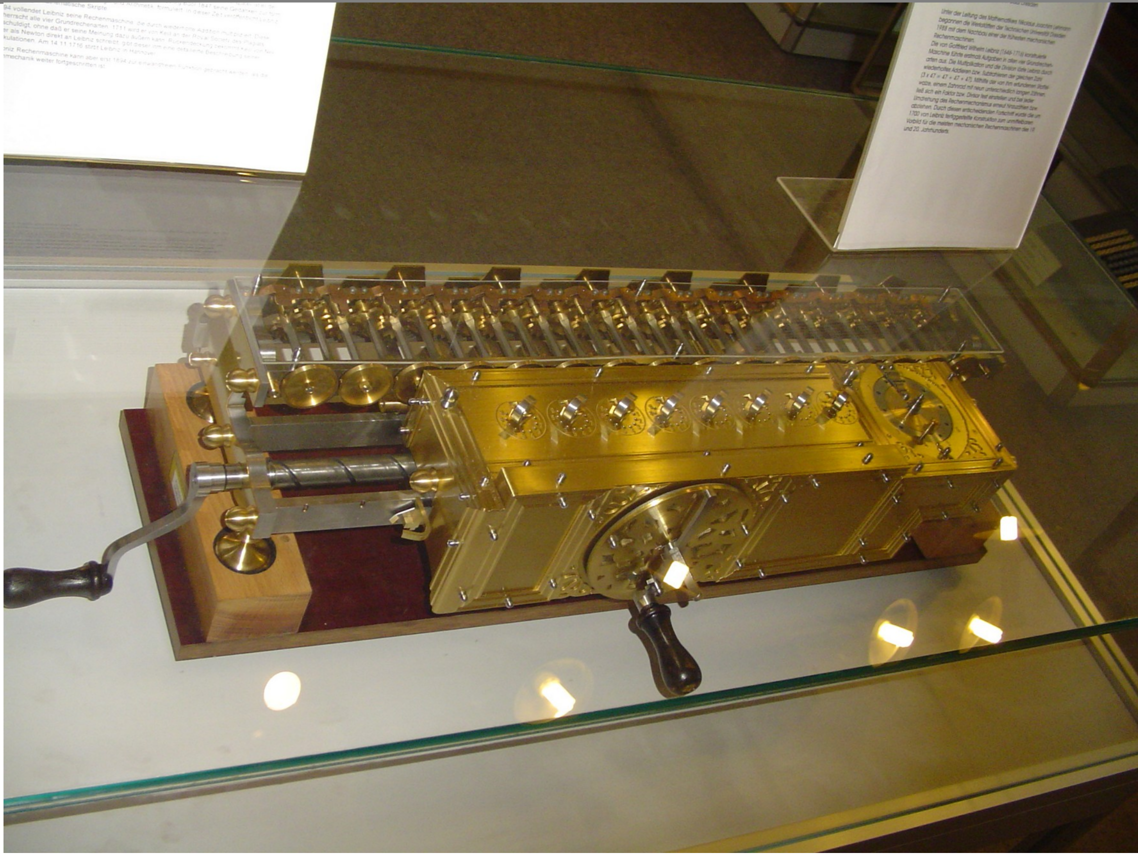
CHAOS
The von Neumann architecture

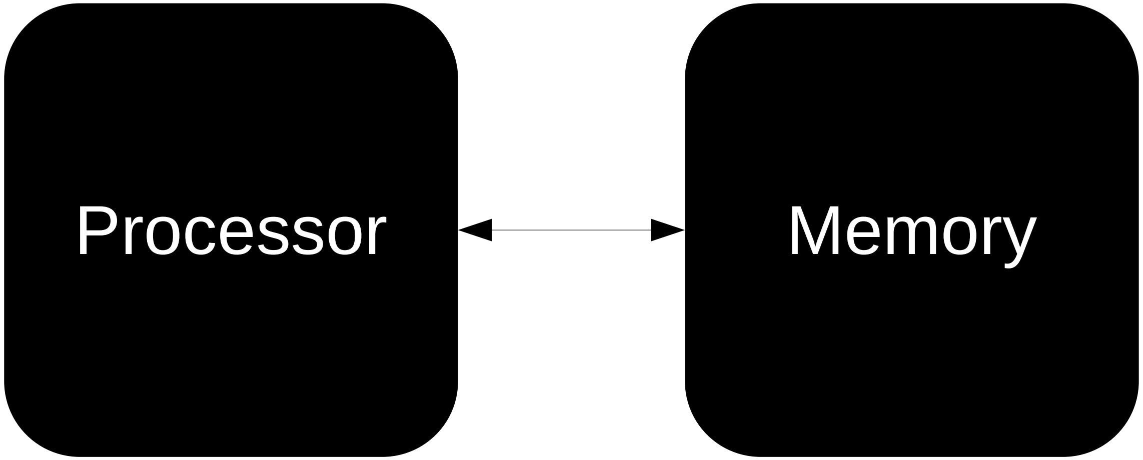
Adam Sampson
School of Design and Informatics
Abertay University

Recap: history of computing

- 17th century: the first mechanical calculators
  - e.g. Leibniz – calculus, binary maths, symbolic logic...
- These have:
  - a **store** of numbers
  - **operations** the user can perform – e.g. **+**



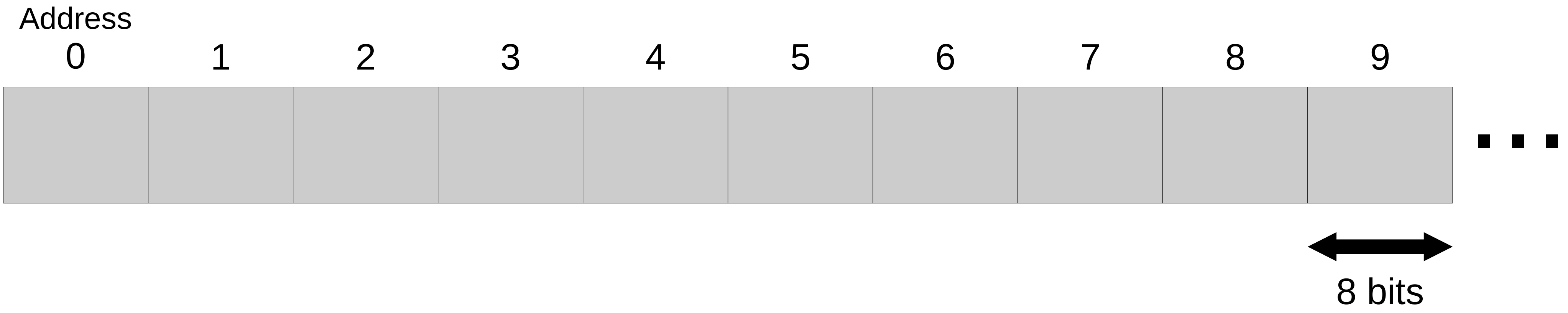
A calculator



Processor

Memory

Memory



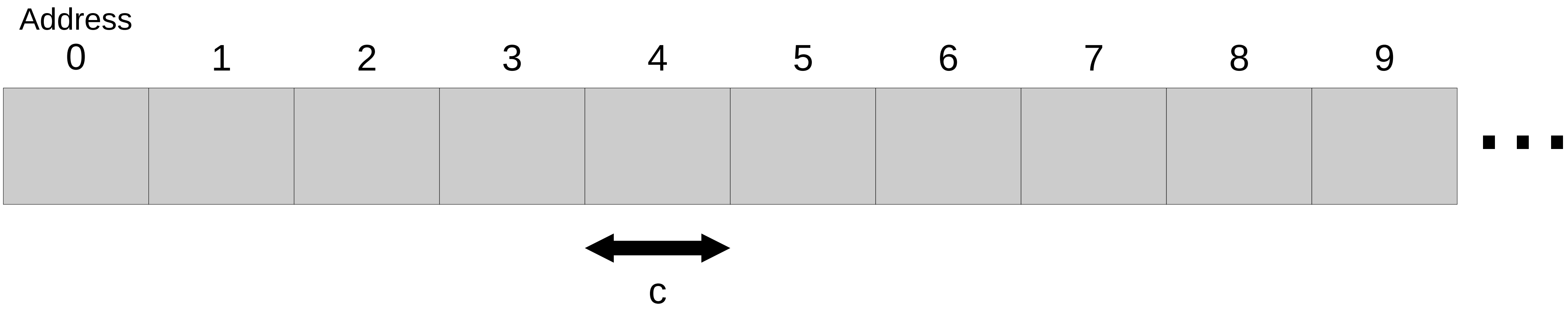
Address

0 1 2 3 4 5 6 7 8 9

...

8 bits

Memory



Address

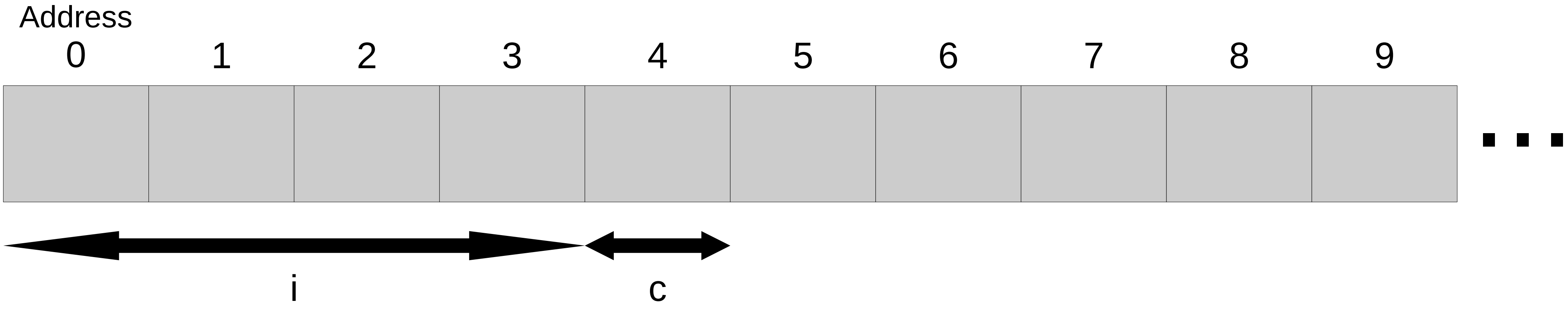
0 1 2 3 4 5 6 7 8 9

...

c

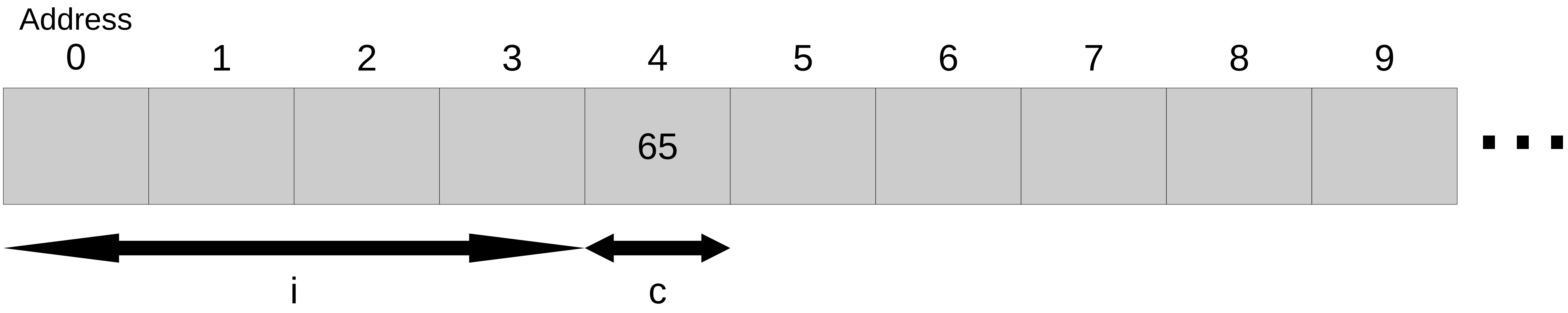
```
char c;
```

Memory



```
char c;
int i;  // 32-bit integer
```

Memory



Address

0 1 2 3 4 5 6 7 8 9

65

...

i

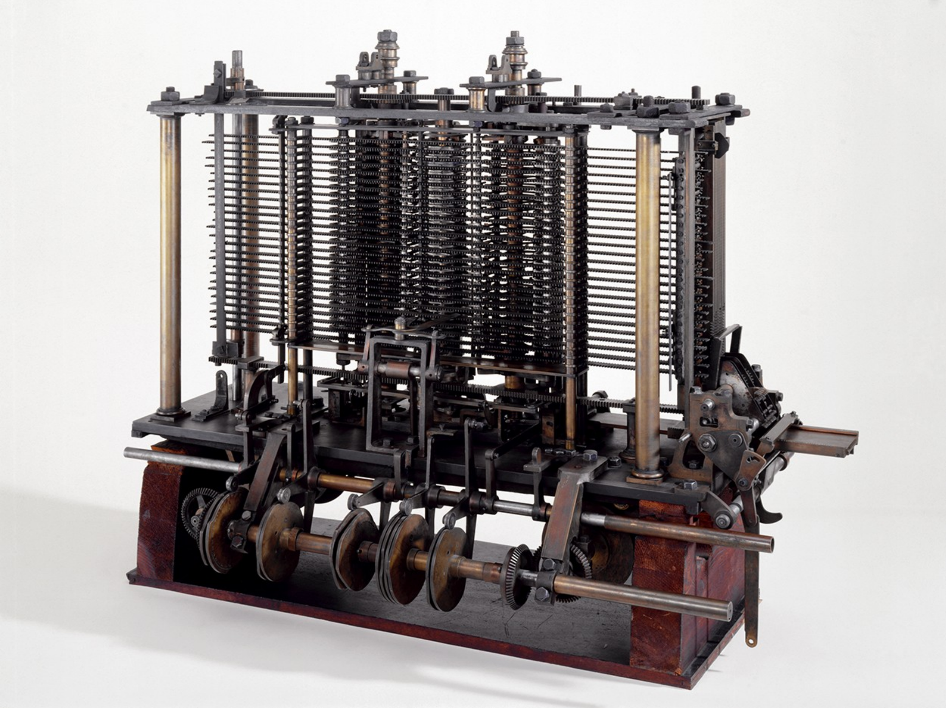
c

```
char c;
int i;  // 32-bit integer

c = 65;
```

Recap: history of computing

- 19th—20th century: automated computing
  - Babbage – good ideas, thwarted by political and business problems
  - Telephone technology
  - Unit record equipment – IBM, Hollerith, etc. – very successful by WW2
- These machines had a store of data, plus a separate **sequence of operations**



A very brief history of computing

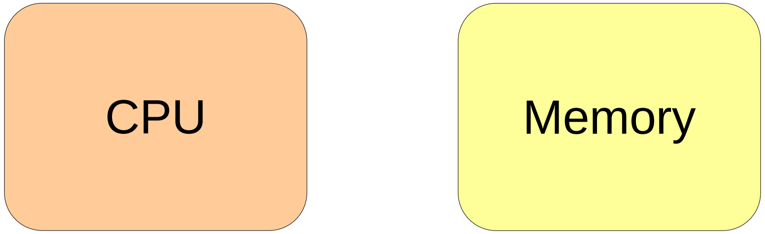
- But we want a **universal** computer – one that can solve any problem
  - … given enough memory and time
  - No rewiring necessary
- Around 1935, several people simultaneously have a really good idea...
  - Alan Turing
  - Konrad Zuse
  - John von Neumann
- … the **stored-program** computer

Stored-program computing

- The binary number 01000001 stored in a computer's memory might represent...
  - the integer **65**...
  - … or the floating-point number **3.72**...
  - … or the letter **A**...
  - … or the colour **dark purple**
- The stored-program idea was that it could also represent an **instruction**
  - e.g. “add A to B”, or “store X in location 5”

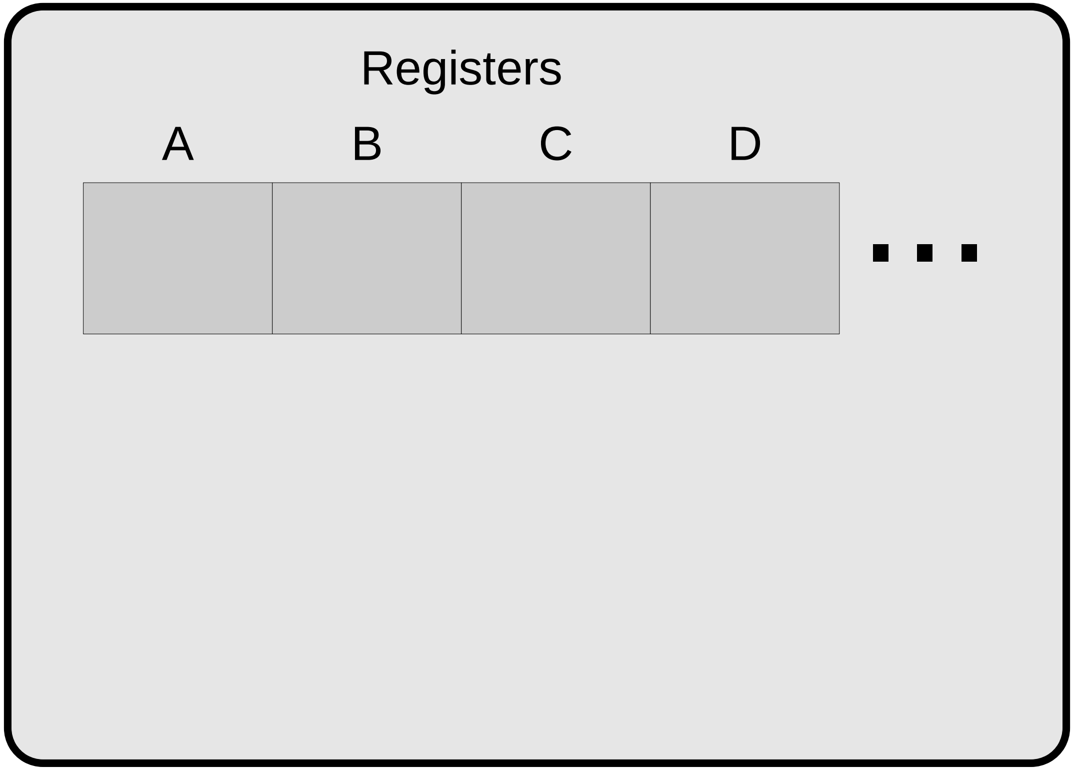
A simple computer

- The **processor** (or **central processing unit** or **CPU**) controls the rest of the machine
- A program is a list of instructions for the CPU to follow, stored in memory
- CPU can **load** data from memory, and **store** data into it



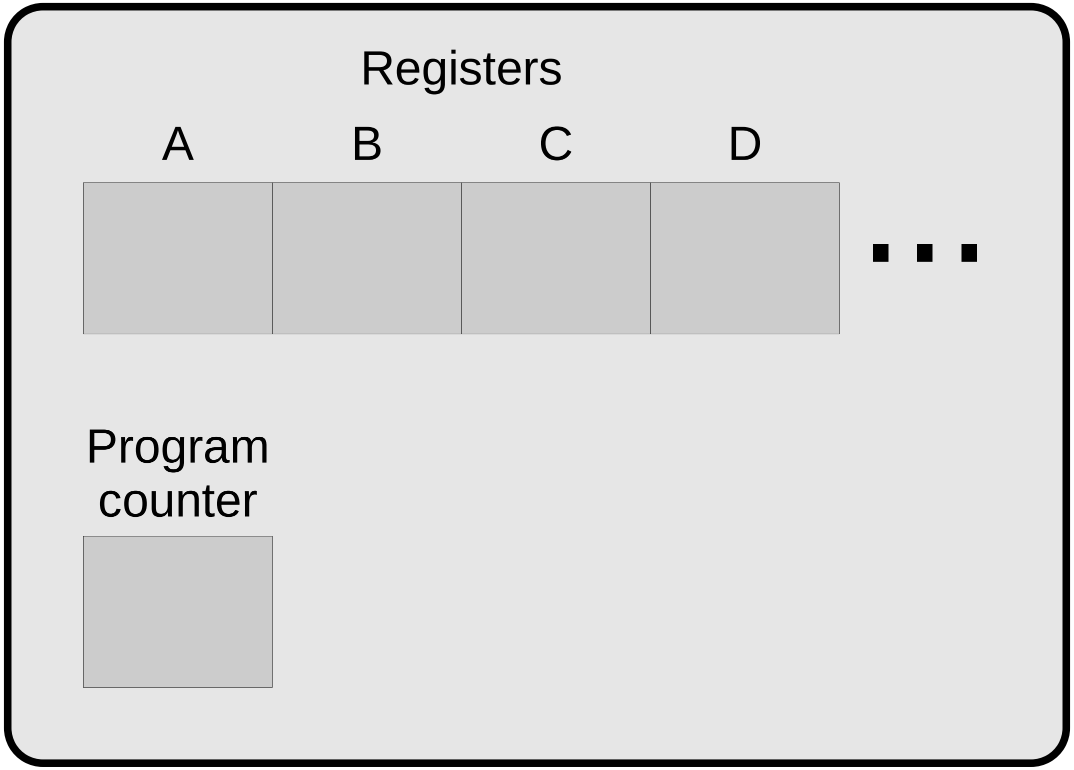
What's inside the processor?

What's inside the processor?



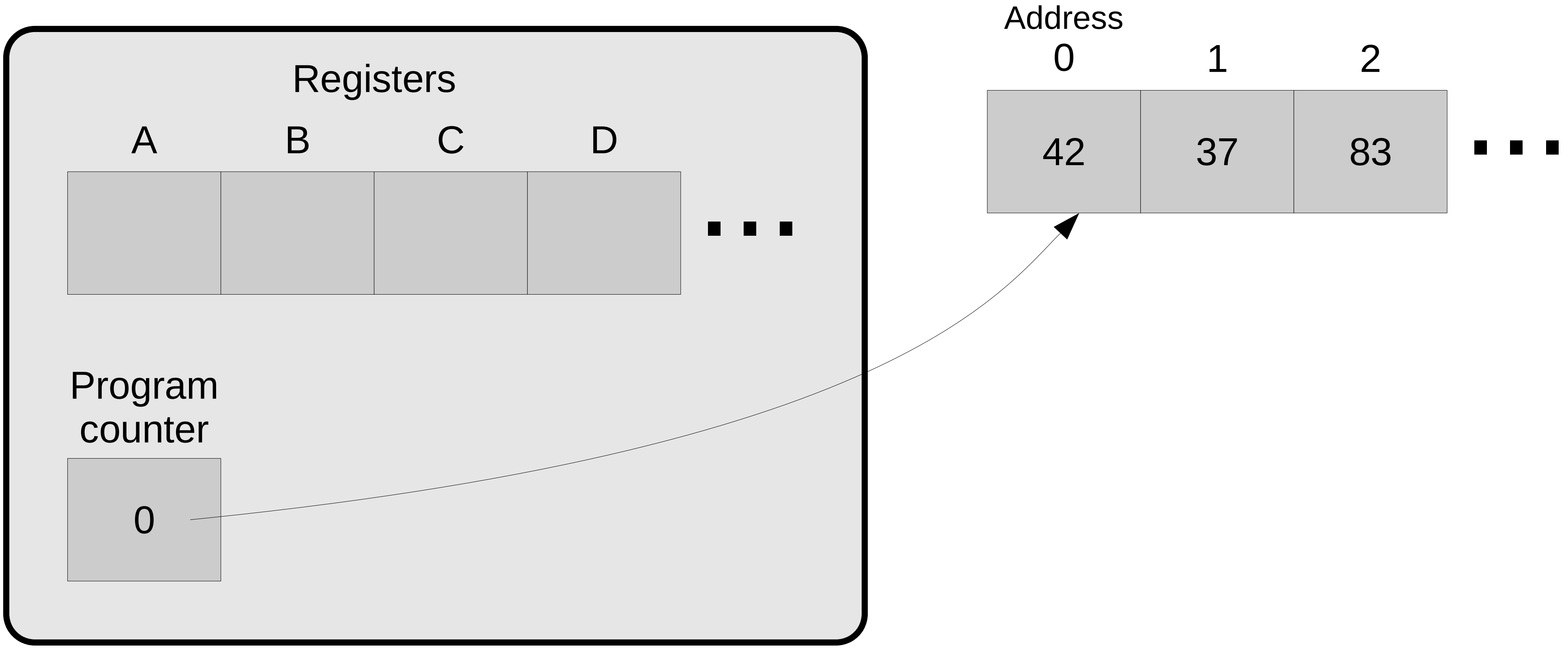
Some **registers** – fast temporary storage for use during computation.

What's inside the processor?



The **program counter** says where in memory the next instruction is stored.

Running a program



Registers

A B C D ...

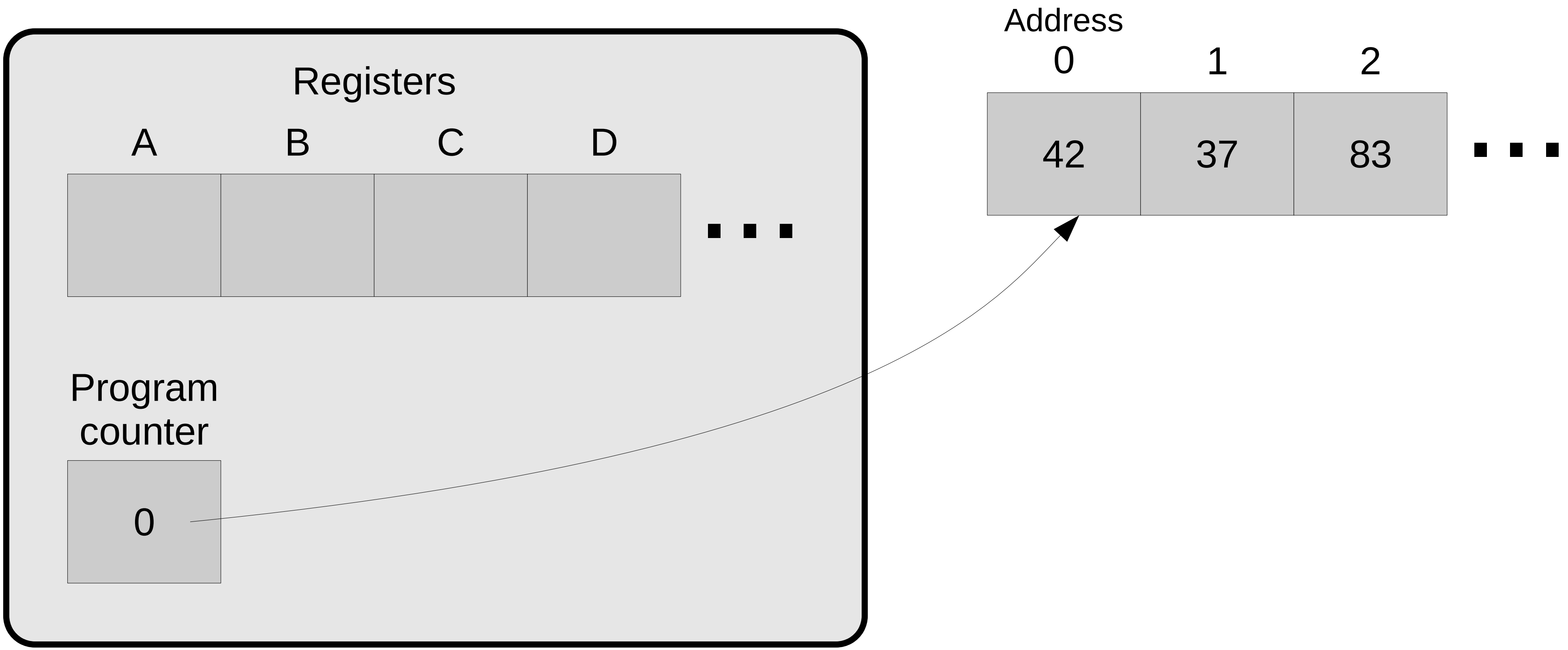
Program counter

0

Address

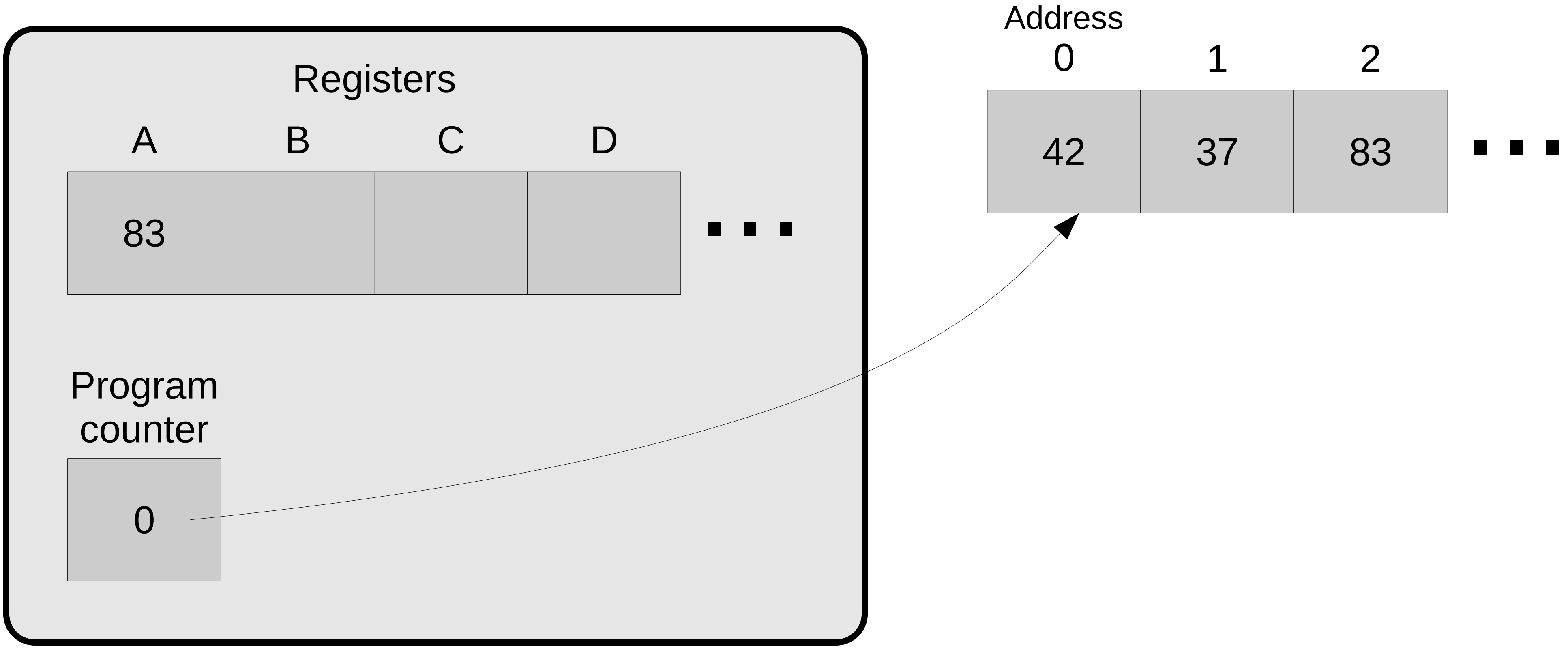
0	1	2	
42	37	83	...

Running a program



Fetch and decode instruction:
suppose 42 means “load register A from memory address 2”

Running a program



Registers

A	B	C	D	...
83				

Program counter

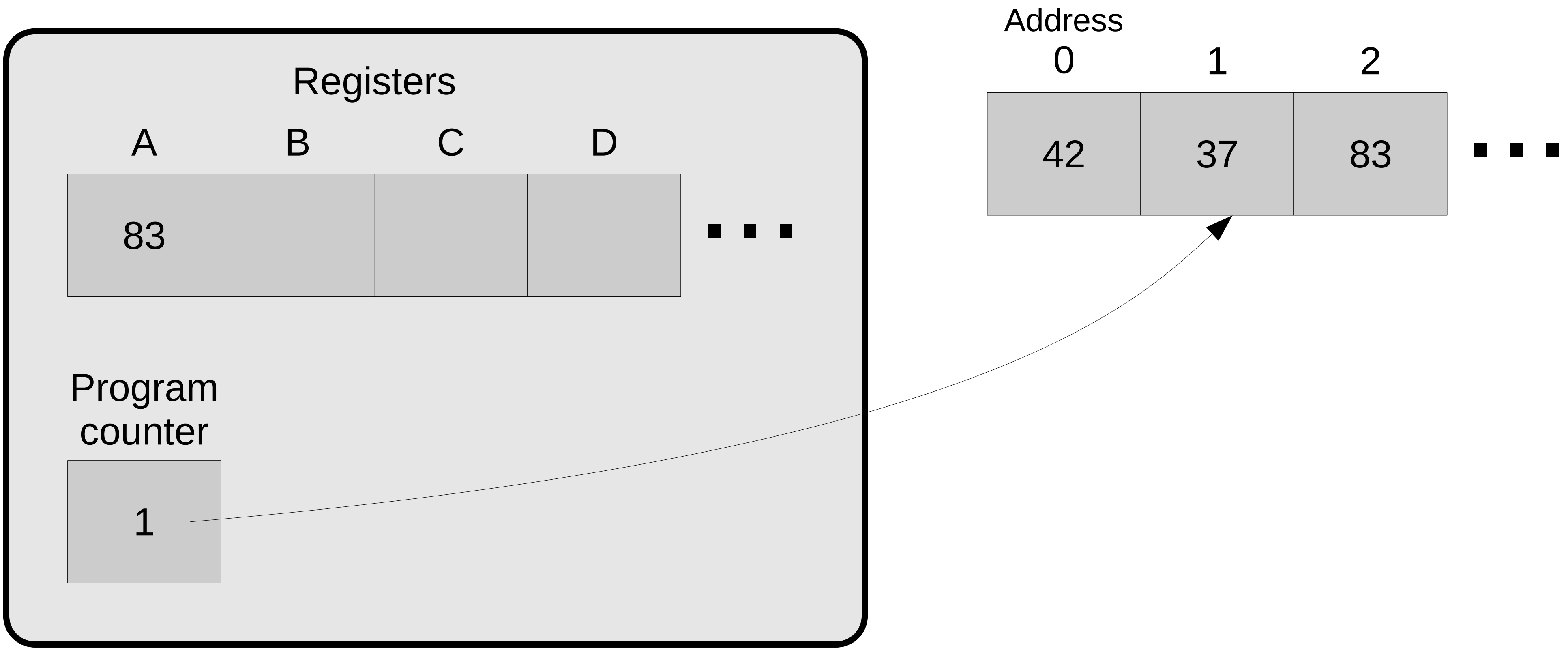
0

Address

0	1	2	...
42	37	83	

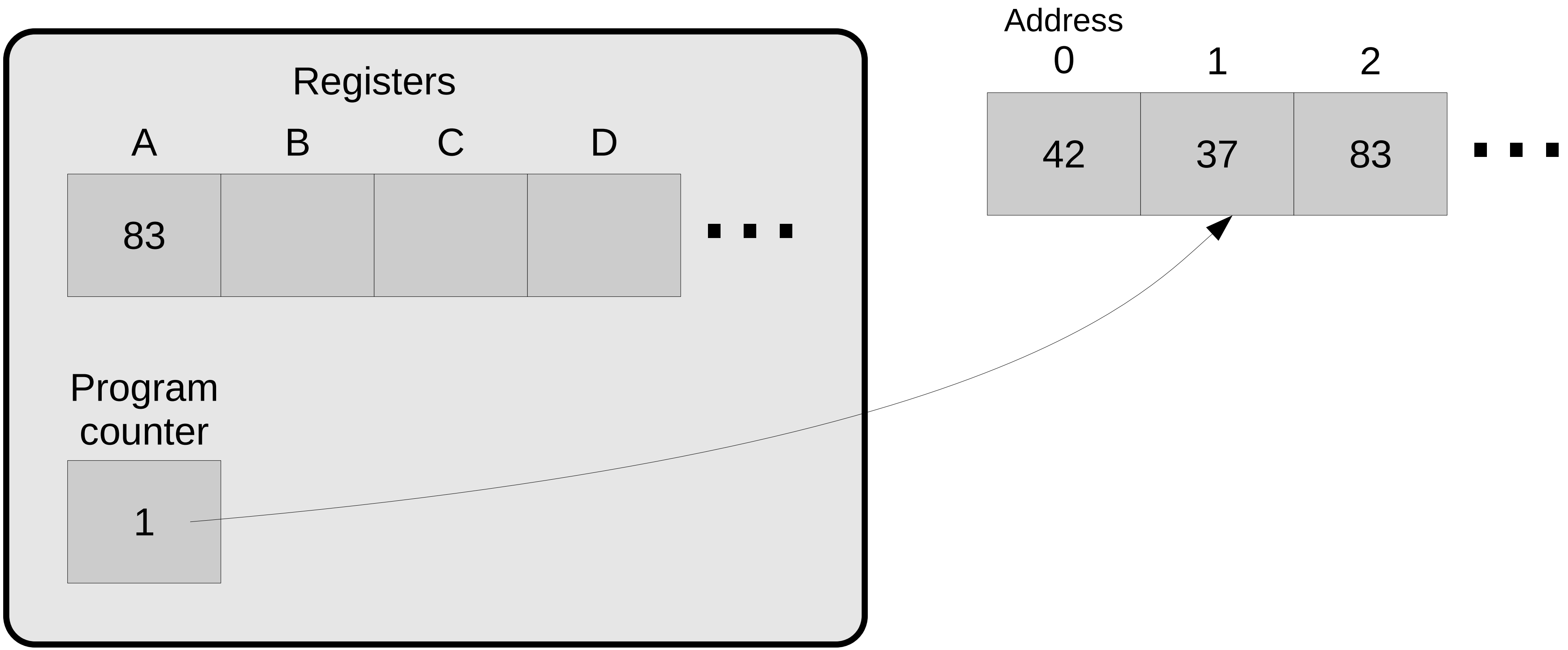
Execute instruction

Running a program



Increment program counter

Running a program

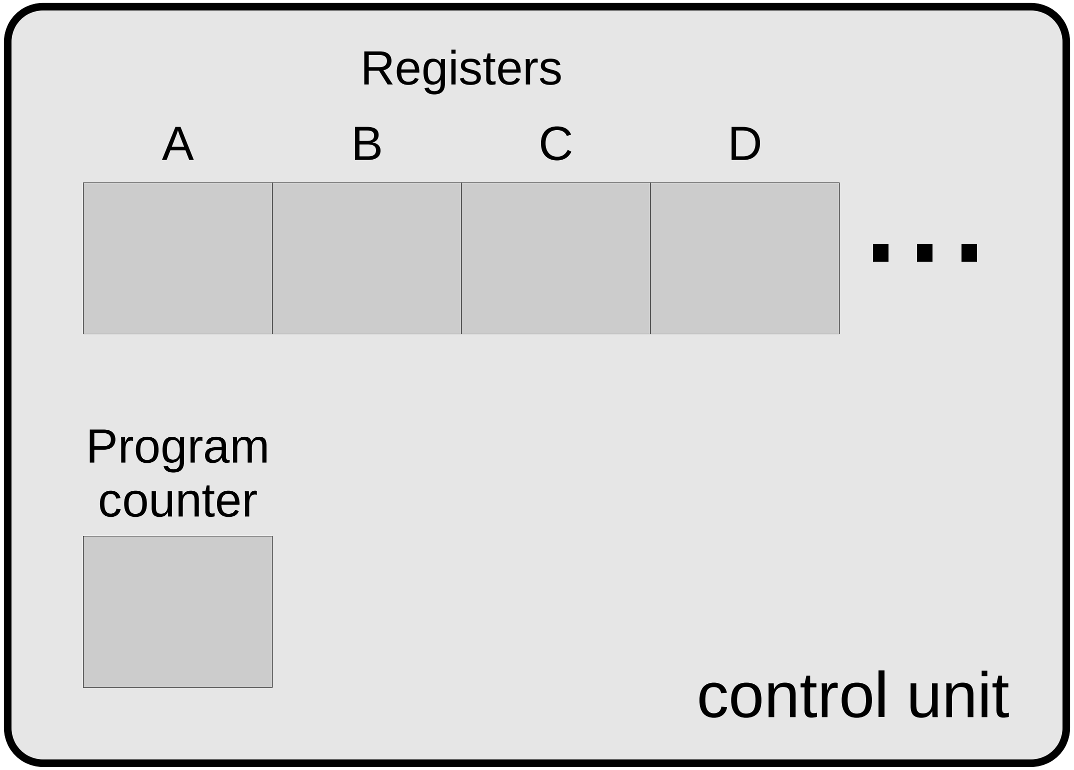


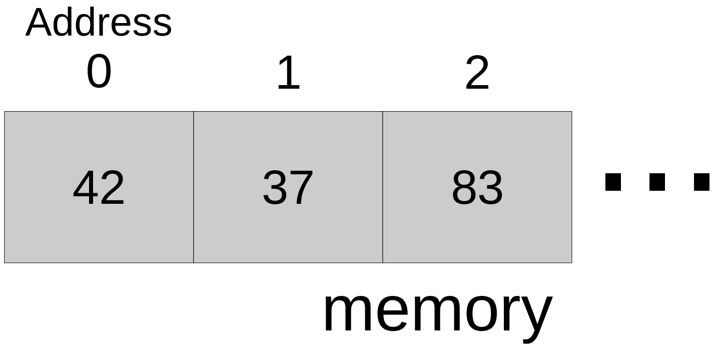
Fetch and decode instruction:
37 means …

Doing arithmetic

- Just moving data around isn't very interesting – we need to perform **operations** on it too
  - e.g. arithmetic (**`+-/*`**), comparisons...
  - “add A to B, put the result in A”
- This is done by an **arithmetic logic unit** (**ALU**)

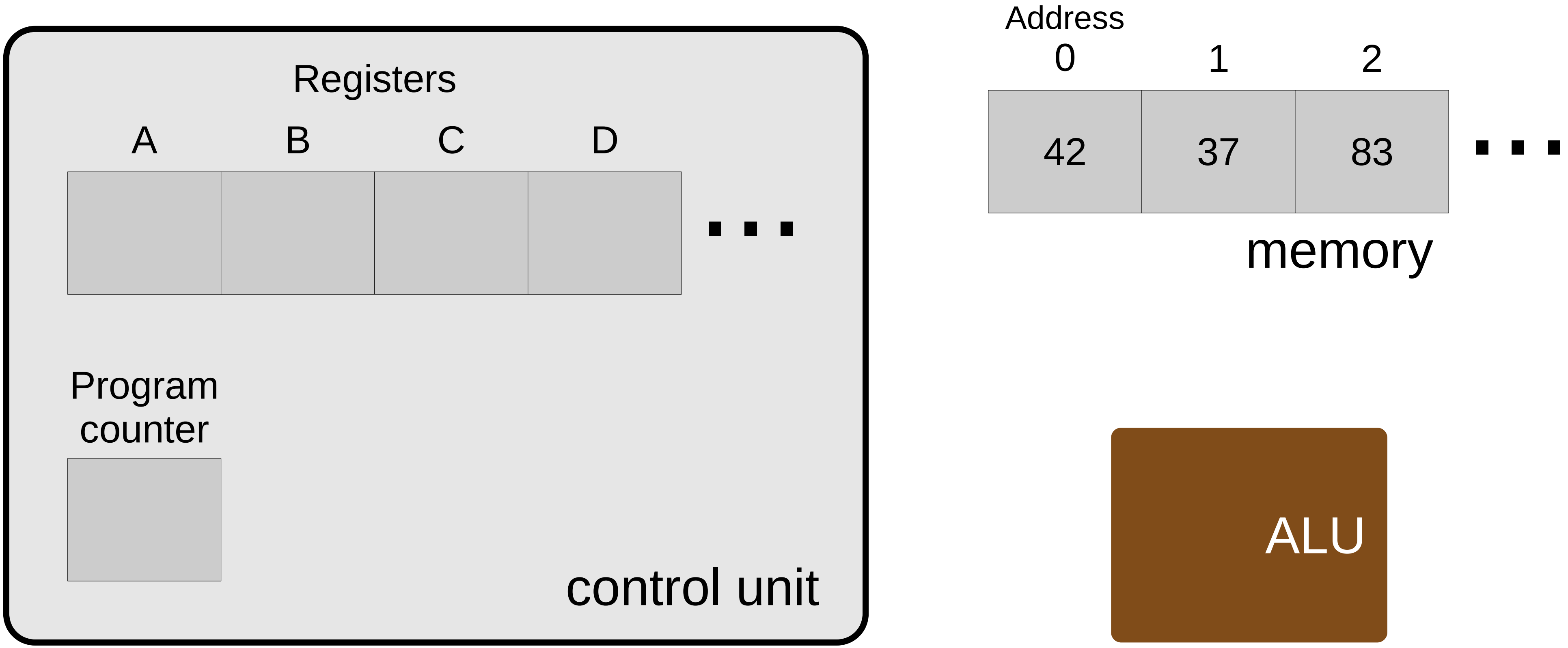
What's inside the processor?





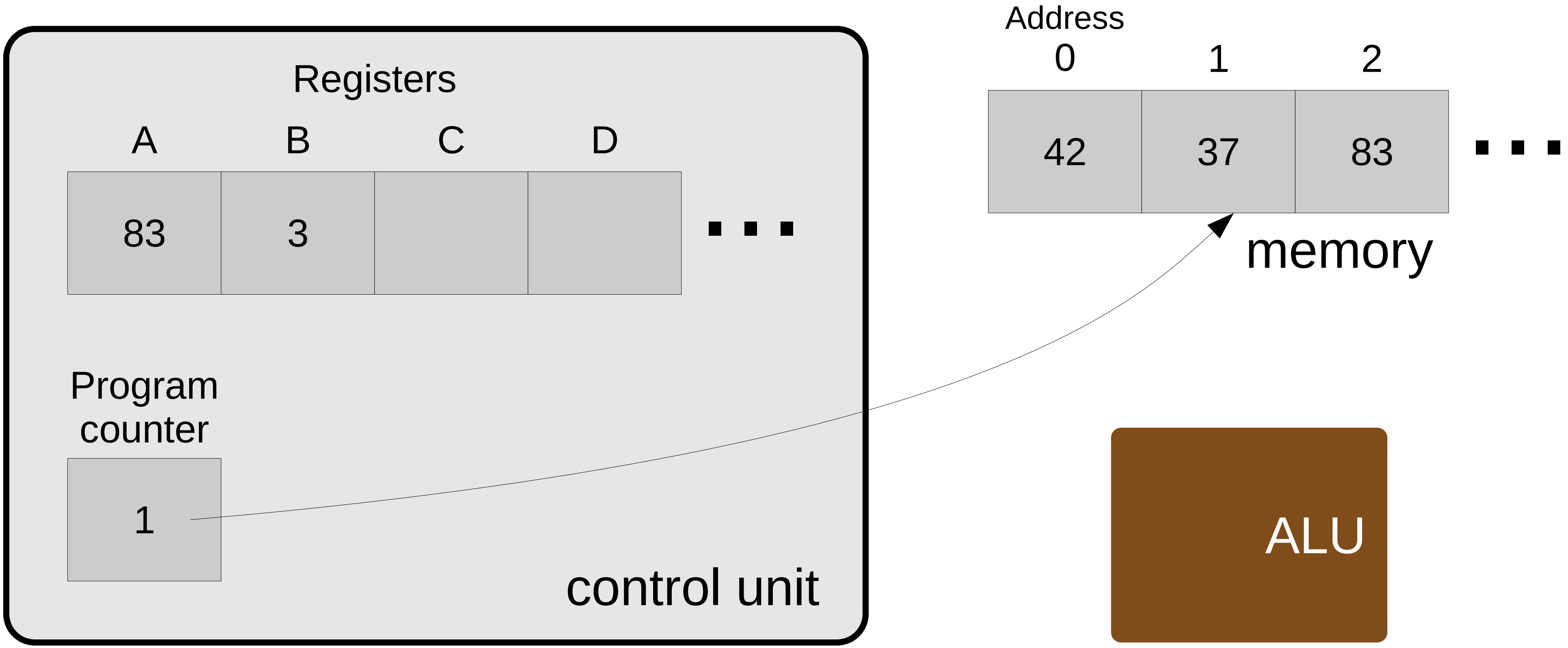
The bit we've described already is the **control unit**, which decodes instructions and controls the rest of the computer.

What's inside the processor?



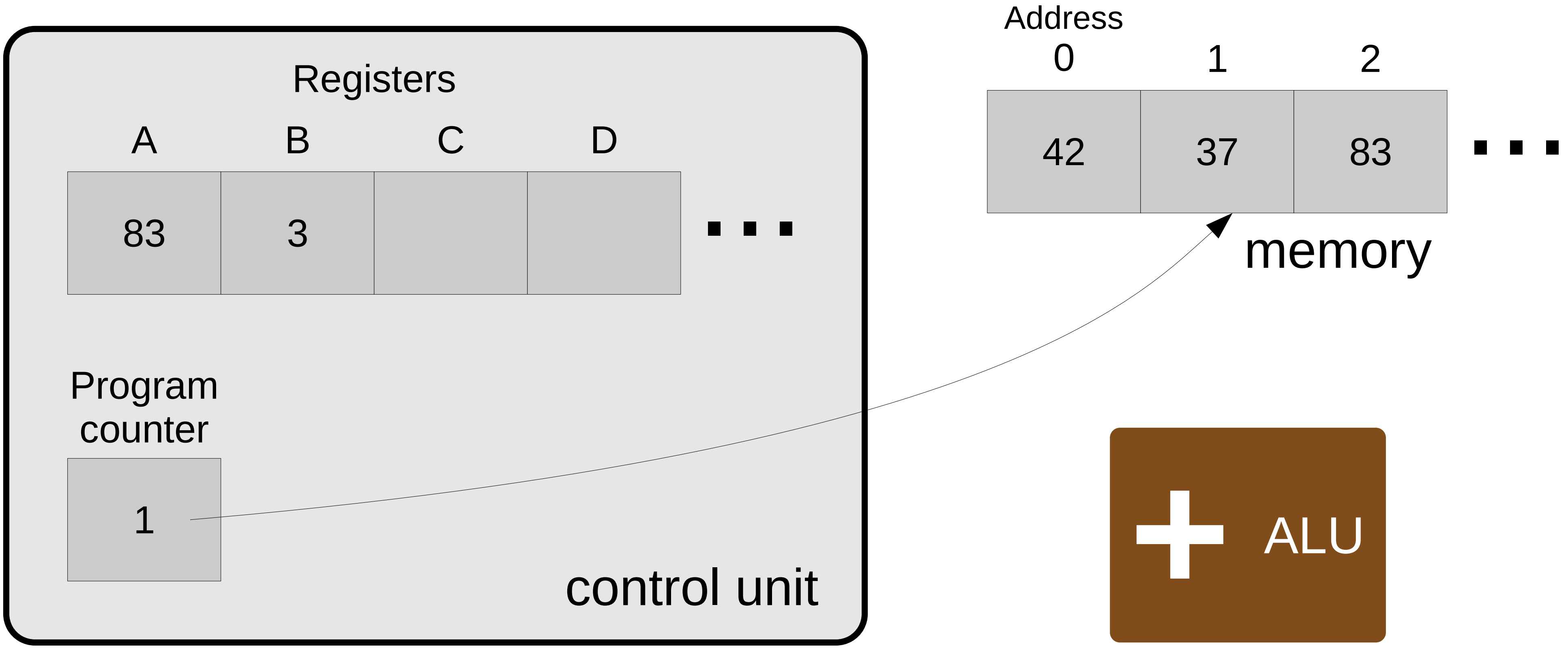
The **ALU** performs arithmetic operations.

Running a program



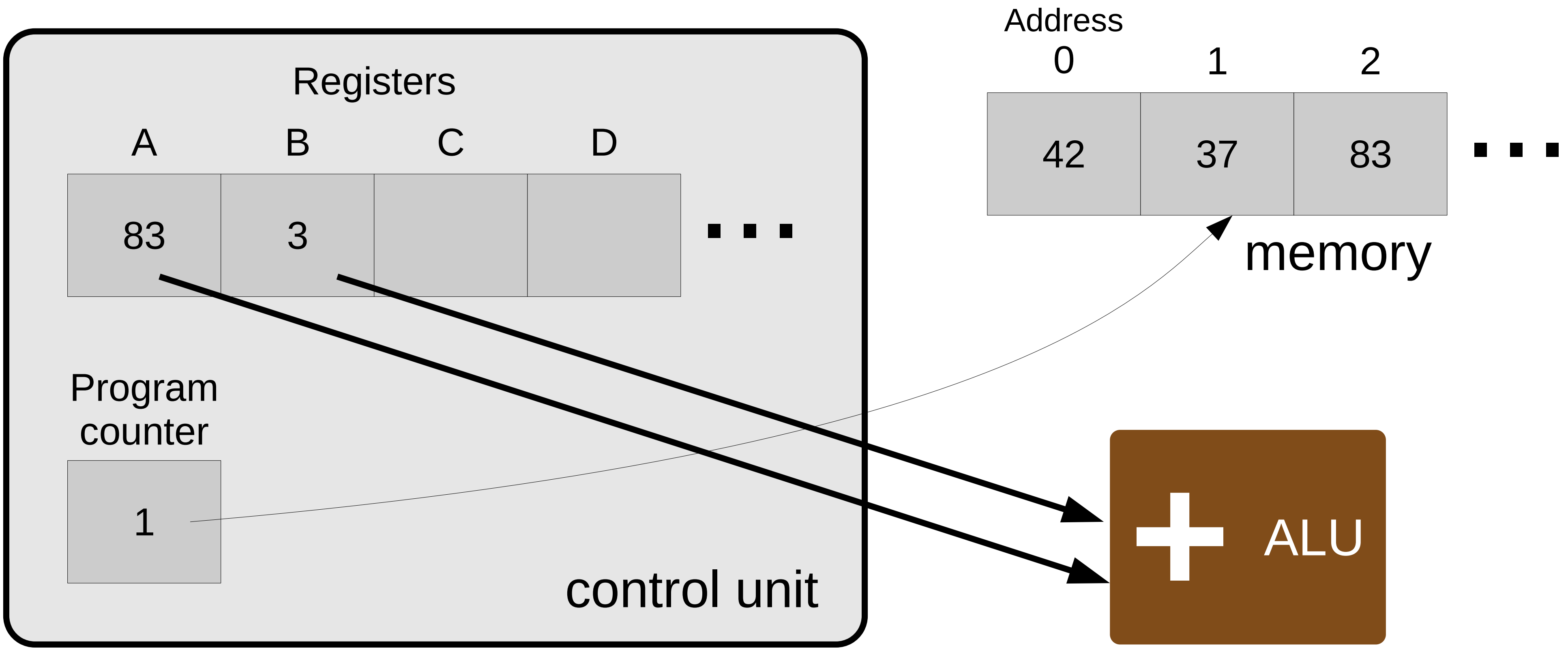
Fetch and decode instruction:
Suppose 37 means “add A to B, store result in C”

Running a program



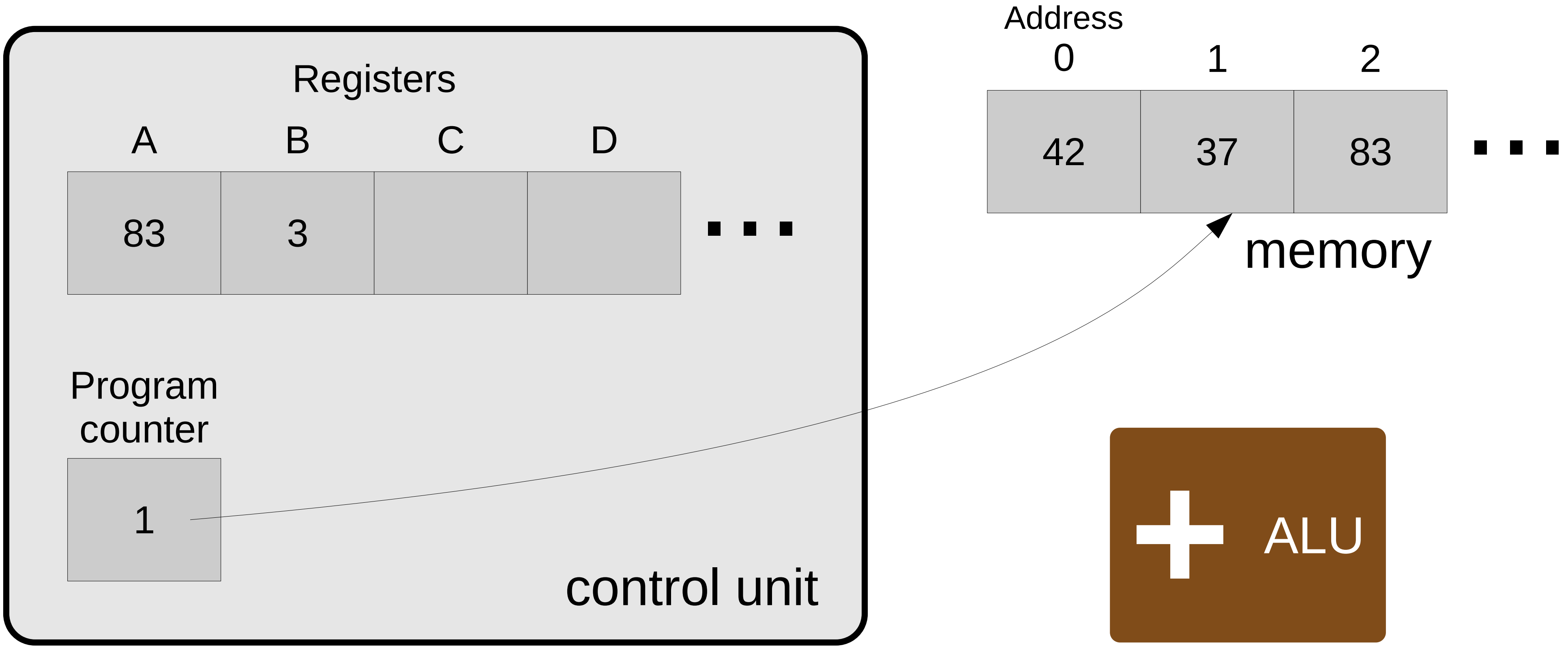
Fetch and decode instruction:
Control unit tells the ALU to perform an addition...

Running a program



Fetch and decode instruction:
… then gives it the two input values...

Running a program



Registers

A	B	C	D	
83	3			...

Program counter

1

control unit

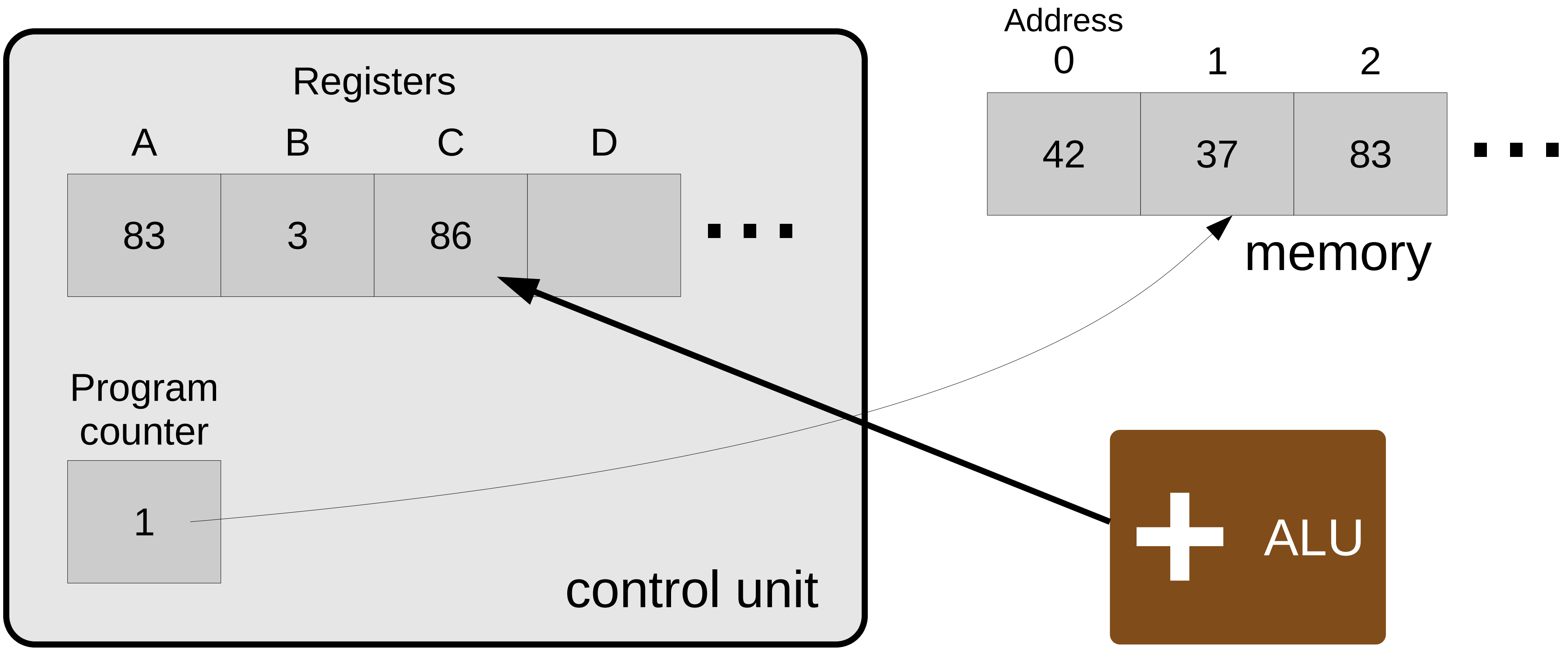
Address

0	1	2	
42	37	83	...

memory

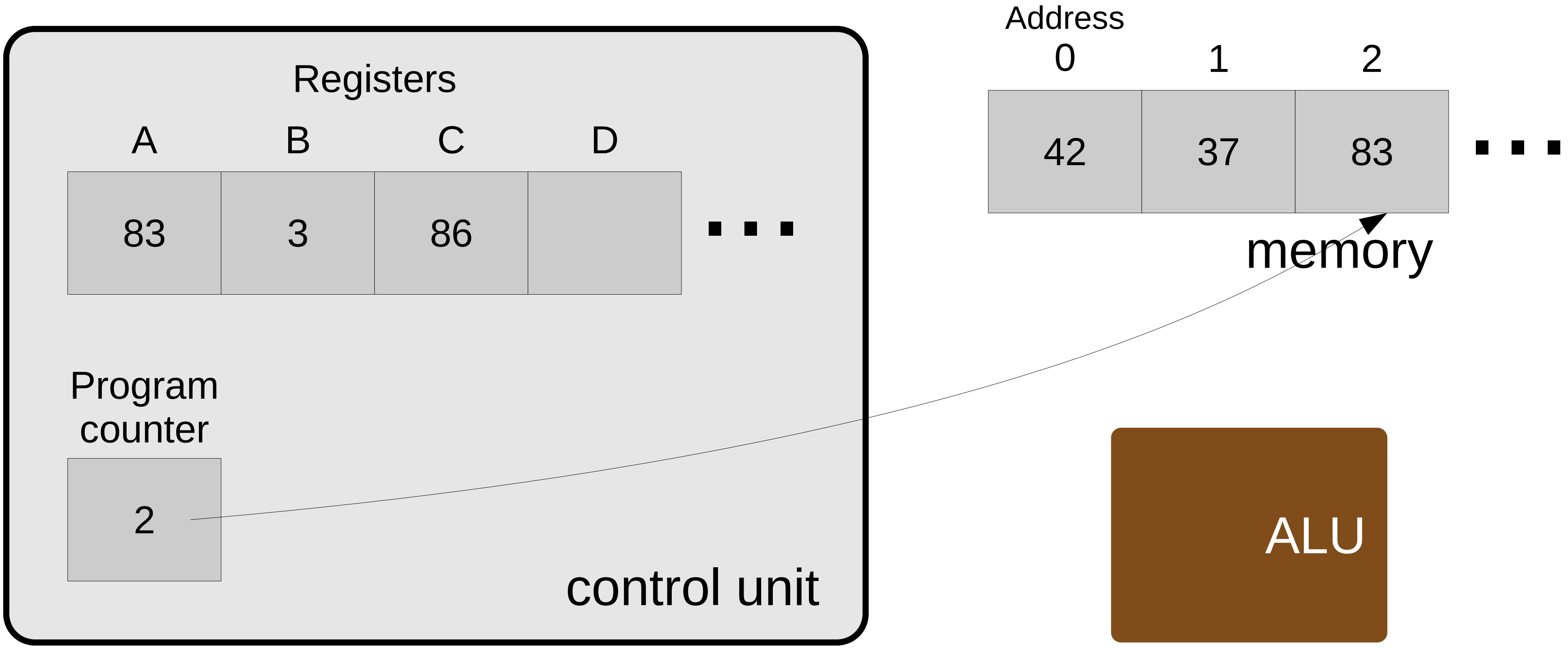
Execute instruction:
The ALU does the addition...

Running a program



Execute instruction:
… and the control unit retrieves the result

Running a program



Registers

A	B	C	D	
83	3	86		. . .

Program counter

2

control unit

Address

0	1	2	
42	37	83	. . .

memory

ALU

Increment program counter

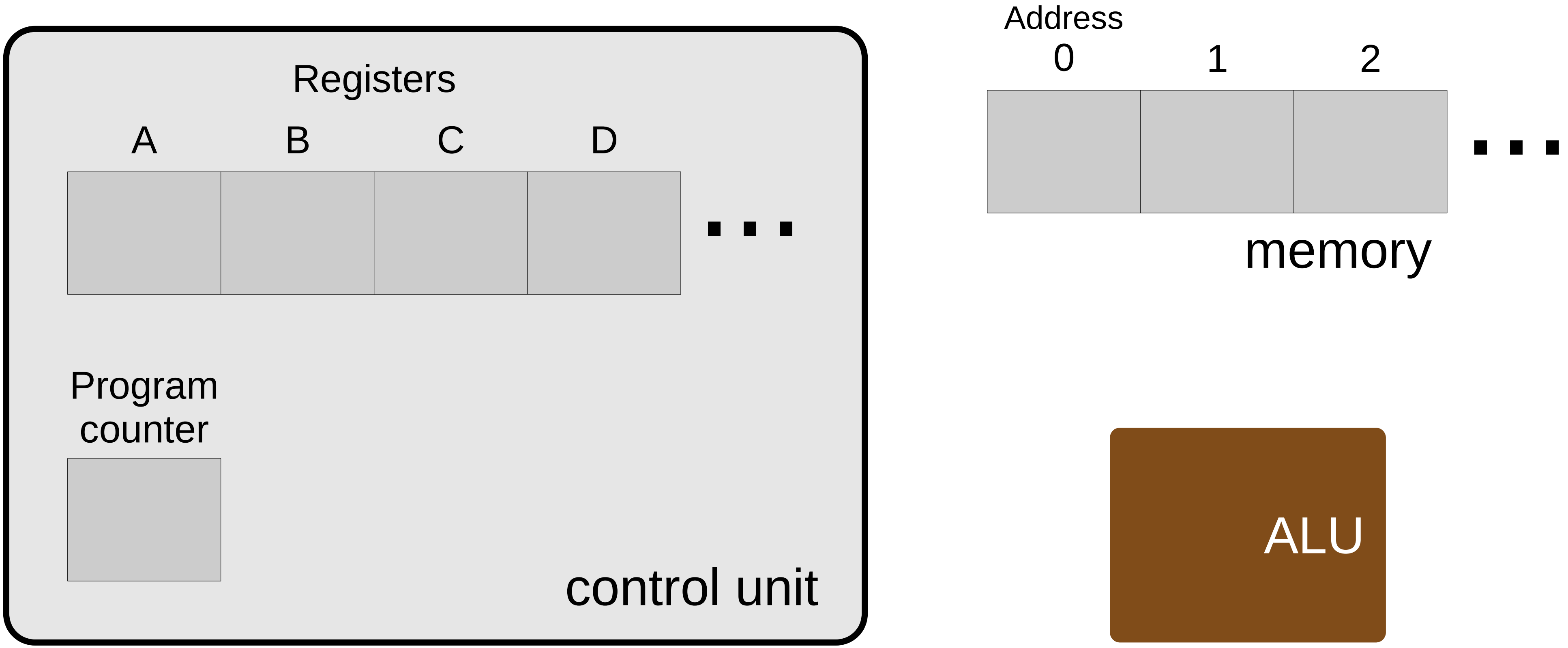
A “universal computer”

- We can do any computation by breaking it down into simple instructions – **machine code**
  - Load from memory, store to memory
  - Arithmetic
  - Tests – “is A bigger than B?”
  - Jumps – “go to instruction 4”
- We'll see lots of these later...
  - using a real processor as an example
  - and showing how your C++ code gets translated into machine instructions

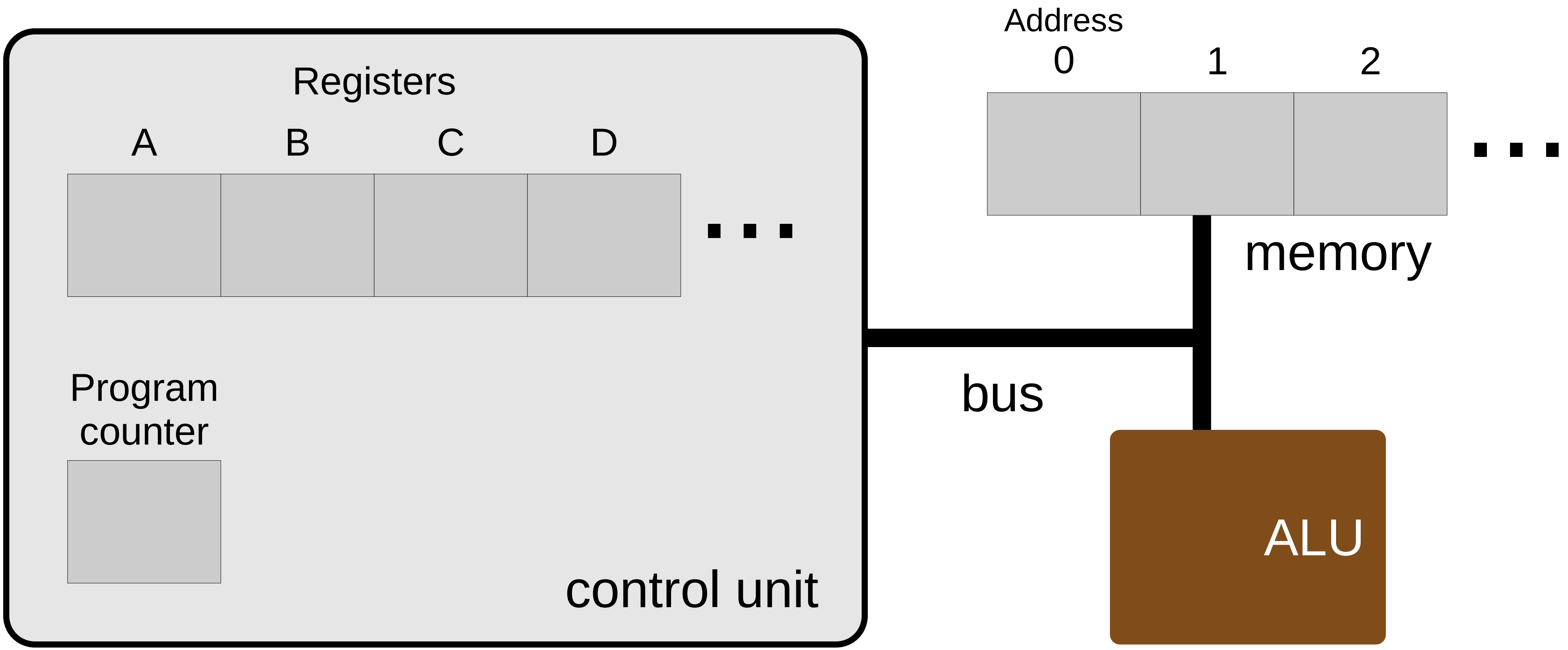
Programs as data

- You could have separate memory for instructions and data (the **Harvard architecture**)...
- ... but storing both in the same memory is usually a better idea
- We only need one kind of storage
  - Less complex wiring, less expensive to build
- We can write programs that write programs!
  - Compilers
  - Debuggers
  - Self-modifying code – e.g. “just-in-time” compilation

The basic components

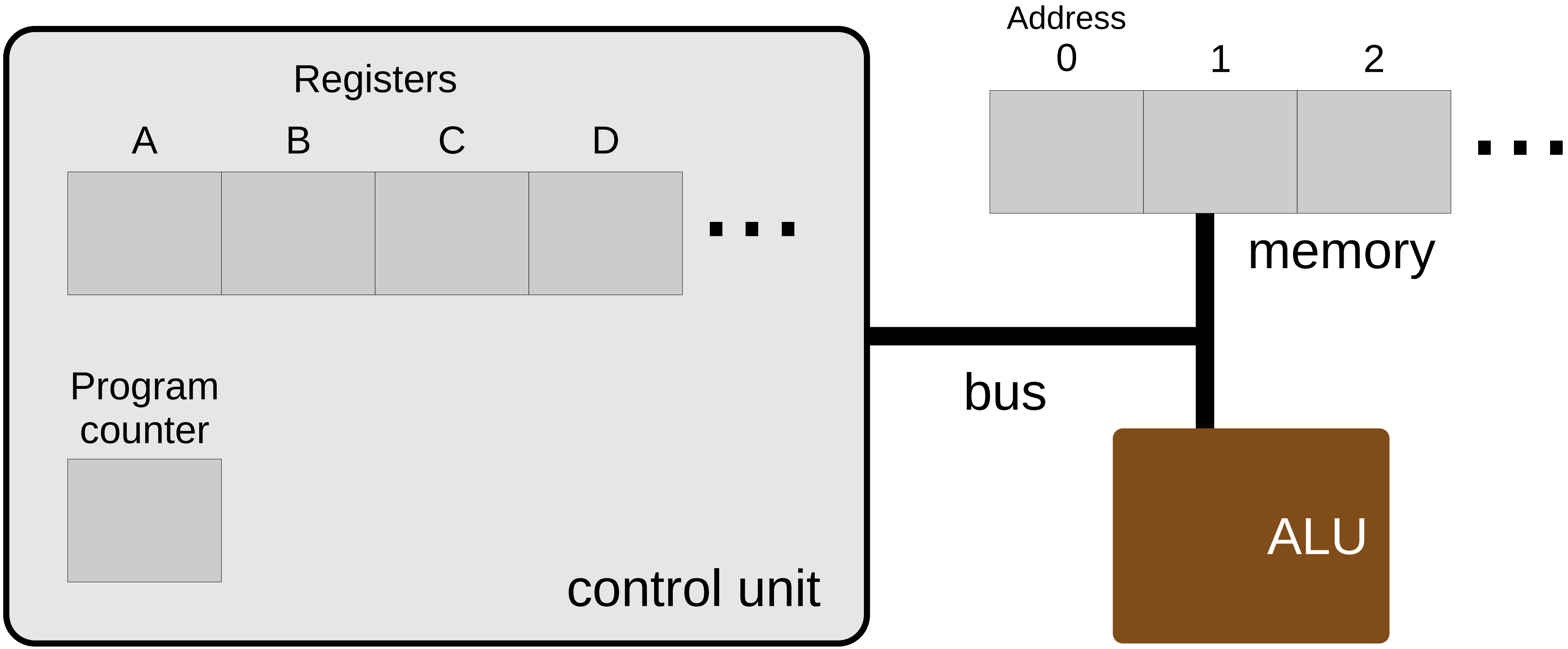


One more thing...



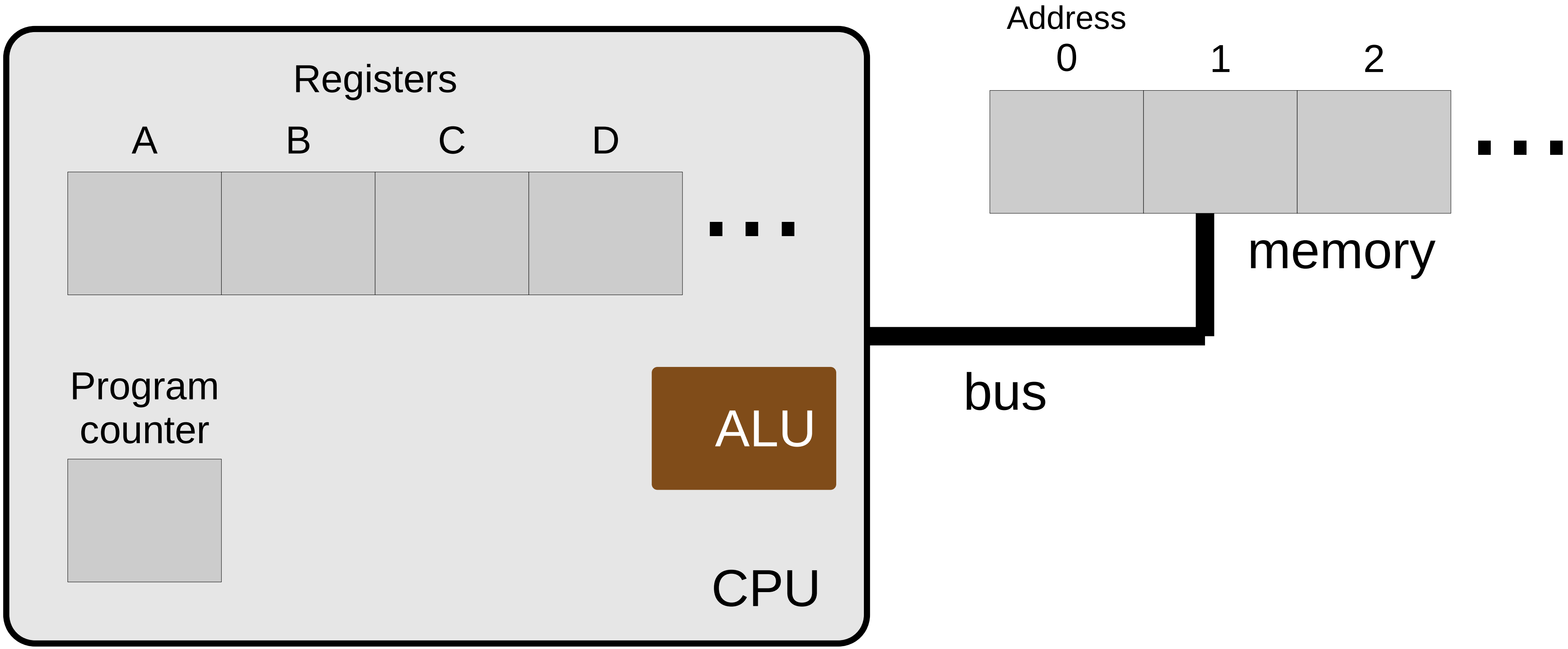
A shared **bus** connects data and control signals between units. (A bunch of wires!)

1945



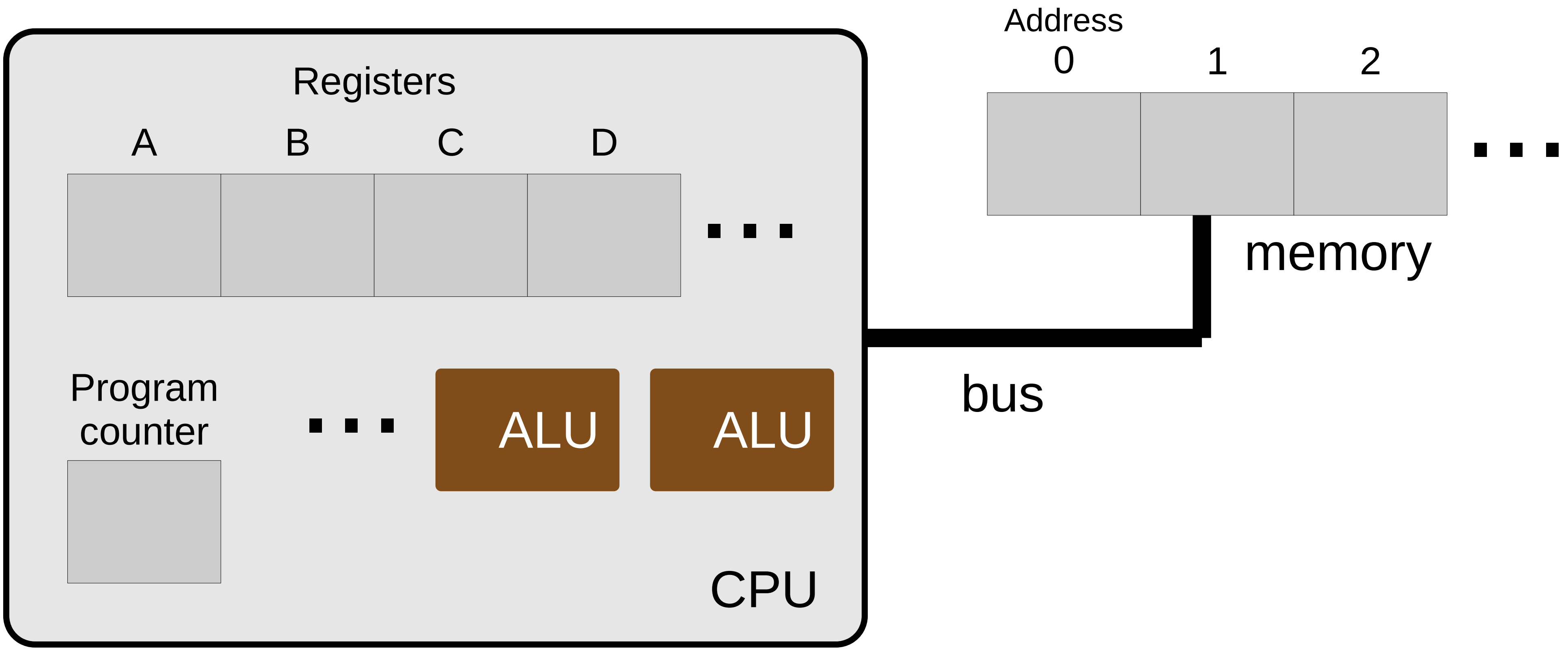
This is called the **von Neumann architecture** – because John von Neumann published an early description of it: “First Draft of a Report on the EDVAC”

1980



In more recent processors, the ALU is part of the CPU – it's all on one chip

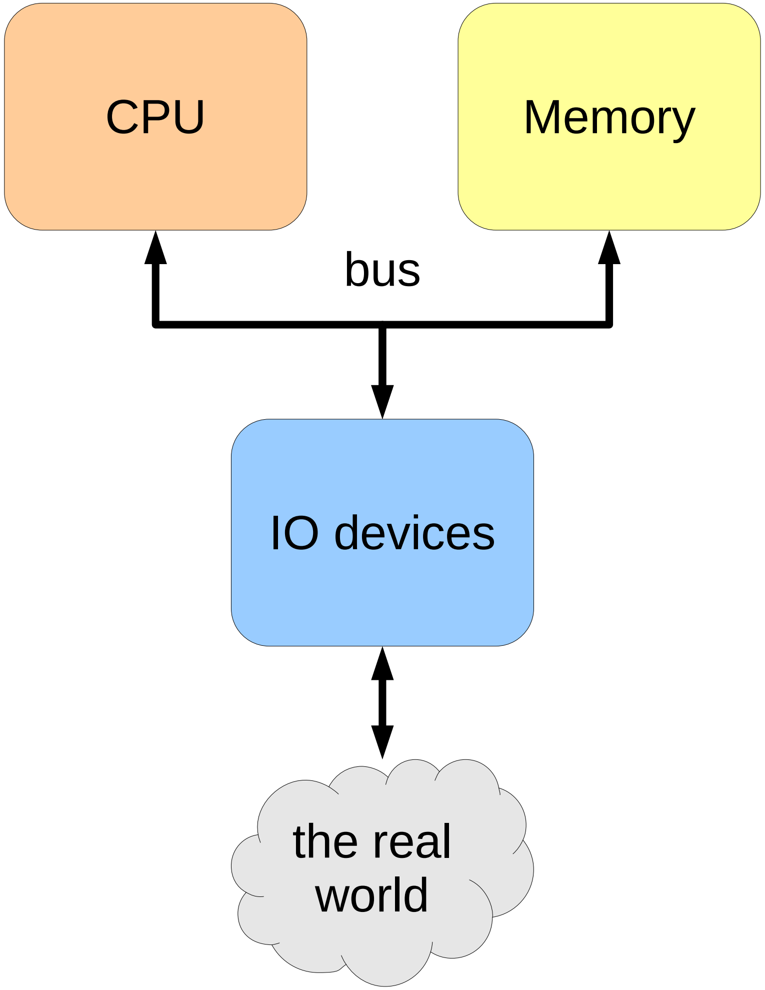
2000



… and these days you usually have multiple ALUs – e.g. one for integers, one for floats – more on that next year

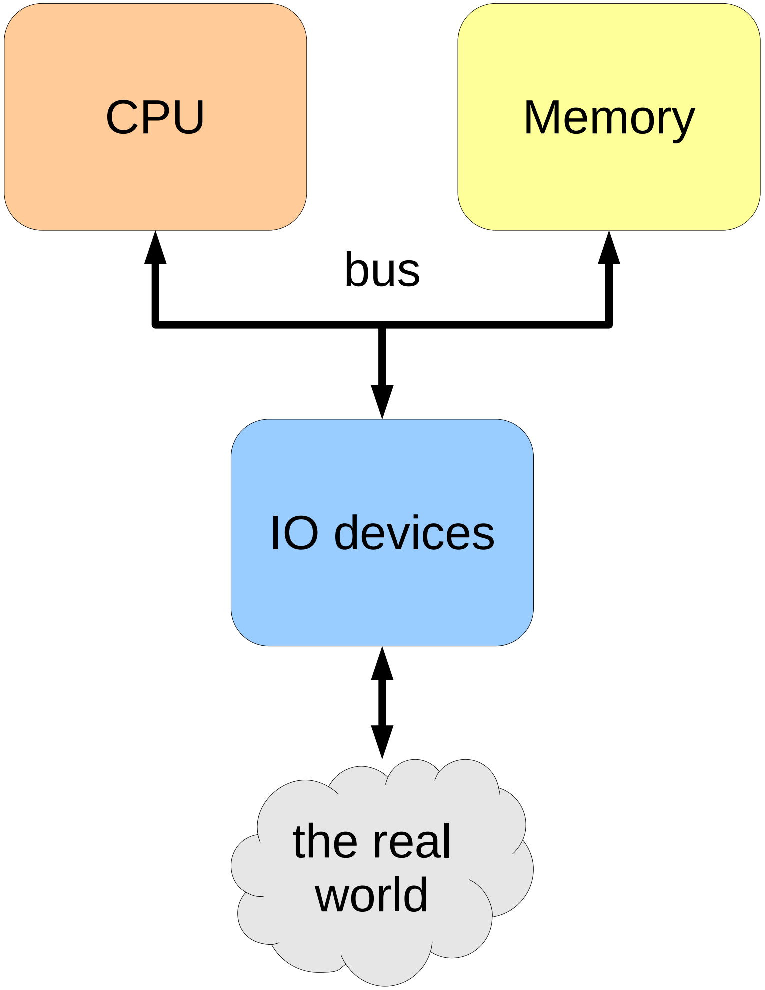
A simple computer

- The computer's only useful if it can interact with the real world: **input-output devices**
- Graphics displays, keyboards, hard disks, network ports...
- CPU controls IO devices
  - In modern machines, many IO devices include their own processors (e.g. GPUs, hard disks)



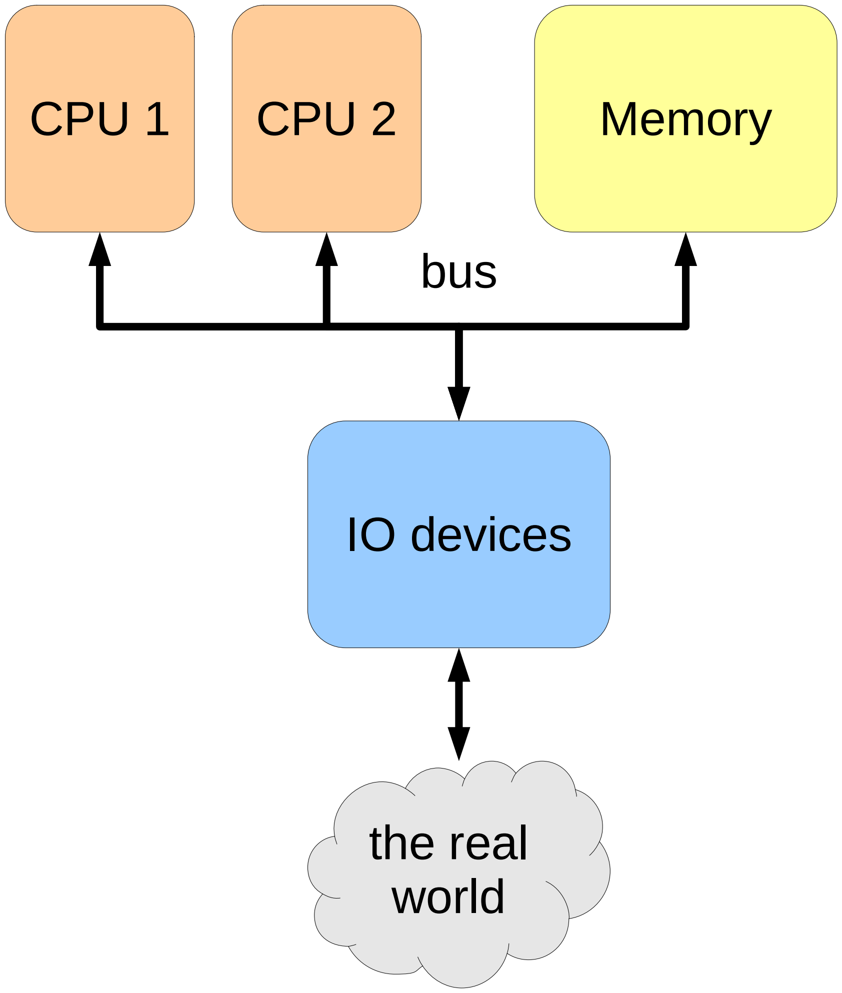
A less simple computer

- In practice, modern machines are more complicated...



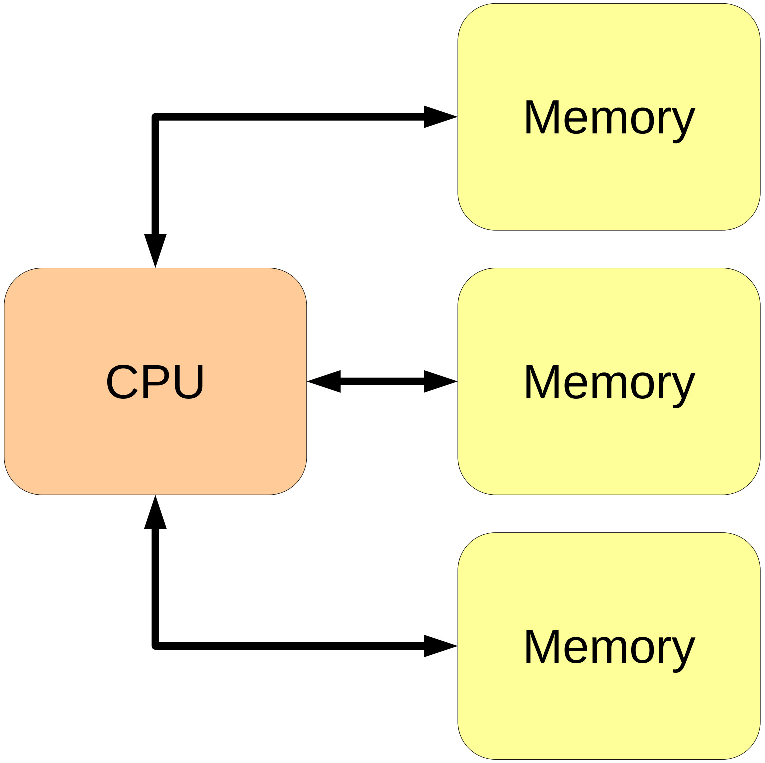
A less simple computer

- Multiple CPUs are very common
  - … often multiple **cores** within the same **CPU chip**
  - *Much* more on this later!

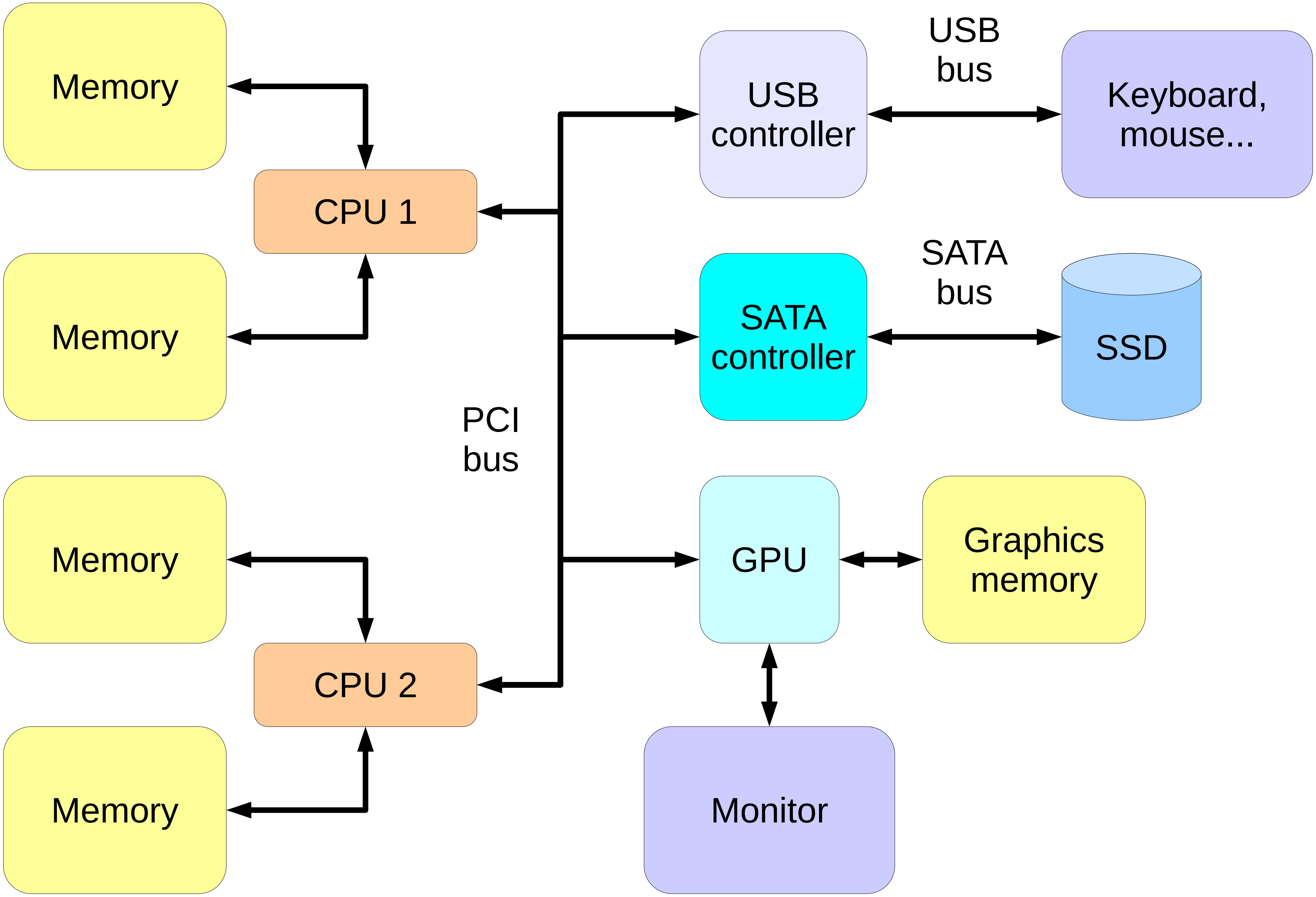


A less simple computer

- **Memory** is typically much slower than the **CPU**
  - 10x or worse
  - Your CPU spends most of its time waiting for memory...
- Have multiple separate memories, and a separate bus for each one



A typical PC


Memory
CPU 1
Memory
Memory
CPU 2
Memory
PCI bus
USB controller
USB bus
Keyboard, mouse...
SATA controller
SATA bus
SSD
GPU
Graphics memory
Monitor

Summary

- Basic components of a computer system
- More depth on much of this later...

- The “stored program” idea, and instructions
- More about instructions later too!