



# Computers and Microprocessors

Lecture 34

PHYS3360/AEP3630

# Contents

- Computer architecture / experiment control
- Microprocessor organization
- Basic computer components
- Memory modes for x86 series of microprocessors

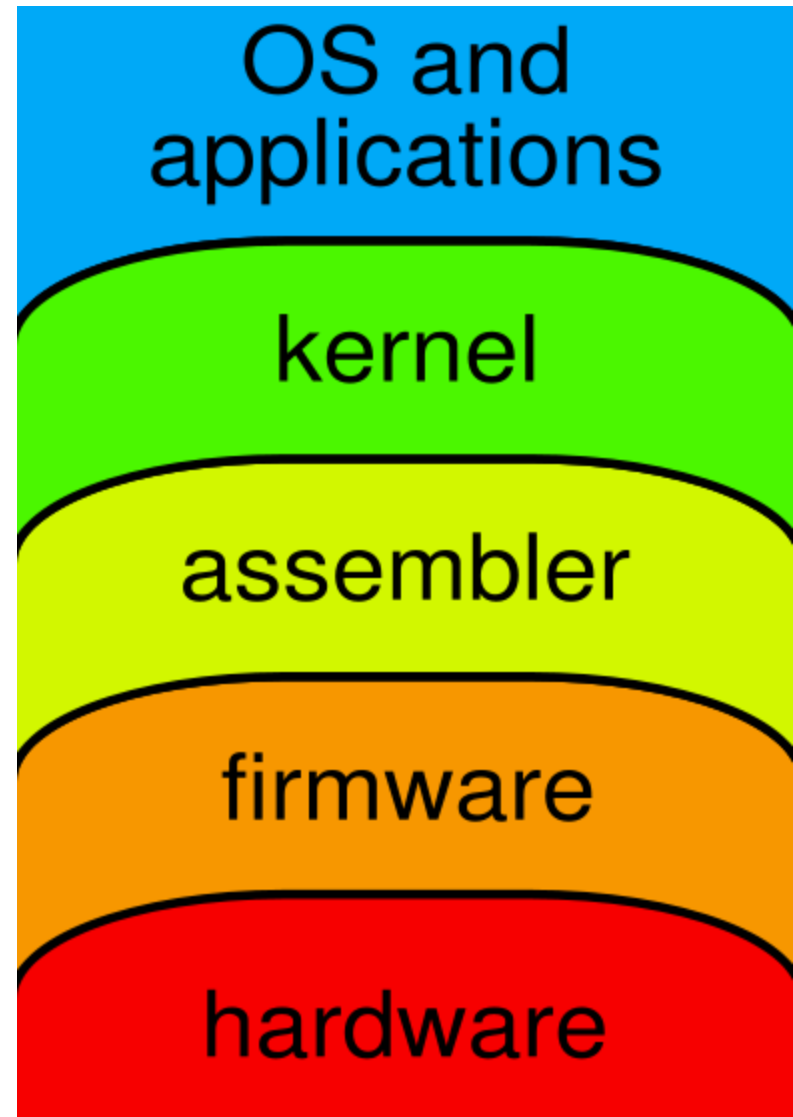
# Laboratory computers

- Many experiments utilize computers interfaced to the physical world via **transducers** (previous lecture)
- This allows **control of experiment conditions** and **data acquisition** not otherwise possible (e.g. nonlinear control of a measured parameter, various feedback/readout loops, setup procedures)
- Computers can be either **standalone** or **embedded** (sometimes also called **controllers**)

# Computer Abstraction Layers

## Basic Computer Organization

- Has several *abstraction layers* that *hide implementation details*
- **OS & applications**: a gateway for human interaction
- **kernel**: mandatory part of OS common to all software; CPU tasks, memory, I/O interrupts; drivers for various devices
- **assembler**: turns *assembly language* instr. into *opcodes* (binary stream of commands that CPU understands)

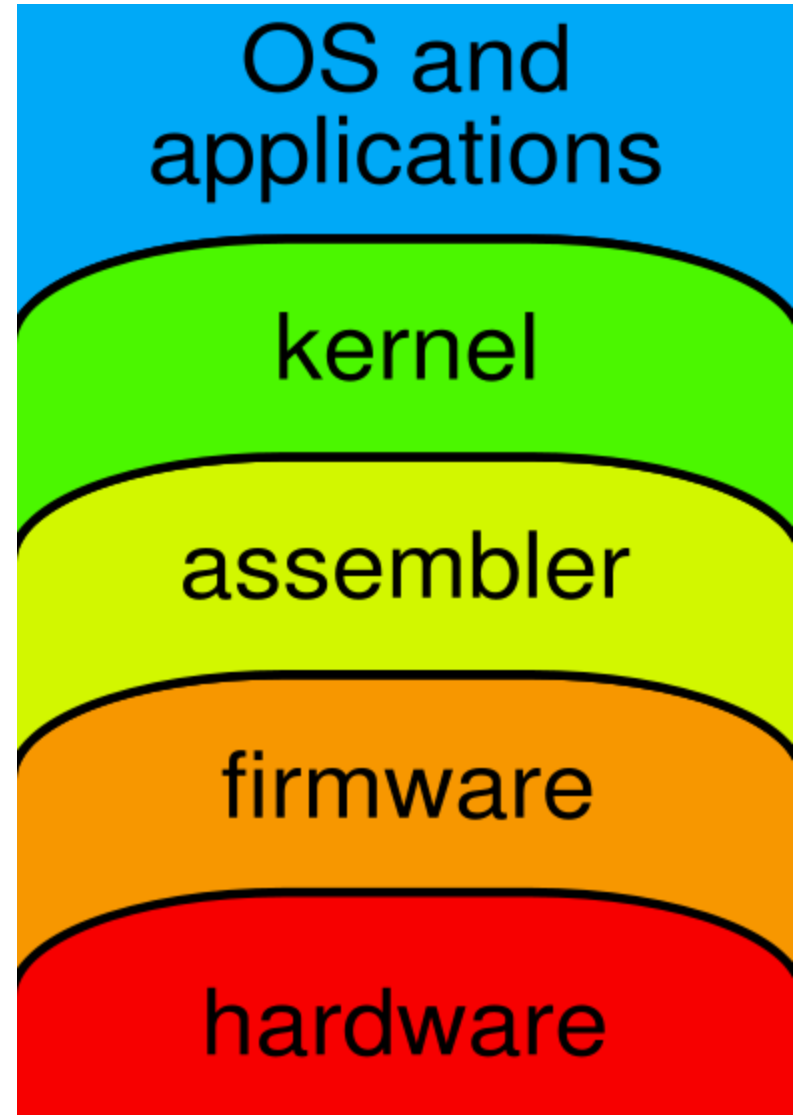


# Computer Abstraction Layers (2)

- **firmware**: a low-level program embedded into a hardware that enables its functionality; hardware specific.

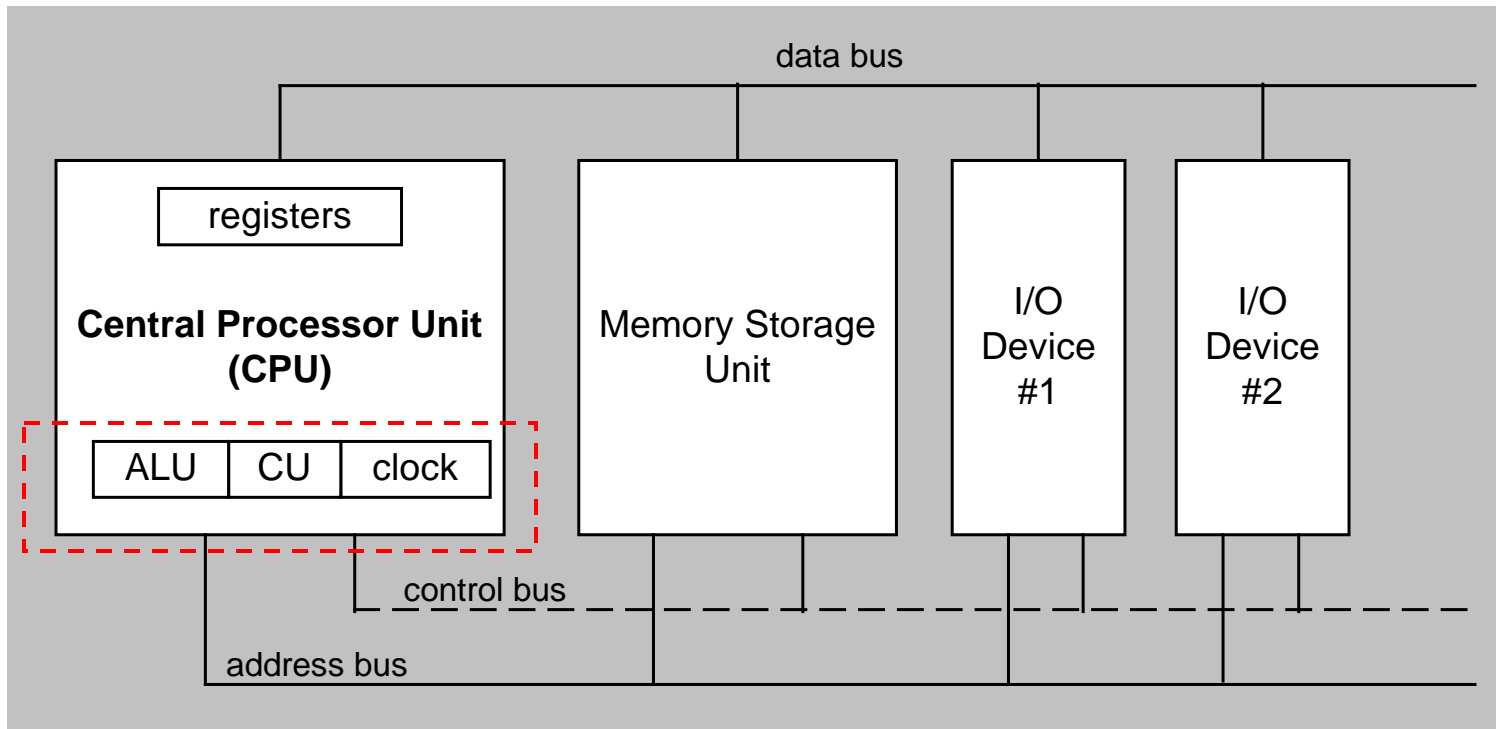
Ex: Basic Input/Output System  
Microcode

- **hardware**: microprocessors, buses, memory, input/output devices, ports, etc.



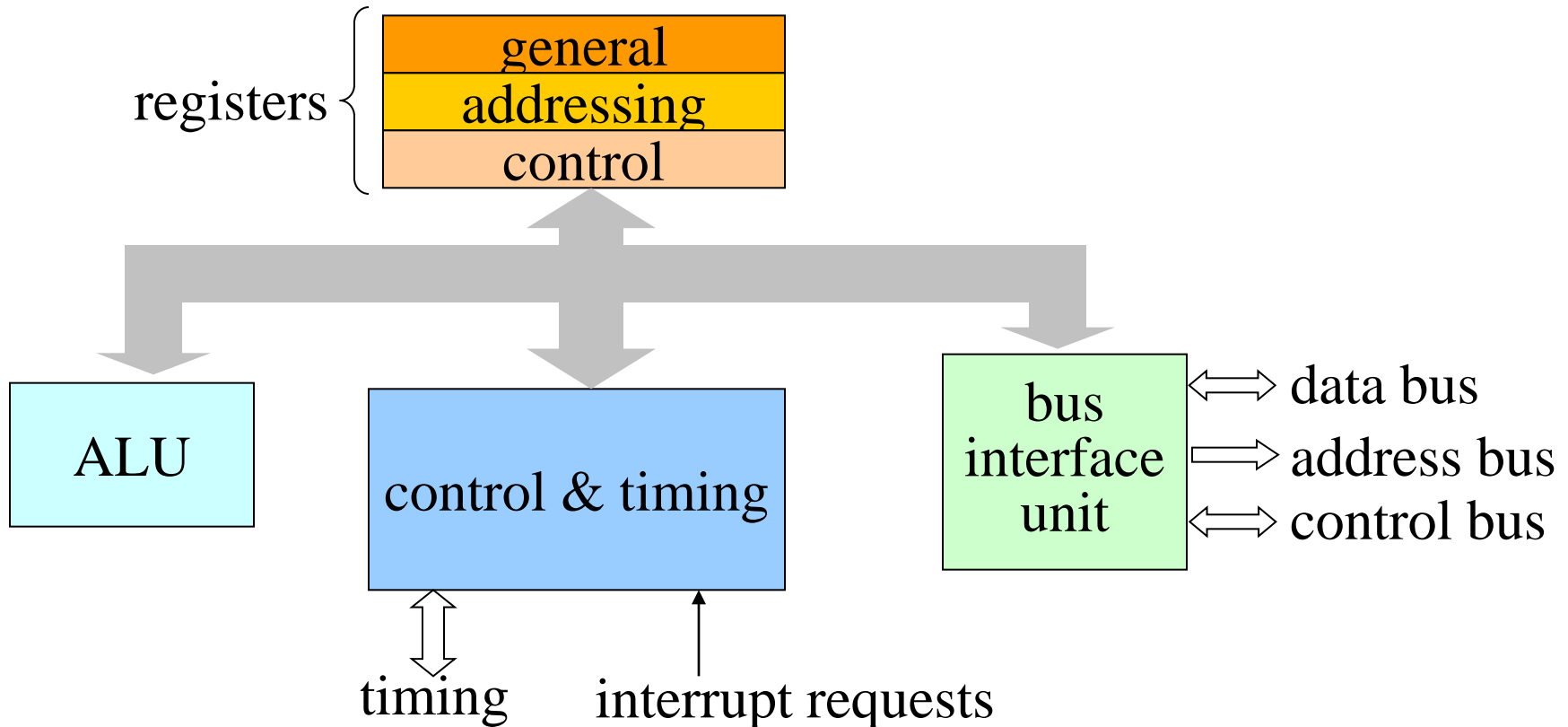
# Computer Organization

- **control unit (CU)**: controls operation timing / sequencing
- **arithmetic logic unit (ALU)**: performs all math & logic operations, receives command sequence from CU
- **clock**: synchronizes CPU operation



# Central Processing Unit (CPU)

- CPU = ALU + CU + registers
- **Registers**: arrays of D flip-flops for storage of operands, addresses, and instructions, characterized by **quick access**
- **Microprocessor**: CPU on a single chip

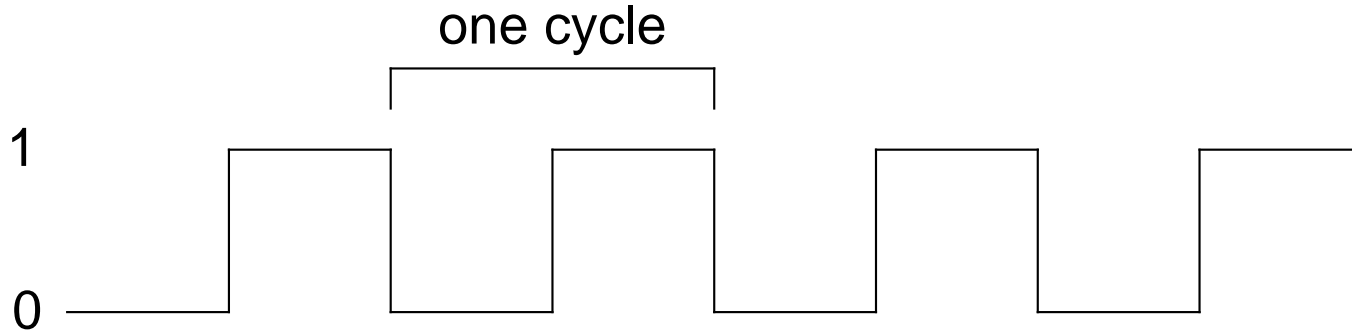


# Buses

- **Bus**: a group of wires that transfer data from one part to another (data, address, control)
- **Data bus**:
  - bi-directional (read/write)
  - 8, 16, 32-bit wide common (same as ‘word size’)
- **Address bus**:
  - specifies memory location in RAM/ROM/interface device to be accessed; monodirectional
  - address space: 16-bit wide  $\Rightarrow 2^{16} = 64 \times 2^{10} = 64\text{KB}$   
32-bit wide  $\Rightarrow 2^{32} = 4\text{GB}$
- **Control bus**: carries commands from the CPU and returns status signals from the devices

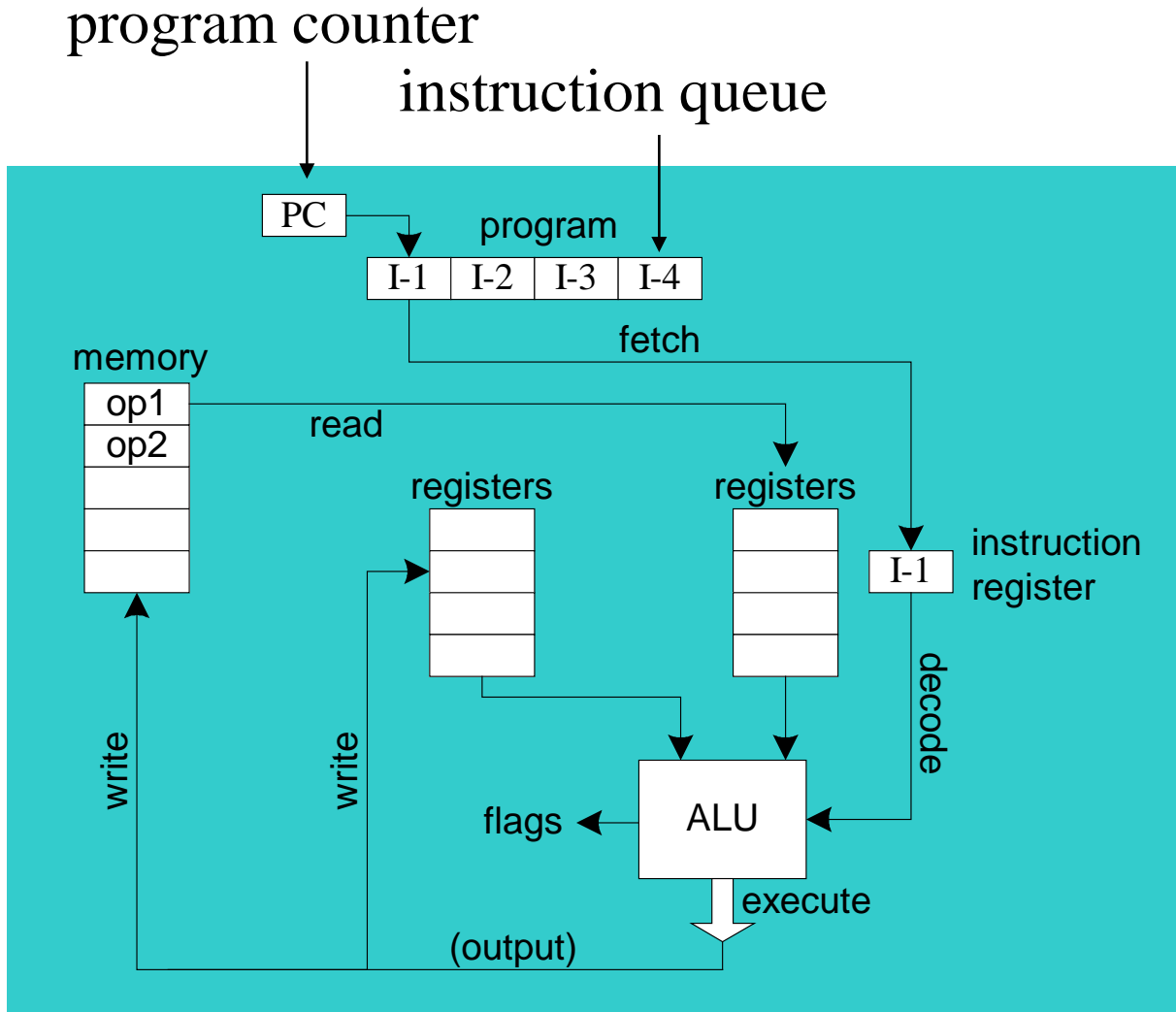


# Clock



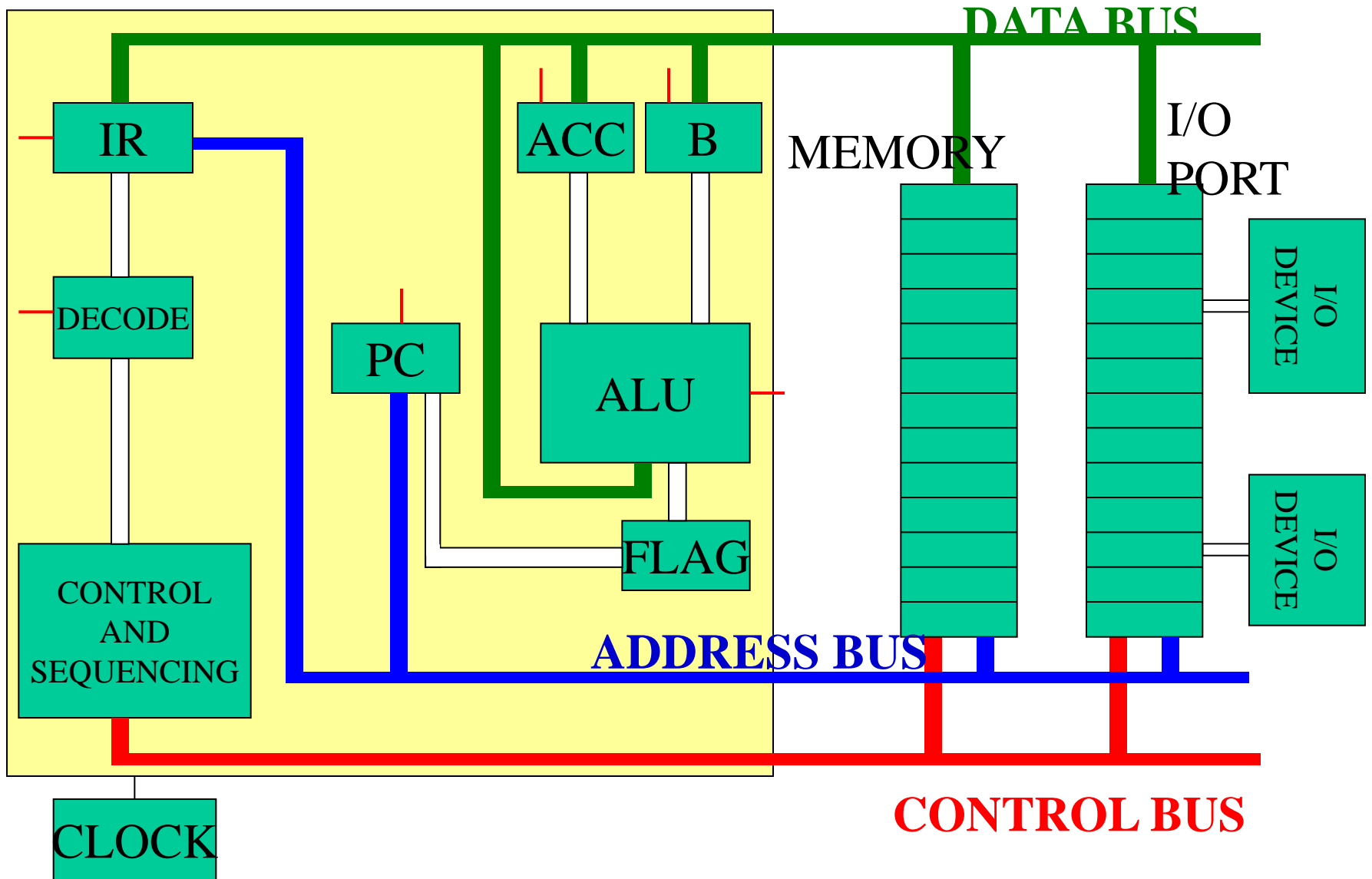
- all CPU and bus operations are synchronized to the clock
- machine cycle (clock) measures time for a single operation
- e.g. 2 GHz  $\rightarrow$  clock cycle 0.5 ns
- CPU frequency is not necessarily indicative of the execution speed; e.g. floating pt. operation of multiplication may take  $\sim 10$  to  $\sim 100$  cycles depending on the processor
- FLOPS: floating operations per second, a useful measure for (super)computers dedicated to extensive computations

# Instruction execution cycle

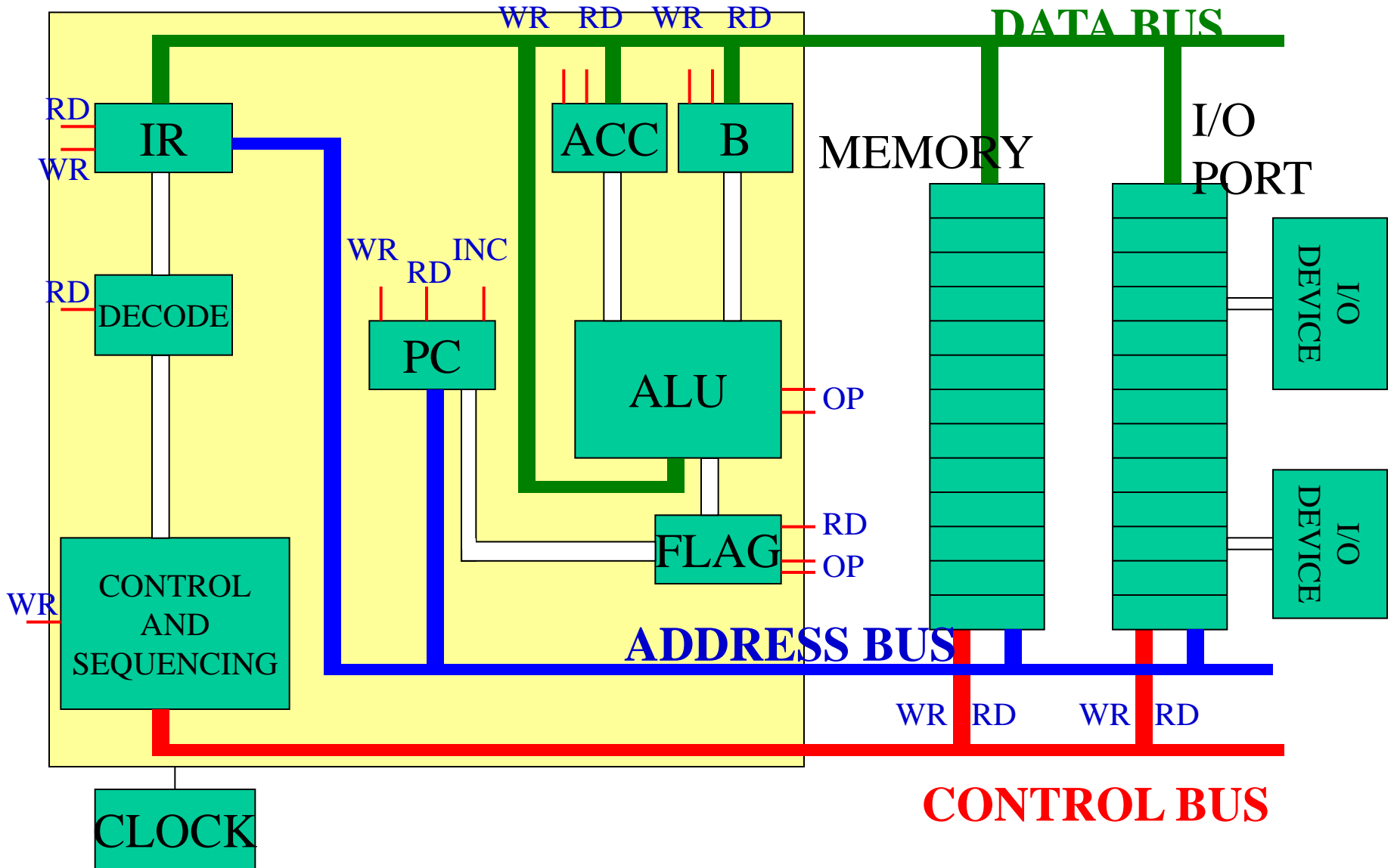


- **Fetch**
- **Decode**
- **Fetch operands**
- **Execute**
- **Store output**

# Simple microcomputer



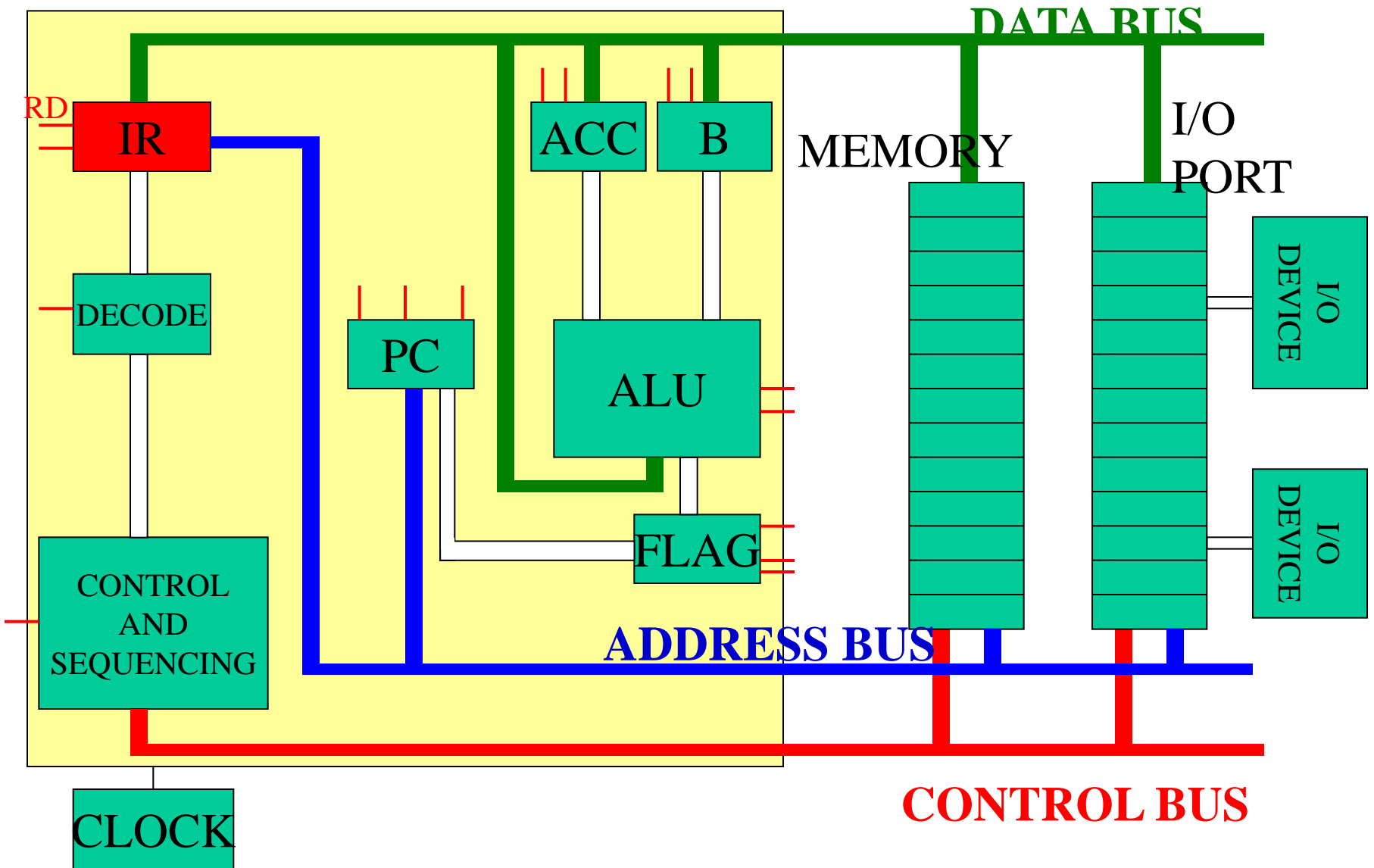
# Control signals (e.g. 20 in total)



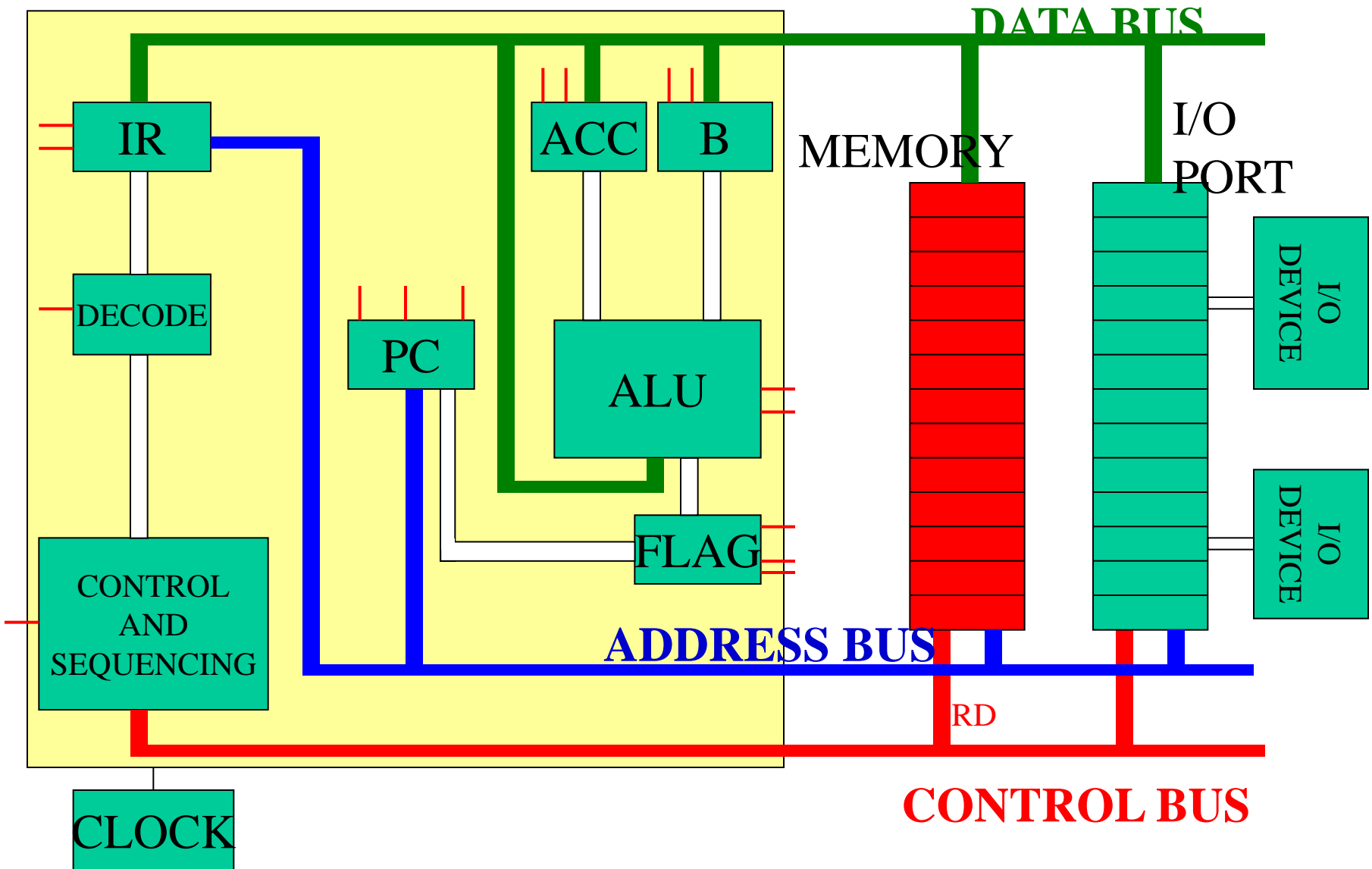
# Example of execution sequence

- LDA – assembler mnemonic for ‘load accumulator *register*’, e.g. for performing arithmetic operation, etc.

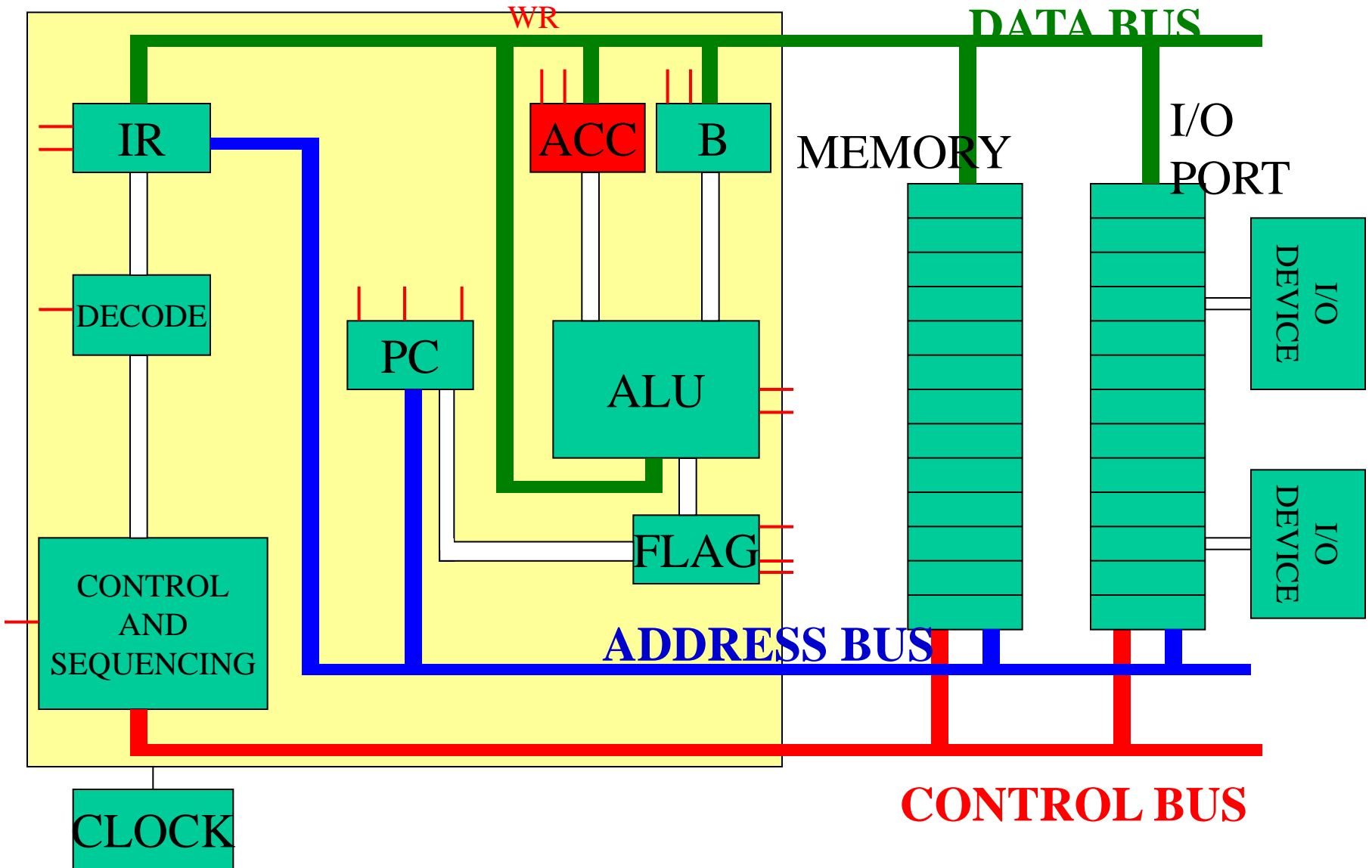
# LDA (execution cycle 1): $IR_{RD}$



# LDA (execution cycle 2): MEM<sub>RD</sub>



# LDA (execution cycle 3): $ACC_{WR}$

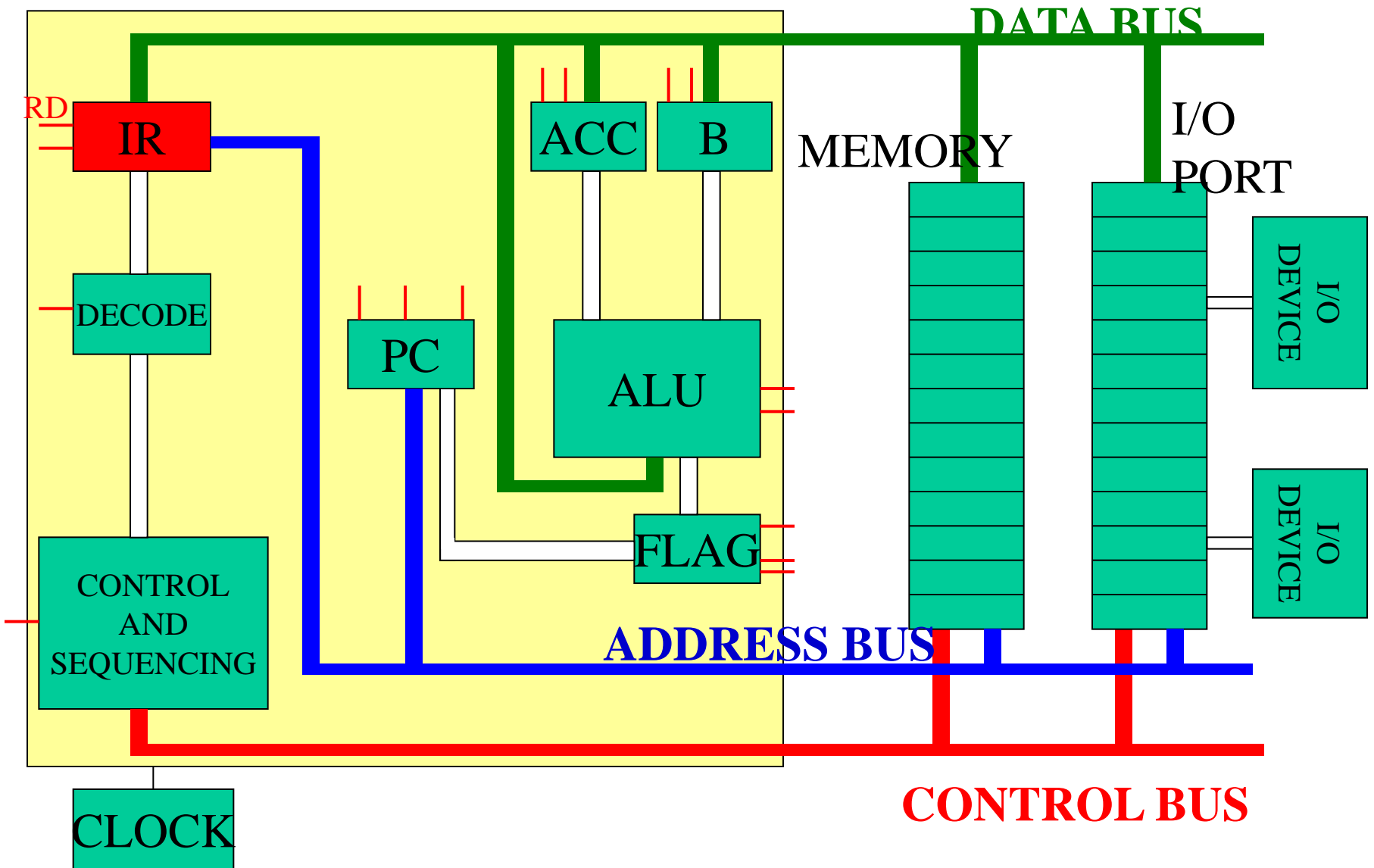




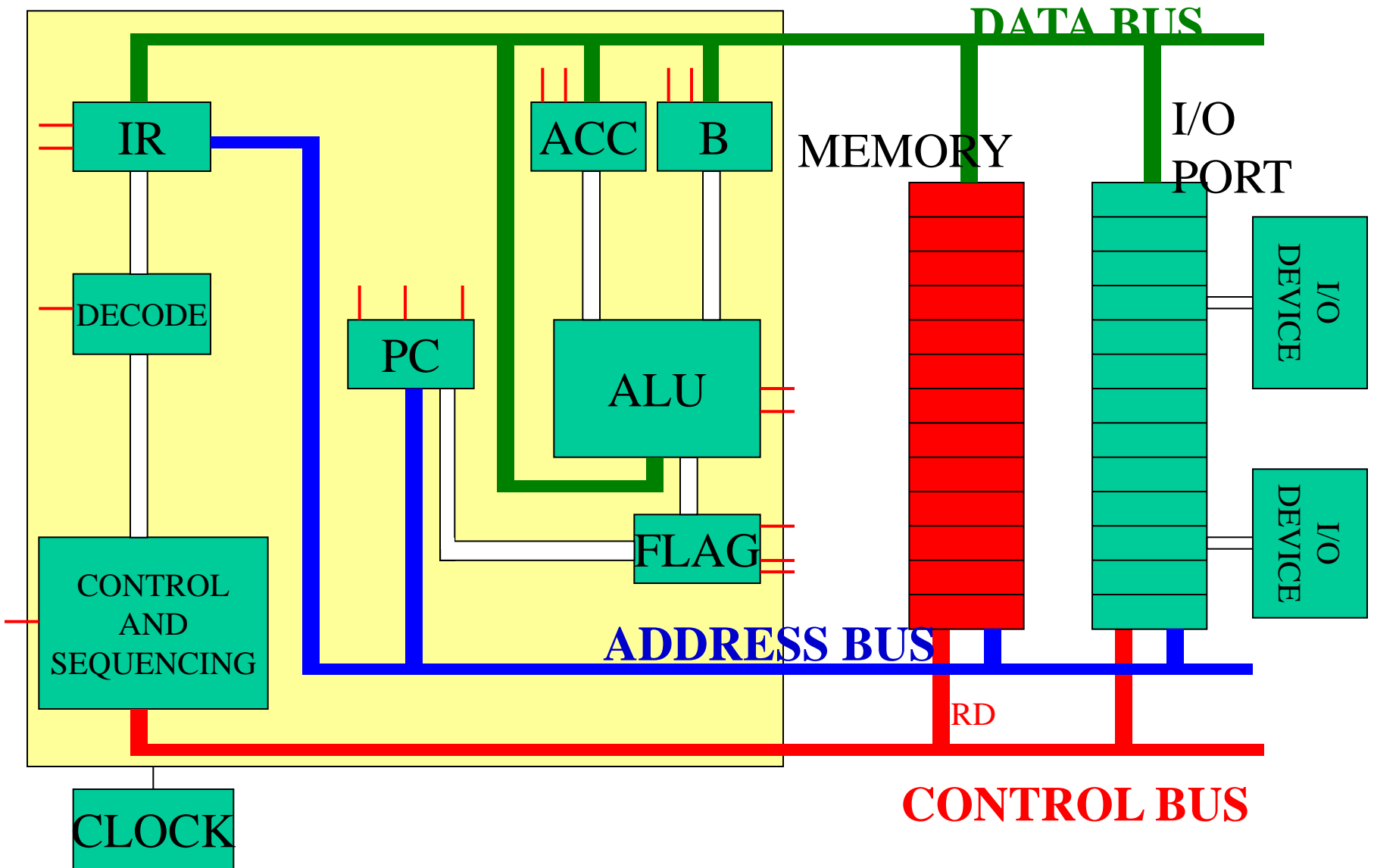
# Example of execution sequence

- ADD – adds a number (ALU function) to one already stored in ACC

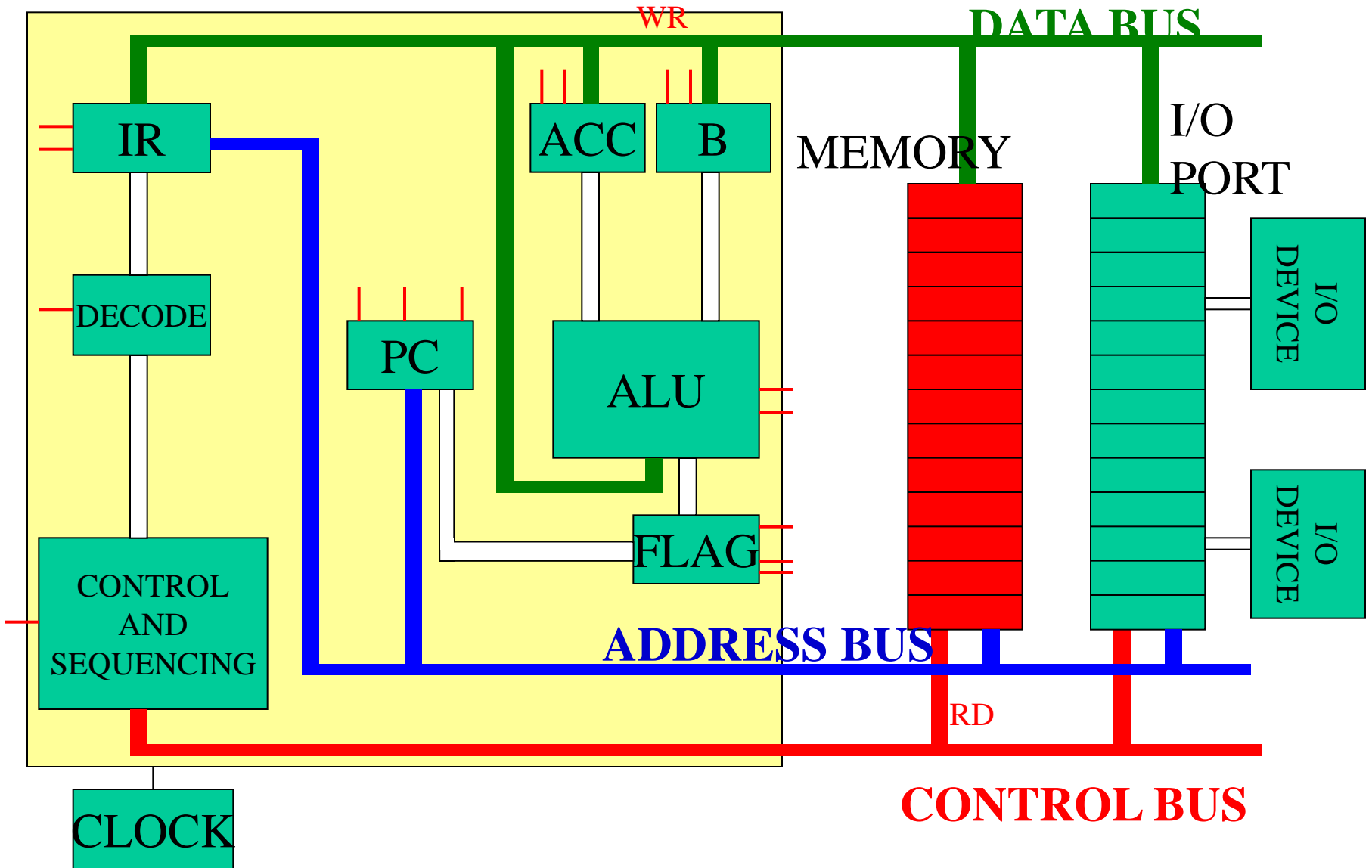
# ADD (execution cycle 1): $IR_{RD}$



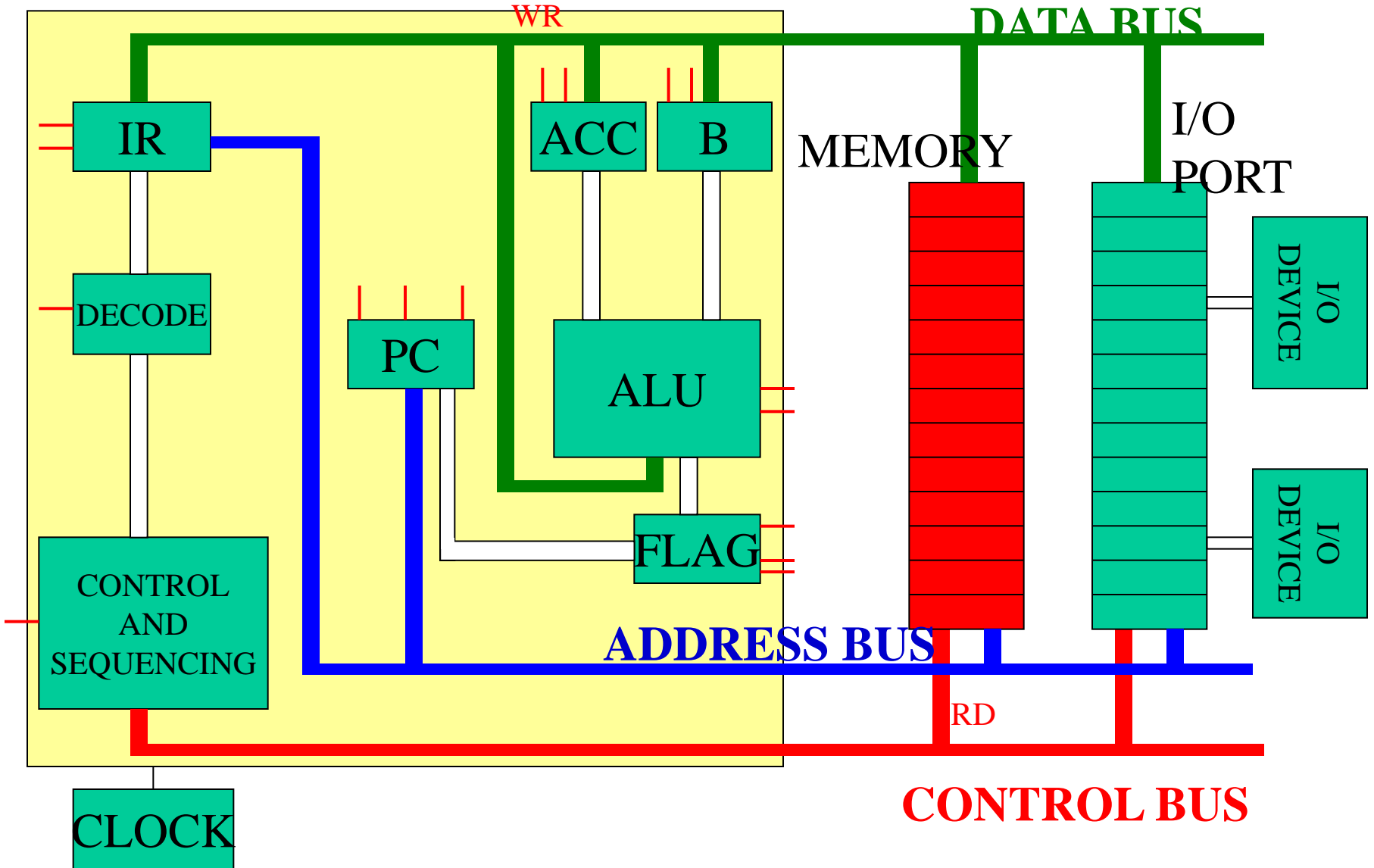
# ADD (execution cycle 2): MEM<sub>RD</sub>



# ADD (execution cycle 3): $B_{WR}$



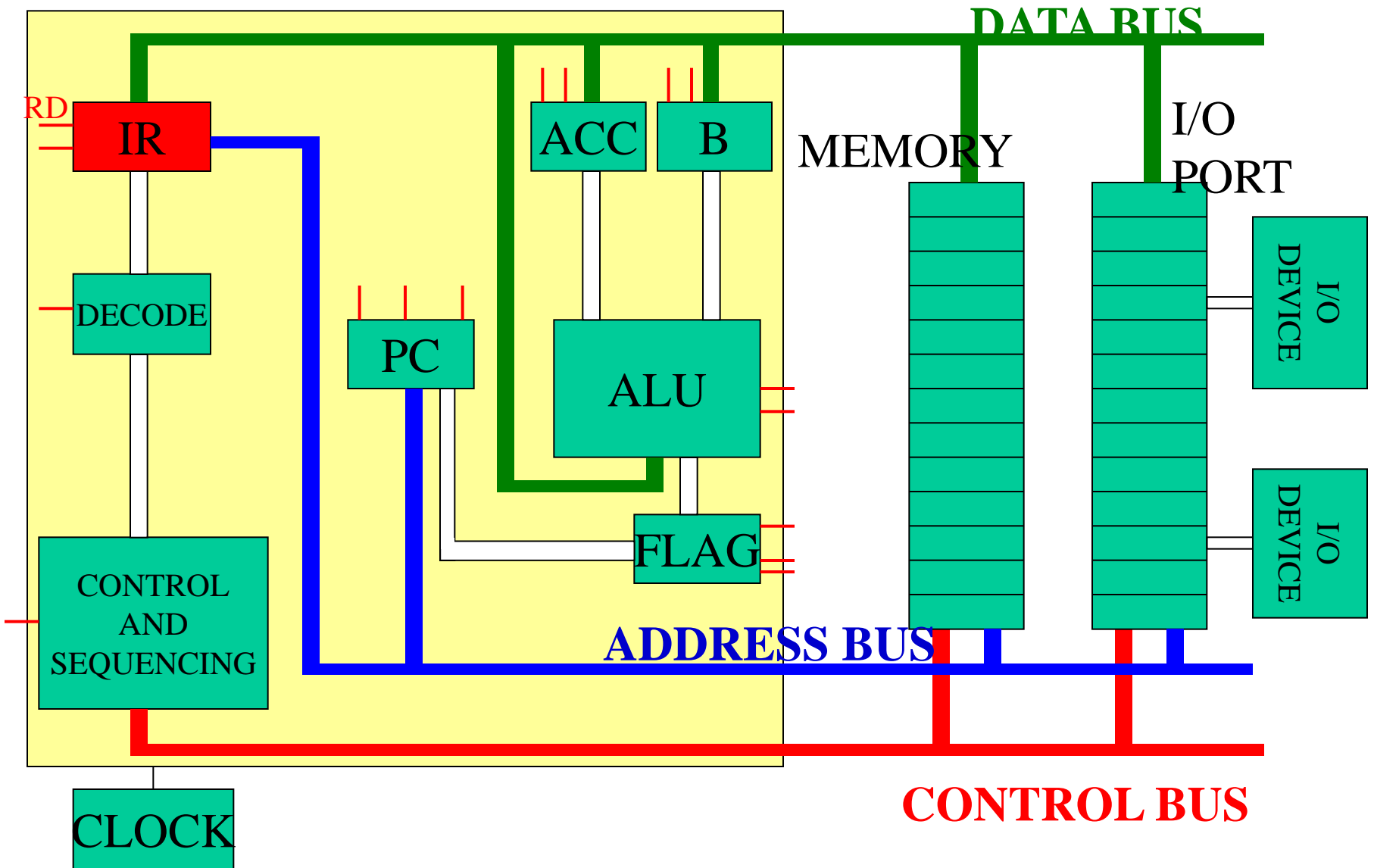
# ADD (execution cycle 4): $ALU_{10}$ , $ACC_{WR}$



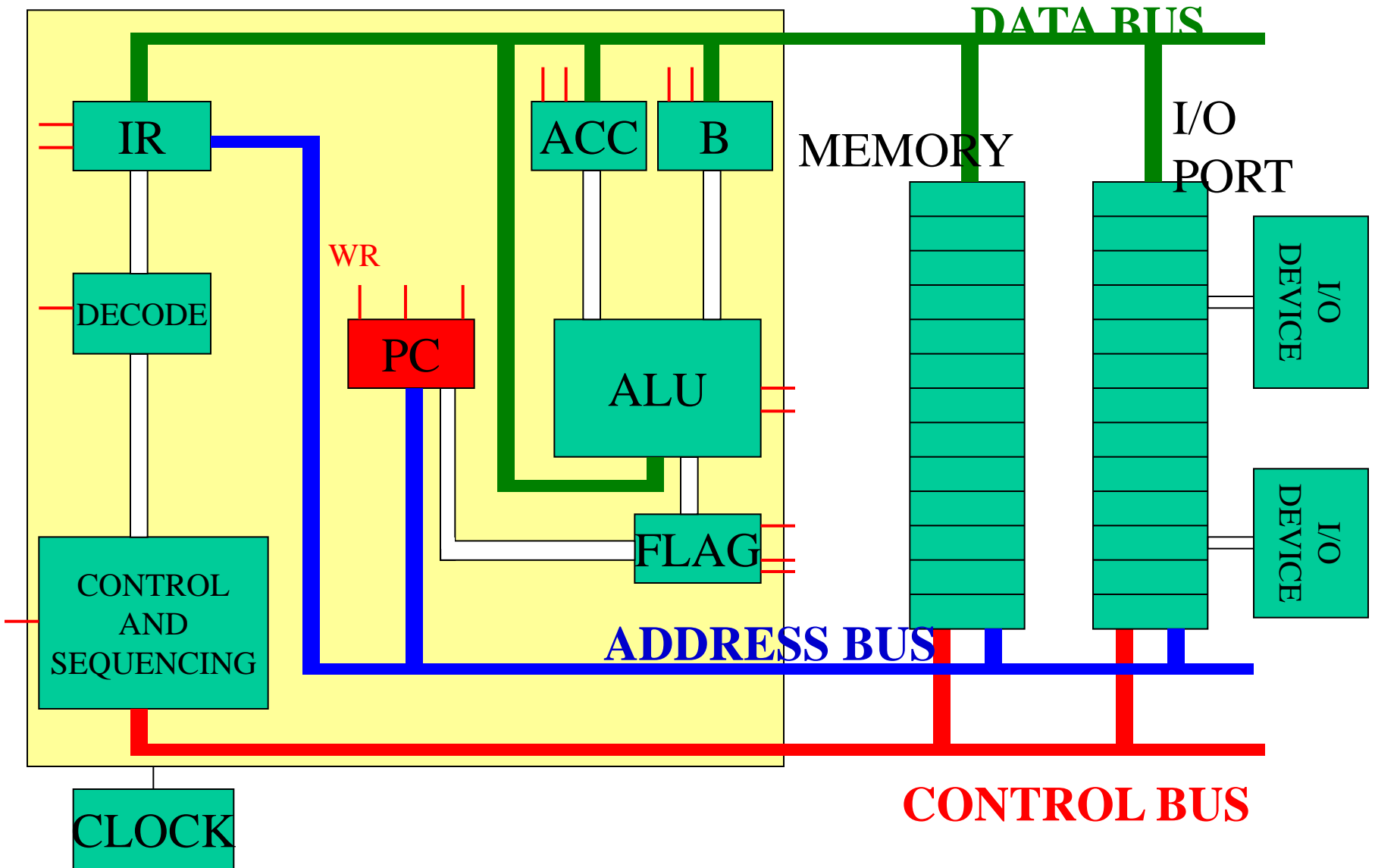
# Example of execution sequence

- JMP – jump to another command (aka GOTO)

# JMP (execution cycle 1): $IR_{RD}$



# JMP (execution cycle 2): $PC_{WR}$





# Intel Architecture 32 (IA-32)

- Instruction set for CPUs from 386 to the latest 32-bit processor, also known as x86-32
- From programmer's point of view, IA-32 has not changed all that much except for the introduction of high-performance instructions
- Now being succeeded by IA-64 and AMD-64 (but not everything needs a bleeding-edge speed, memory, etc. so many applications / *embedded controllers* are just fine with 32-bit, 16-bit or even 8-bit processors)

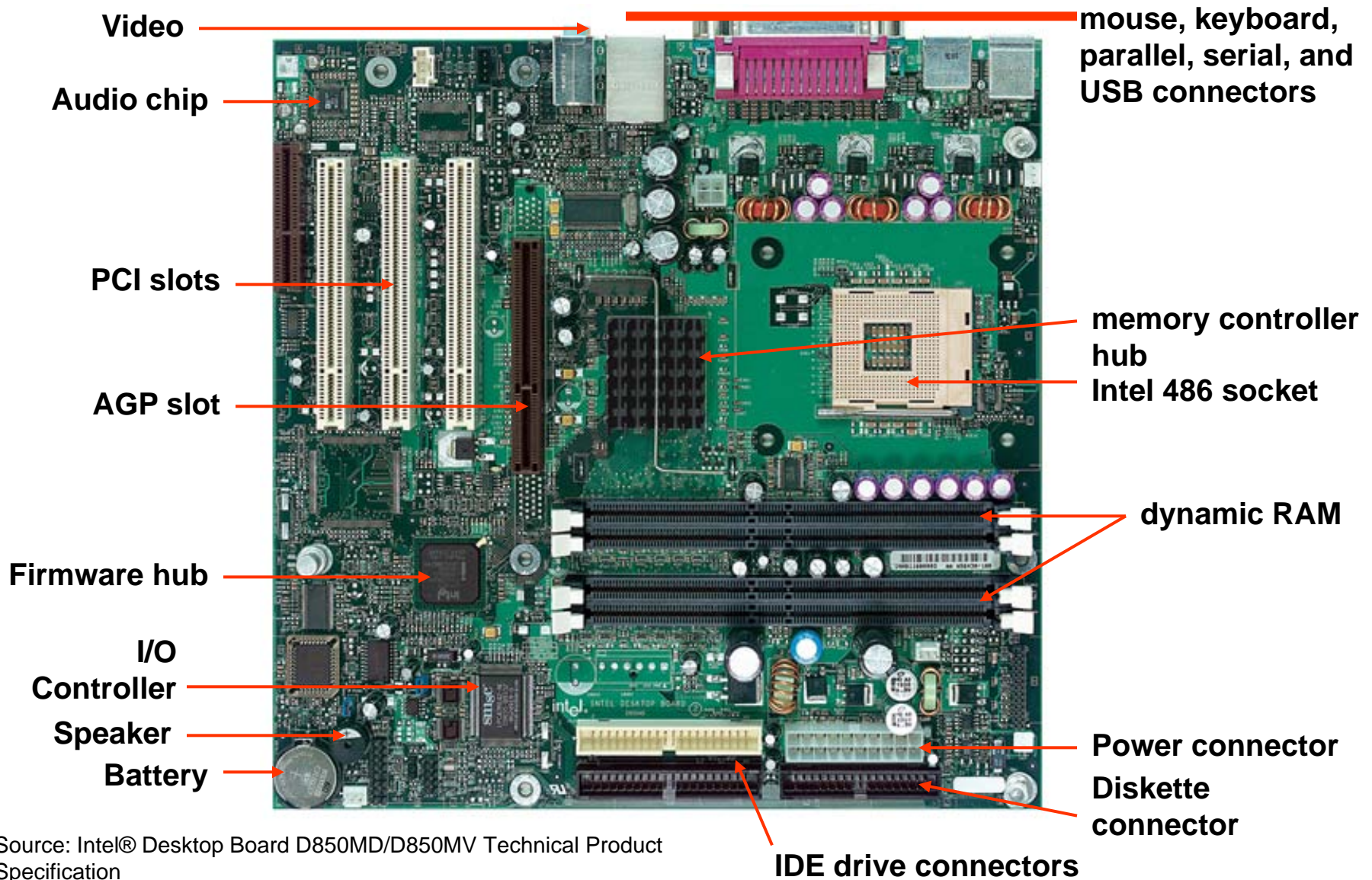
# Basic hardware constituents

- Chassis
- Controller board / Motherboard
- Peripherals: video, disk, etc. optional
- Memory
- I/O ports

# Motherboard / controller board

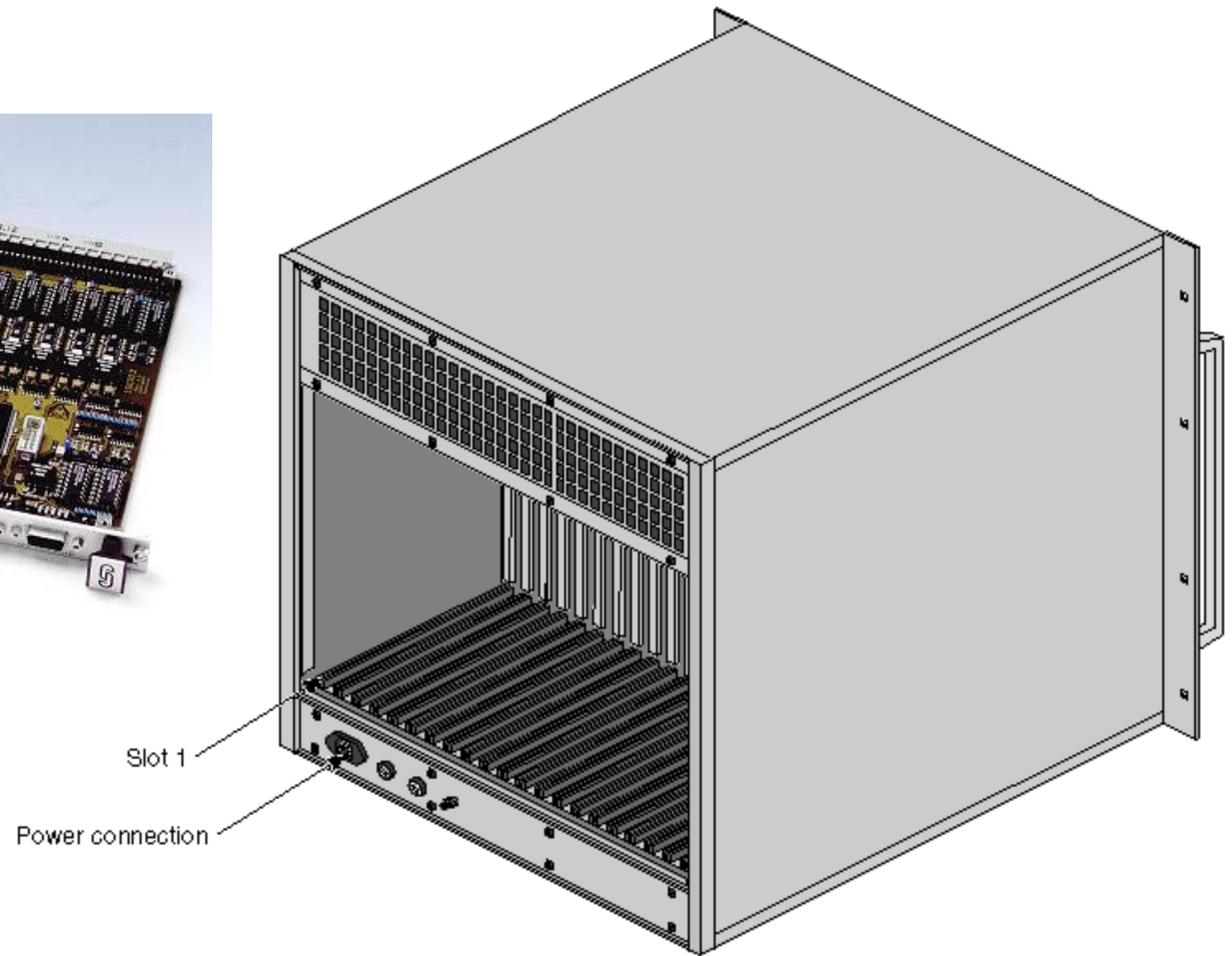
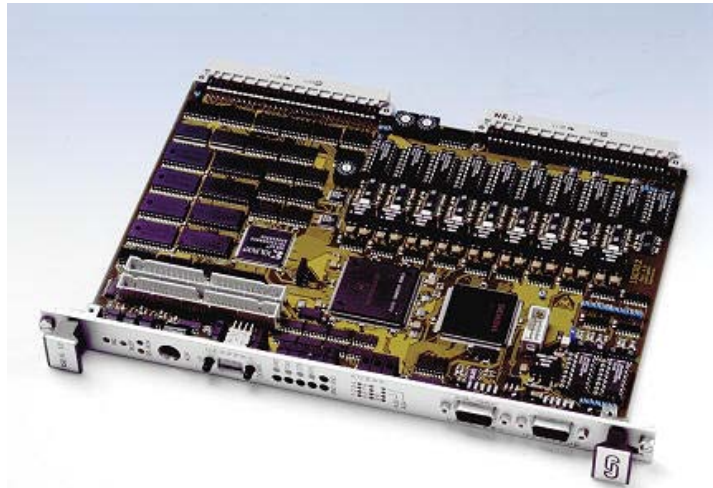
- CPU socket
- External cache memory slots
- Main memory slots
- BIOS chips
- *Video/sound synthesizer chip (optional)*
- I/O ports and bus standards:
  - Integrated Drive Electronics (IDE), parallel, serial, Universal Serial Bus, (video, keyboard, network, and mouse connectors)
  - Peripheral Component Interconnect bus connectors (expansion cards)
  - Other bus standards such as VMEbus

# Intel D850MD Motherboard



Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification

# VME Controller Board & Chassis

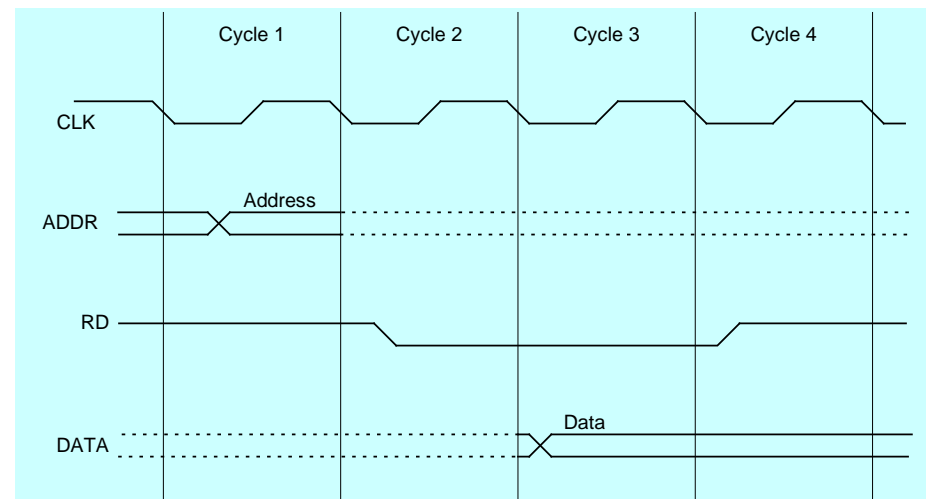


# Memory

- **Dynamic RAM (DRAM)**
  - stores each bit of data on tiny capacitors on IC
  - inexpensive, but must be refreshed constantly
- **Static RAM (SRAM)**
  - uses RS or JK latches
  - expensive; used for cache memory; no refresh required
- **Video RAM (VRAM)**
  - dual ported (multiple read/writes occurring at the same time); optimized for constant video refresh
- **CMOS RAM**
  - refreshed by a battery
  - system setup information

# Reading from memory

- Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:
  - address placed on address bus
  - Read Line (RD) set low
  - CPU waits one cycle for memory to respond
  - Read Line (RD) goes to 1, indicating that the data is on the data bus



# Cache memory

- High-speed expensive static RAM both inside and outside the CPU.
  - Level-1 cache: inside the CPU
  - Level-2 cache: outside the CPU
- Cache hit: when data to be read is already in cache memory
- Cache miss: when data to be read is not in cache memory
- Main design problem: minimize cache misses at a reasonable cost (compulsory, capacity, conflict)



# Memory access: real mode

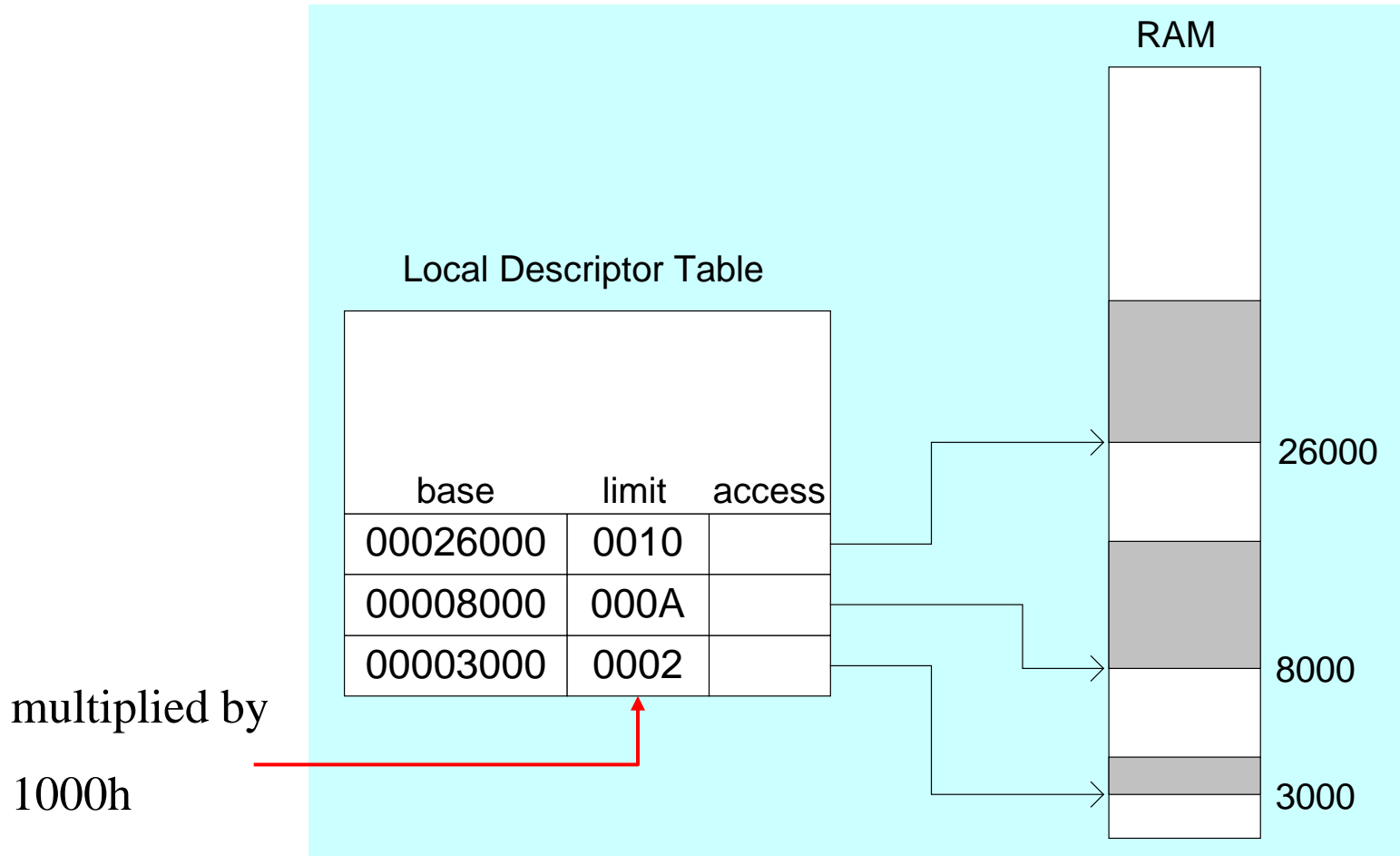
- 1 MB RAM maximum addressable (20-bit address)
- Application programs can access any area of memory
- Early  $\mu$ -processors; single tasking; supported by MS-DOS operating system

# Memory access: protected mode

- 4 GB addressable RAM (32-bit address)
  - (00000000 to FFFFFFFF<sub>H</sub>)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking; supported by Linux & Windows

# Multi-segment model

- Each program has a local descriptor table (LDT)
  - holds descriptor for each segment used by the program



# Labs this week

- Learn about CPU operations by inspecting a simple assembler program execution
- Control USB device with digital inputs/outputs (C++ code with National Instruments driver for Windows XP/Vista)
- LTspice experiment: encryption/decryption of digital information using a pseudo-random number generator