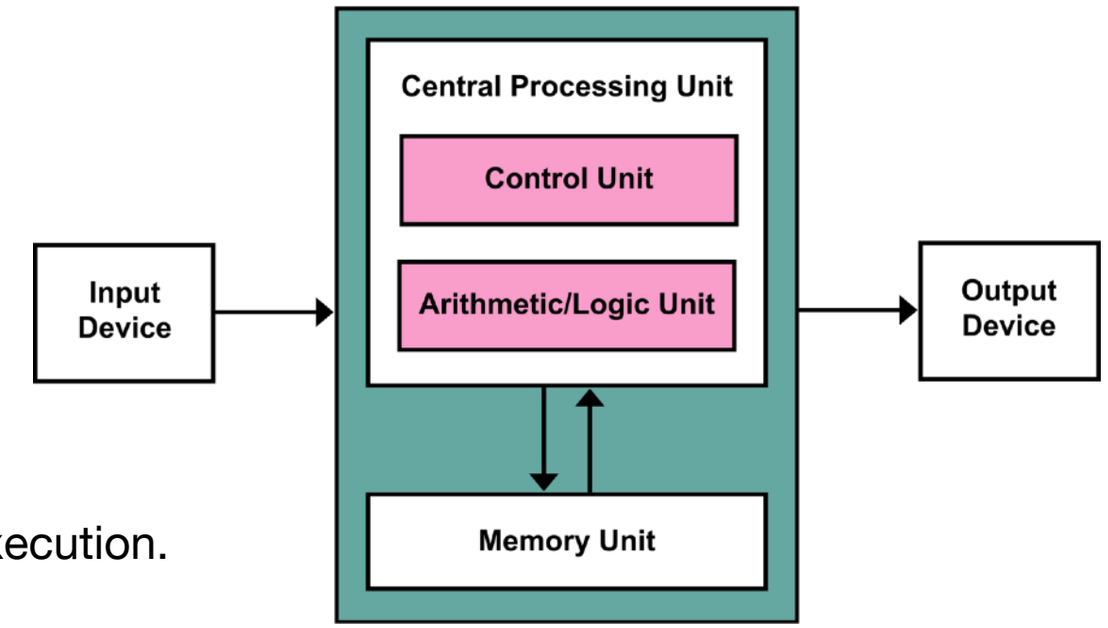
# Princípios da Computação

Conventional computer system

# The Von Neumann architecture

- Proposed in 1945 by John Von Neumann.
- Model for designing and building computers following three characteristics:

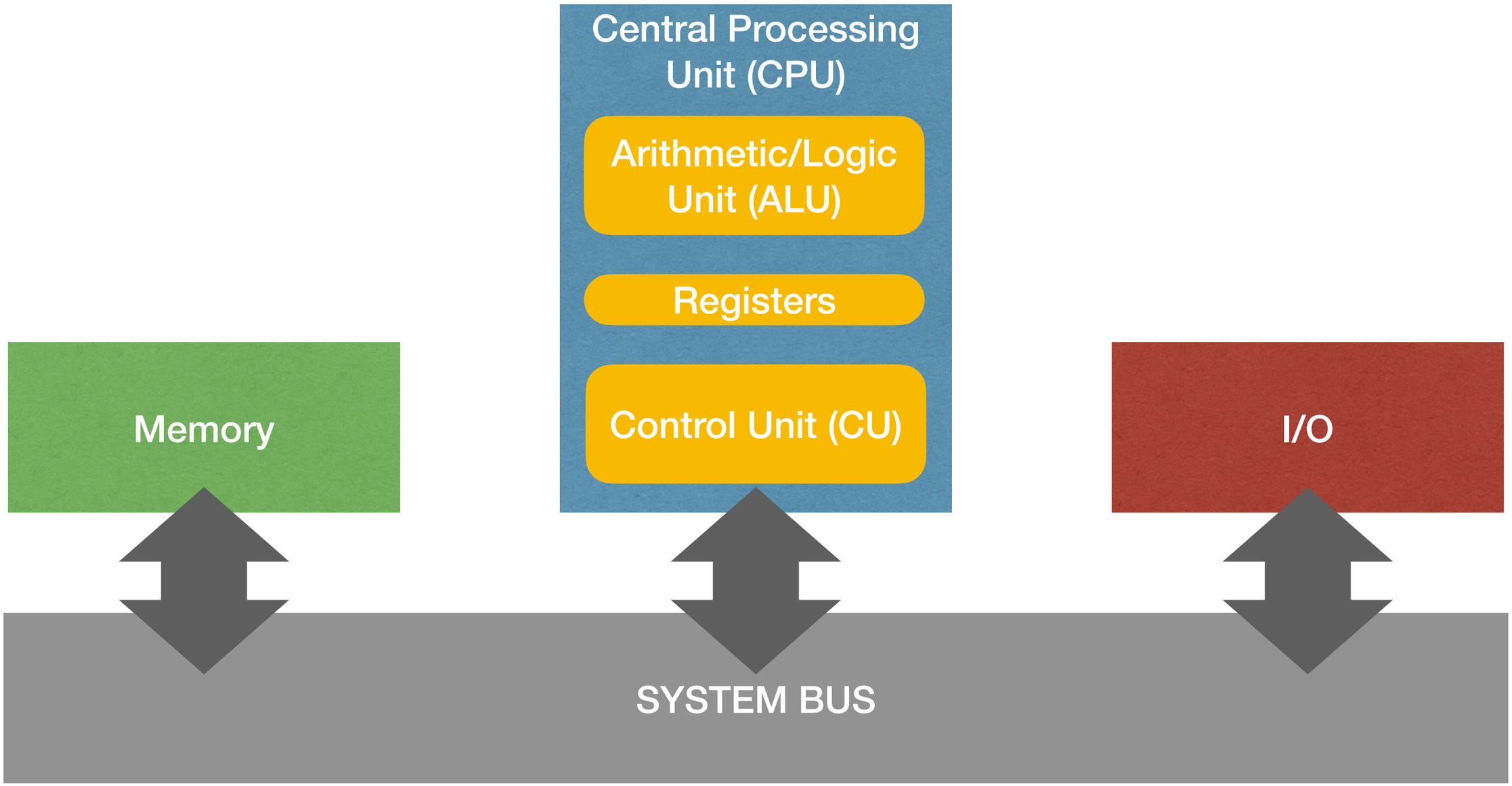
    1. The computer of four main subsystems:
        - Memory
        - Arithmetic/Logic Unit (ALU)
        - Control Unit (CU)
        - Input/Output system (I/O)
    2. The program is stored in memory during execution.
    3. Program instructions are executed sequentially.



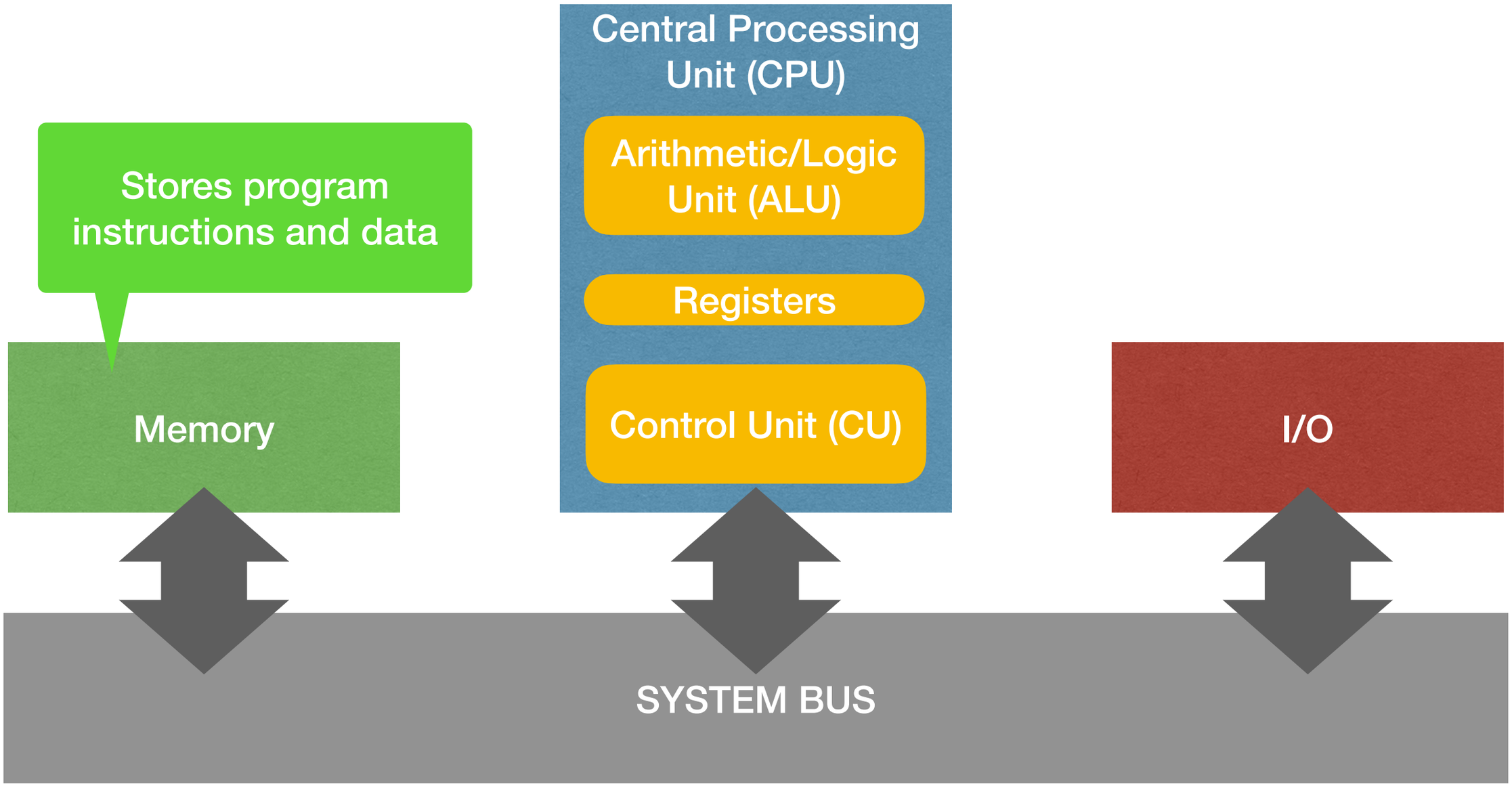
# The Von Neumann architecture

- Memory can store binary numbers.
- Instructions must be coded into numbers.
- Processor provides an **instruction set**:
    - map of all instructions supported by the processor and their respective unique ids.
- The processor has (permanent) circuitry to execute each instruction: the **microarchitecture**
    - for each instruction, the control unit determines which circuits should operate.

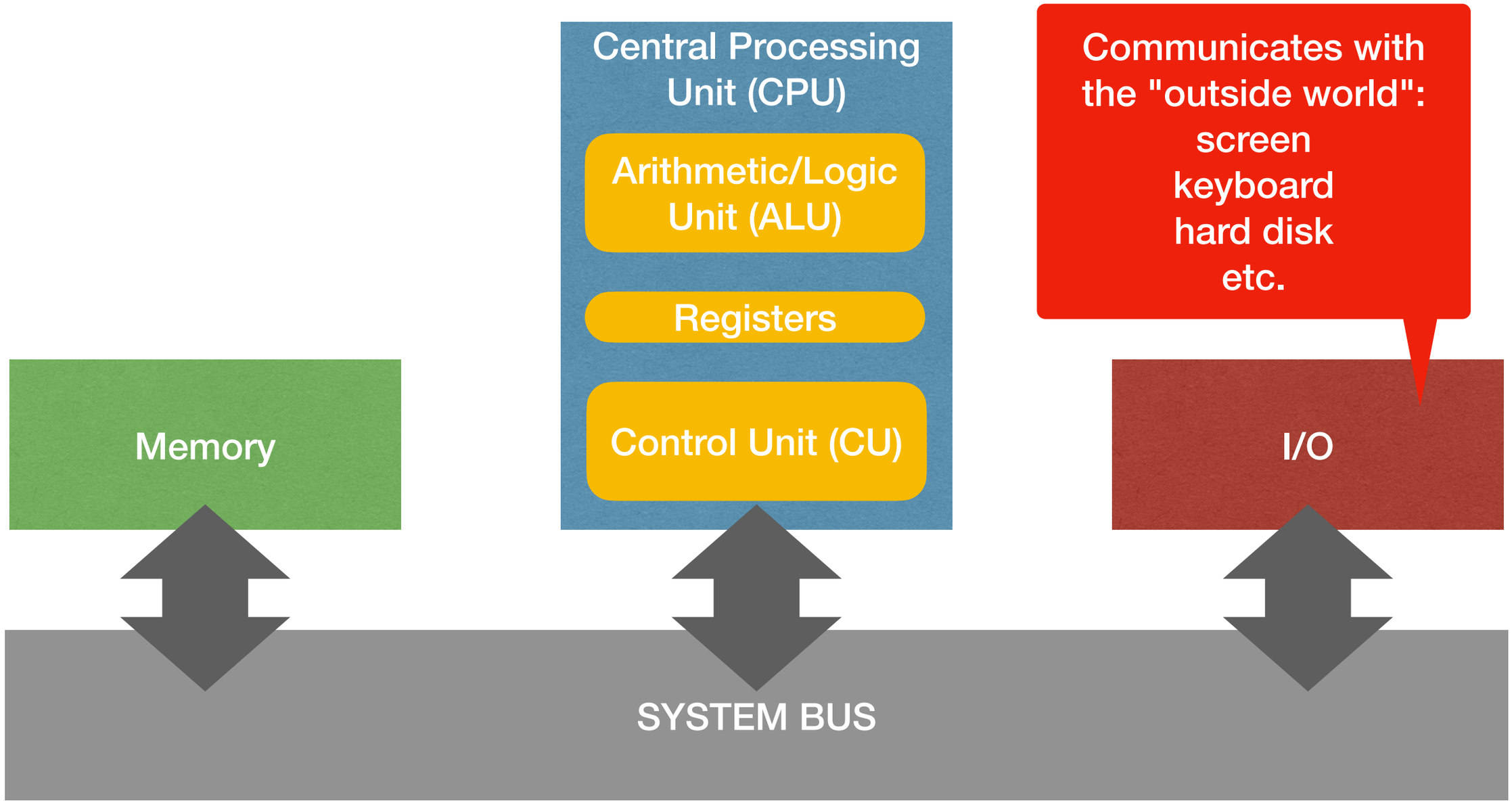
# The Von Neumann architecture



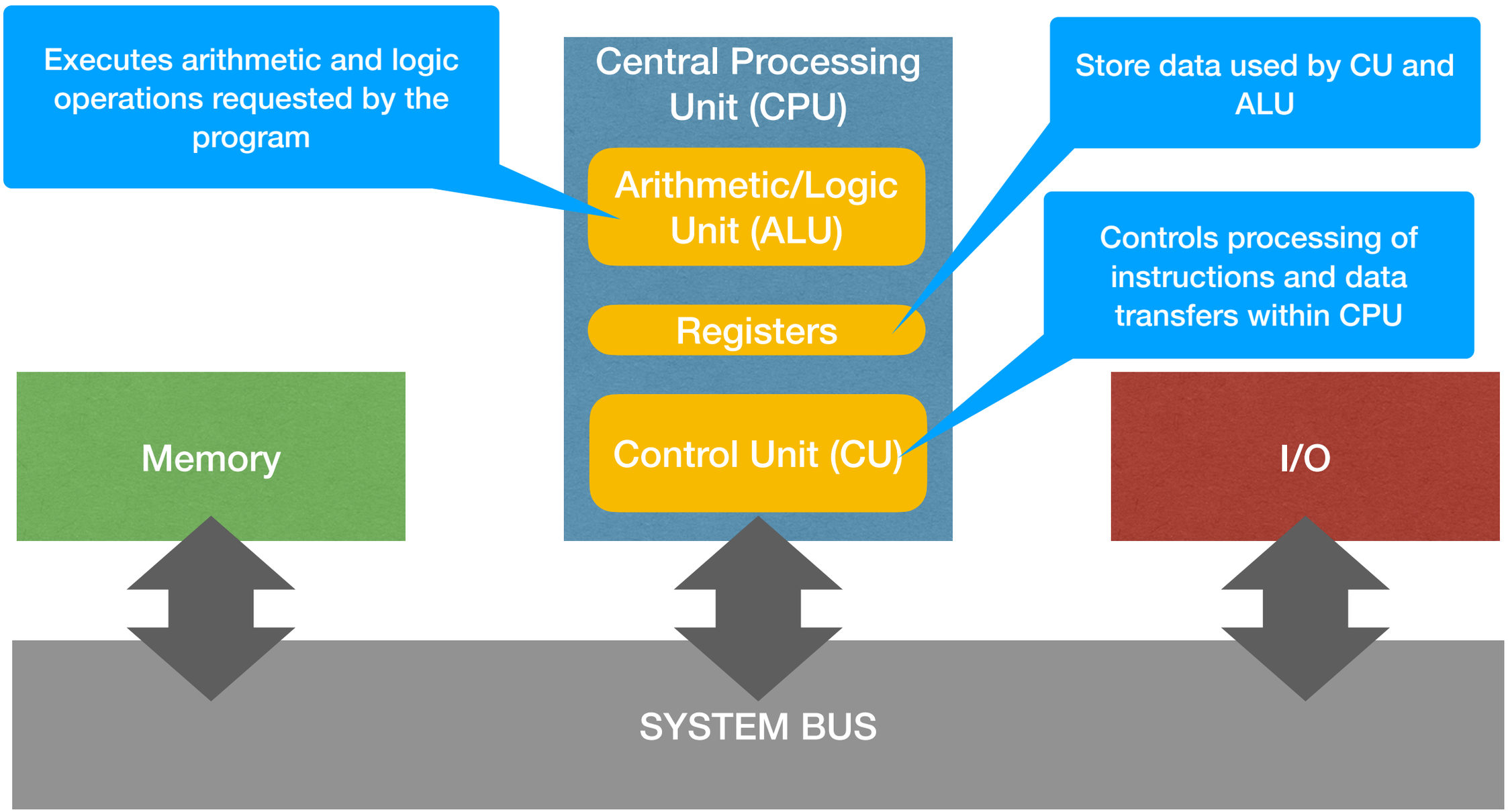
# The Von Neumann architecture



# The Von Neumann architecture



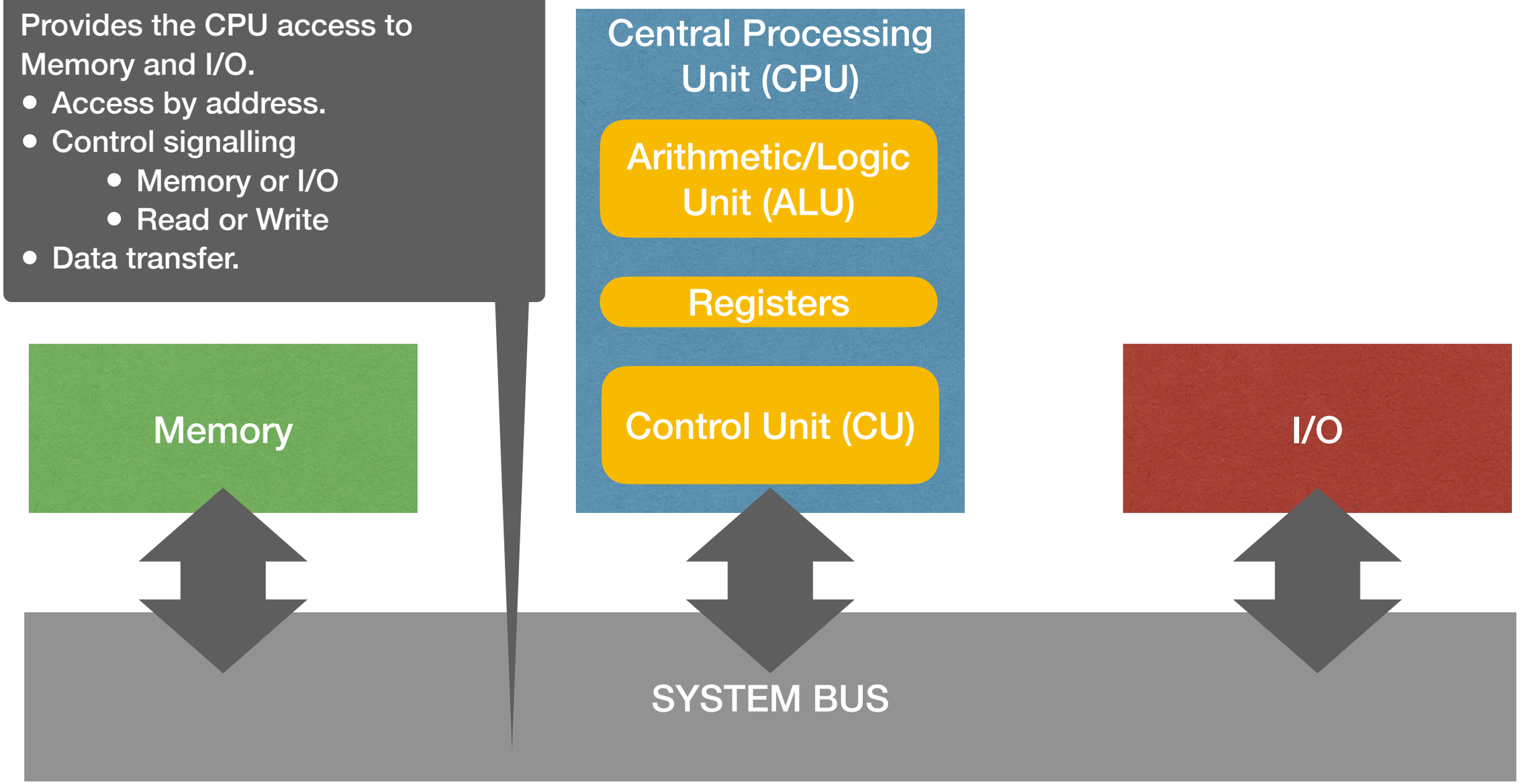
# The Von Neumann architecture



# The Von Neumann architecture

Provides the CPU access to Memory and I/O.

- Access by address.
- Control signalling
  - Memory or I/O
  - Read or Write
- Data transfer.



# Memory

- It is the **primary storage** system of a computer.
    - The processor can only execute program **instructions** and used program **data** that are **loaded** into memory!
- Constituted mainly of RAM (Random Access Memory)…
    - You can **read** from and **write** to this memory.
    - It is **volatile**: it only stores data as long as the computer is powered on!
- … but also of ROM (Read Only Memory).
    - Stores the instructions for **bootstrapping** (i.e. to start) the computer.

# Memory

- Memory is organised as a group of memory cells of a fixed size (currently 8 bits).
    - Each memory cell is accessed (fetch/store) by its unique **address.**
    - A memory cell is the minimum unit of access… but more addresses can be accessed at once.
        - E.g. a 64-bit processor may access 8 addresses (64 bits) at once.
- The access time is (generally) the same for all cells…
    - Memory access operations are quite slow, as seen from the processor!

# Input/Output (I/O)

- Handles devices that allow the computer to…
    - Communicate with the outside world (**peripherals** and **communication**)
        - Screen, keyboard, printer, network adapter, etc.
    - Mass-store information (**secondary storage**)
        - Direct Access Storage Devices (DASD): hard disks, flash drives, optic discs, etc.
        - Sequential Access Storage Devices (SASD): magnetic tapes.

# Input/Output (I/O)

- Typically, I/O devices are quite slower than memory!
- Each I/O device has an embedded **controller** that autonomously executes the requested operations.
    1. The device controller executes the operation requested by the processor.
    2. The device signals the processor (external interrupt) when operation is done.
- Direct Memory Access (DMA).
    1. The processor initiates the transfer.
    2. Data is transferred between memory and I/O controller without the processor coordination.
    3. The processor is signalled (external interrupt) when transfer is done.
- I/O controller and DMA release the processor to do other operations!

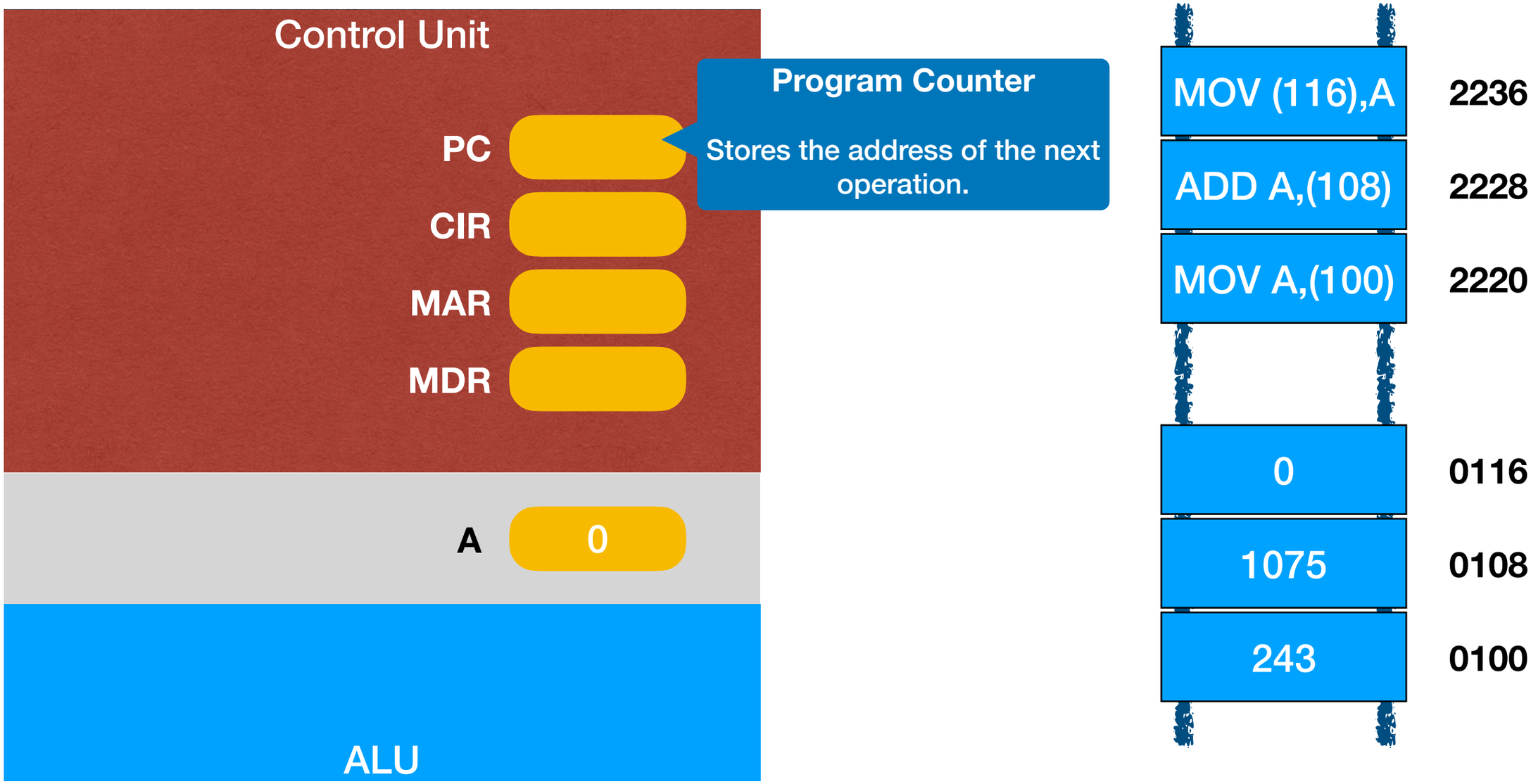
# Arithmetic/Logic Unit (ALU)

- Performs arithmetic and logic operations.
- Consists of:
  - circuits to perform the operations
  - registers to store operands and results

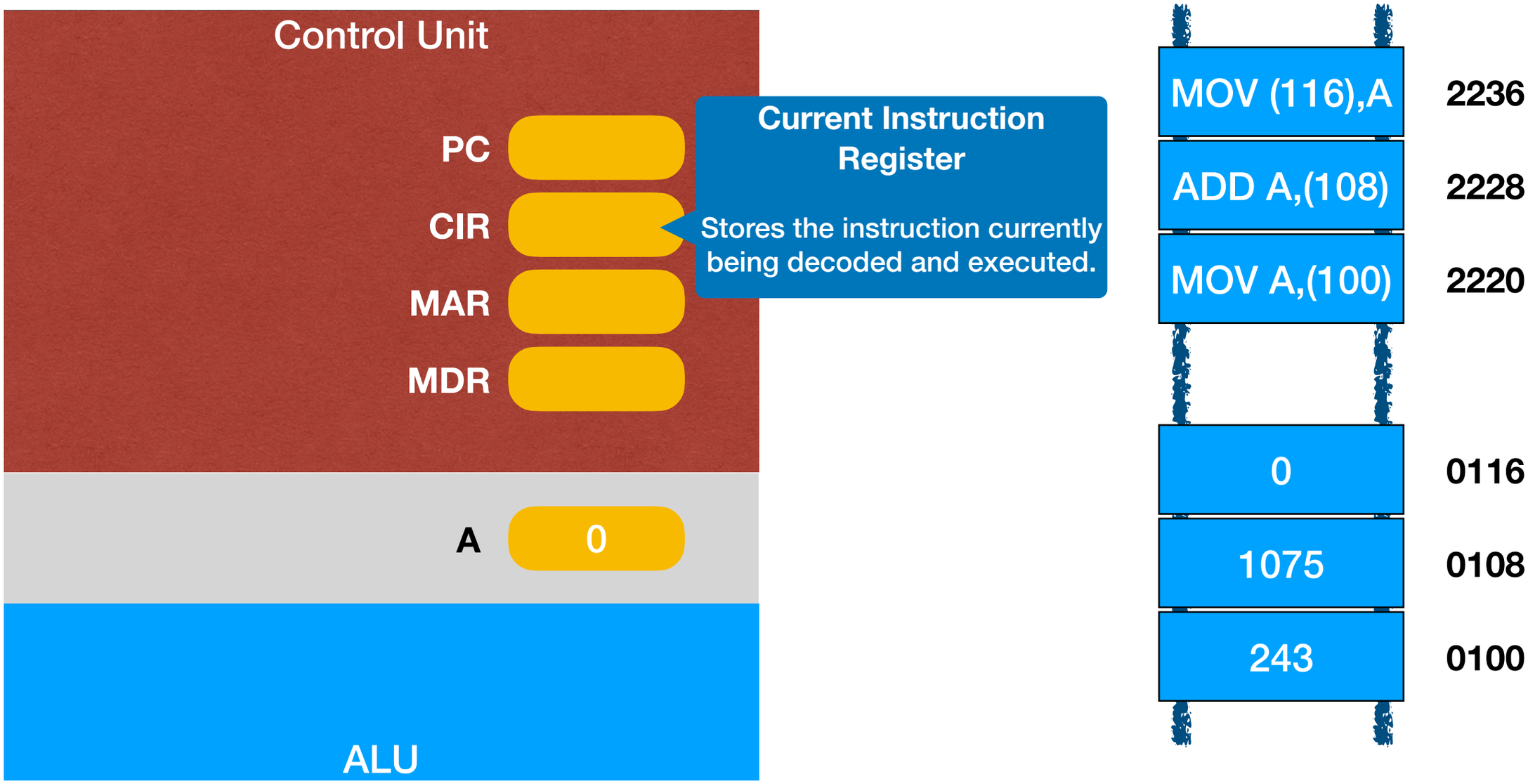
# Control Unit (CU)

- System responsible to execute the program.
- Applies the **Fetch - Decode - Execute** cycle:
    1. **Fetch** from memory the next instruction to execute.
    2. **Decode** the instruction, i.e. to determine what to do.
    3. **Execute** the instruction by signalling the appropriate circuits (ALU, memory, I/O).
- This cycle is repeated until the CU fetches the HALT instruction.
    - The HALT instruction stops the CU until it receives an external interrupt.

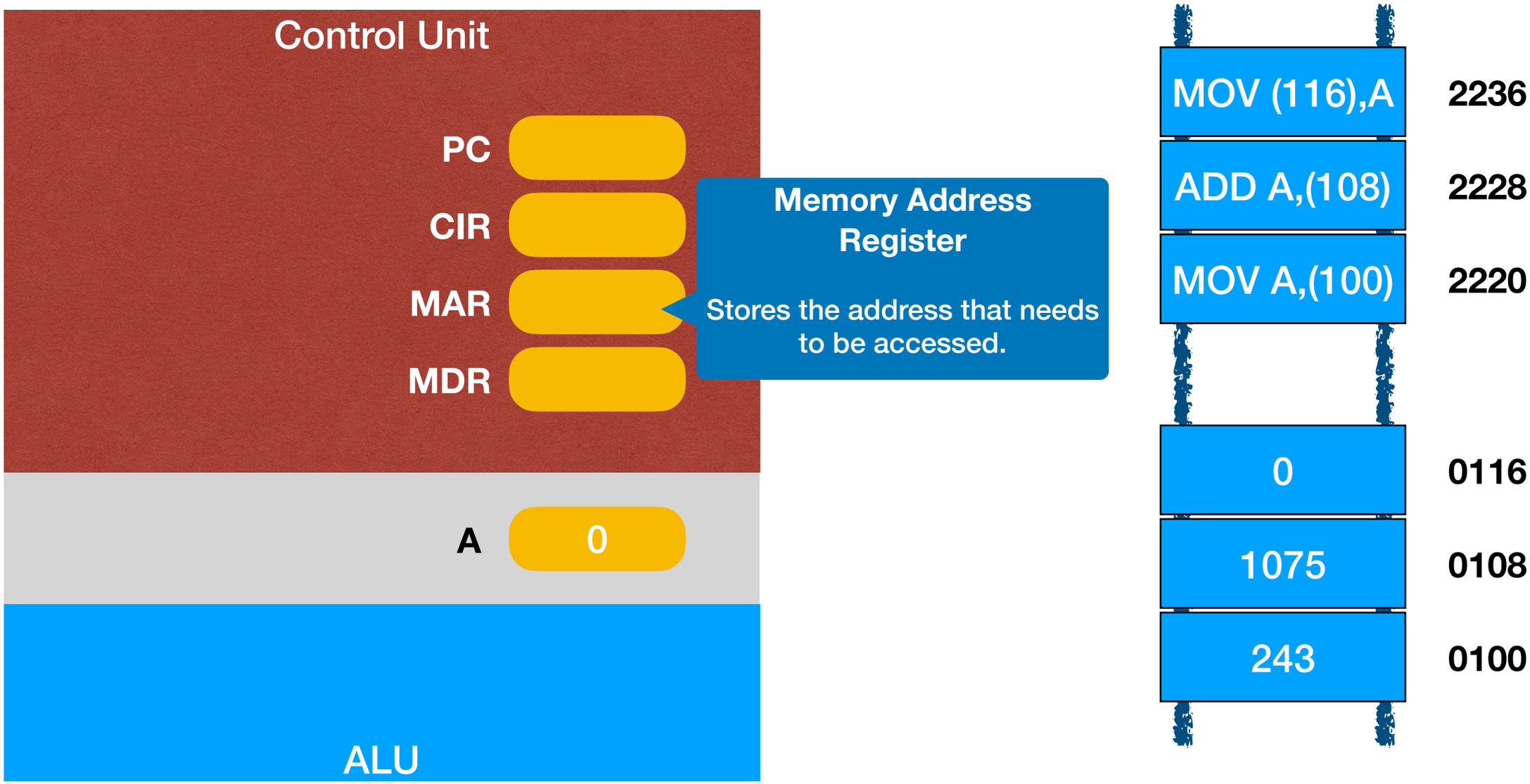
# Fetch-Decode-Execute cycle



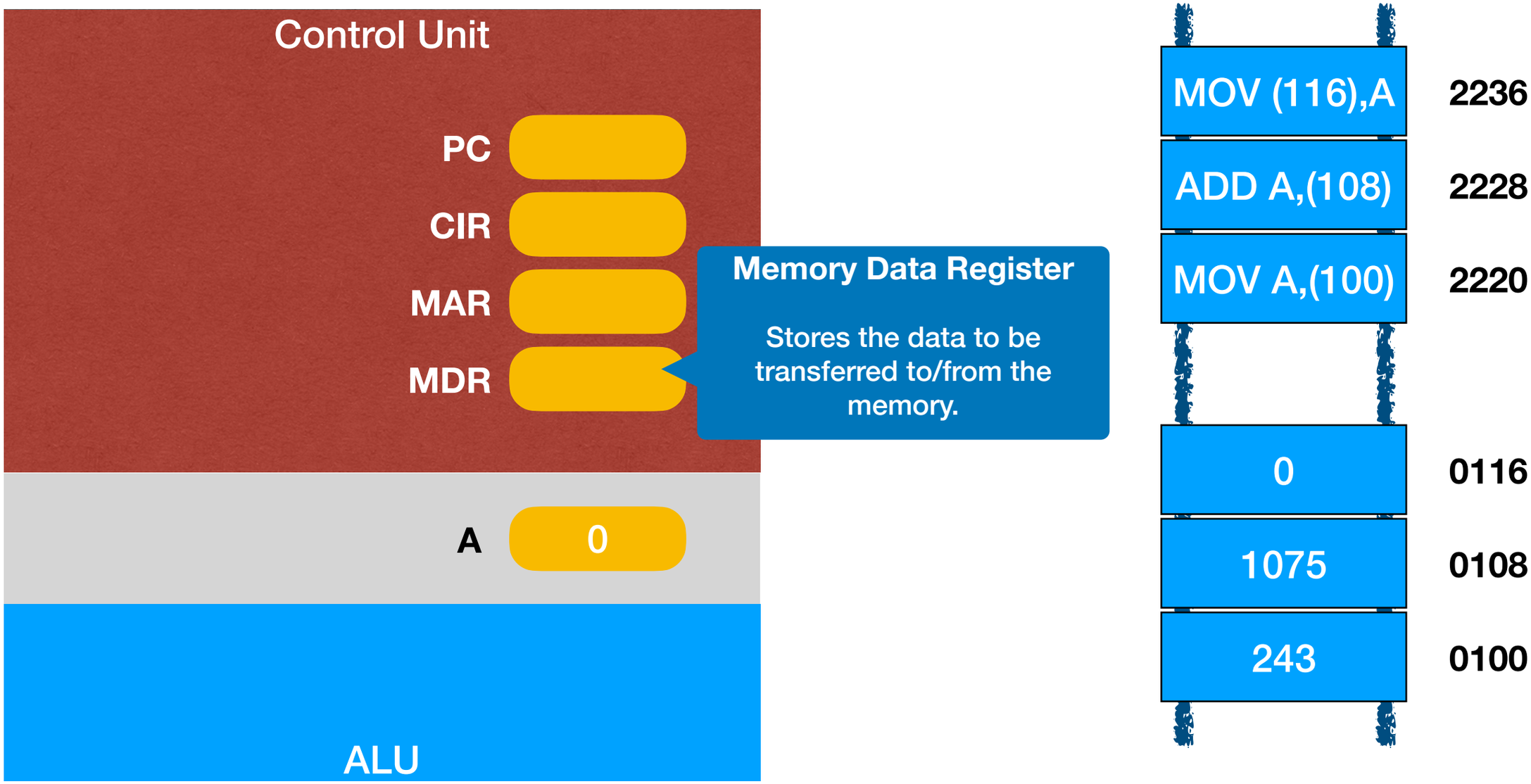
# Fetch-Decode-Execute cycle



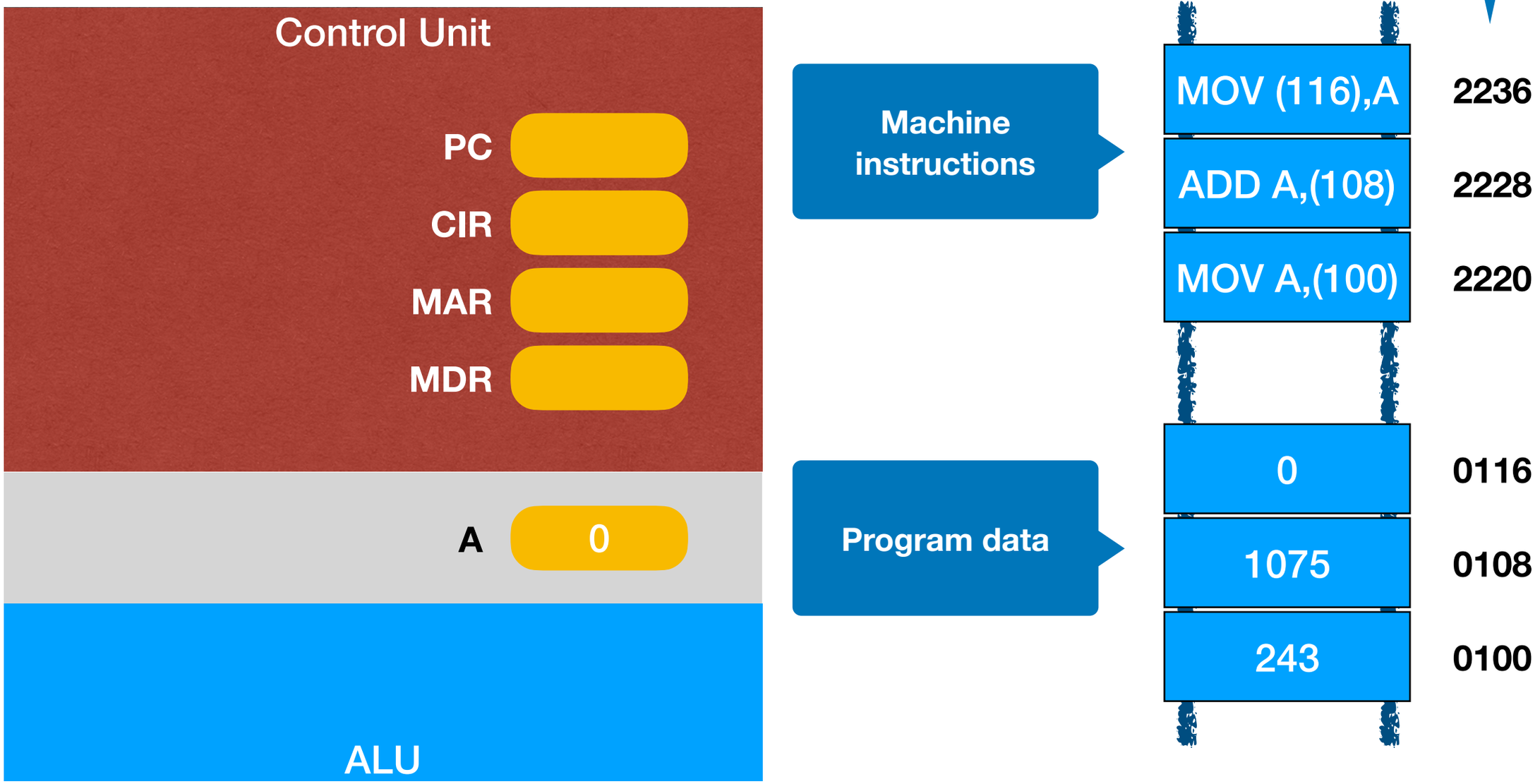
# Fetch-Decode-Execute cycle



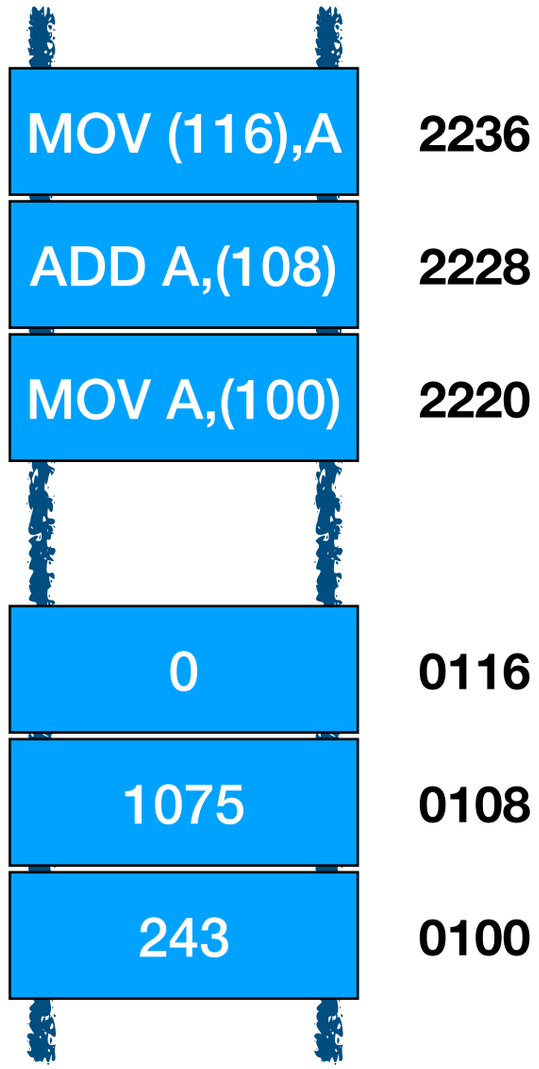
# Fetch-Decode-Execute cycle



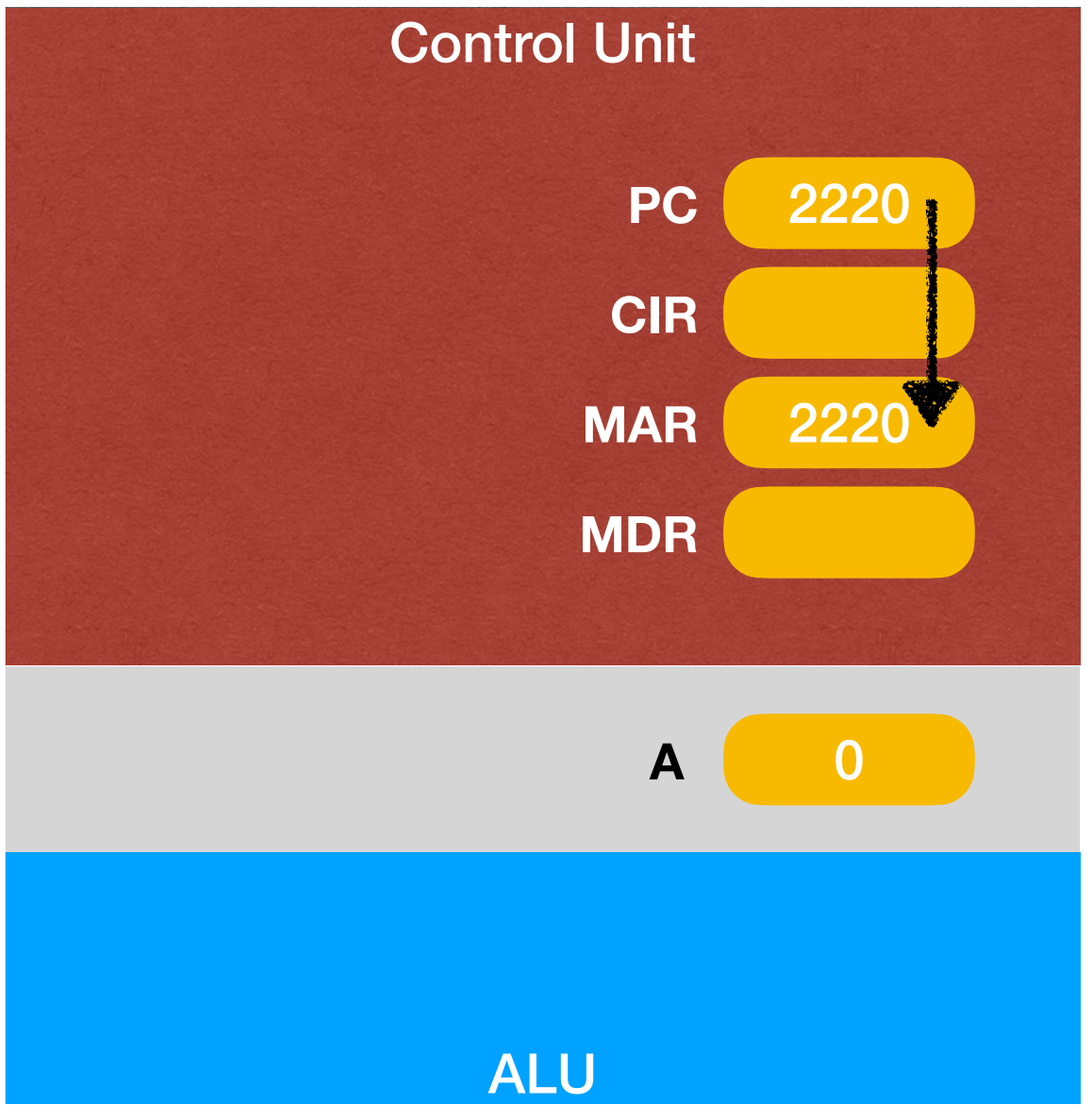
# Fetch-Decode-Execute cycle

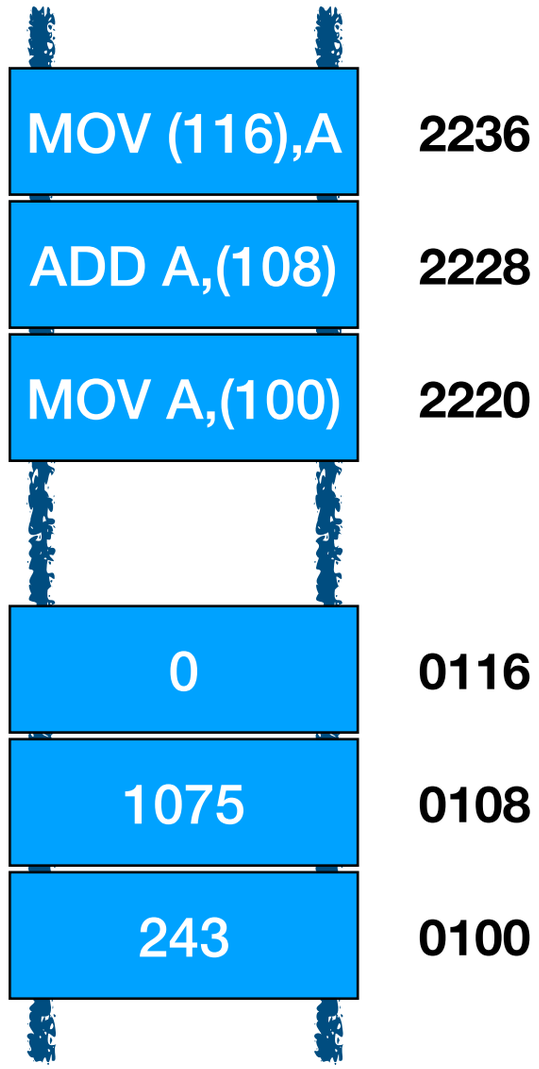


# Fetch-Decode-Execute cycle

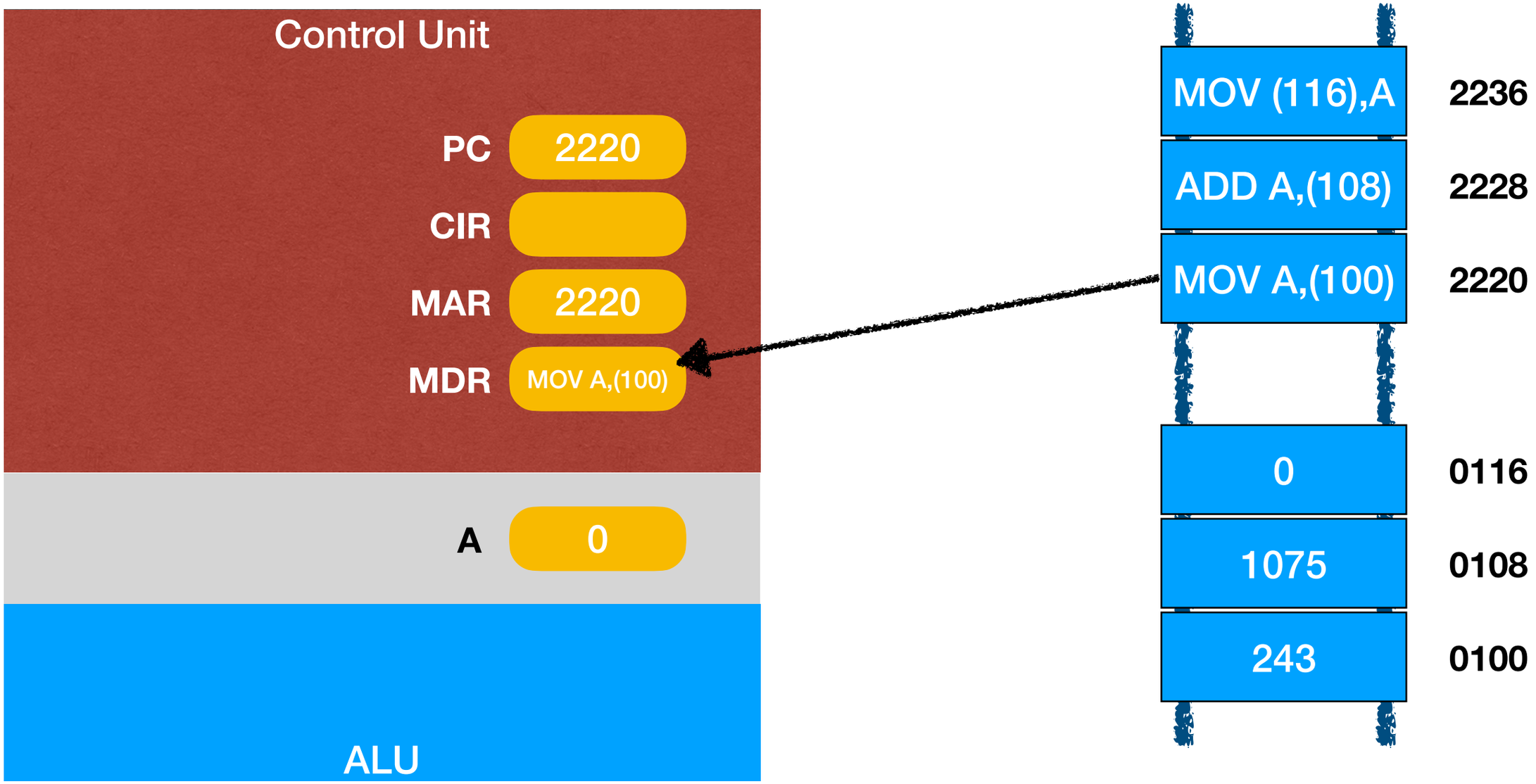



# Fetch-Decode-Execute cycle

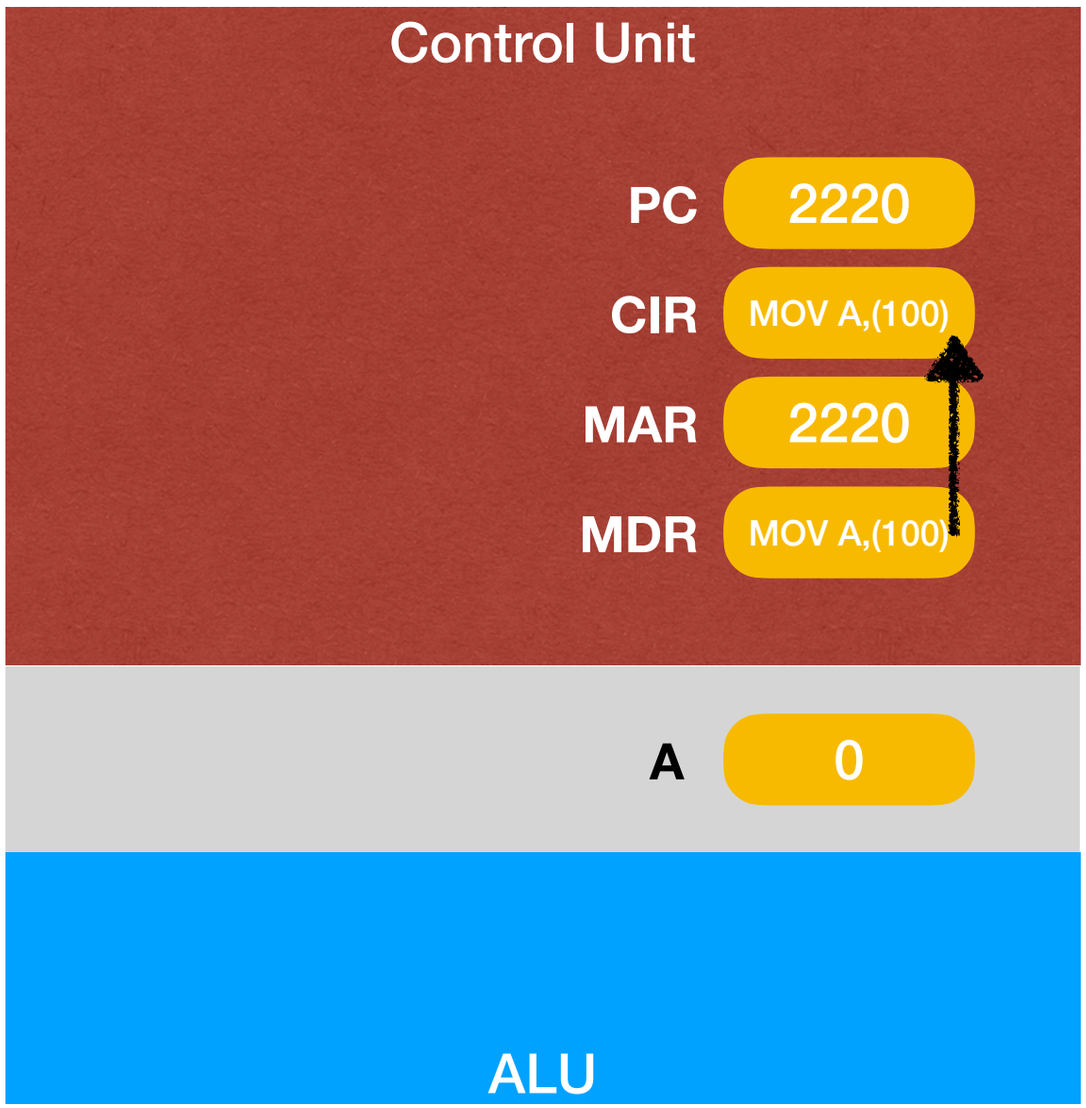


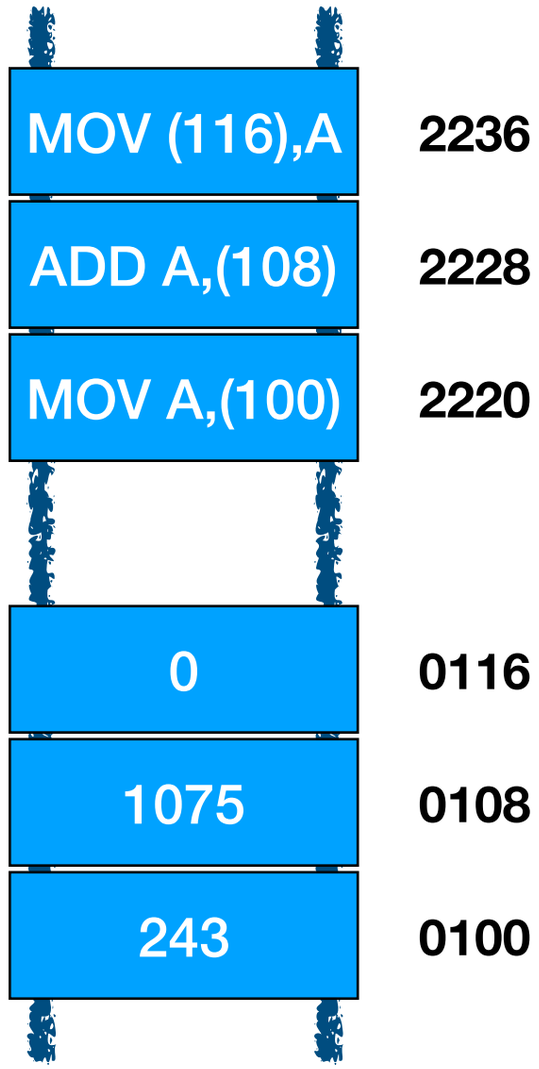



# Fetch-Decode-Execute cycle

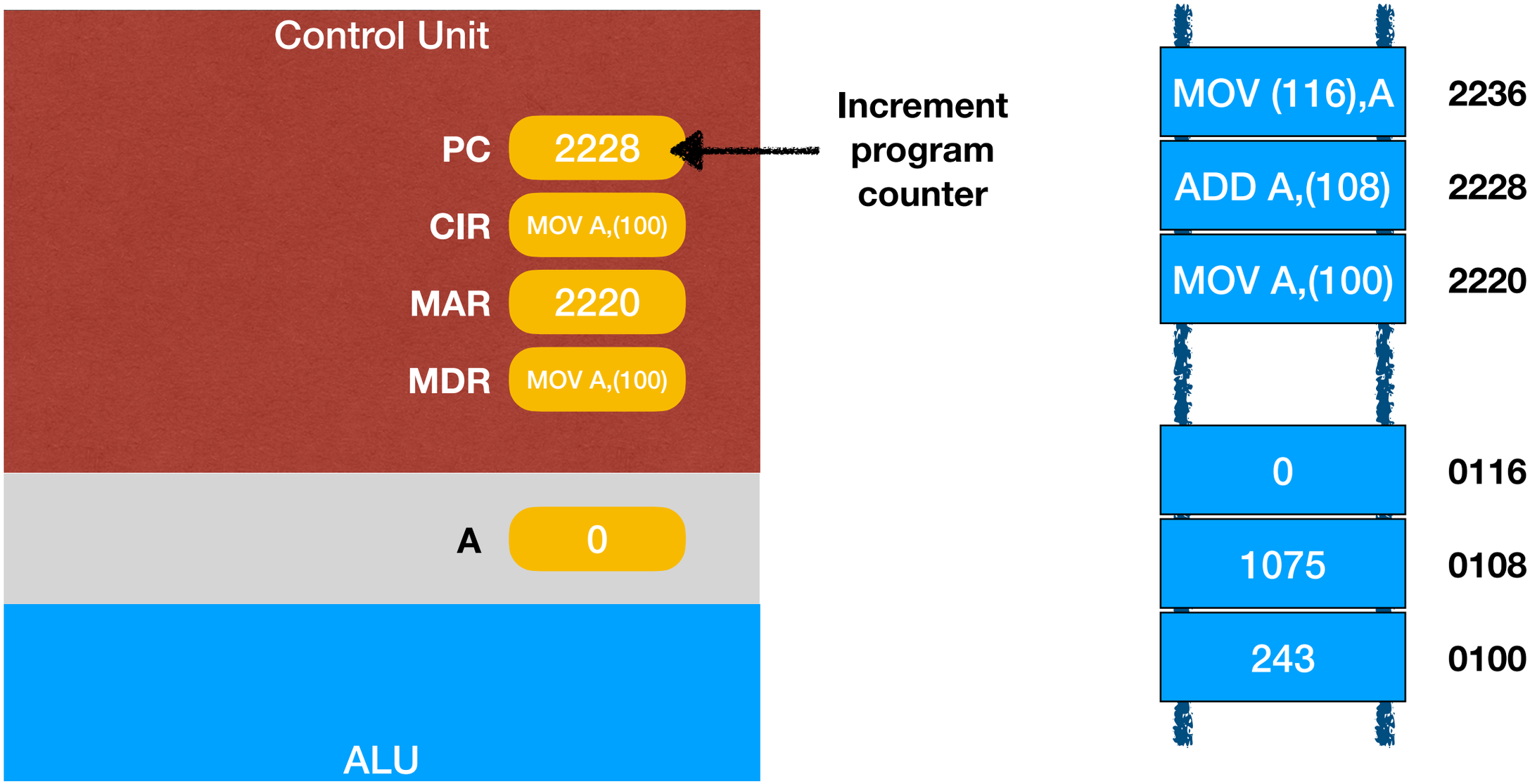


# Fetch-Decode-Execute cycle

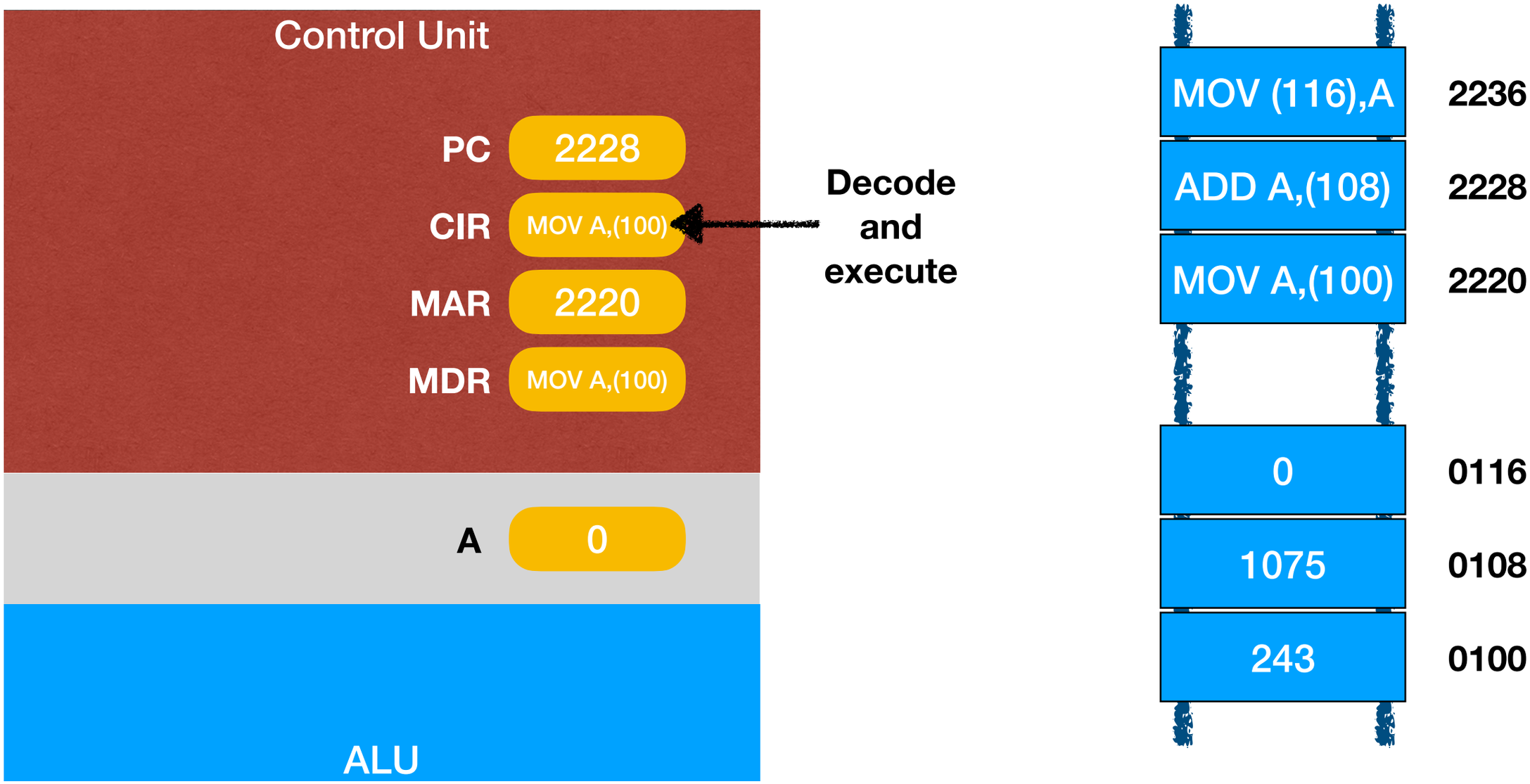


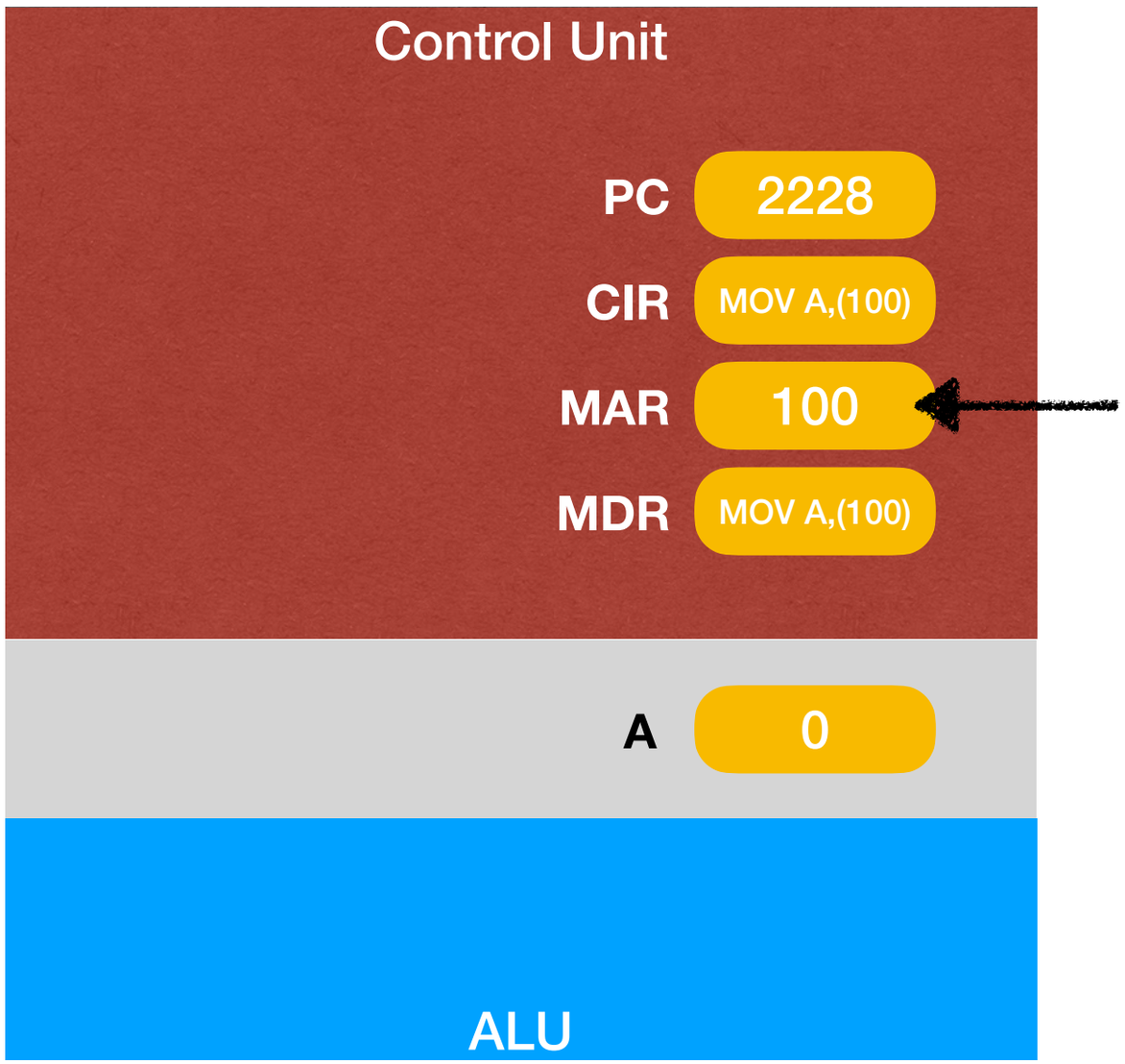
# Fetch-Decode-Execute cycle

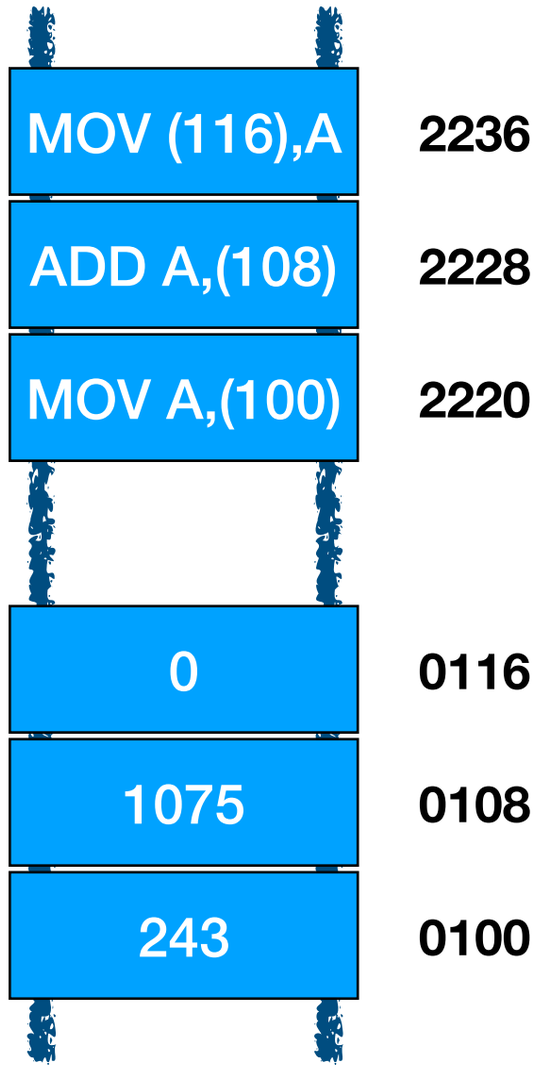


# Fetch-Decode-Execute cycle

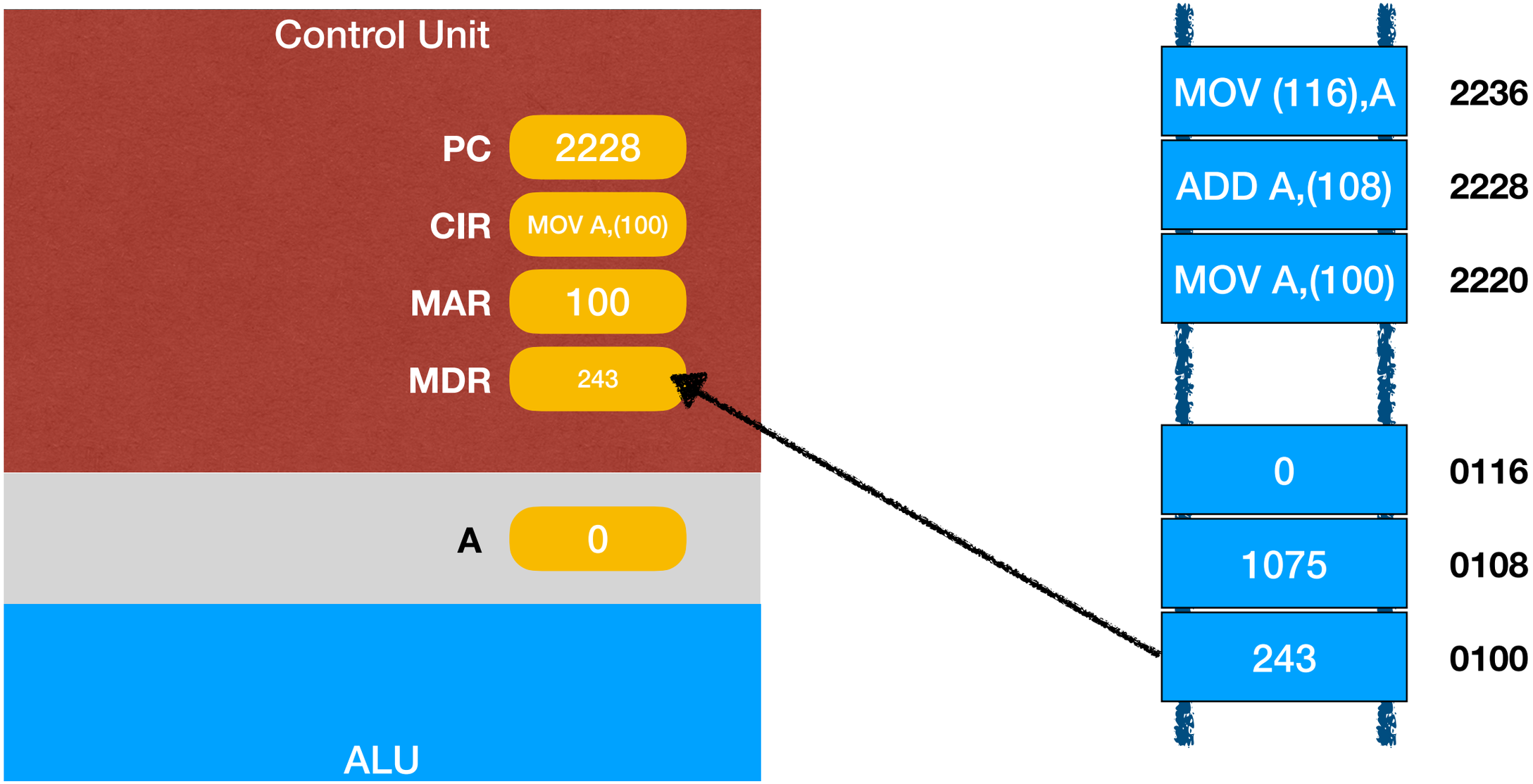


# Fetch-Decode-Execute cycle

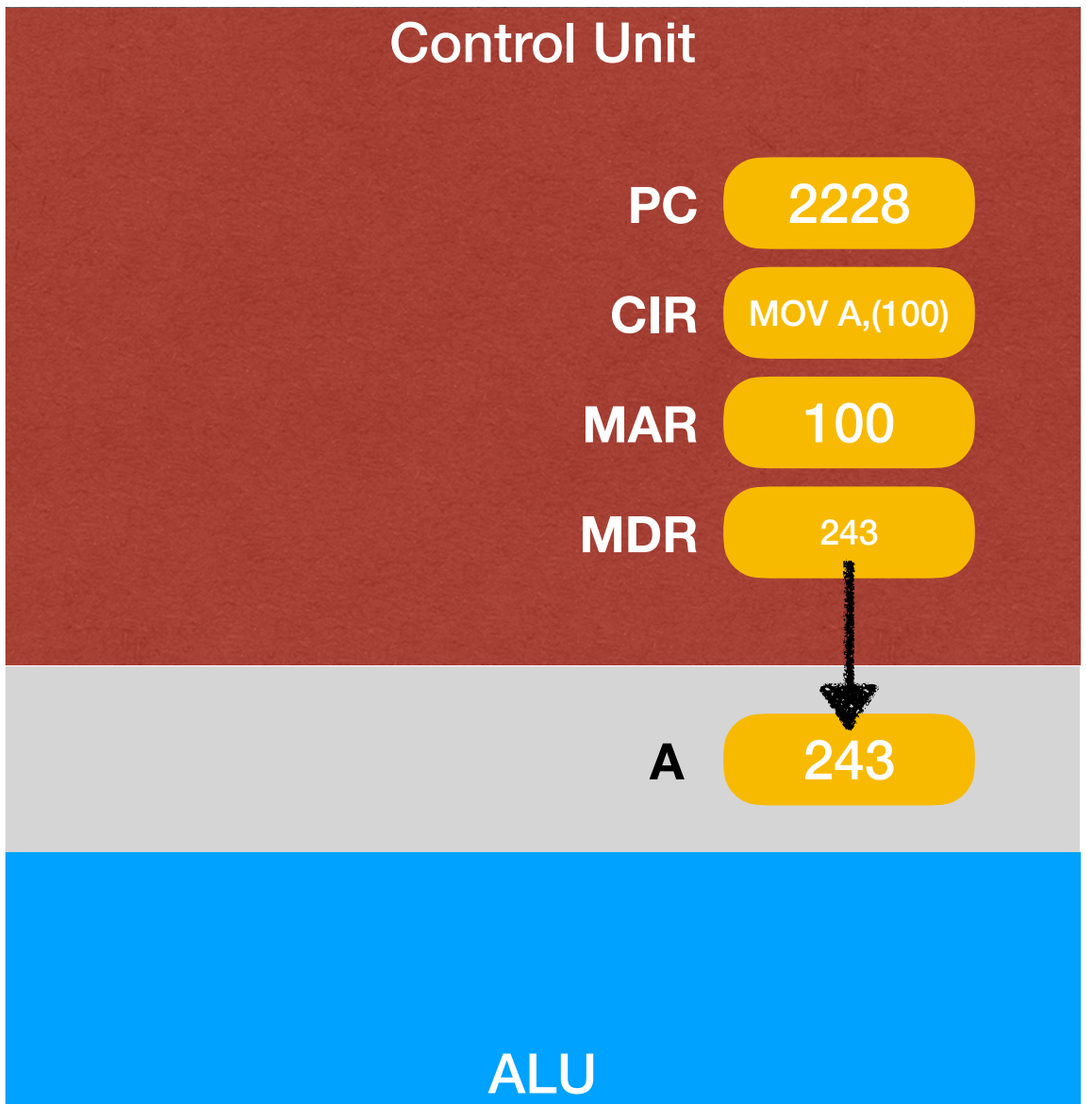


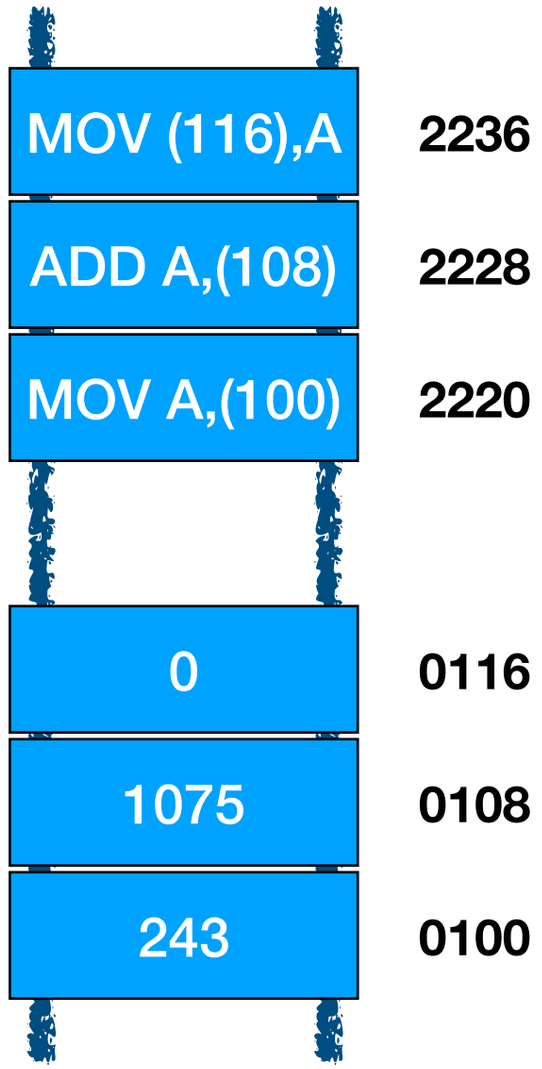



# Fetch-Decode-Execute cycle

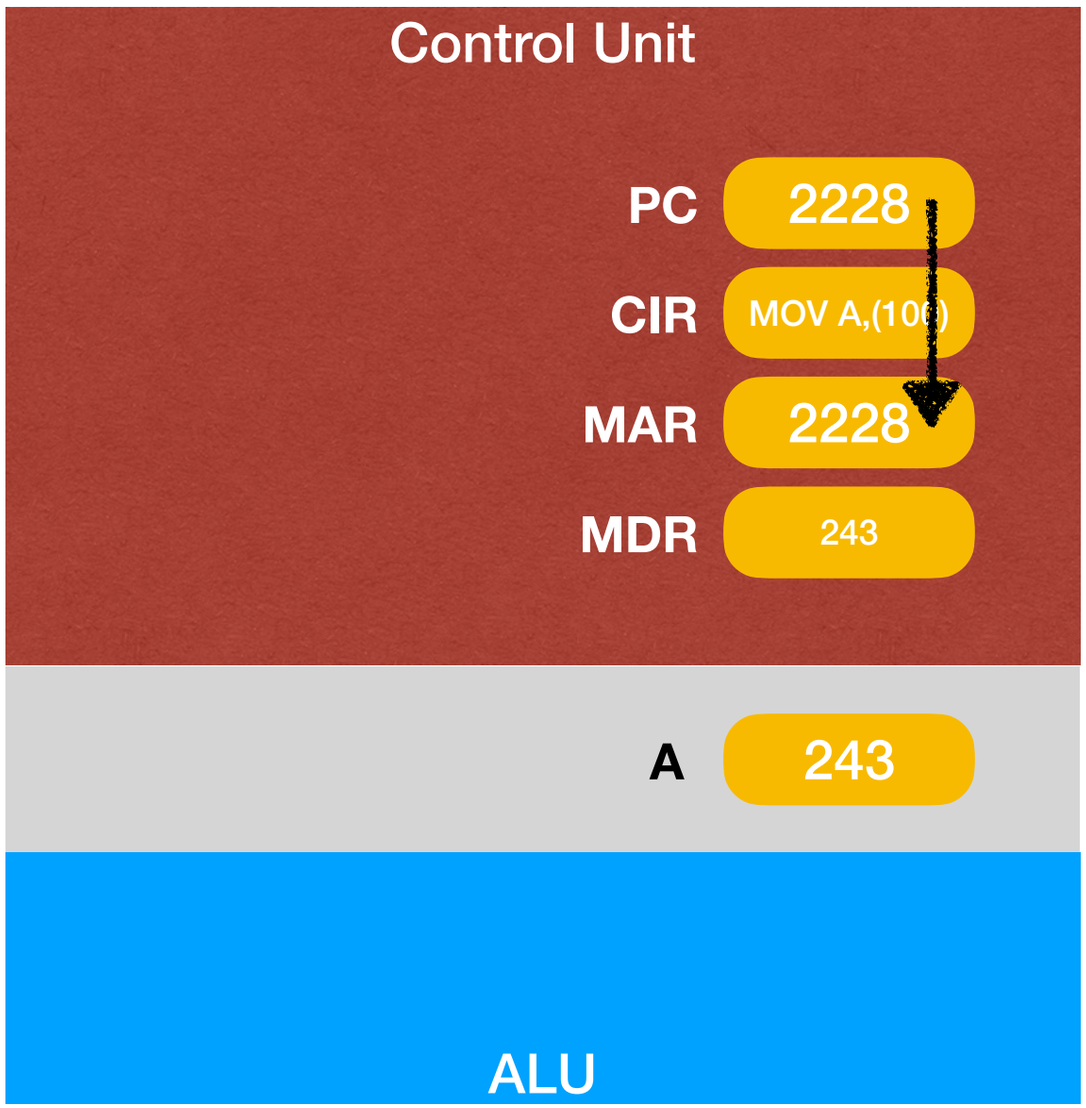


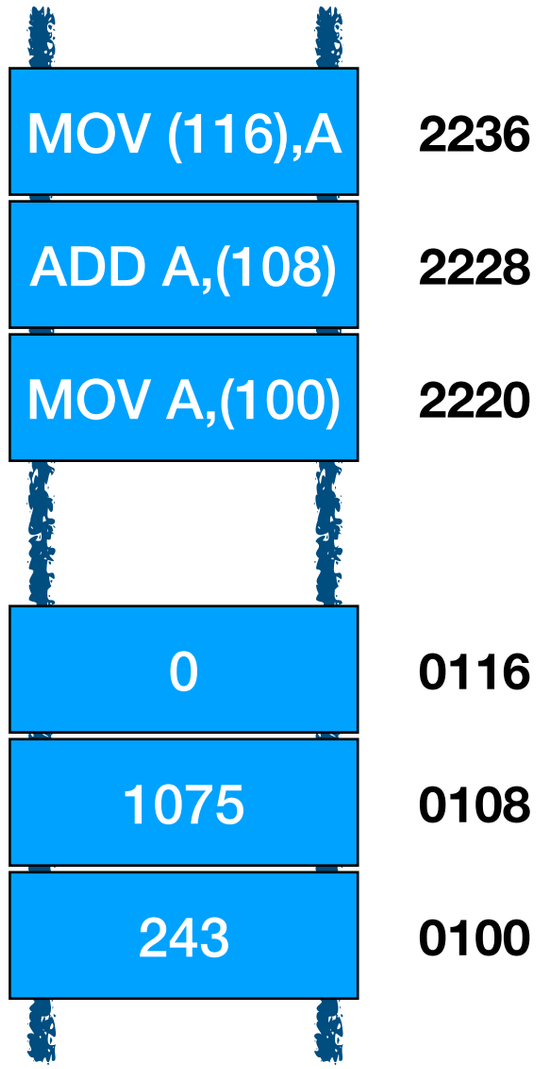
# Fetch-Decode-Execute cycle



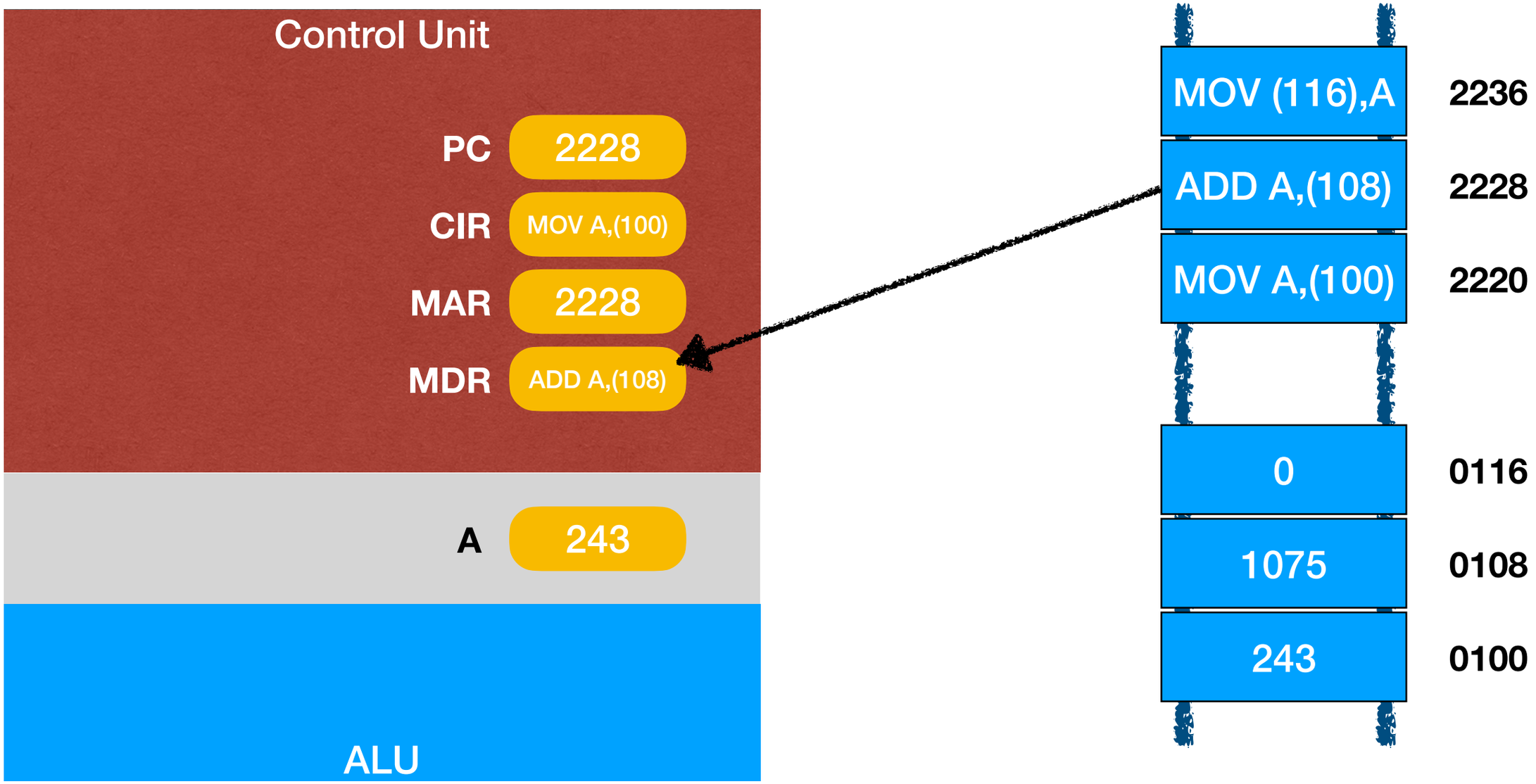



# Fetch-Decode-Execute cycle

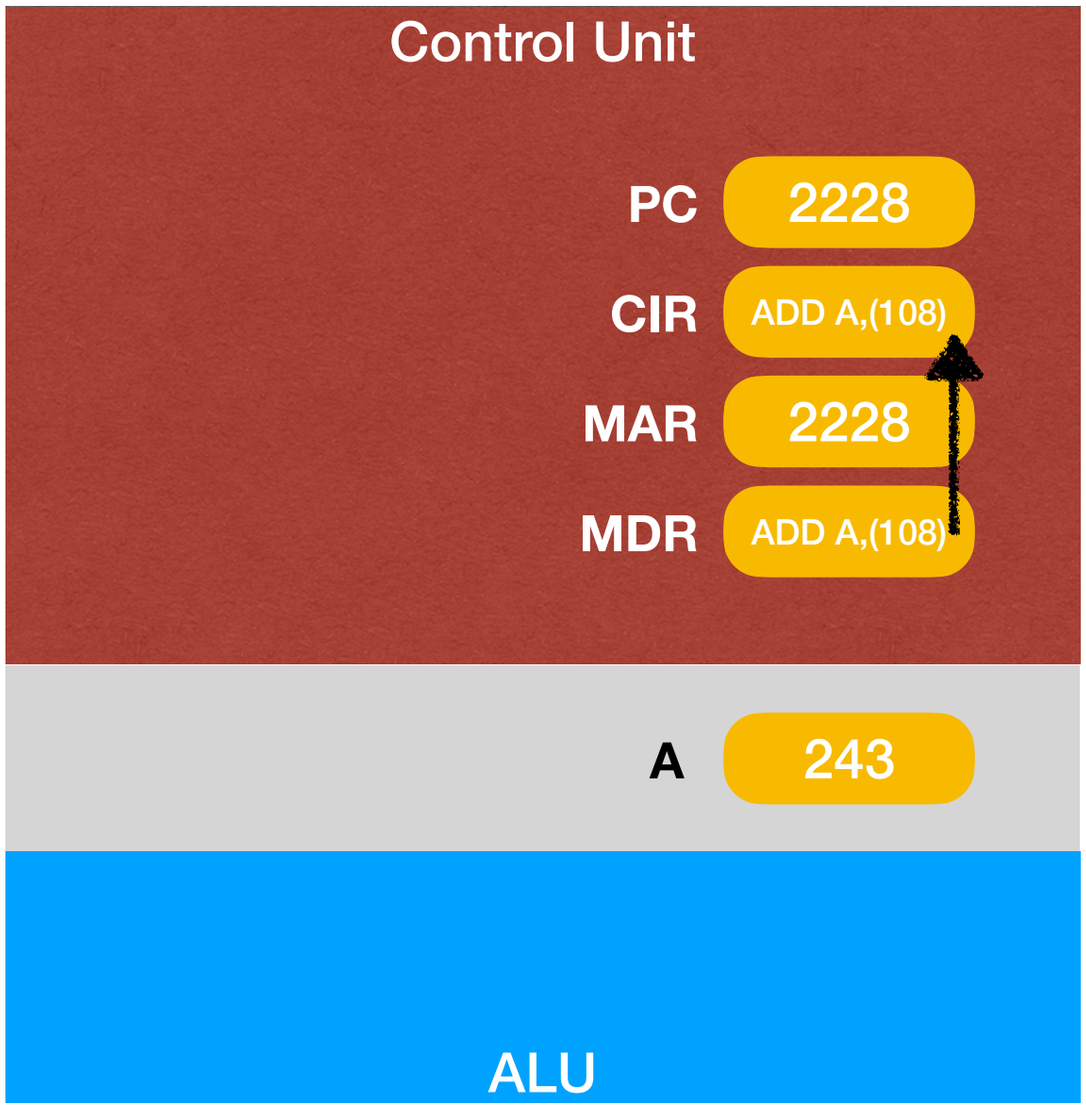


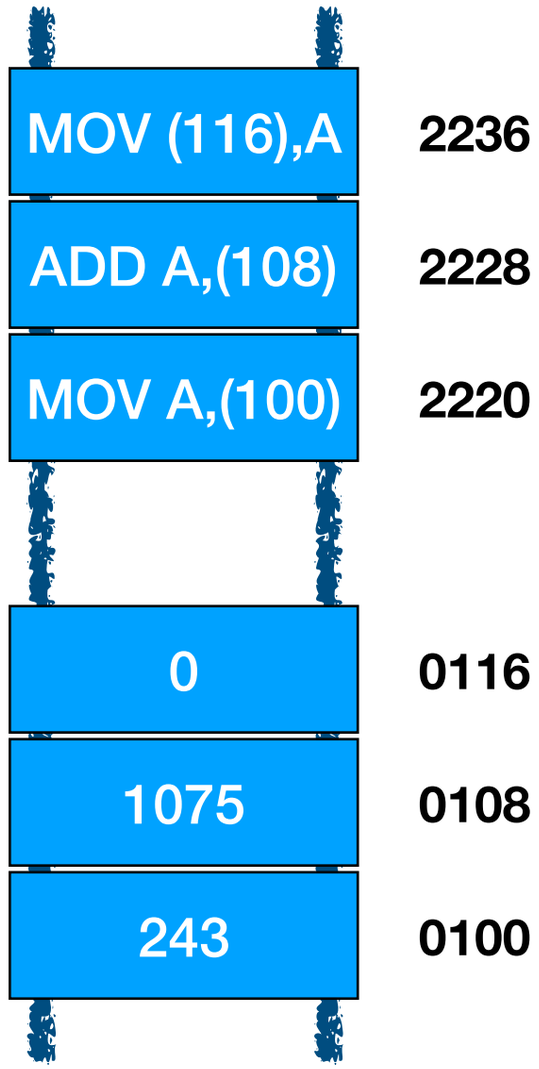



# Fetch-Decode-Execute cycle

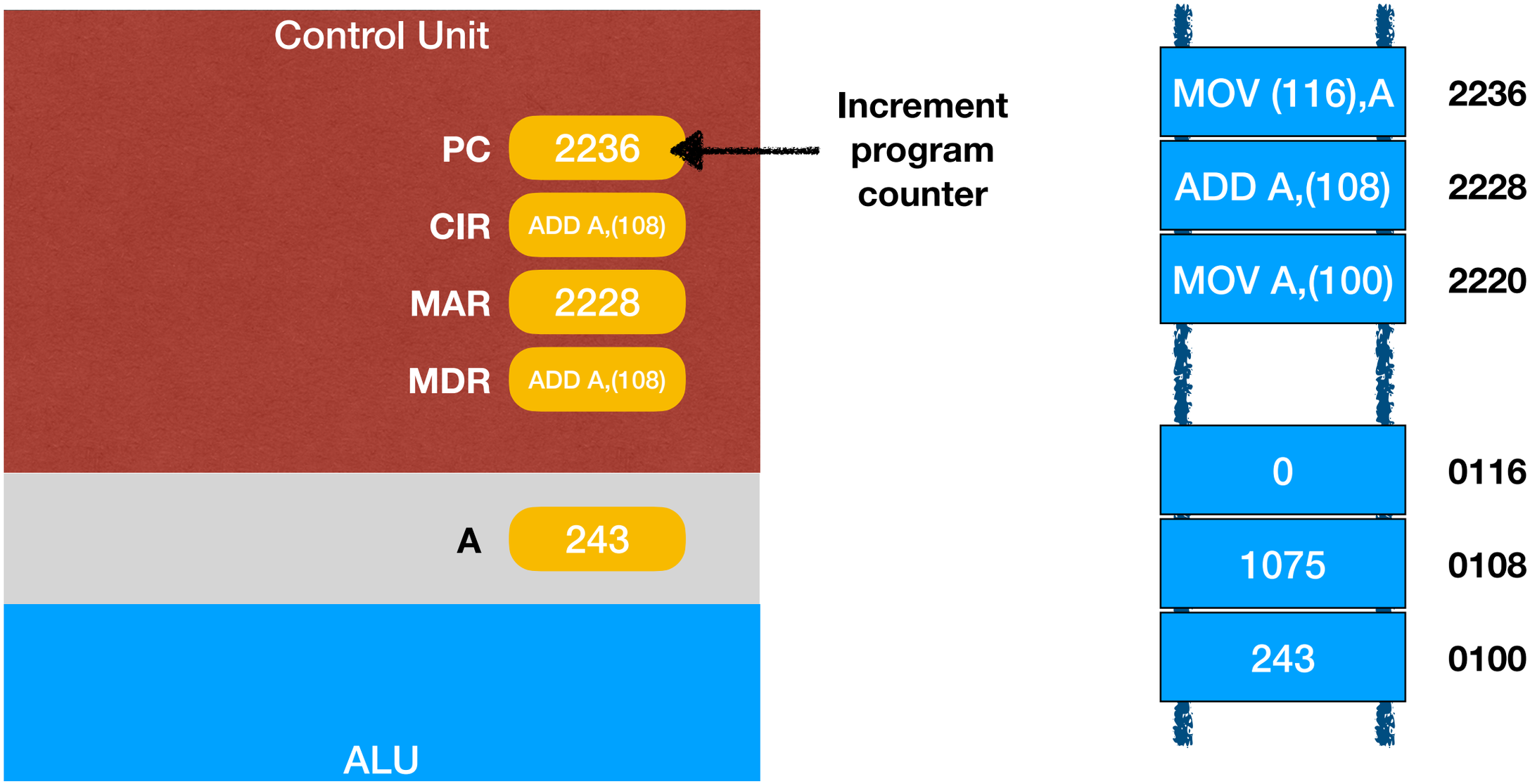


# Fetch-Decode-Execute cycle

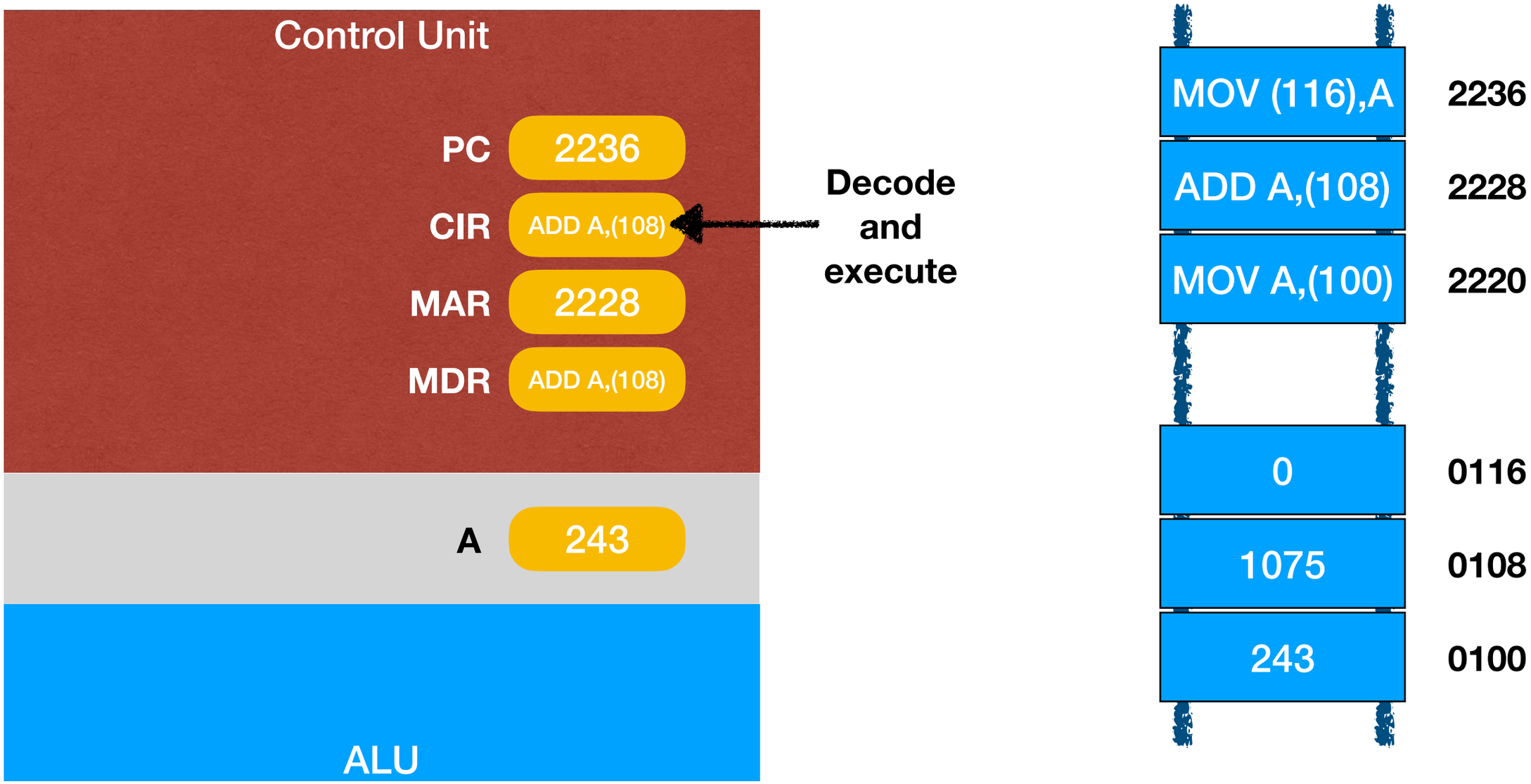


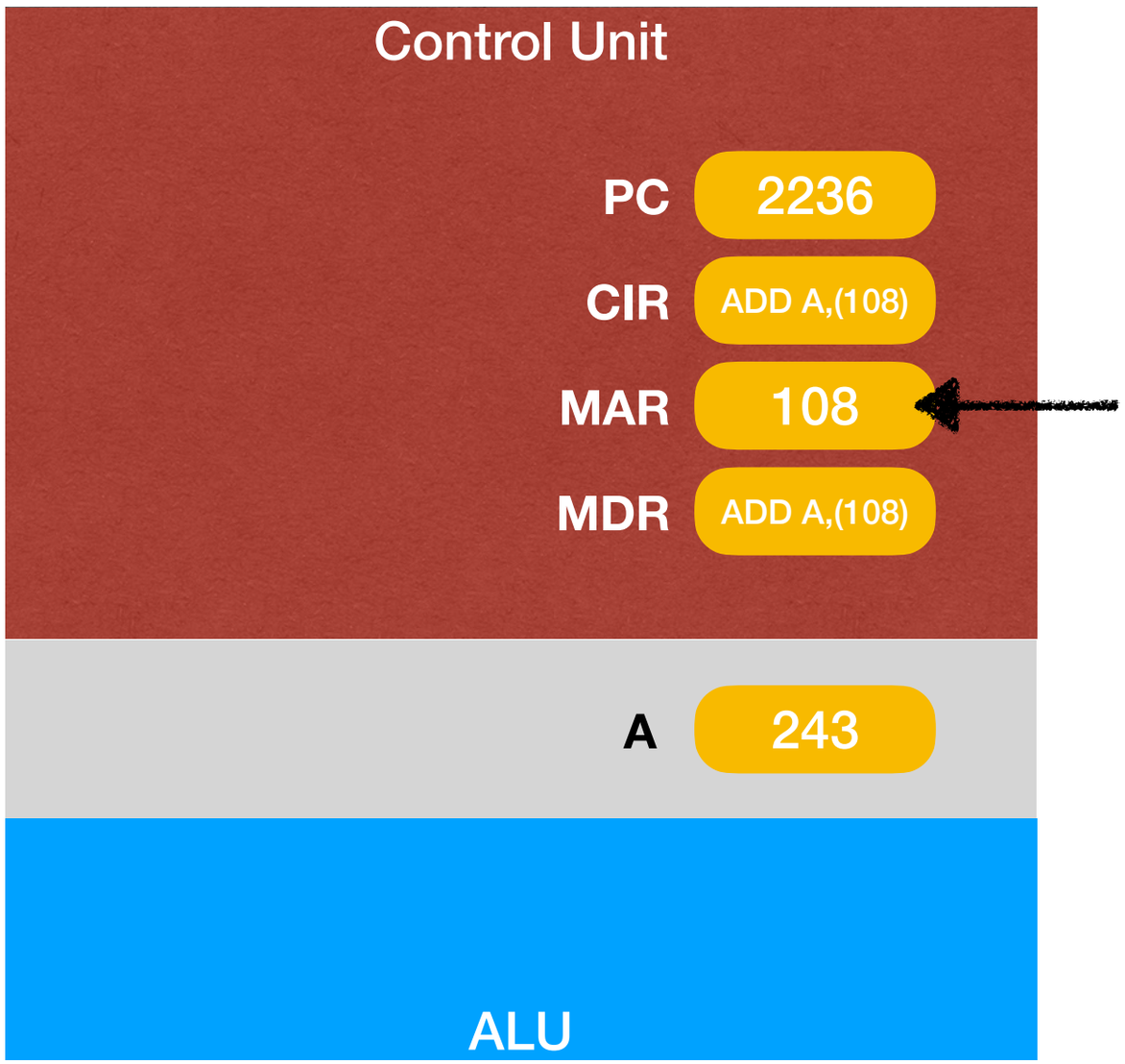
# Fetch-Decode-Execute cycle

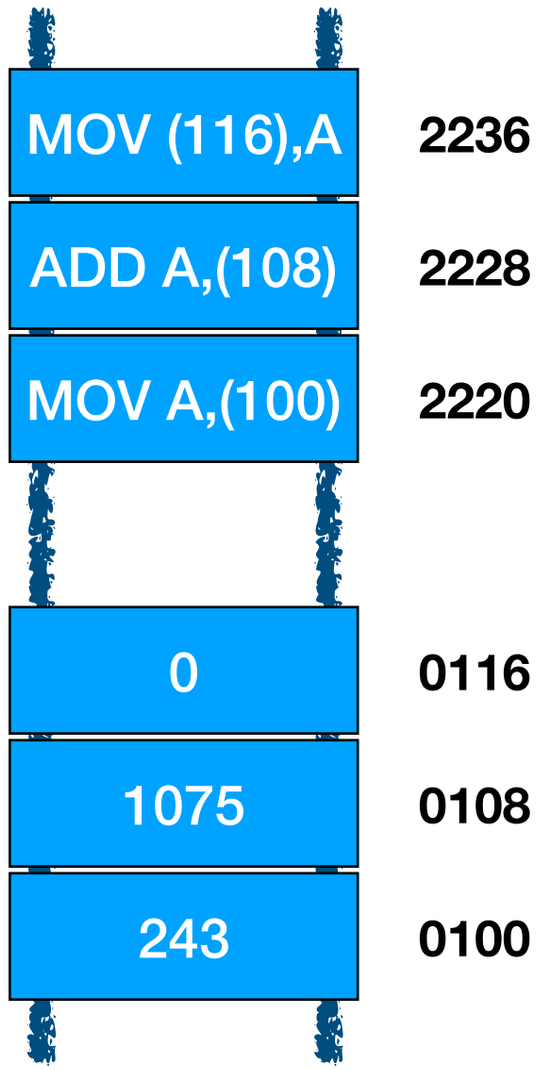


# Fetch-Decode-Execute cycle

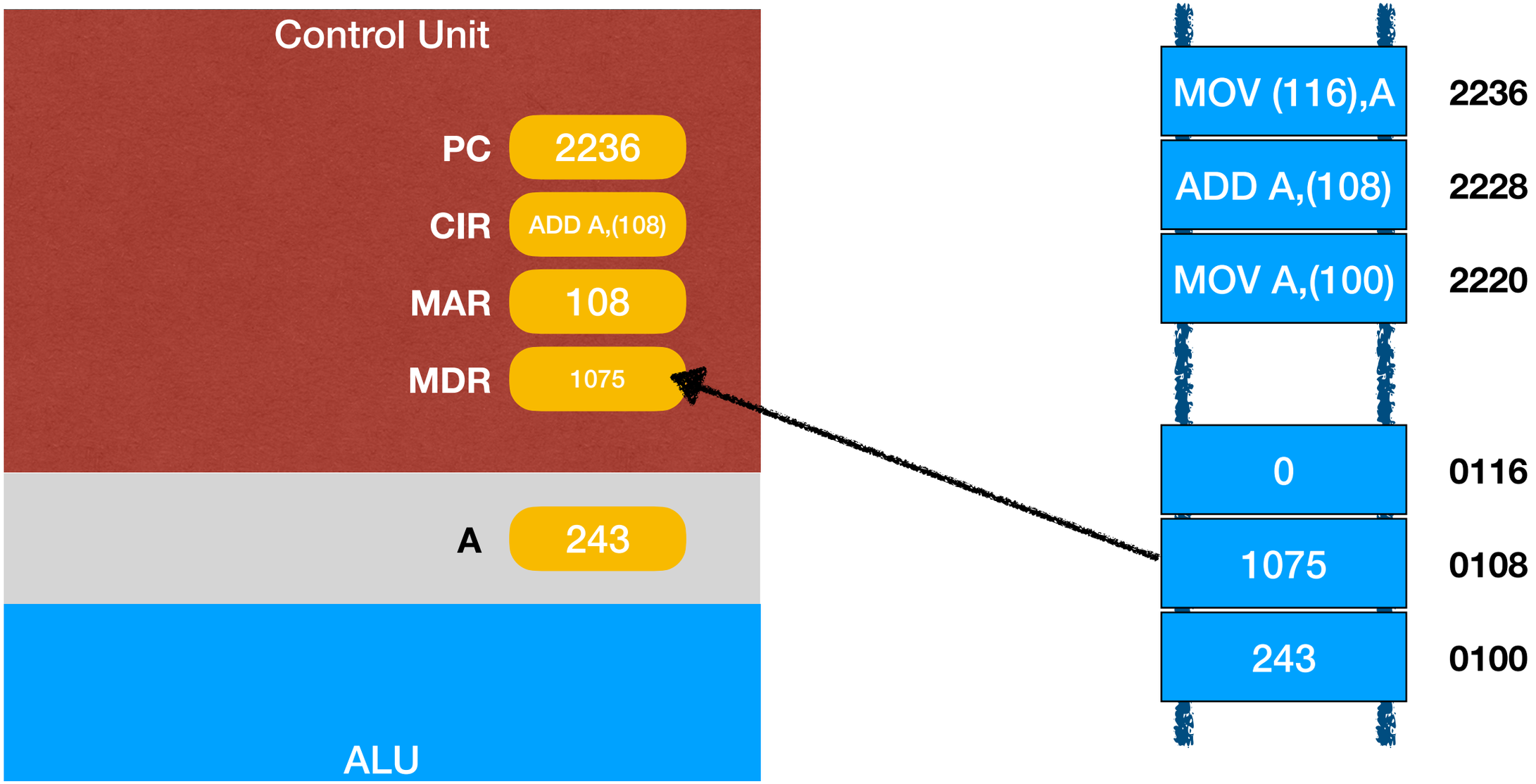


# Fetch-Decode-Execute cycle

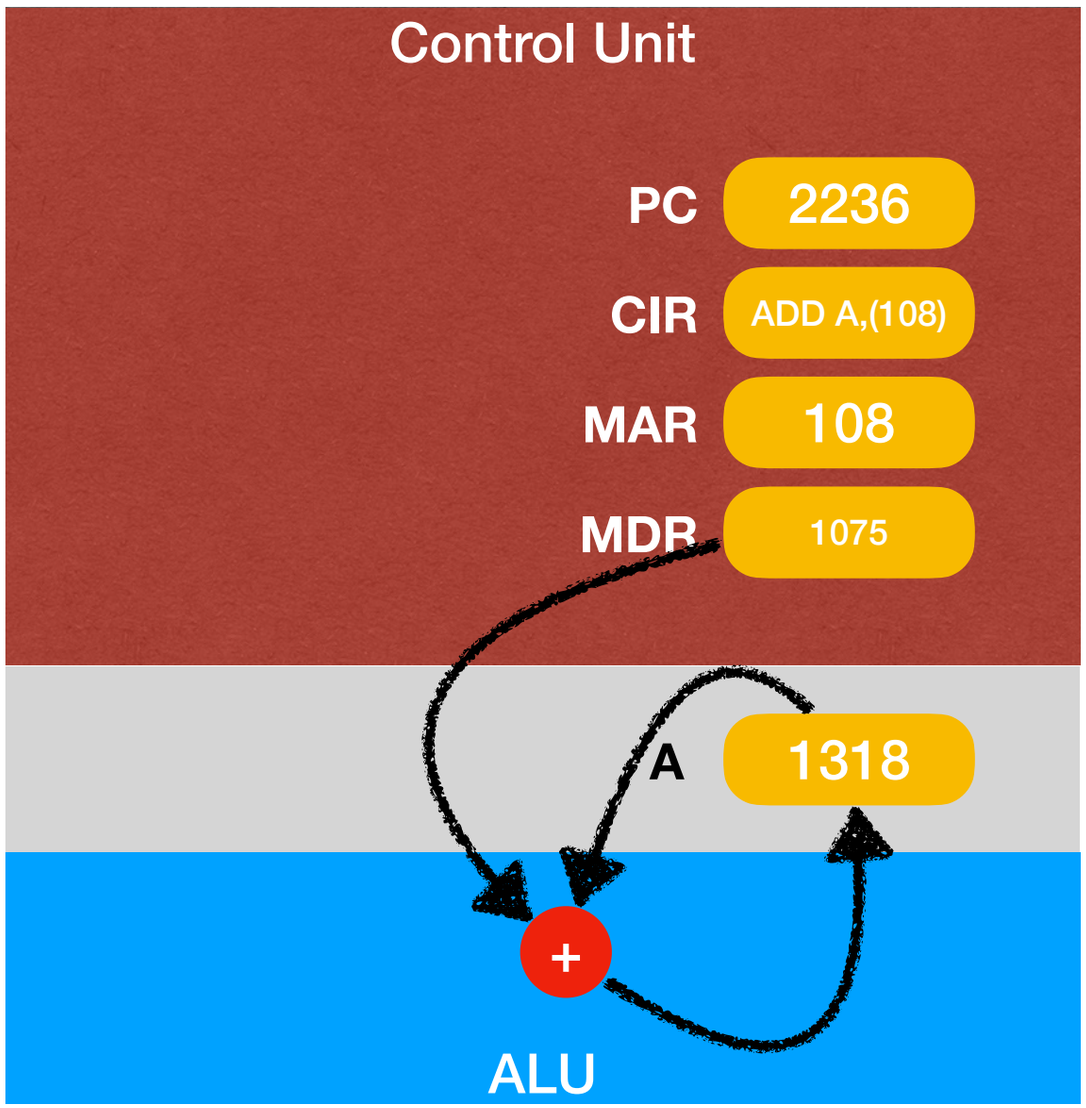


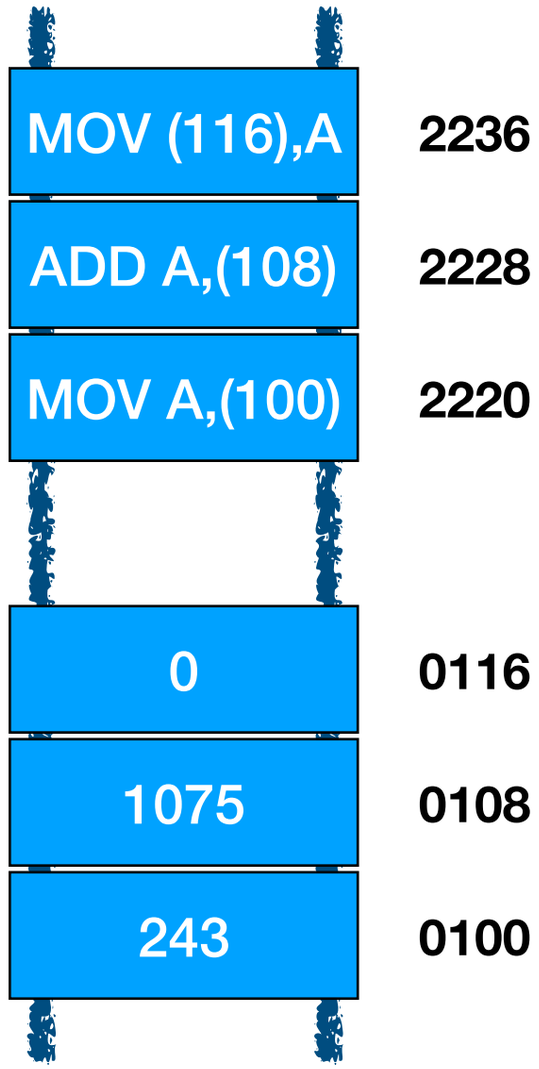



# Fetch-Decode-Execute cycle

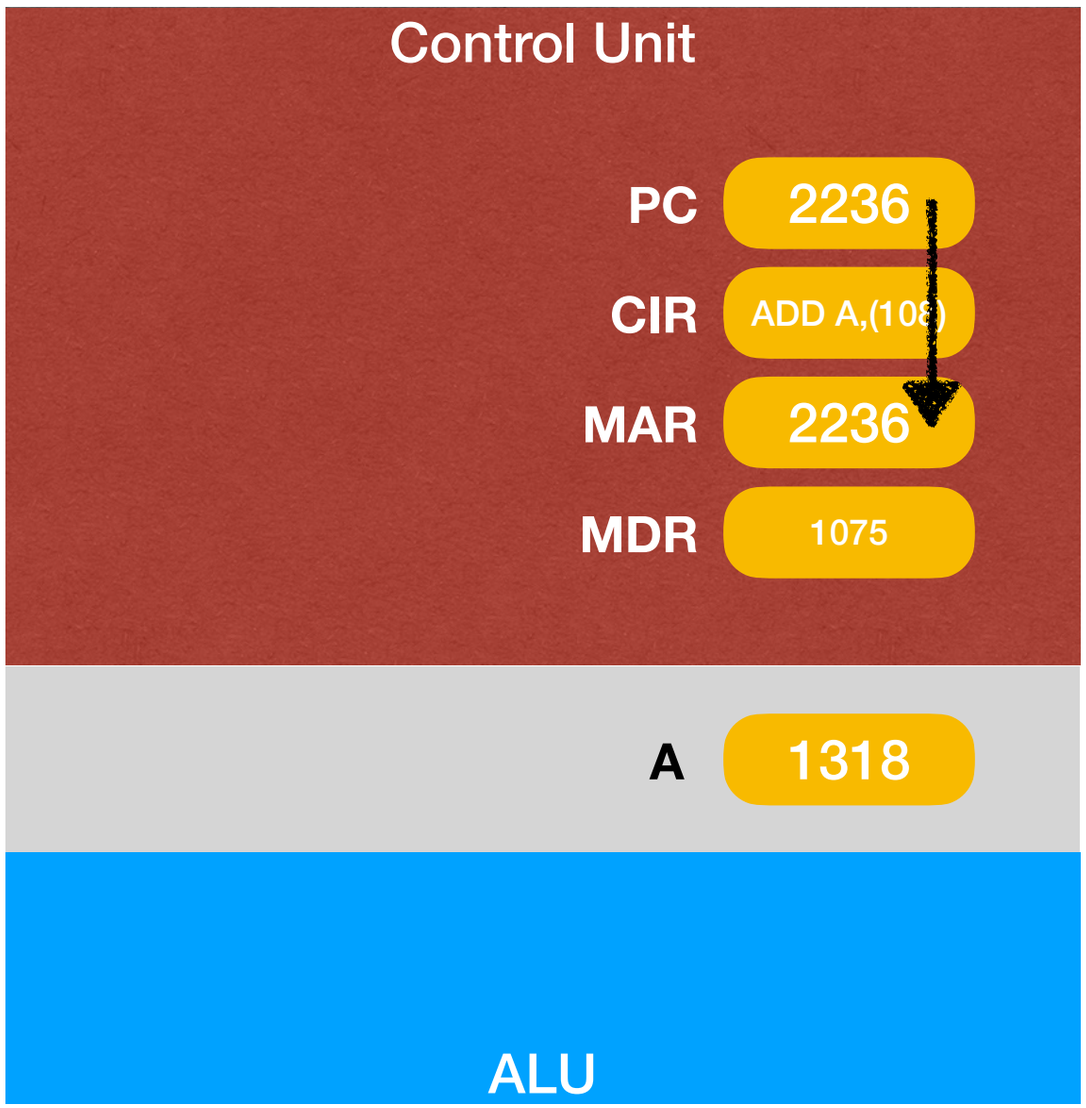


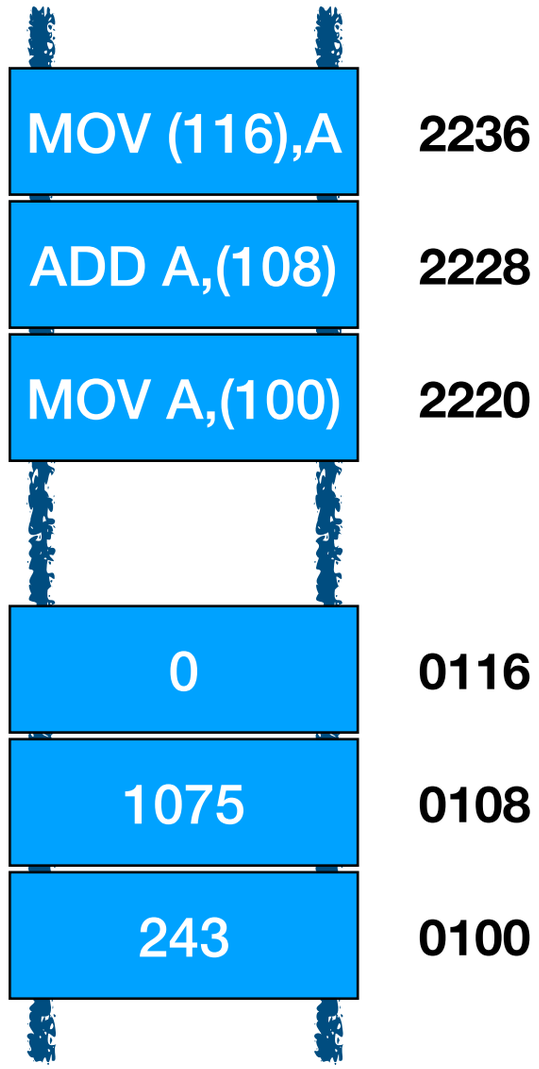
# Fetch-Decode-Execute cycle



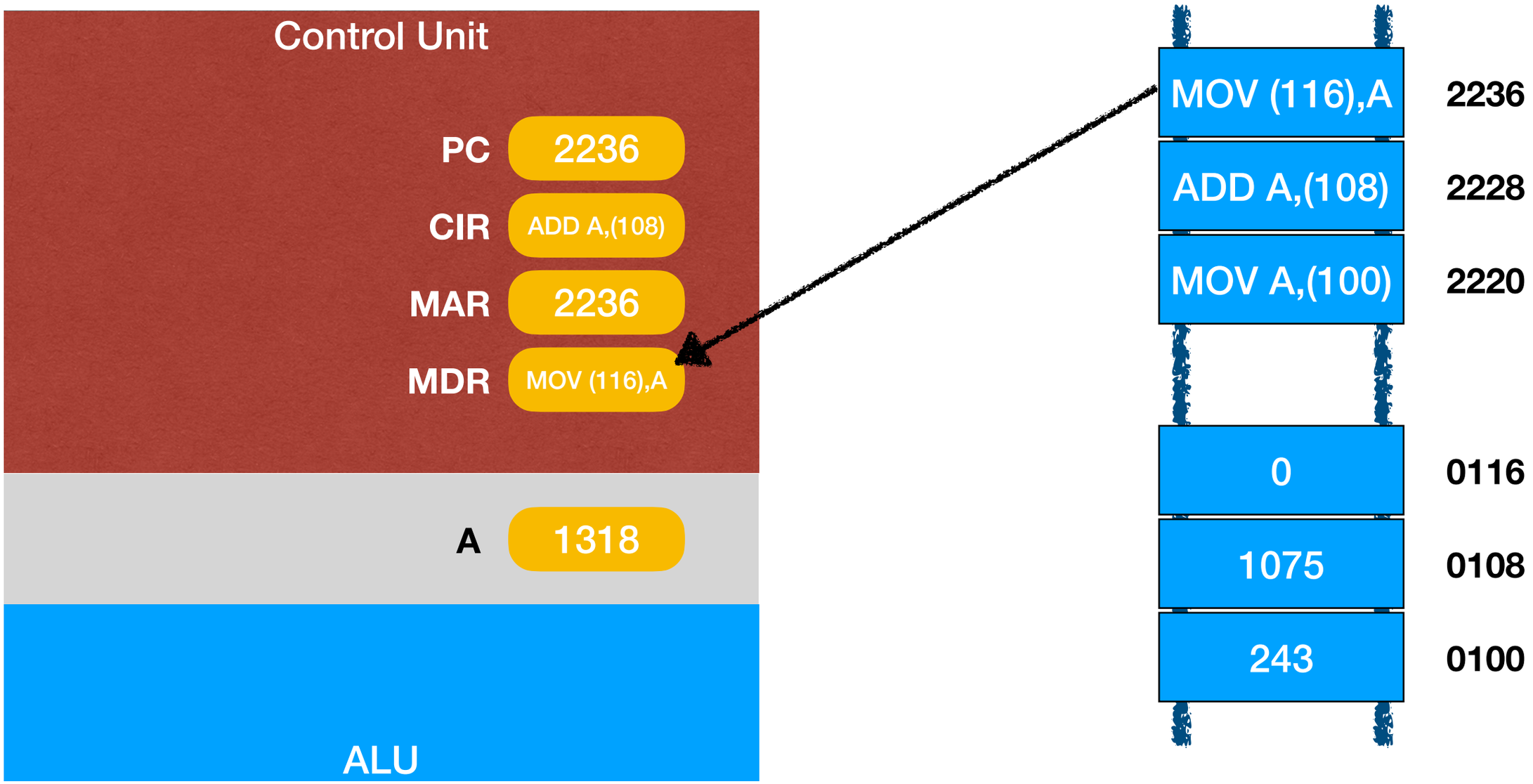



# Fetch-Decode-Execute cycle

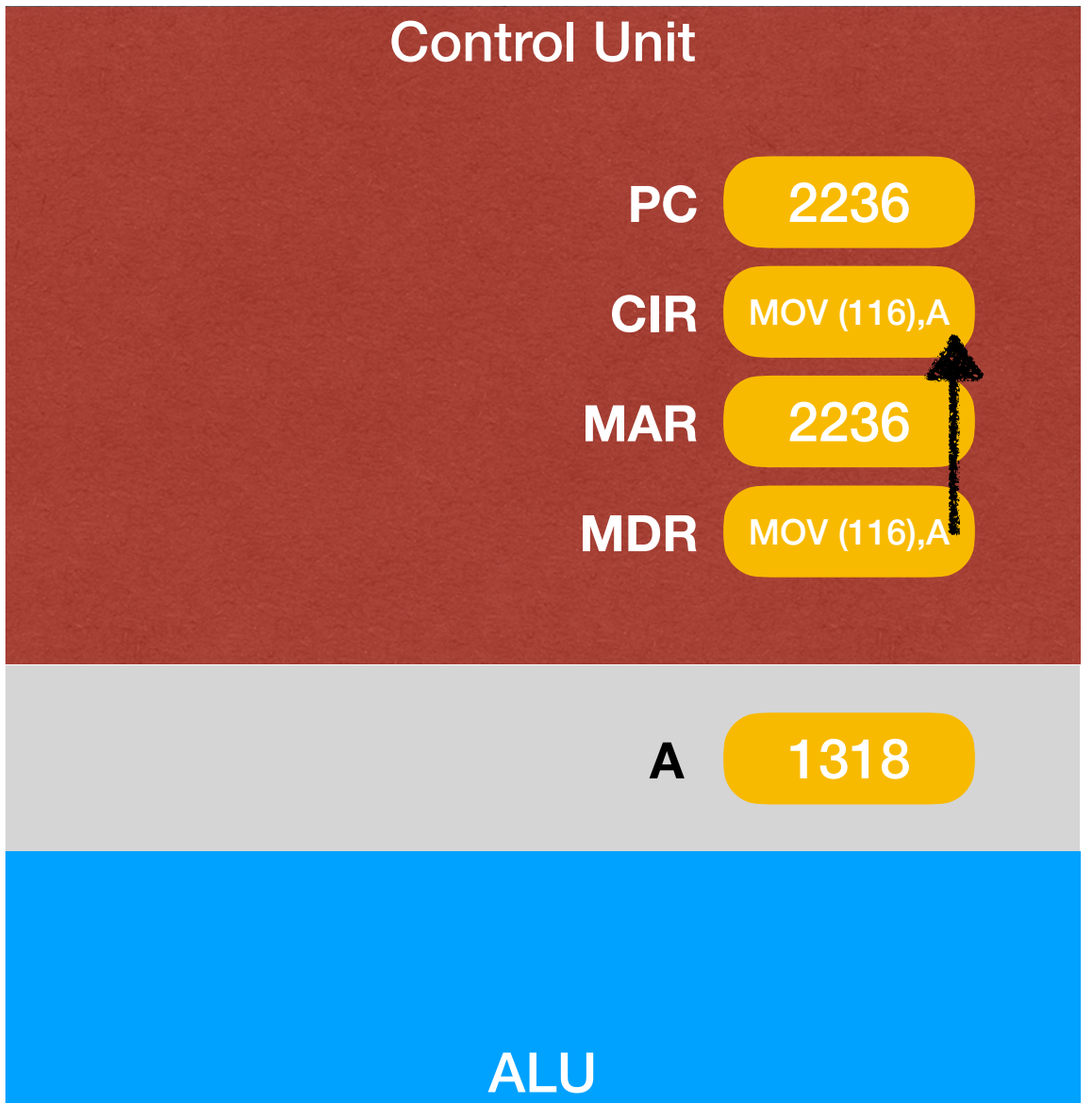


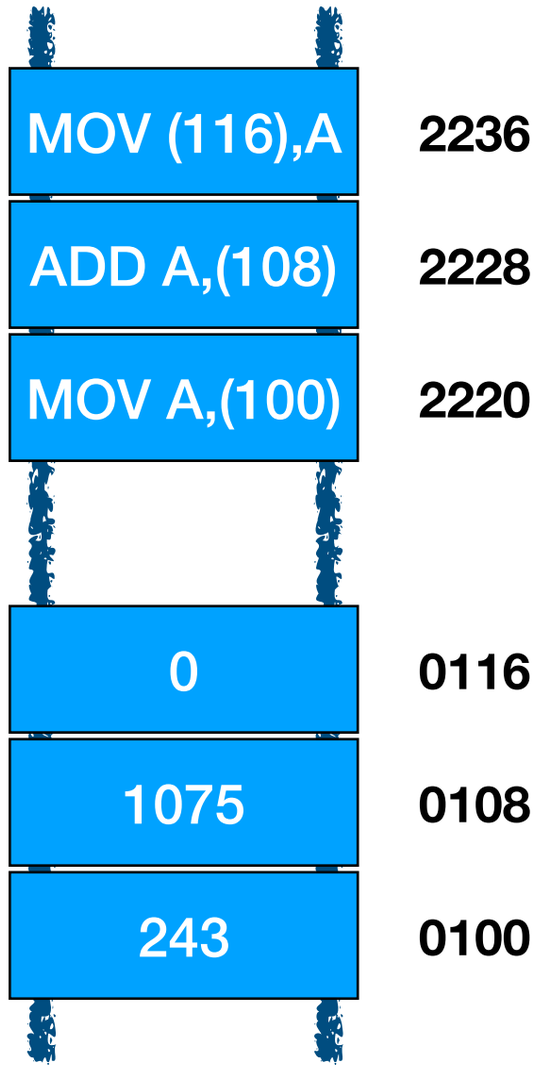



# Fetch-Decode-Execute cycle

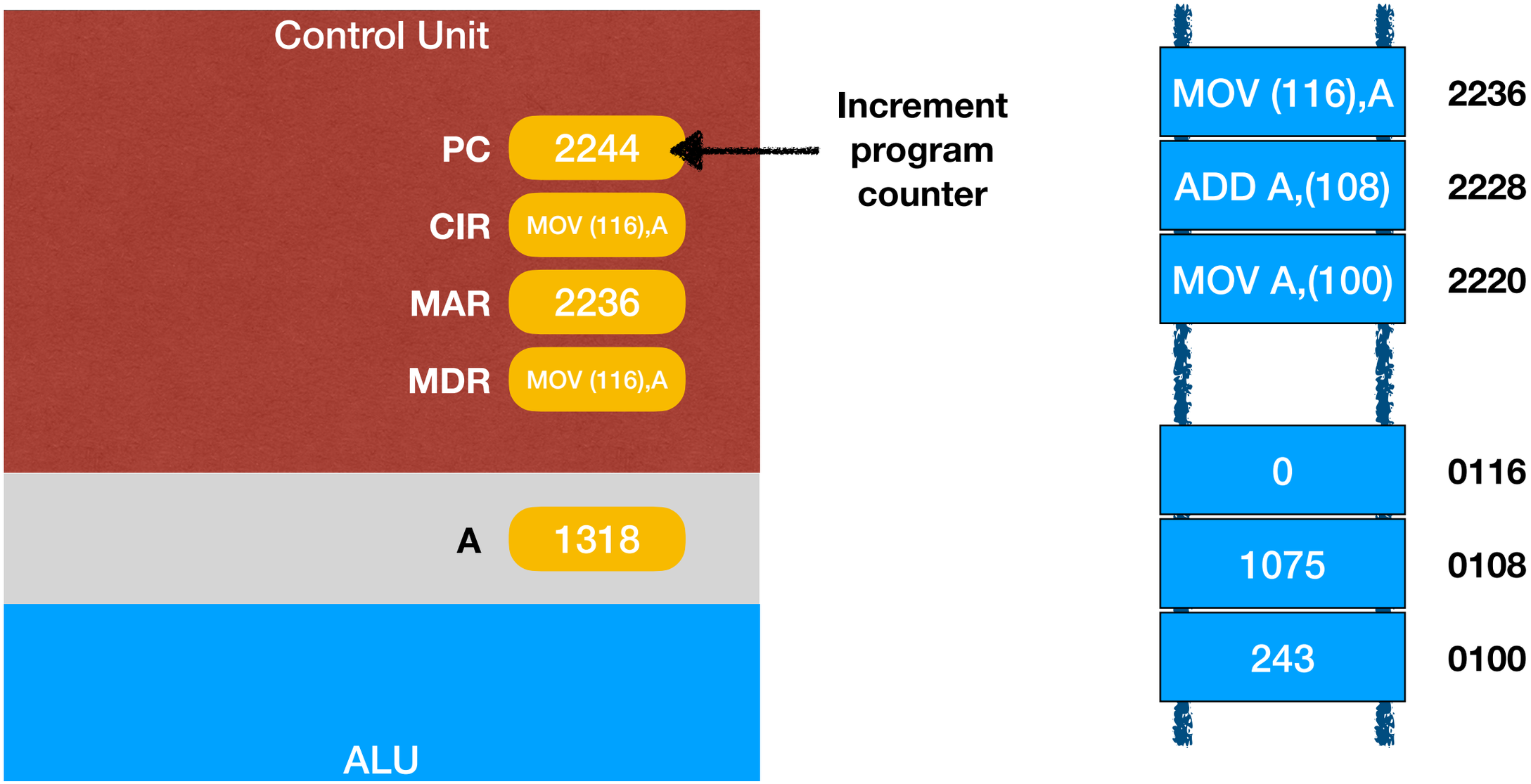


# Fetch-Decode-Execute cycle

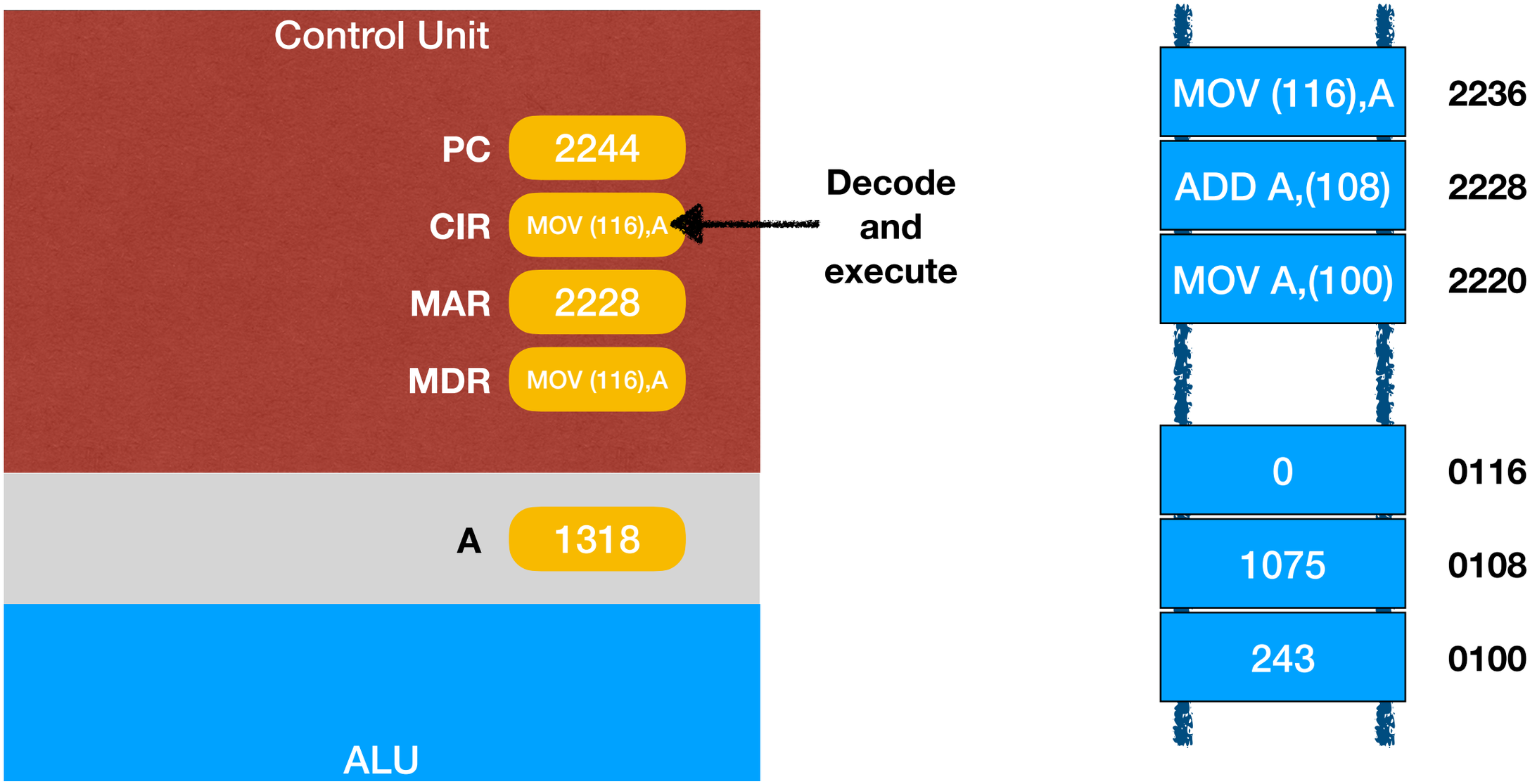


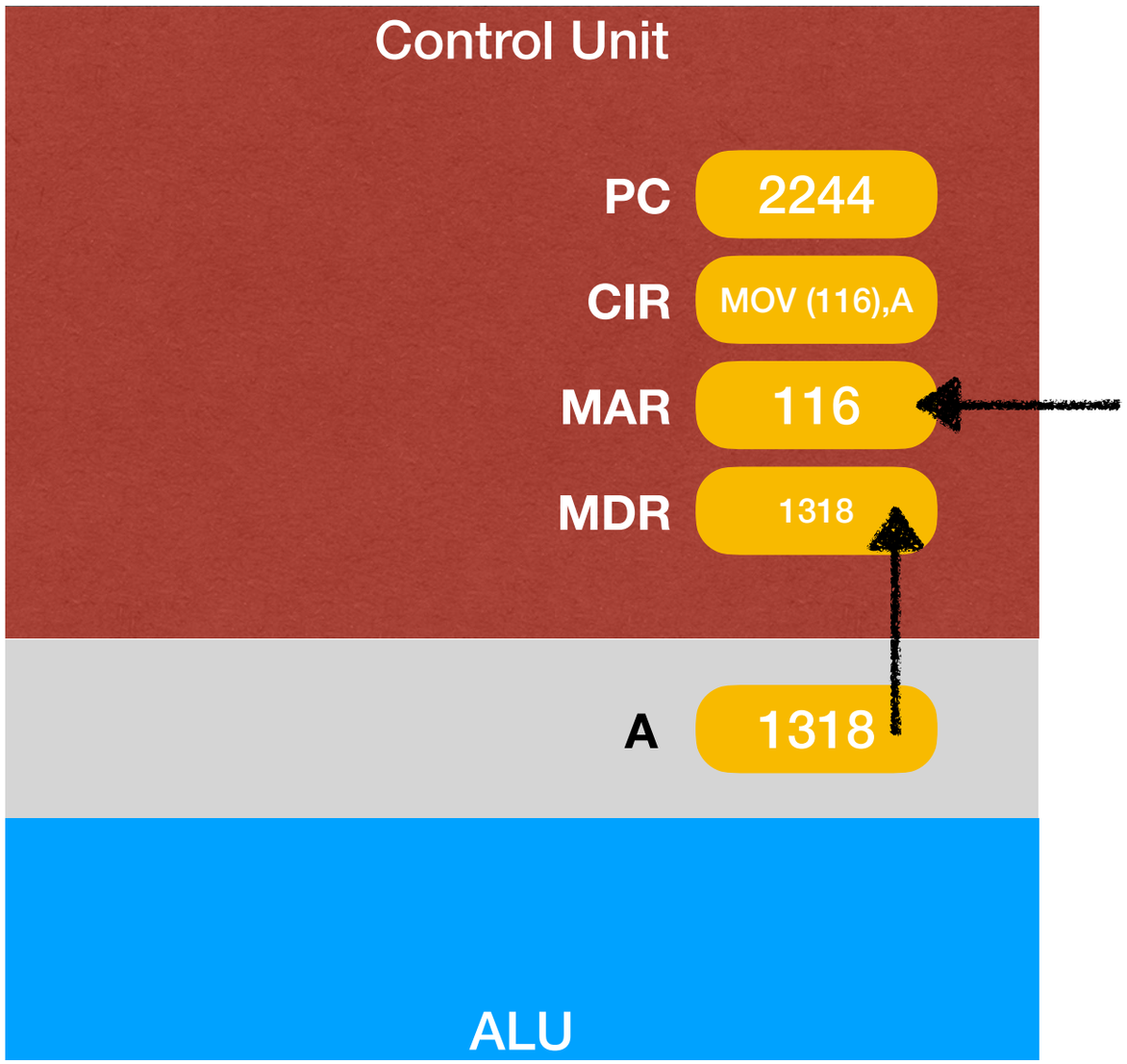
# Fetch-Decode-Execute cycle

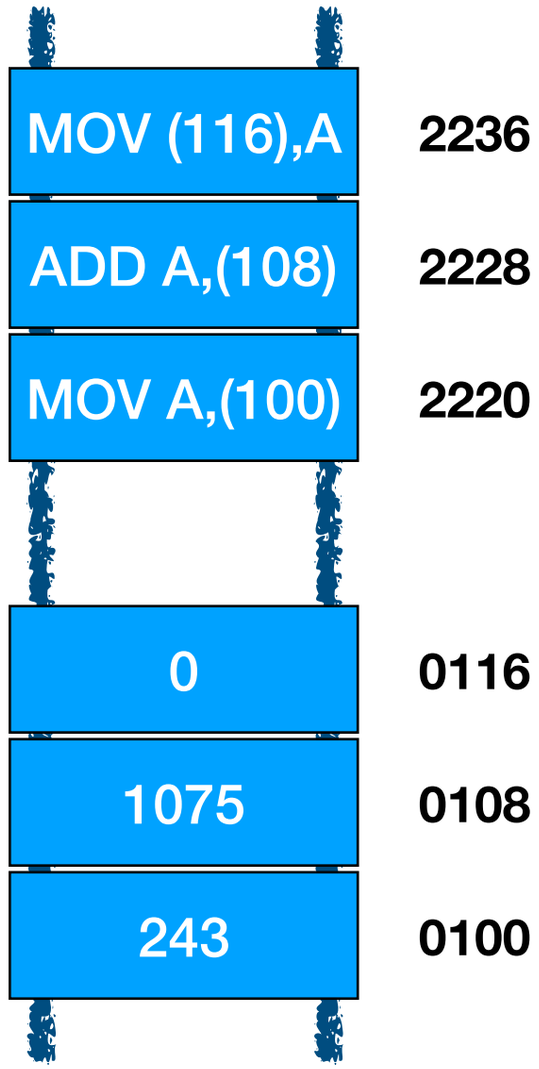


# Fetch-Decode-Execute cycle

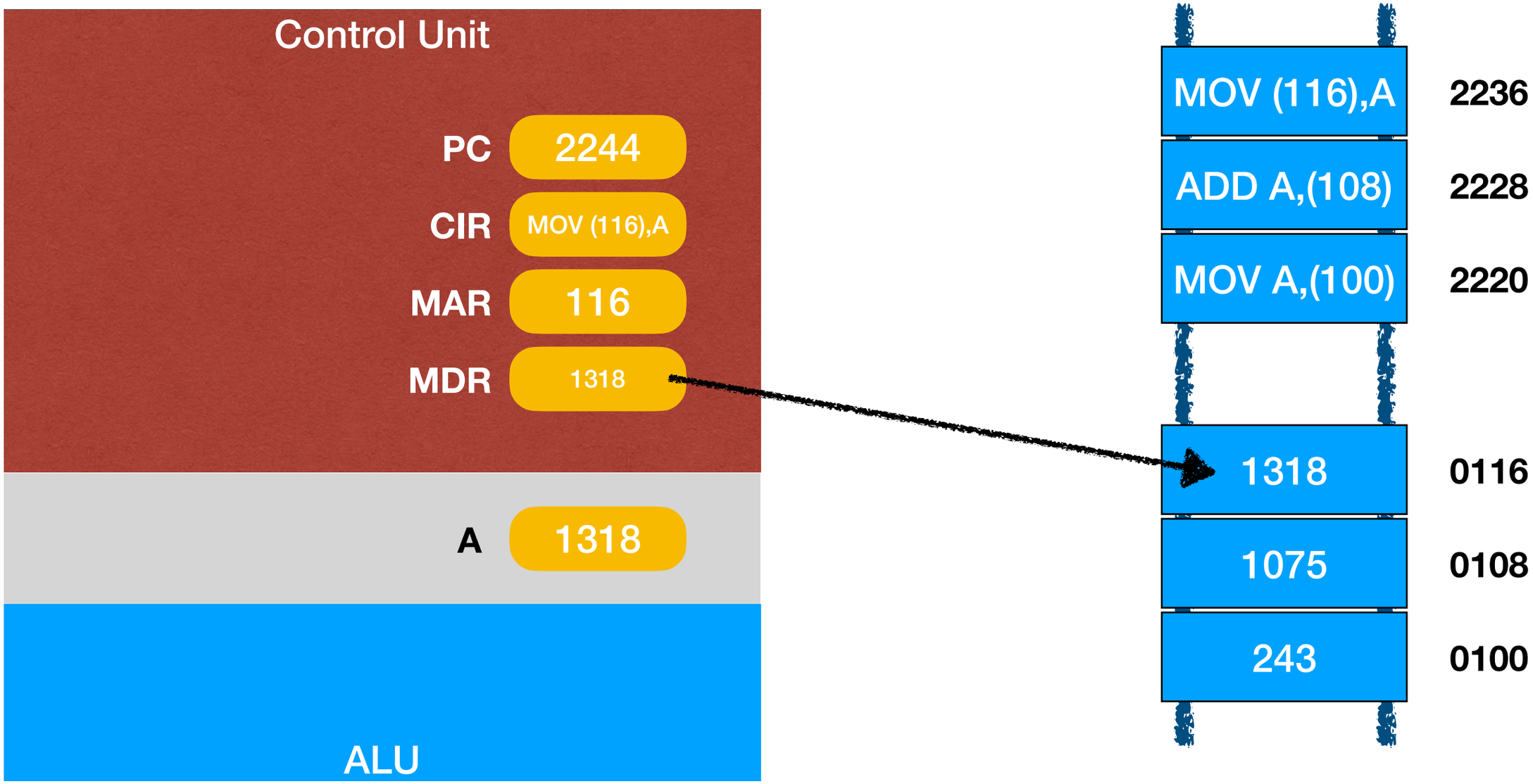


# Fetch-Decode-Execute cycle



# Fetch-Decode-Execute cycle



# Improving CPU performance

- Certain techniques are applied to improve the CPU performance, namely:
    - instruction pipelining
    - superscalar processor
    - caches
    - multiple cores on a chip
    - vector processing / GPUs

# Instruction pipelining

- At each clock cycle, different stages of the fetch-decode-execute cycle are carried out in parallel.

- These stages are handled by separate circuits, allowing the execution time for a set of instructions to be significantly reduced.

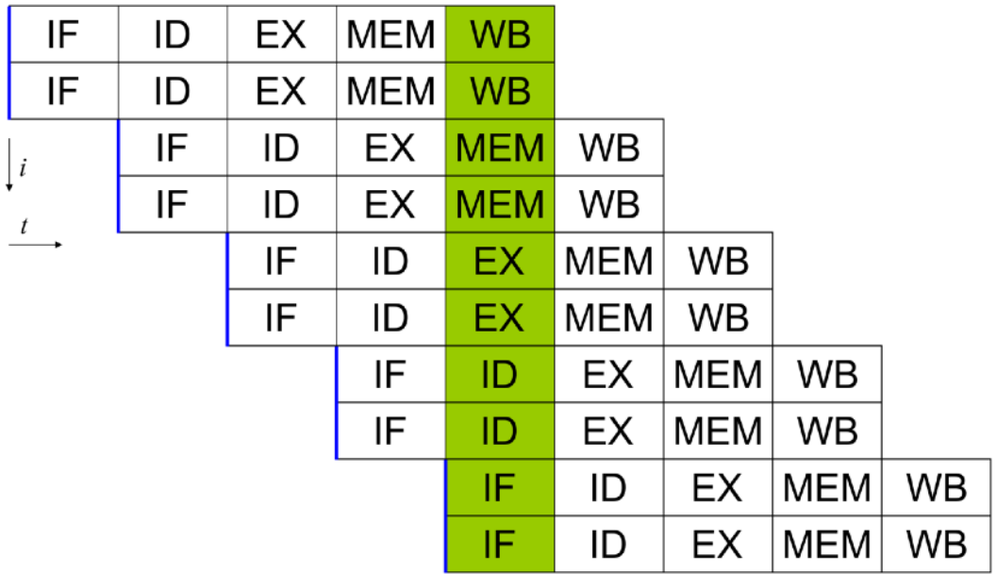
| Instr. No. | Pipeline Stage | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | IF | ID | EX | MEM | WB | | |
| 2 | | IF | ID | EX | MEM | WB | |
| 3 | | | IF | ID | EX | MEM | WB |
| 4 | | | | IF | ID | EX | MEM |
| 5 | | | | | IF | ID | EX |
| Clock Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

# Superscalar processor

- A superscalar processor features multiple execution units but only a single control unit. These execution units include:
    - Arithmetic/Logic Units (ALU) for integer operations,
    - Floating Point Units (FPU) for floating-point calculations,
    - Load/Store Units for memory operations,
    - Branch Units for handling conditional flow.
- The processor can dispatch multiple instructions **from the same program** for execution in each clock cycle, to enhance performance.

# Superscalar processor

- A superscalar processor achieves higher performance by exploiting **instruction-level parallelism** (ILP), where independent instructions are executed in parallel rather than sequentially.



# Superscalar processor

- **Dynamic Scheduling.** The processor dynamically decides which instructions to execute in parallel, based on instruction dependencies, availability of execution units, and other factors.

- **Instruction Reordering.** Superscalar processors use techniques to track dependencies between instructions and reorder them for maximum parallelism without violating program correctness.

- **Branch Prediction.** Superscalar processors use advanced branch prediction to minimize delays from branch instructions, allowing speculative execution to maintain instruction flow.

# Cache memory

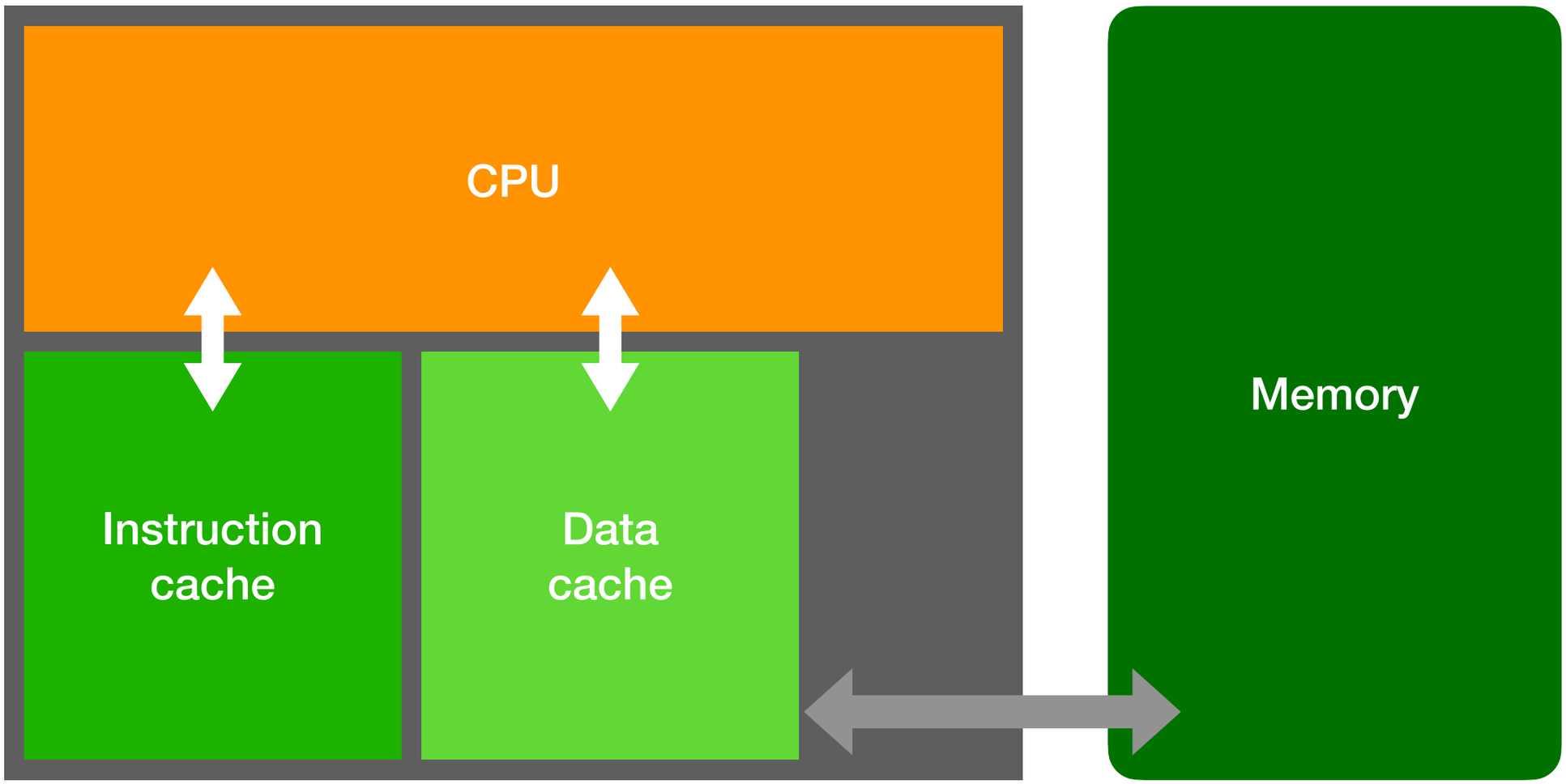
- Cache memory is small, high-speed memory located close to the CPU (in the same chip).

- Reduces the time the CPU takes to access data from slower main memory (RAM), allowing the CPU to fetch data more quickly.

- Used to store frequently accessed data (data cache) and instructions (instruction cache).

- A **cache miss** occurs when the needed data is not in the cache, forcing access to slower memory.

# Cache memory



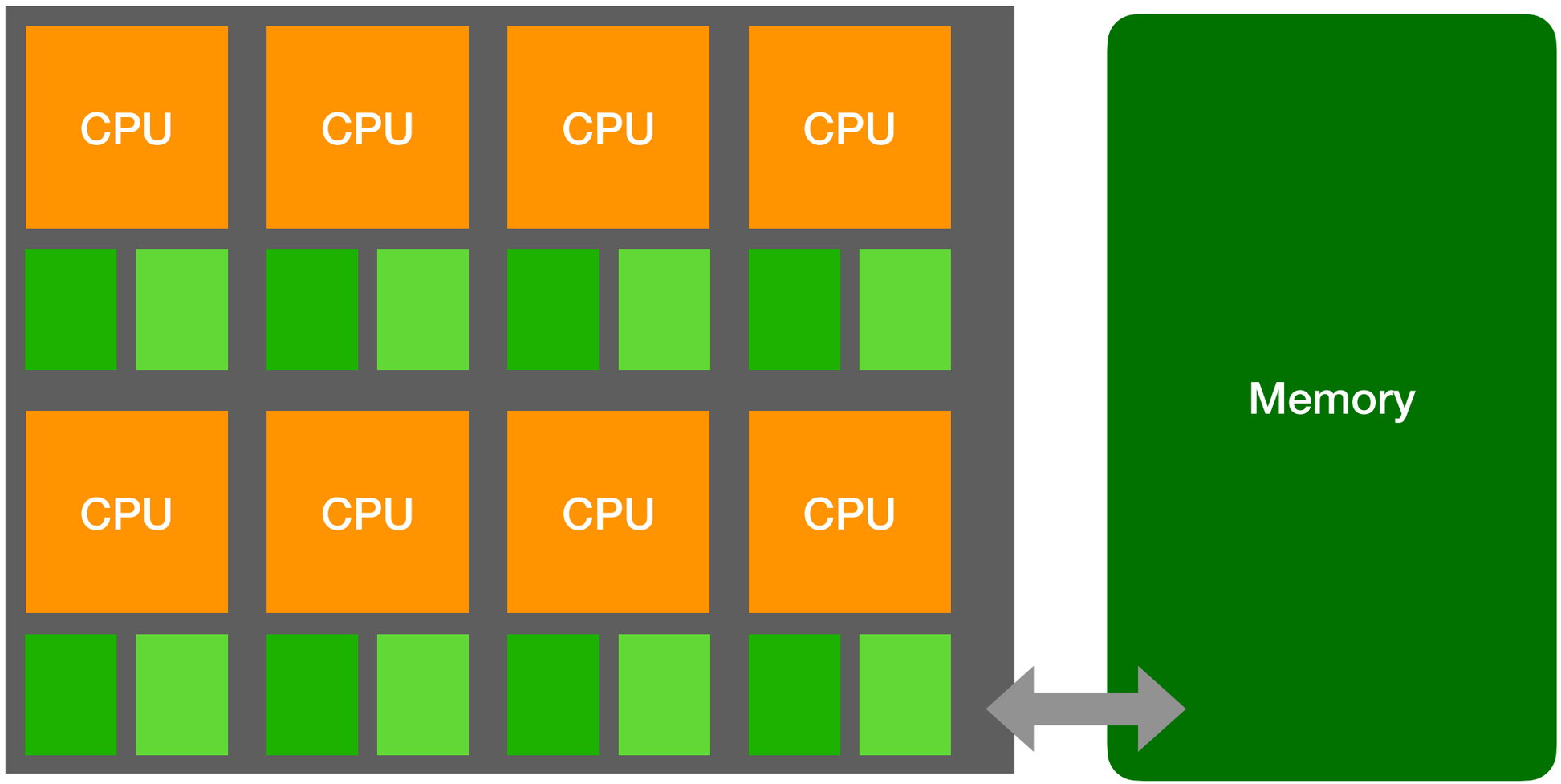
# Multiple core chips

- Moore's Law is an observation made by Gordon Moore, co-founder of Intel, in 1965.
    - "The number of transistors on a microchip doubles approximately every two years, leading to an exponential increase in computing power while simultaneously reducing relative cost."
- In the early years of this century, power density reached a physical limit. The performance of a processor could not be enhanced by increasing the density of transistors or the clock speed.
- Solution: Integrate multiple low-power CPUs onto the same chip.

# Multiple core chips

- Moore's Law is an observation made by Gordon Moore, co-founder of Intel, in 1965.
    - "The number of transistors on a microchip doubles approximately every two years, leading to an exponential increase in computing power while simultaneously reducing relative cost."
- In the early years of this century, power density reached a physical limit. The performance of a processor could not be enhanced by increasing the density of transistors and, simultaneously, the clock speed.
- Solution: Integrate multiple low-power CPUs onto the same chip.

# Multiple core chips



# Multiple core chips

- The enhancement of computing power is realised by facilitating parallelism through:
  - Simultaneously executing multiple programs in true parallel.
  - Running different segments of the same program concurrently (multithreading).

# Graphics Processing Unit (GPU)

- GPU and the CPU are designed with different goals in mind.
    - CPUs are designed for general-purpose tasks, executing a sequence of operations, called a thread, as fast as possible. They are good at handling a wide range of operations such as managing the operating system and running programs.
    - GPUs are specialized for highly parallel computations (originally, 3D rendering) and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control.

# Graphics Processing Unit (GPU)

- CPUs typically feature fewer cores (4 to 16), optimised for sequential processing with higher clock speeds.
- GPUs contain thousands of smaller cores, designed to handle multiple tasks simultaneously through parallel processing:
    - One instruction is executed over multiple data (vectors).

# Graphics Processing Unit (GPU)

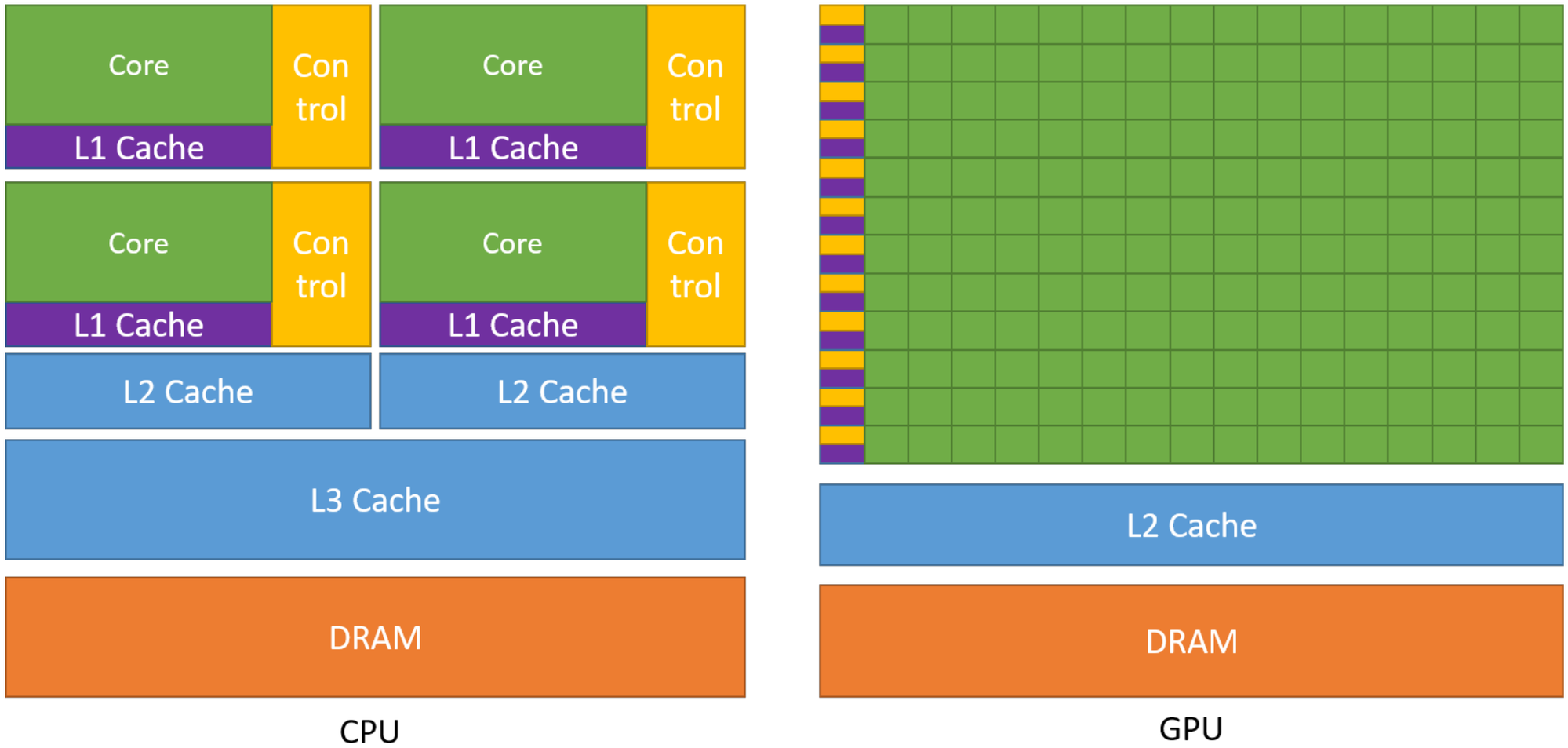



**Image credits: https://docs.nvidia.com/cuda/cuda-c-programming-guide/**

# Graphics Processing Unit (GPU)

- **Smaller, Focused Instruction Sets.** GPUs have a streamlined set of instructions optimized for specific tasks like rendering graphics and handling parallel computations.

- **Parallel Processing Power.** Designed to excel at executing the same instruction across multiple data streams simultaneously, making them ideal for tasks such as graphics rendering, machine learning, and scientific simulations.

- **Peripheral Role.** The GPU functions as a specialized hardware unit controlled by the CPU. While the CPU handles general-purpose tasks, the GPU takes over intensive parallel computations to boost overall performance.