MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT2100 Semester 2 2018

Lecture 1:
Computer System Overview

(Reading: Stallings, Chapter 1)

Week 1

GROUP OF EIGHT AUSTRALIA

CONTEXT

- An Operating System…
  - Manages hardware resources on behalf of the user
  - Provides services to other programs
  - Makes the computer more useful
- Designing an operating system is like **playing a game**
  - Games have rules
  - The hardware sets the rules
  - Let’s try to figure out what some of the rules are…

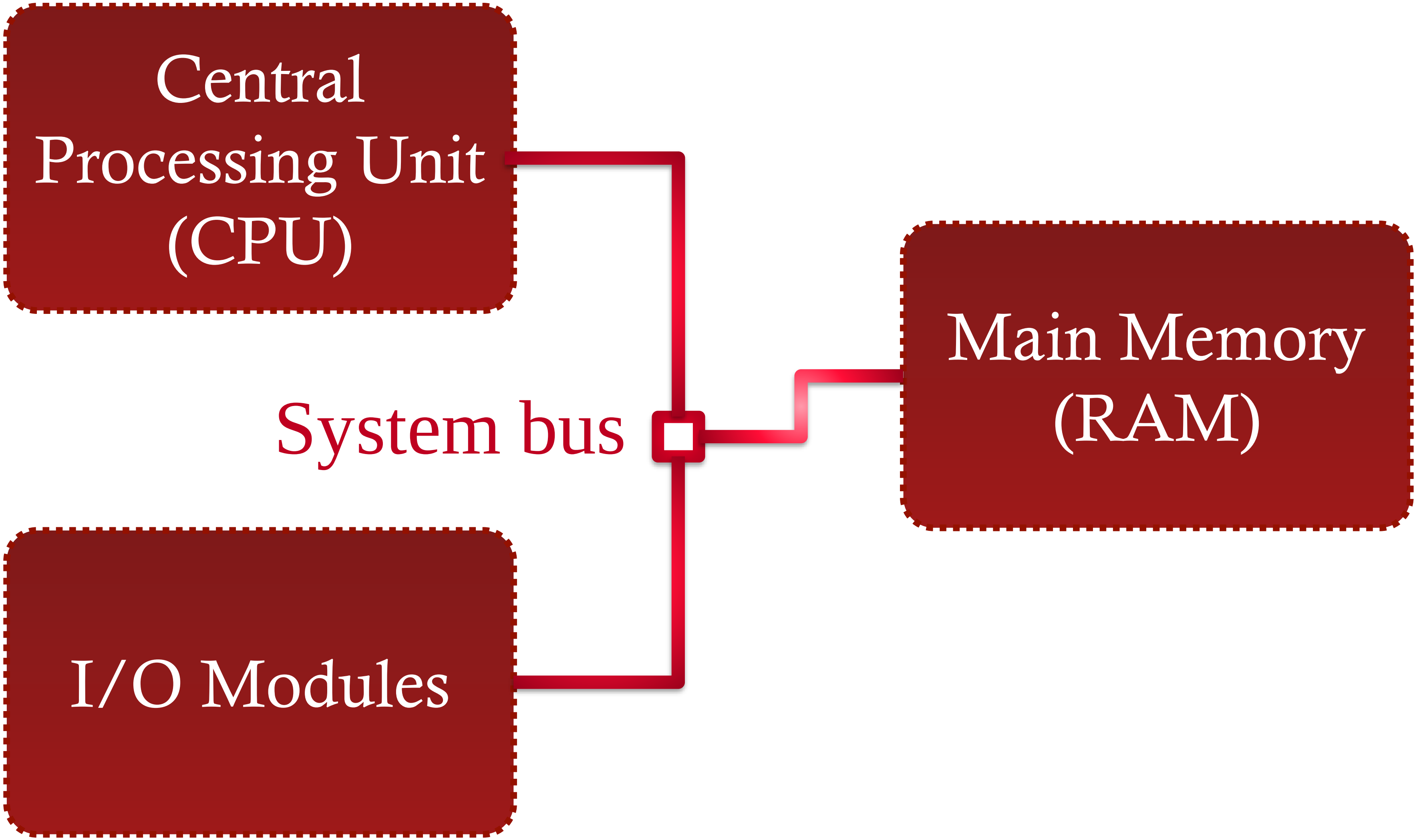
Lecture 1: Learning Outcomes

- Upon the completion of this lecture, you should be able to:
  - Understand the building blocks of a computer system
  - Explain the steps taken by a processor to execute a program instruction
  - Discuss the concept of interrupts and how and why a processor uses interrupts
  - Discuss the typical computer memory hierarchy

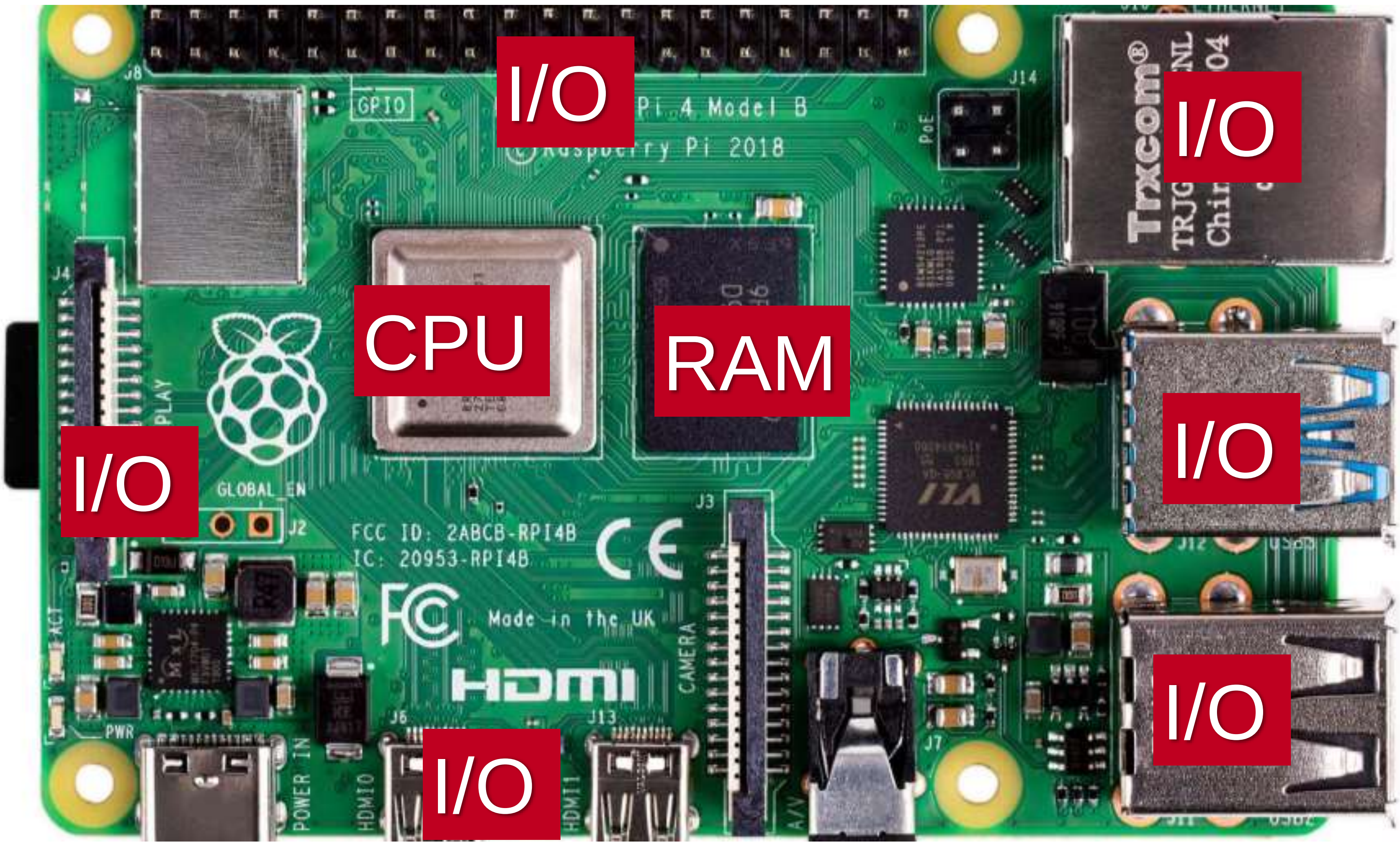
MONASH University

What are the basic elements of a computer system?

Computer System: Basic Elements



Raspberry Pi 4 (just an example)



MONASH University

Central Processing Unit (CPU)

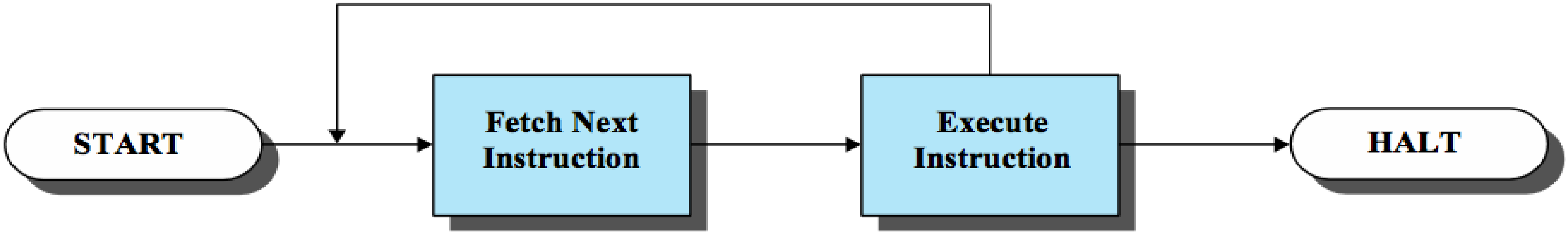
What is a CPU?

Central Processing Unit (CPU)

- A *program interpreter* built in hardware.
  - What kind of programs can it run?
    - Machine code
- RULE 1 → **The CPU can only run instructions written in its own machine code**
  - C: you must **compile** your entire C program into machine code before it can be loaded into the CPU
    - Entire program runs in the native language of CPU
  - Python: a special **interpreter** program reads your instructions and does everything on your behalf
    - Convenient to code; *much* slower to execute

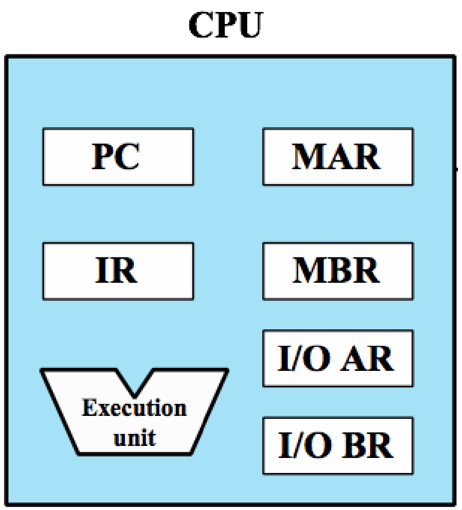
Basic Instruction Cycle

- The CPU **fetches** each instruction from main memory, **decodes** and **executes** it.
- Then fetch the next instruction and repeat.



What happens inside the CPU?

- CPU contains a number of **registers**
  - Very fast working memory for various values
  - Each register is like a **variable** built in hardware rather than software.
- PC: program counter
  - Stores **memory address** of next instruction in program
- IR: instruction register
  - Stores the current machine code **instruction** being worked on
- Assorted registers for working with data.



MONASH University

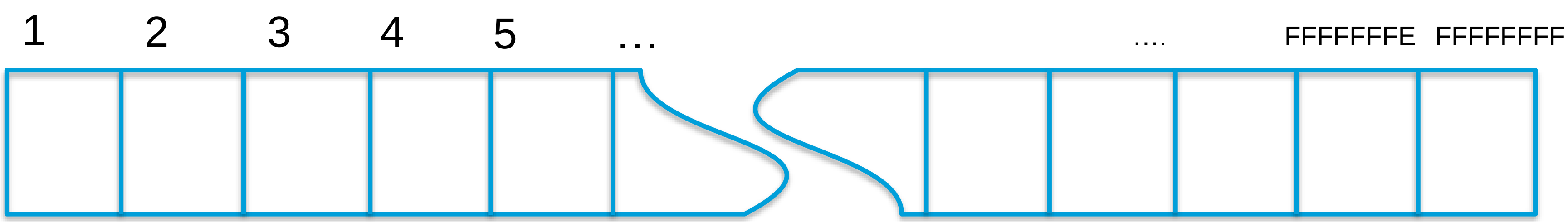
What is main memory?

Main Memory (RAM)

Main Memory

- Stores data and programs that have been loaded for execution
- **RULE 2 →** Program must be loaded into main memory before it can be run
  - Otherwise the CPU can't fetch the instructions
- Volatile
  - Contents of the memory is lost when the computer is shut down
- A.k.a. real memory or primary memory or random access memory

How can memory be represented?

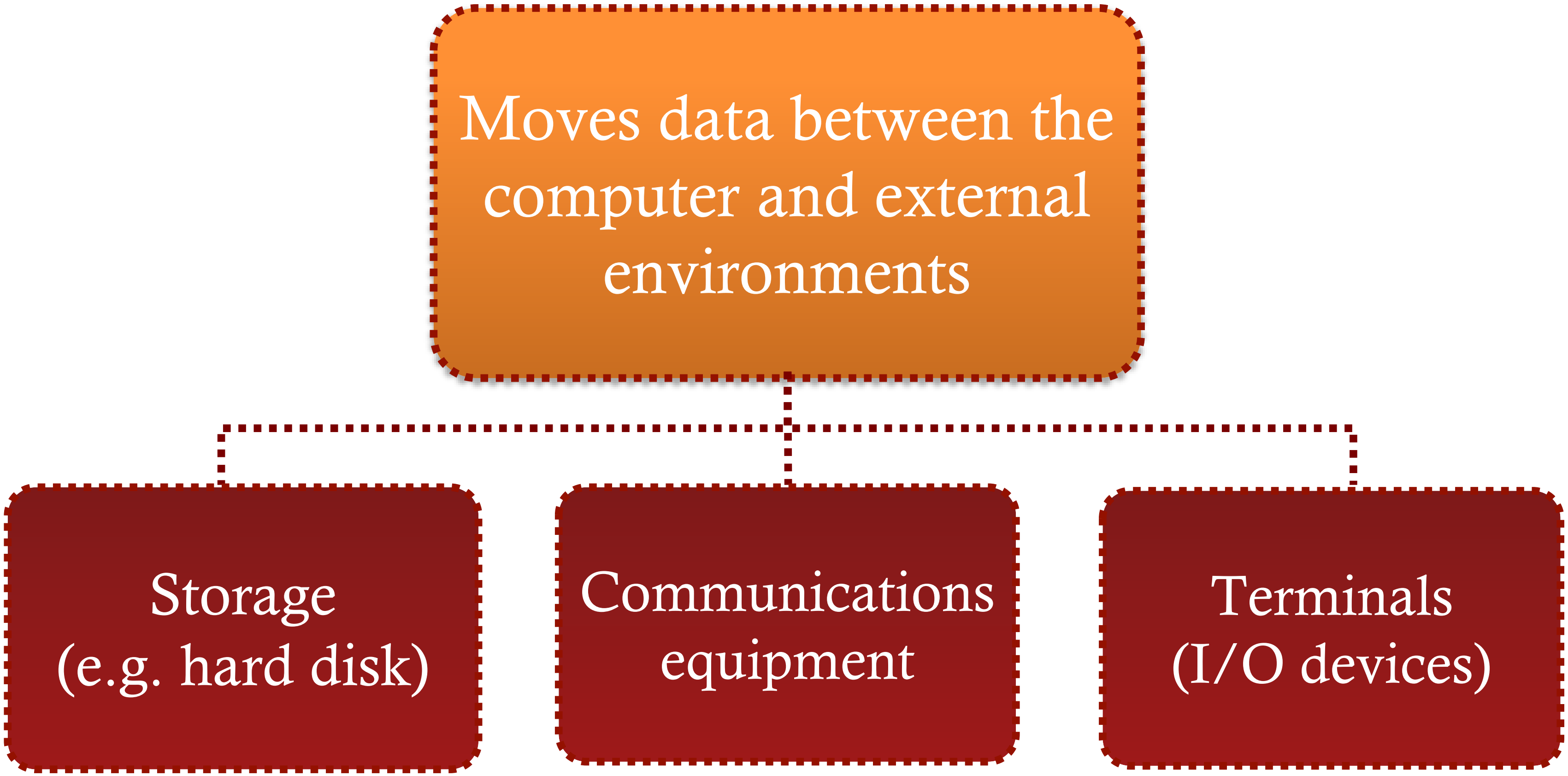


- ❑ You can think of main memory as a long list of bytes
  - Each byte element stores an 8-bit value (e.g. 0-255)
  - Each byte element is *indexed* by a **memory address**
- ❑ Examples of data types (our 32-bit VM environment)
  - a single character code value (**char**) requires 1 byte.
  - An integer (**int**) is made up of 4 bytes put together
  - (An **instruction** is also 4 bytes → 32 bits)
- ❑ **RULE 3 →** To access something in memory, you need to know its starting address, and how many bytes you need.

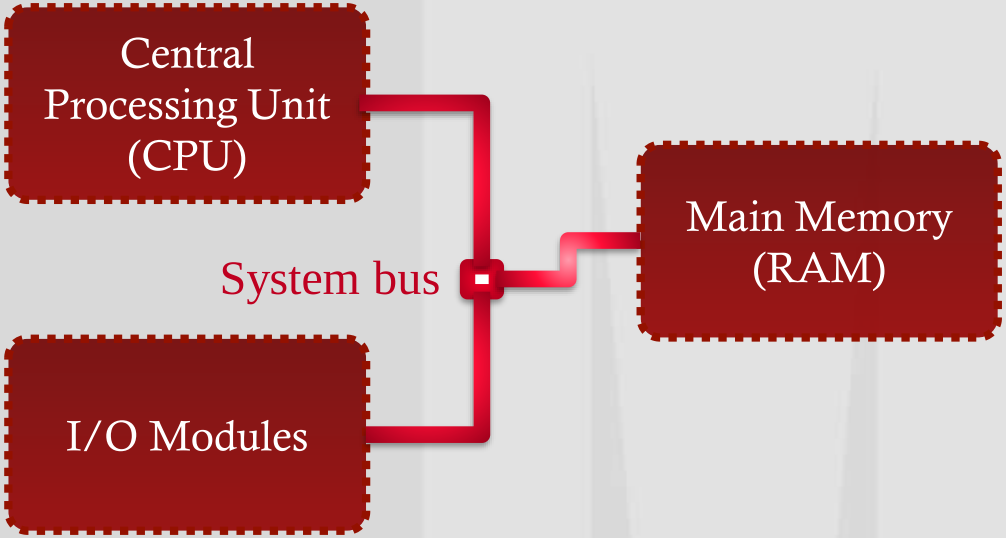
What are I/O Modules?

I/O Modules

I/O Modules


Moves data between the computer and external environments
Storage (e.g. hard disk)
Communications equipment
Terminals (I/O devices)

What about the system bus?



System Bus

- Provides for means of communication among processors, main memory, and I/O modules
- Comprises of:
  - Address bus
  - Data bus
  - Control bus

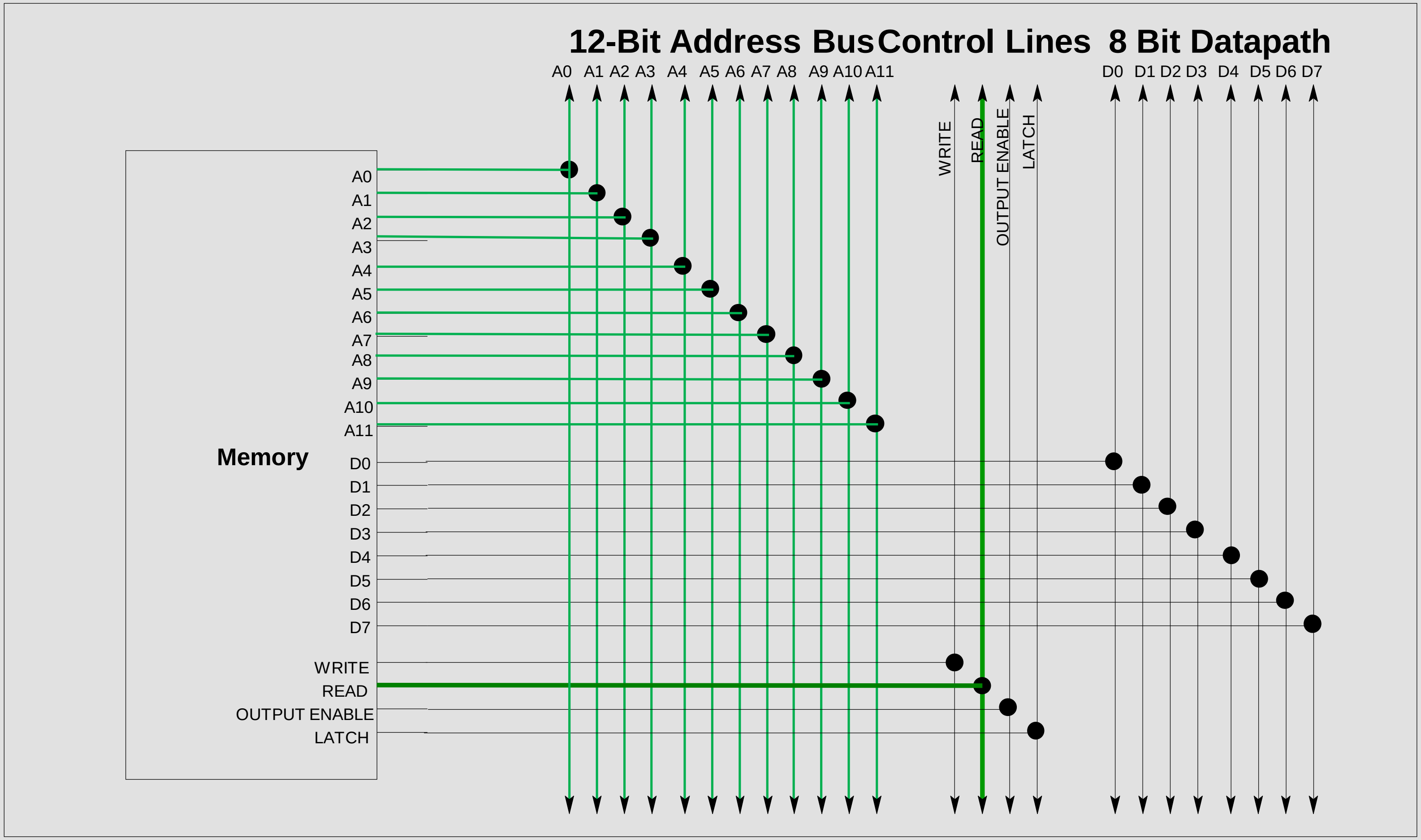
MONASH University

A WORMHOLE TO FIT3159 HAS OPENED…

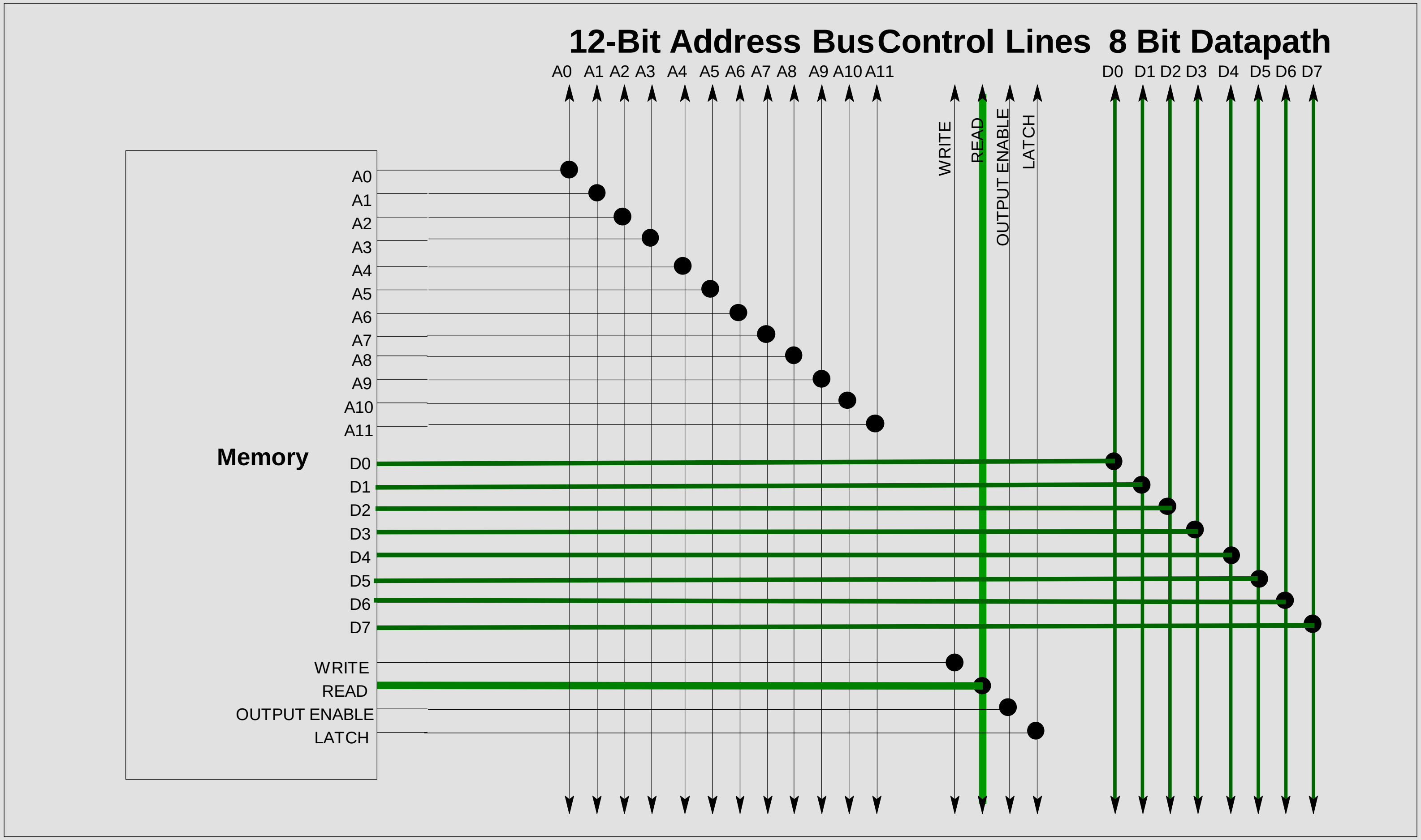
COMPUTER ARCHITECTURE

GROUP OF EIGHT AUSTRALIA

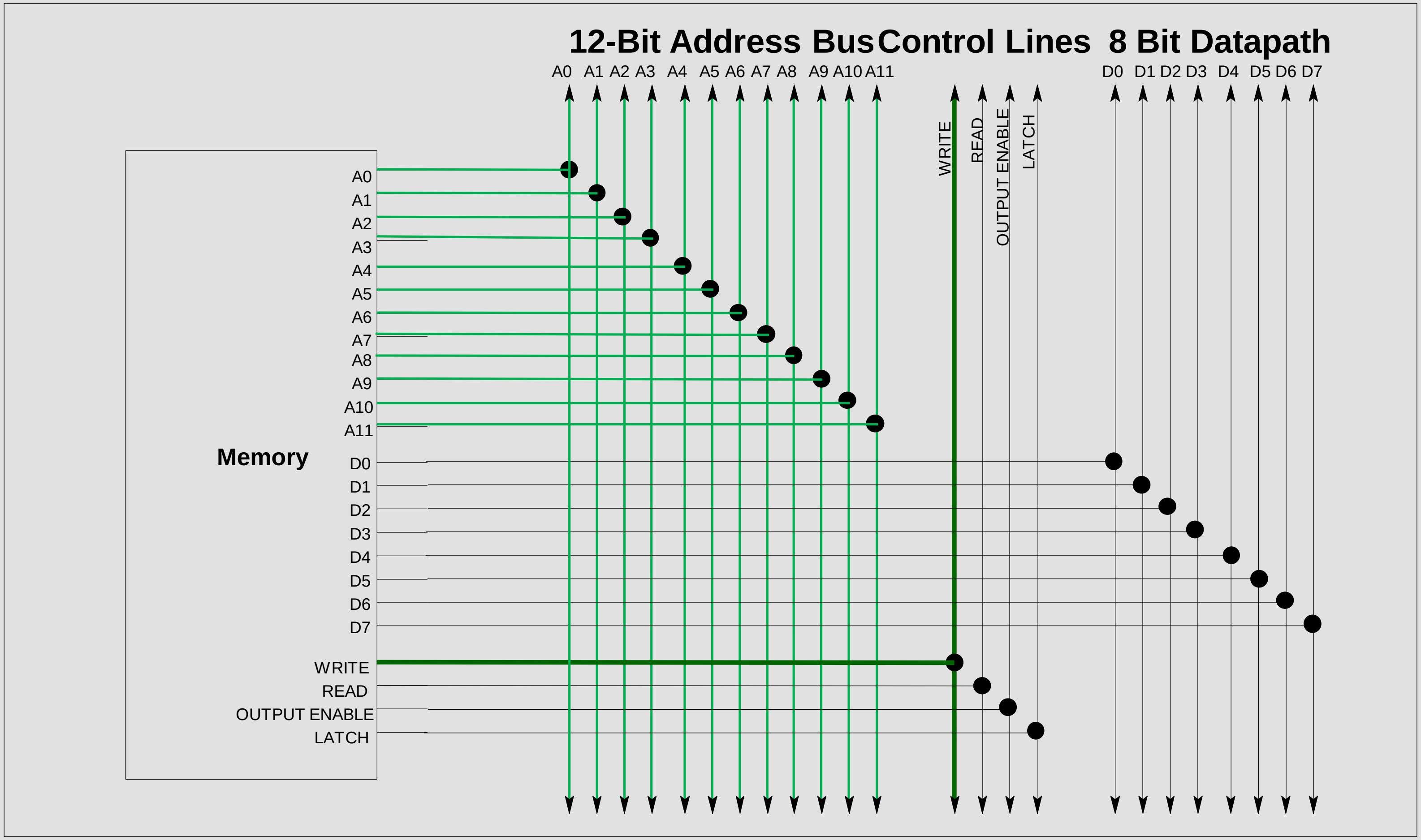
Reading: Assert Read Address and Control



Reading From Memory: Obtain Read Data



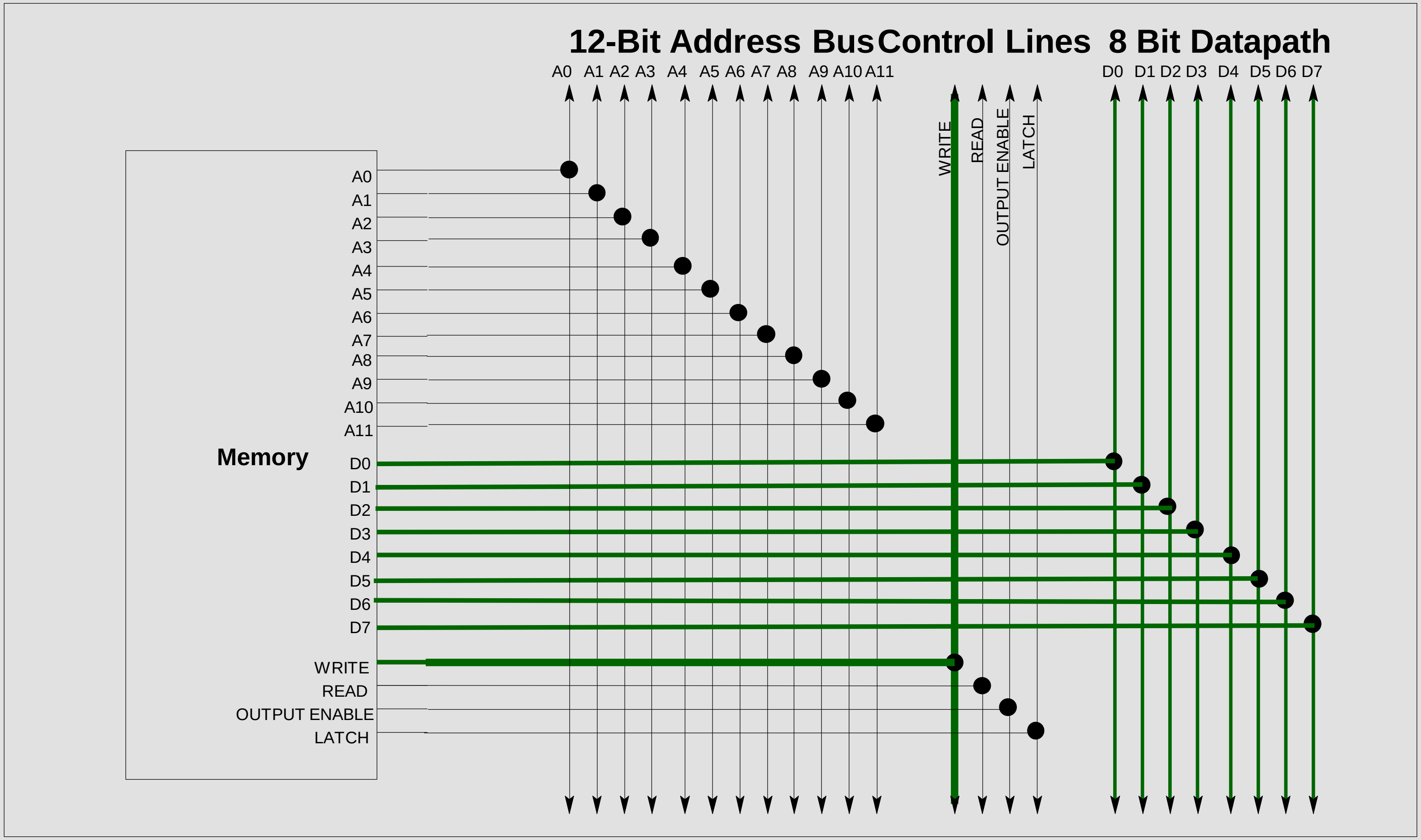
Writing: Assert Write Address and Control



MONASH University
Information Technology

www.infotech.monash.edu

Writing: Assert Write Data



MONASH University
Information Technology

www.infotech.monash.edu

HEY, THAT WAS WEIRD

WE ARE NOW BACK IN FIT2100

How does a program get executed by a processor?

Example: A "Hypothetical" Processor (Stallings textbook)

(a) **Instruction format**

(b) **Integer format**

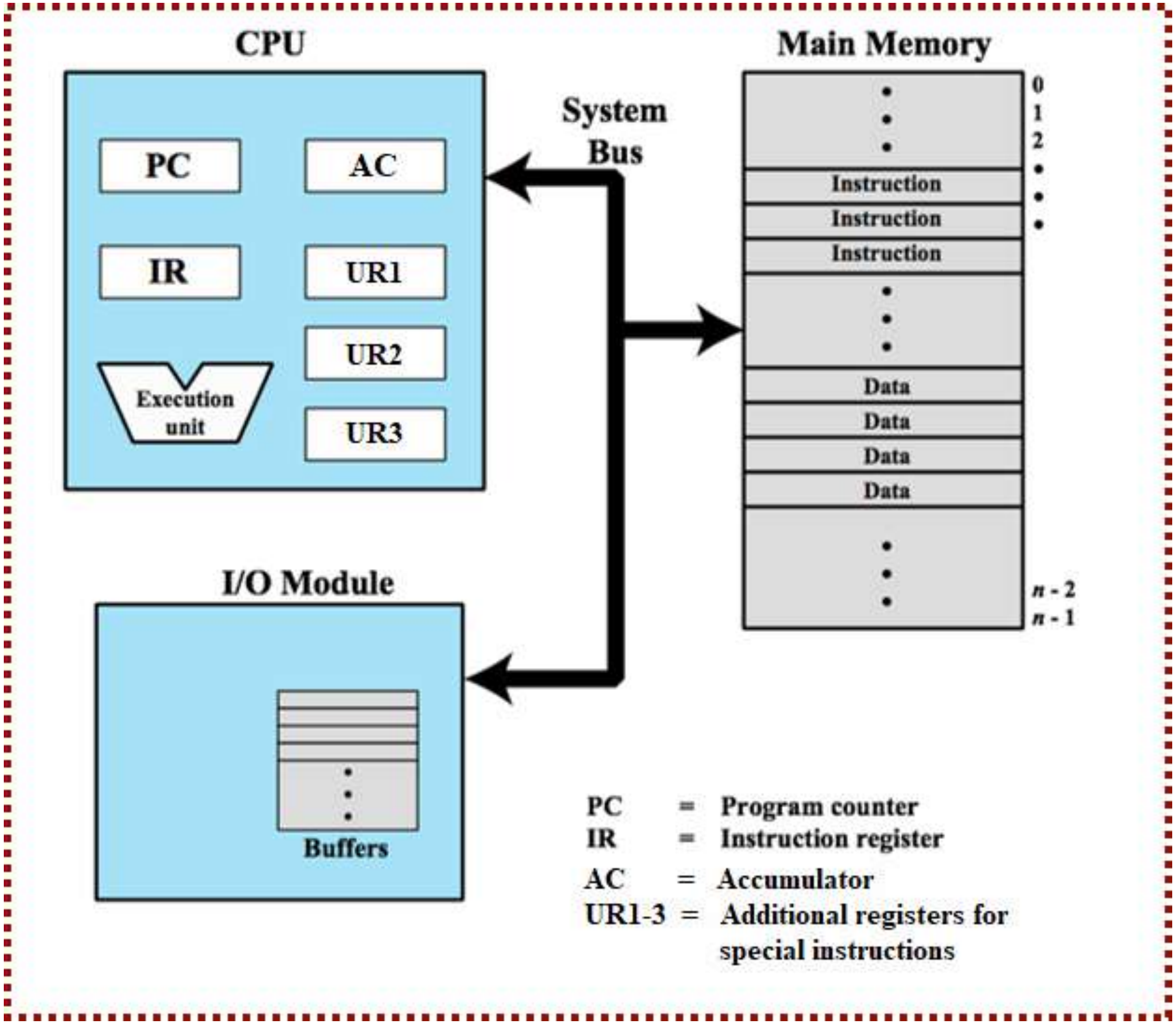
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(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**

Computer Components: Top-level View



Instruction Execution: Two Steps

- A program consists of a set of instructions stored in memory

processor reads (*fetches*) instructions from memory

processor *executes* each instruction

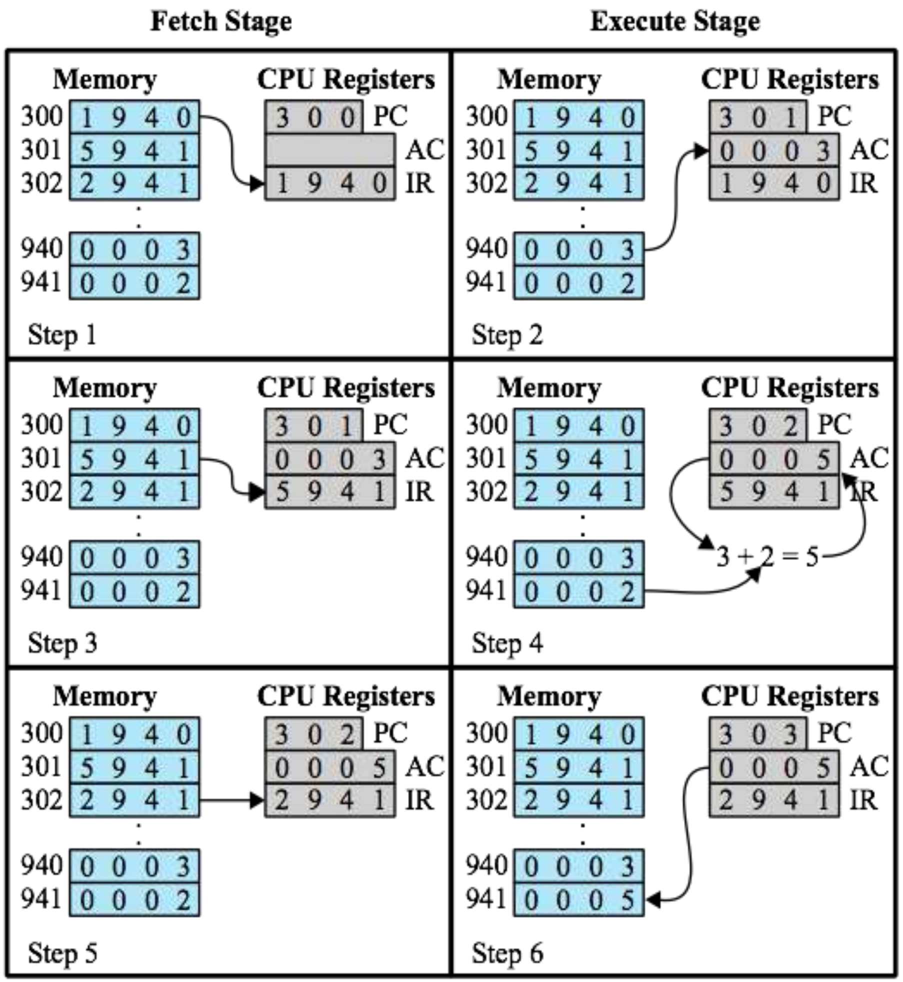
Fetch Stage

- The processor fetches the instruction from memory
- Program Counter (PC) holds address of the instruction to be fetched next
- PC is incremented after each fetch in order to fetch the next instruction

Execute Stage

- Fetched instructions are loaded into Instruction Register (IR)
- The processor interprets the instruction and performs the required action
- Four categories of actions:
  - Processor-memory (load and store instructions)
  - Processor-I/O (system calls)
  - Data processing (arithmetic and logical operations)
  - Control (flow of execution)

Example: Program Execution



Opcode		Action
0001	**1**	Load to AC
0010	**2**	Store in AC
0101	**5**	Add to AC from RAM

What are interrupts?
Why a processor uses interrupts

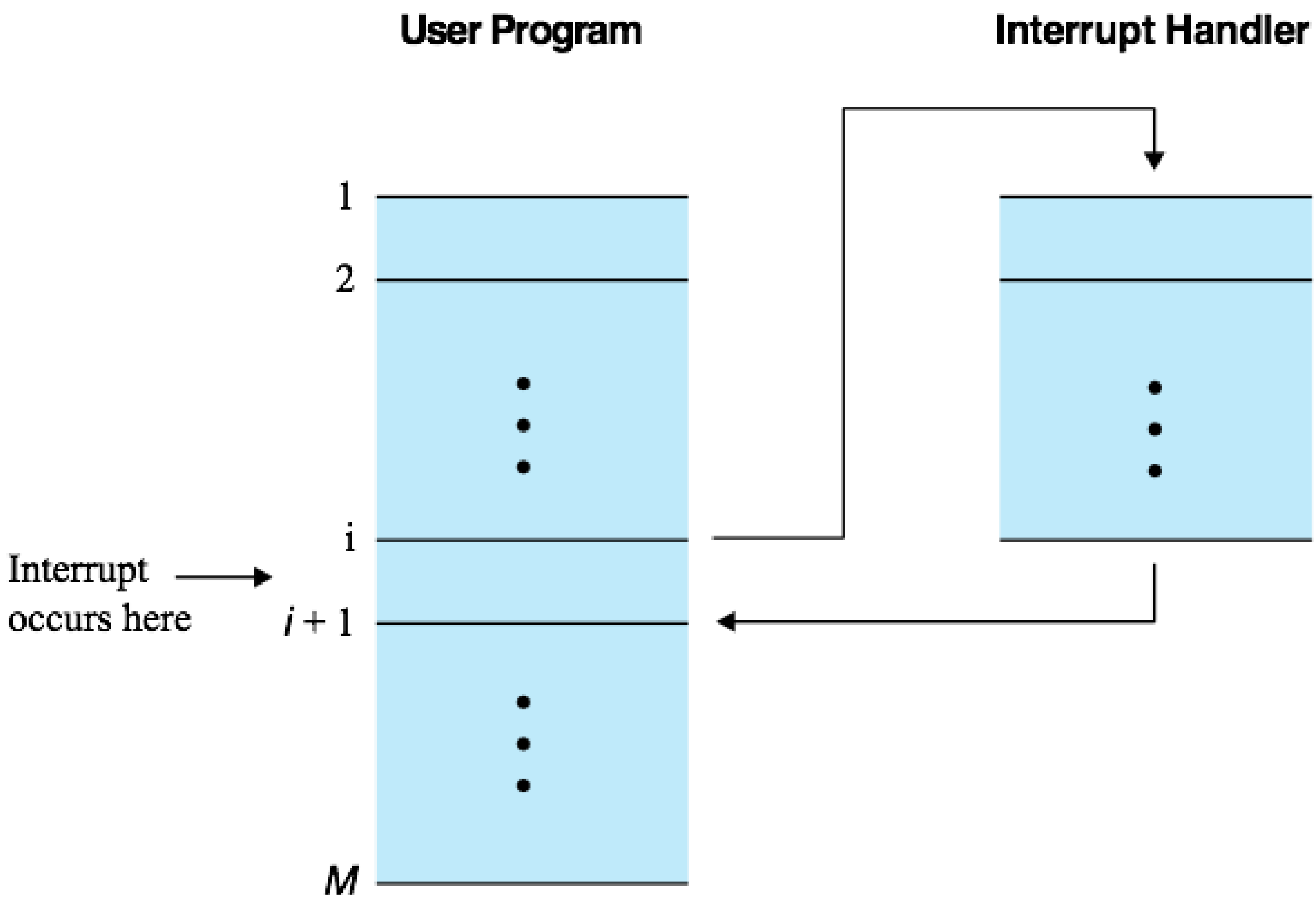
Interrupts

- Mechanisms where the normal sequencing of the processor can be interrupted
- Provided to improve processor utilisation
- Most I/O devices are slower than the processor
  - Processor must pause to wait for the I/O device to complete its operation
  - Wasteful use of the processor

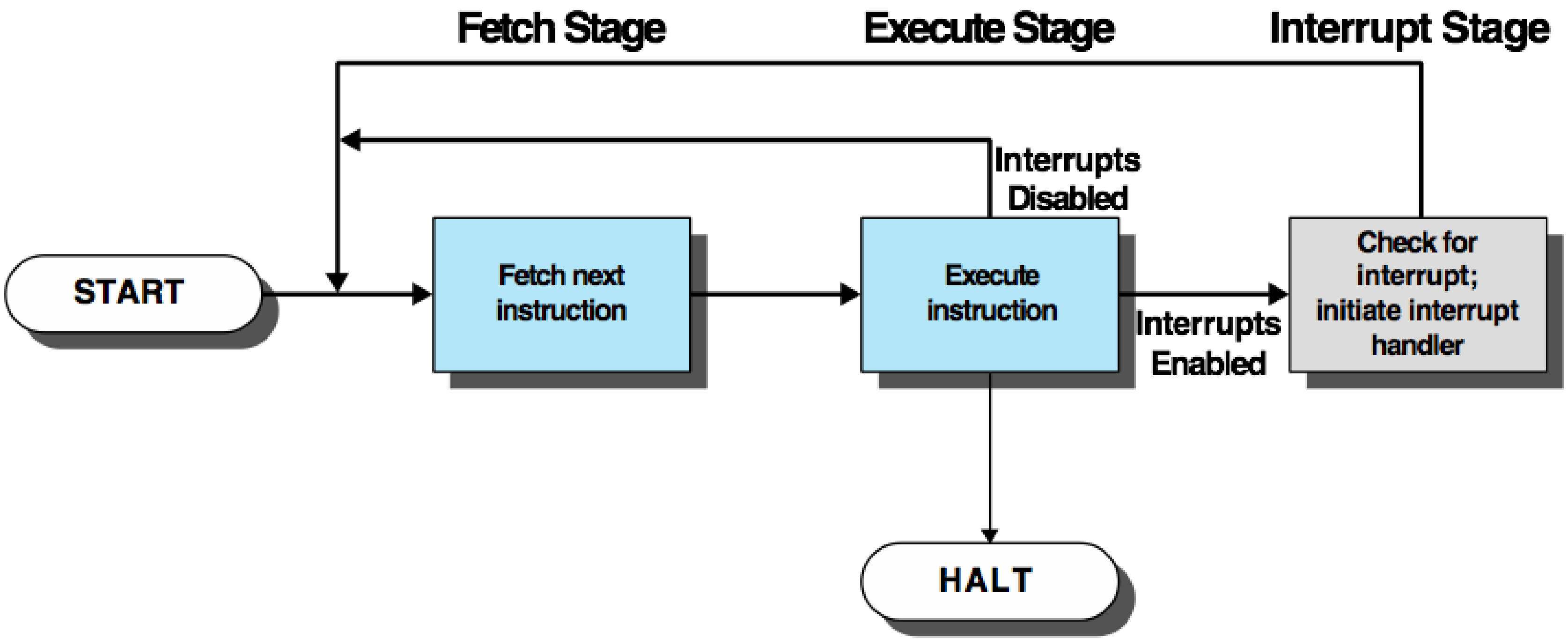
Common Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, attempt to execute an illegal machine instruction, and 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	Generated by a failure, such as power failure or data corruption.

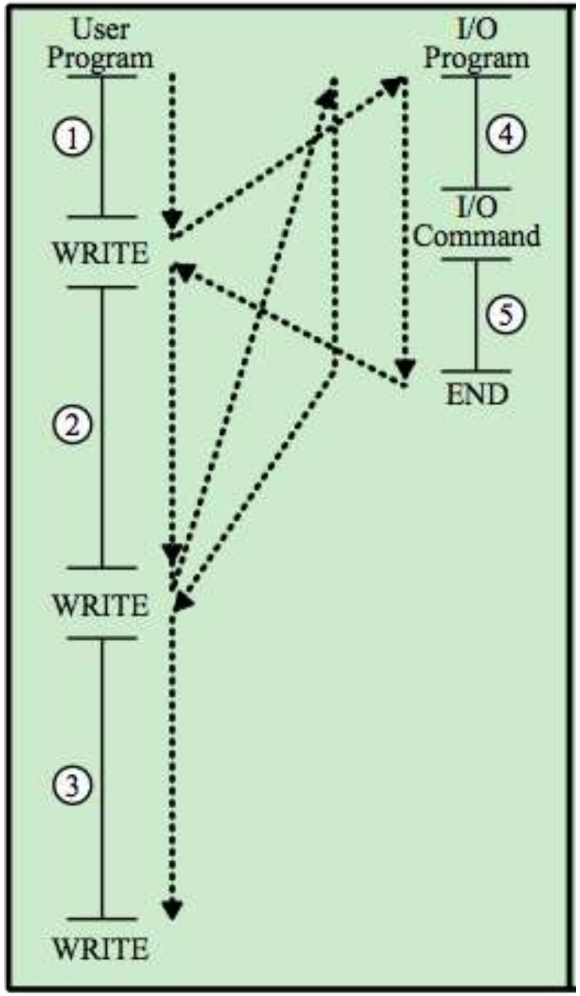
Interrupts: Transfer of Control



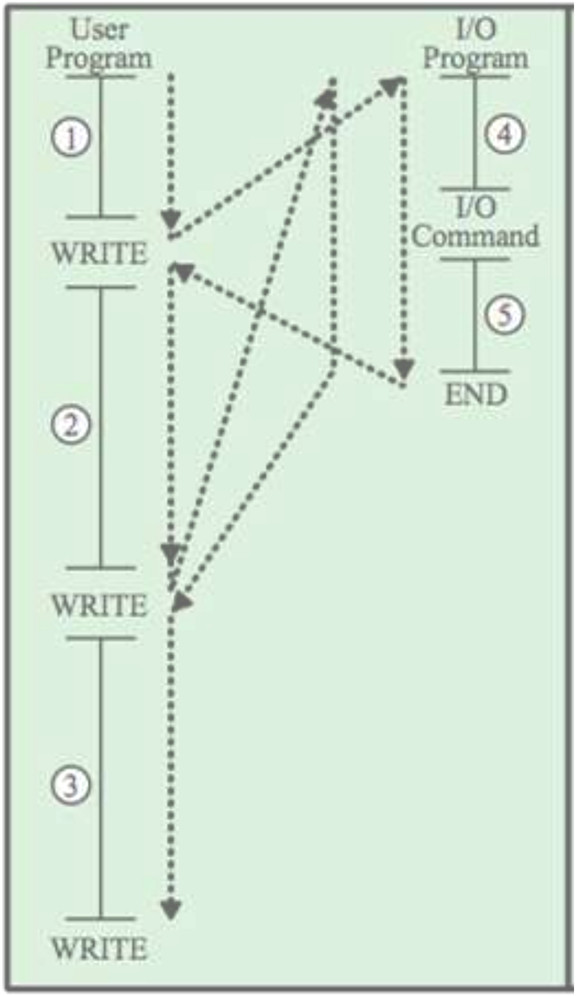
Instruction Cycle: With Interrupts

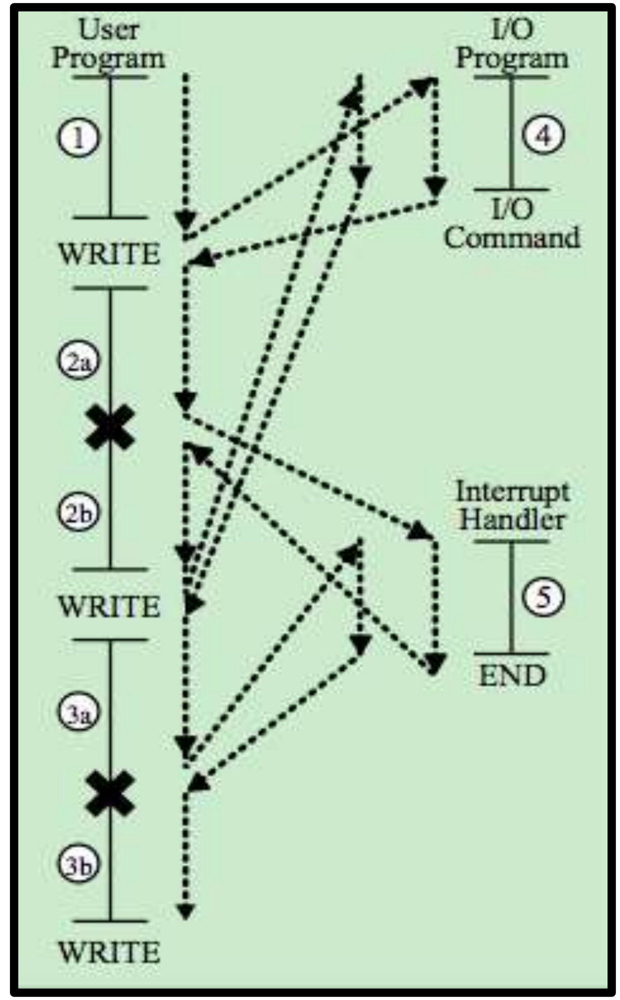


Flow of Control: Without Interrupts

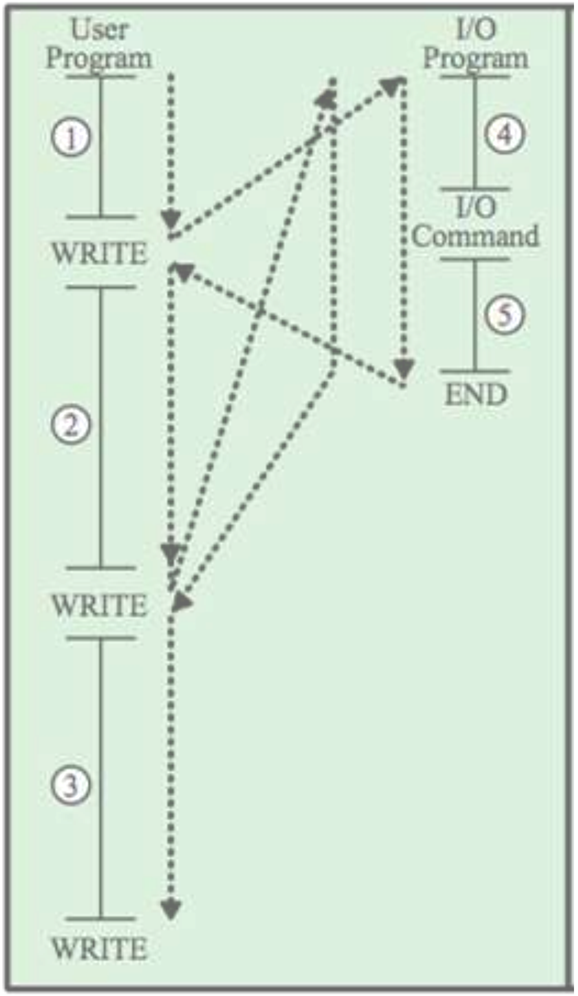


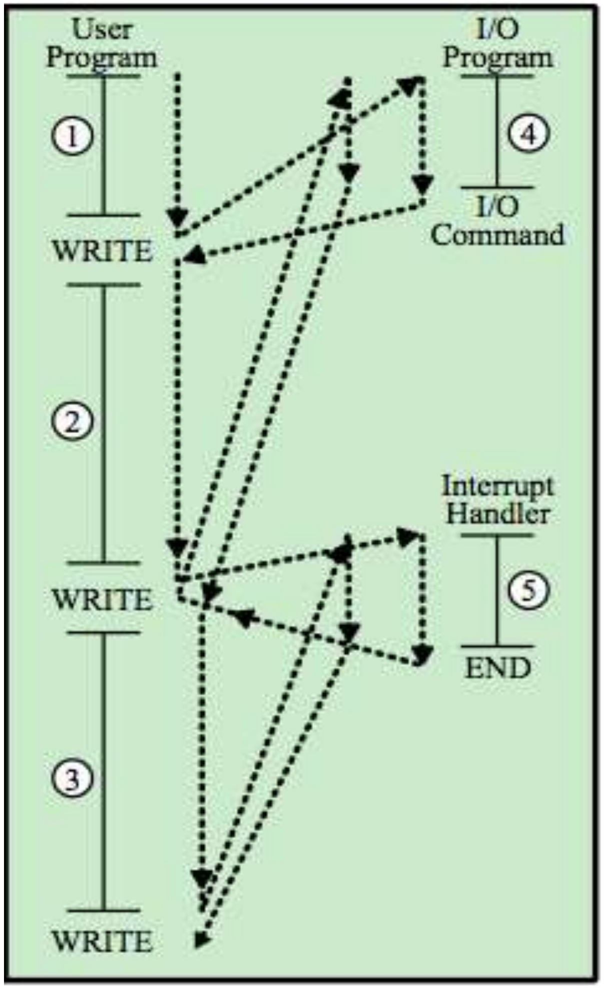
Flow of Control: Short I/O Wait





Flow of Control: Long I/O Wait





MONASH University

We now know the rules of RAM.
What are the design constraints?

Computer Memory

- Major constraints in a computer memory:
  - Amount (capacity)
  - Speed (access time)
  - Expense (cost)
- Memory must be able to keep up with the processor
- Cost of memory must be reasonable in relationship to other components

Memory Relationships

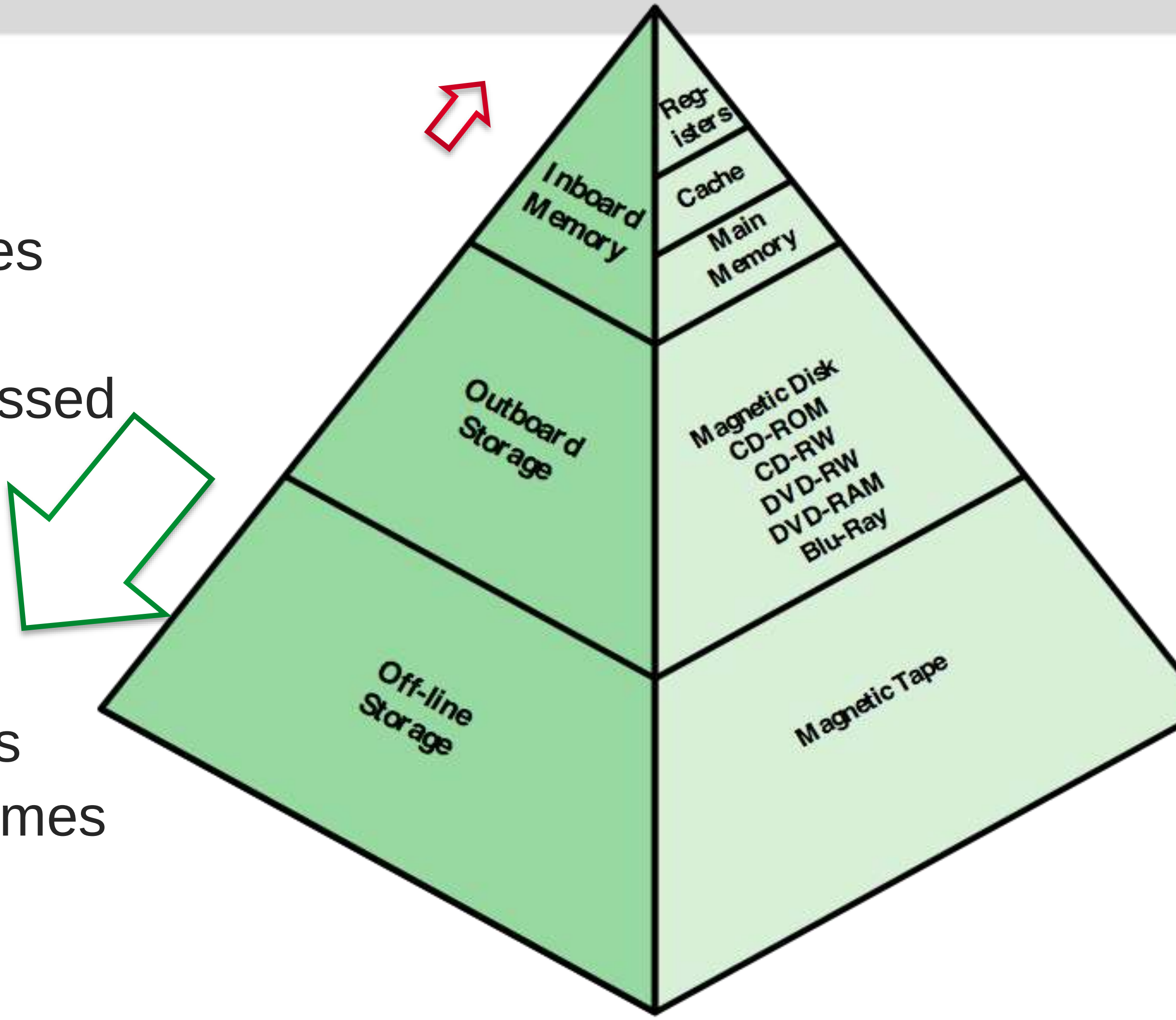
Faster access time = greater cost per bit

Greater capacity = smaller cost per bit

Greater capacity = slower access speed

Memory Hierarchy

- **Closer to the CPU:**
  - Fast access times
  - Small capacity
  - Frequently-accessed data
- **Moving down the hierarchy:**
  - Cheaper per bit
  - Bigger capacities
  - Slower access times
  - Less frequently accessed



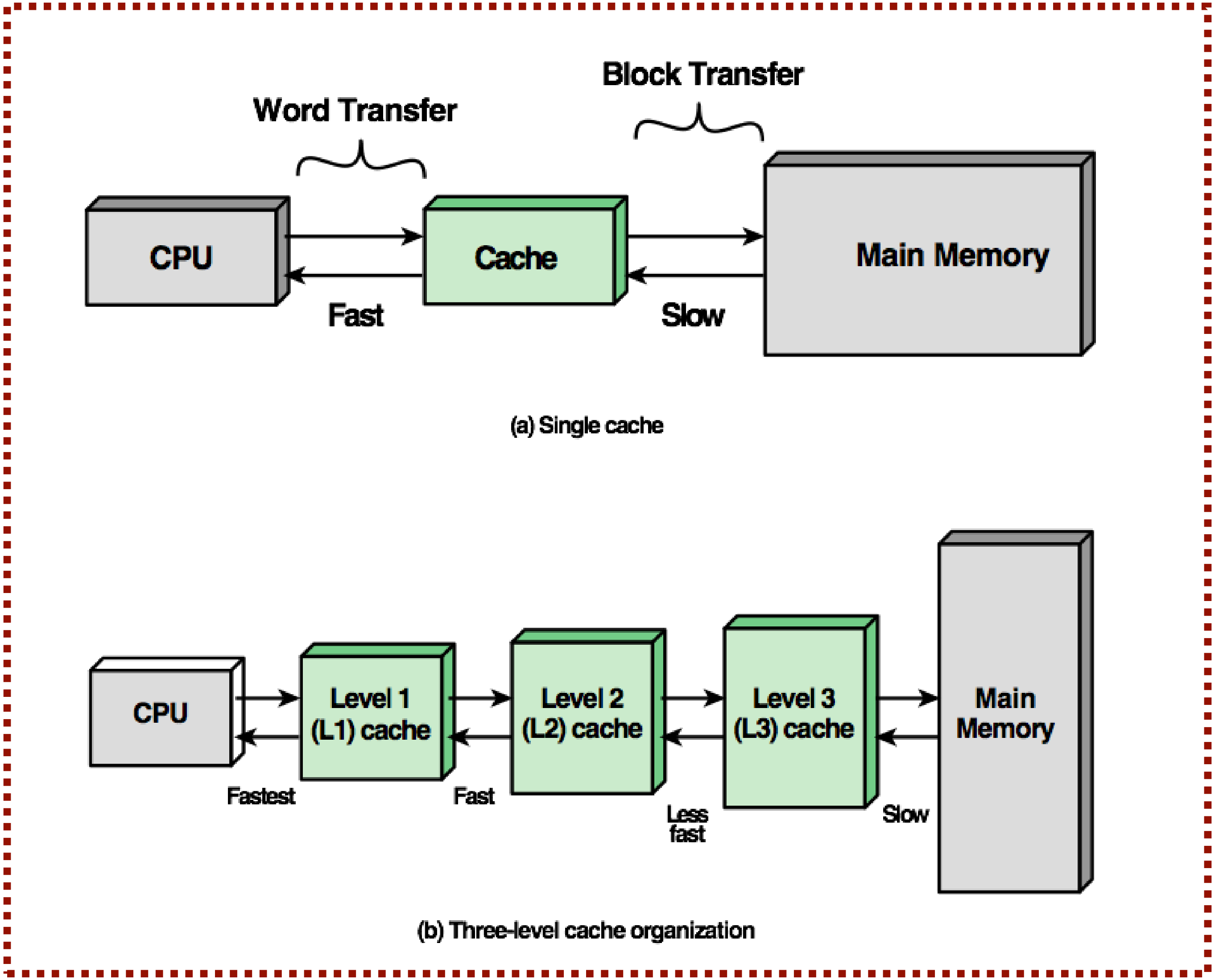
Principle of Locality

- Memory references by the processor tend to cluster
- Data is organised so that the percentage of accesses to each successively lower level is substantially less than that of the level above
- Can be applied across more than two levels of memory
  - Registers (fastest, smallest, most expensive)
  - Main memory (slower, larger, less expensive, *volatile*)
  - Secondary memory (slowest, much larger, least expensive, *non-volatile*)

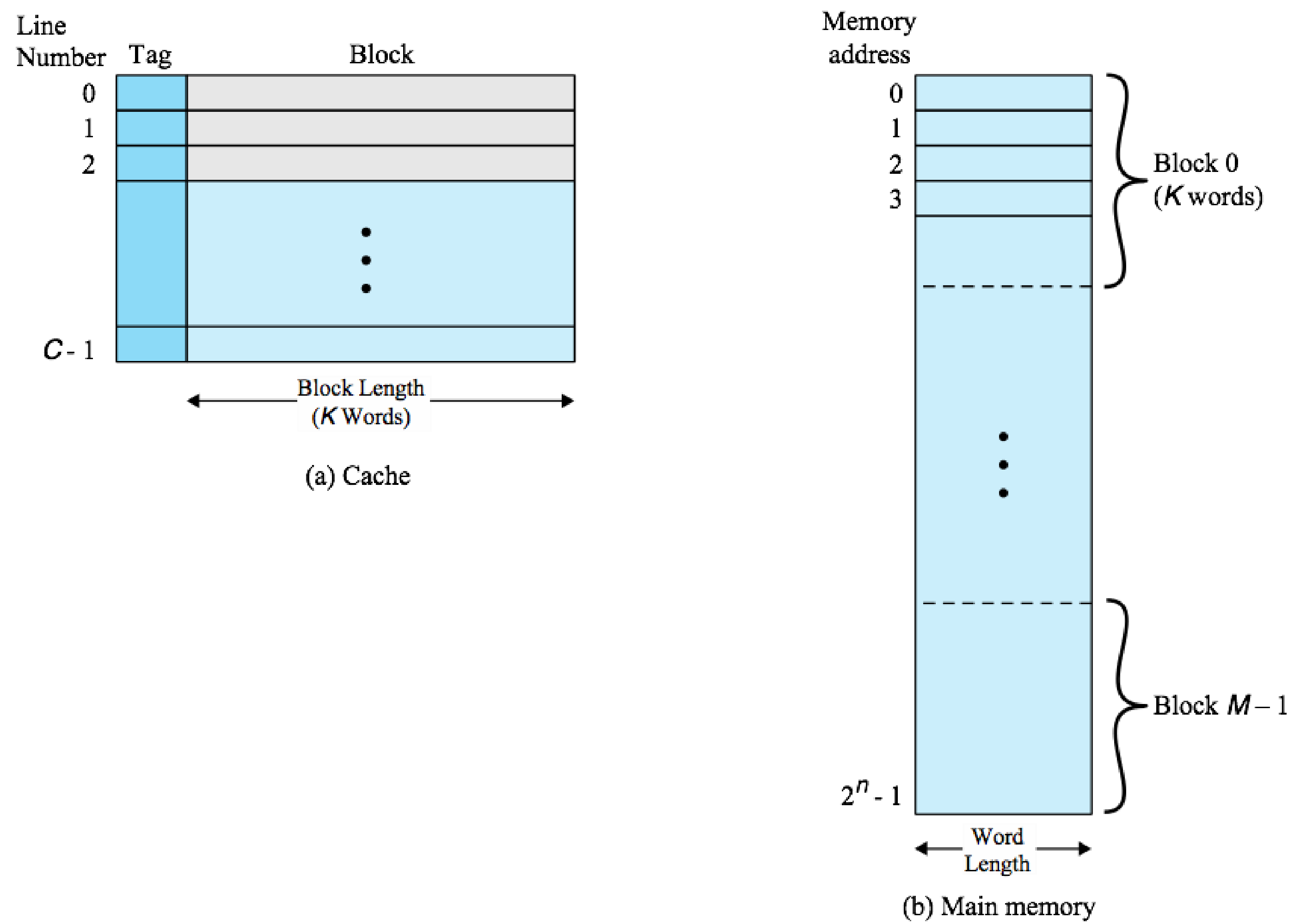
Cache Memory

- Cache contains a copy of portion of main memory
- Exploit the principle of locality with a small, fast memory
  - Because of locality of reference, it is likely that many of the future memory references will be to other bytes within the block
- Processor first checks the cache
  - If not found, a block of main memory is read into the cache

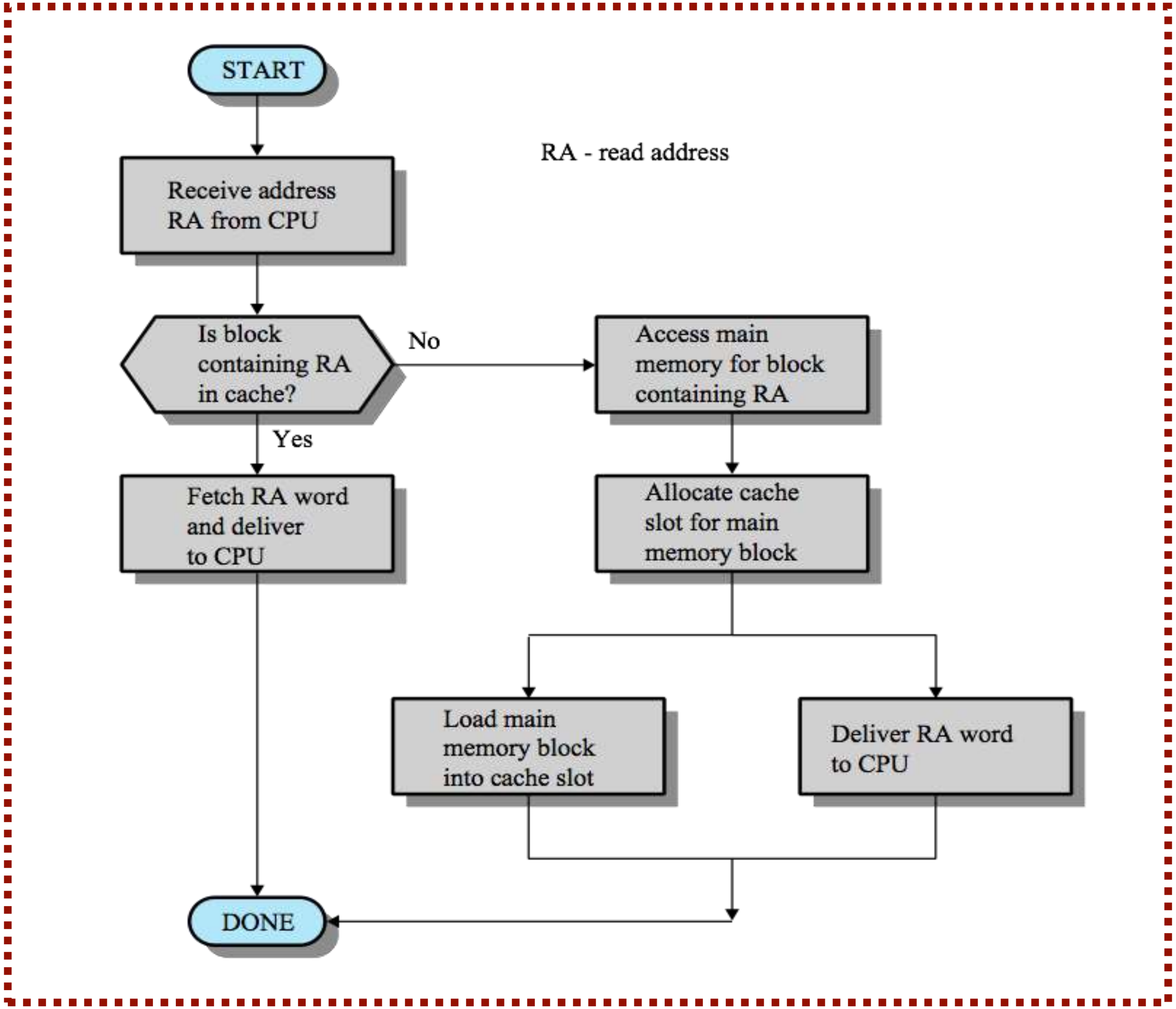
Cache Memory vs Main Memory



Structure: Cache Memory vs Main Memory



Cache: Read Operation



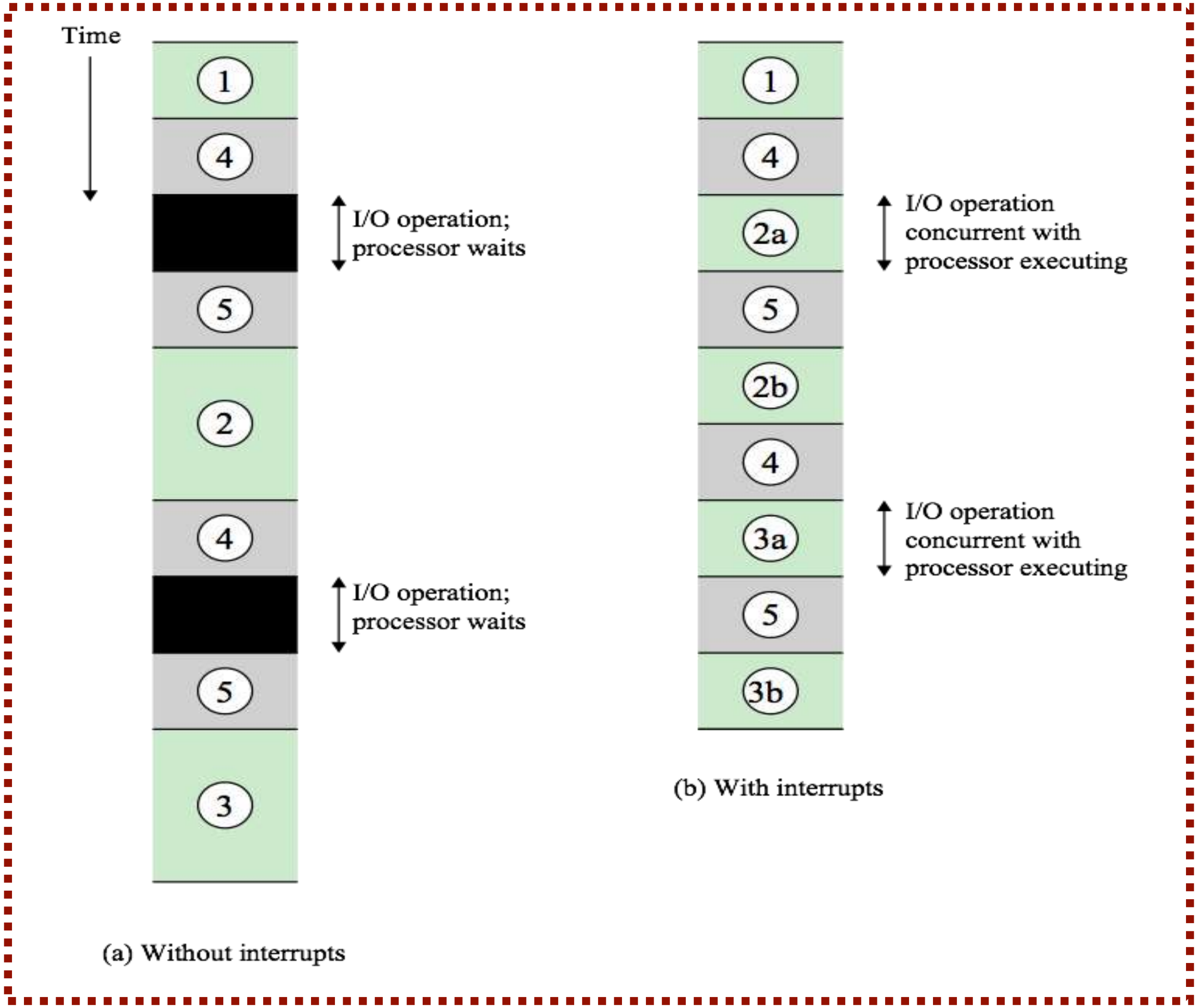
Lecture 1: Summary

- So far, we have discussed:
  - Basic elements of a computer system
  - Instruction execution cycle
  - Interrupts and interrupt processing
  - Memory hierarchy and cache memory
  - Some rules of the game
- Next week:
  - Overview of operating systems
- Reading:
  - Stallings, Chapter 1 (9th or 8th Edition)

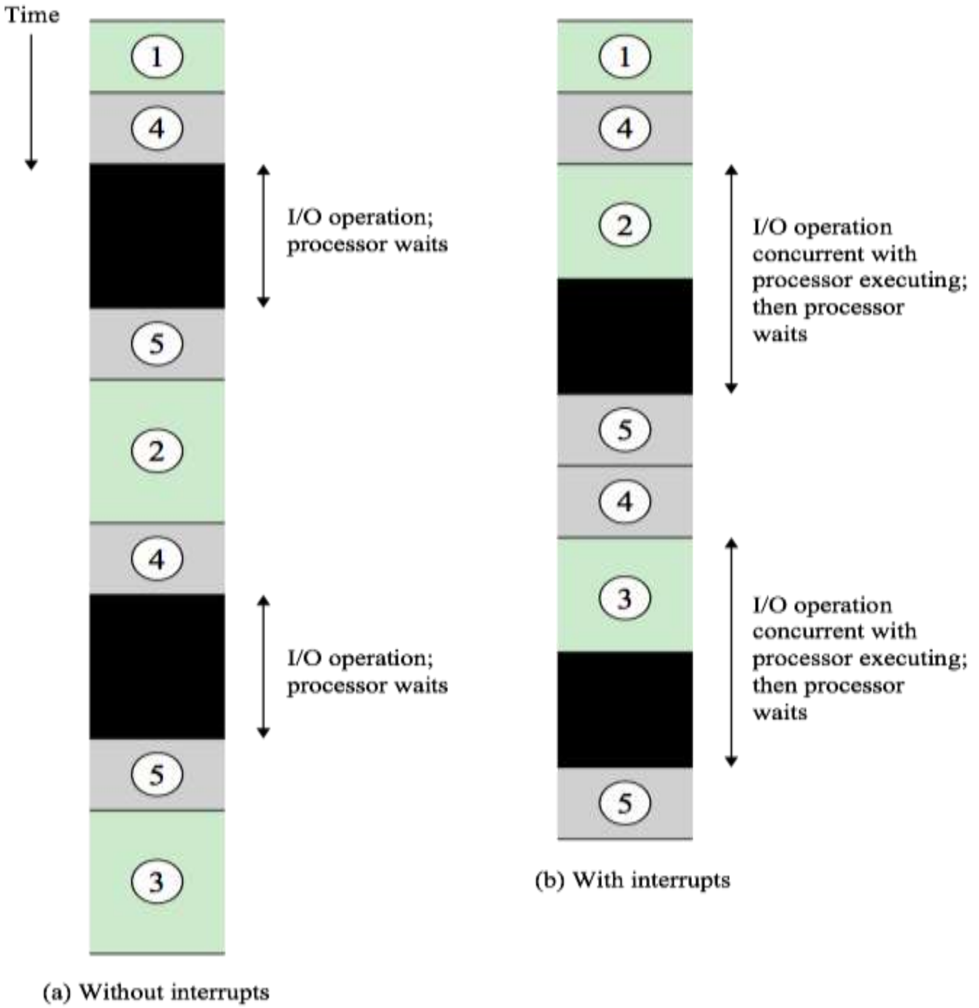
Reminder: Tutorials this week. Practicals next week.

Supplementary slides

Program Timing: Short I/O Wait



Program Timing: Long I/O Wait



Simple Interrupt Processing