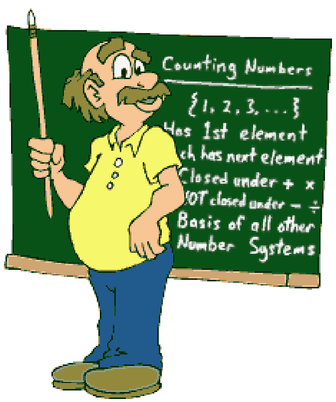
Ingeniørhøjskolen i Århus

IECA

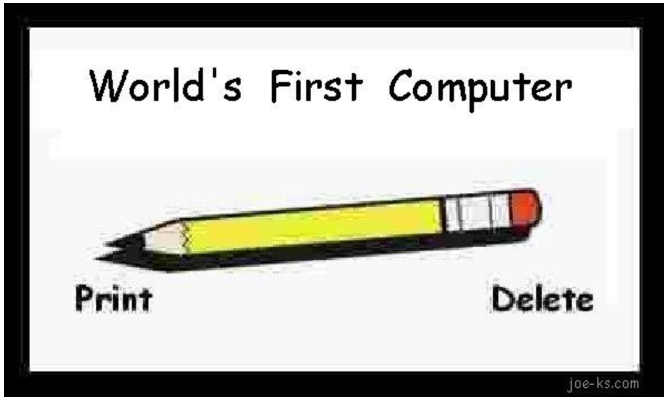
Embedded Computer Architecture

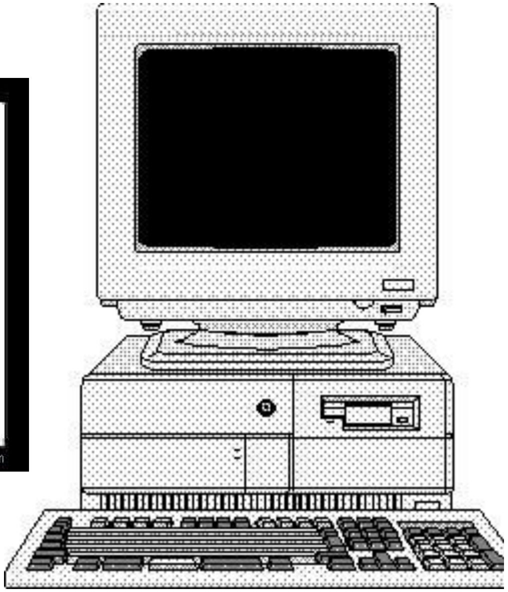
Lesson 2: Introduction to computers



Version: 23-11-2014, Henning Hargaard

What is a computer ?

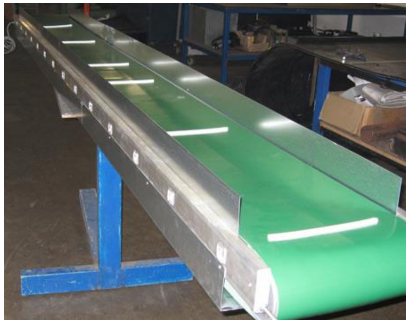




• Basicly a ”computer” is simply an electronic device capable of doing some sort of calculations.

• A PC is a ”general purpose” computer.
Many electronic devices contains a ”special purpose” computer (typically a microcontroller).

Microcontroller = ”Very small computer”



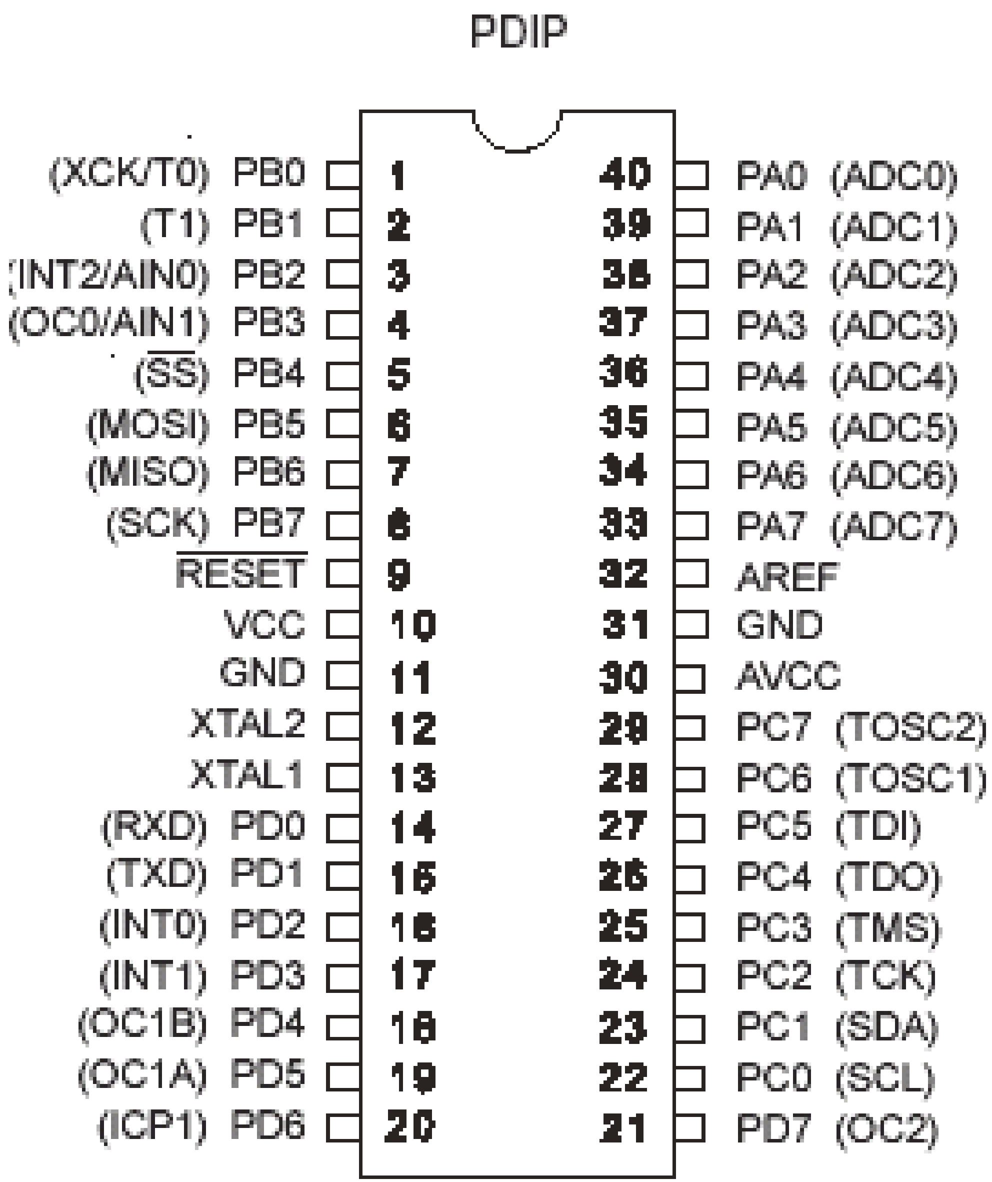
What's computer architecture ?

The way a computer is internally build / organized (HW).

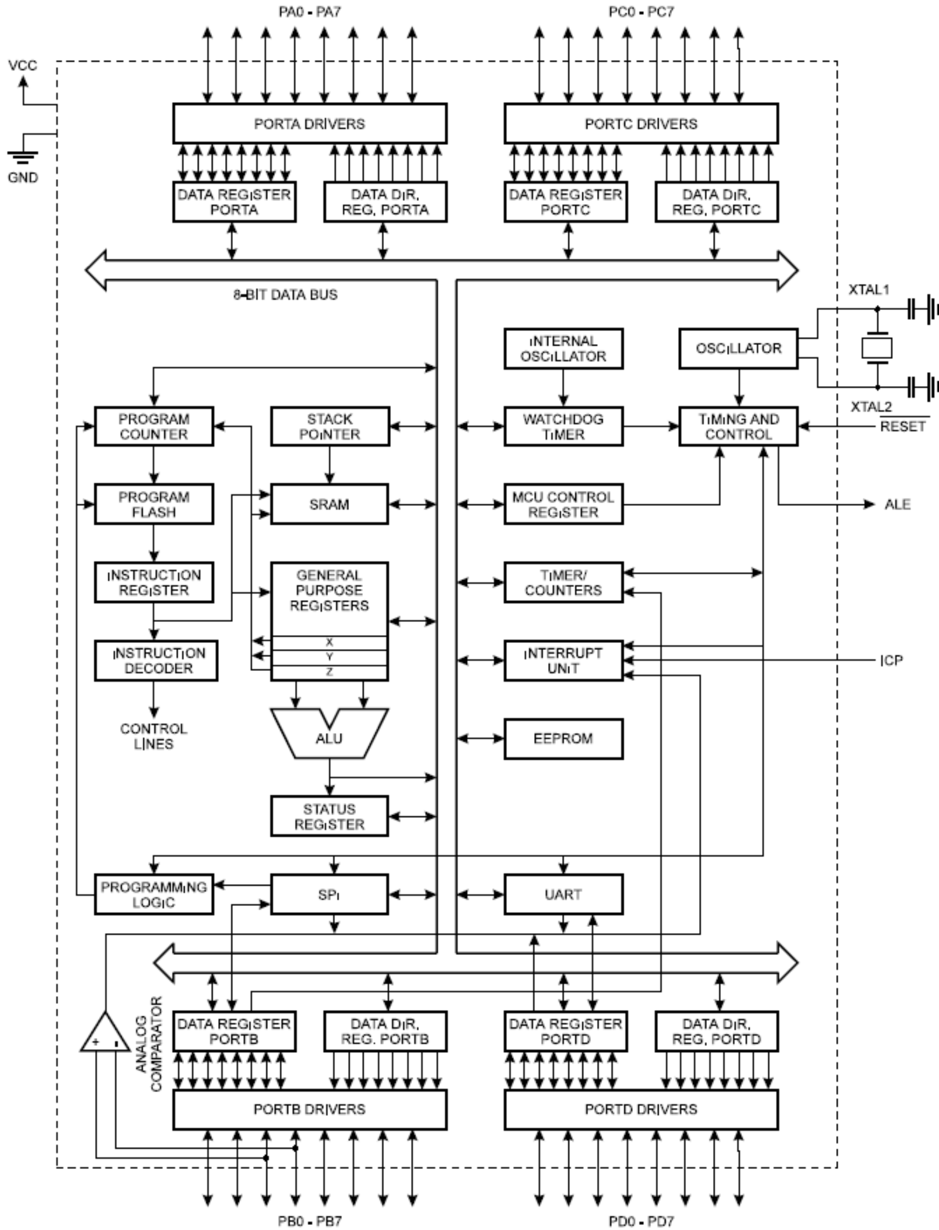
What must we have to form a computer?

- Unit for calculations (ALU) and a status register.
- Memory (program and data).
- Program counter.
- Instruction decoder.
- Input and output –units.

Atmel AVR: Single chip microcontroller



AVR Internal Architecture



Program (SW)

- The sequence of instructions that describe what a computer should do (functionality).
- The program must "downloaded" or "installed" on your computer before it can be executed.
- A PC is "general purpose", and typically perform many different programs for different times (depending on the purpose).
- A microcontroller will typically manage a device with a specific functionality ("special purpose") and the program is only installed once.

Machine Code

- Basically, a microcontroller or microprocessor <u>only</u> understands and runs machine code, being binary numerical codes.

- The codes run sequentially ("in turn"), primitively controlling the internal hardware of the computer.

- Often a symbolic language for the machine codes are used (called ASSEMBLY language).

- Symbolic machine code are translated to pure machine code using an assembler (a program running on a PC).

Example:

```
LDS  R26,_led_status
CPI  R26,LOW(0xFF)
BRNE _0x4
```

High level languages

• It is difficult and requires deep insight into the computer's internal architecture to be able to write assembly code.

• Most programs are therefore written a higher abstraction level (high level language).

• Some examples are: Pascal, Basic, Java, C++, C, C#.

• All high level programs must be translated to mascine code in order to be executable by the computer.
This is done using a program called a COMPILER.

High level program example :

```
if ( button_pressed() )
   motor_start();
else
   motor_stop();
```

C contra Assembly

- C :

```
if (led_status==0xff)
```

- Assembly :

```
LDS  R26,_led_status
CPI  R26,LOW(0xFF)
BRNE _0x4
LDI  R30,LOW(254)
STS  _led_status,R30
```

Advantages / disadvantages ?

Programming in C

- C is the ”predecessor” for C++ (simpler than C++).
  C is very often used for programming microcontrollers.

Example:

```
#include <avr/io.h>

int main()
{
unsigned char i = 0;
  DDRA = 0xFF; //port A as output
  DDRB = 0xFF; //port B as output
  DDRC = 0xFF; //port C as output
  PORTA = 0xAA;
  while (1)
  {
    PORTC = PORTC ^ 0x01; //toggle PORTC.0
    PORTB = i;
    i++;
  }
  return 0;
}
```

Compiler

• The compiler ”translates” our C code to a file, that contains all the machine codes for the program.

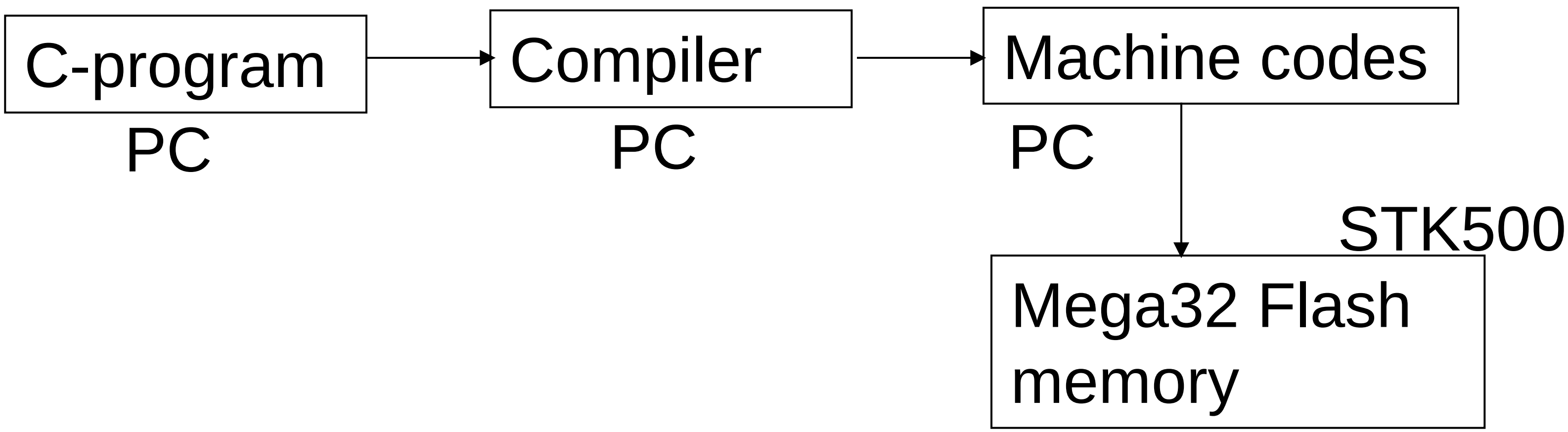
Different types of computers have different mascine codes.

• The compiler itself is a program, typically running at a PC.

Program Download

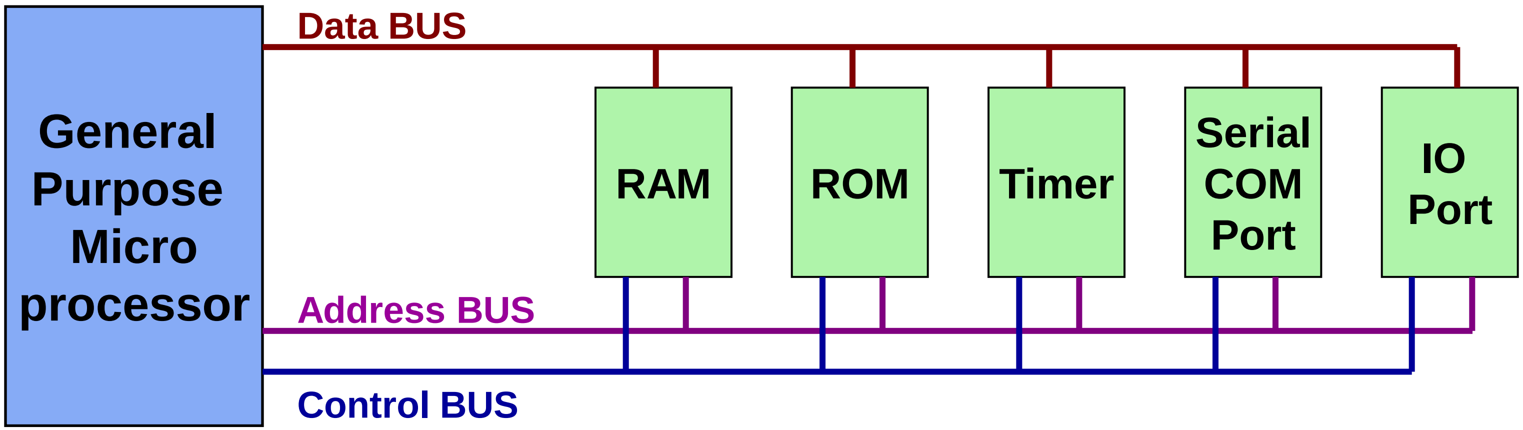
- After having compiled the program (at the PC), the machine codes of the program must be transferred (”downloaded”) to ”our computer” (called the ”target”).
- In our case we use a COM port and a serial cable.
- Our ”target” is the microcontroller Mega32.

It has a flash program memory, being non-volatile (the program is preserved, even when we switch of the power).

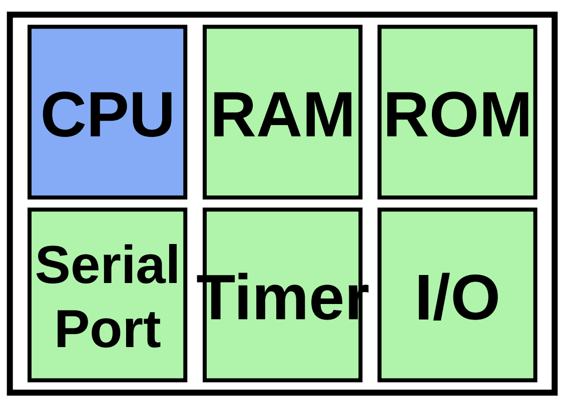


General Purpose Microprocessors vs. Microcontrollers

- General Purpose Microprocessors

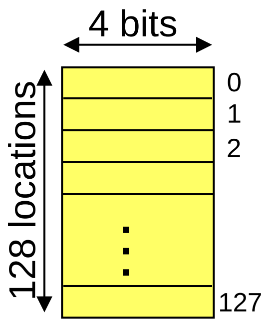


- Microcontrollers



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

Semiconductor memories

- **ROM**
  - Mask ROM
  - PROM (Programmable ROM)
  - **EPROM** (Erasable PROM)
  - **EEPROM** (Electronic Erasable PROM)
  - **Flash Memory**

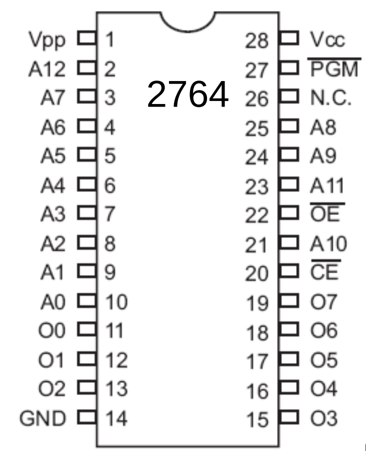
- RAM
  - **SRAM** (Static RAM)
  - **DRAM** (Dynamic RAM)
  - NV-RAM (Nonvolatile RAM)

EPROM (Erasable Programmable ROM)

- UV-EPROM
  - You can shine ultraviolet (UV) radiation to erase it
  - Erasing takes up to 20 minutes
  - The entire contents of ROM are erased

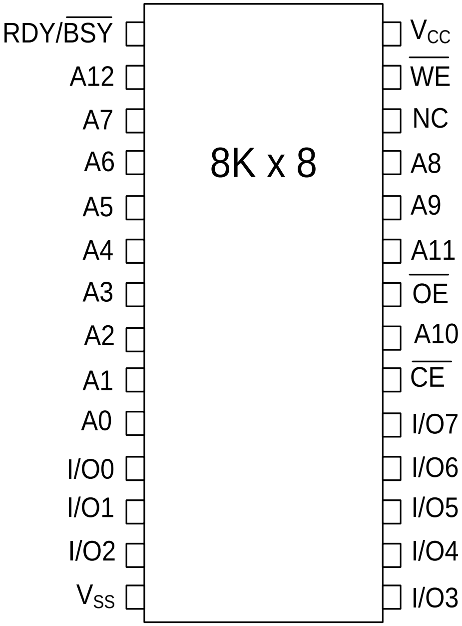
Table 0-5: Some UV-EPROM Chips

Part #	Capacity	Org.	Access	Pins	V_{PP}
2716	16K	2K × 8	450 ns	24	25 V
2732	32K	4K × 8	450 ns	24	25 V
2732A-20	32K	4K × 8	200 ns	24	21 V
27C32-1	32K	4K × 8	450 ns	24	12.5 V CMOS
2764-20	64K	8K × 8	200 ns	28	21 V
2764A-20	64K	8K × 8	200 ns	28	12.5 V
27C64-12	64K	8K × 8	120 ns	28	12.5 V CMOS



Memory\ROM\EEPROM (Electrically Erasable Programmable ROM)

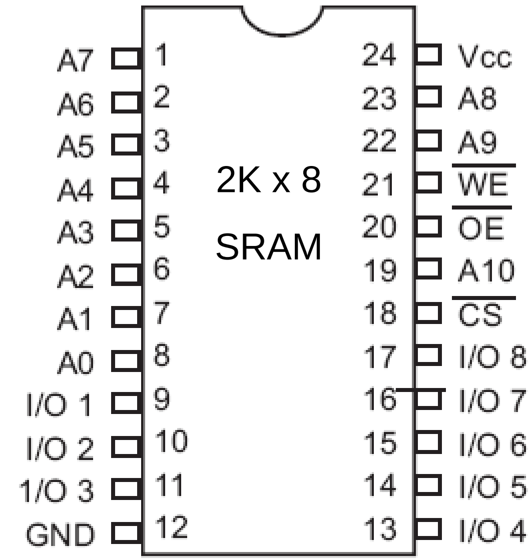
- Erased Electrically
  - Erased instantly
  - Each byte can be erased separately



Part No.	Capacity	Org.	Speed	Pins	V_{PP}
2816A-25	16K	2K × 8	250 ns	24	5 V
2864A	64K	8K × 8	250 ns	28	5 V
28C64A-25	64K	8K × 8	250 ns	28	5 V CMOS
28C256-15	256K	32K × 8	150 ns	28	5 V
28C256-25	256K	32K × 8	250 ns	28	5 V CMOS

SRAM (Static RAM)

- Made of flip-flops (Transistors)
- Advantages:
  - Faster
  - No need for refreshing
- Disadvantages:
  - High power consumption
  - Expensive



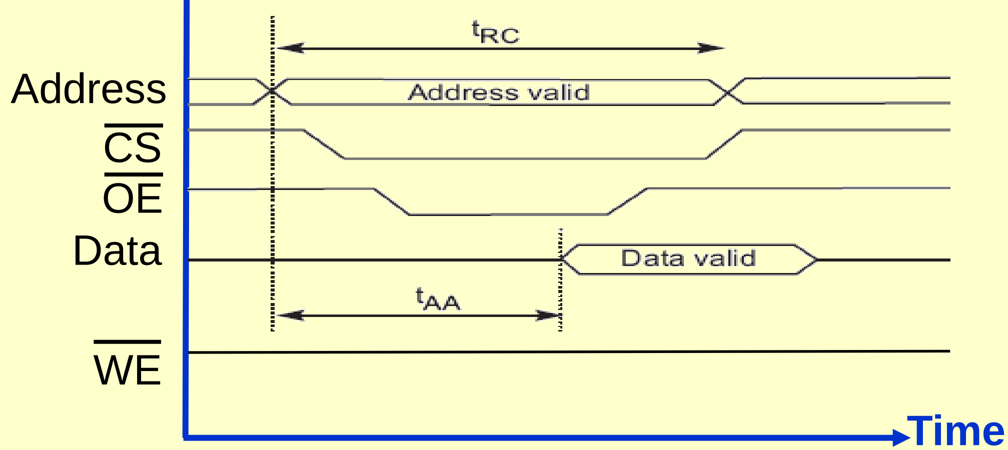
DRAM (Dynamic RAM)

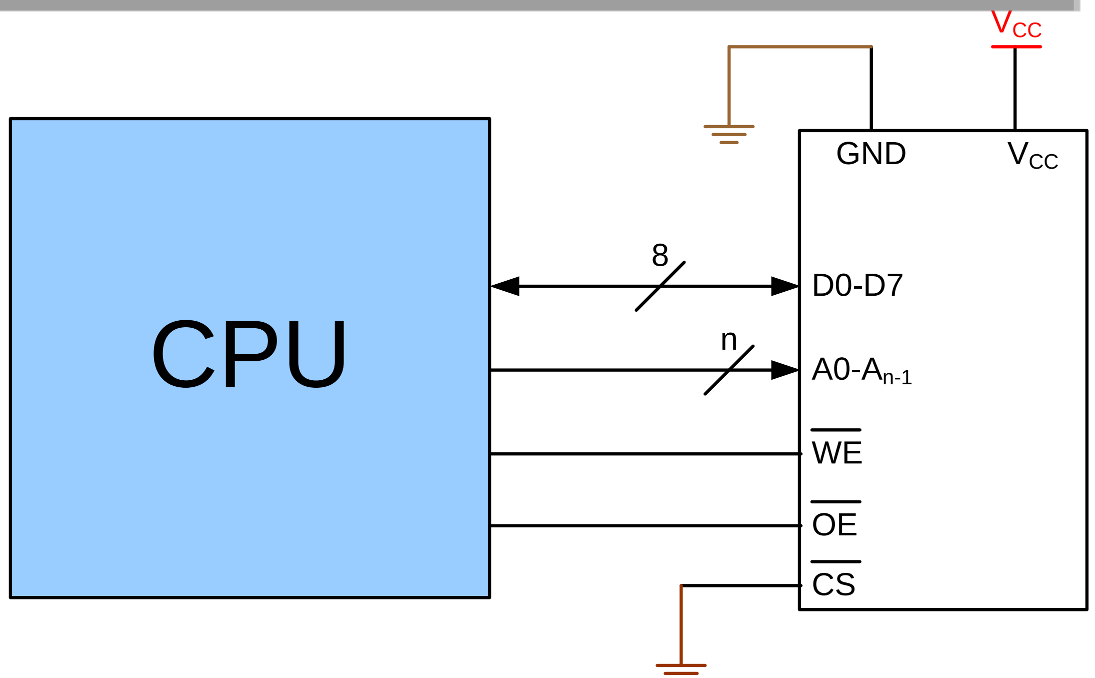
- Made of capacitors
- Advantages:
  - Less power consumption
  - Cheaper
  - High capacity
- Disadvantages:
  - Slower
  - Refresh needed

CPU

- Tasks:
  - It should execute instructions
    - It should recall the instructions one after another and execute them

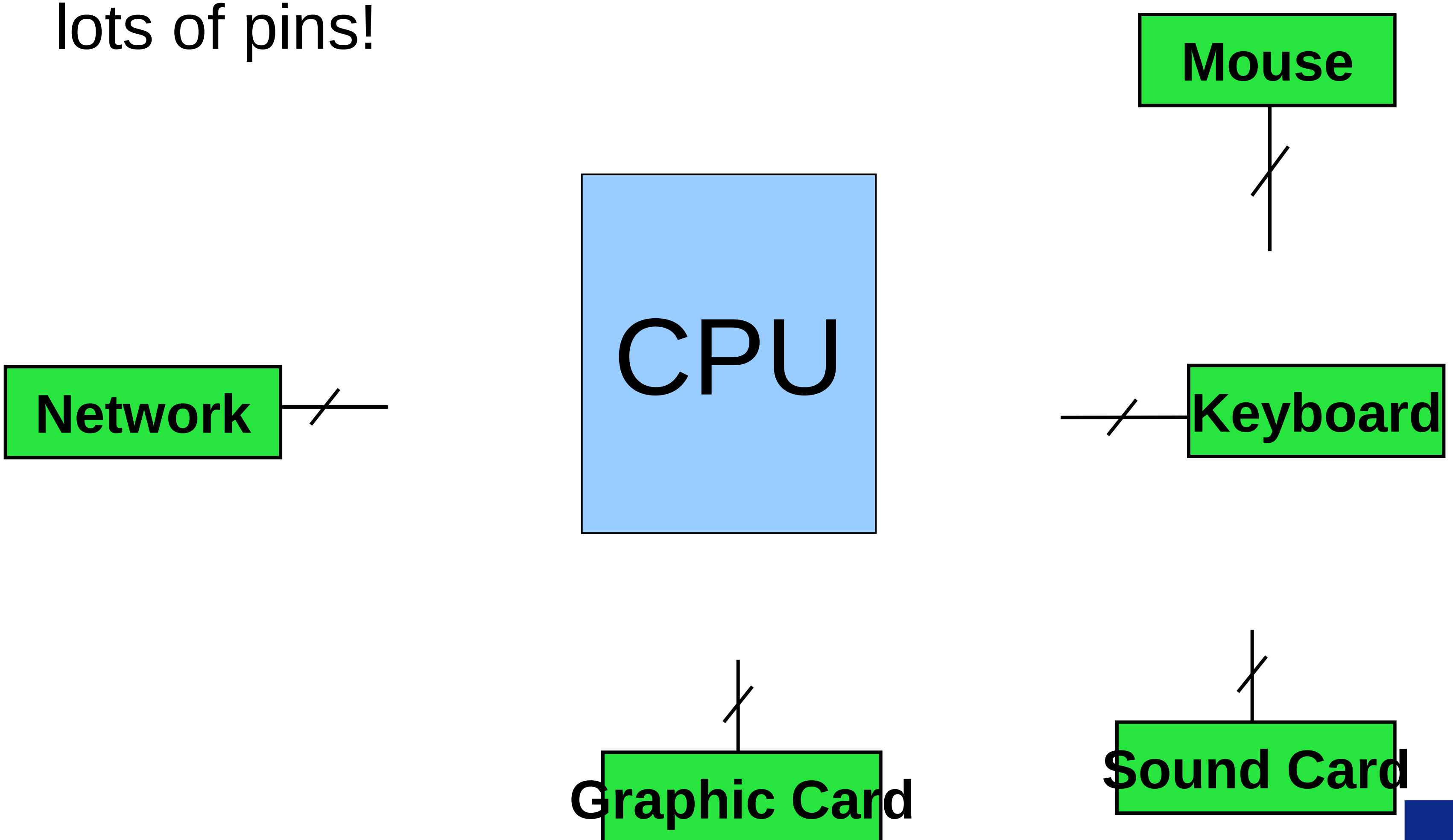
Reading from memory



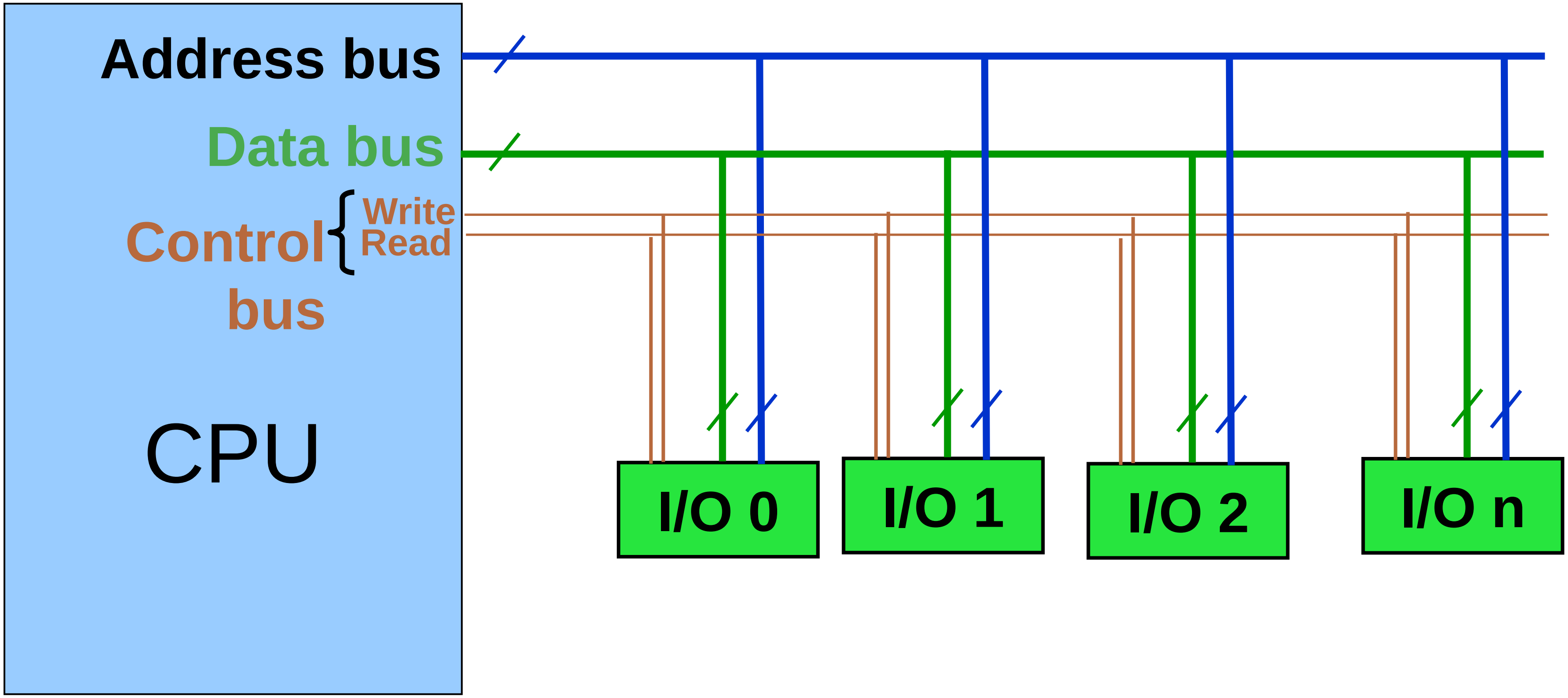


Connecting I/Os to CPU

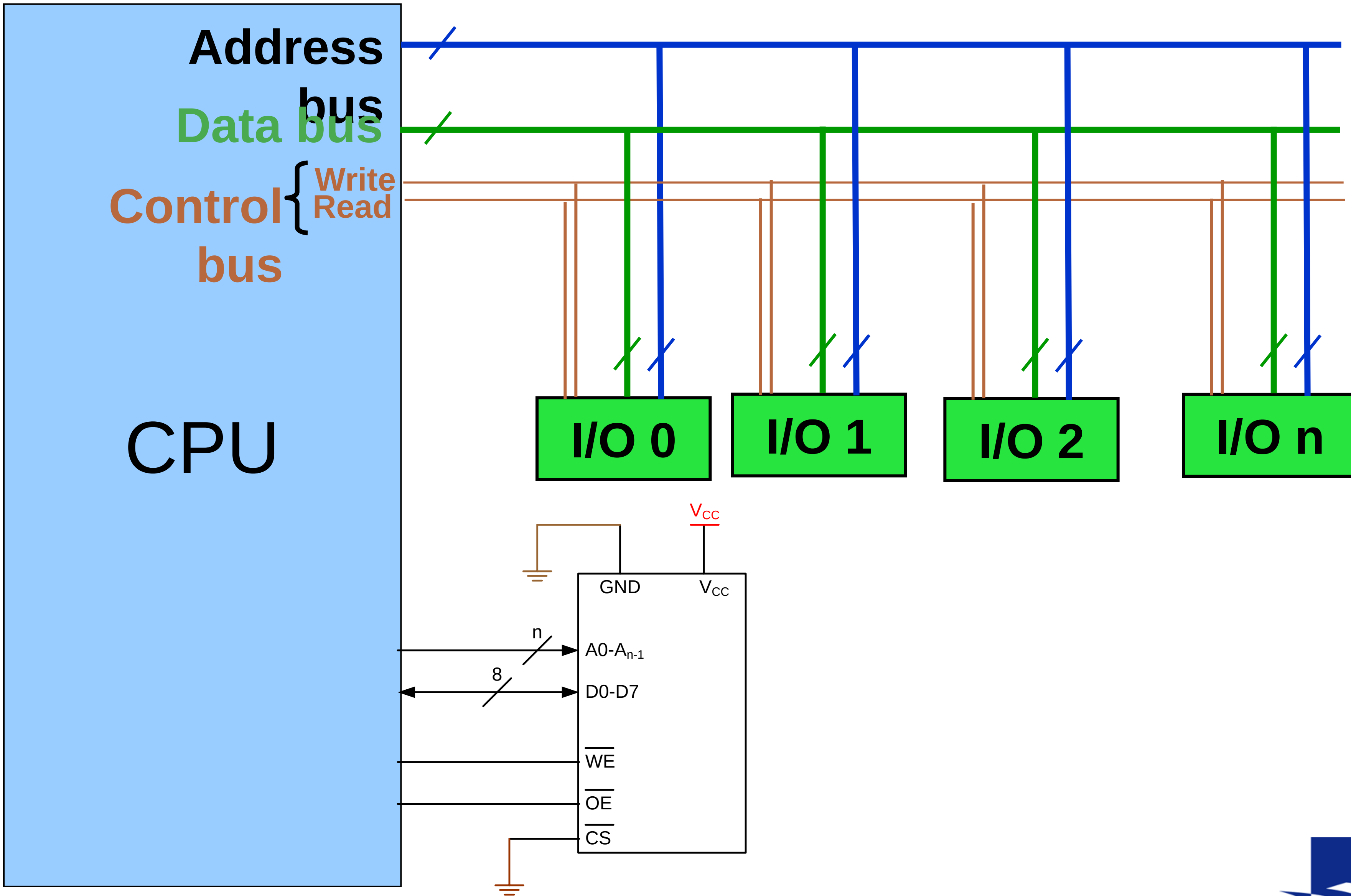
- CPU should have lots of pins!



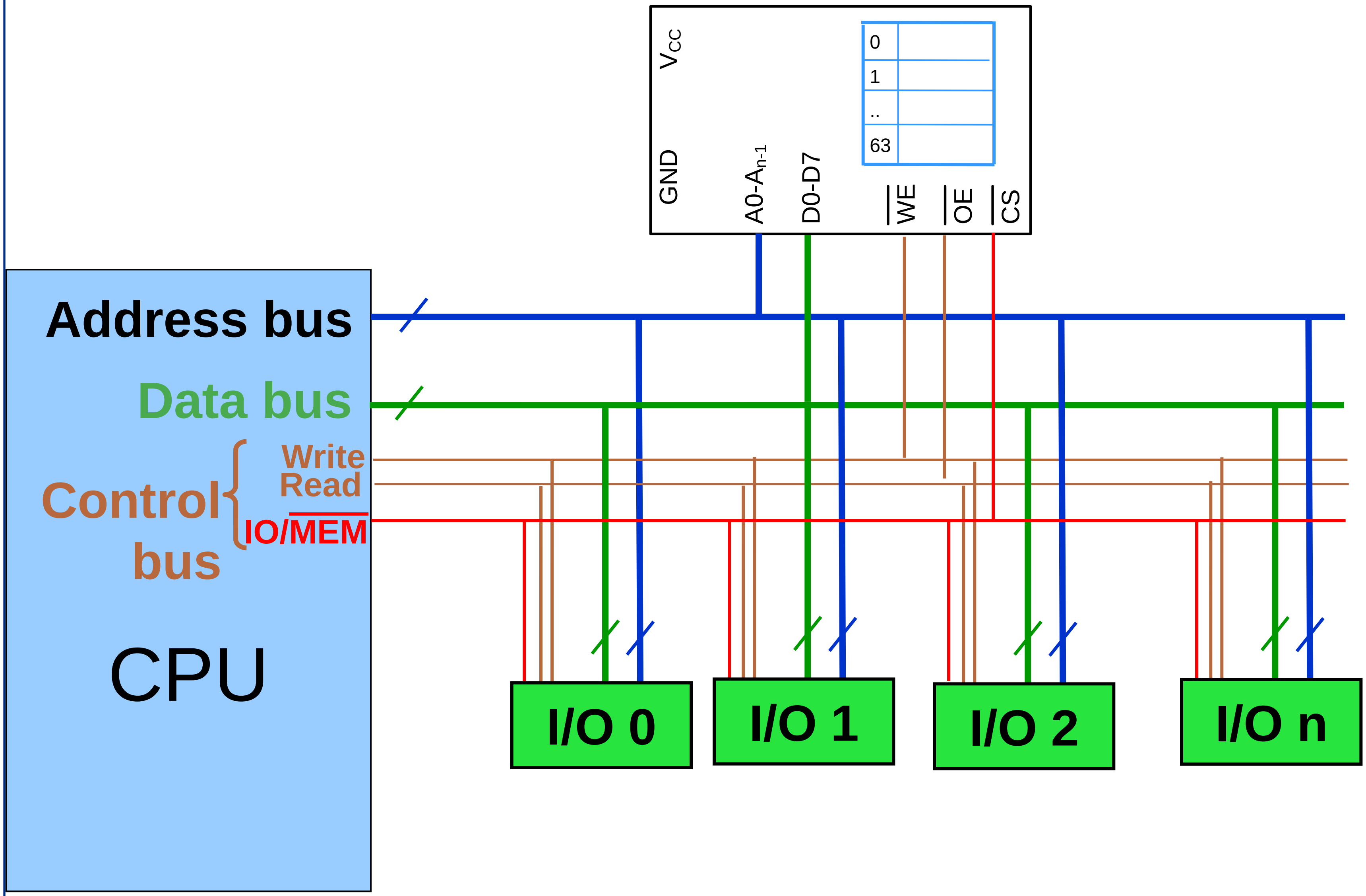
Connecting I/Os to CPU using bus



Connecting I/Os and Memory to CPU

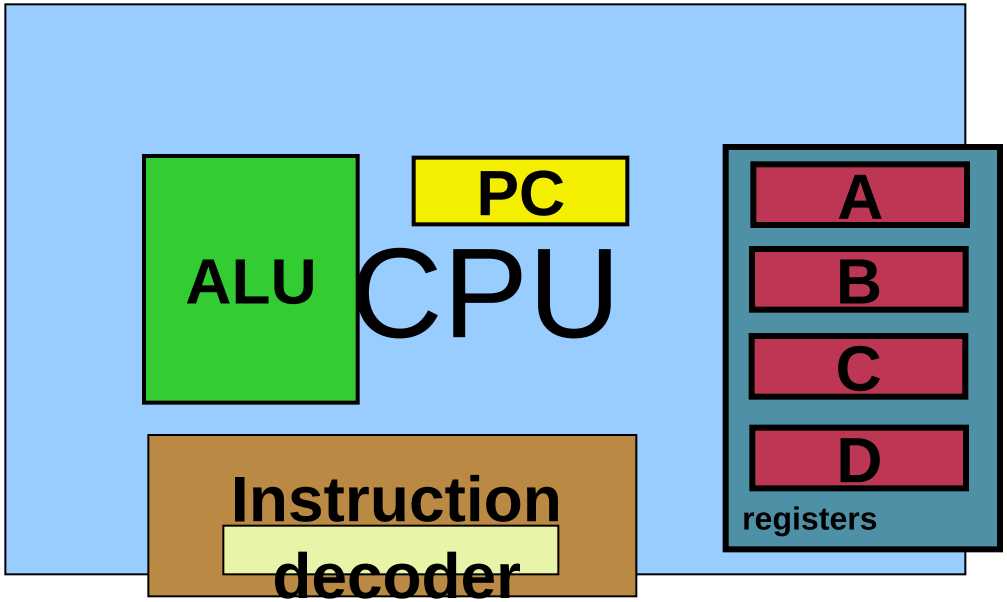


Connecting I/Os and Memory to CPU

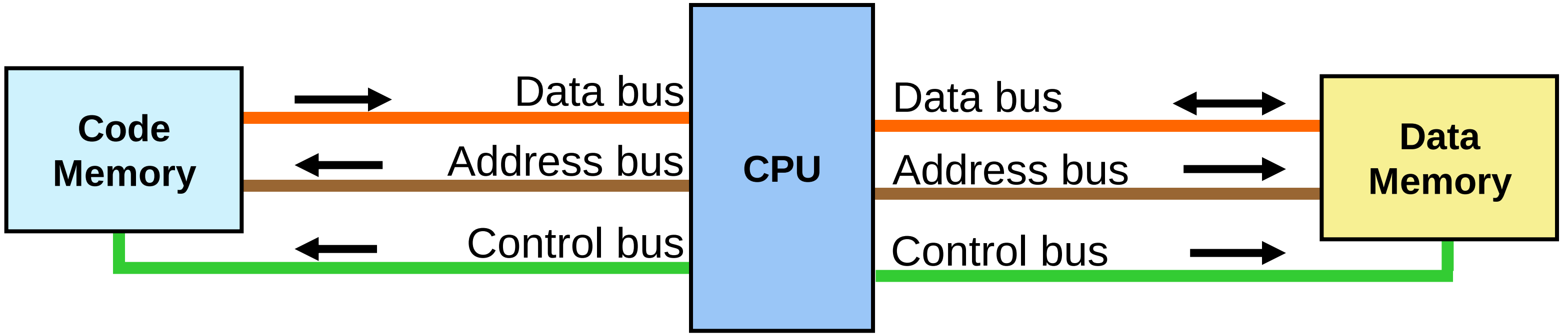


Inside the CPU

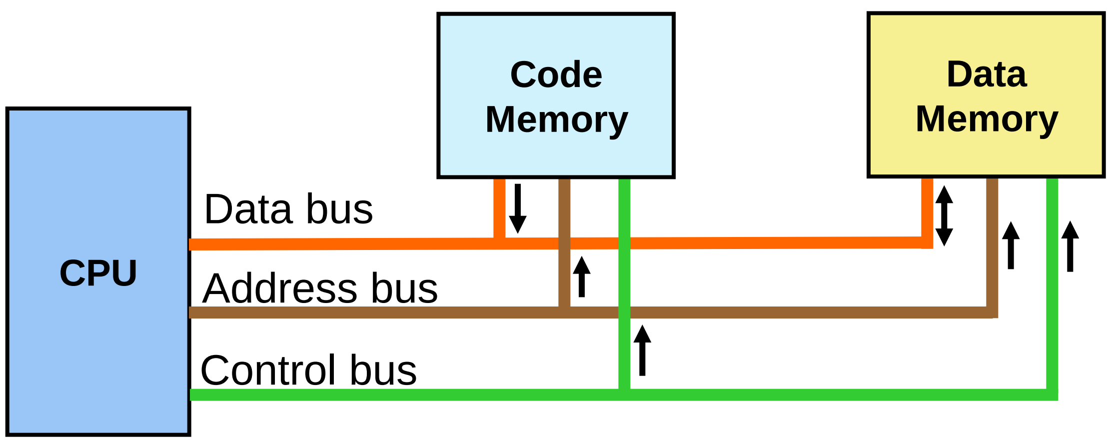
- PC (Program Counter)
- Instruction decoder
- ALU (Arithmetic Logic Unit)
- Registers



Von Neumann vs. Harvard architecture



- Harvard architecture



- Von Neumann architecture

Decimal / binary

- If we had 2 fingers, we would have preferred calculating in binary (like a computer) !

Test (”socrative.com”: Room = MSYS)

- What ’s the benefits of writing programs in assembly (as opposed to for example C) ?

A:

The code will match all computers.

B:

You have full control off, what is going on.

C:

It is very easy to read and understand assembly programs.

Decimal and binary numbers

Converting from binary to decimal

To convert from binary to decimal, it is important to understand the concept of weight associated with each digit position. First, as an analogy, recall the weight of numbers in the base 10 system, as shown in the diagram. By the same token, each digit position of a number in base 2 has a weight associated with it:

$740683_{10} =$

3×10^0	=	3
8×10^1	=	80
6×10^2	=	600
0×10^3	=	0000
4×10^4	=	40000
7×10^5	=	700000
		740683

$110101_2 =$

				Decimal	Binary
1×2^0	=	1×1	=	1	1
0×2^1	=	0×2	=	0	00
1×2^2	=	1×4	=	4	100
0×2^3	=	0×8	=	0	0000
1×2^4	=	1×16	=	16	10000
1×2^5	=	1×32	=	32	100000
				53	110101

From decimal to binary

Example 0-1

Convert 25_{10} to binary.

Solution:

	Quotient	Remainder	
25/2 =	12	1	LSB (least significant bit)
12/2 =	6	0	
6/2 =	3	0	
3/2 =	1	1	
1/2 =	0	1	MSB (most significant bit)

Therefore, $25_{10} = 11001_2$.

Hexa-decimal numbers

Table 0-1: Base 16 Number System

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Notice:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

HEX number examples

Example 0-4

Represent binary 100111110101 in hex.

Solution:

First the number is grouped into sets of 4 bits: 1001 1111 0101.

Then each group of 4 bits is replaced with its hex equivalent:

1001	1111	0101
9	F	5

Therefore, 100111110101_2 = 9F5 hexadecimal.

Example 0-5

Convert hex 29B to binary.

Solution:

		2	9	B
29B	=	0010	1001	1011

Dropping the leading zeros gives 1010011011.

Test (”socrative.com”: Room = MSYS)

- How do you write the decimal number 217 in ”hex” notation ?

A: 17

B: 6C

C: C1

D: D9

Binary addition

Table 0-3: Binary Addition

A + B	Carry	Sum
0 + 0	0	0
0 + 1	0	1
1 + 0	0	1
1 + 1	1	0

Example 0-8

Add the following binary numbers. Check against their decimal equivalents.

Solution:

	Binary	*Decimal*
	1101	13
+	1001	9
	10110	22

Binary addition

				1	1	1	1			
X	190		1	0	1	1	1	1	1	0
Y	+ 141	+	1	0	0	0	1	1	0	1
$X + Y$	331	1	0	1	0	0	1	0	1	1

				1		1	1			
X	173		1	0	1	0	1	1	0	1
Y	+ 44	+	0	0	1	0	1	1	0	0
$X + Y$	217		1	1	0	1	1	0	0	1

Figure 2–1 Examples of decimal and corresponding binary additions.

Hexa-decimal addition

Example 0-10

Perform hex addition: 23D9 + 94BE.

Solution:

	23D9	LSD:	9 + 14 = 23	23 – 16 = 7 with a carry
+	94BE		1 + 13 + 11 = 25	25 – 16 = 9 with a carry
	B897		1 + 3 + 4 = 8	
		MSD:	2 + 9 = B	

End of lesson 2

Questions / comments ?

iha.dk