

CS-200

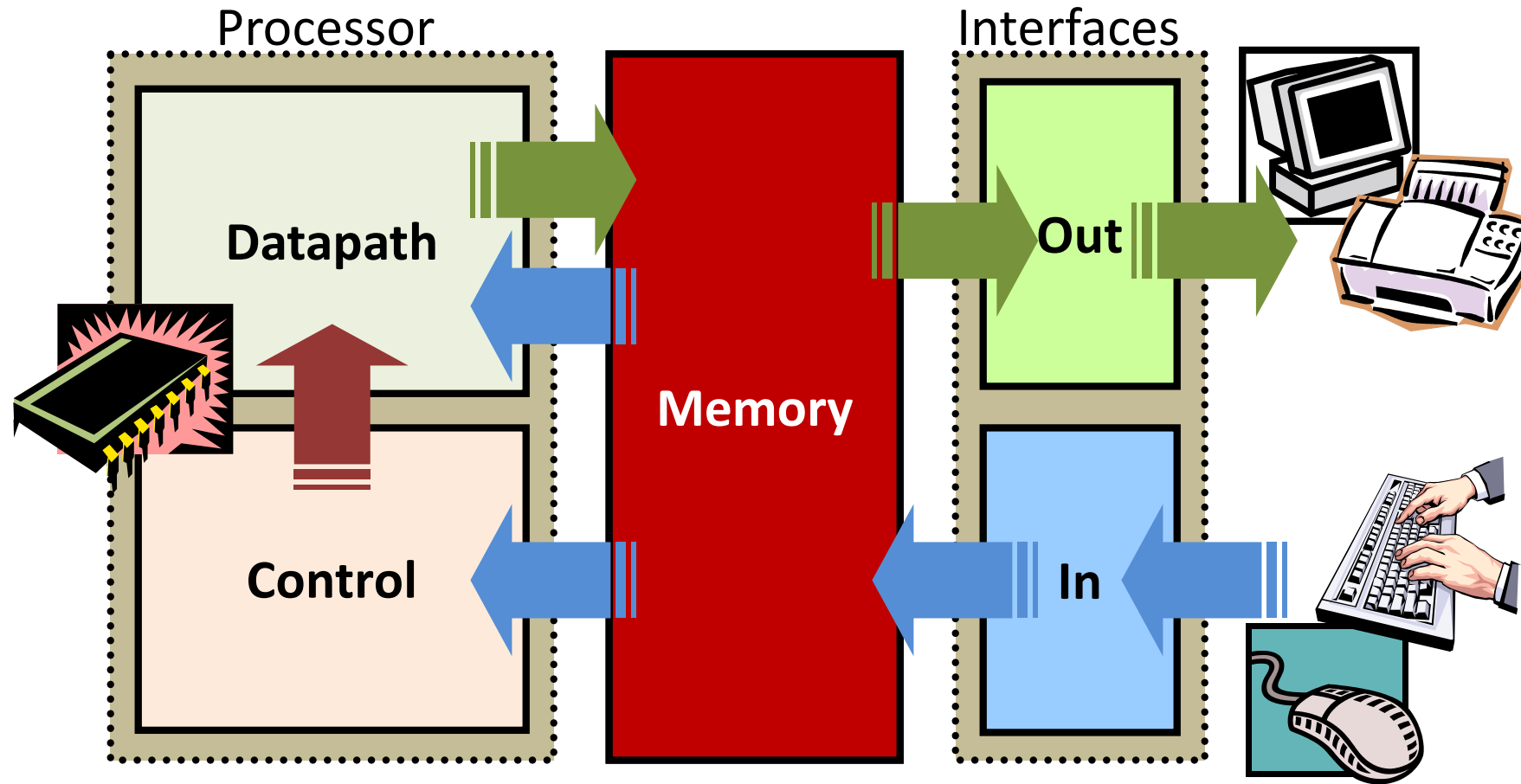
Computer Architecture

Part 2b. Processor, I/Os, and Exceptions

Inputs and Outputs

Paolo Ienne
<paolo.ienne@epfl.ch>

The Five Classic Components of a Computer



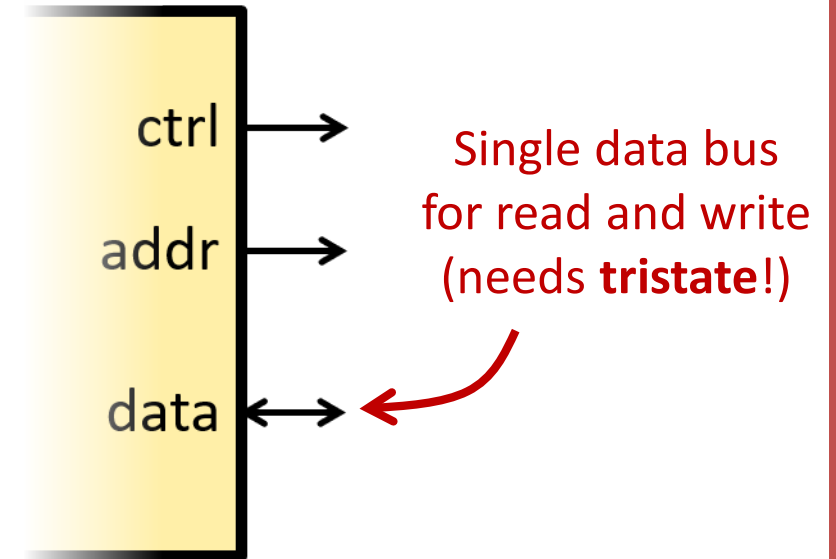
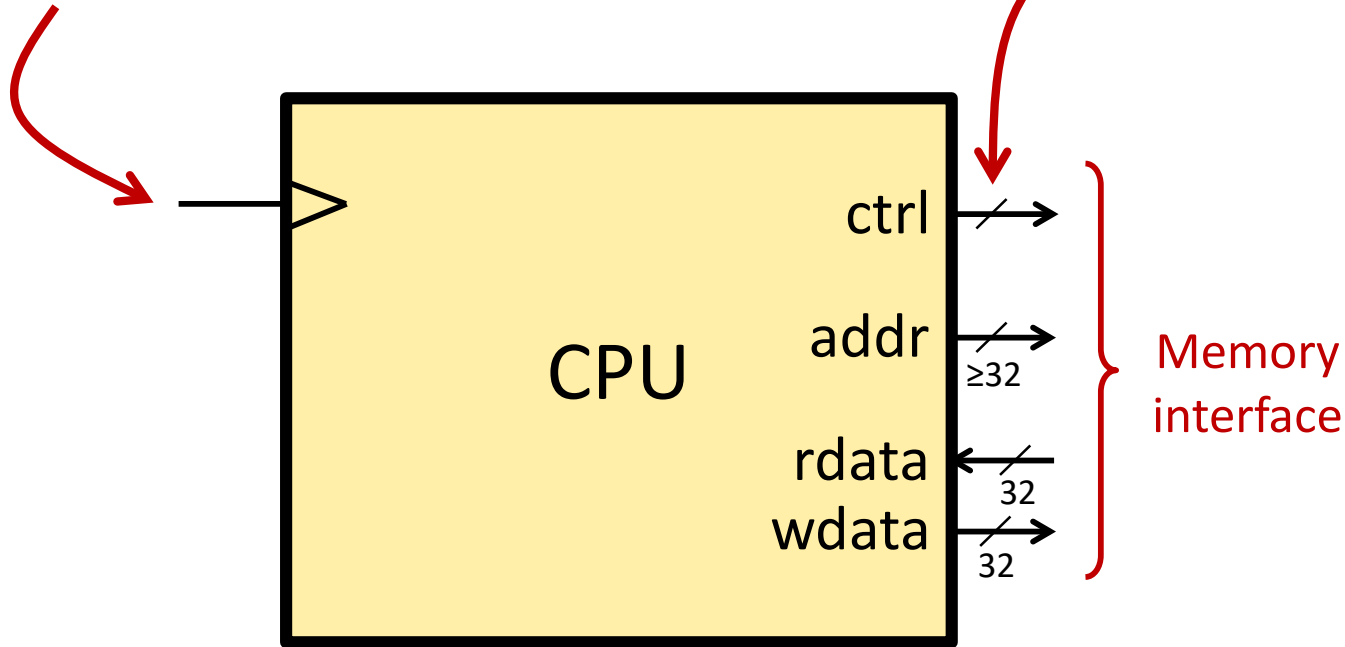
The CPU

A “very” **sequential** component
(but from now on we may
omit the **clock** in diagrams)

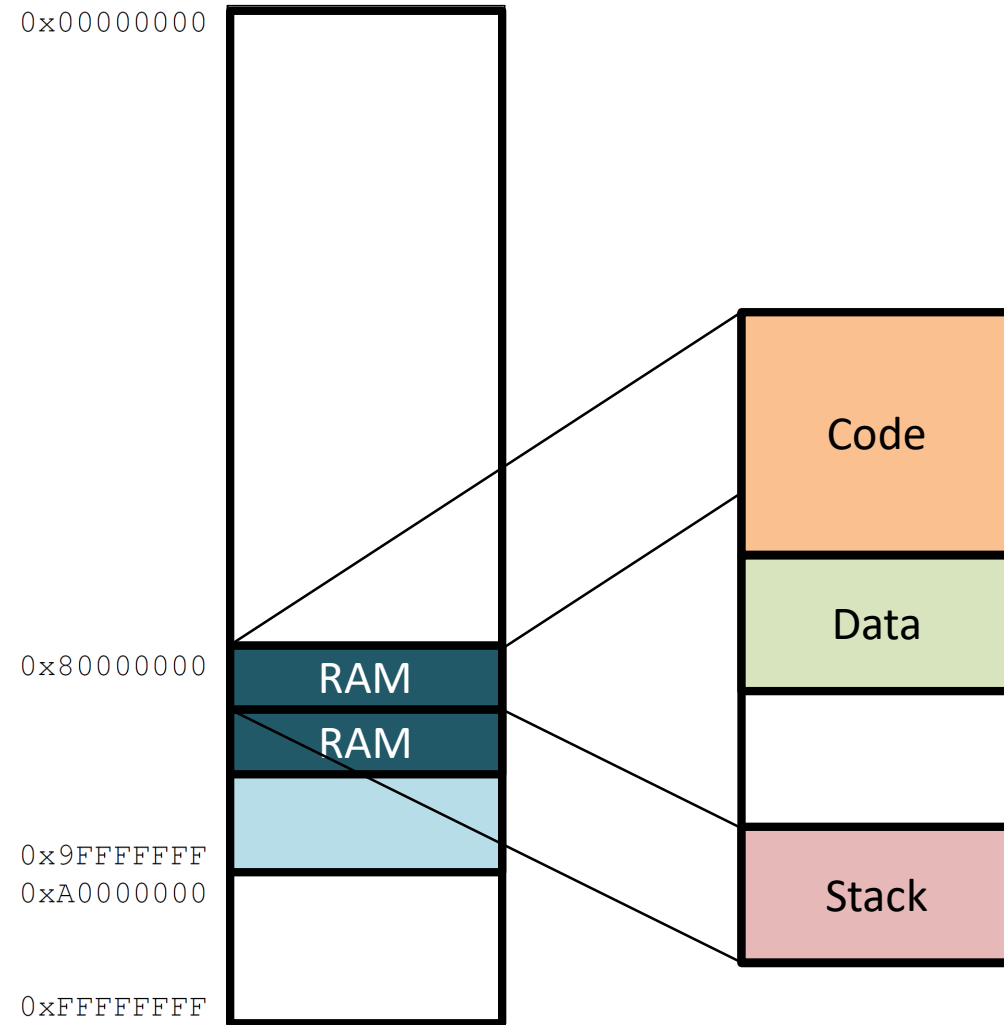
Signals like

CE = Circuit Enable = address is valid

WE = Write Enable = access is a store



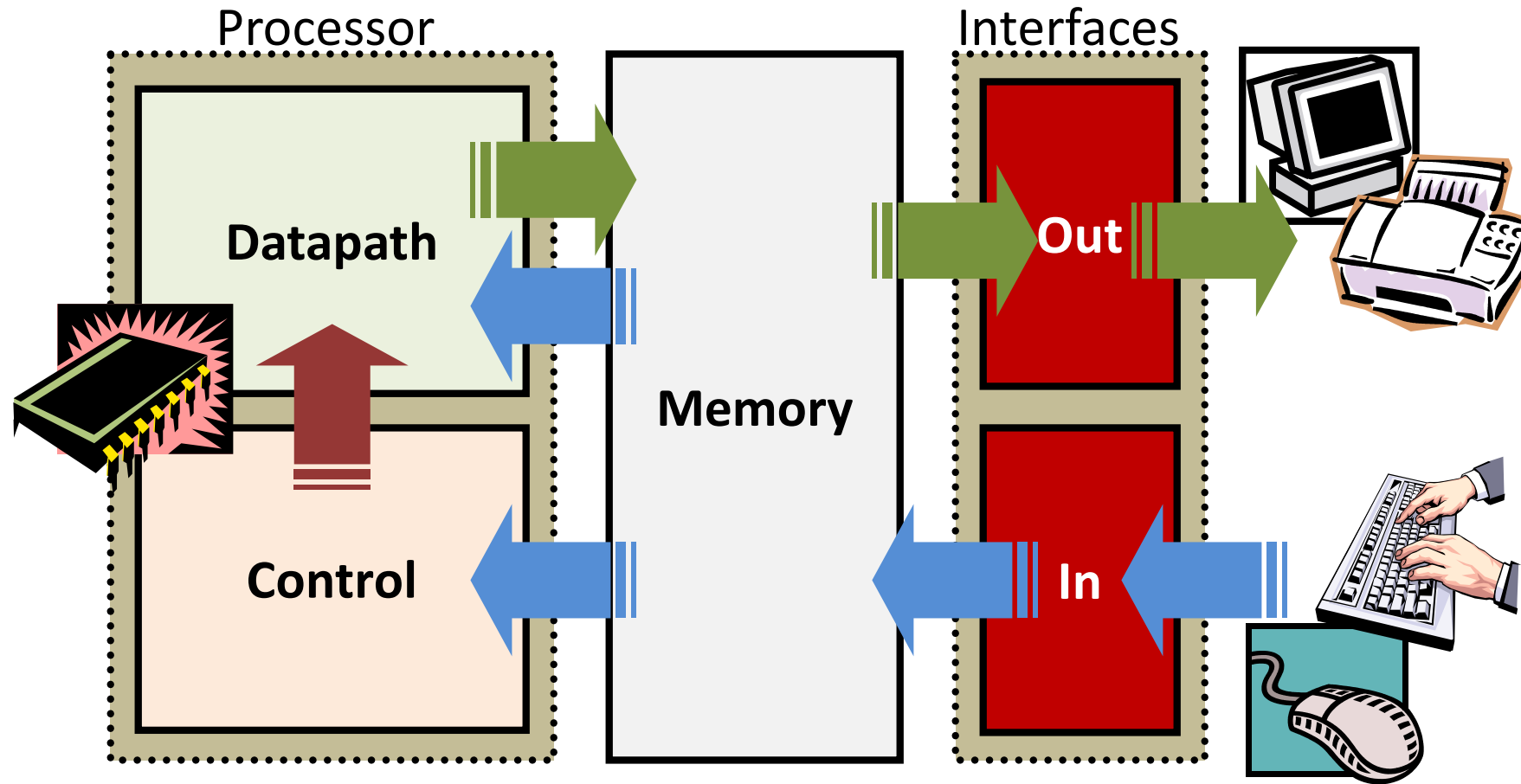
Physical Memory Map



4



The Five Classic Components of a Computer

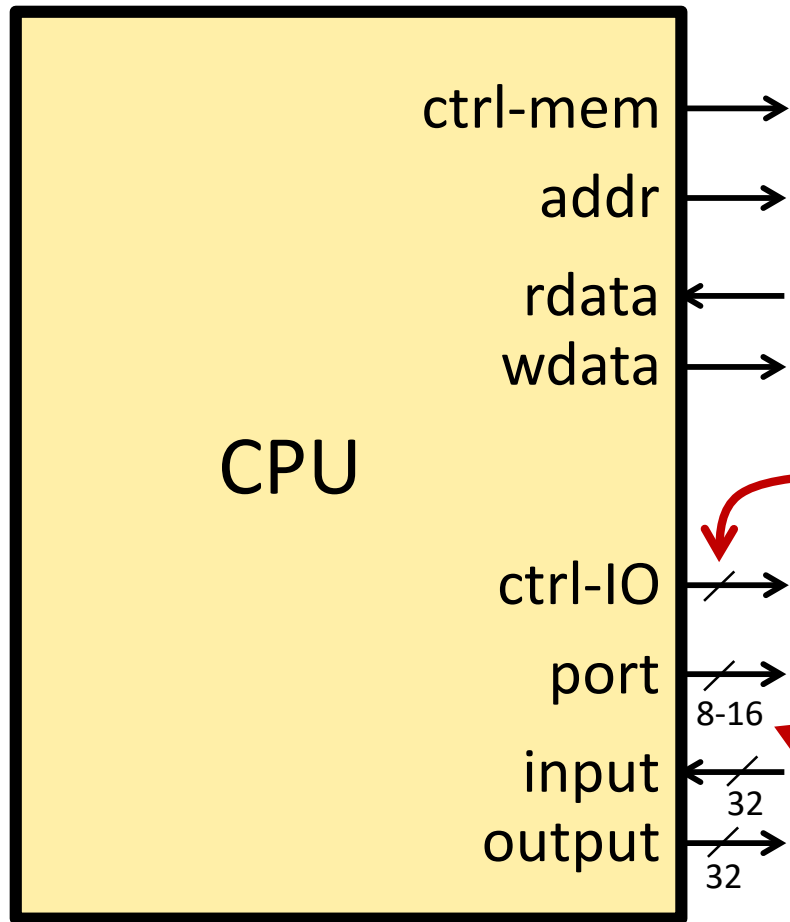


Input/Output Devices (I/Os)

Type	Peripheral	Data Rate
Human Interaction	Keyboard	~kbps
Human Interaction	Mouse	~kbps
Generic	Serial Port (RS-232)	115.2 kbps (max)
Generic	Parallel Port (LPT)	150 kbps
Generic	USB 4.0	20-40 Gbps
Generic	Bluetooth 5.0	2 Mbps
Generic	PCIe 4.0	16 Gbps per lane
Storage	SATA III (HDD/SSD)	6.0 Gbps
Storage	NVMe (PCIe 4.0)	64 Gbps (4-lane)
Networking	Ethernet (10BASE-T)	10 Mbps
Networking	10 Gigabit Ethernet (10GBASE-T)	10 Gbps
Networking	Wi-Fi 6 (802.11ax)	Up to 9.6 Gbps
Displays	VGA (analog video)	0.6-1.5 Gbps (approx.)
Displays	HDMI 2.1	48 Gbps
Optical Discs	CD-ROM	150 KB/s (1x) - 7.68 MB/s (52x)
Optical Discs	DVD-ROM	1.32 MB/s (1x) - 21.1 MB/s (16x)
Optical Discs	Blu-ray	4.5 MB/s (1x) - 54 MB/s (12x)

Accessing I/Os: Port Mapped I/O (PMIO)

- Create a **new interface** similar to the memory one



New instructions (e.g., x86 but seldom used):

IN register, port
OUT port, register

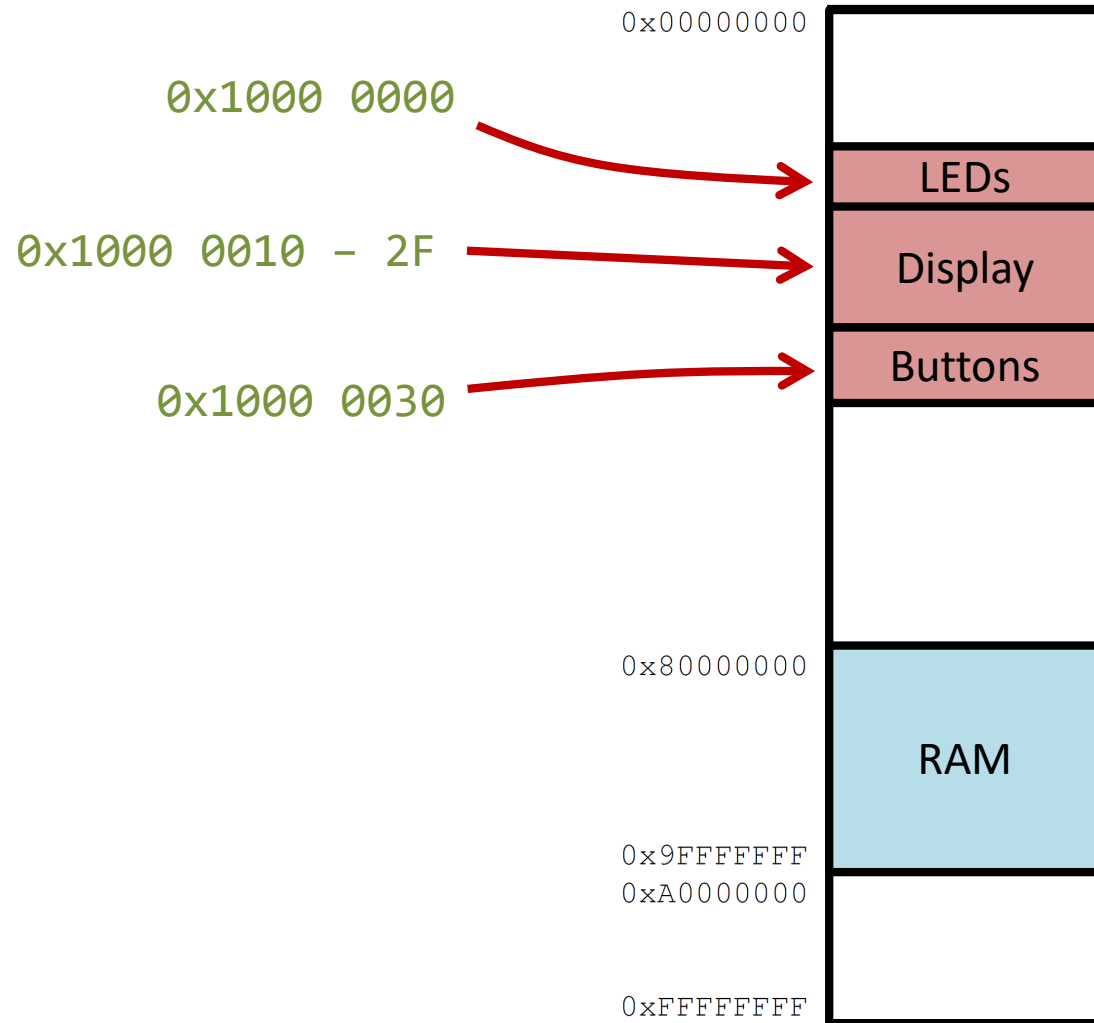
e.g., *IN AL, keyboard*

CE = Circuit Enable =
port number is valid

OE = Output Enable =
I/O access is an output

Maybe not 32 bits for we
have only a few peripherals...

Accessing I/Os: Memory Mapped I/O (MMIO)

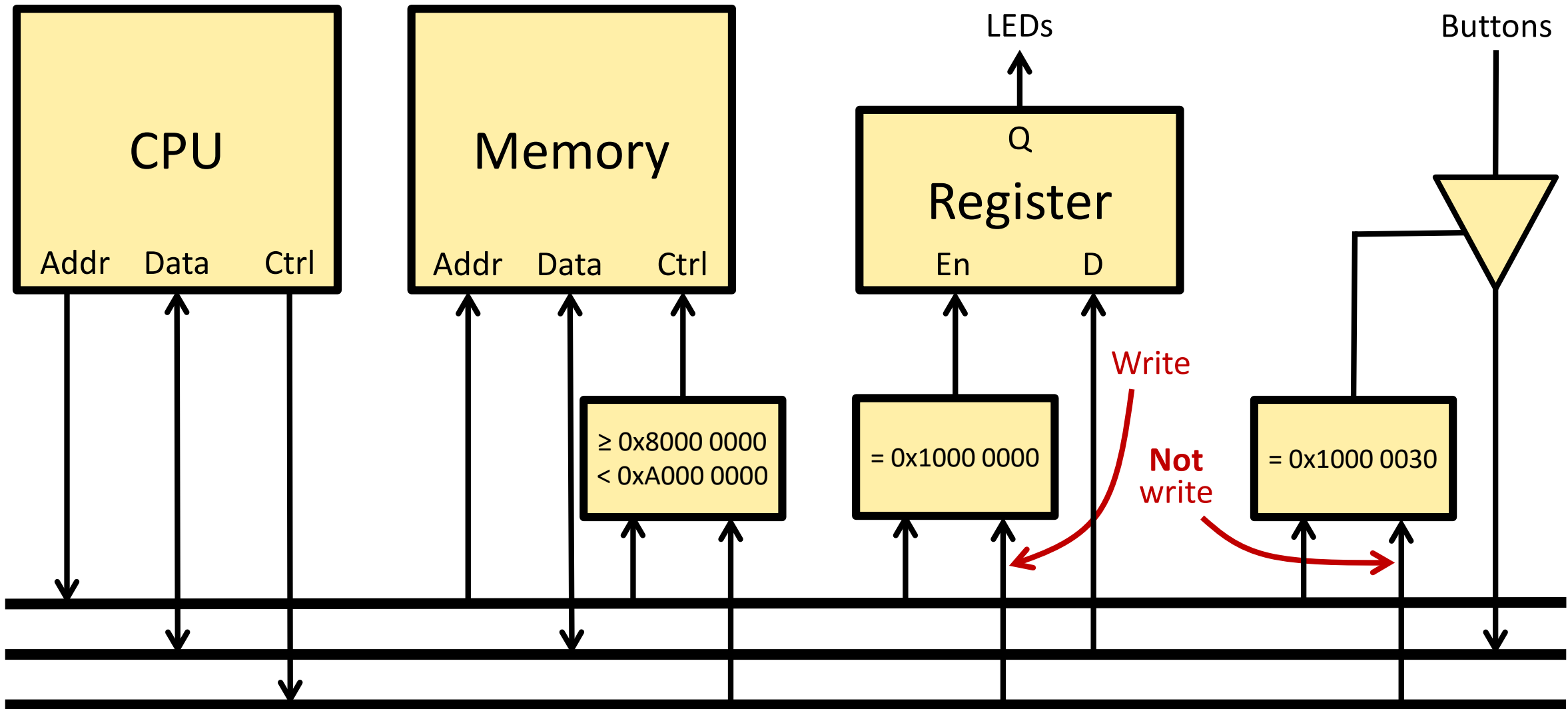


No special hardware needed in the CPU

No special instructions needed:

```
lui    t0, 0x10000    # pointer to I/Os
sw     t1, 0(t0)      # write LEDs
lw     t2, 0x30(t0)   # read buttons
```

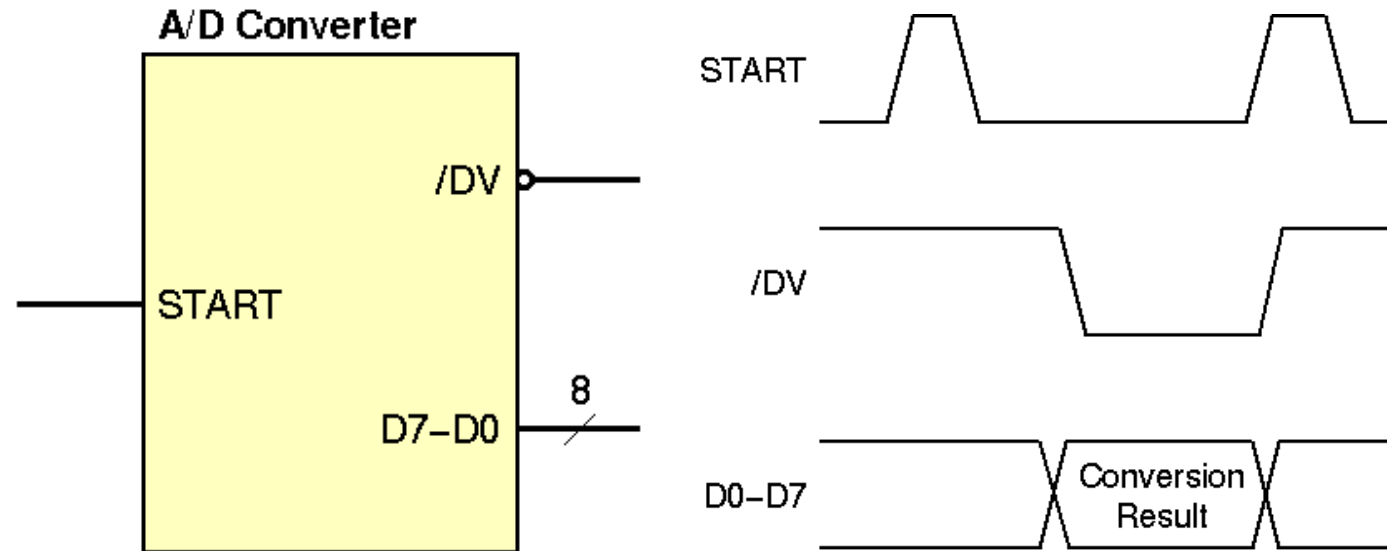
Accessing I/Os: Memory Mapped I/O (MMIO)



Example:

A/D Converter

- Signals:
 - **Start** (START): input; when active begins a new conversion
 - **Data Valid** (/DV): output; when active, D7—D0 are valid
 - **Data** (D7—D0): output; last conversion result



Example:

Simple Bus Interface

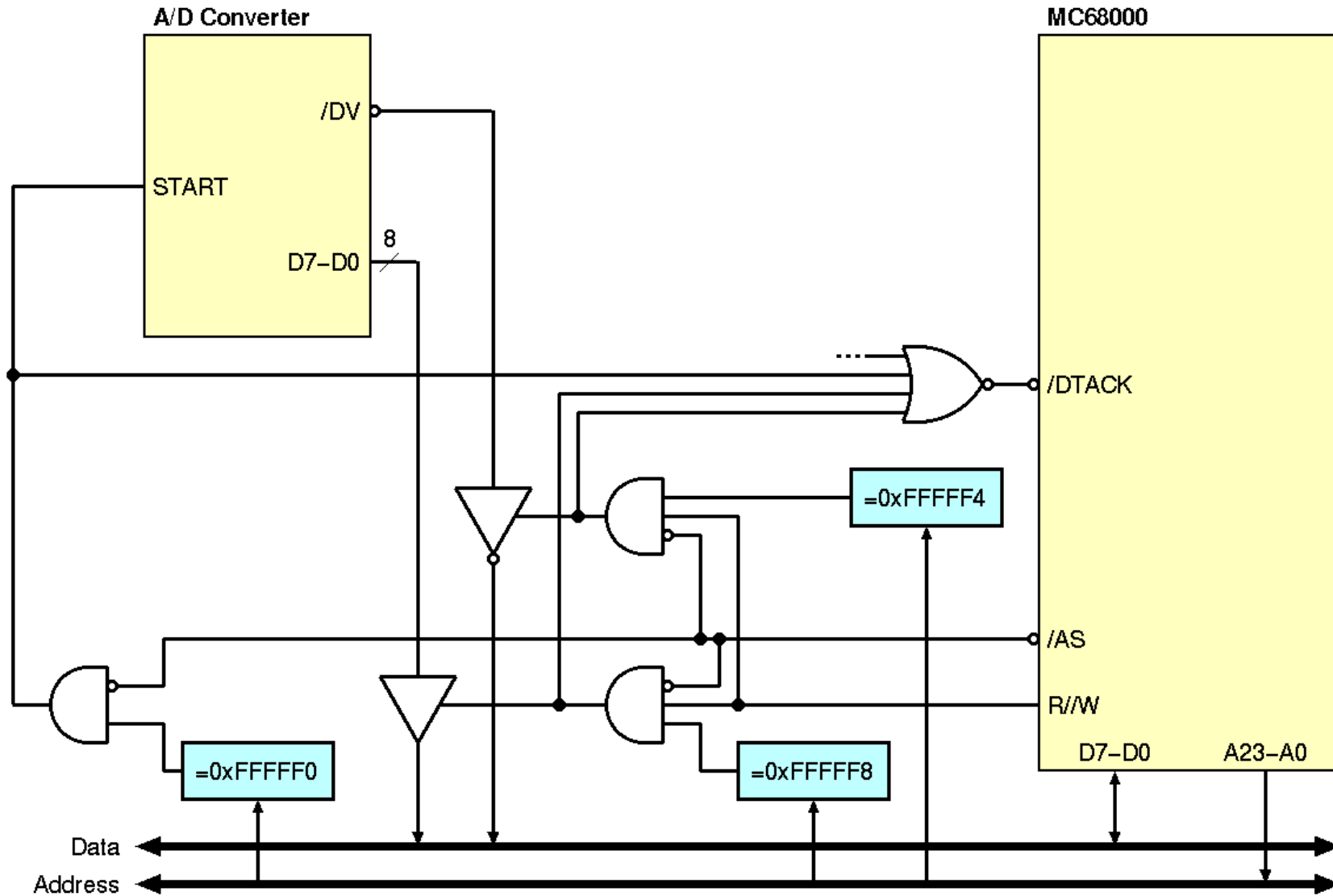
- Suppose that a 8-bit processor has the following signals:
 - **Address** (A23—A0): output; address bus
 - **Data** (D7—D0): input/output; data bus
 - **Address Strobe** (/AS): output; signals the presence of a valid address on the Address bus during a memory access cycle
 - **Read/Write** (R//W): output; signal the direction of the data flow
 - **Data Acknowledge** (/DTACK): input; must be activated at the end of a memory access, when the written data have been latched or the read data are ready
- Similar but not identical to the MC68000
- Just an example but already more complex than busses described so far (/DTACK)

Example:

Memory-Mapped Interface

- Connect the A/D converter described in the previous slide so that:
 - **Any access** (R or W) to address **0xFFFF0** starts a **new conversion**
 - The **Data Valid** signal can be read by the processor at address **0xFFFF4** (bit 0)
 - The **result of the conversion** can be read by the processor at address **0xFFFF8**

A/D Converter: Circuit



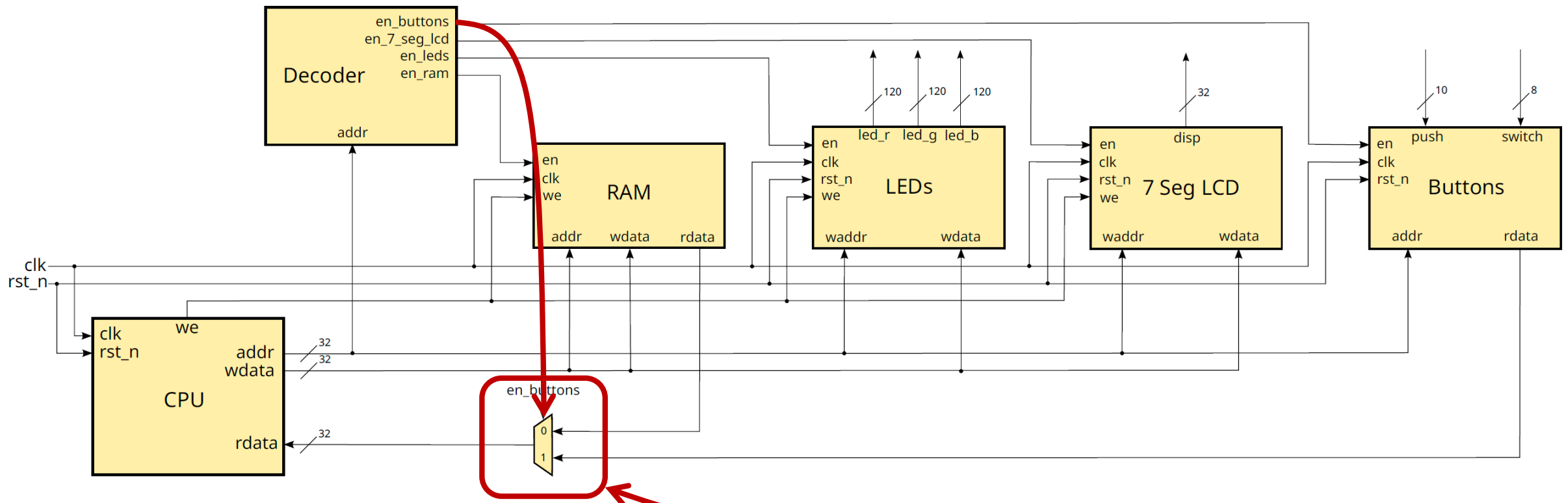
A/D Converter: Software

```
read_adc:    lui    t1, 0xffff
             addi   t1, t1, 0xff0      # t1 = 0xfffff0
             sw     zero, 0(t1)        # start conversion

poll:        lw     t0, 4(t1)          # t0 = DV signal
             beqz   t0, poll           # wait until done

end:         lw     a0, 8($t1)         # a0 = A/D output
             ret
```

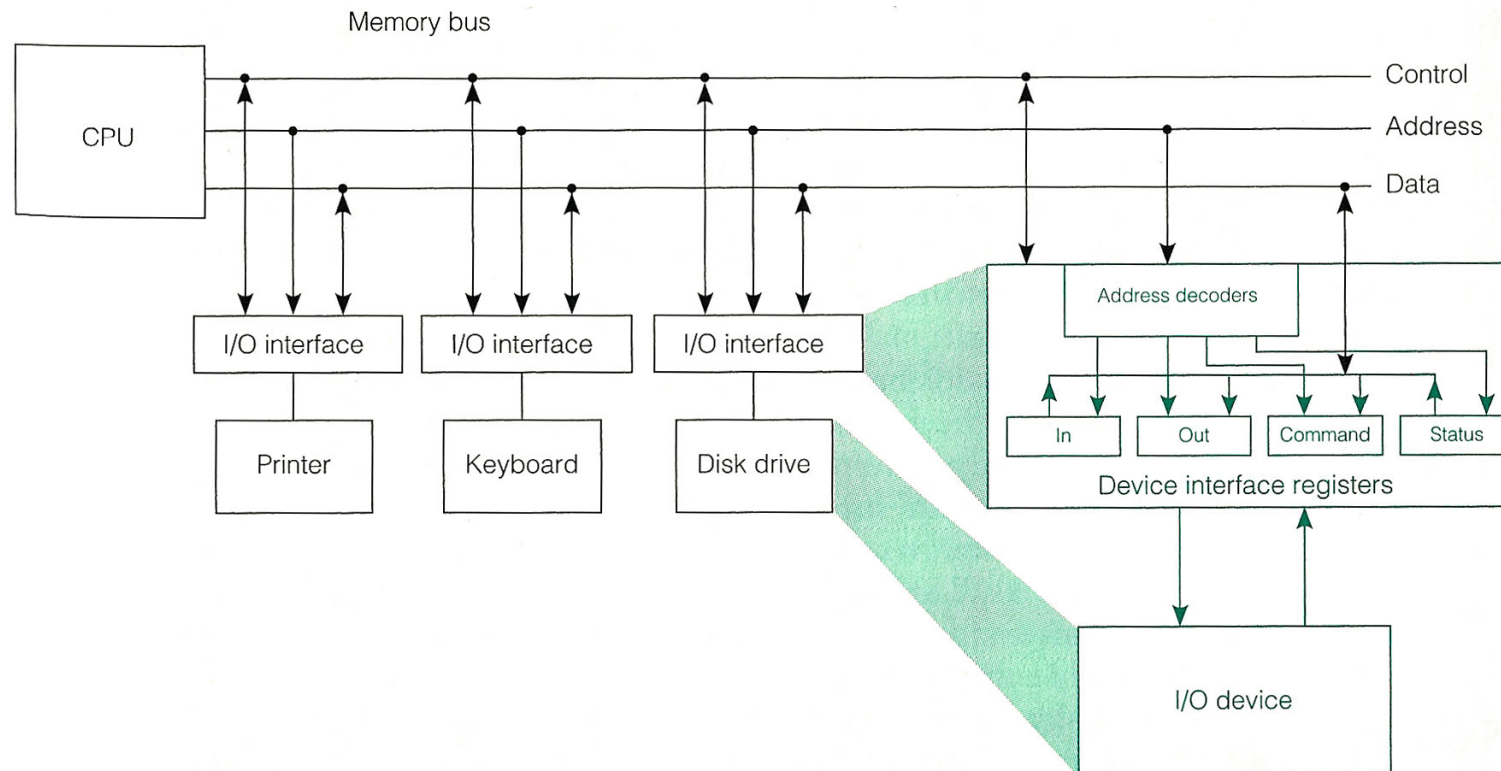
Your System in Lab B



Exactly the same function as the tristate buffers in the previous example

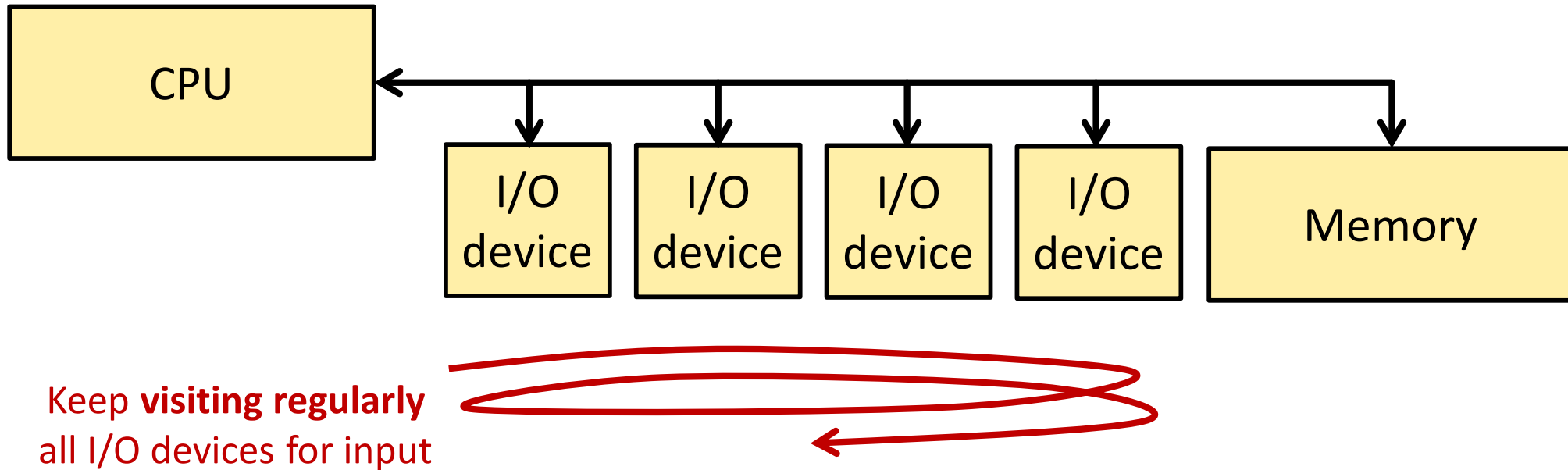
Programmed I/Os

Modern peripherals are more developed programmable systems and have a set of registers which the processor reads and writes (a) to **send** and **receive data** and (b) to **issue commands** and **read the status**



I/O Polling

- How do we know if **a peripheral has data** for us (key pressed, packet arrived, etc.)?



- Very expensive:** if the device is fast and requires immediate action, the processor must spend too much time to check **frequently**