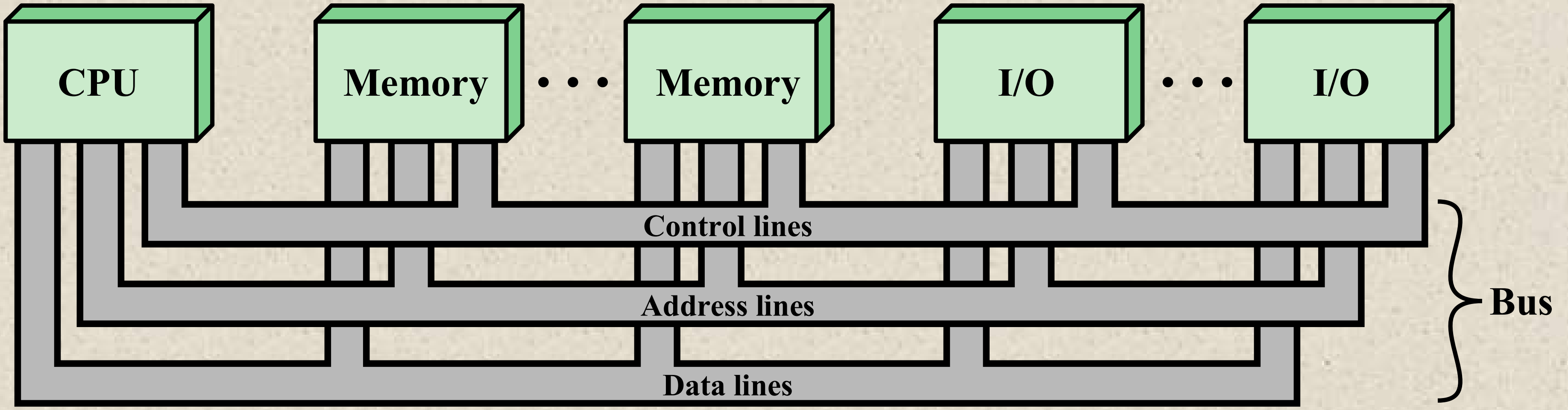


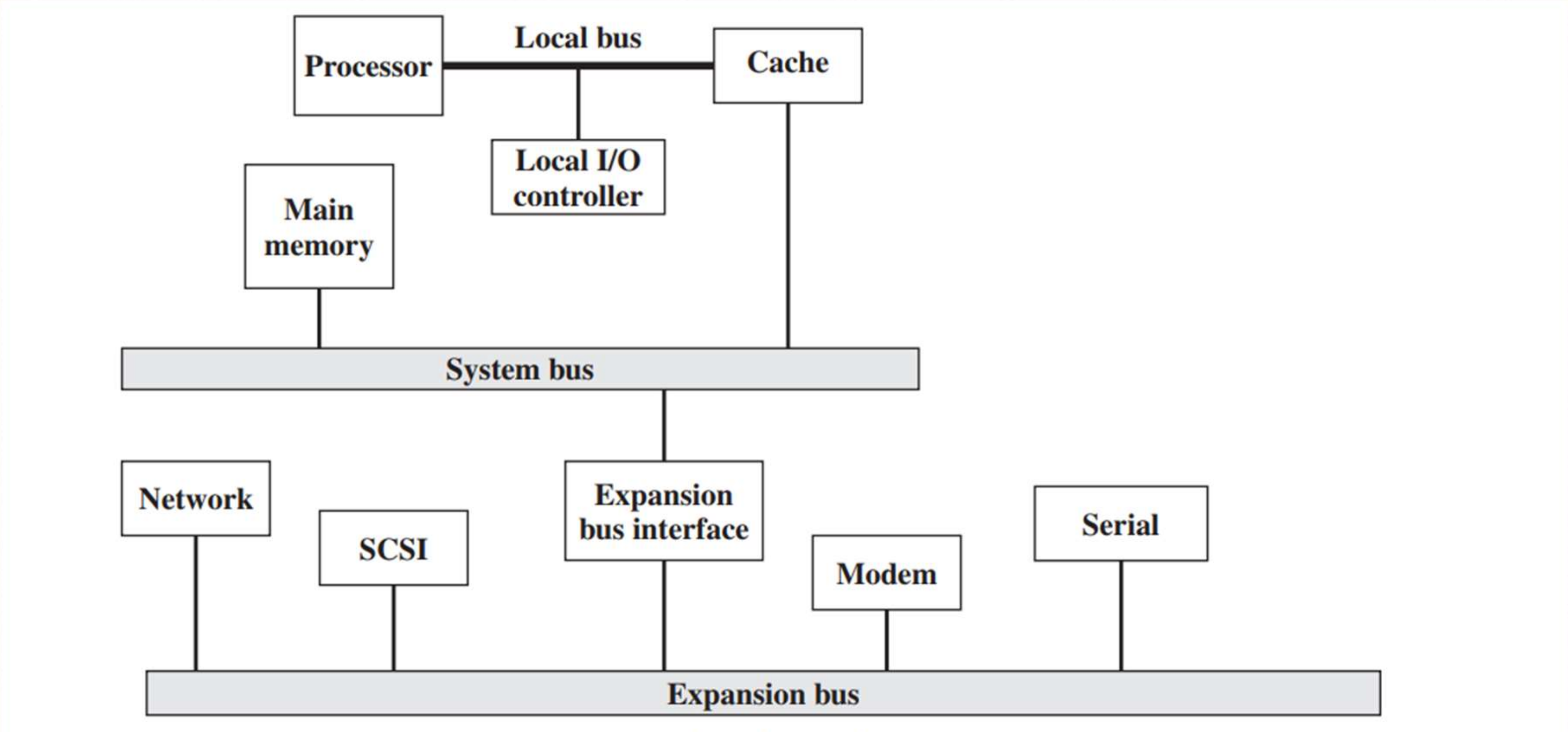
Figure 3.16 Bus Interconnection Scheme

Bus Interconnection Scheme

- Operation of the bus is as follows:
  - Obtain use of bus, bus granted, transfer data
  - If one module wishes to request data from another, Obtain use of bus, bus granted, request data, wait for response

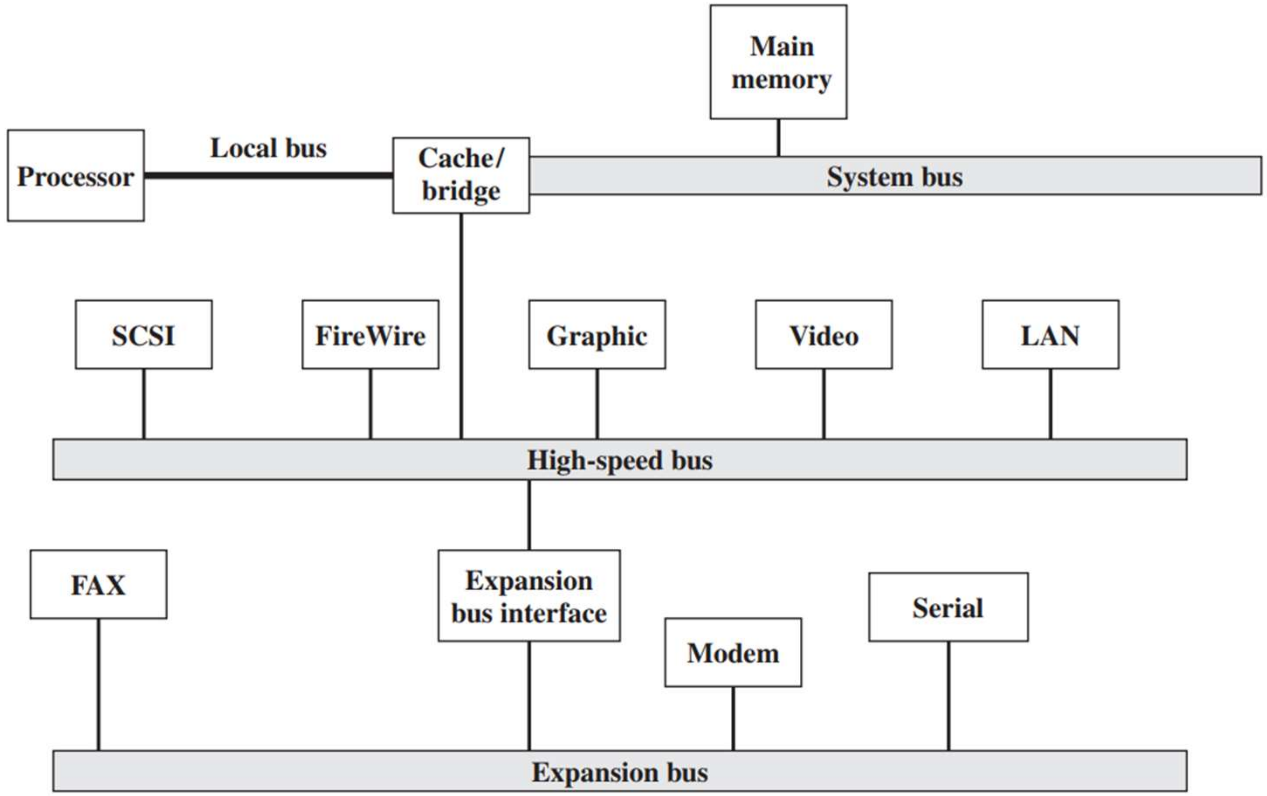
- On chip bus (local bus) / on-board bus - Modern computers tend to have all major components on board and with more elements on same chip as the processor so
  - On-chip bus connecting processor & cache
  - On-board bus may connect processor to main memory and other components

+ Traditional Bus Architecture



(a) Traditional bus architecture

High Performance Bus Architecture



(b) High-performance architecture

Design Elements

- Type (Dedicated, Multiplexed)
- Method of Arbitration (Centralized, Decentralized)
- Timing (Synchronous, Asynchronous)
- Data transfer type (Read, Write, Read-Modify-Write, Read after Write, Block)

Bus Types

Two generic types:

- Dedicated
  - Separate data & address lines
  - Cost increases
  - Address Bus for Address
  - Data Bus for Data

+
Multiplexed

- Multiplexed
  - Shared lines for data and addresses
  - Address valid or data valid control line:
    - First address is copied and its activated
    - After determining the address, its deactivated and data is copied
  - Advantage - fewer lines
  - Disadvantages
    - More complex control
    - Reduction in performance:
      - CPU to I/O in progress, memory will have to wait

+

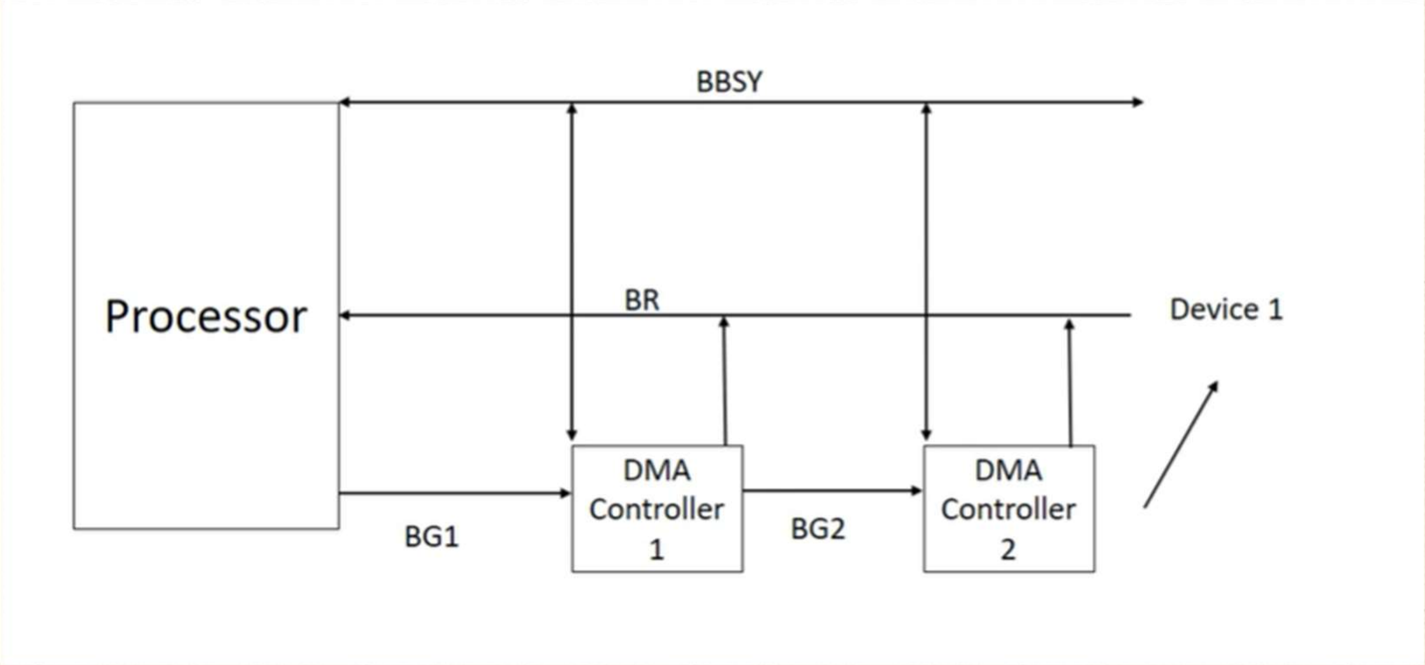
Bus Arbitration

- More than one module controlling the bus
  - e.g. CPU and DMA controller
- Only one module may control bus at one time
  - Read and Write
- **Bus Arbitration** refers to the process by which the current bus master accesses and then leaves the control of the bus and passes it to another bus requesting processor unit.
- Arbitration may be centralised or distributed

Centralised or Distributed Arbitration

- Centralised
  - Single hardware device controlling bus access
    - Bus Controller or Arbiter
  - May be part of CPU or a separate device
- Distributed
  - In which every device takes part in choosing the new bus master.
  - Sometimes few devices may be ignored always, because of low priority

+


BBSY
Processor
BR
Device 1
DMA Controller 1
DMA Controller 2
BG1
BG2

Timing

- Refers to the way in which events are co-ordinated on bus – Synchronous/ Asynchronous

- Synchronous
  - Events determined by clock signals
  - Control Bus includes clock line
  - A single 1-0 is a bus cycle!!! Clock cycle
  - All devices can read clock line

  - Asynchronous
  - → The occurrence of one event on a bus follows and depends on the occurrence of a previous event