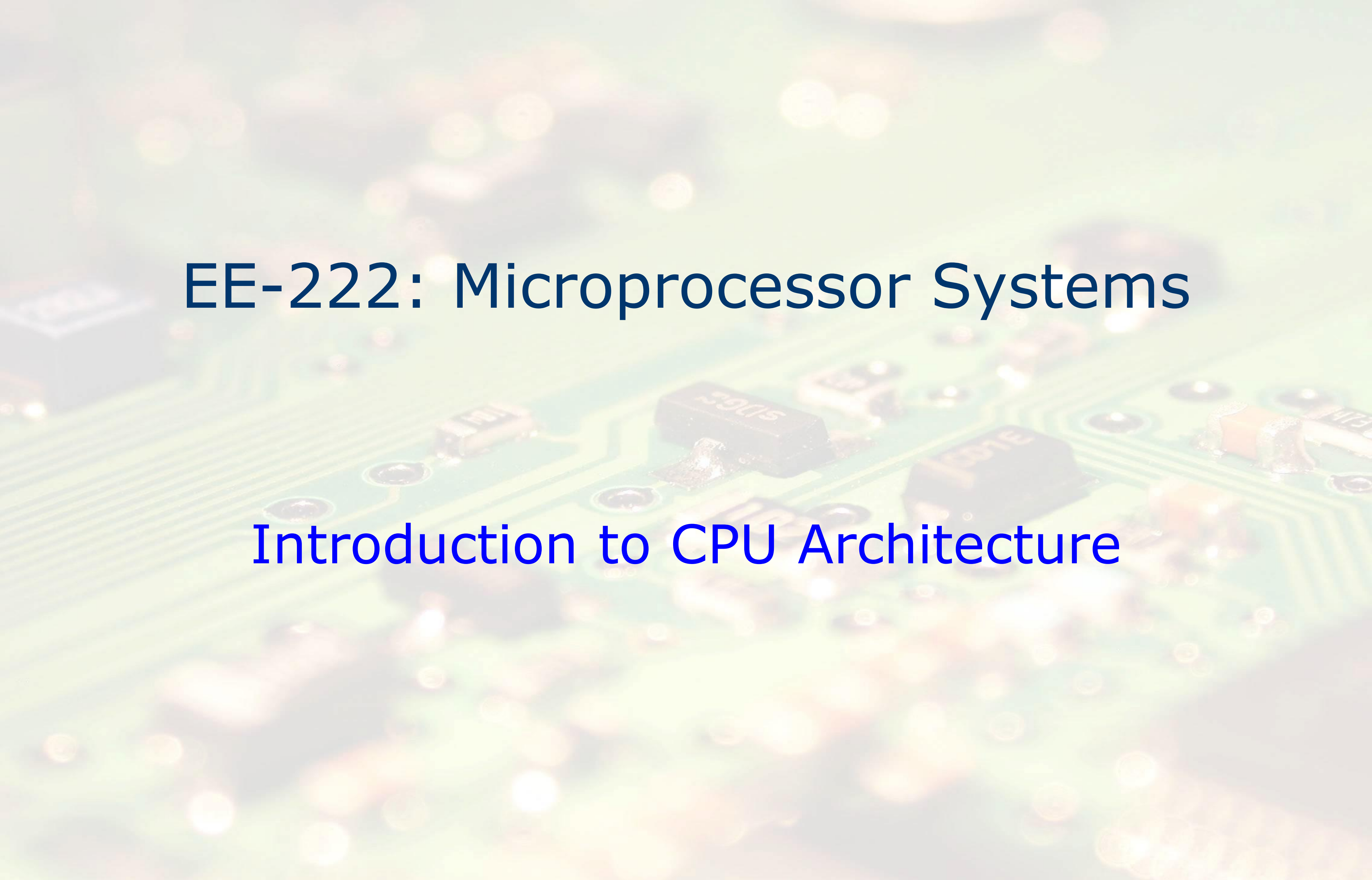


EE-222: Microprocessor Systems

Introduction to CPU Architecture

Instructor: Dr. Arbab Latif

NUST

SCHOOL OF ELECTRICAL ENGINEERING &
COMPUTER SCIENCE (SEECS)

Recall: Basic Ingredients of a μ-Processor

- Any useful microprocessor-based computing system must have:

1. A Processing Unit:

- complex circuitry that manipulates data and controls I/O devices according to the program stored in memory

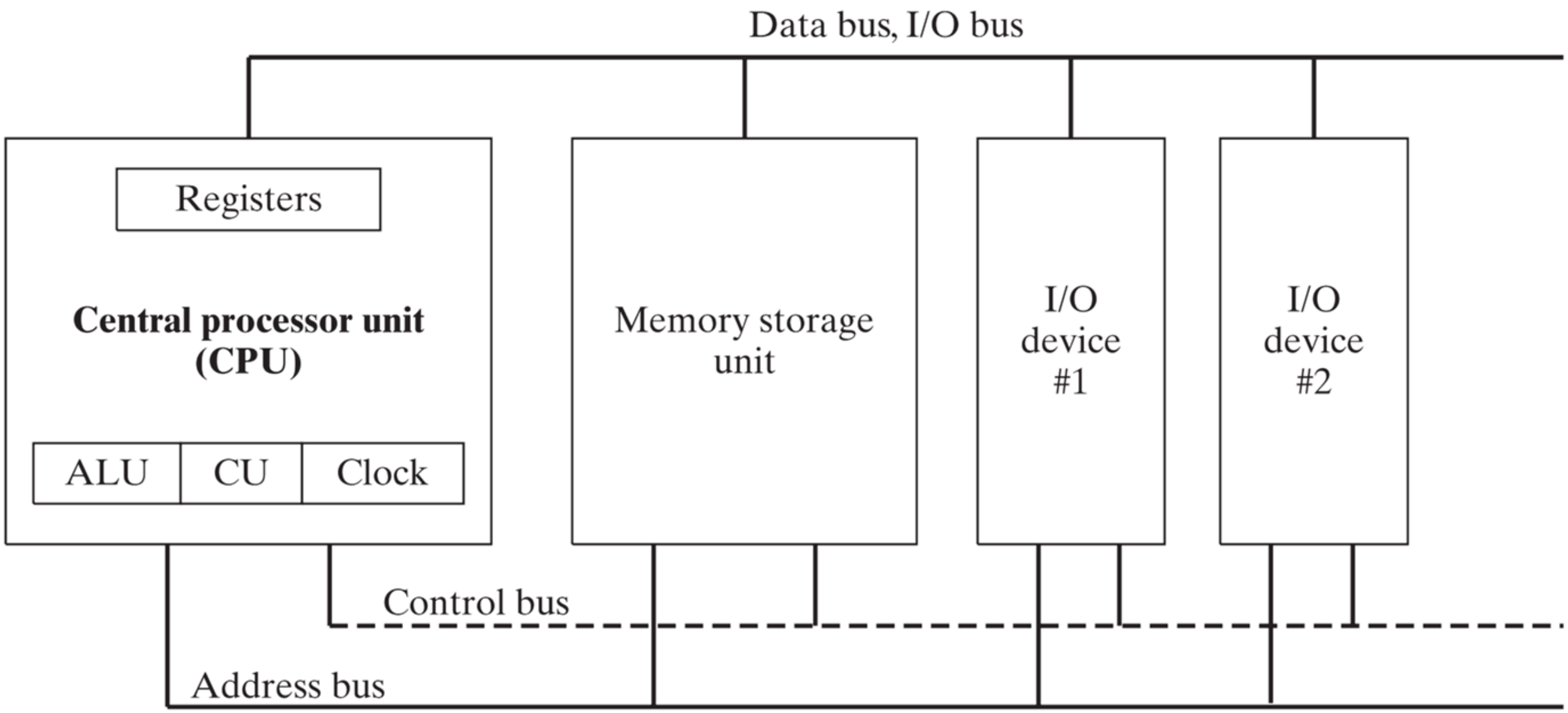
2. Memory:

- To store programs.
- To store data.

3. I/O Devices:

- To allow information to be input and output
- LEDs, 7-Segment Displays, Video Monitors, Keyboards, Motors, Relays

Block Representation of a Microprocessor



Central processing unit

Memory/Registers

Arithmetic logic unitB

I/O port & buses

Connecting the different parts:

- Connecting memory to CPU
- Connecting I/Os to CPU

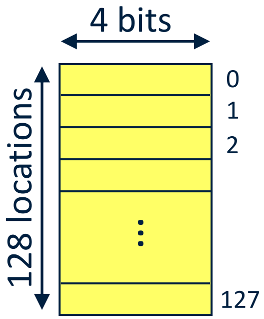
Memory

- Everything that can store, retain, and recall information.
  - E.g. hard disk, a piece of paper, etc.

Memory characteristics

- Capacity
  - The number of bits that a memory can store.
    - E.g. 128 Kbits, 256 Mbits

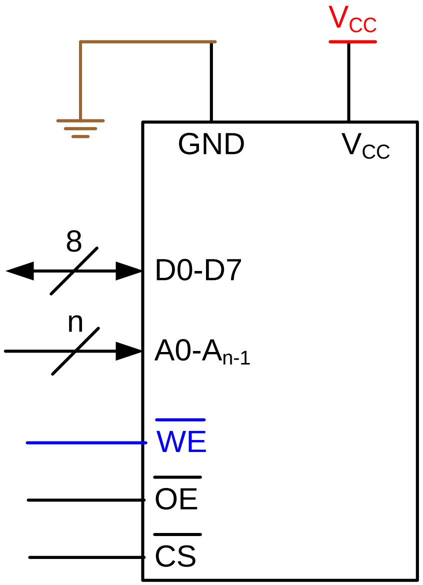
- Organization
  - How the locations are organized
    - E.g. a 128 x 4 memory has 128 locations,4 bits each



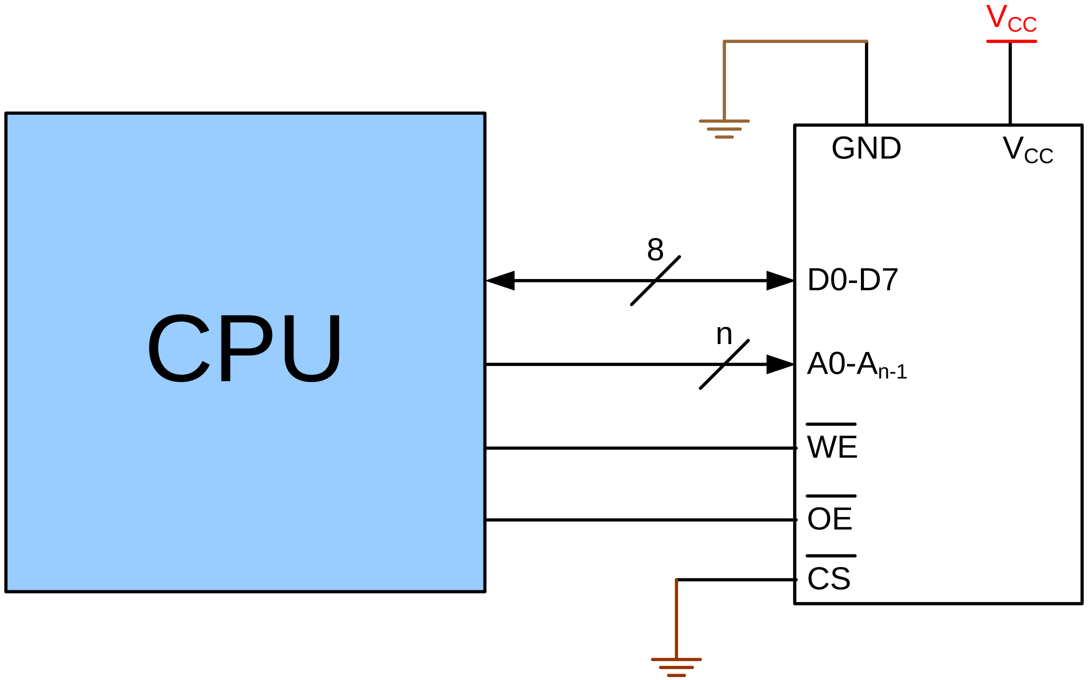
- Access time
  - How long it takes to get data from memory

Memory Pinout

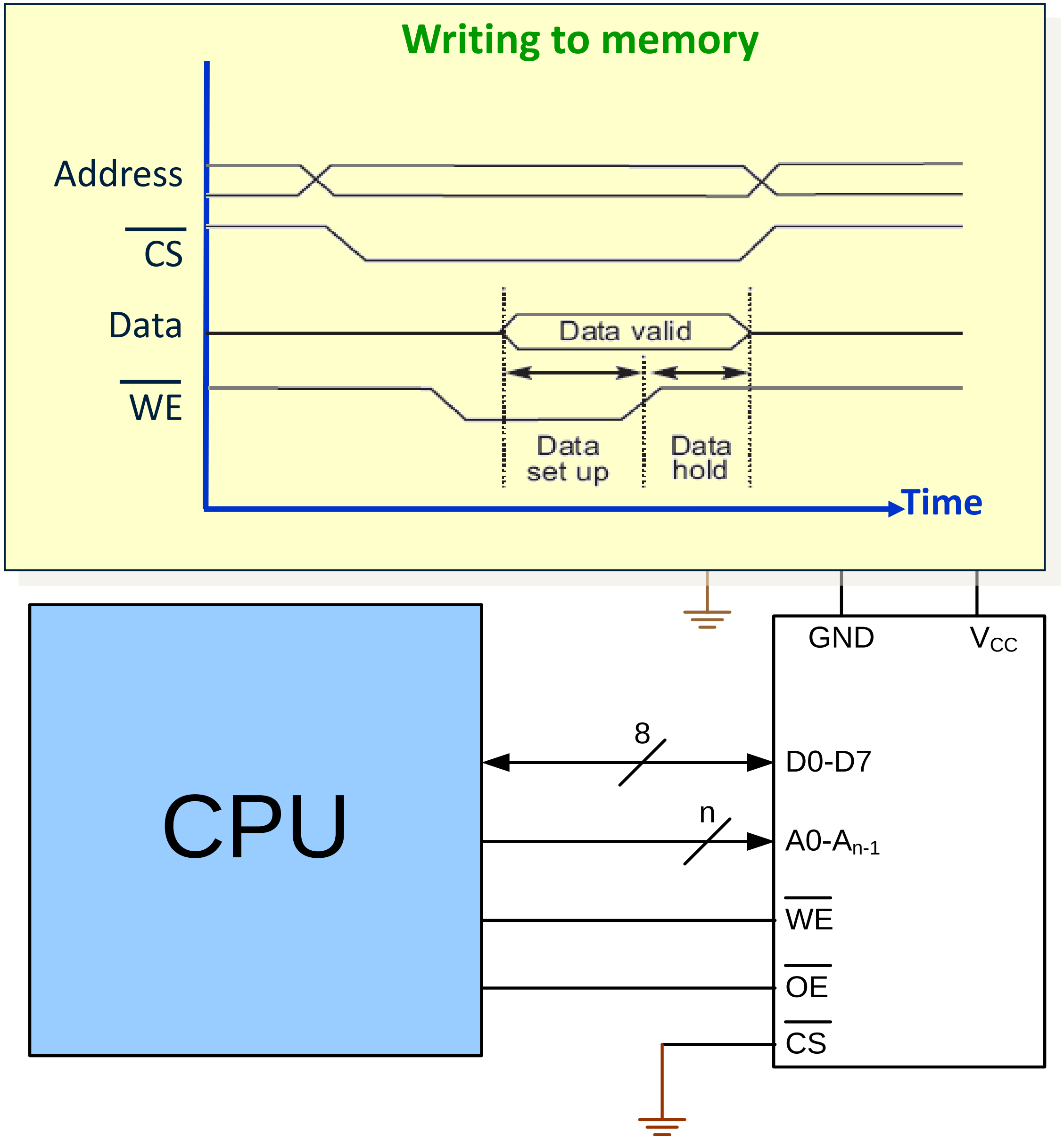
- D0-D7: are for data I/O
- An: are for address inputs
- WE: Write Enable
  - is for writing data into memory
- OE: Output Enable
  - Is for reading data out from memory
- CS: Chip Select
  - is used to select the memory chip



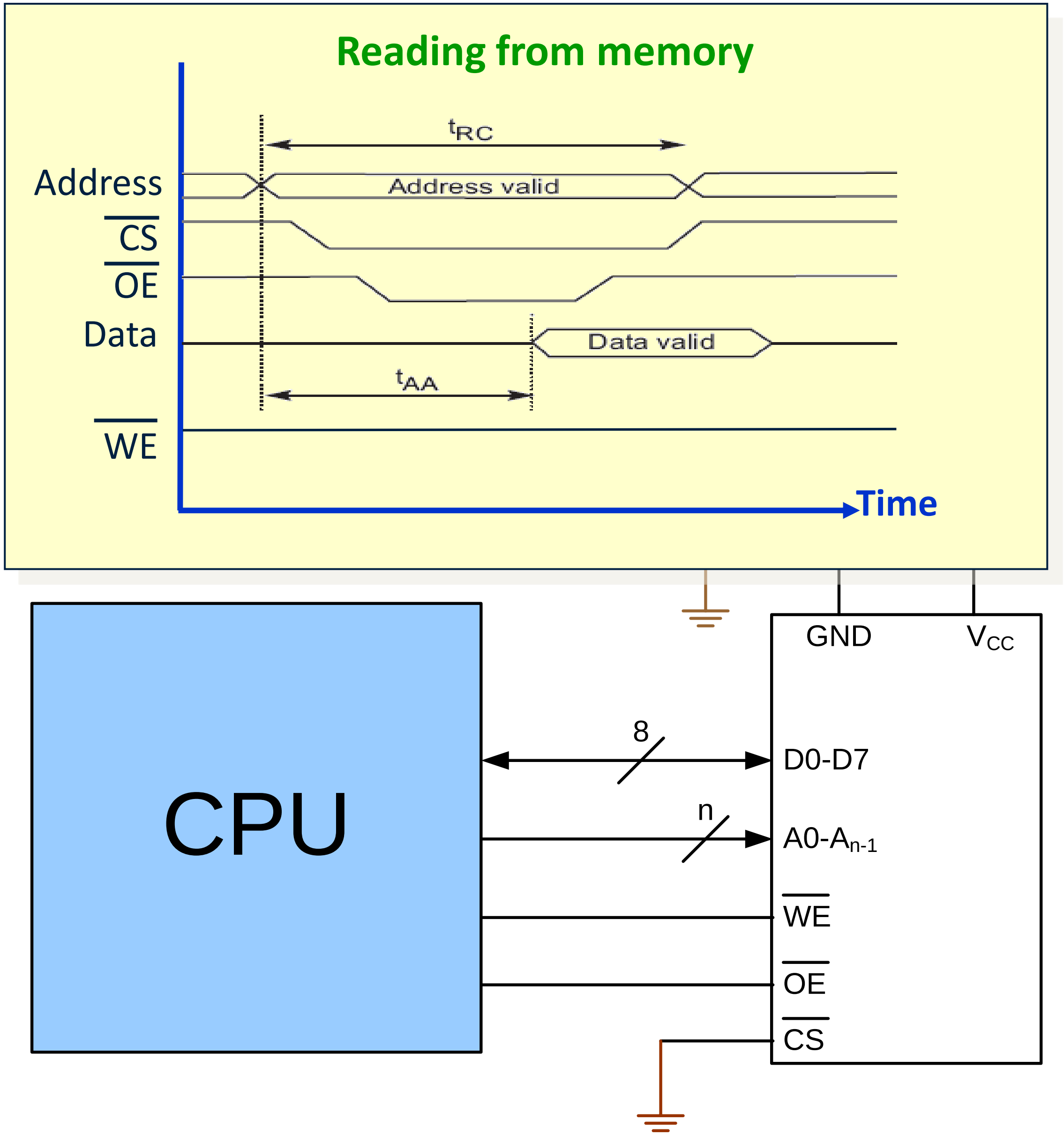
Connecting Memory to CPU



Writing to Memory

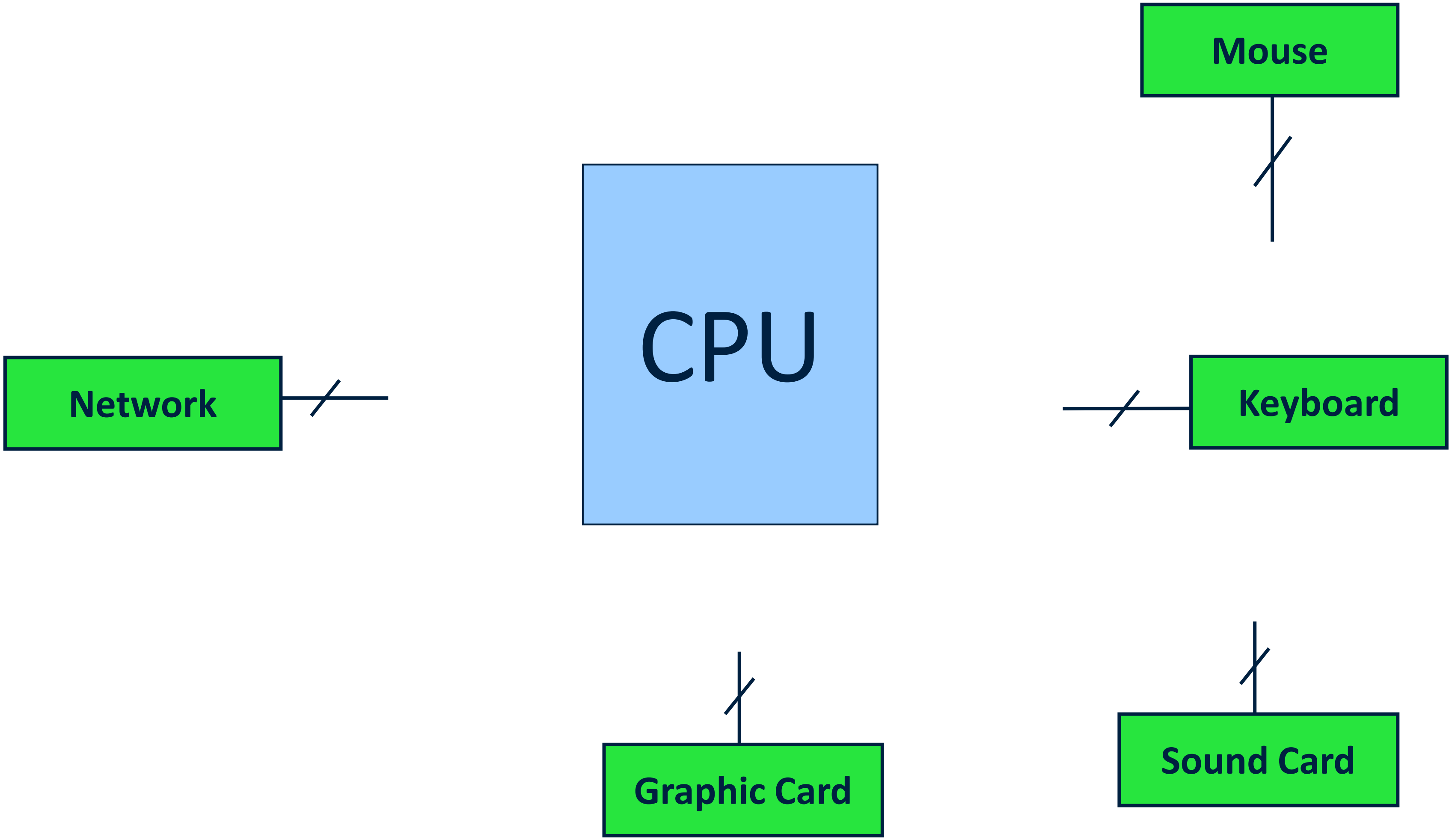


Reading from Memory

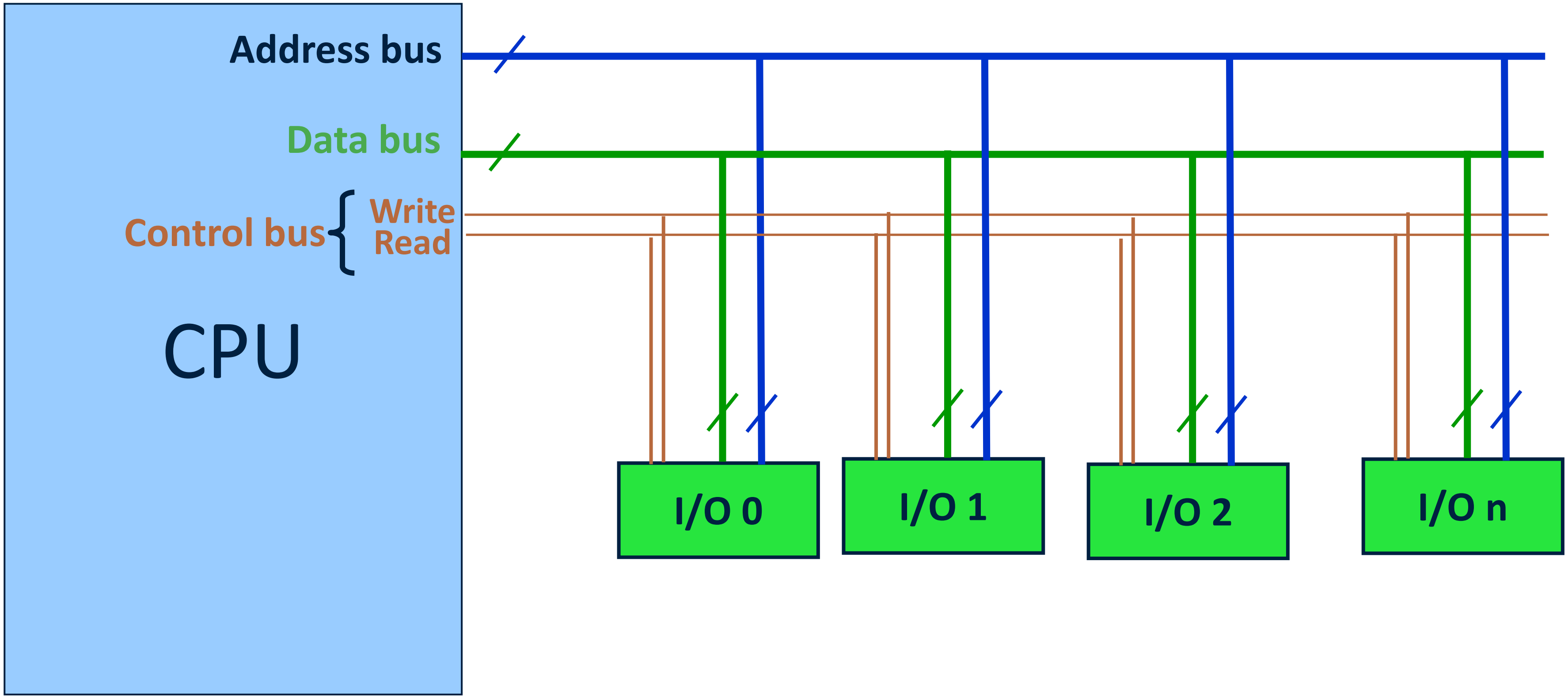


Connecting I/Os to CPU

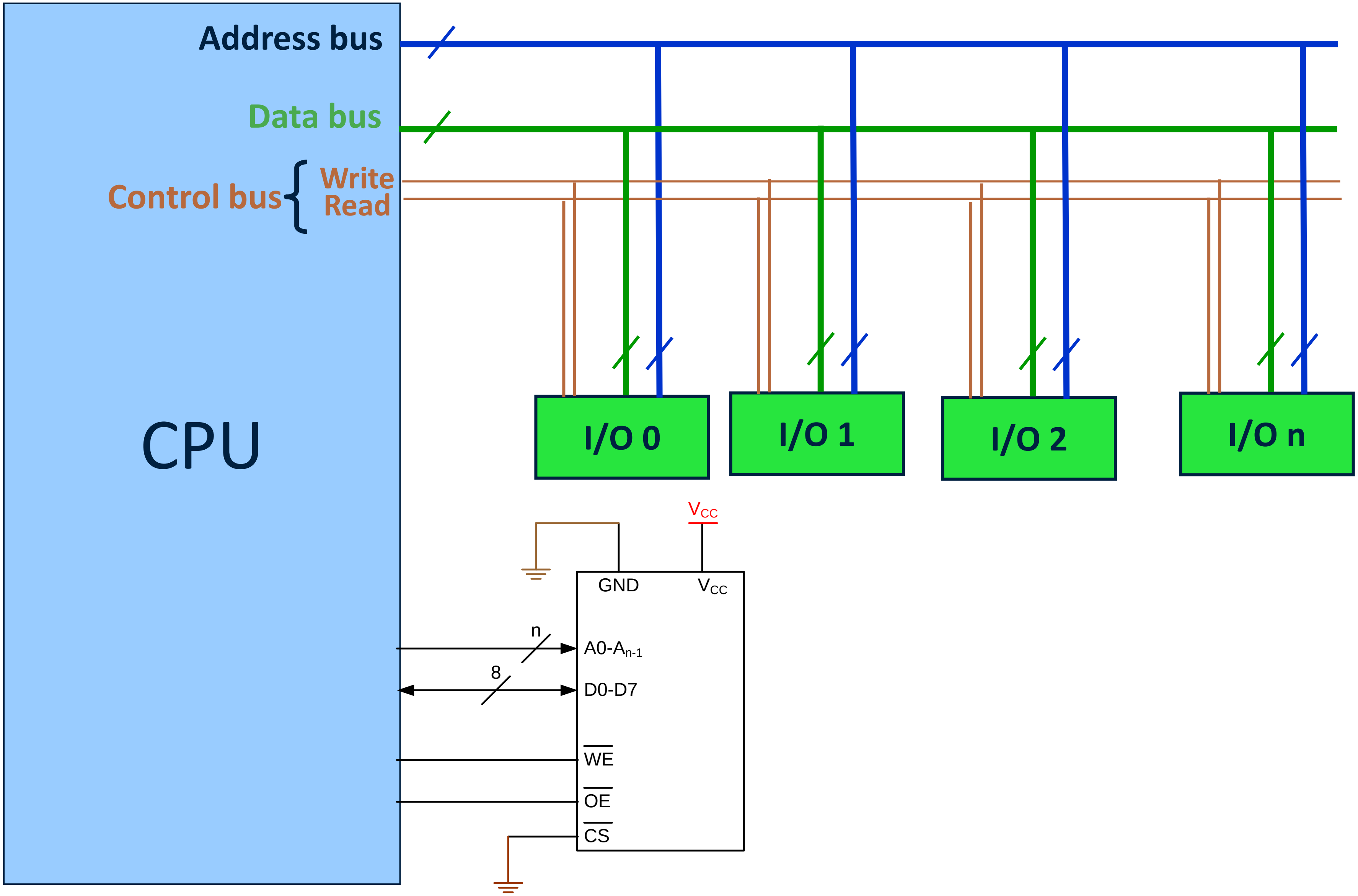
- CPU should have lots of pins!



Connecting I/Os to CPU using bus

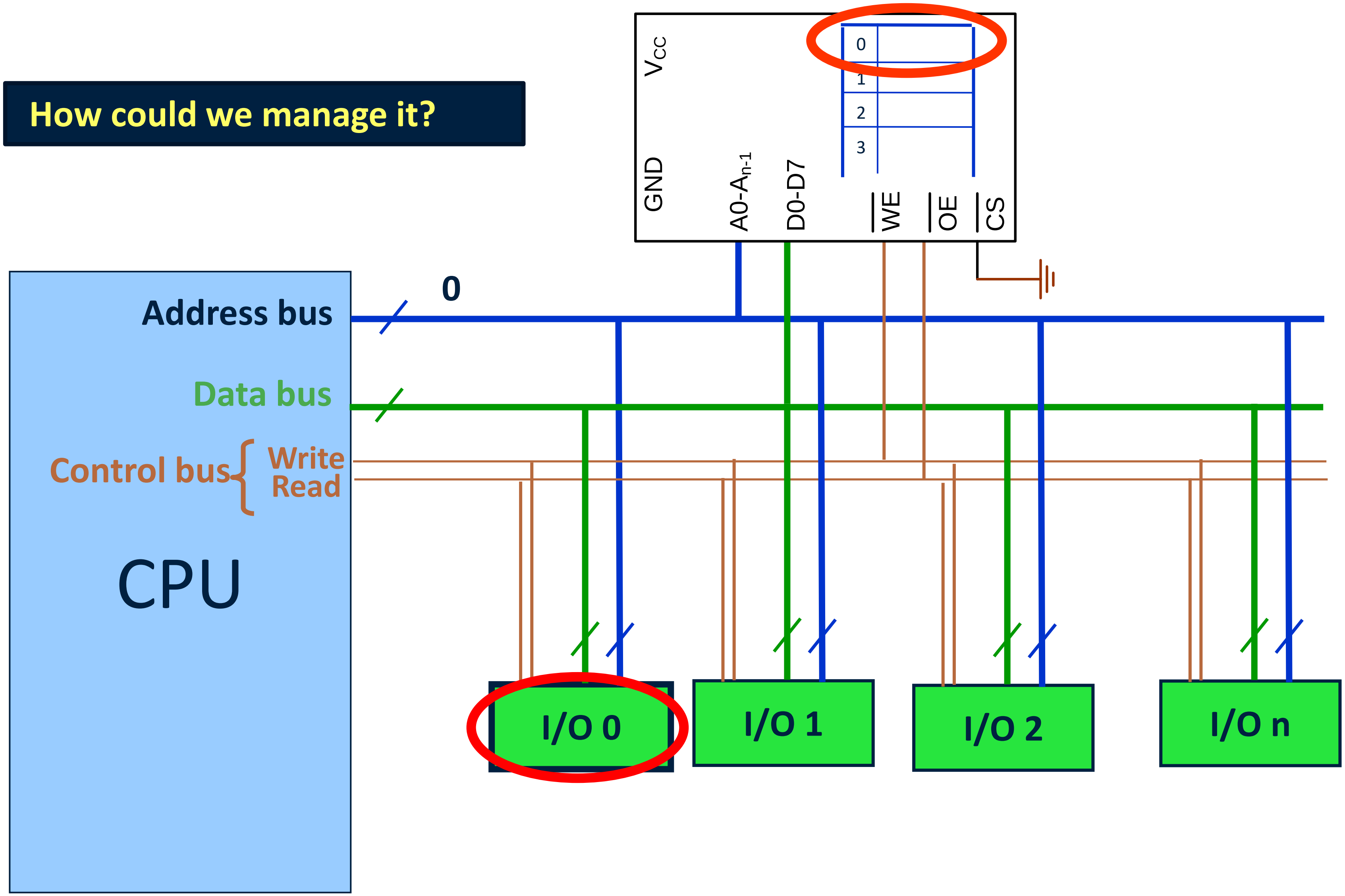


Connecting I/Os and Memory to CPU

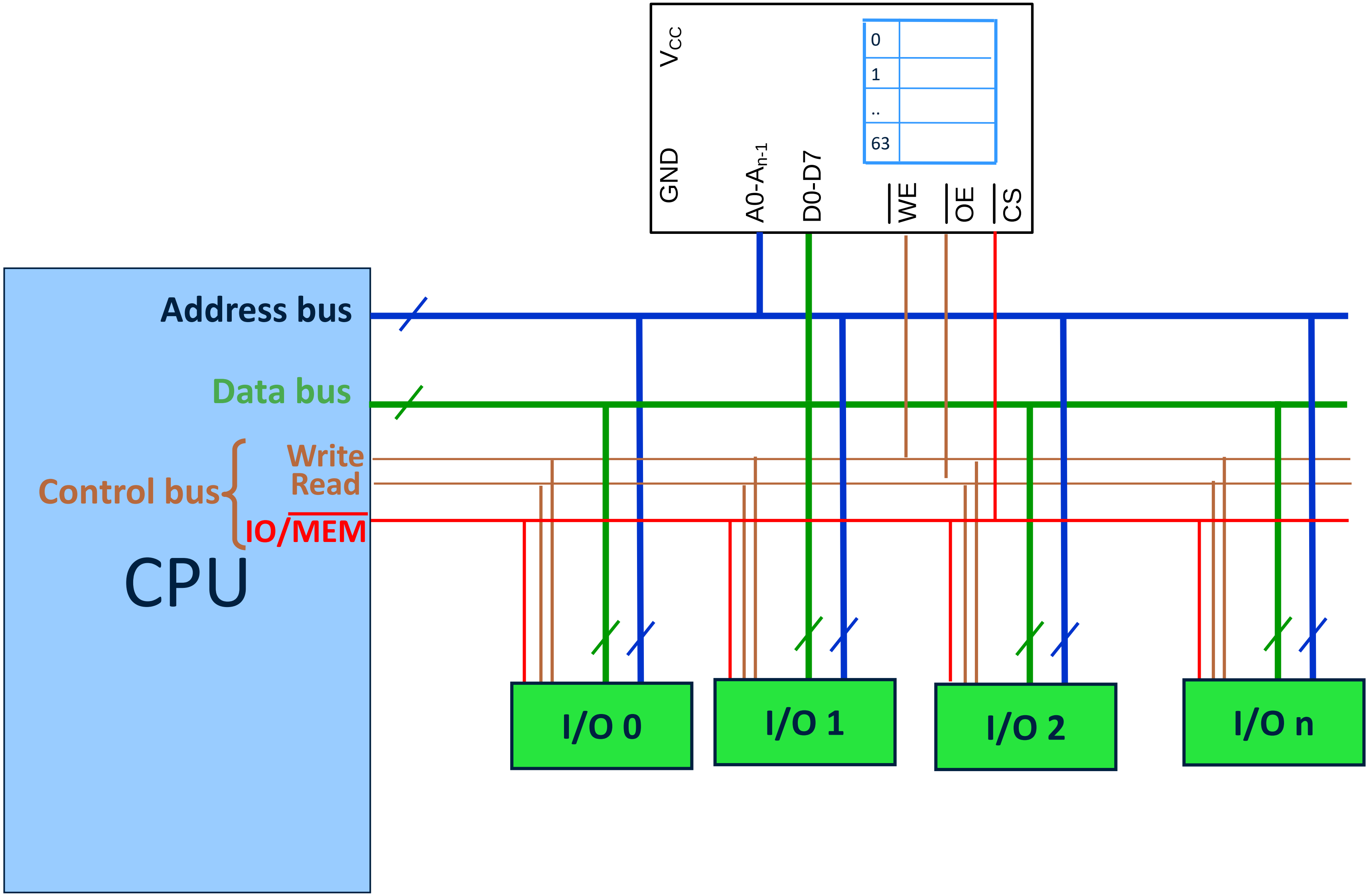


Connecting I/Os and Memory to CPU using bus

How could we manage it?

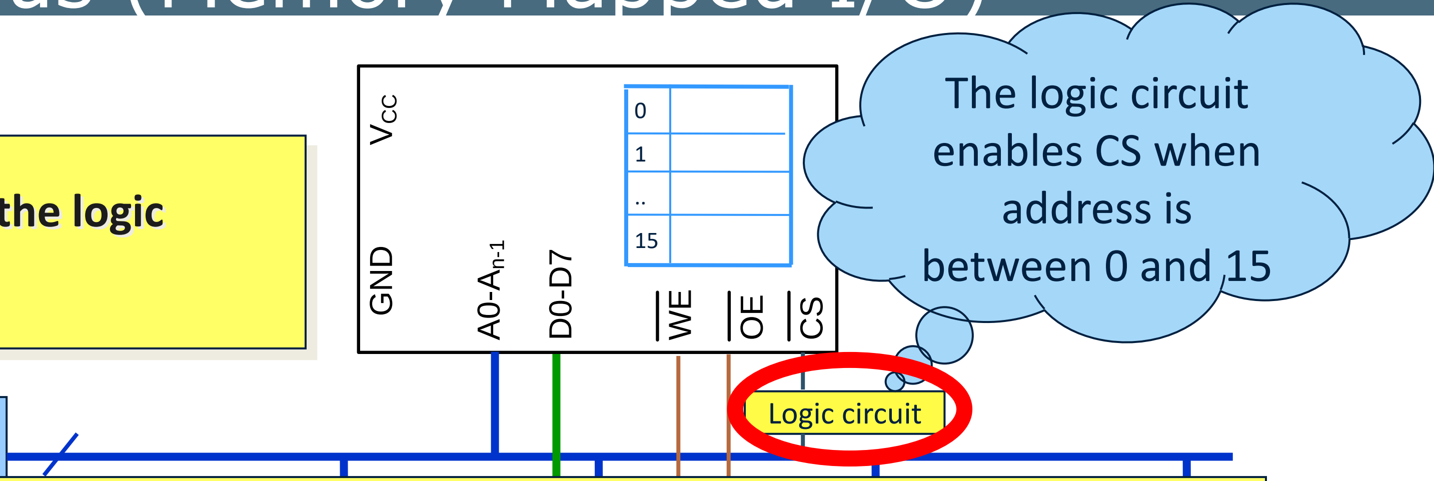


Connecting I/Os and Memory to CPU using bus (Peripheral I/O)



Connecting I/Os and Memory to CPU using bus (Memory Mapped I/O)

How could we make the logic circuit?

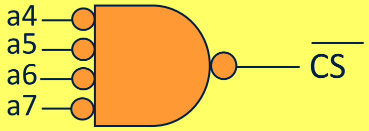

VCC
GND
A0-An-1
D0-D7
0
1
..
15
WE
OE
CS
Logic circuit
The logic circuit enables CS when address is between 0 and 15

Address bus

Solution

1. Write the address range in binary
2. Separate the fixed part of address
3. Using a NAND, design a logic circuit whose output activates when the fixed address is given to it.

	a7	a6	a5	a4	a3	a2	a1	a0
From address 0 ➔	0	0	0	0	0	0	0	0
To address15 ➔	0	0	0	0	1	1	1	1



Another example for address decoder

- Design an address decoder for address of 300H to 3FFH.

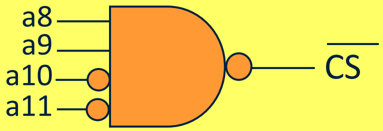
Solution

1. Write the address range in binary
2. Separate the fixed part of address
3. Design the logic circuit.

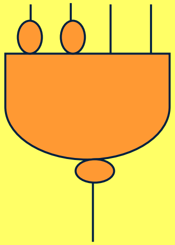
a11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0

From address 300H ➔ 0 0 1 1 0 0 0 0 0 0 0 0

To address 3FFH ➔ 0 0 1 1 1 1 1 1 1 1 1 1



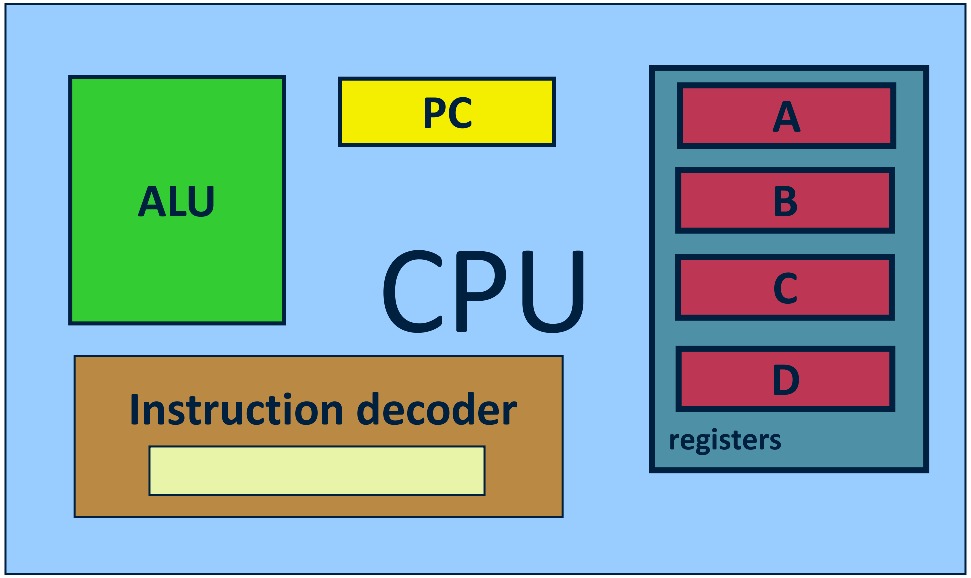
An easy way of designing



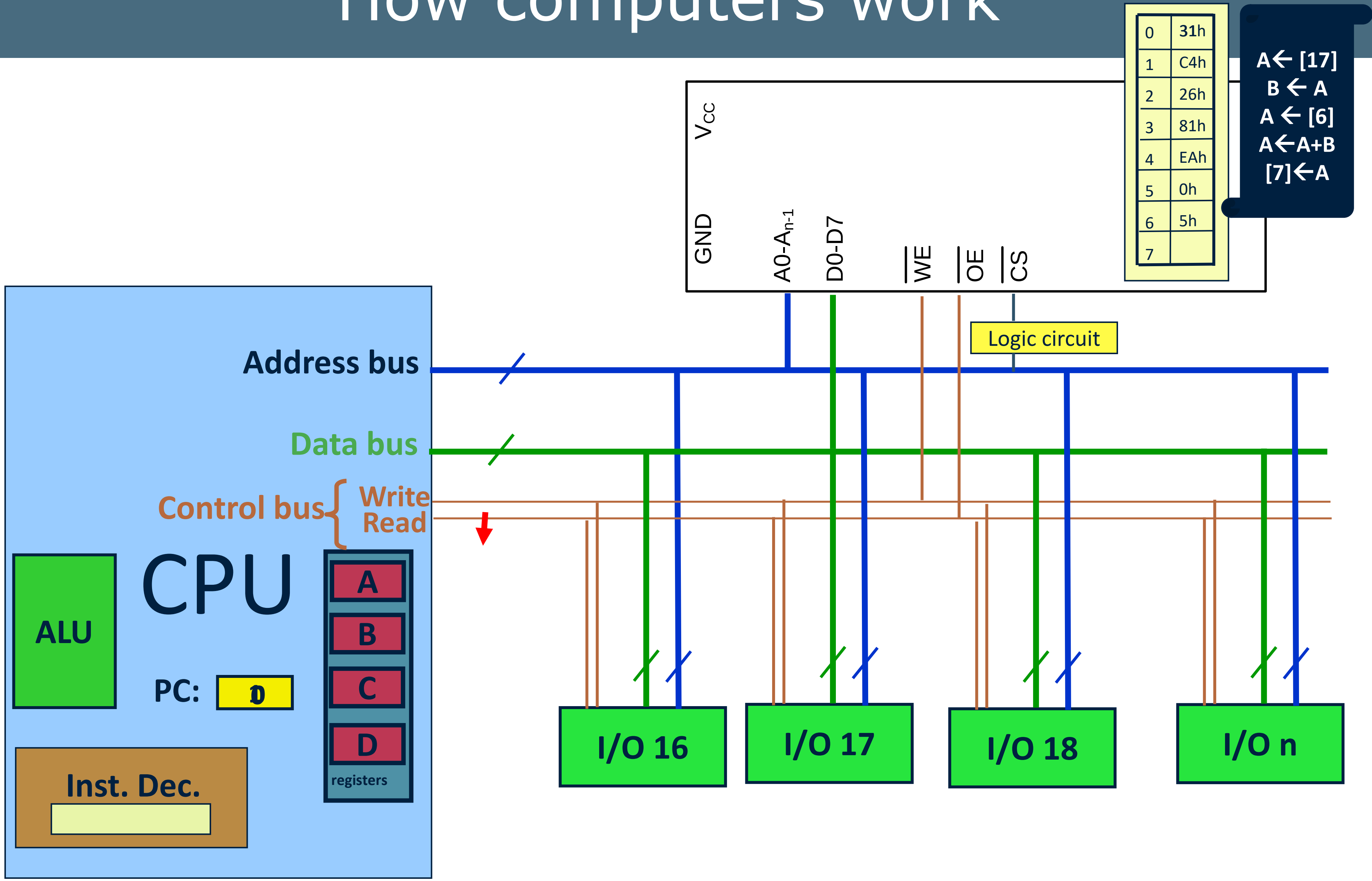
CPU Architecture

Inside the CPU

- PC (Program Counter)
- Instruction decoder
- ALU (Arithmetic Logic Unit)
- Registers



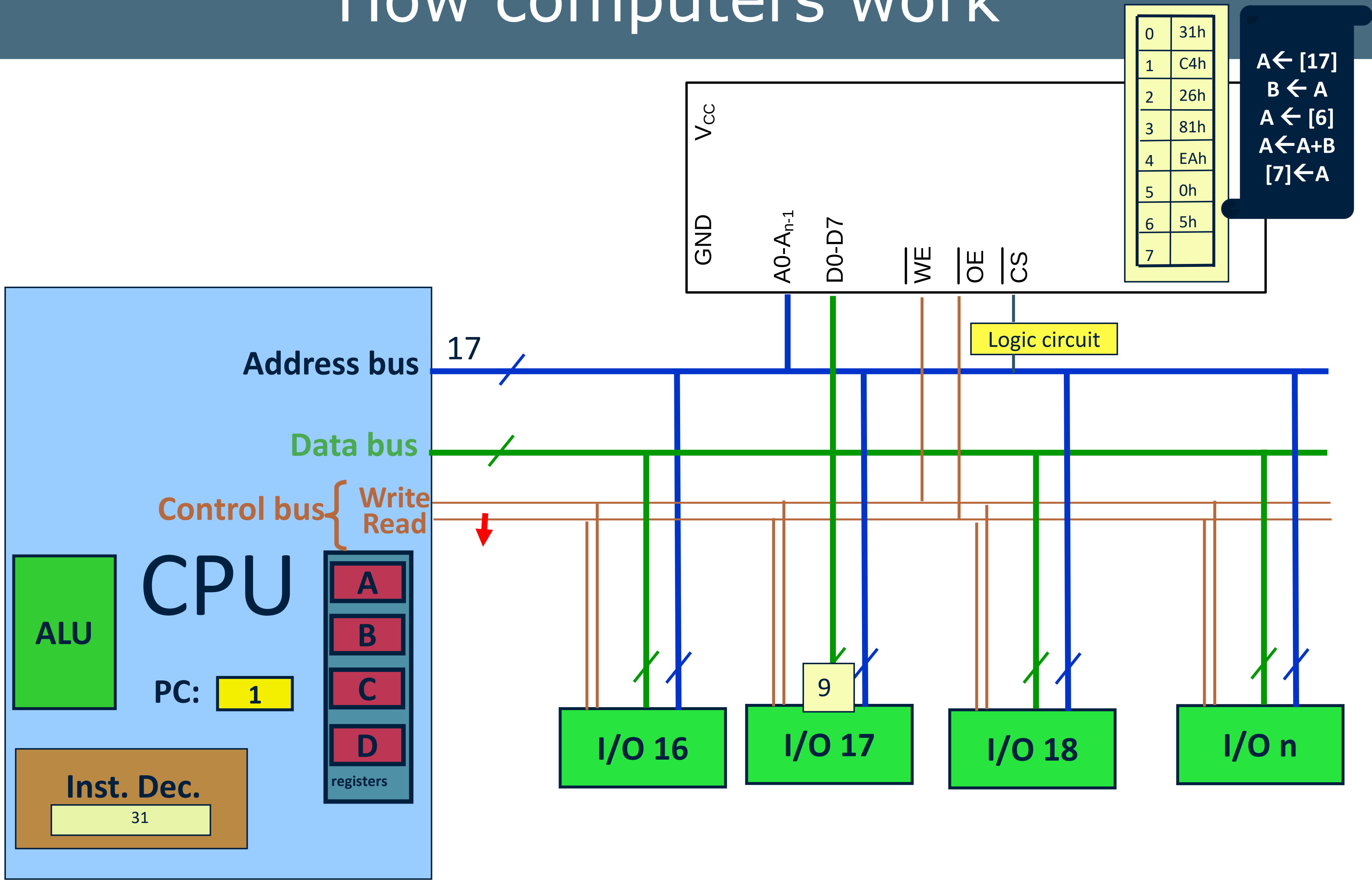
How computers work



0	31h
1	C4h
2	26h
3	81h
4	EAh
5	0h
6	5h
7	

A← [17]
B ← A
A ← [6]
A←A+B
[7]←A

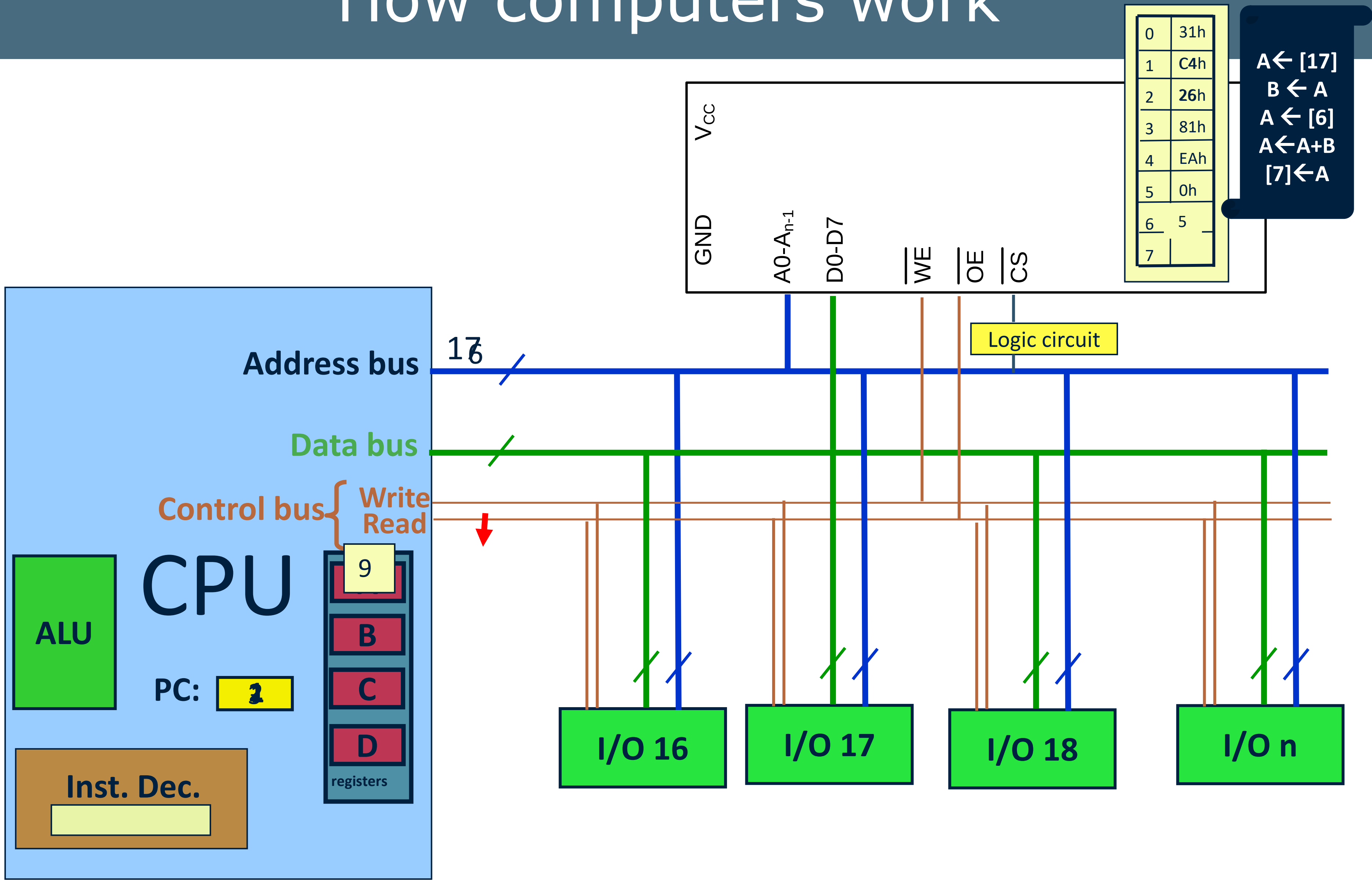
How computers work



0	31h
1	C4h
2	26h
3	81h
4	EAh
5	0h
6	5h
7	

A← [17]
B ← A
A ← [6]
A←A+B
[7]←A

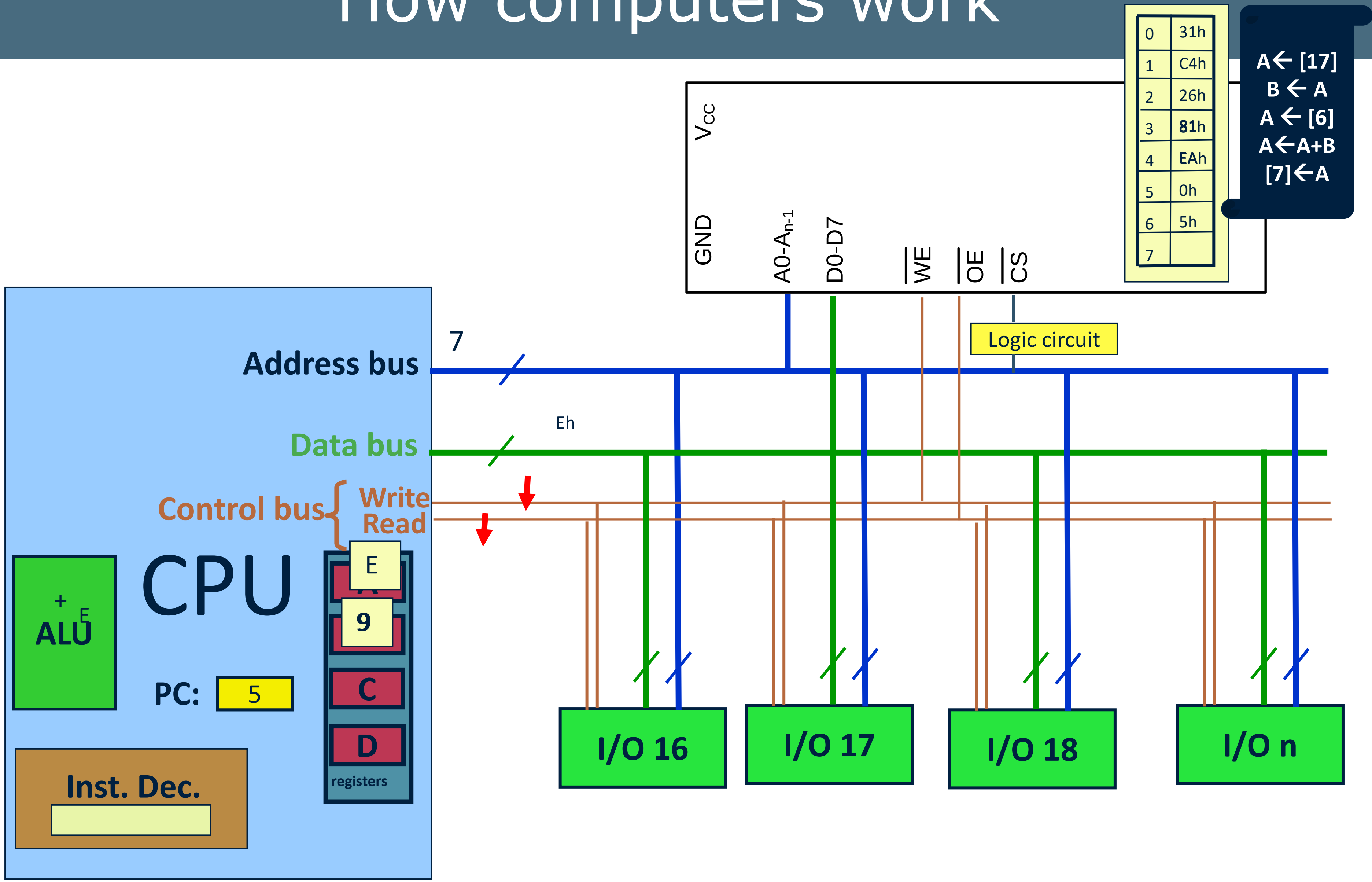
How computers work



0	31h
1	C4h
2	26h
3	81h
4	EAh
5	0h
6	5
7	

A← [17]
B ← A
A ← [6]
A←A+B
[7]←A

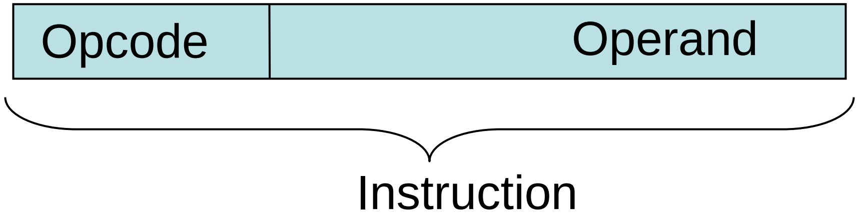
How computers work

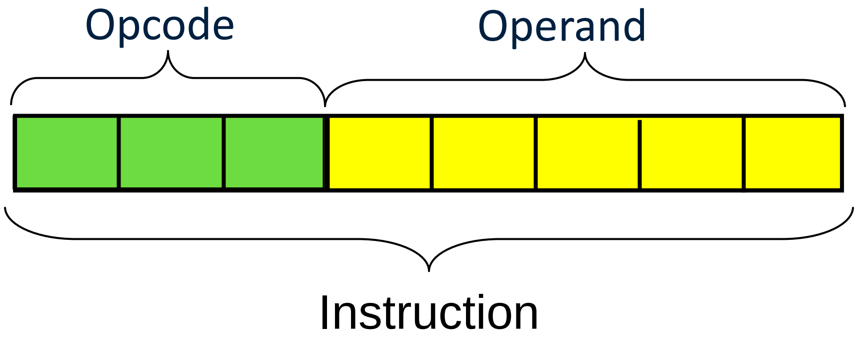


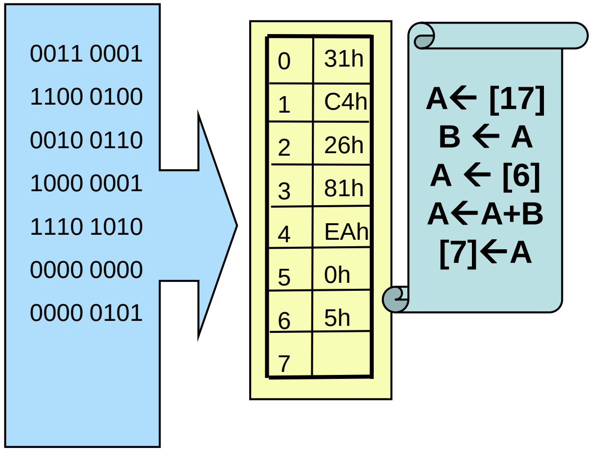
0	31h
1	C4h
2	26h
3	**81**h
4	EAh
5	0h
6	5h
7	

A← [17]
B ← A
A ← [6]
A←A+B
[7]←A

How Instruction Decoder works





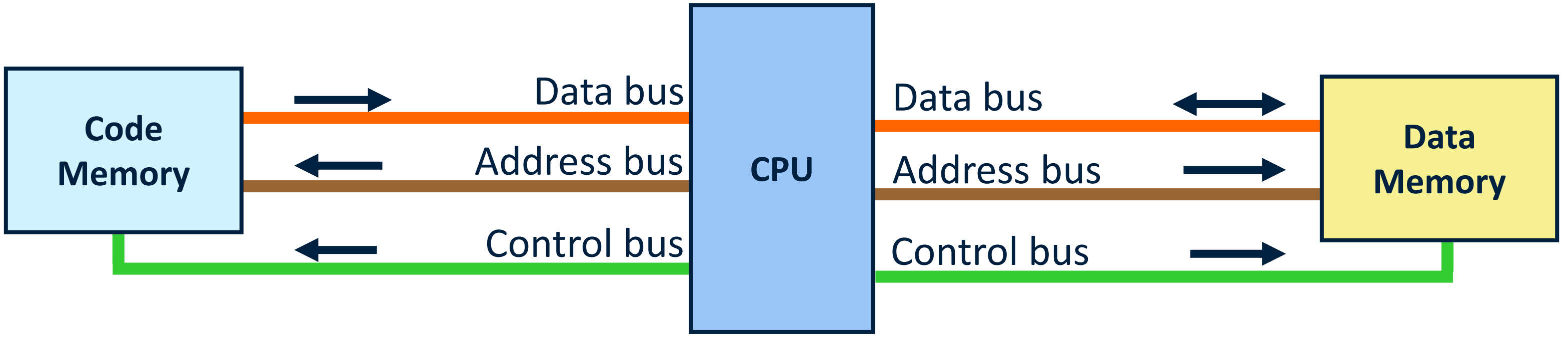


0	31h
1	C4h
2	26h
3	81h
4	EAh
5	0h
6	5h
7	

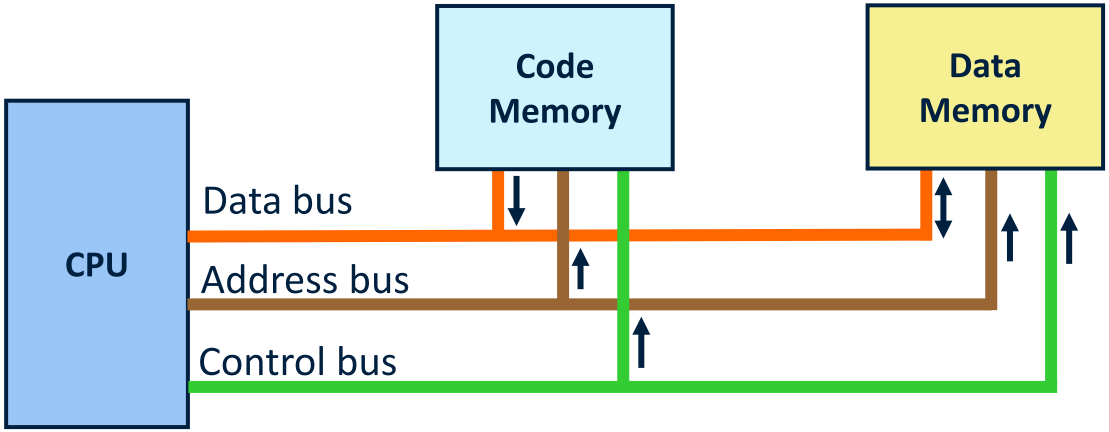
A← [17]
B ← A
A ← [6]
A←A+B
[7]←A

Operation Code	Meaning
000	A ← x
001	A ← [x]
010	A ← A – register (x)
011	A ← A + x
100	A ← A + register (x)
101	A ← A – x
110	Register (x_H) ← Register (x_L)
111	[x] ← A

Von Neumann vs. Harvard architecture

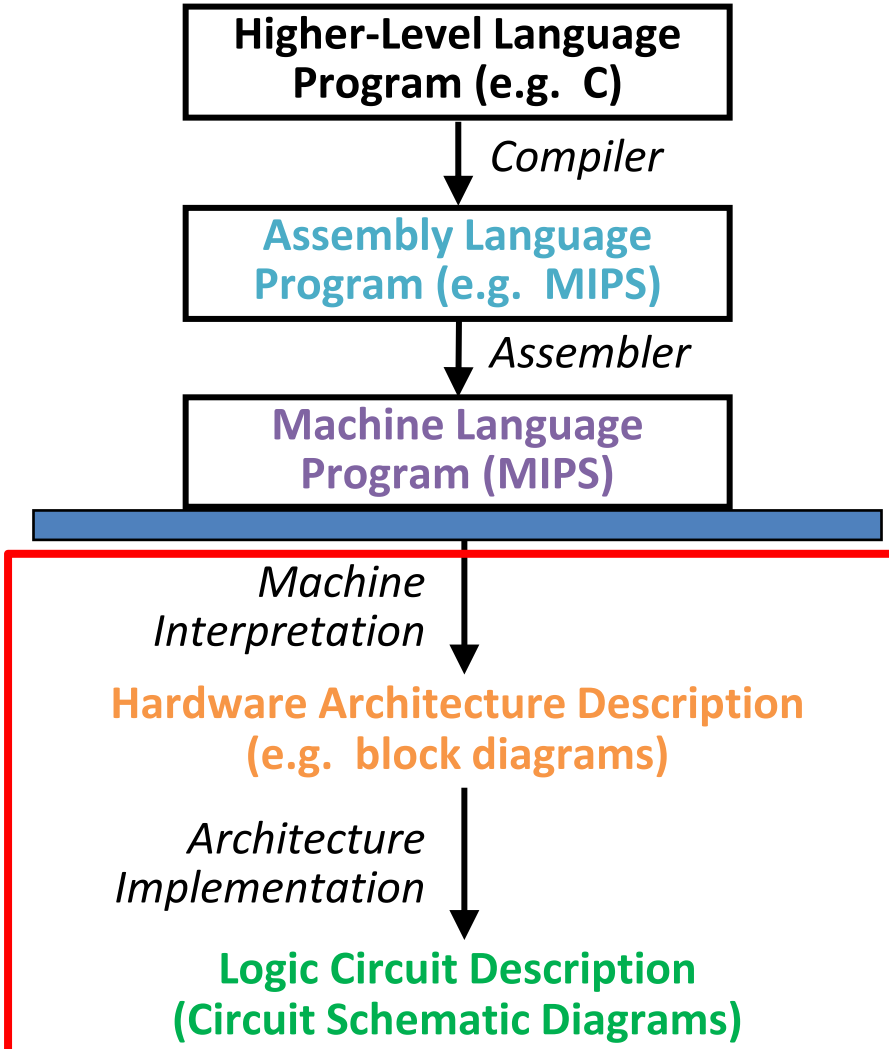


- Harvard architecture



- Von Neumann architecture

So Far: In General



Higher-Level Language Program (e.g. C)

Compiler

Assembly Language Program (e.g. MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g. block diagrams)

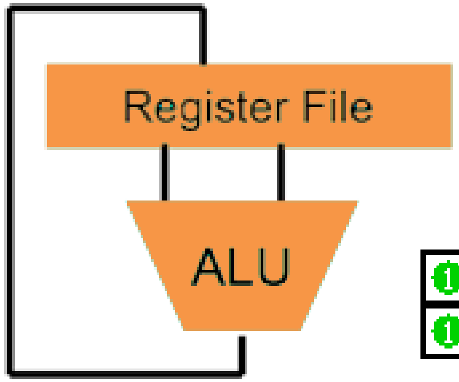
Architecture Implementation

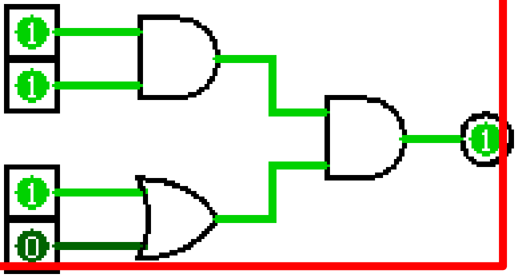
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

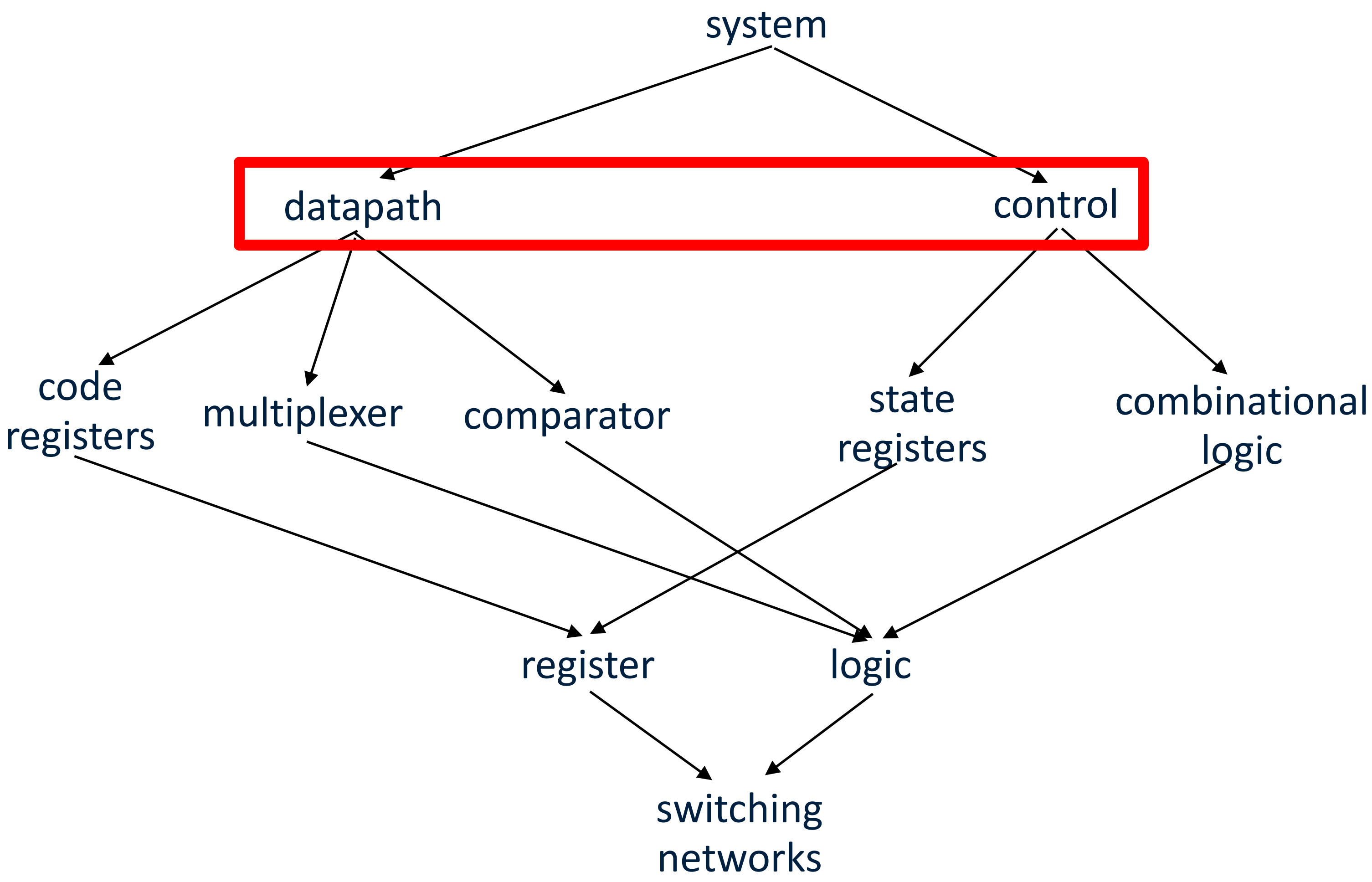
```
lw    $t0, 0($2)
lw    $t1, 4($2)
sw    $t1, 0($2)
sw    $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



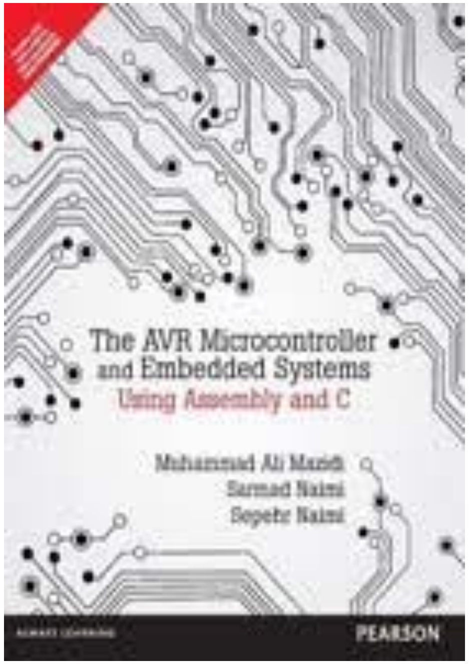


Hardware Design Hierarchy



Reading Assignment

- The AVR Microcontroller and Embedded Systems: Using Assembly and C by Mazidi et al., Prentice Hall
  - Chapter-1 -> section 3 and 4



THANK YOU