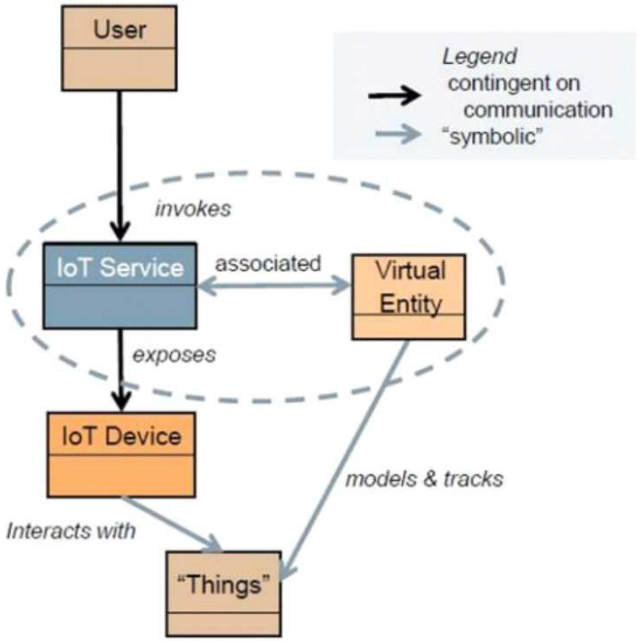
EIOT

CPU (examples of 8 and 16-bit CPUs)

Aleksander Pruszkowski

Instytut Telekomunikacji Politechniki Warszawskiej



This lecture

- Basic info on CPUs and MCUs
- Structure of microprocessor-based system
- Examples of MCUs

Basic info on CPUs and MCUs

Basic info on CPUs and MCUs

- Microprocessor (CPU) vs. microcontroller (MCU)
  - a microprocessor (CPU) does not contain memory or peripherals; it cannot work on its own
    - you need to interface to it non-volatile memory, data memory, a clock, etc.
  - a microcontroller contains different memories (both non-volatile and data memory) and a whole bunch of peripherals
    - but adding memory is an issue – extending the capacity of data memory (e.g., by using external memory to emulate data memory) needs special handling in software
      - the emulation approach is limited in that you can't use the memory for some purposes, e.g., to hold the stack; also, access to such memory is slower
  - currently, the distinction between CPU and MCU is blurred
    - many microprocessors features some minimal collection of peripherals, say, UART/USB interfaces

Basic info on CPUs and MCUs

- CPU, the heart of a system
  - can be described by
    - word size: how many bits can be processed in parallel
    - how much memory can it „see” (actually, address)
  - A typical CPU also contains
    - control unit: responsible for interpreting (executing) instructions from the CPU’s instruction set
    - program counter, PC (also called instruction pointer, IP): a register addressing the next instruction to be executed
    - arithmetic-logic unit, ALU: responsible for processing data bits (e.g., addition)
    - CPU (internal) registers
      - general purpose registers
      - control registers
      - status register
    - memory and peripherals control circuitry

Basic info on CPUs and MCUs

- CPU, the heart of the system, cont.
  - code and data storage architectures
    - Harvard — data and code stored in different, separate memories
    - von Neumann — same memory can store both data and code
  - instruction set and its complexity
    - CISC – complex instruction set computer, advanced addressing modes
    - RISC – reduced instruction set computer, simple addressing modes

CISC

```
mov ax, cs:[bp+bx*4+al]
```

RISC

```
mov   R0, 4
mul   R1, R0
add   R2, R1
add   R2, R3
ld    R3, [R2]
```

Basic info on CPUs and MCUs

- CPU, the heart of the system, cont.
  - registers
    - temporary, fast storage for intermediate results
    - number of registers
      - few registers: only special-purpose registers (e.g., accumulator) and a fast access memory region
      - many general-purpose registers

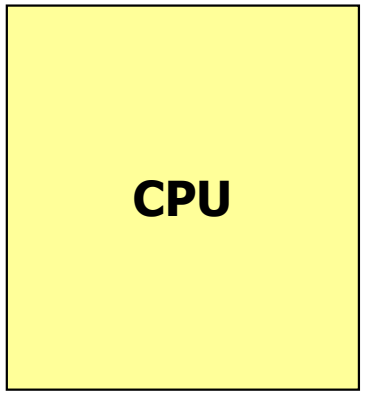
        AX,BX,CX,DX, ... - x86 architecture

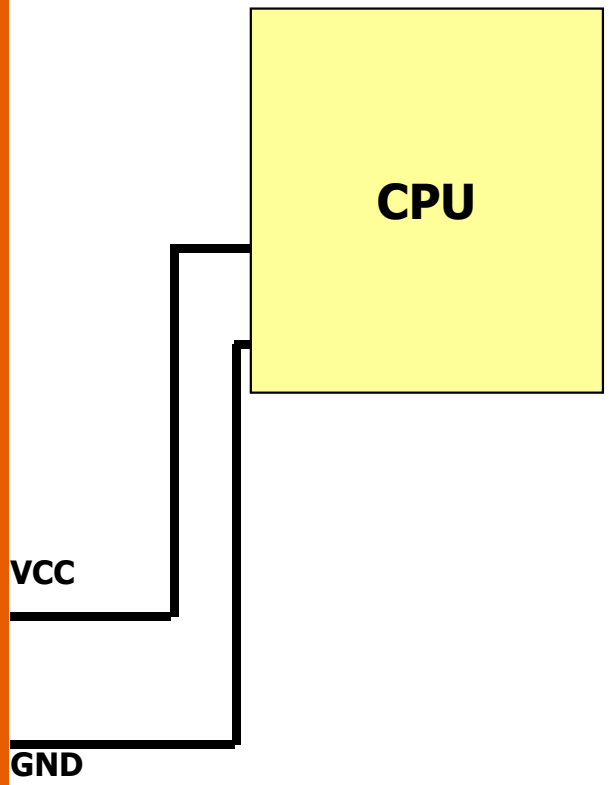
        R0...R15 – ARM architecture
  - stack
    - stack implementation
      - the stack shares RAM memory with other data
      - ”hardware” stack in a separate, special-purpose stack buffer (usually very small)
    - what is stack used for
      - return address (where to return when a function execution is completed)
      - arguments passed when a function is invoked
      - local variables

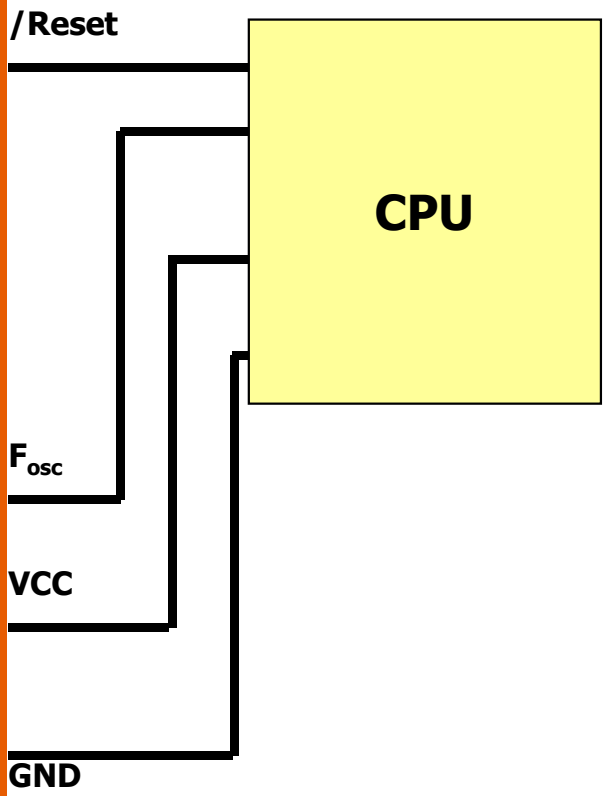
Basic info on CPUs and MCUs

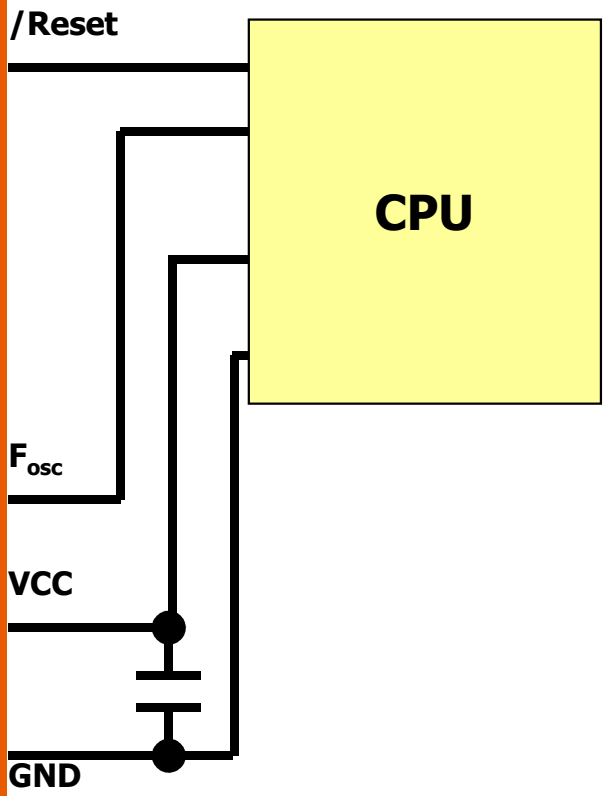
- CPU, the heart of the system, cont.
  - embedded peripherals
    - memory management and caching
      - MMU, memory management unit: paging, mapping virtual memory addresses to physical memory addresses
      - protected mode: assigning „privilege levels" to processes and granting access (or not) to portions of memory based on them
      - caching: speeding up access by holding frequently accessed data in fast, small cache memory
    - interrupt system
      - makes it possible to immediately react (handle) to a change in the state of peripheral devices
      - interrupts have priorities
      - aligned with event-driven programming (events can be triggered by hardware interrupts)
    - (MCU) embedded communication interfaces
      - RS232/UART, SPI, I2C, SATA, USB (host/device), ETH, WIFI, Radio/ISM, ...
    - (MCU) ADC and DAC converters
    - ... and other peripherals

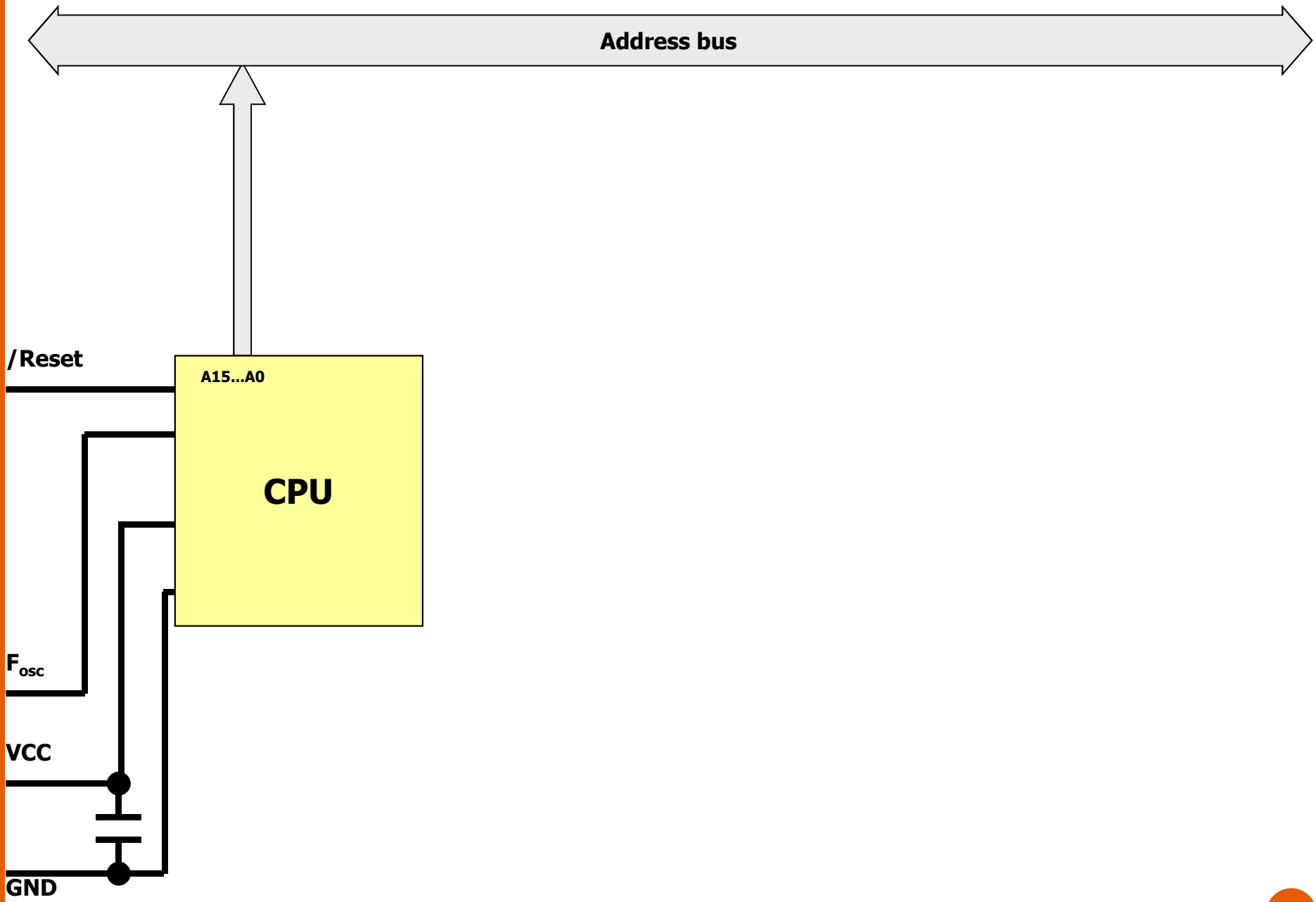
Structure
of microprocessor-based system

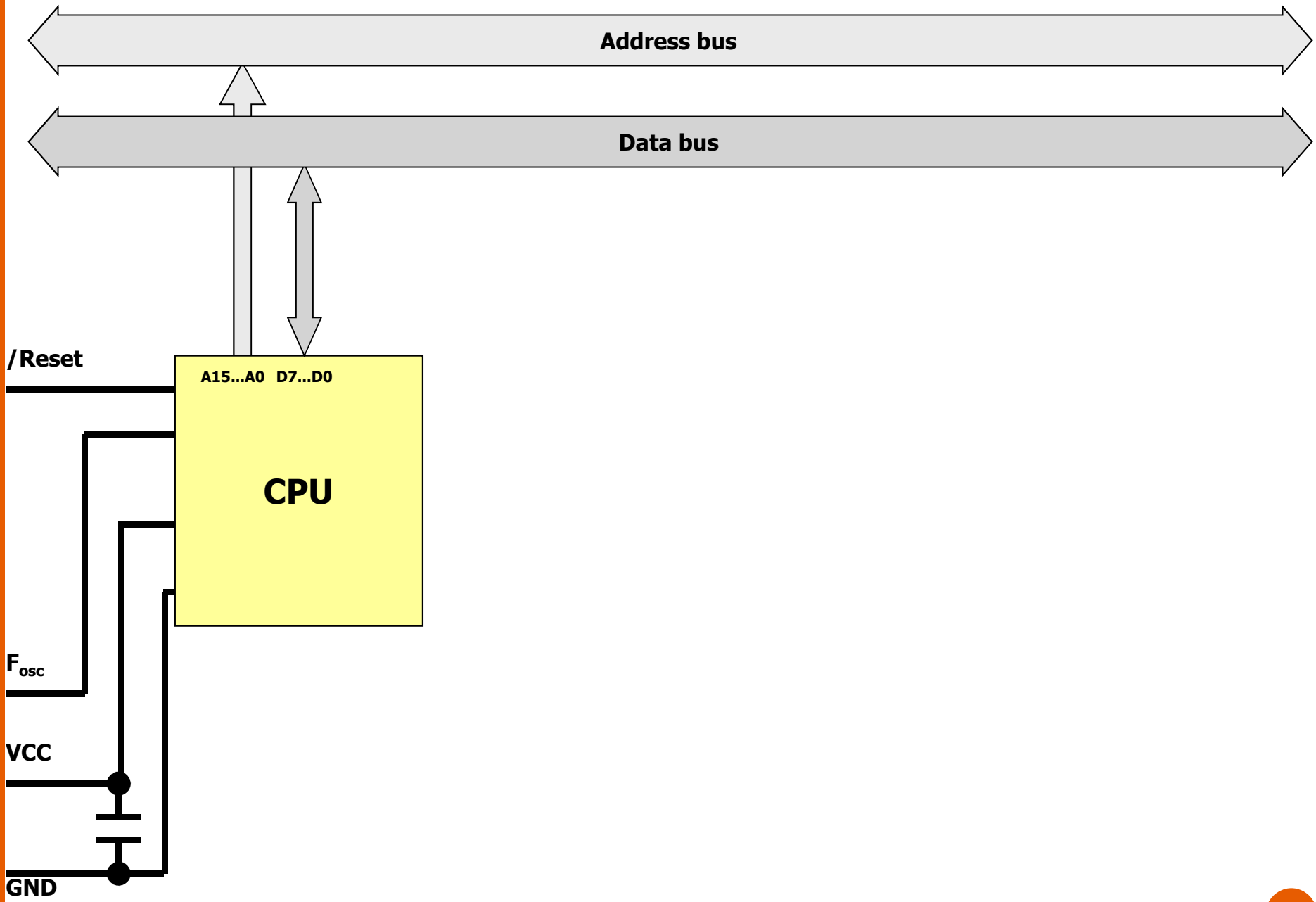

CPU

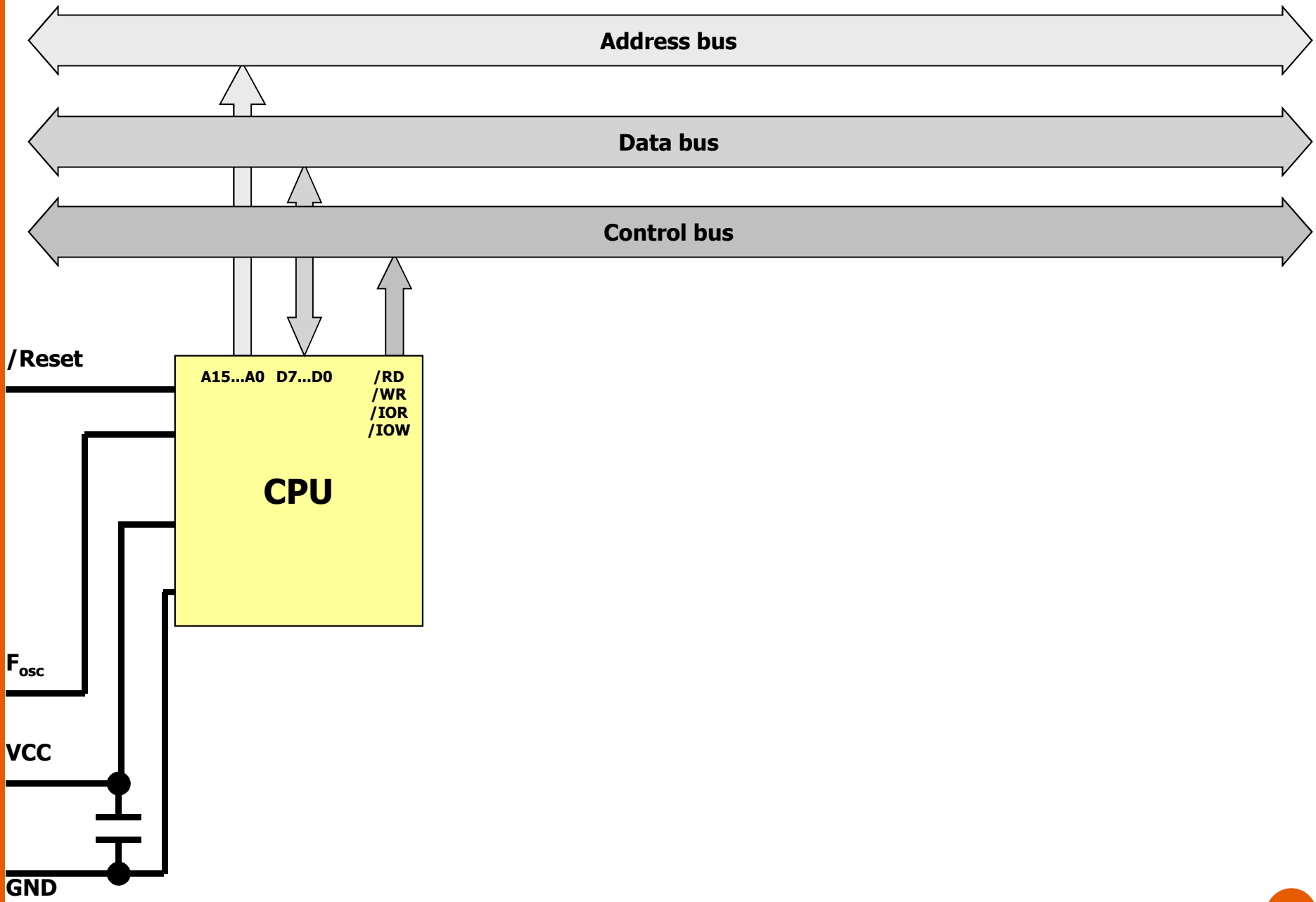

CPU
VCC
GND

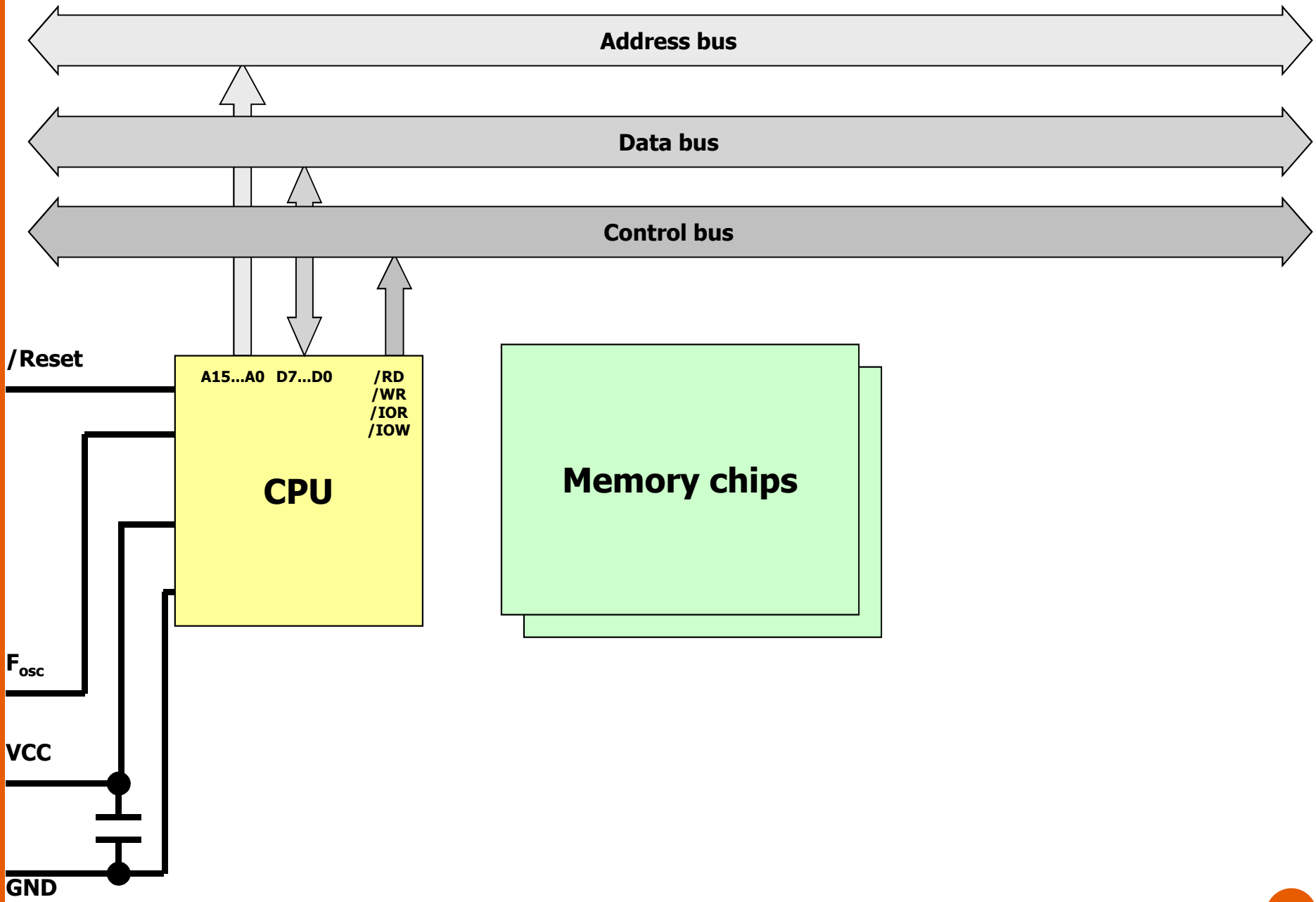

/Reset
CPU
F_{osc}
VCC
GND

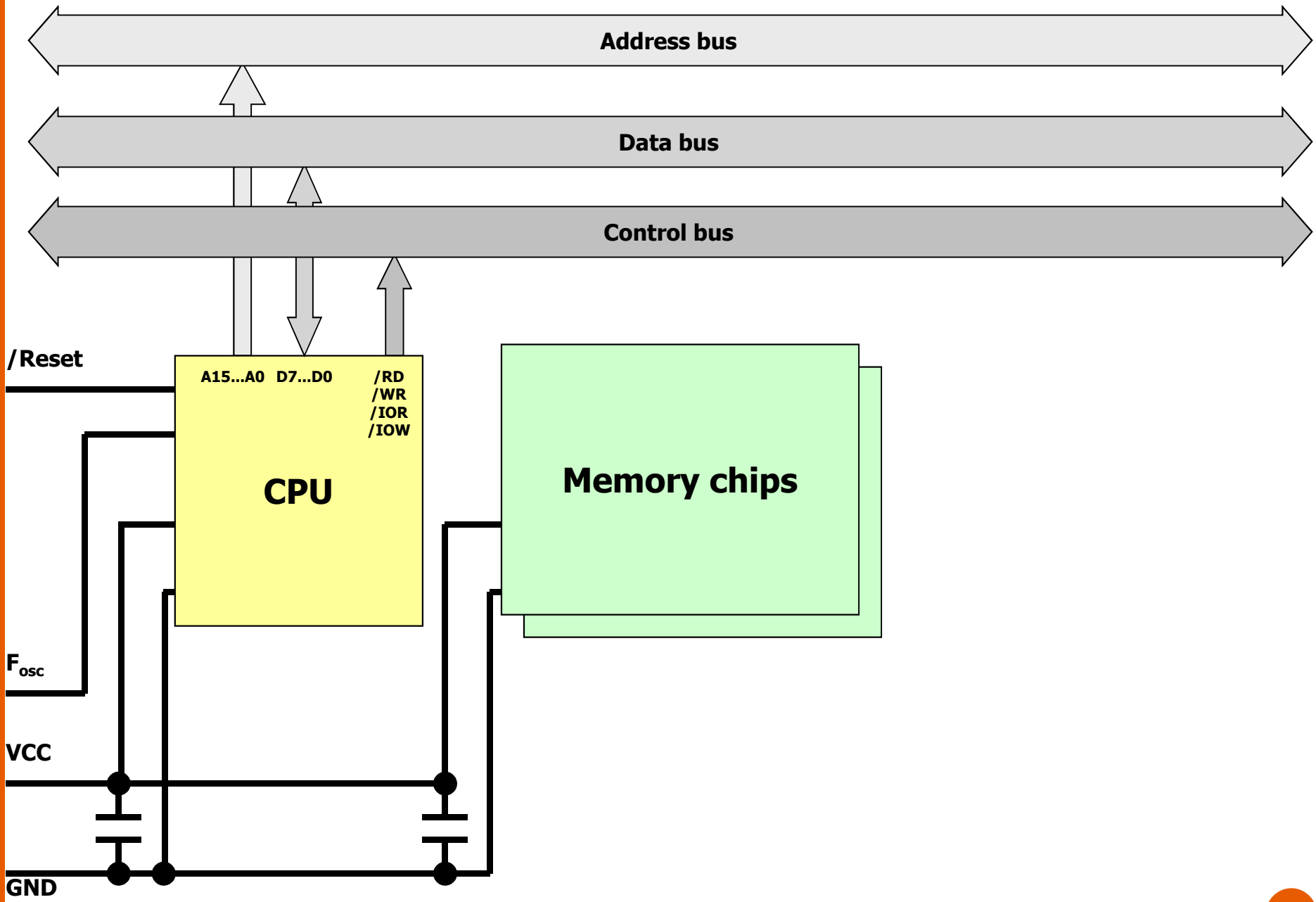

/Reset
CPU
F_{osc}
VCC
GND

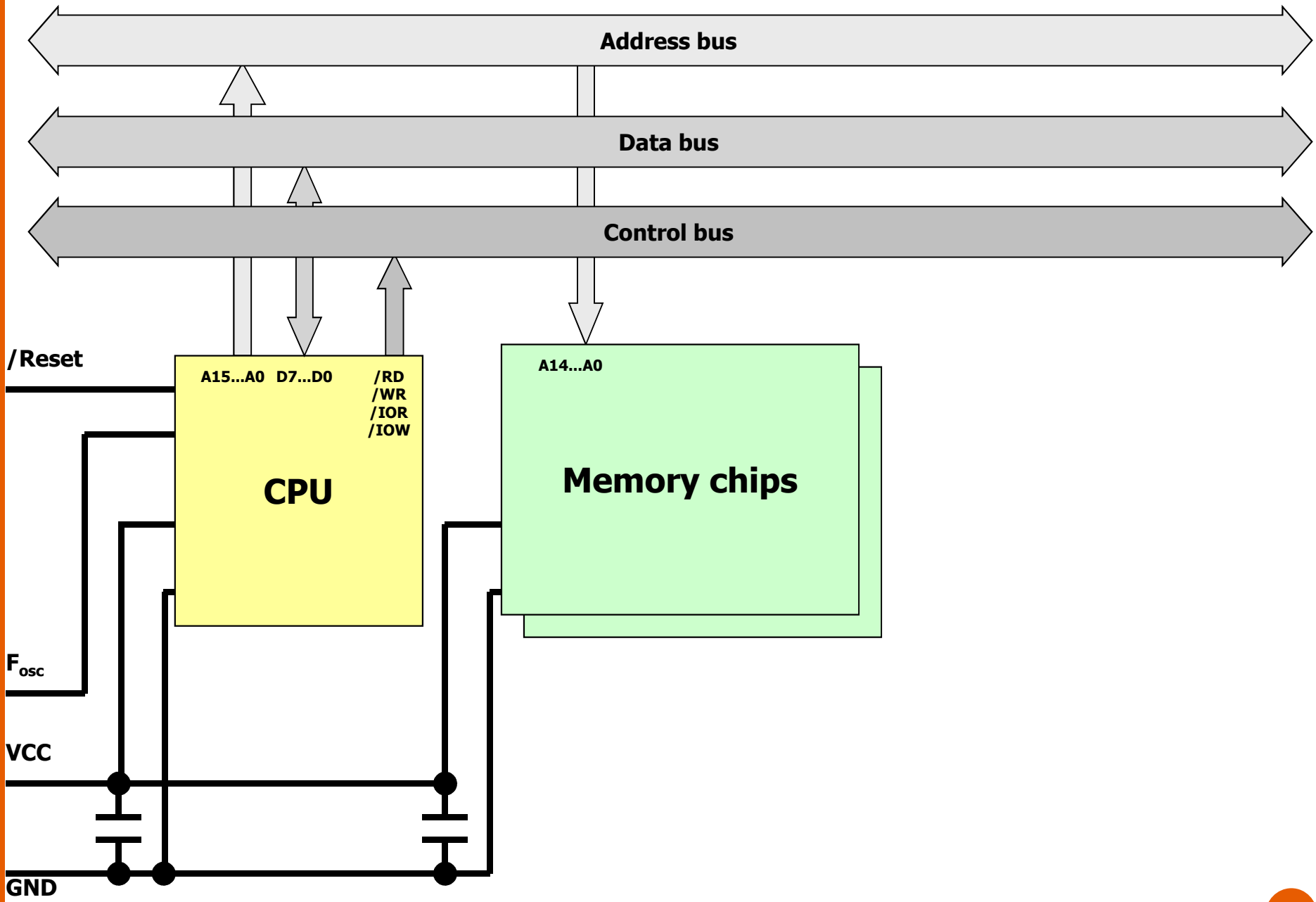

Address bus
/Reset
A15...A0
CPU
F_{osc}
VCC
GND

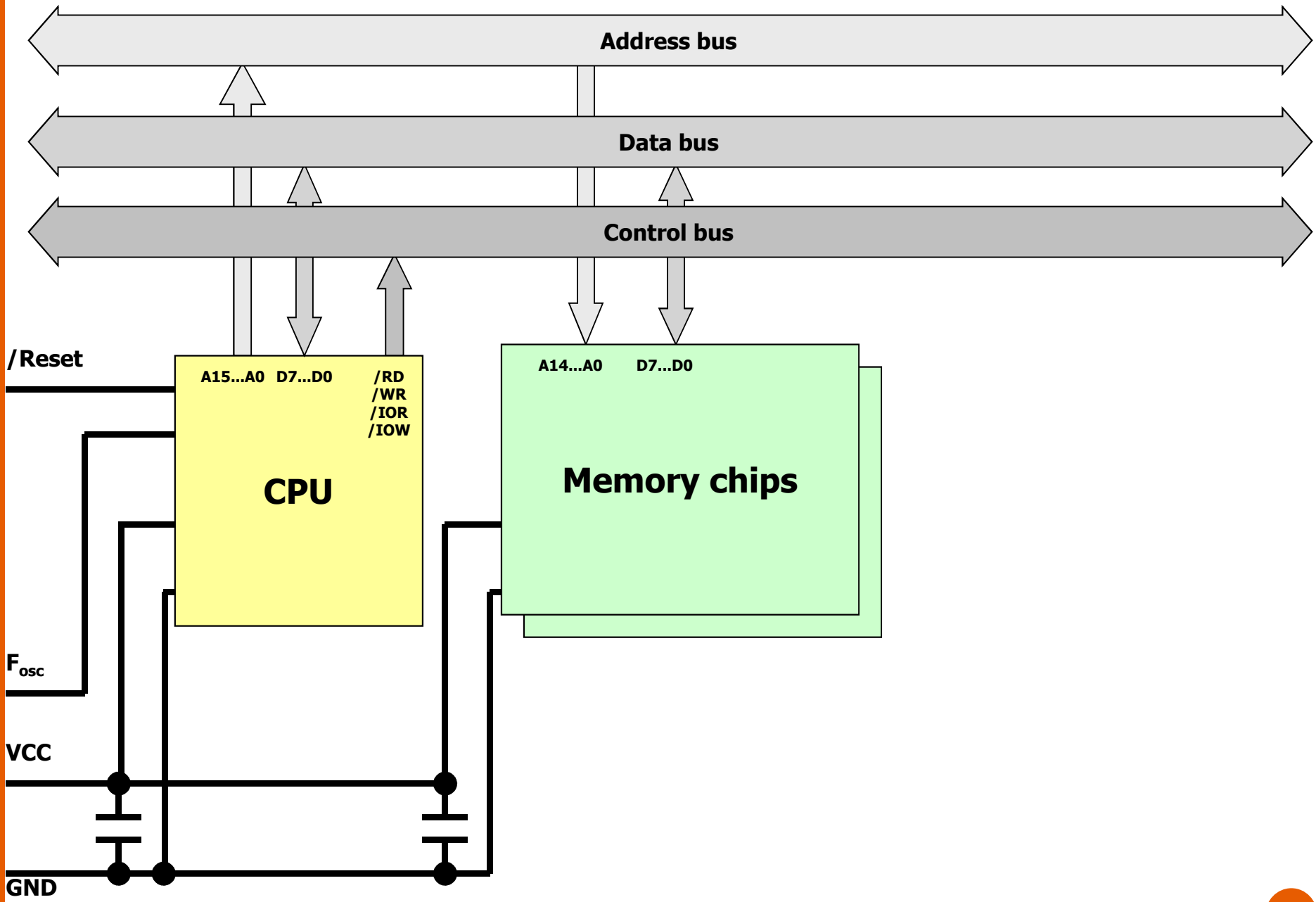

Address bus
Data bus
/Reset
A15...A0 D7...D0
CPU
Fosc
VCC
GND

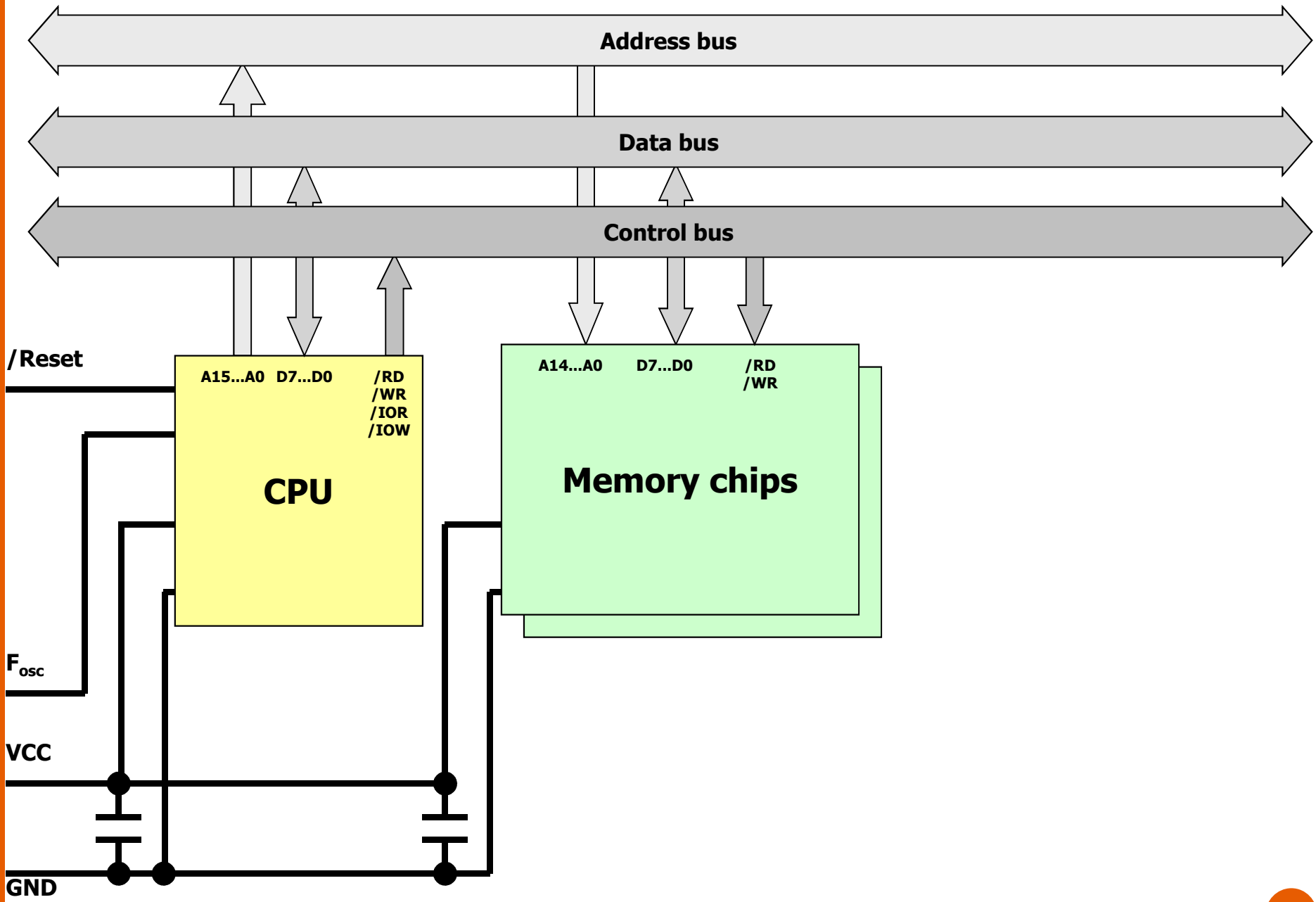

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
F_{osc}
VCC
GND

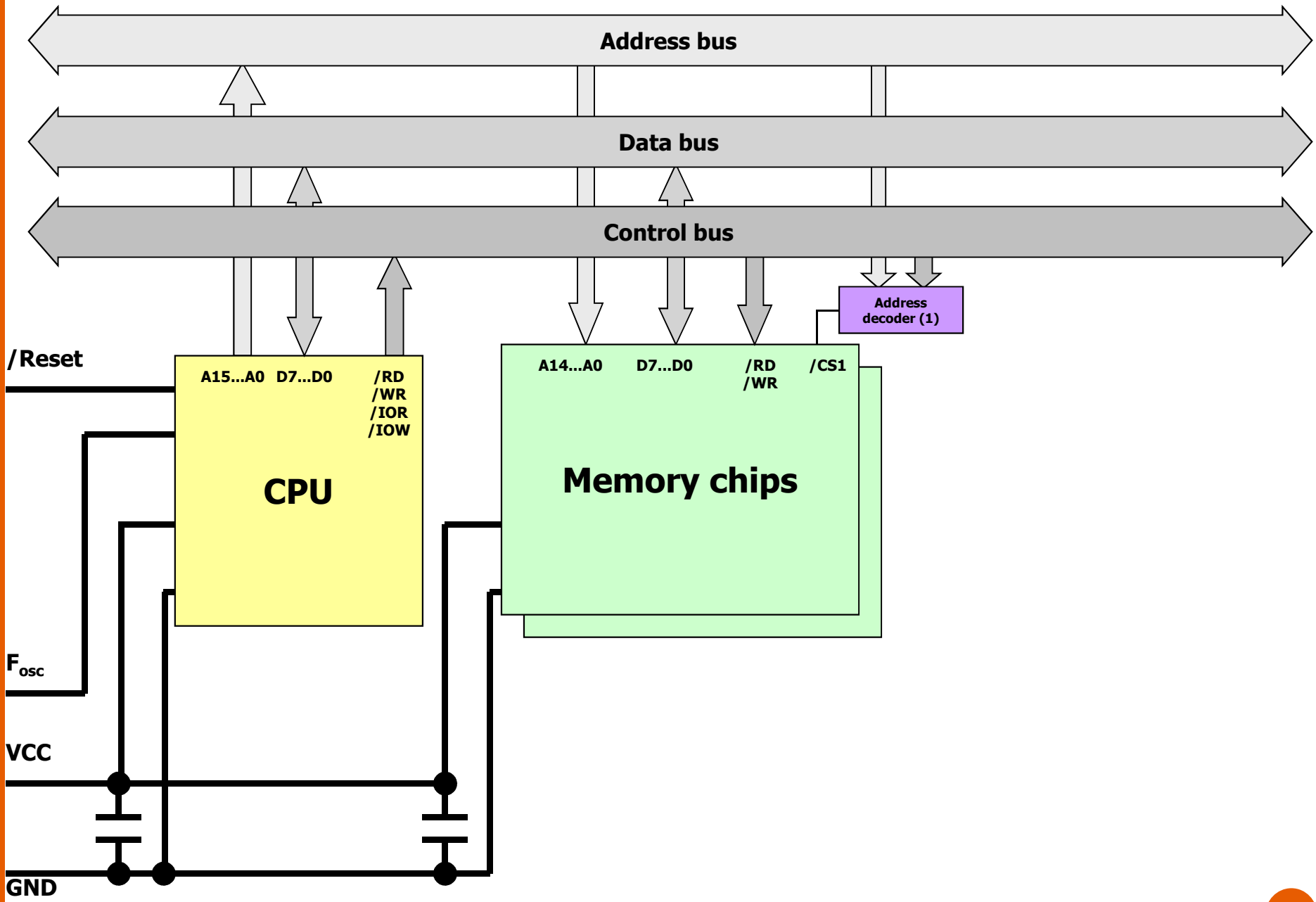

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
Memory chips
F osc
VCC
GND

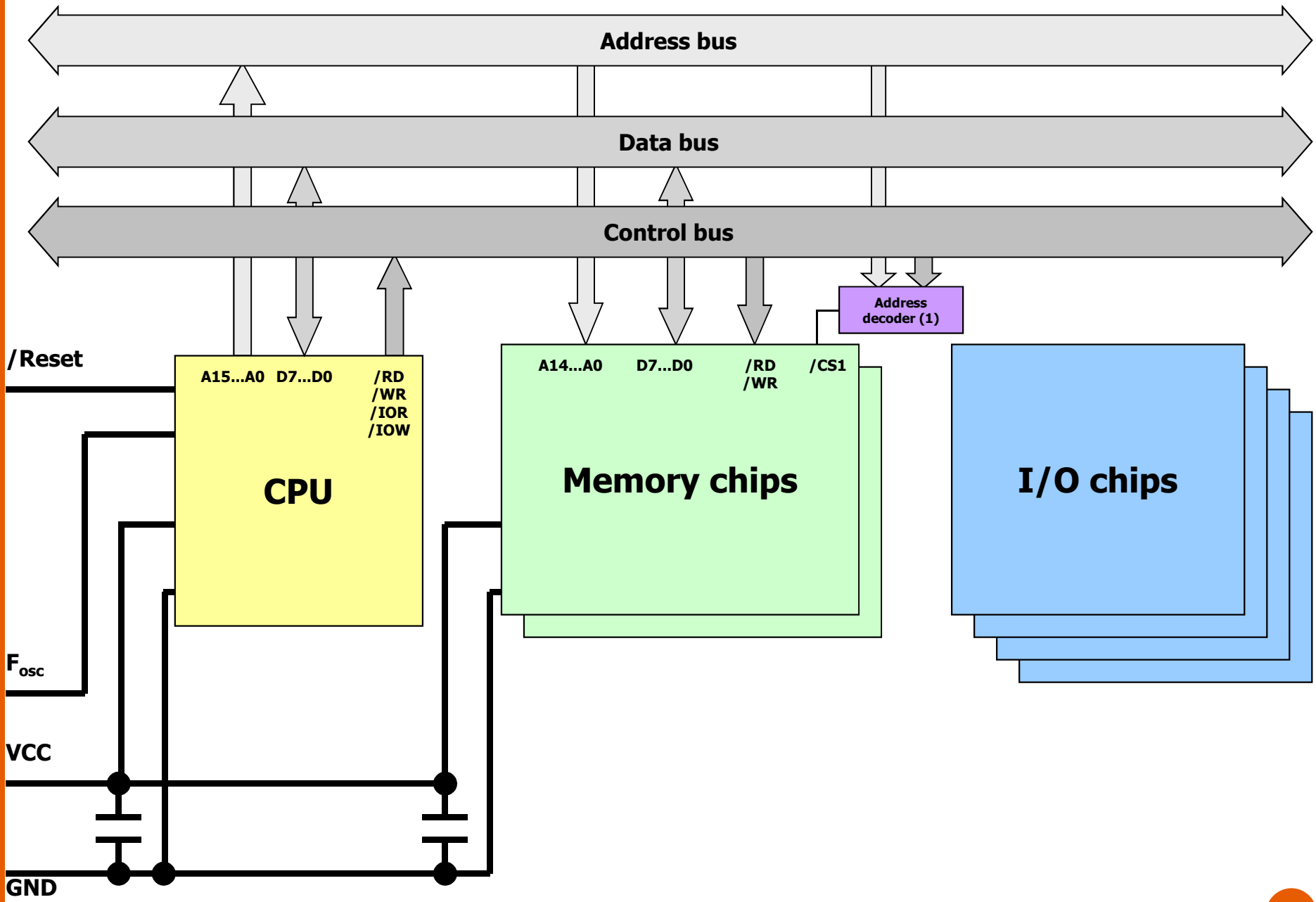

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
Memory chips
Fosc
VCC
GND

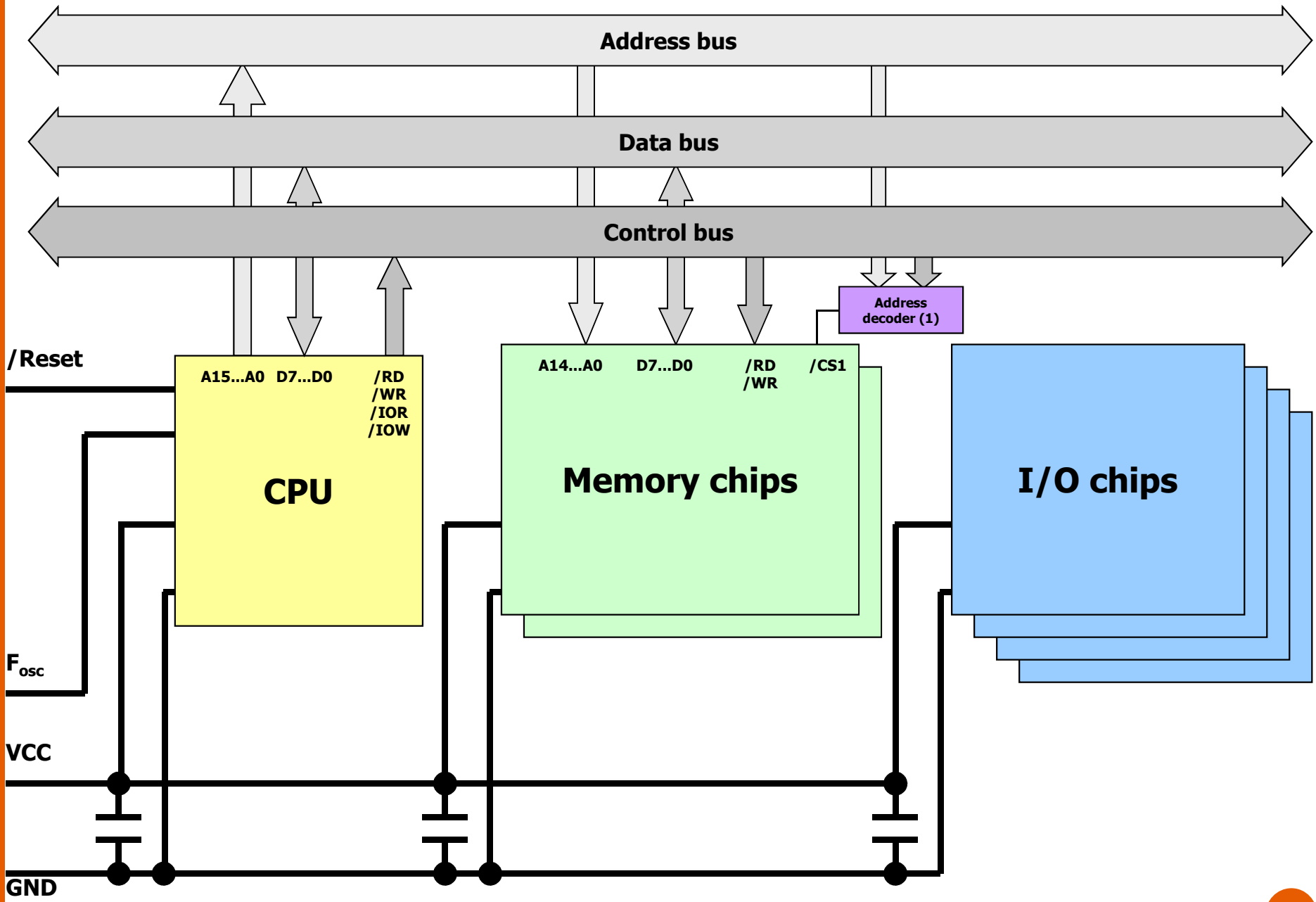

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
Memory chips
Fosc
VCC
GND

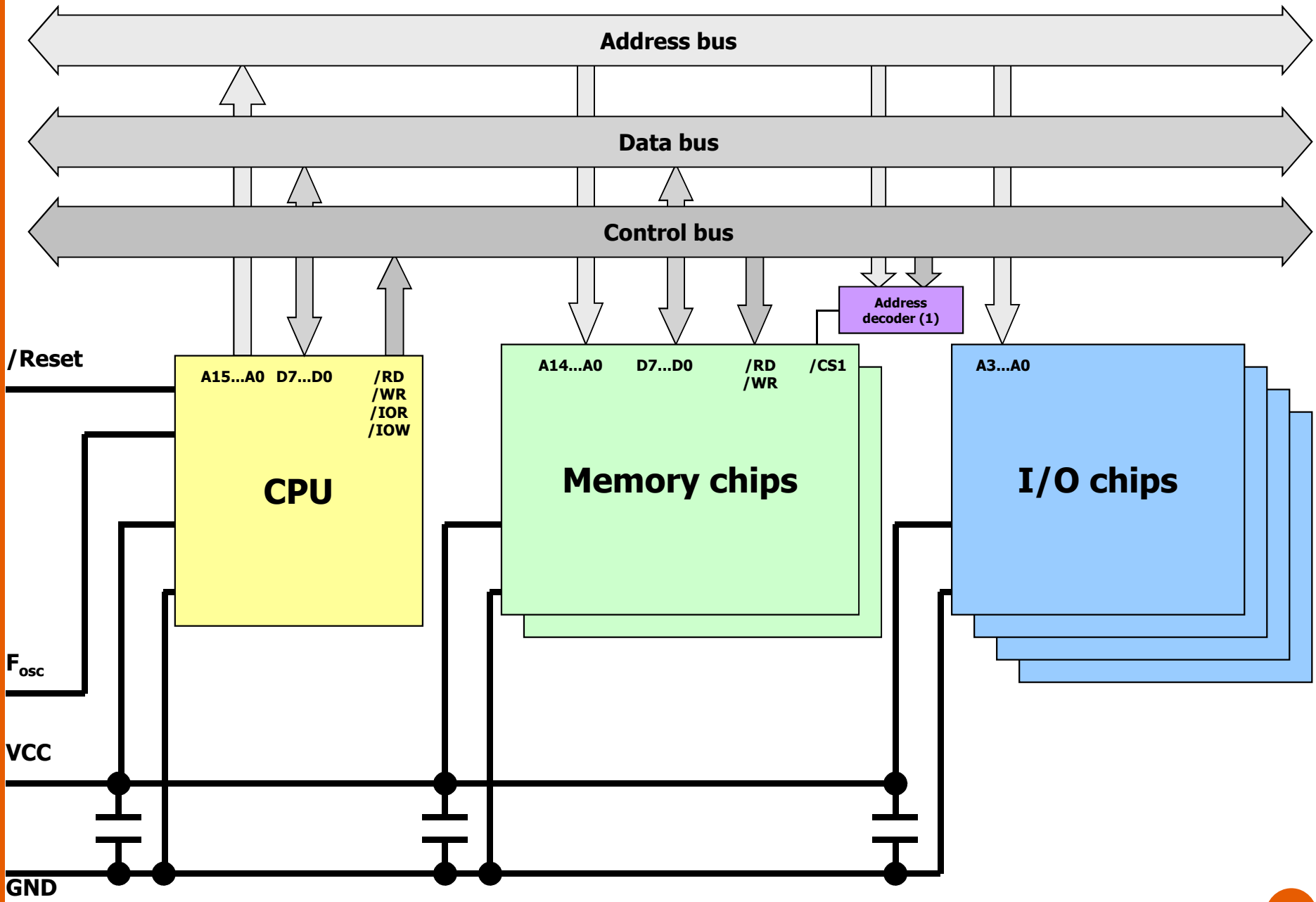

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
Memory chips
F osc
VCC
GND

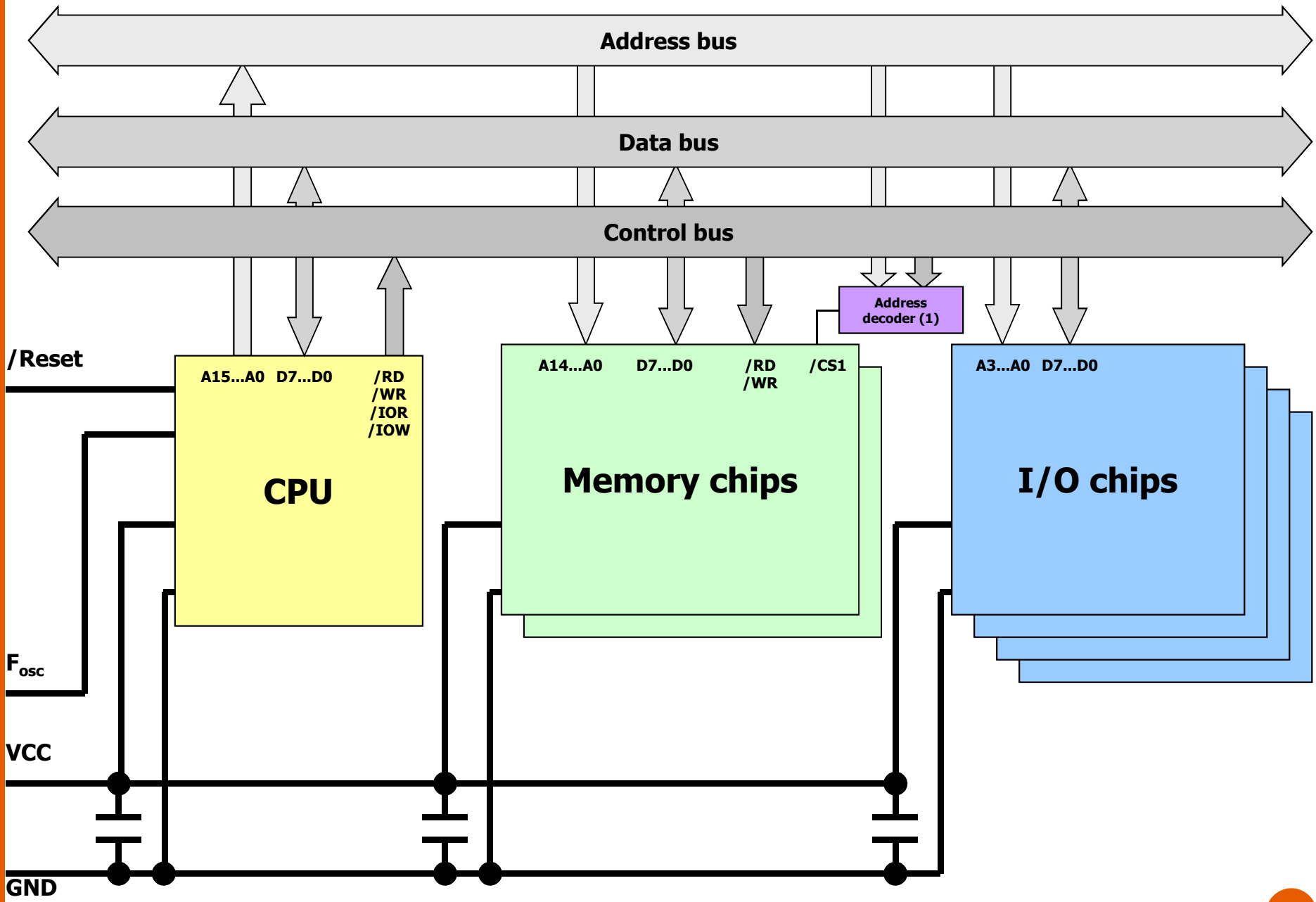

Address bus
Data bus
Control bus
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
Memory chips
Fosc
VCC
GND

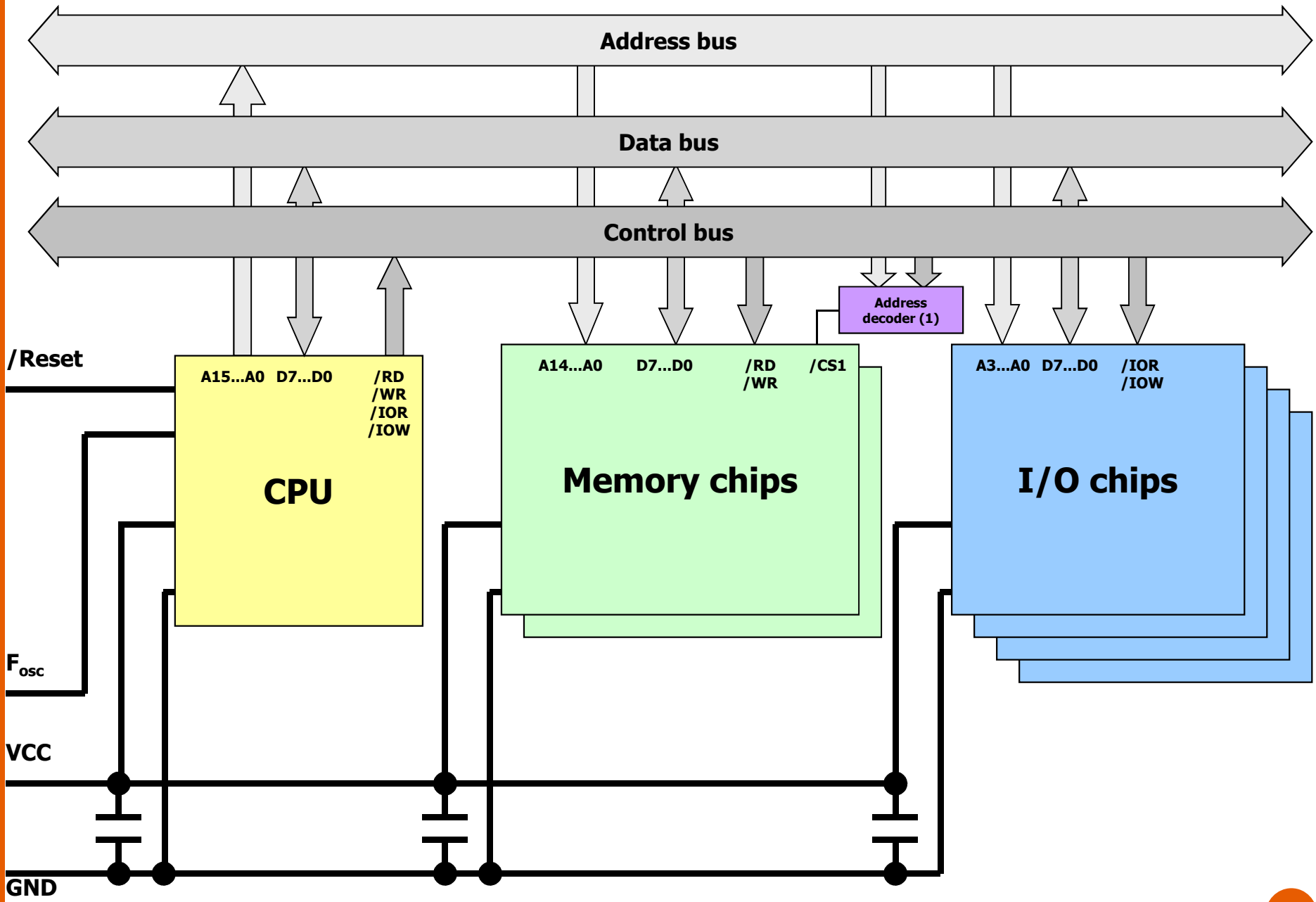

Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
F osc
VCC
GND

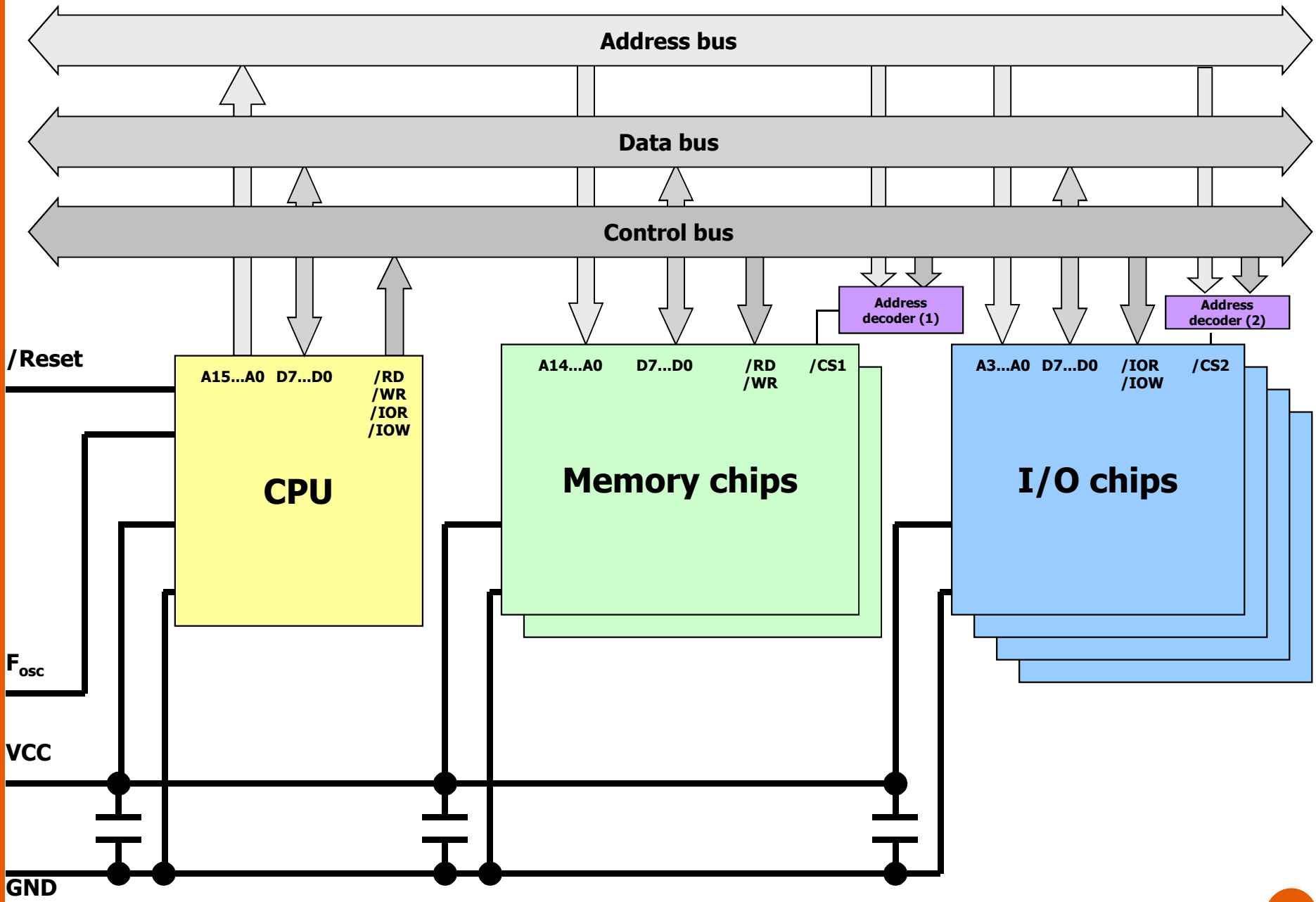

Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
I/O chips
F osc
VCC
GND


Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
I/O chips
Fosc
VCC
GND


Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
A3...A0
I/O chips
F osc
VCC
GND


Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
A3...A0
D7...D0
I/O chips
F osc
VCC
GND

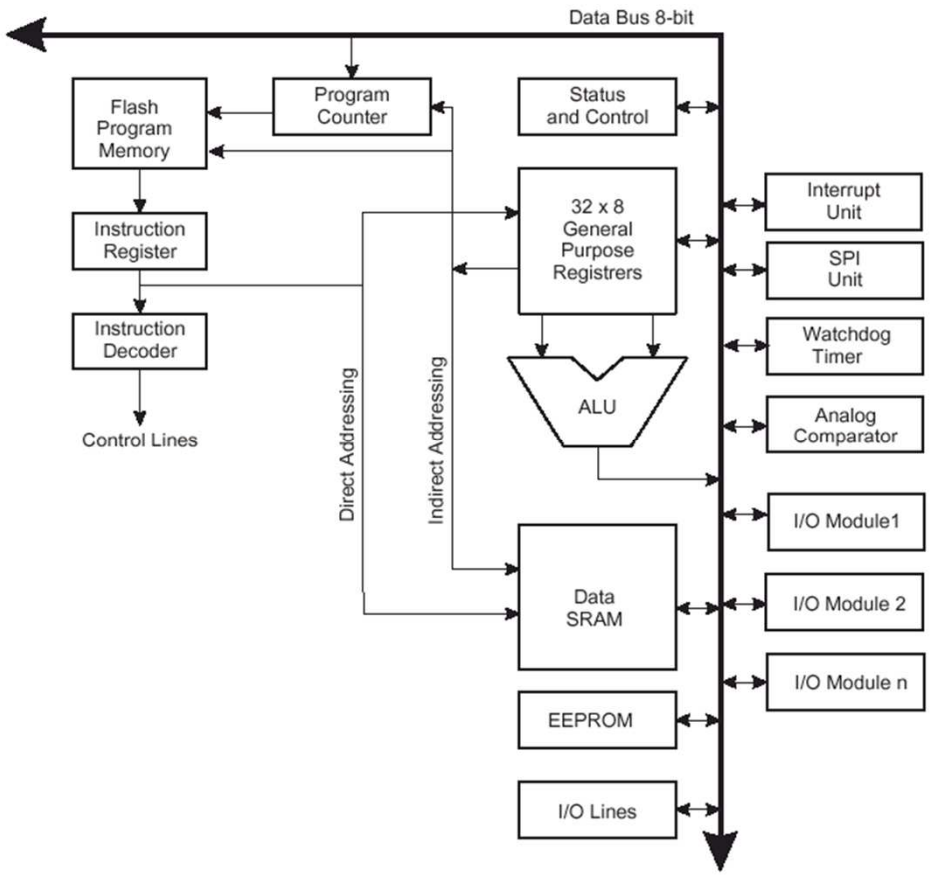

Address bus
Data bus
Control bus
Address decoder (1)
/Reset
A15...A0
D7...D0
/RD
/WR
/IOR
/IOW
CPU
A14...A0
D7...D0
/RD
/WR
/CS1
Memory chips
A3...A0
D7...D0
/IOR
/IOW
I/O chips
F_{osc}
VCC
GND


Address bus
Data bus
Control bus
Address decoder (1)
Address decoder (2)
/Reset
A15...A0 D7...D0 /RD /WR /IOR /IOW
CPU
A14...A0 D7...D0 /RD /WR /CS1
Memory chips
A3...A0 D7...D0 /IOR /IOW /CS2
I/O chips
Fosc
VCC
GND

Examples of MCUs

Examples of MCUs

- AVR family
  - CPU: 8-bit, Harvard architecture, RISC
  - Internal memory
    - ROM (<256KB)
    - RAM (<8KB)
  - Plenty of internal peripherals
    - USART/UART, I2C/TWI, SPI, USB, A/D, PWM, ...
  - High processing power
    - 1 MIPS/MHz
  - Many varieties in the family
    - 99(mega), 33(tiny), 30(Xmega)

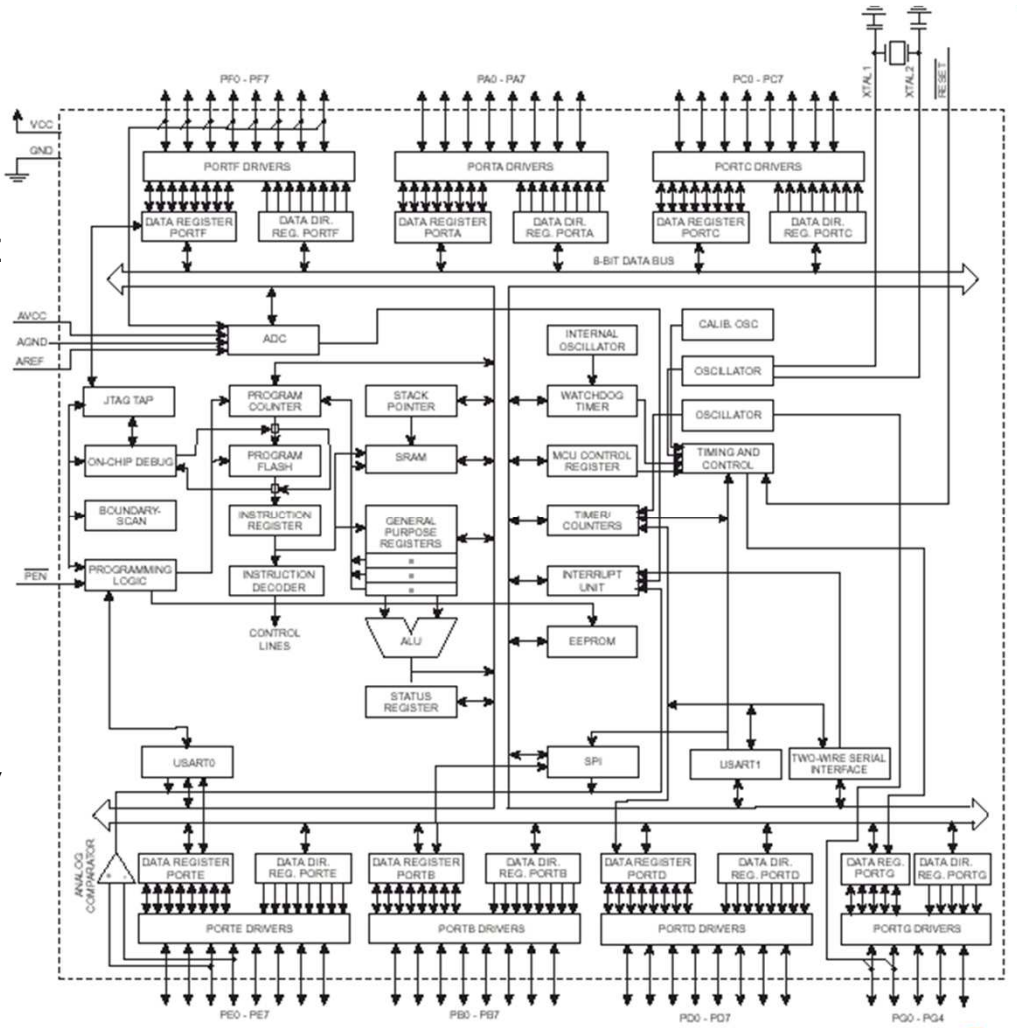


Źródło: Logo i materiały informacyjne firmy Atmel

Examples of MCUs

- AVR family
  - Most popular family member: ATMega328P
    - ROM/EEPROM/RAM: 32KB/1KB/2KB
    - Fosc (clock frequency): 16MHz
    - Vcc: 2.7...5V
  - Most powerful: ATMega2561
    - ROM/EEPROM/RAM: 256KB/4KB/8KB
    - Fosc: 16MHz
    - Vcc: 2.7...5V
- OS for AVR

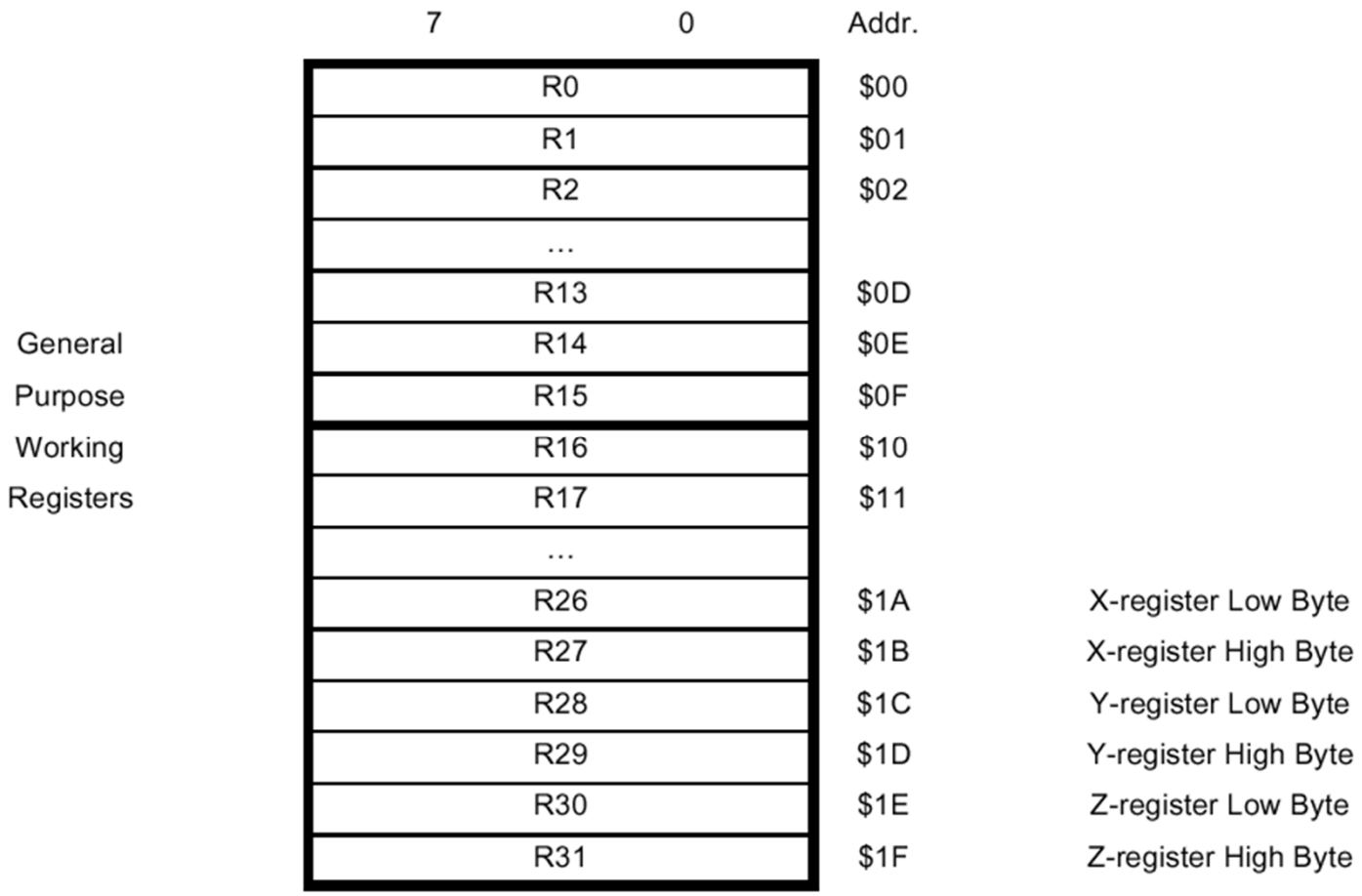
  TinyOS, Contiki, FreeRTOS, SOS, pico OS, ChibiOS, ...



Źródło: Logo i materiały informacyjne firmy Atmel

Examples of MCUs

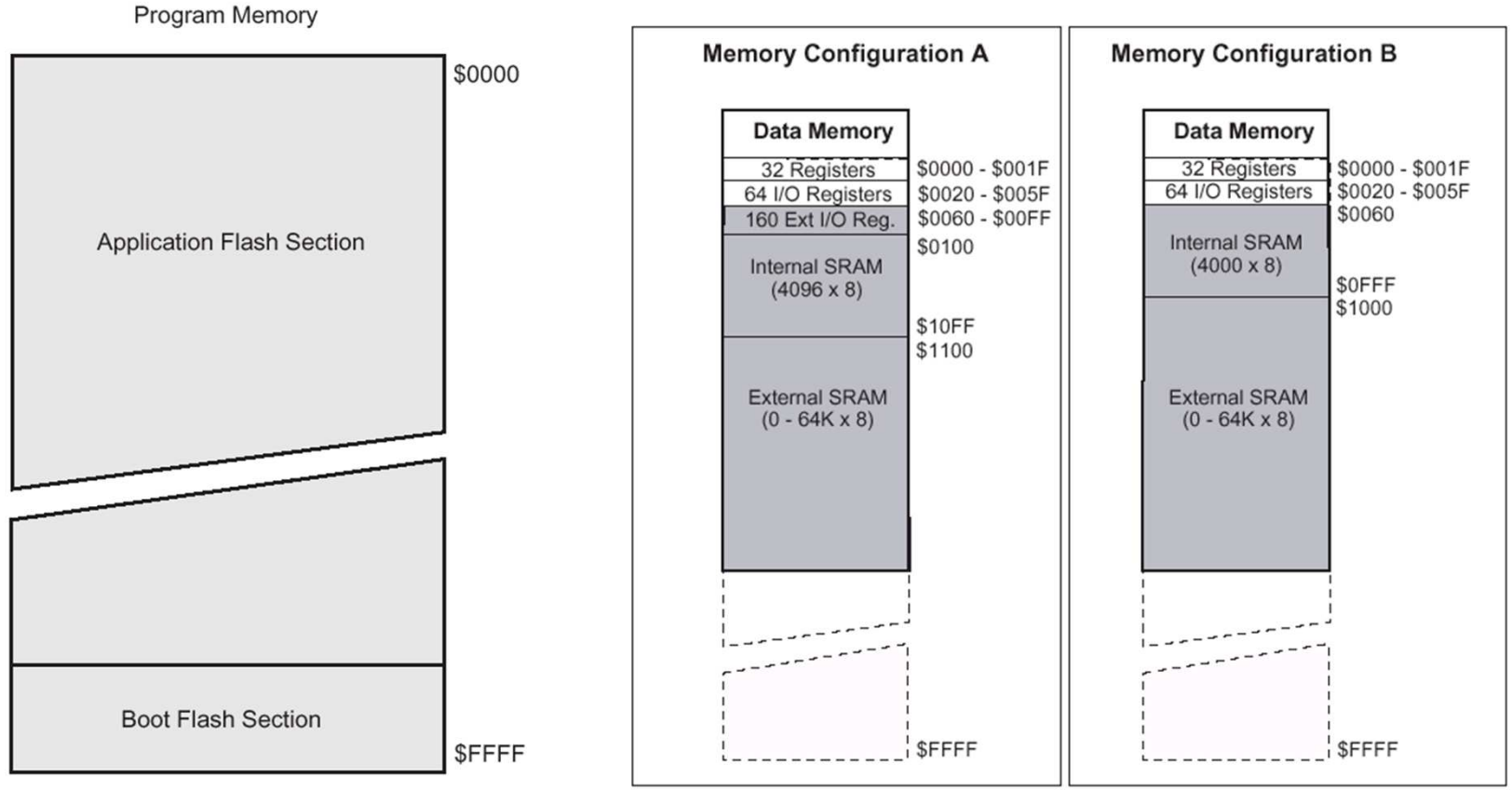
- AVR: registers



Źródło: materiały informacyjne firmy Atmel

Examples of MCUs

- AVR: memory maps



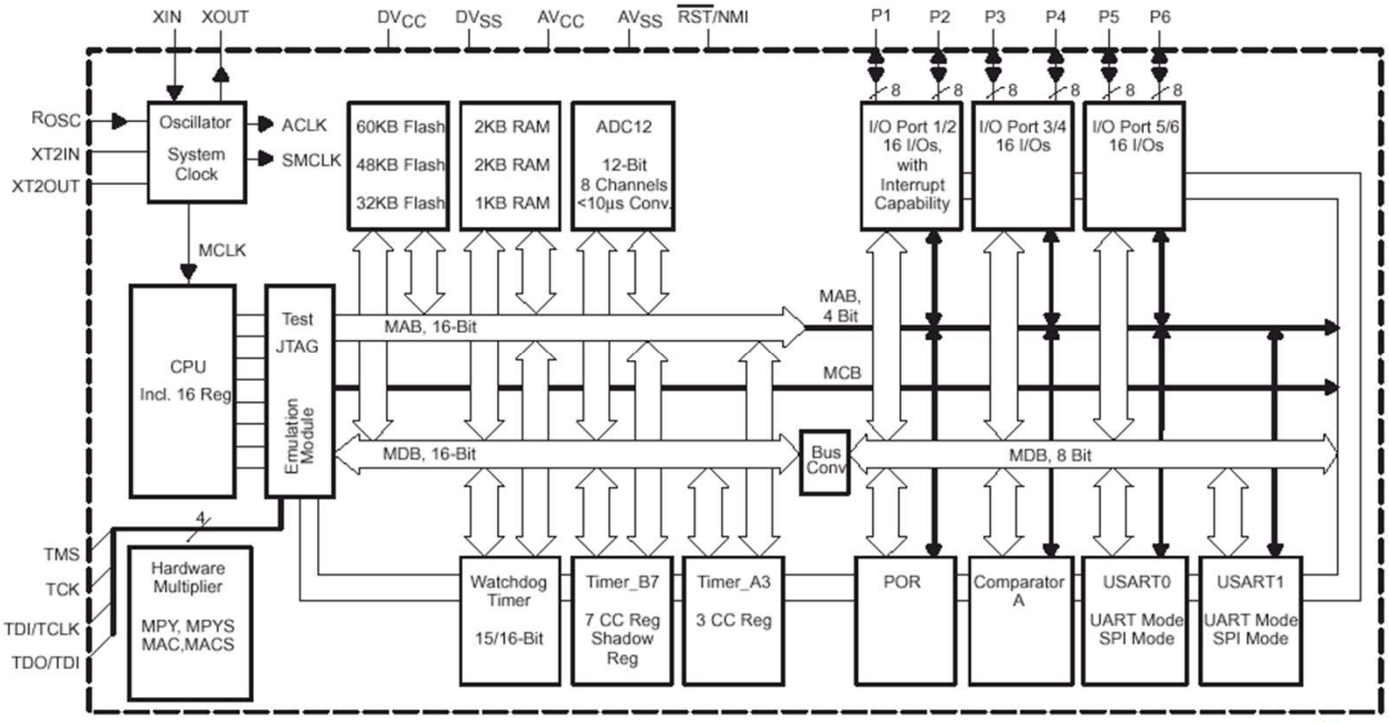
Źródło: materiały informacyjne firmy Atmel

Examples of MCUs

- MSP430 family
  - CPU: 16-bit, Harvard architecture, RISC
  - Internal memory: ROM (<60KB), RAM (<8KB)
  - Highly energy efficient
    - Five operating modes, lowest power mode: 0.1uA, fast waking-up: 6us
    - Active

mode:

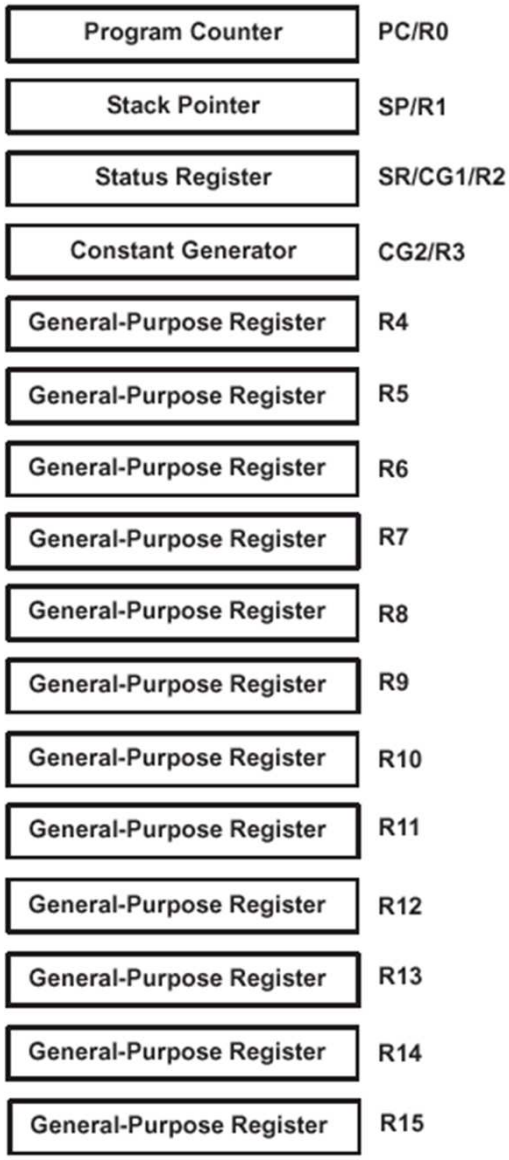
260uA/1MHz



Źródło: Logo i materiały informacyjne firmy TI

Examples of MCUs

- MSP430: registers



Źródło: Materiały informacyjne firmy Texas Instruments

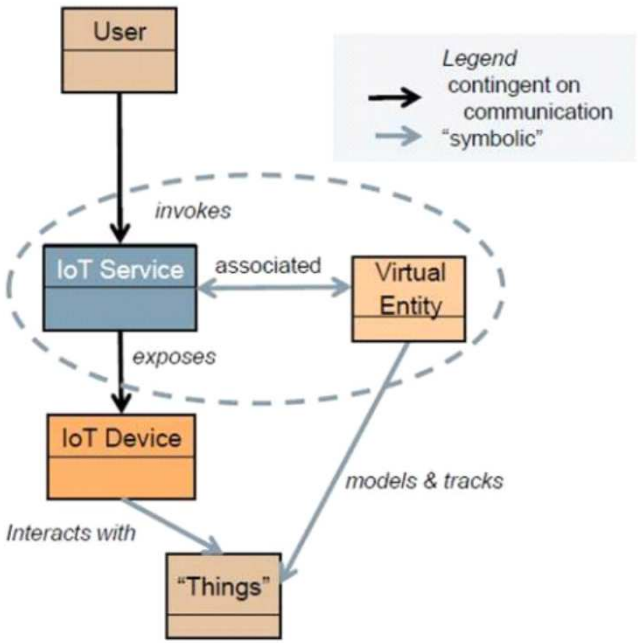
Examples of MCUs

- MSP430: memory maps for different devices

		MSP430F133	MSP430F135	MSP430F147 MSP430F1471	MSP430F148 MSP430F1481	MSP430F149 MSP430F1491
Memory Main: interrupt vector Main: code memory	Size Flash Flash	8KB 0FFFFh – 0FFE0h 0FFFFh – 0E000h	16KB 0FFFFh – 0FFE0h 0FFFFh – 0C000h	32KB 0FFFFh – 0FFE0h 0FFFFh – 08000h	48KB 0FFFFh – 0FFE0h 0FFFFh – 04000h	60KB 0FFFFh – 0FFE0h 0FFFFh – 01100h
Information memory	Size Flash	256 Byte 010FFh – 01000h	256 Byte 010FFh – 01000h	256 Byte 010FFh – 01000h	256 Byte 010FFh – 01000h	256 Byte 010FFh – 01000h
Boot memory	Size ROM	1KB 0FFFh – 0C00h	1KB 0FFFh – 0C00h	1KB 0FFFh – 0C00h	1KB 0FFFh – 0C00h	1KB 0FFFh – 0C00h
RAM	Size	256 Byte 02FFh – 0200h	512 Byte 03FFh – 0200h	1KB 05FFh – 0200h	2KB 09FFh – 0200h	2KB 09FFh – 0200h
Peripherals	16-bit 8-bit 8-bit SFR	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h	01FFh – 0100h 0FFh – 010h 0Fh – 00h

Źródło: Materiały informacyjne firmy Texas Instruments

Thank you!


User
Legend
contingent on
communication
"symbolic"
invokes
IoT Service
associated
Virtual
Entity
exposes
IoT Device
models & tracks
Interacts with
"Things"