COMS12200 Introduction to Computer Architecture

Building simple processors

Simon Hollis (simon@cs.bris.ac.uk)

Overview

- Memory as a place to store data and instructions
- Memory addressing
- Instruction storage
- The fetch-execute cycle

COMS12200 Topic 2

MEMORY

Memory

- Memory is simply a place to store information.
- It does not care what the information is, or what it is for. It simply stores.
- Memories allow two basic operations:
  - ***Write***, which allows information to be inserted;
  - ***Read***, which allows information to be extracted.

Memory organisation

- Information in a memory is organised based on a unique address.
- To access or update information, we need to specify this address to a memory, then our information can be returned or changed.
- Addresses are specified as indexes.

Memory addresses

Example of addresses and stored values

Address	Value
0	1
1	82
2	291
3	271
4	22
5	89
6	427

Pop quiz

- On the previous slide, which values were data, and which were instructions?

Instructions in memory

Values can be op-codes, for example

Address	Value (op-code)	Instruction
0	1	‘ADD’
1	2	‘SUB’
2	3	‘MUL’
3	4	‘DIV’
4	1	‘ADD’
5	4	‘DIV’
6	427	?

Instructions + data

- A single memory location can combine both an instruction and some data.
- This can be useful for e.g. constant-based operations
  - Consider **`ADD1`**
  - Or something that loads a constant e.g. **`MOVE2`**

Instructions + data

- A single memory location can combine both an instruction and some data.
- Example: **`ADD1`**
  - **`ADD`** → ‘1’
  - 1 → ‘1’
  - So, ADD1 could be expressed as ‘11’
    - Note that ‘11’ is not necessarily ‘eleven’

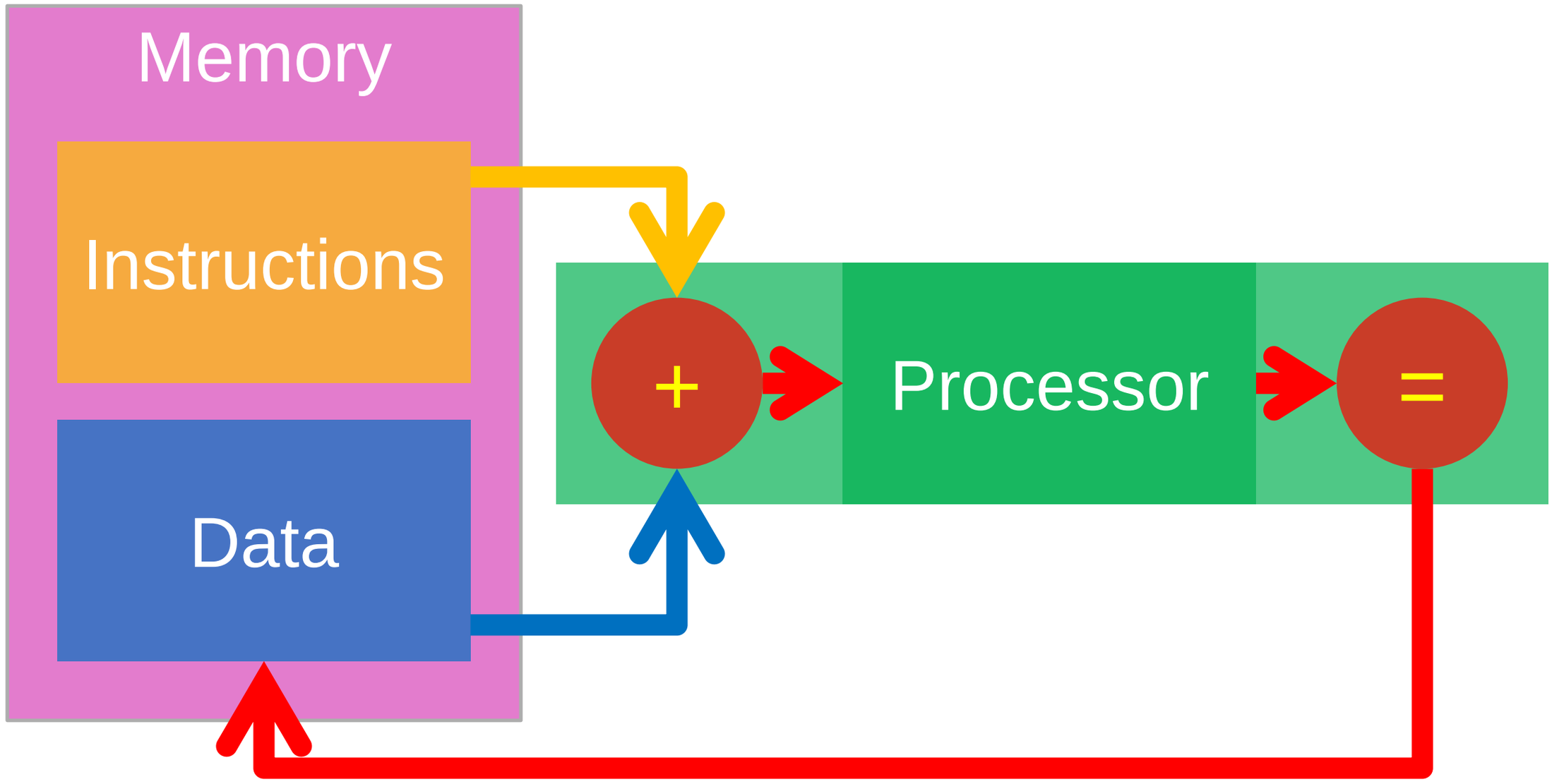
COMS12200 Topic 3

THE “FETCH-EXECUTE” CYCLE

Doing some work

- OK, so we have all these data and instructions, can we put them to work and do something useful?
- What we need to do is feed them in to a processing unit in the right way:
  - An instruction specifies functionality;
  - Some data specifies input(s).
- Then, we will get a result, which we can store as data.

Recap of the flow

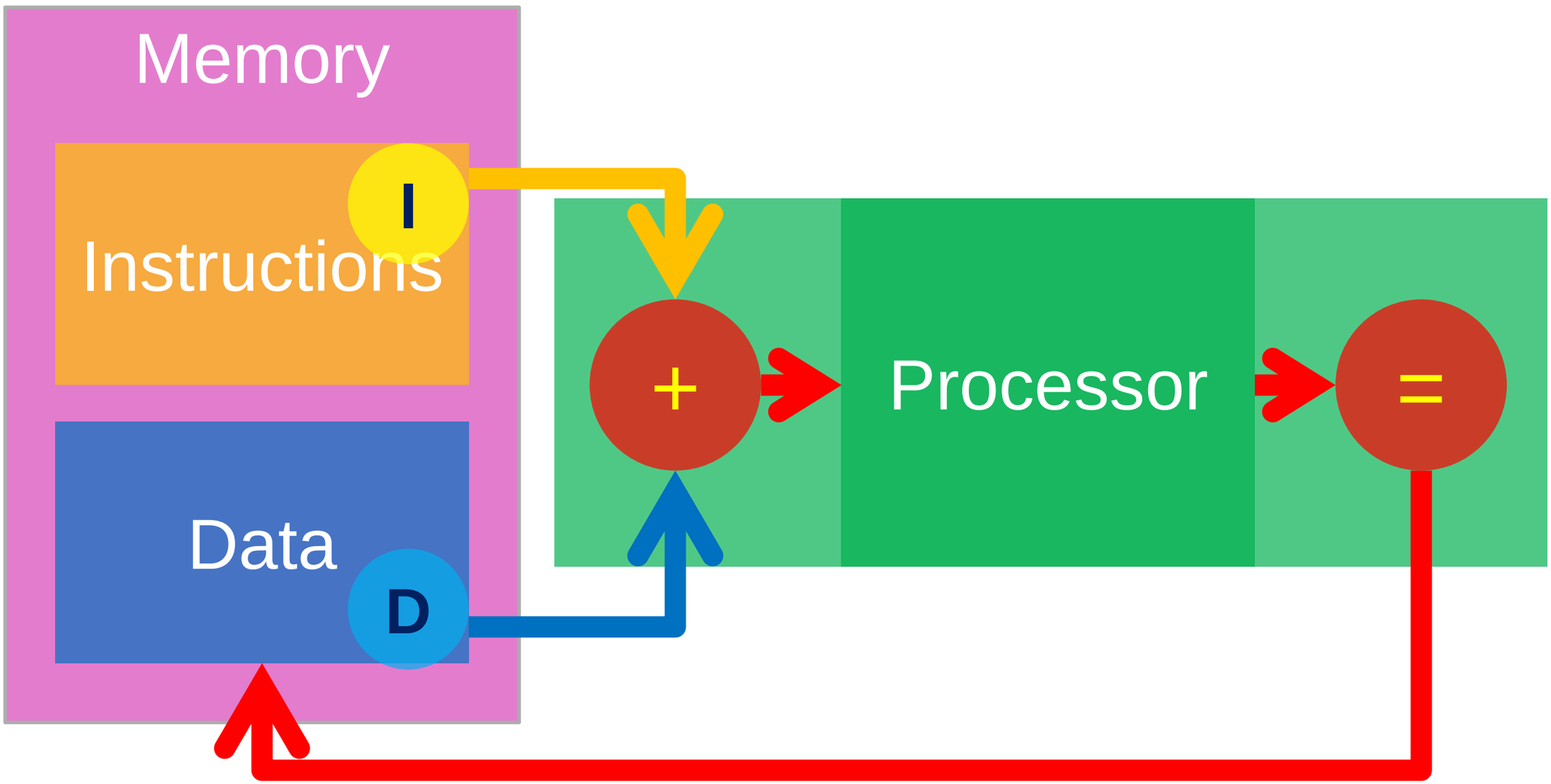


Fetch-execute

- In all modern processors, the necessary instructions and data to do work are gathered in what is called the “fetch-execute cycle”.
- There are two stages to this cycle:
  1. Fetching of instructions and data;
  2. Execution of the instruction and creation of a result.

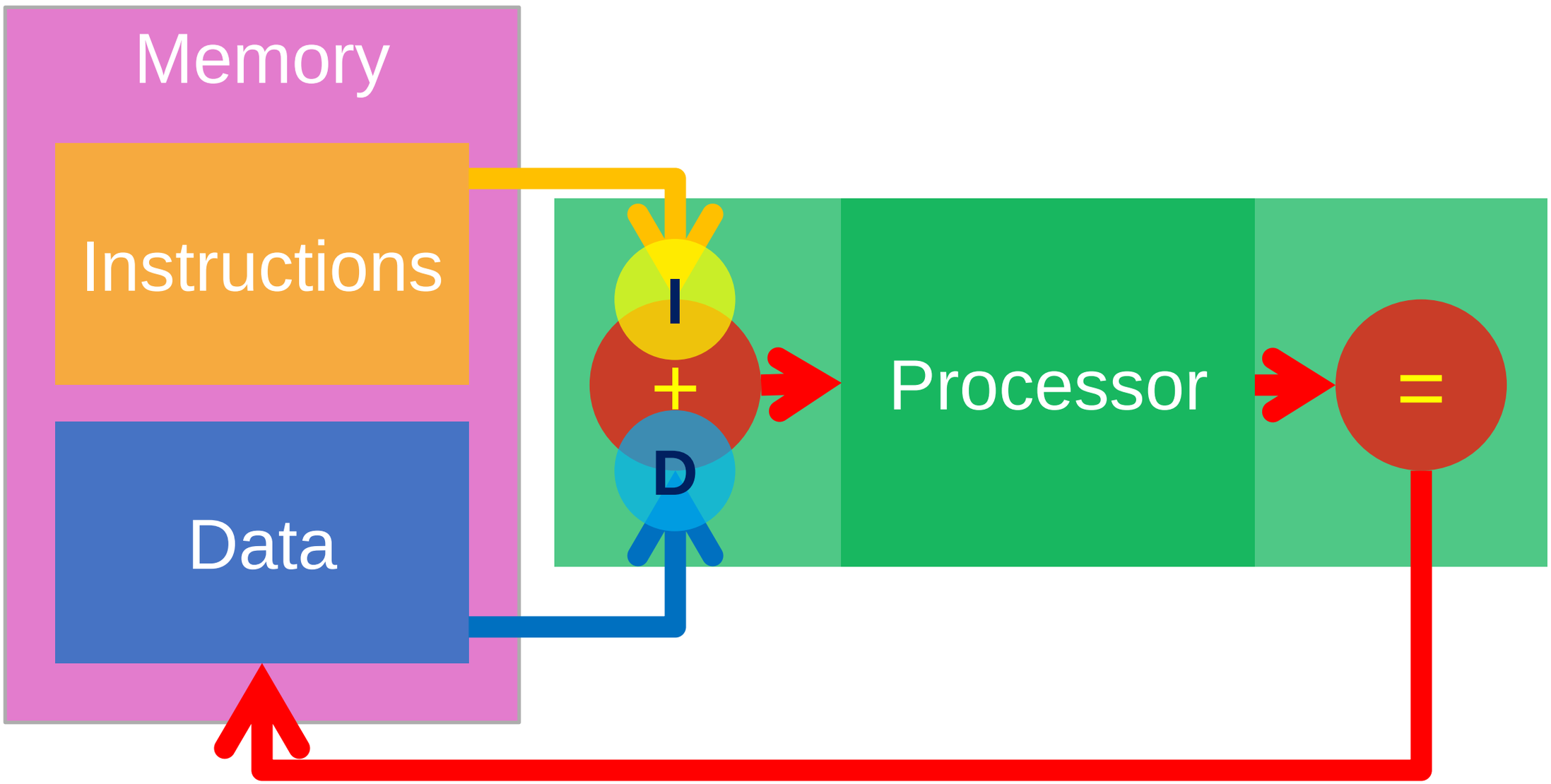
The “fetch”

An instruction and data are loaded into the processor.



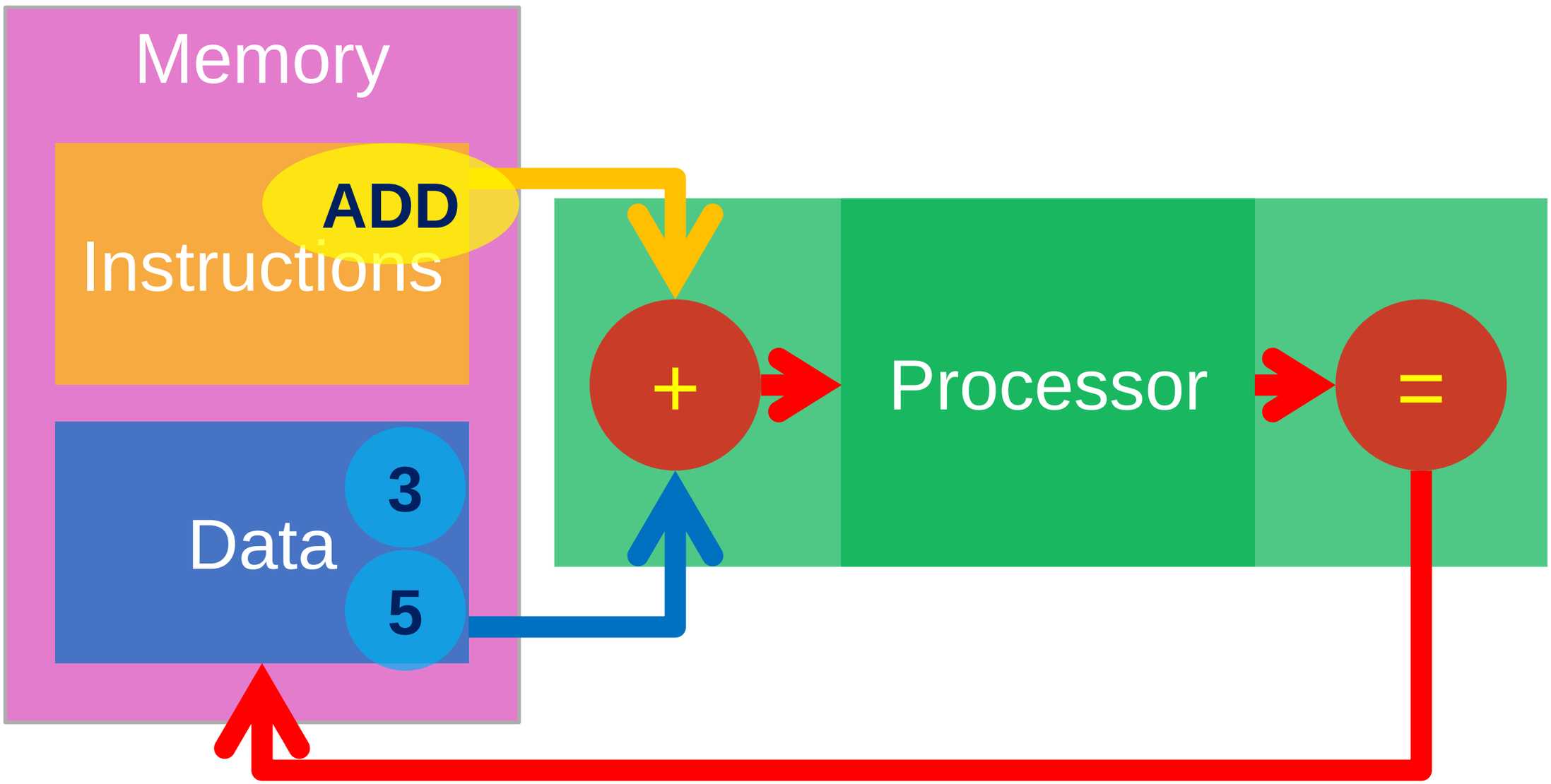
The “execute”

The instruction is interpreted and operated on the data, producing a result



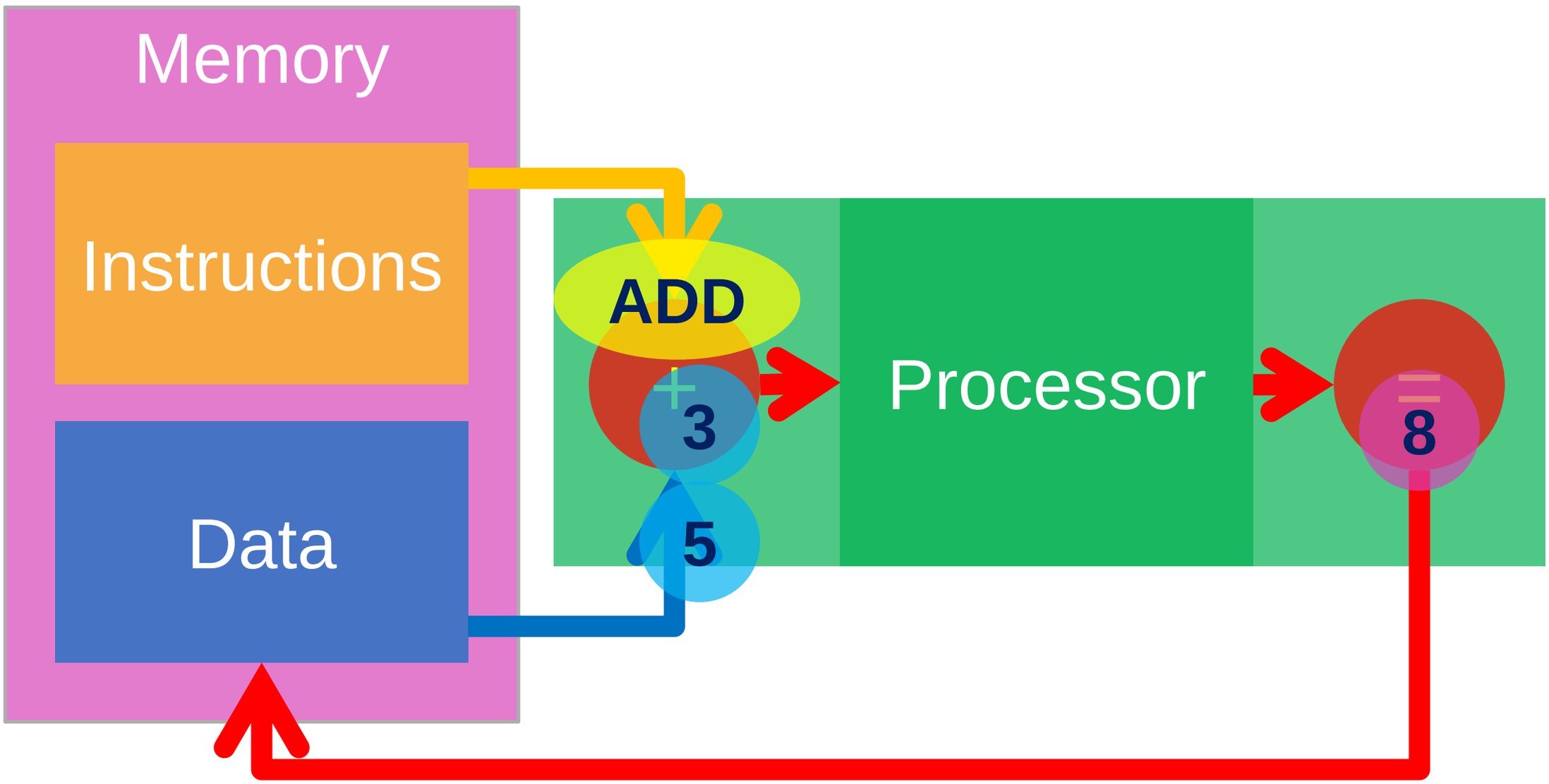
Example: the “fetch”

Here’s an example of how it might work in practice.



Example: the “execute”

Here’s an example of how it might work in practice.



About that result

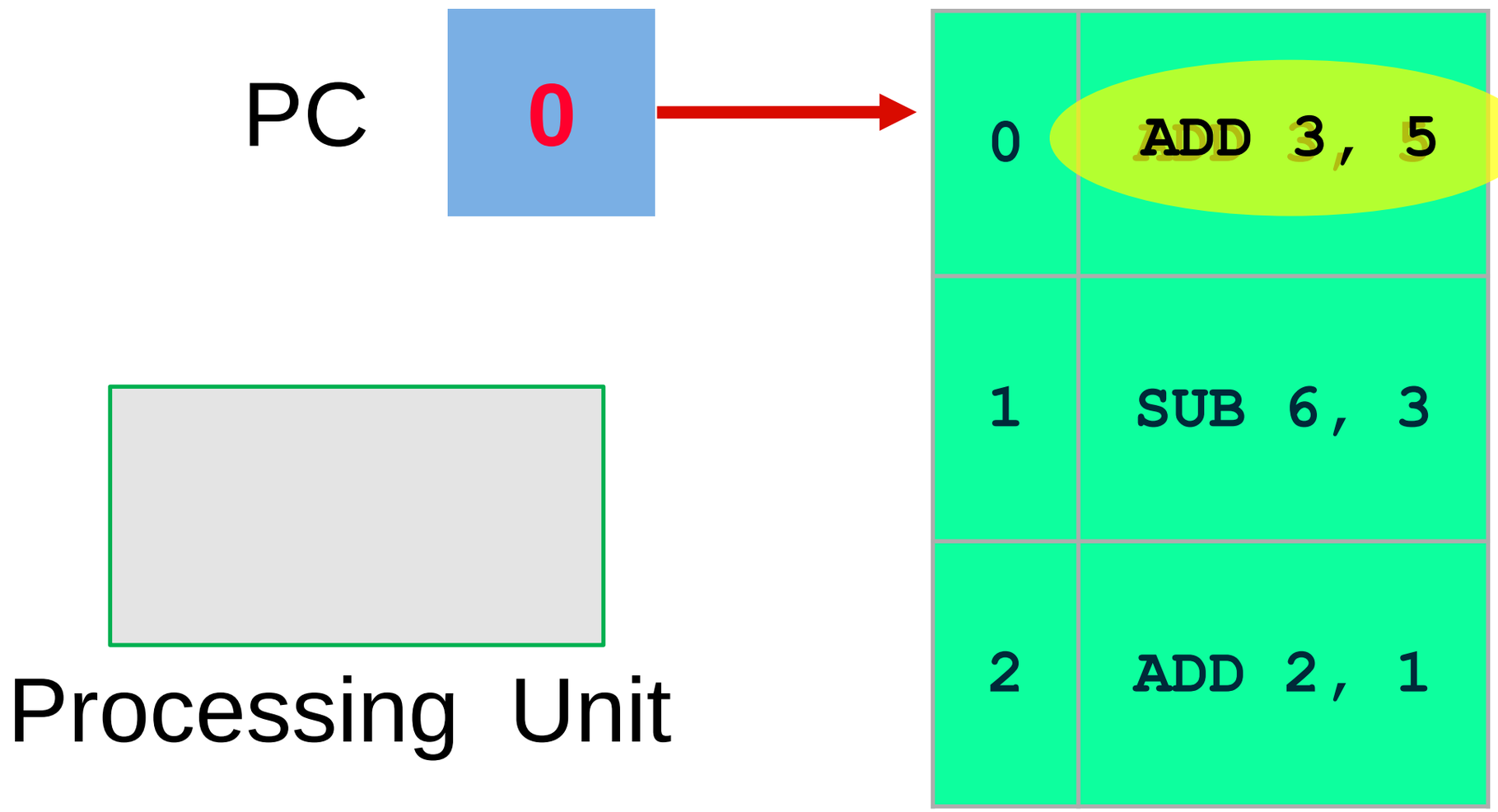
- Did you notice that the generated result went back into memory.
  - This is now ‘value added data’!
- The operation of saving the result is called ‘write-back’.
  - In many machines, this is considered a separate operation.
  - This leads to an extension of the paradigm: “*Fetch, execute, write-back*”
    - This is the method adopted by most real processors. More in COMS35102.

The Program Counter

- How do I know what instruction to fetch?
- Most machines have a dedicated hardware unit called a *Program Counter (PC)*.
- The PC keeps a pointer to the next instruction to be executed. Once the instruction has been executed, it increments and points to the next instruction.

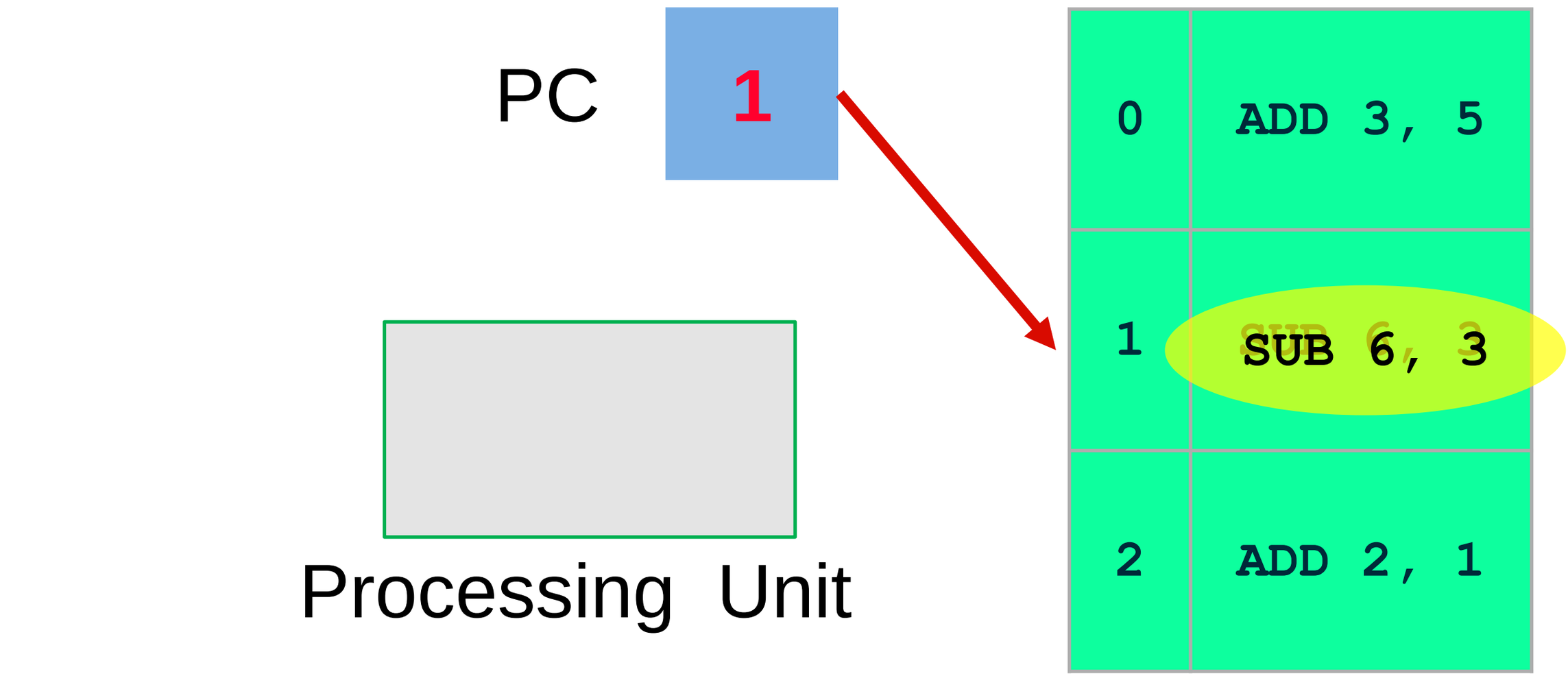
PC Operation

- Under normal program flow, the PC increments by one instruction address per instruction fetched.



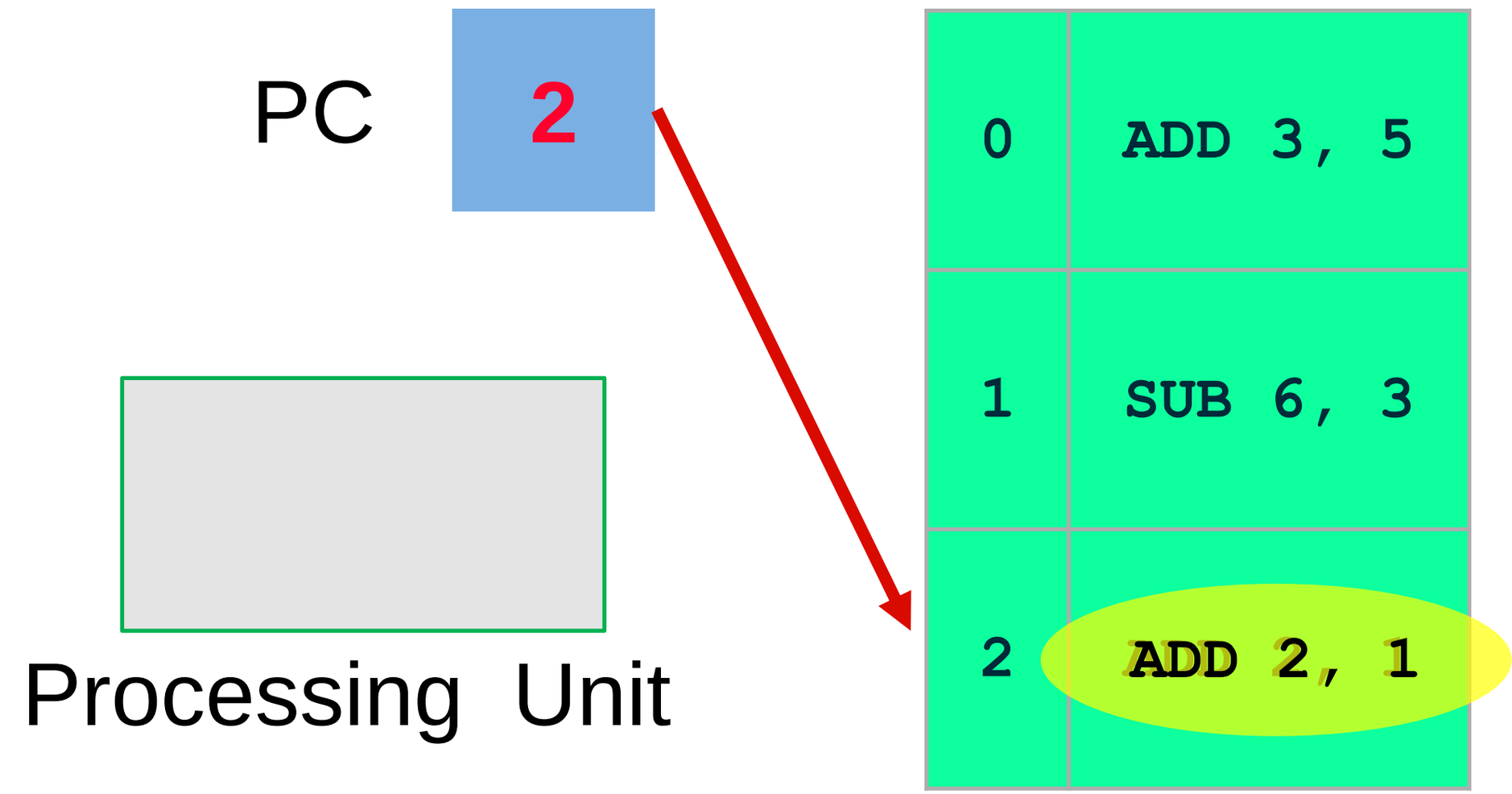
PC Operation

- Under normal program flow, the PC increments by one instruction address per instruction fetched.



PC Operation

- Under normal program flow, the PC increments by one instruction address per instruction fetched.



Summary of today

We have introduced three fundamental concepts of computer architecture:

1. The distinction between data and instructions;
2. How data and instructions are stored in memory;
3. The *Fetch-execute cycle*, including the *Program Counter* and its operation.

Next lecture

- In the next lecture, we will start looking at the internals of the ‘processor’ element of the computer.
- We’ll see what happens to the instructions and data between entering and existing the processor.
- We’ll also outline some very popular paradigms for building a processor.