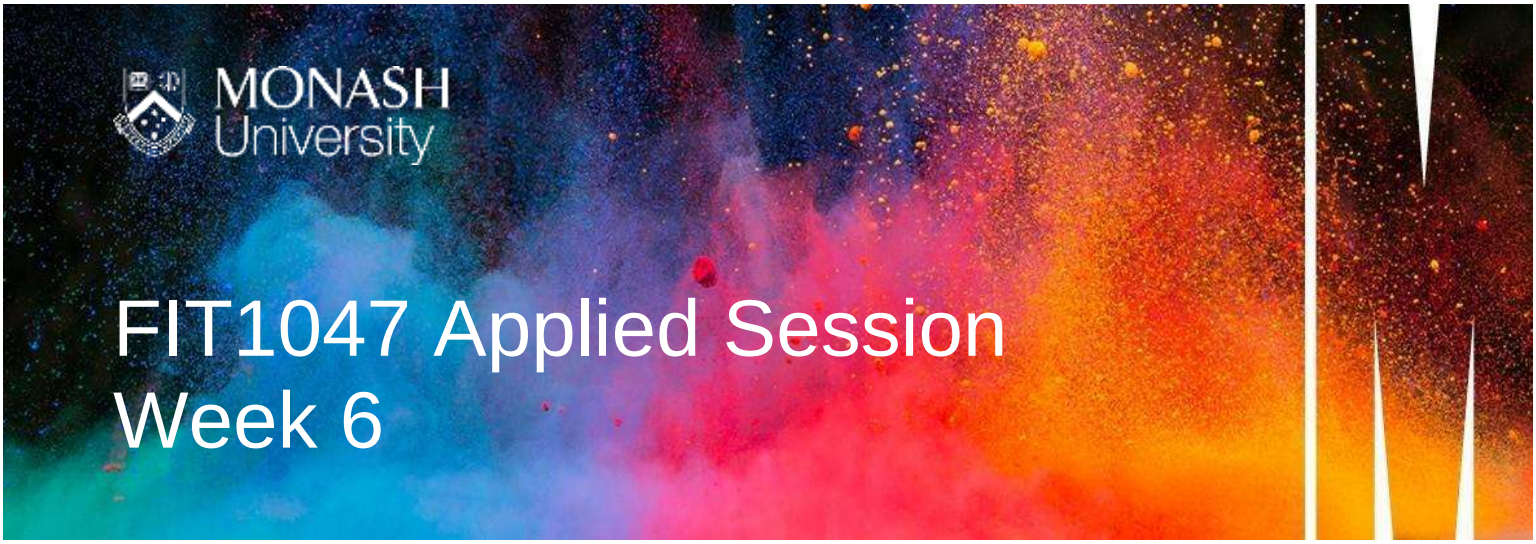


# OPERATING SYSTEMS, INPUT/OUTPUT AND BOOT PROCESS

## OBJECTIVES

- The purpose of this applied session is getting to know the computer boot process, I/O mechanisms, memory and process management.

## INSTRUCTIONS

- For some of the questions, you may have to refer to the pre-clas video and associated slides available in the Moodle.
- You may work in a small group.

## Activity 1: Input and Output

(a) You must have noticed that in our MARIE working environment, we can input data from Keyboard and send out data to MARIE output log. Investigate both the data movements (i) from keyboard to the CPU registers and (ii) from the CPU registers to the output log. Name the devices and registers involved in the data movement process. You may use different “View” options in the MARIE simulator environment to help you in the investigation.

(b) Input/output (I/O) devices are an essential part of computer systems. Memory-mapped I/O and Port-mapped I/O are the two methods that a CPU uses to receive data from and send data to I/O devices. Which one of these techniques require special instructions to carry out its I/O activity? One of the I/O methods requires no new instructions? Elaborate on the differences between these two methods.

MONASH University

## Sample Solution:

(a) Please try the following simple MARIE program in **"MARIE -> View ->DataPath"** mode performing "microstepping" to see the CPU registers in action.

```
input
output
Halt
```

(i) Keyboard -> IN Register -> AC
(ii) AC -> OUT Register -> Display Log

(b) Port-mapped I/O requires special instruction to be added to the CPU's Instruction Set Architecture (ISA). Memory-mapped I/O doesn't require any new instruction to be added to the ISA, rather the CPU accesses the I/O devices using the same memory accessing technique it uses to access code and data from memory. Each I/O register is "mapped" to a special memory address and the commands like **Load/Store** can be used to access the device.

In port-mapped I/O technique, each I/O register has an address (separate from memory address space), called ports. For example:

- Load A000 loads from memory address A000
- Input A000 loads from I/O register A000 (e.g. the keyboard)

MONASH University

# Activity 2: Boot Process and Operating System

(a) You are in front of a computer with the power supply turned-on. Explain how the different software components (BIOS, OS, and User Programs) are executed one after the other to provide a working environment for you.

**Figure 1**

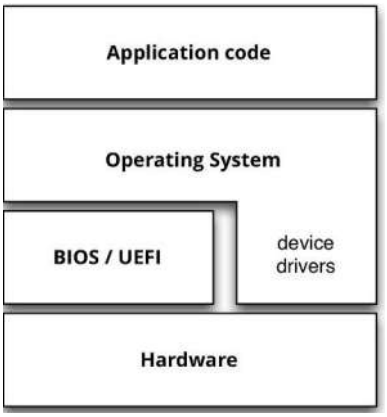


**Figure 2**

(b) During the booting process:

(i) From where (hard disk, memory, keyboard etc.) is the first instruction of BIOS loaded into CPU for execution?

(ii) Where is the Operating System (OS) stored? It must be loaded into memory for execution. Which software unit will load the OS into memory?

(iii) Device drivers help the OS to manage the I/O devices. Who will load the essential drivers (for keyboard, display etc.) into memory? And who will load the drivers for non-essential devices (printer, scanner etc.)?

(c) Referring to Figure 2, it is obvious that Application Code can't command or request the hardware for any kind of services (like, read from hard disk, accessing the printer) as the OS is in between them. What is the mechanism for Application Code to avail the services of hardware?

## Sample Solution:

(a) **Step 1**: The CPU starts execution of an instruction in the ROM (BIOS). The BIOS program in ROM starts with Power-On Self Test (POST). POST: checks RAM, CPU, GPU, communicates result via beep, initialises other hardware (hard disks, network, sound). Find Operating System (OS), load the OS boot code from disk into RAM, then jump to execute that code. **Step 2**: OS boot code loads OS "kernel". "Kernel" initialises drivers, then loads the rest of the OS. **Step 3**: Once everything is initialised, the Graphical User Interface is started. The computer is ready to use.

(i) from the ROM part of computer memory.

(ii) Can be on hard disk, USB disk, DVD, or in the network. BIOS will load the OS Kernel into memory (RAM part)

(iii) initial device drivers loaded by BIOS. And the advanced ones are loaded by the OS.

(b) Application code uses “system call” to generate a request to the OS to seek services from hardware.

## Activity 3: Running Processes “in parallel” on a Single CPU (Multi-Processing)

(a) For A Round Robin scheduling process, the queue condition is given below where the jobs are to be executed giving an equal time slice to each process maintaining the order of the queue. Assuming the time quantum is 2 time-unit (time slice =2), calculate the turnaround time for the following processes.

| No. | Process | Processing Time Required (Time Unit) | Start (Time) | End (Time) | Turn around (Time) |
|---|---|---|---|---|---|
| 1 | P1 | 4 | | | |
| 2 | P2 | 6 | | | |
| 3 | P3 | 9 | | | |
| 4 | P4 | 3 | | | |
| 5 | P5 | 1 | | | |
| 6 | P6 | 6 | | | |

(b) If we increase the time quantum to 4 time-units (time slice =4), will there be any improvement in the turnaround time for the processes? We can calculate the average turnaround time in both the cases to compare.

## Sample Solution:

(a) Time Slice = 2 time-unit

Time Steps (0-19)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Processes | | | | | | | | | | | | | | | | | | | |
| P1 | P1 | P2 | P2 | P3 | P3 | P4 | P4 | P5 | P6 | P6 | P1 | P1 | P2 | P2 | P3 | P3 | P4 | P6 | P6 |

Time Steps (20-39)

| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Processes | | | | | | | | | | | | | | | | | | | |
| P2 | P2 | P3 | P3 | P6 | P6 | P3 | P3 | P3 | | | | | | | | | | | |

| No. | Process | Processing Time Required (Time Unit) | Start Time | End Time | Turn around Time |
|---|---|---|---|---|---|
| 1 | P1 | 4 | 0 | 12 | 13 |
| 2 | P2 | 6 | 2 | 21 | 20 |
| 3 | P3 | 9 | 4 | 28 | 25 |
| 4 | P4 | 3 | 6 | 17 | 12 |
| 5 | P5 | 1 | 8 | 8 | 1 |
| 6 | P6 | 6 | 9 | 25 | 17 |

Average turnaround time = (13 + 20 + 25 + 12 + 1 + 17 ) / 6 = 87 / 6 = 14.66

(b) Time Slice = 4 time-unit

Time Steps (0-19)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P1 | P1 | P1 | P2 | P2 | P2 | P2 | P3 | P3 | P3 | P3 | P4 | P4 | P4 | P5 | P6 | P6 | P6 | P6 |

Time Steps (20-39)

| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P2 | P2 | P3 | P3 | P3 | P3 | P6 | P6 | P3 | | | | | | | | | | | |

| No. | Process | Processing Time Required (Time Unit) | Start Time | End Time | Turn around Time |
|---|---|---|---|---|---|
| 1 | P1 | 4 | 0 | 3 | 4 |
| 2 | P2 | 6 | 4 | 21 | 18 |
| 3 | P3 | 9 | 8 | 28 | 21 |
| 4 | P4 | 3 | 12 | 14 | 3 |
| 5 | P5 | 1 | 15 | 15 | 1 |

| No. | Process | Processing Time Required (Time Unit) | Start Time | End Time | Turn around Time |
|---|---|---|---|---|---|
| 6 | P6 | 6 | 16 | 27 | 10 |

Average turnaround time = (4 + 18 + 21 + 3 + 1 + 10 ) / 6 = 57 / 6 = 9.5

Yes, by increasing the time slice, we can reduce the average turnaround time in this case.

## Activity 4: Virtual vs Physical Memory

In Memory Management, we have learnt that when user programs become processes, each process is given its own address space and the addresses used are called virtual addresses. The OS maps virtual addresses to physical addresses in RAM. It is done by the Memory Management Unit (MMU). Help the MMU by proposing a mapping mechanism for Virtual Address to Real Memory Address for the following scenario where two user processes (web browser and email) are in the memory.

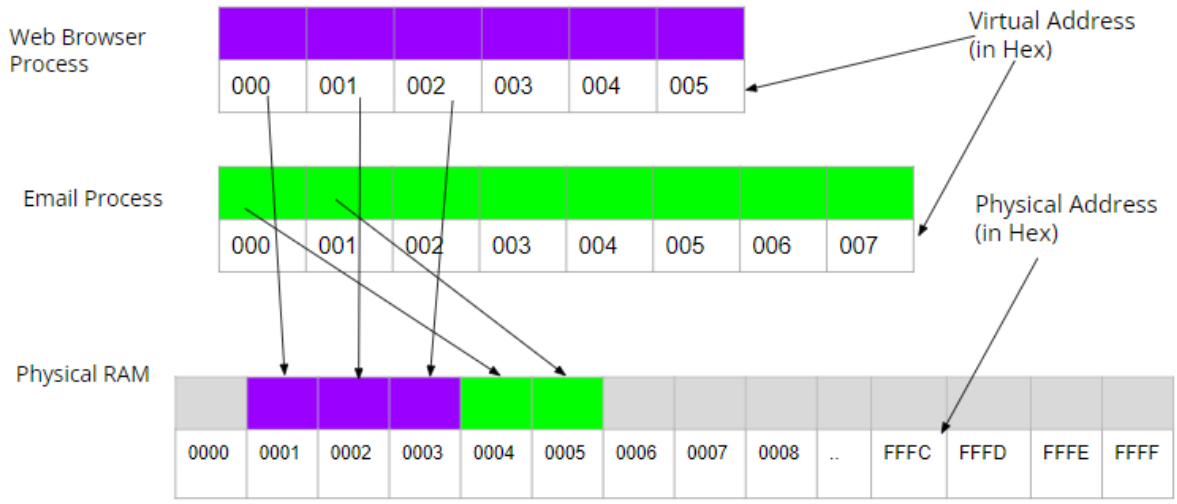


**Figure 3**

## Sample Solution:

MMU can use mapping tables for this purpose. Two sample tables are given below.

**Mapping Table for Web Browser Process**

| Virtual memory Address (in Hex) | Physical Memory Address (in Hex) |
|---|---|
| 000 | 0001 |
| 001 | 0002 |

| | |
|---|---|
| 002 | 0003 |

**Mapping Table for Email Process**

| Virtual memory Address (in Hex) | Physical Memory Address (in Hex) |
|---|---|
| 000 | 0004 |
| 001 | 0005 |
| | |