FETCH-EXECUTE CYCLE AND ASSEMBLY LANGUAGE

HILLVIEW INTERNATIONAL SCHOOL – YEAR 7

LESSON OUTCOMES

By the end of this lesson, you will be able to:

- Understand what the CPU does and how it works, including:
  - Control Unit
  - Registers (aka “immediate access store”)
  - Arithmetic and Logic Unit (ALU)
- Understand what the Fetch-Execute Cycle is and how instructions are executed in computers
- Be able to write simple assembly language programs

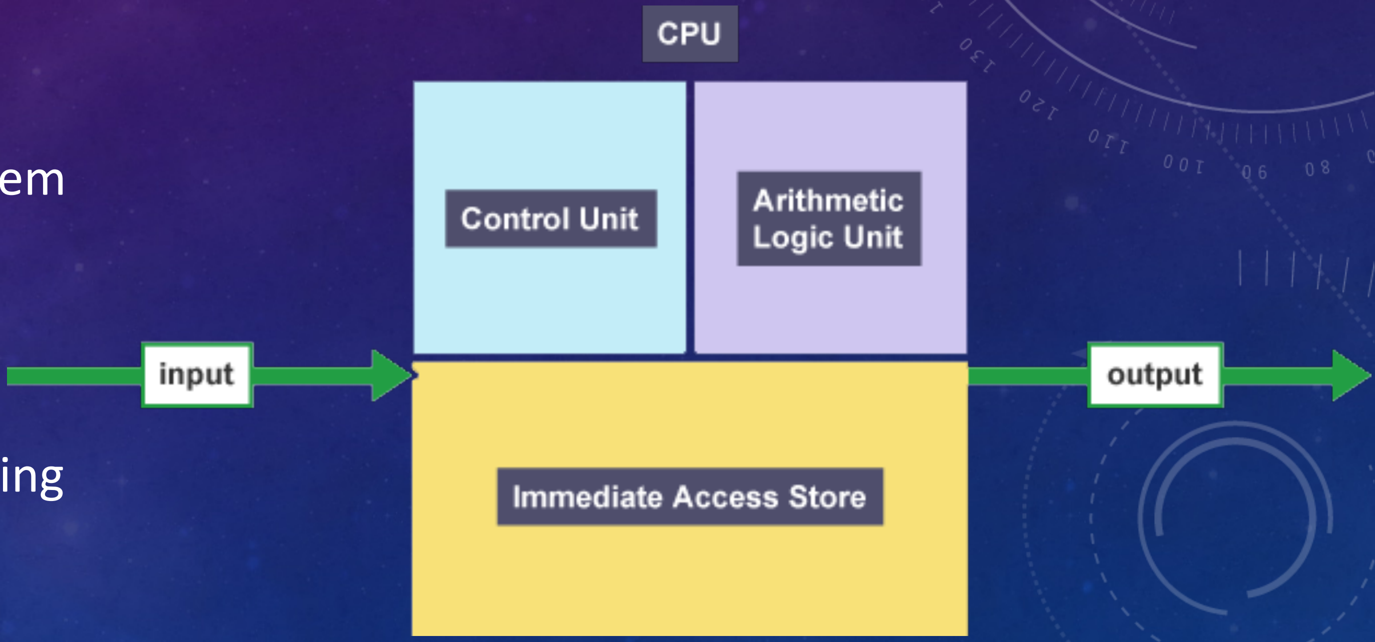
CENTRAL PROCESSING UNIT

Made up of 3 parts:

1. **Control Unit**
   - Controls the flow of data within the system

1. **Arithmetic Logic Unit (ALU)**
   - Where the CPU holds all the data and programs that it is currently using

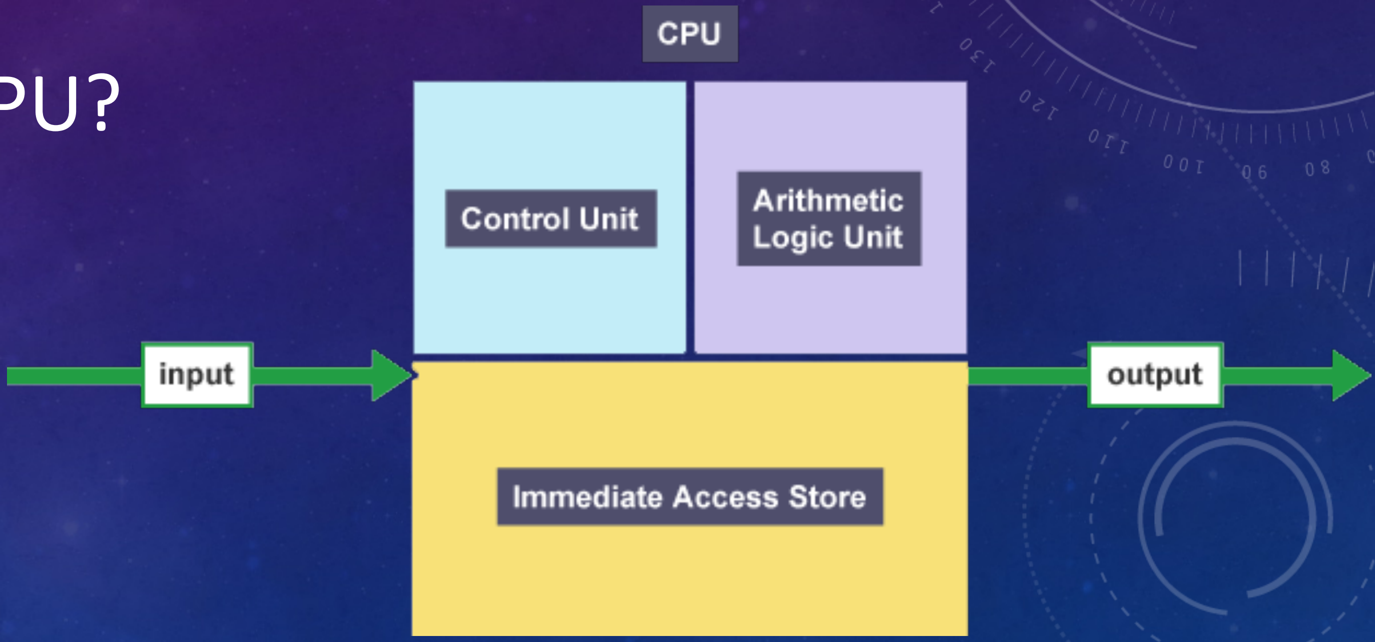
1. **Immediate Access Store (Registers)**
   - Where the CPU performs the arithmetic and logic operations



RECAP QUESTION

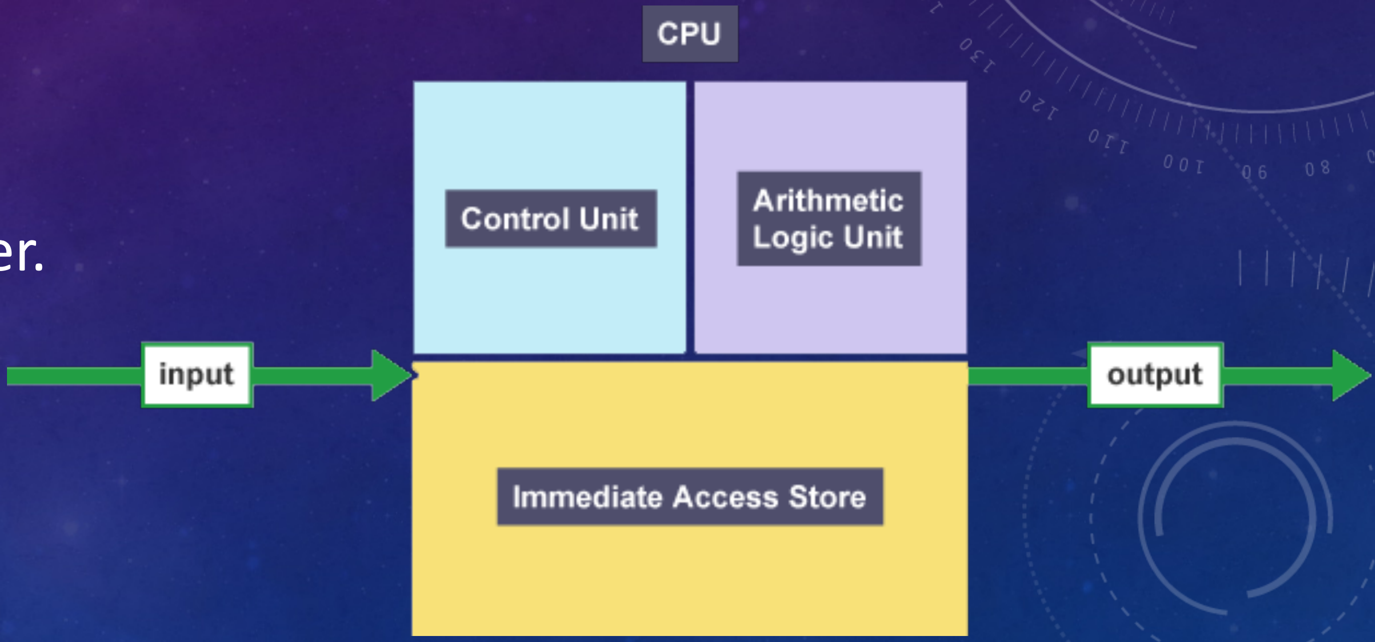
What are the 3 parts of the CPU?

1. Control Unit
2. Arithmetic Logic Unit (ALU)
3. Immediate Access Store (Registers)



CONTROL UNIT

- Controls the flow of data within the system.
- Monitors communications between the hardware attached to the computer.
- Controls the input and output of data:
  - Checks that signals have been delivered successfully
  - Makes sure that data goes to the correct place at the correct time.

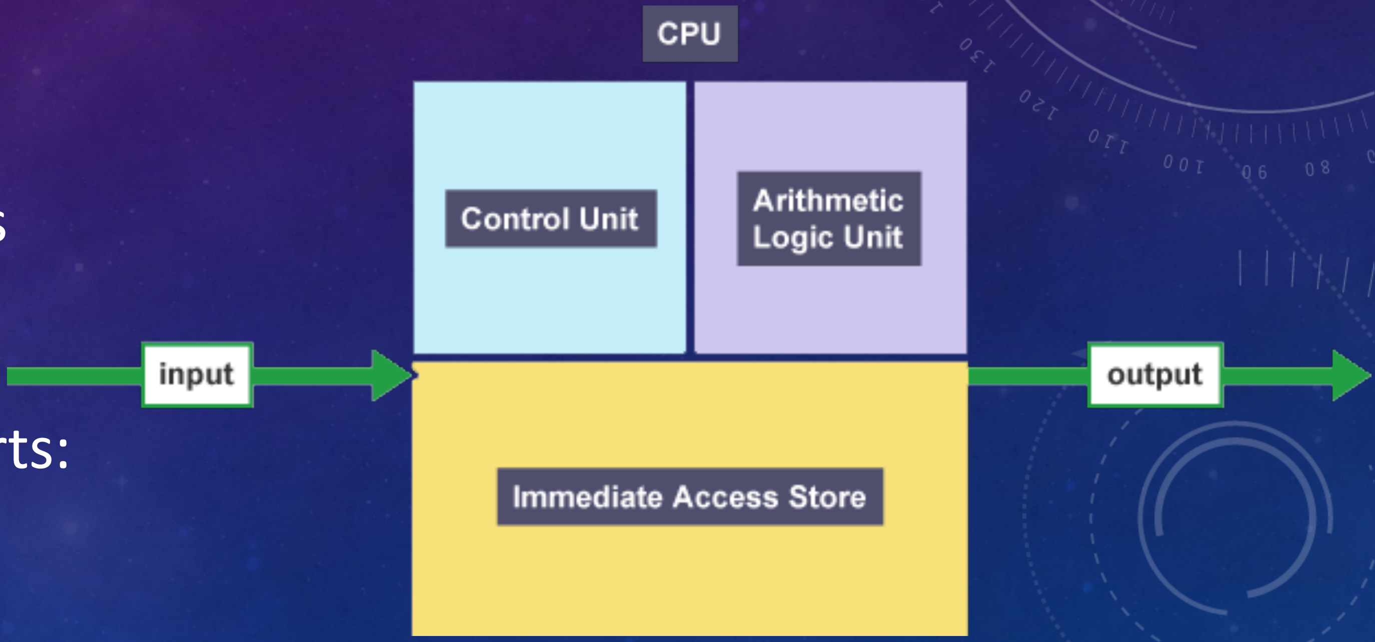


ARITHMETIC LOGIC UNIT (ALU)

- Where the CPU performs the arithmetic and logic operations.

- Every task that your computer carries out is completed here.

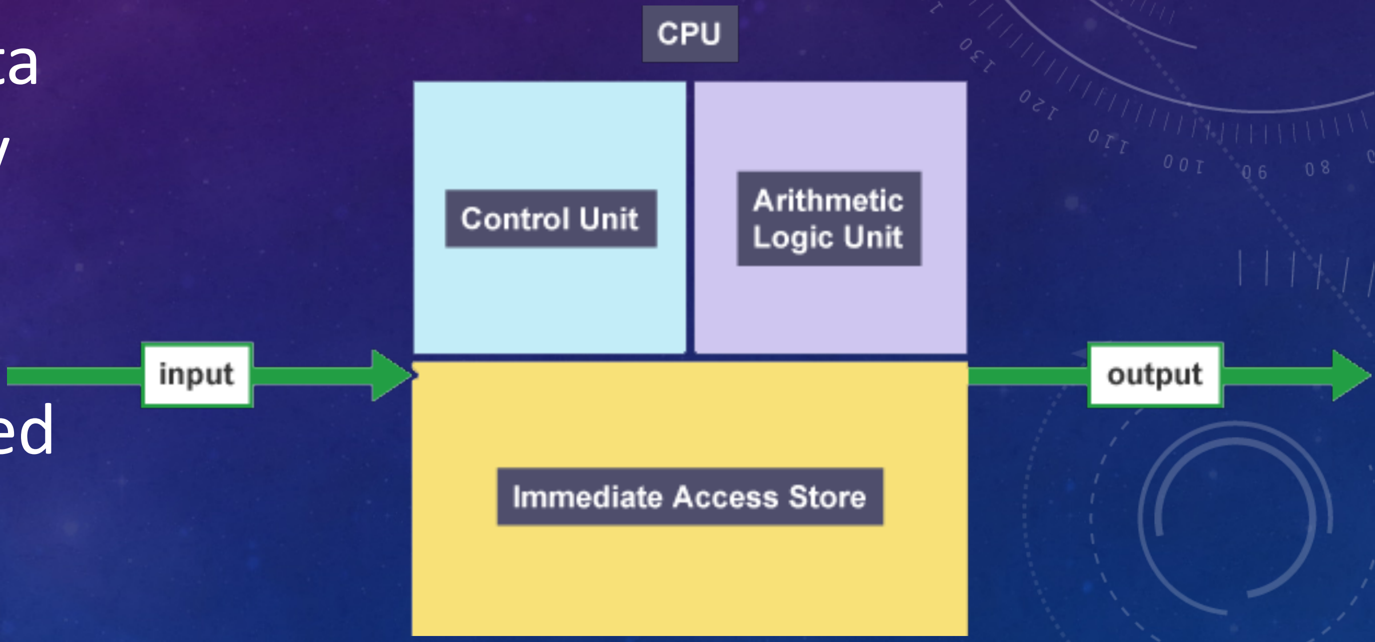
- The ALU's operations fall into two parts:
  1. Arithmetic part, which deals with calculations, eg, 1 + 2 = 3
  2. Logic part, which deals with any logical comparisons, eg, 2 > 1



IMMEDIATE ACCESS STORE (REGISTERS)

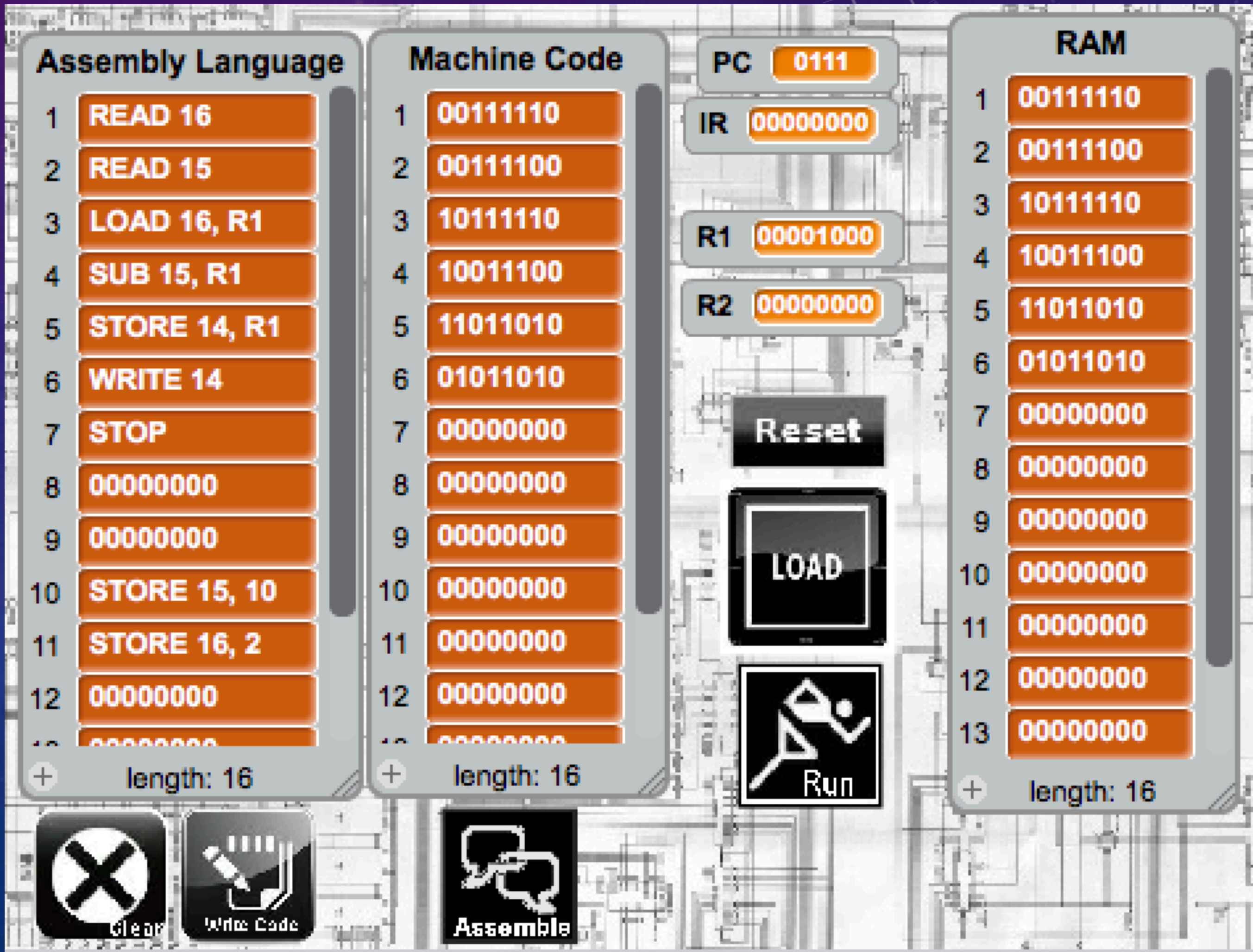
- Where the CPU holds all the data and programs that it is currently using.

- Think of it like the numbers typed into a calculator:
  - They are stored inside the calculator while it processes the calculations.



ASSEMBLY PROGRAMMING

- Copy **AL.sb2** to your P: drive folder
- Open it with Scratch
- PC = Program Counter (which line is running)
- IR = Intermediate Representation (current instruction in binary)
- R1 = Register 1
- R2 = Register 2



	Assembly Language	Machine Code	RAM
1	READ 16	00111110	00111110
2	READ 15	00111100	00111100
3	LOAD 16, R1	10111110	10111110
4	SUB 15, R1	10011100	10011100
5	STORE 14, R1	11011010	11011010
6	WRITE 14	01011010	01011010
7	STOP	00000000	00000000
8	00000000	00000000	00000000
9	00000000	00000000	00000000
10	STORE 15, 10	00000000	00000000
11	STORE 16, 2	00000000	00000000
12	00000000	00000000	00000000
13			00000000
	length: 16	length: 16	length: 16

PC 0111

IR 00000000

R1 00001000

R2 00000000

TASK 1: INTRODUCING ASSEMBLY PROGRAMMING

Write an Assembly Language program that subtracts one number from another and outputs the result.

The code is shown here →

You have 10 minutes

Example assembly program:

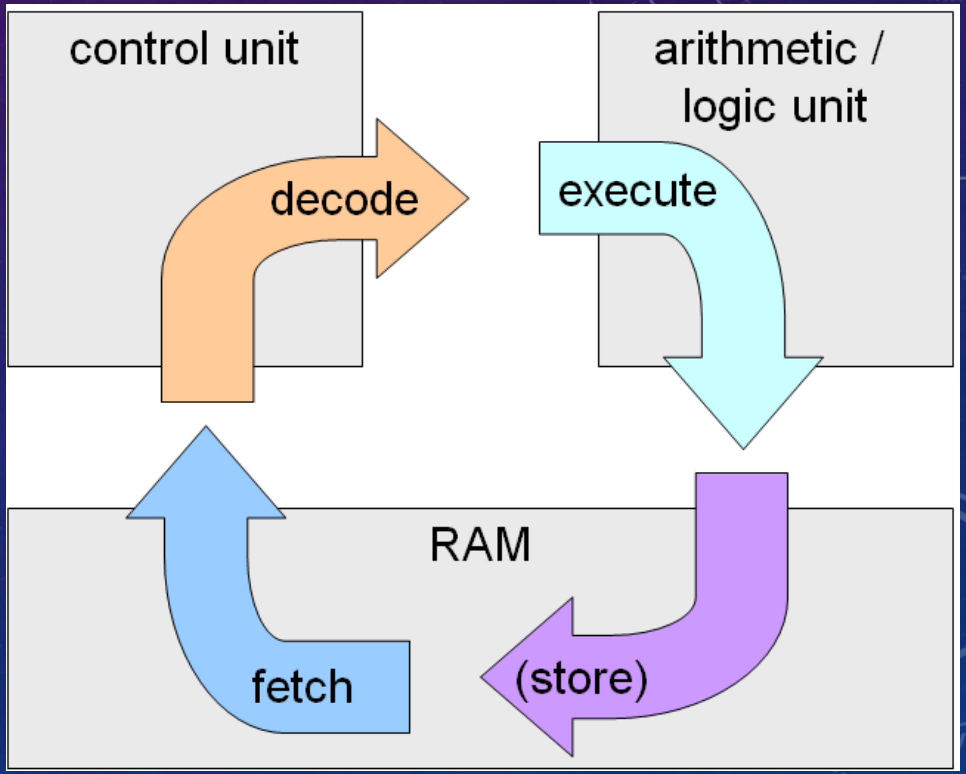
```
READ  16
READ  15
LOAD  16, R1
SUB   15, R1
STORE 14, R1
WRITE 14
STOP
```

FETCH-EXECUTE CYCLE

- The basic operation of a computer is called the ‘fetch-execute’ cycle

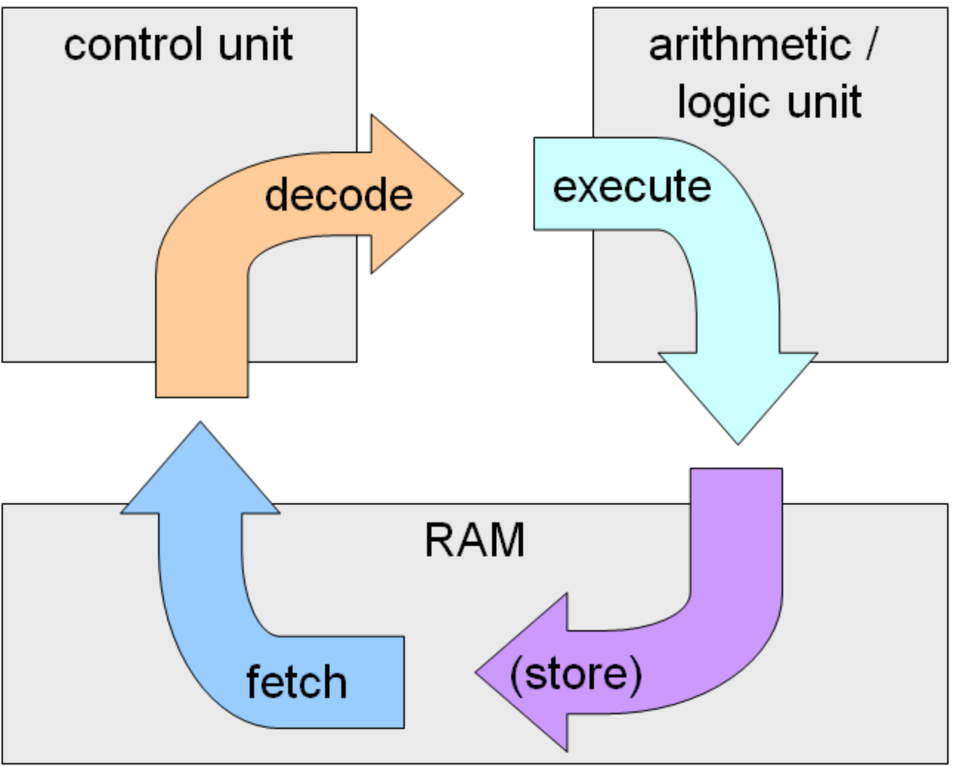
- The computer fetches the instruction from its memory and then executes it

- This is done repeatedly from when the computer is booted up to when it is shut down



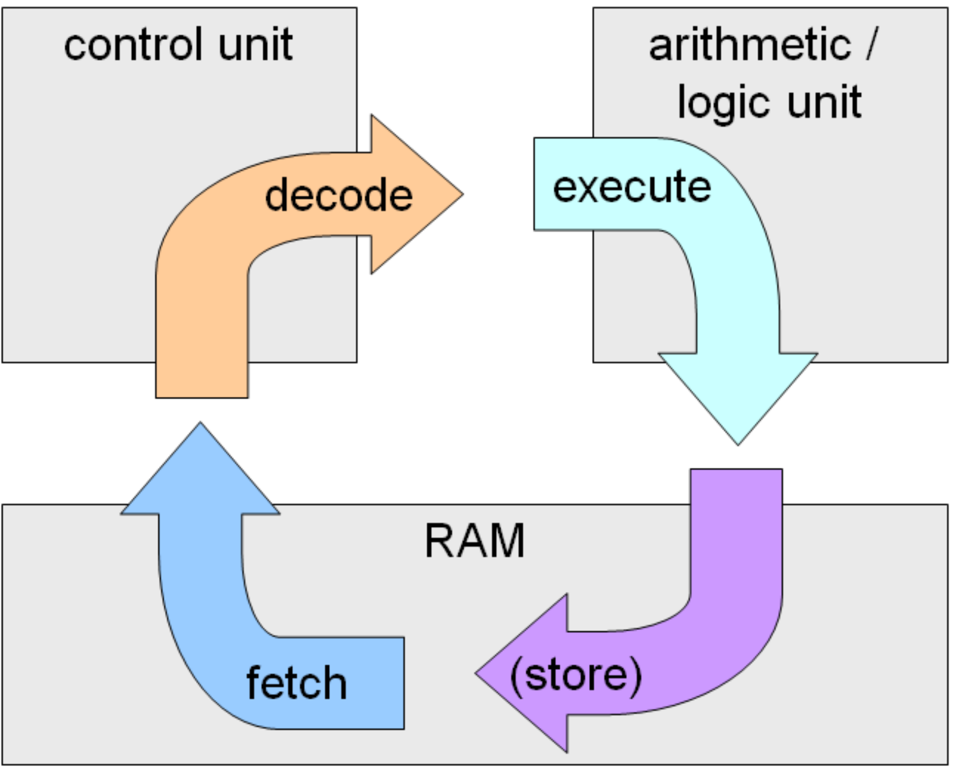
FETCHING THE INSTRUCTION

- The first step the fetch-execute cycle carries out is fetching the instruction
- The CPU fetches the instruction from the main memory and stores it in the CPU temporary memory, the immediate access store (registers)
- Once the instruction has been fetched, the CPU will need to understand the instruction to action it
  - This is called **decoding**



EXECUTING THE INSTRUCTION

- When the instruction has been decoded, the CPU can carry out the action that’s needed
  - This is called executing the instruction
- The CPU is designed to understand a set of instructions - the instruction set
- A single piece of program code usually requires several instructions



EXECUTING THE INSTRUCTION

Look at this Python code:

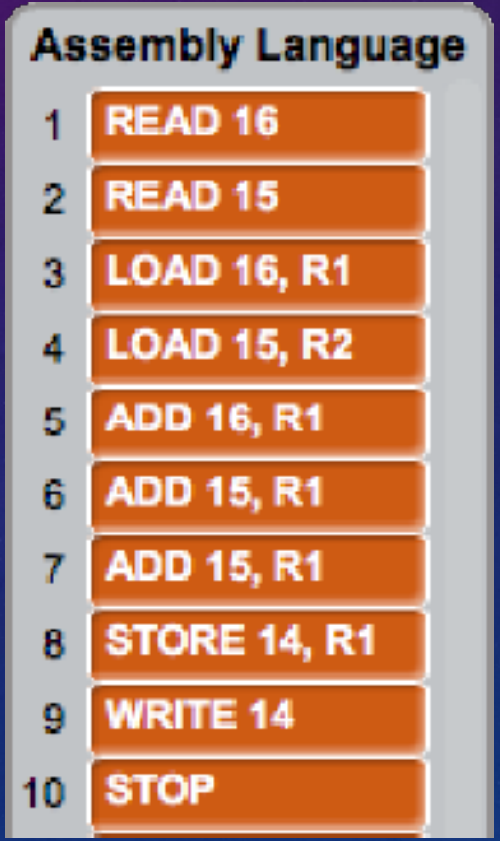
```
perimeter = length + length + width + width
```

1. First, the computer loads the value of the variable `length` into the immediate access store (registers)
2. Next it needs to load in the value of the variable `width`
3. Then it needs to add the two numbers together (twice) – executing the command in the ALU
4. Finally it needs to store the result in the variable `perimeter`

TASK 2: MORE ASSEMBLY PROGRAMMING

- Write the following code →
- Assemble
- Load
- Run
- Watch the FE Cycle in action

- You have 5 minutes



Assembly Language

```
READ 16
READ 15
LOAD 16, R1
LOAD 15, R2
ADD 16, R1
ADD 15, R1
ADD 15, R1
STORE 14, R1
WRITE 14
STOP
```

Python: `perimeter = length + length + width + width`

ASSESSMENT TASK

- Complete the Assessment sheet titled “The CPU and the Fetch-Execute Cycle”
- Put your answers, a, b or c, in the boxes
- Put your name at the top
- You have 5 minutes

ASSESSMENT TASK ANSWERS

1. c
2. a
3. a
4. c
5. b
6. a
7. c
8. c
9. b
10. c

LESSON SUMMARY

You should now be able to:

- Understand what the CPU does and how it works, including:
  - Control Unit
  - Registers (aka “immediate access store”)
  - Arithmetic and Logic Unit (ALU)
- Understand what the Fetch-Execute Cycle is and how instructions are executed in computers
- Be able to write simple assembly language programs

NEXT WEEK

- Malware
  - Viruses
  - Trojans
  - Worms
  - Spyware
  - Adware
  - Ransomware and scareware
- Phishing scams