REVIEW · UNITS 1 TO 3

Sessions 1, 2 and 3, in fifteen slides

Everything taught in Sessions 1, 2 and 3 — for anyone who missed one, and for everyone before the Prelim.

2ND SEMESTER · SCHOOL YEAR 2026-2027

Introduction to Computing · Prof. Ryan V. Labrador

Sessions 1-3 · 15, 22 and 29 August 2026

WHERE THIS FITS

Three Saturdays, one argument

SESSION 1	**What computing is**	15 August · Unit 1. The difference between a computer, computing and computer science. Where the machines came from, why they got fast, and what the industry looks like in the Philippines.
SESSION 2	**What is inside the box**	22 August · Unit 2. The five parts of a computer system, the IPO+S model, what a CPU actually does, and why memory comes in four speeds instead of one.
SESSION 3	**What makes it do anything**	29 August · Unit 3. Software in three layers, what an operating system manages, how software is licensed — and your group's website going live on the public internet.

The three sessions are one argument in order: what computing is, what the machine is made of, and what turns that metal into something that answers.

Three words people mix up

1 Computer — a machine

A device that accepts data, processes it by following instructions, and produces a result. It is an object. You can point at it.

2 Computing — an activity

Any goal-oriented work that uses, builds or studies computers. Much bigger than programming: planning a system, managing data and training people are all computing.

3 Computer Science — a discipline

The formal study of computation, algorithms, information, and the systems that handle them. It is a field of study, not a job description.

You are enrolled in Information Systems

Computing applied to real organisations and the people inside them. It sits closest to the business and furthest from the physics — a deliberate choice, not a lesser one. Computer Science, Computer Engineering, Information Technology and Software Engineering are the neighbouring fields, and a working IT project needs all of them.

Likely exam item. Be able to define all three and say which one you are studying.

Where computing came from



The ideas came first, then speed, then connection. Programmable machines existed in 1804, long before electricity was involved.

The dates worth remembering

1804	The Jacquard loom. Punched cards, and the first program.	**1843**	Ada Lovelace writes the first algorithm made for a machine.
1936	Turing defines what a computer is, before one exists.	**1947**	The transistor, at Bell Labs. Everything follows from it.
1969	ARPANET connects four computers. This becomes the internet.	**1991**	Berners-Lee opens the World Wide Web to the public.

Why it accelerated — and why that matters here

2,300	billions	$40B+	1.97M
transistors on the Intel 4004	transistors on a phone chip	IT-BPM export revenue	jobs projected in the sector
1971	today	2025 · IBPAP	by end of 2026

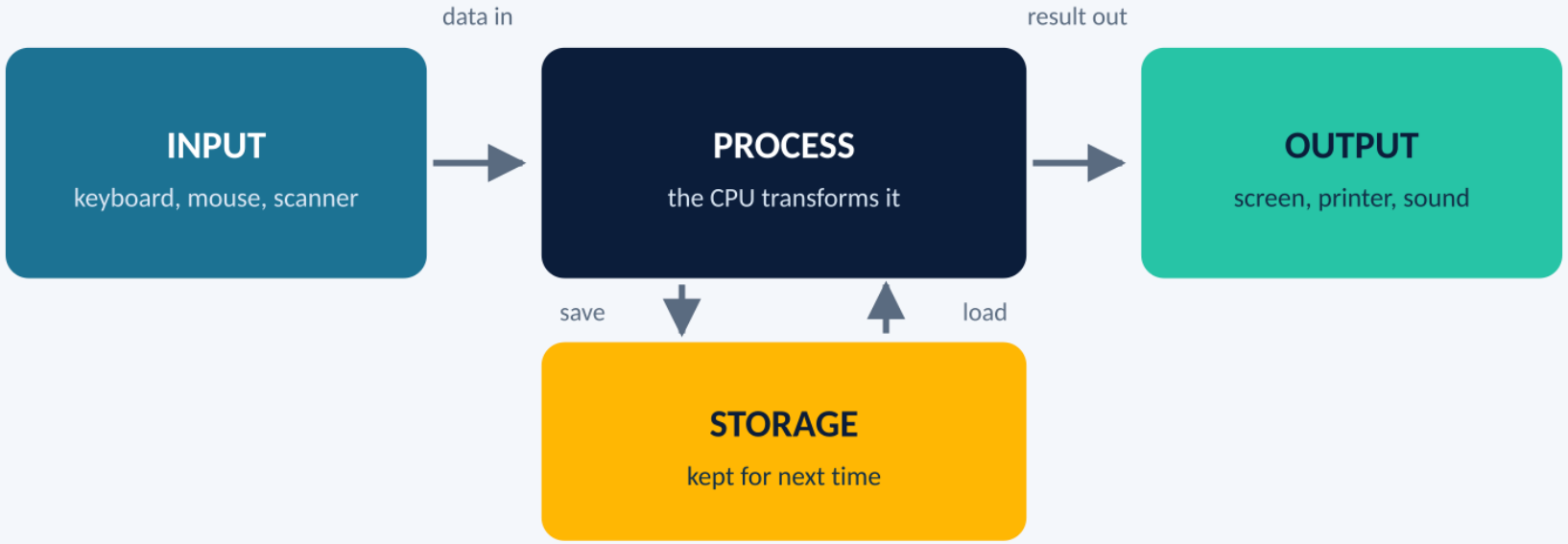
MOORE'S LAW	**It compounds**	In 1965 Gordon Moore observed that the number of transistors on a chip roughly doubles every two years. It held for decades — which is why change feels fast rather than steady.
ZERO COPY COST	**Software is different**	Once written, a program costs almost nothing to copy. That single economic fact is why a student in Ilagan can build something a global company could not have afforded in 1990.
LOCATION	**The constraint moved**	Remote work and global capability centres moved real technical jobs out of Metro Manila. The constraint used to be geography. Now the constraint is skill.

A computer system is five things

HARDWARE	**The physical parts**	If it can fall on your foot, it is hardware. Manufactured, has a serial number, wears out, and copying one costs money.
SOFTWARE	**The instructions**	Written by people, stored as files. Cannot be dropped. Does not wear out — it becomes outdated instead. Copying costs almost nothing, which is why licensing exists.
DATA	**What goes in and comes out**	The reason the whole system exists at all. Everything else is machinery for handling this.
PEOPLE	**Users, operators, administrators**	The most common point of failure in any real system. Not a joke, and not an insult — it is why procedures exist.
PROCEDURES	**The agreed steps**	Backup schedules, sign-in rules, who is allowed to do what. Invisible until the day one is missing.

Most people think 'computer' means only the first one. Four of the five are invisible. Firmware sits exactly on the line — software that lives permanently on a hardware chip.

The model you will use all semester



Storage sits under Process, not after Output — a system can save or reload at any point.

WORKED EXAMPLE · THE MRT-3 TURNSTILE IN MANILA

INPUT
You tap. The card touches the reader and a card number enters the system.

PROCESS
It decides. Is the card valid? Is the balance enough? Compute the fare.

OUTPUT
The gate opens. The barrier releases and the balance shows on the display.

STORAGE
It is written down. Station, time and balance are logged for reconciliation.

Inside the processor

CONTROL UNIT	**The traffic officer**	Fetches the next instruction, works out what it means, and tells every other part when to act. It does no arithmetic itself.
ALU	**The calculator**	Arithmetic Logic Unit. Adds, subtracts, compares, and answers true/false questions. Every decision your computer makes ends here.
REGISTERS	**The scratchpad**	A handful of tiny slots inside the CPU holding the values being worked on right now. The fastest storage that exists, and the smallest.

THE MACHINE CYCLE · REPEATED BILLIONS OF TIMES A SECOND, FOR AS LONG AS THE POWER IS ON

1 · FETCH
The control unit collects the next instruction.

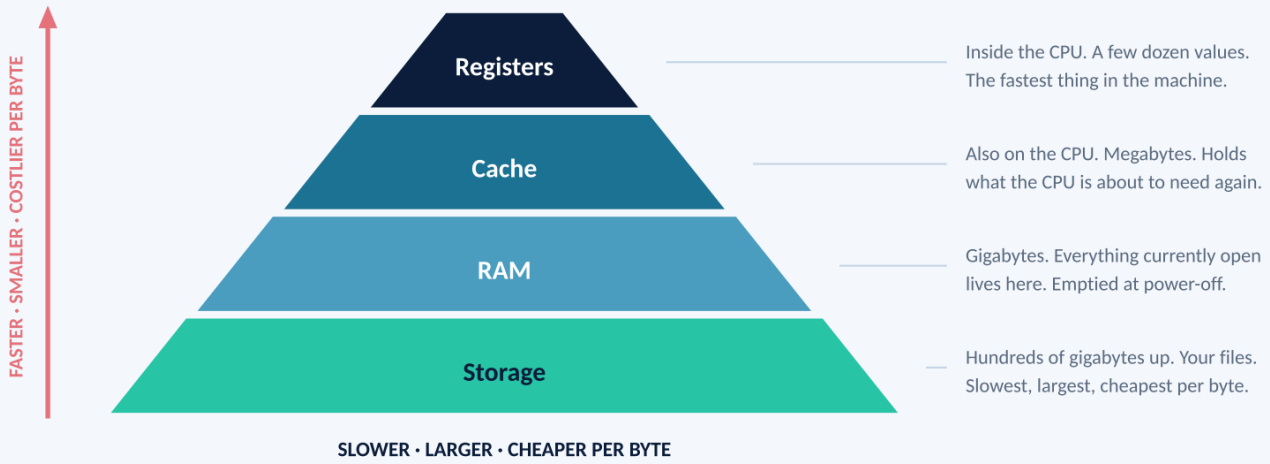
2 · DECODE
It works out which operation that instruction is.

3 · EXECUTE
The ALU, or another unit, performs it.

4 · STORE
The result goes back to a register, or to memory.

A CPU cannot store your files and a hard drive cannot add two numbers. Different jobs.

Four kinds of remembering



RAM — volatile

The working desk. Holds whatever is open right now, and loses every bit of it the moment power is cut. Usually the cheapest real upgrade.

ROM — non-volatile

Instructions built into the machine. Holds the firmware that starts the computer, and keeps its contents with no power at all.

HDD · SPINNING DISK

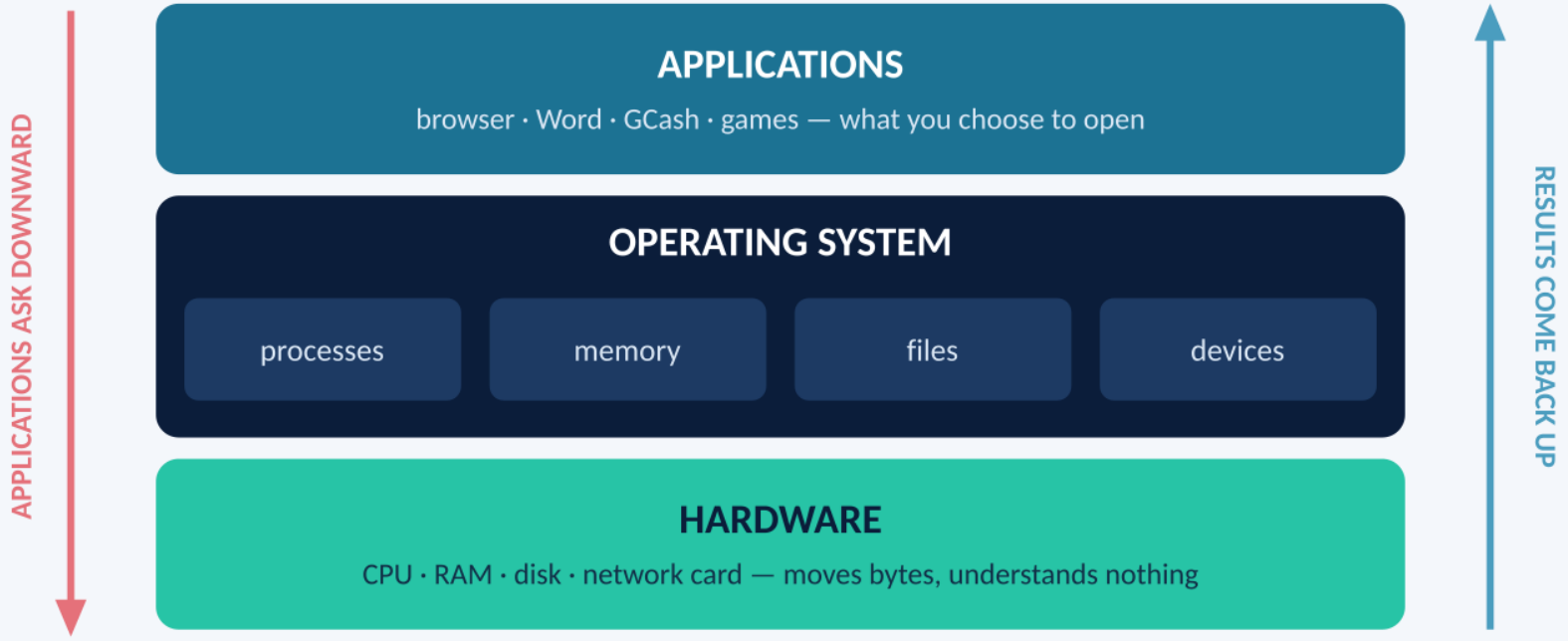
Magnetic platters under a moving head. Cheapest per gigabyte. Moving parts, so drops kill it. Slower to boot and open files. Still sensible for bulk backup.

SSD · FLASH MEMORY

No moving parts at all. Costs more per gigabyte, less every year. Survives being knocked. Dramatically faster — the upgrade that makes an old laptop feel new.

The unplug test: if pulling the plug erases it, it was volatile. HDDs and SSDs are both non-volatile — both keep your files with the power off.

Software comes in three layers



An application cannot touch the disk. It has to ask. That is why operating systems exist.

The four grey boxes inside the middle layer are the operating system's four jobs — processes, memory, files, devices. They are the next slide, and they are a near-certain exam item.

The operating system is a manager

PROCESSES	**Who runs, and when**	Decides which program runs on which core and for how long. This is why your phone can play music while you type a message.
MEMORY	**Who gets which space**	Gives every program its own space in RAM and stops one program reading another's. It is also why one app crashing does not take the machine with it.
FILES	**Where everything sits**	Folders, filenames, permissions. It tracks where every byte of your file physically sits on the disk, so you never have to.
DEVICES	**How it talks to things**	Talks to the printer, the Wi-Fi card, the keyboard. A driver is how the operating system learns a device it has never met before.

FROM COLD METAL TO A DESKTOP

1 Firmware wakes and checks the hardware is present — that is ROM doing its job. 2 The bootloader finds the operating system on the drive and loads it into RAM. 3 The kernel takes control of the CPU, the memory and the devices. 4 The interface loads last. By the time you see a desktop, most of the work is finished.

Two families, and five licences

System software

- Runs the machine, not your task
- Operating systems, and the device drivers through which they learn a new device
- Firmware, and utilities: antivirus, backup, compression, disk cleanup, updates, task manager
- You rarely open it on purpose. You notice it when it fails.

Application software

- Runs your task, not the machine
- Productivity, communication, creative, entertainment, financial
- Business systems — payroll, inventory, enrolment, hotel booking
- If you chose to open it, it is application software.

FIVE LICENCE MODELS · FREE IS ABOUT PRICE. OPEN IS ABOUT ACCESS TO THE SOURCE. THEY ARE NOT THE SAME WORD.

PROPRIETARY
You buy permission. Windows, Adobe. Source closed.

OPEN SOURCE
Read it, change it. Linux, LibreOffice, VLC.

FREEWARE
Free, but still closed. You cannot see the code.

SHAREWARE
Free for a trial, then it asks for payment.

SAAS
You rent access. Stop paying and it ends.

What a web server actually is



A COMPUTER THAT NEVER SLEEPS

The same parts you named in Session 2 — CPU, RAM, storage — running Linux. Ours is at 58.71.69.30 and is never switched off.

A PROGRAM THAT IS LISTENING

Something on it has to be awake and waiting for requests. Apache is. Since Session 3, so is your group's Python program.

A NAME THAT POINTS AT IT

g01.cic.rmlsystems.net refers to that same computer. How a name finds a machine is Session 9.

There is no cloud. There is somebody else's computer, in a building, with a power bill — and programs running on it.

The same page, written two ways

index.html — a file Apache sends a copy of

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Welcome Group 1</title>
</head>
<body>
  <h1>Welcome, Group 1</h1>
  <p>Introduction to Computing</p>
</body>
</html>
```

index.py — a program that builds a reply

```
from flask import Flask, render_template_string
app = Flask(__name__)

INFO = {"group": "1"}
PORT = 8000 + int(INFO["group"])   # group 7 -> 8007

PAGE = """<!DOCTYPE html>
<html lang="en">
<head><title>Welcome Group {{ i.group }}</title></head>
<body>
  <h1>Welcome, Group {{ i.group }}</h1>
  <p>Introduction to Computing</p>
</body>
</html>"""

@app.route("/")
def welcome():
    return render_template_string(PAGE, i=INFO)

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=PORT,
            debug=False, use_reloader=True)
```

@APP.ROUTE("/")

The address. When somebody asks for your homepage, run the function underneath. Change this line and the page moves.

INFO AND PORT

Your group number is typed once. Every other line reads it — including your port. Two groups on one port stops both.

WHY FLASK

A browser expects an HTTP reply, in a format agreed in 1991. Flask is a library that speaks it so you never have to.

Identical pages. On the left, Group 1 is typed three times and is there forever. On the right it is typed once — and the page can work something out before it answers.

CATCHING UP

If you missed a Saturday, do these five things

1 · READ	The full deck for that session	The slides and the worksheets are on download.cic.rmlsystems.net. This summary keeps the ideas; the full deck keeps the examples.
2 · ASK	Your group, not the internet	You are in a group of eight for the whole semester. Someone at your table sat through the session you missed. Ask them before you ask anything else.
3 · OPEN	Your group's live site	g01 through g10 dot cic.rmlsystems.net. It answers right now. Look at what your group published in Session 3. The index.py that builds that page is on the server — step 4 is how you reach it.
4 · CHECK	That you can reach the server	SFTP to 58.71.69.30 on port 22, username g01 to g10, into the app folder — not public_html. Your group leader has the password on paper.
5 · REDRAW	IPO+S, from memory	Four boxes, storage under process. If you can draw it for a system you use and name the four parts of the machine cycle, you are level with the room.

Do number 4 before the next laboratory. A connection problem discovered at 1pm on a Saturday costs you the whole afternoon.