







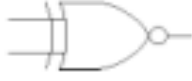
Sistemas Digitais

Arquitetura Básica

Aula 10

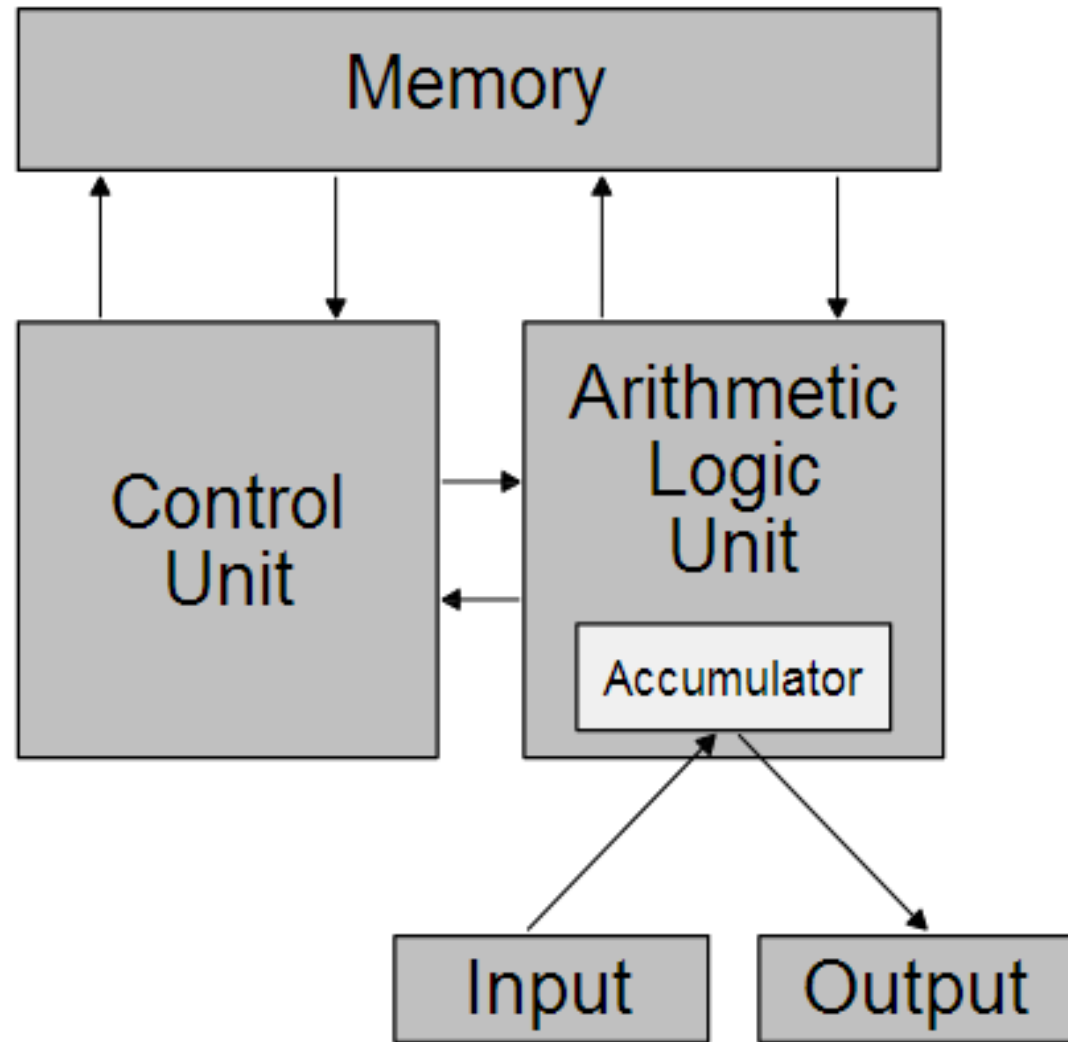
Prof. Leandro Nogueira Couto
UFU – Monte Carmelo
05/2013



E AND		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	Função E: Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	S=A.B
A	B	S																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OU OR		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	Função OU: Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	S=A+B
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NÃO NOT		<table><tr><th>A</th><th>S</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	S	0	1	1	0	Função NÃO: Inverte a variável aplicada à sua entrada.	S=\overline{A}									
A	S																		
0	1																		
1	0																		
NE NAND		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	Função NE: Inverso da função E.	S=$\overline{(A.B)}$
A	B	S																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOU NOR		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	Função NOU: Inverso da função OU.	S=$\overline{(A+B)}$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
OU EXCLUSIVO		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0	Função OU Exclusivo: Assume 1 quando as variáveis assumirem valores diferentes entre si.	S=A⊕B S= $\overline{A}.B + A.\overline{B}$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
COINCIDÊNCIA		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1	Função Coincidência: Assume 1 quando houver coincidência entre os valores das variáveis.	S= A⊙B S= $\overline{A}.\overline{B} + A.B$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

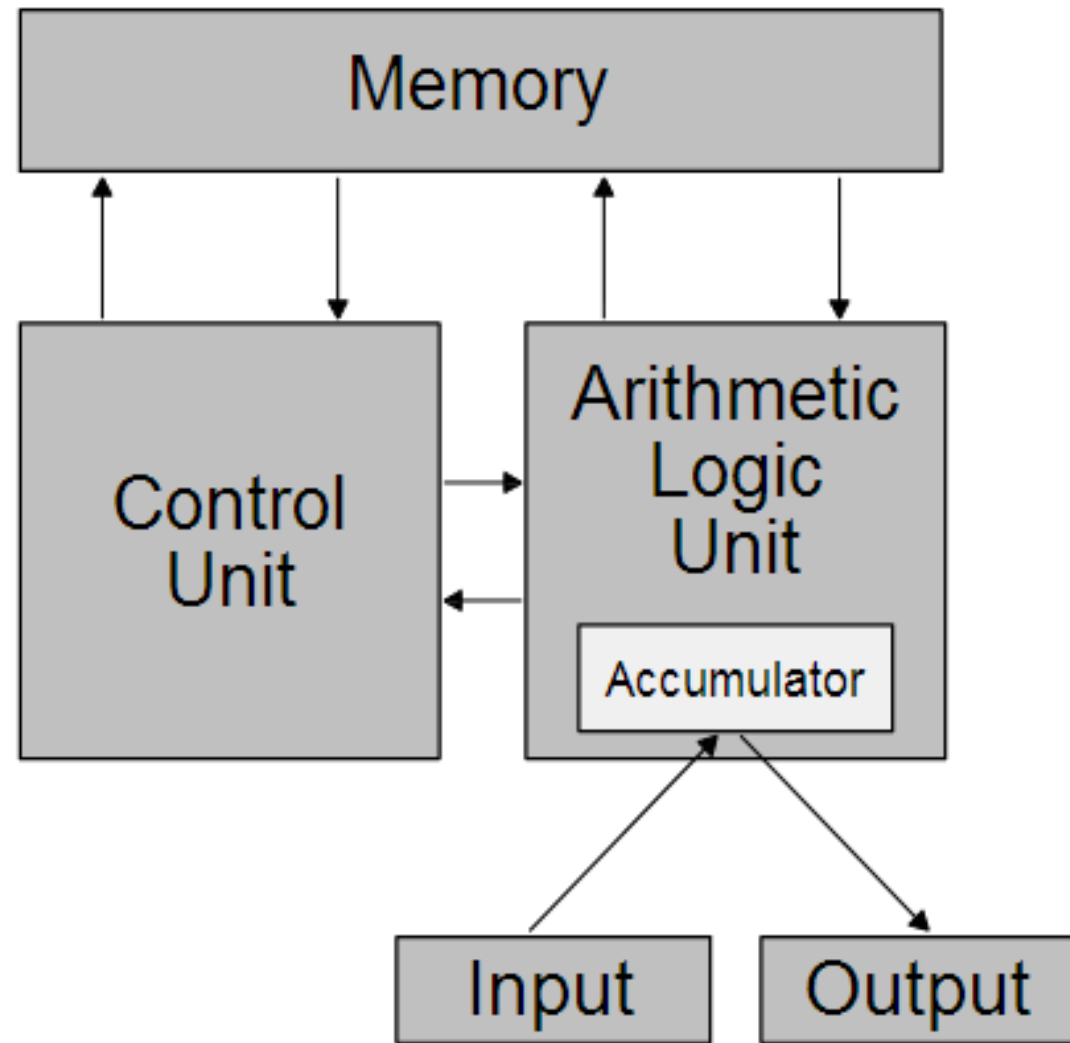
Arquitetura

- Cada hardware tem uma organização e arquitetura particular
- Cada arquitetura permite a realização determinadas operações
- A maioria dos computadores, porém, seguem o design de arquitetura geral de Von Neumann



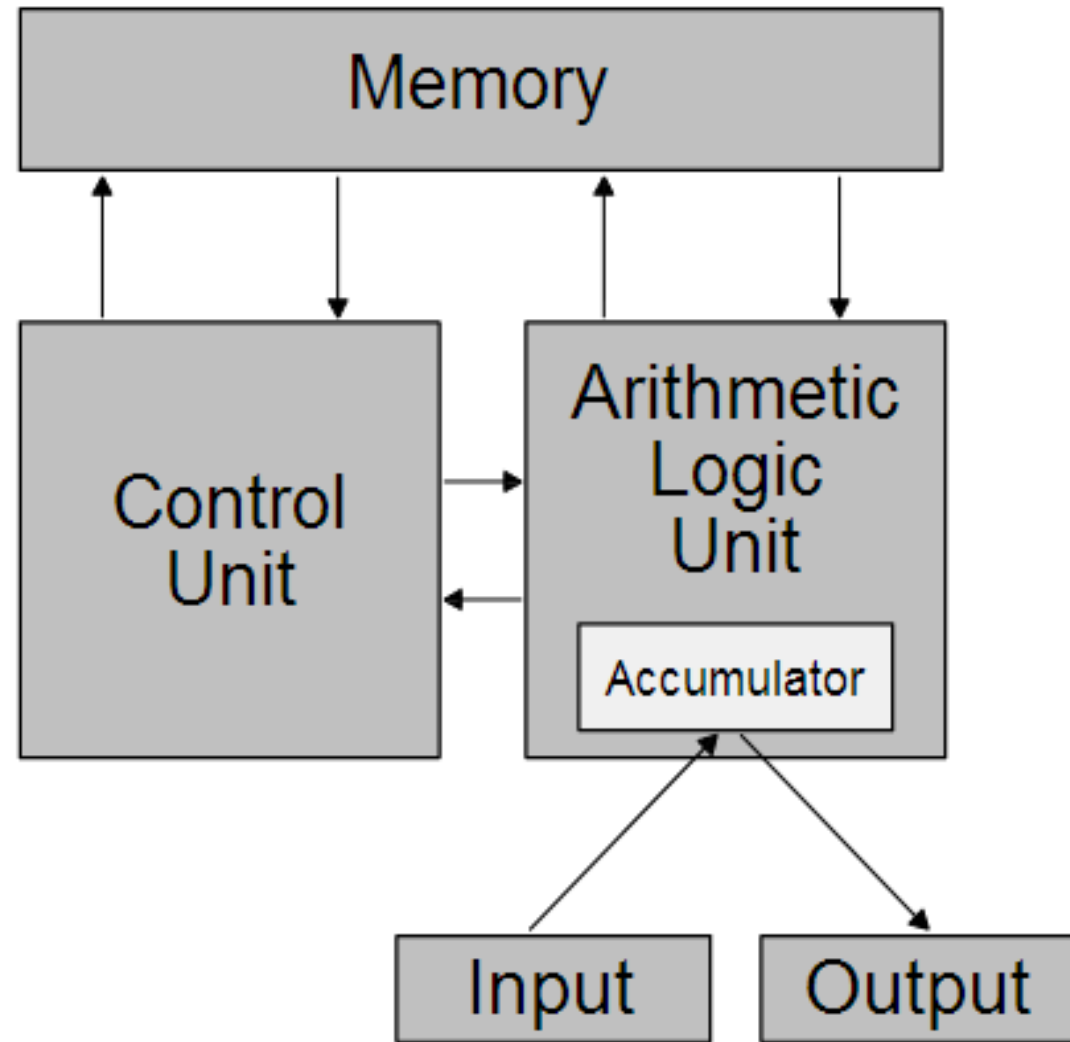
Arquitetura

- A maioria dos computadores realiza instruções como Move, Store, Read, Add, Subtract, Multiply, Jump-if-Zero, Jump-to, Jump-if-Equal
- para um referência dos Jumps no x86, veja <http://www.unixwiz.net/techtips/x86-jumps.html>



Arquitetura

- Temos conjuntos de instruções complexos (http://en.wikipedia.org/wiki/X86_instruction_listings) e reduzidos (<https://sites.google.com/site/6502assembly/6502-instruction-set>)
- Qual o mínimo de instruções que podemos ter em um computador?
- SBNZ a,b,c,d

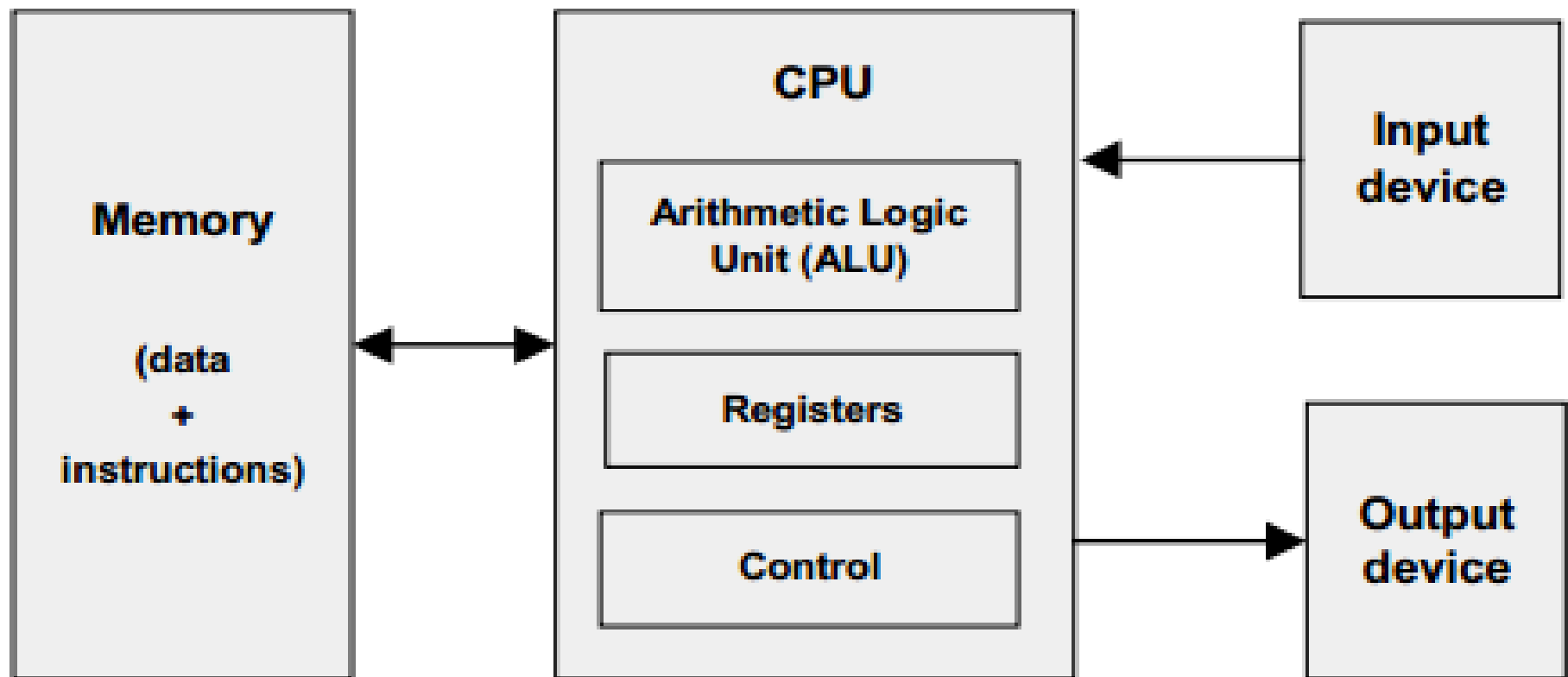


Arquitetura

- Qual o mínimo de instruções que podemos ter em um computador?
- **Uma! SBNZ a,b,c,d**
- **Subtract and branch if not zero** ou seja, subtraia A de B, armazene em C e vá para o endereço D se o resultado não for zero.

Arquitetura

- Figura alternativa de Von Neumann:



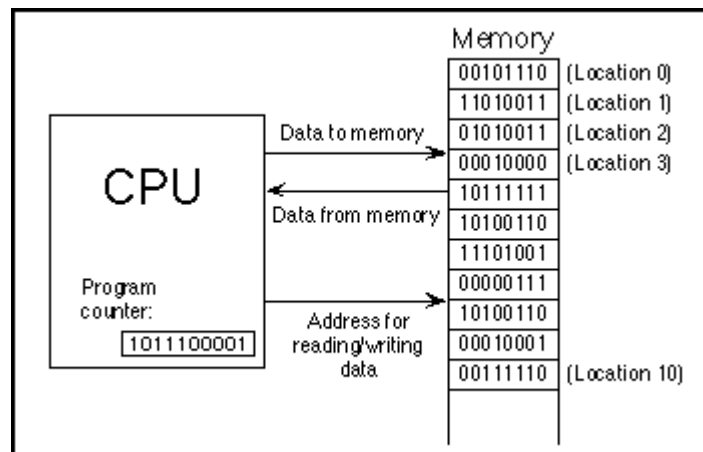
Arquitetura

- Partes
 - Memória
 - Dados

Um endereço de dado pode ser escrito ou lido (**write** ou **read**)
 - Instruções
 - Uma instrução como **while j<100**
{ sum=sum+j } é traduzida para várias instruções de processador pelo compilador
 - O processador deve **ler e decodificar** cada instrução antes de executá-la
 - Em geral (mas nem sempre): uma instrução = uma **palavra** da memória (word)

Arquitetura

- Partes
 - Memória
 - Certas memórias precisam de alimentação para permanecer ativas e não perder informações. São memórias **voláteis**, como flip-flop e RAM
 - Algumas memórias são **não-voláteis**, como o CD-ROM ou memórias eletrônicas como EEPROM



Arquitetura

- Partes
 - CPU
 - ULA
 - Registradores
 - UC

Arquitetura

- Partes
 - CPU
 - Unidade Lógica Aritmética (ULA):
Operações matemáticas, por exemplo:
 - Adição
 - Subtração
 - Testar se um número é positivo
 - Testar se 2 números são iguais
 - Fazer “shift” (deslocamento) em um número

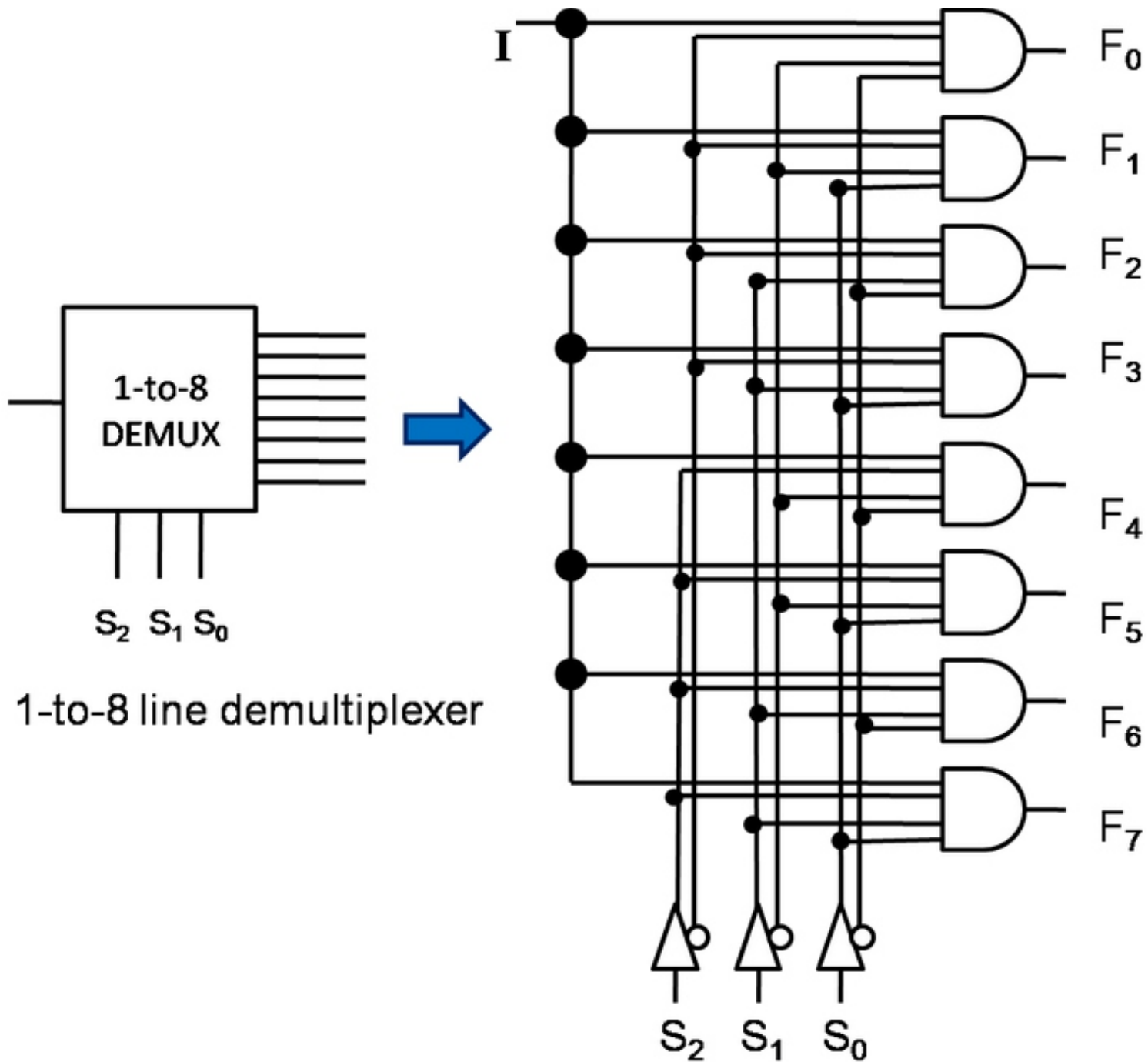
Arquitetura

- Partes
 - CPU
 - Registradores
 - **Armazenamento local** de valores, para evitar acesso constante e demorado à memória
 - Poucos, dentro da CPU
 - **Acumulador** é um registrador especial onde resultados intermediários da ULA são guardados

Arquitetura

- Partes
 - CPU
 - Unidade de Controle (UC)
 - Uma instrução de computador é normalmente uma palavra de 16 a 32 bits
 - A unidade de controle deve decodificá-la e direcionar o fluxo dos bits para a execução apropriada

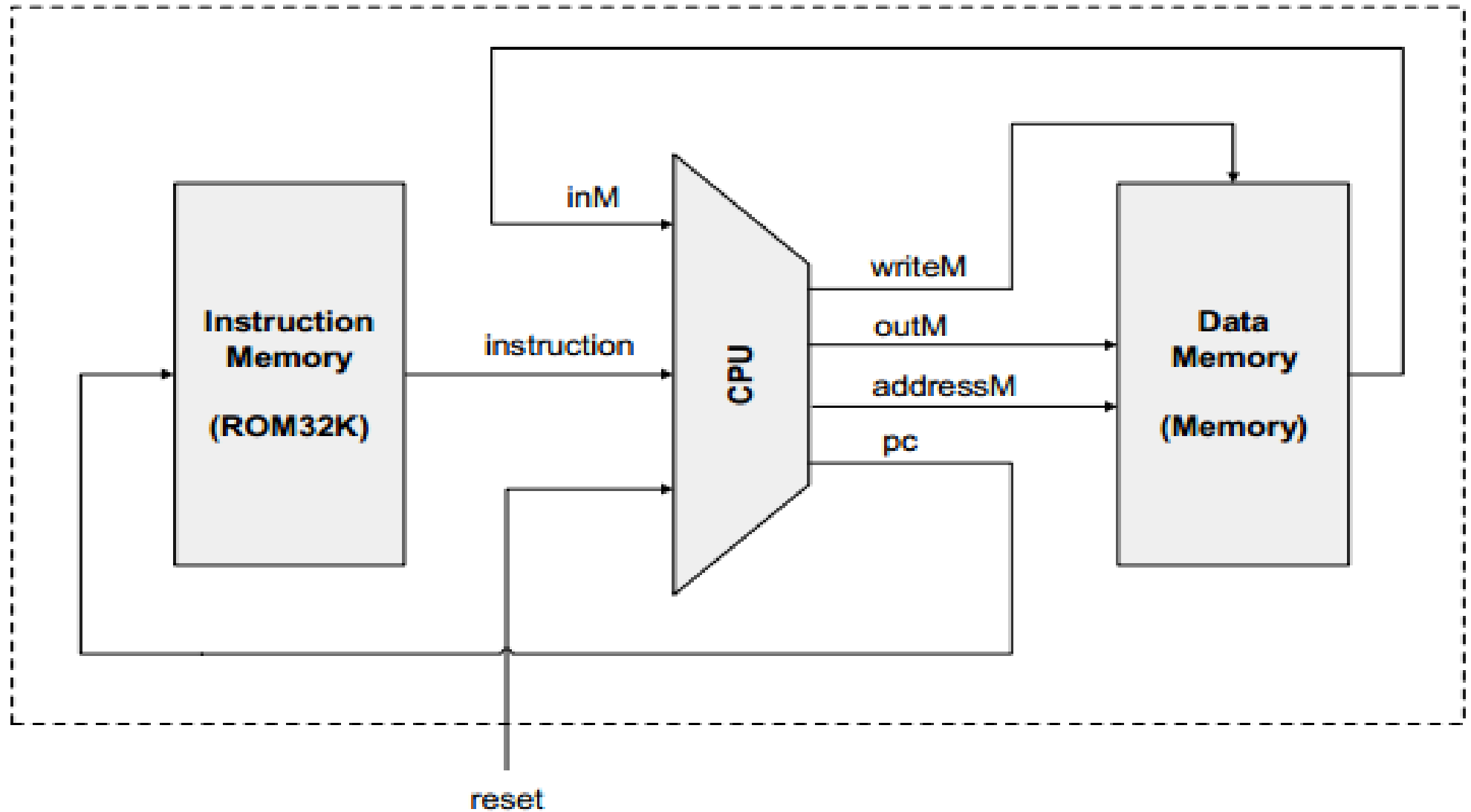
Arquitetura



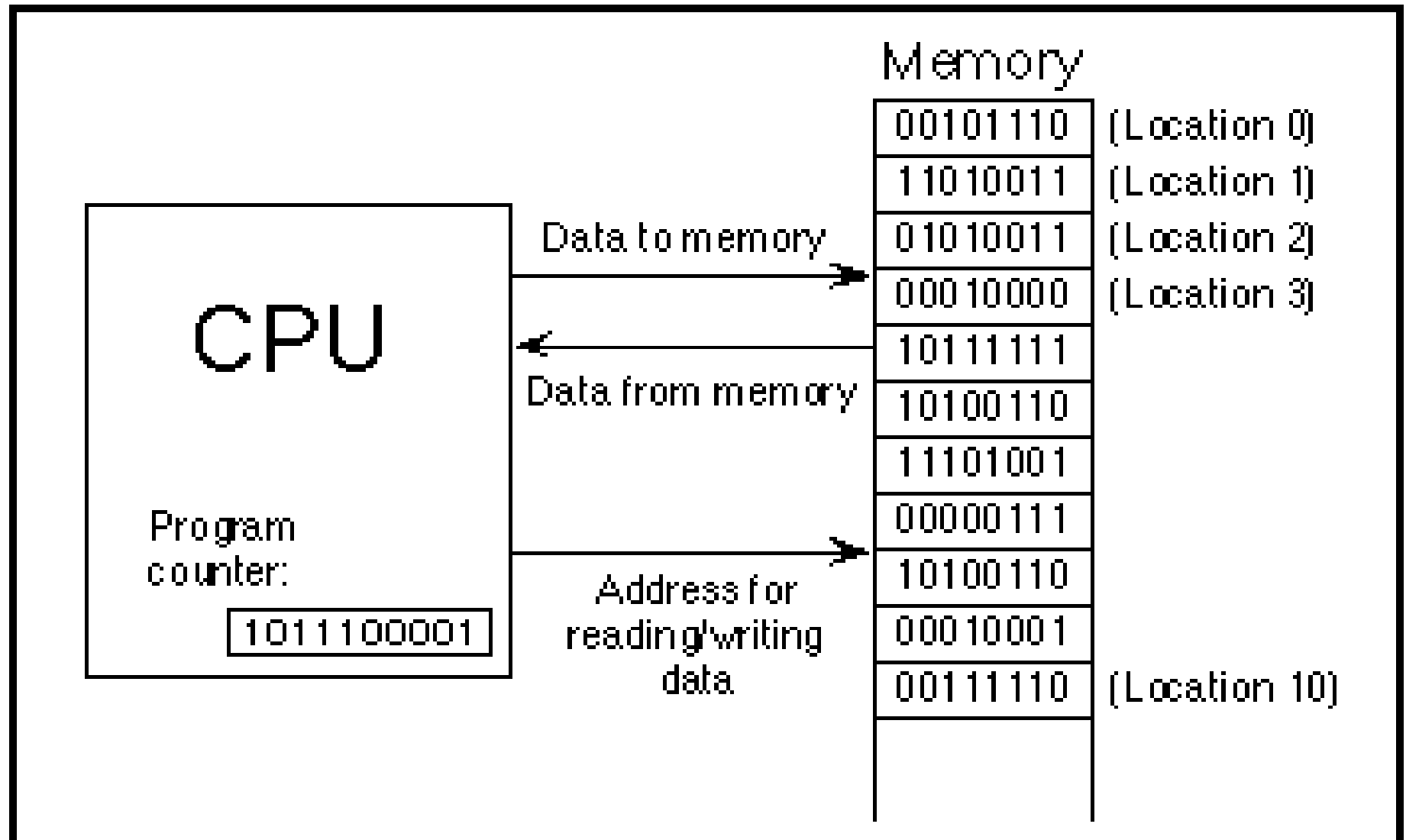
Arquitetura

- Ciclo para execução de uma instrução:
 - Fetch
 - Incrementa o Program Counter (ou pula PC para endereço apropriado) para ler a próxima instrução
 - Decode
 - Descobre o que a instrução faz
 - Execute
 - Enviando sinais de controle para ativar as partes apropriadas, o hardware realiza a instrução

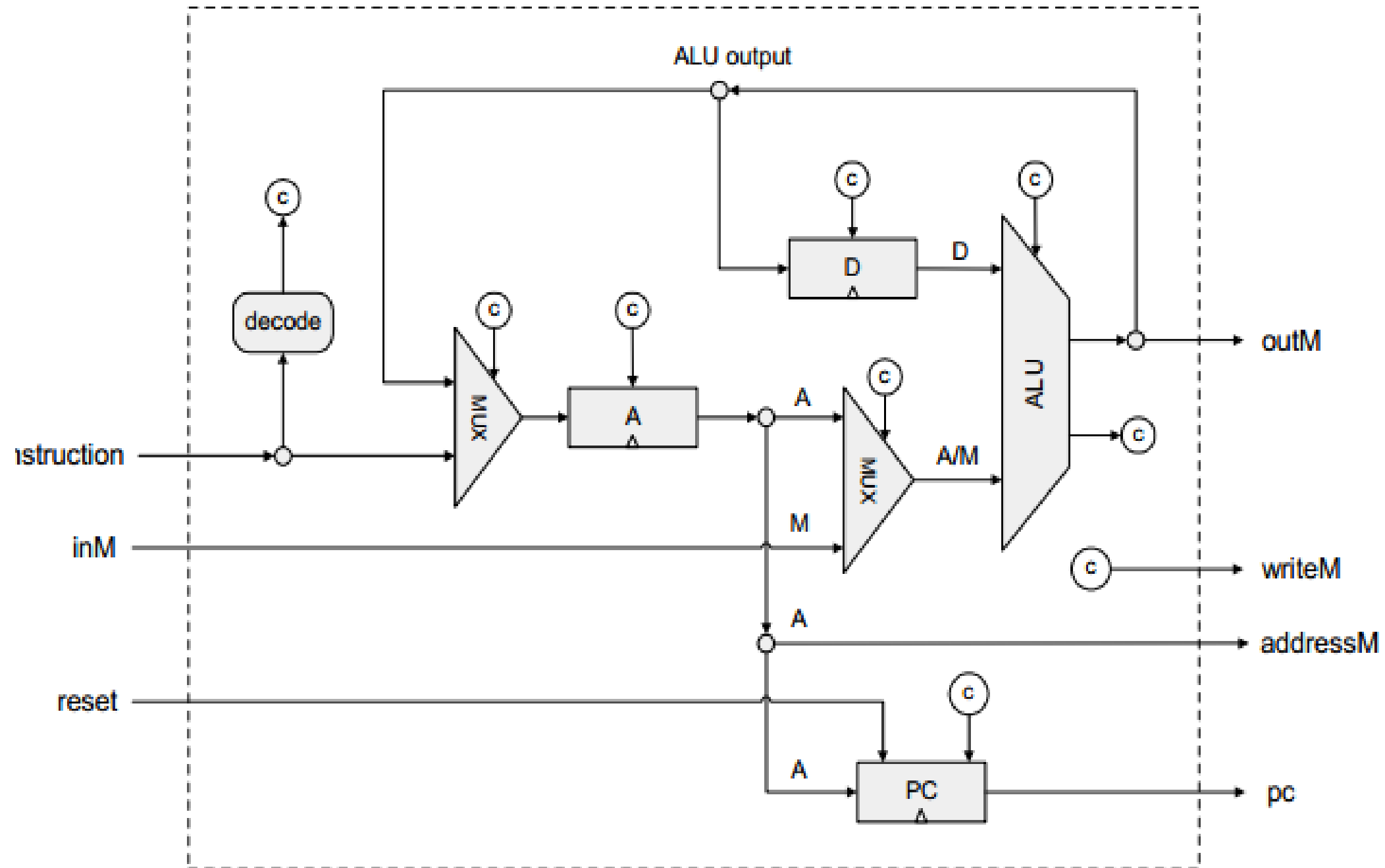
Arquitetura



Arquitetura

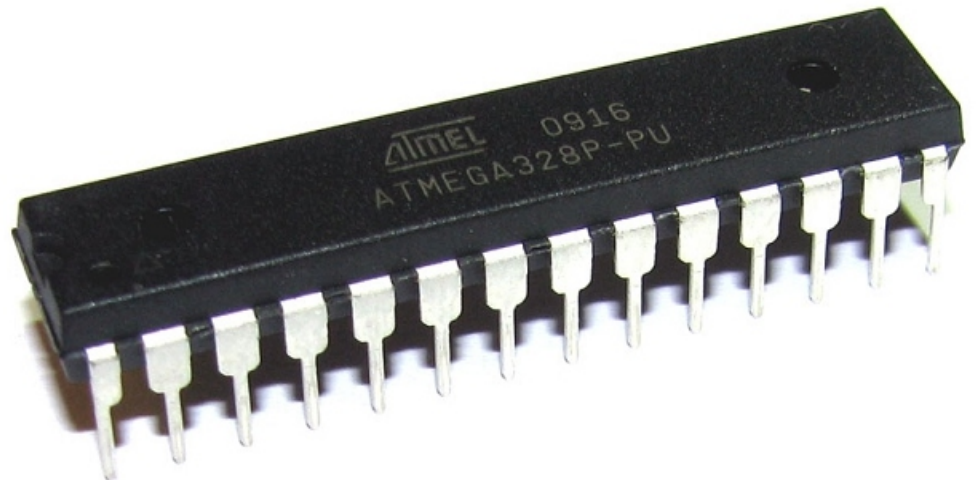
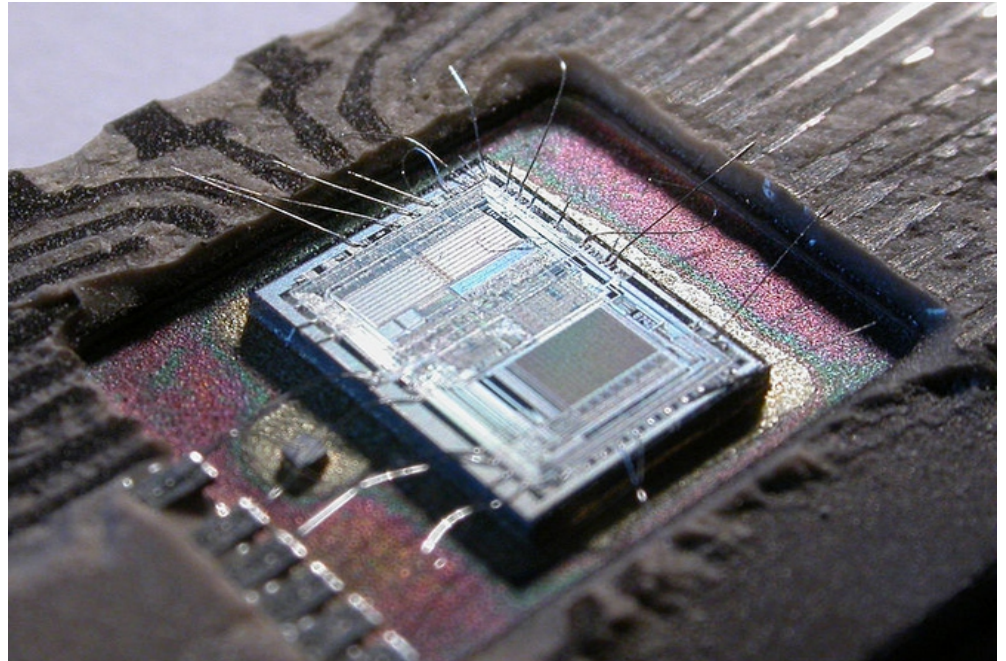


Arquitetura



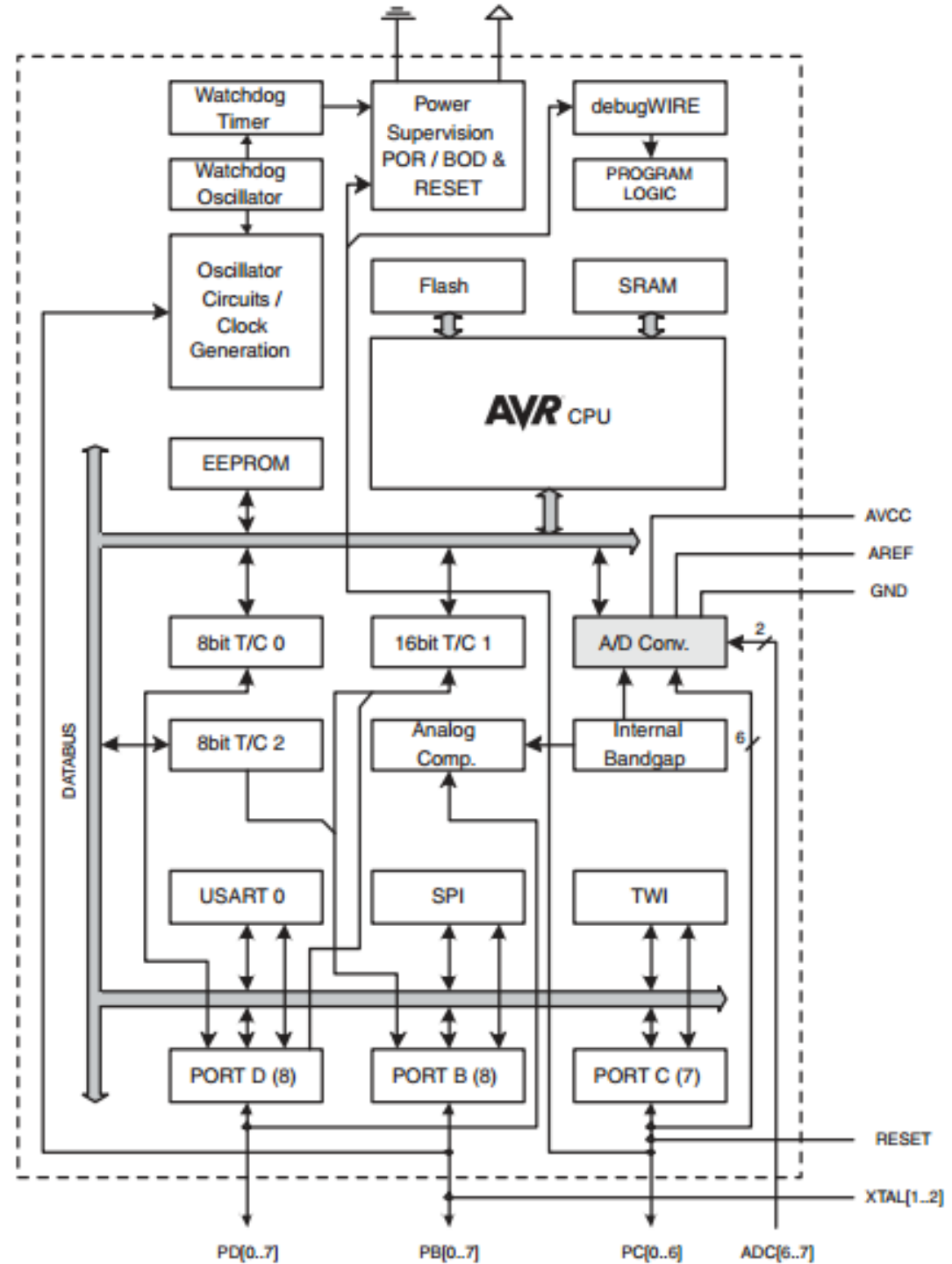
Microcontroladores

- Microcontroladores são pequenos computadores embutidos em circuitos integrados
- Possuem núcleo de processador, memória e E/S programáveis. Geralmente um pouco de RAM e memória EEPROM/Flash para armazenar programas.
- Voltados para **sistemas embarcados** e prototipação



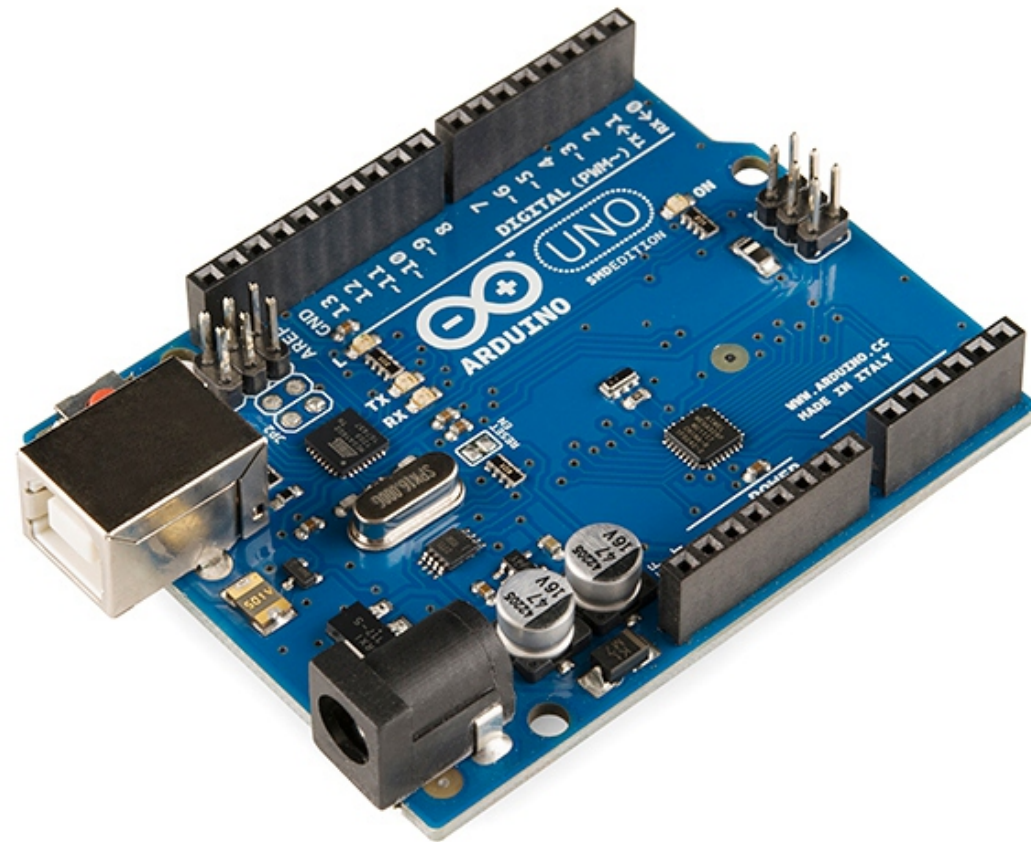
Microcontroladores

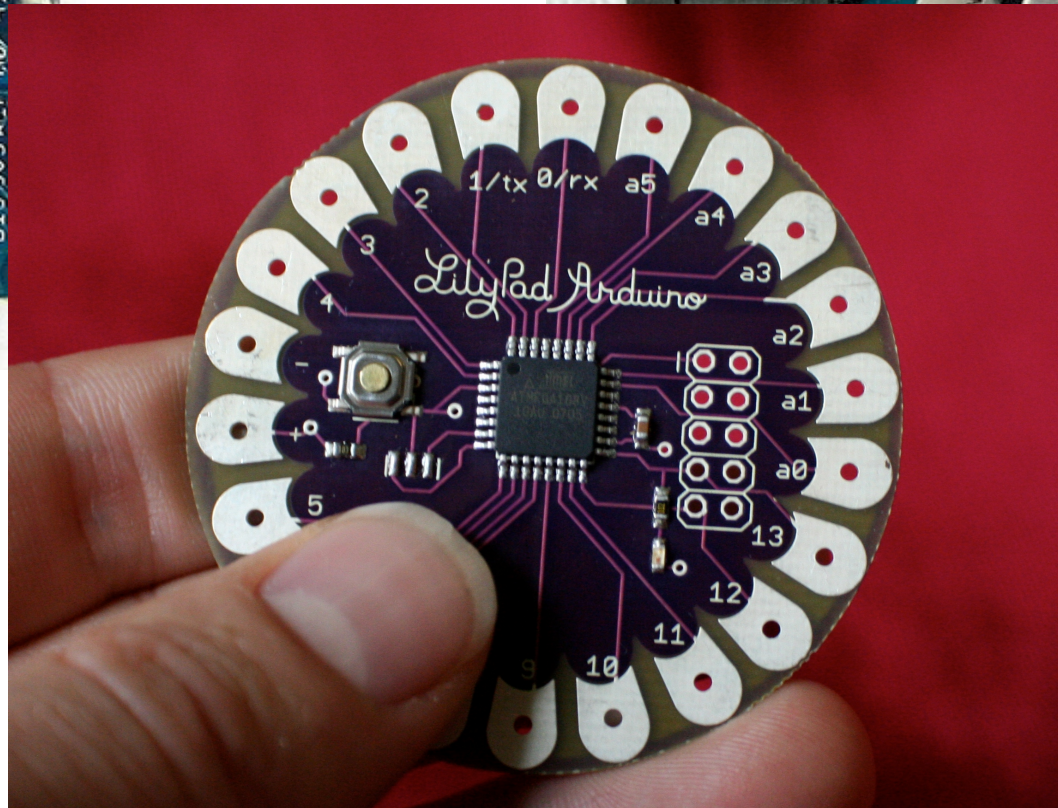
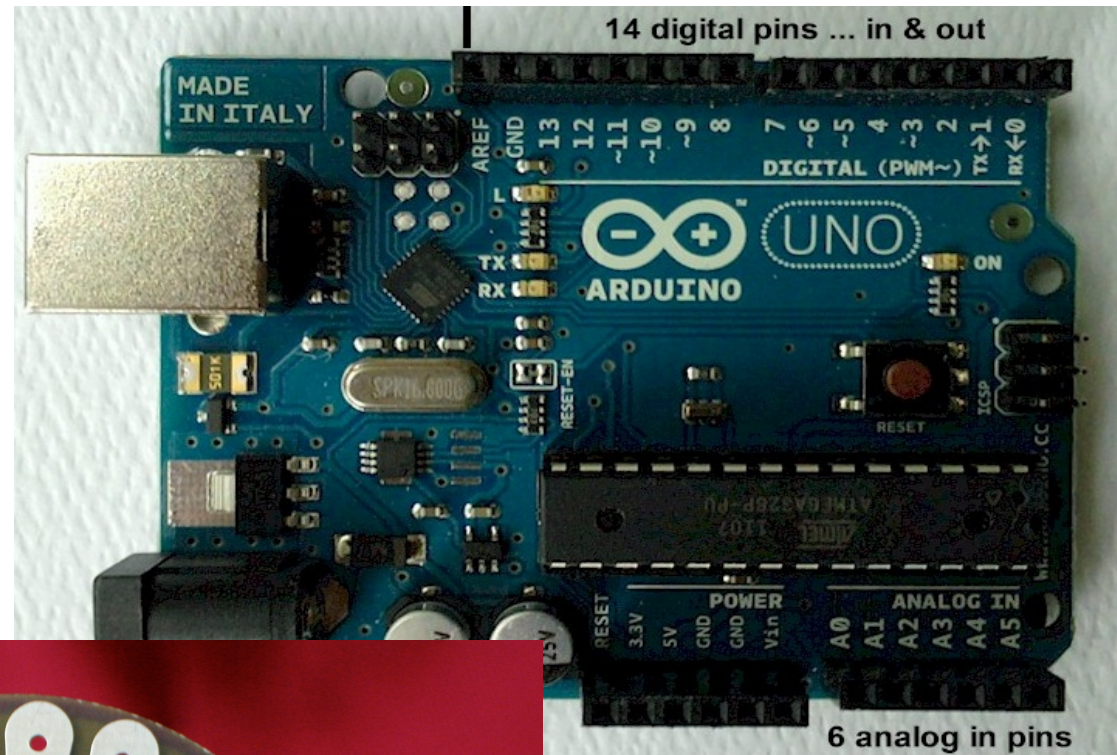
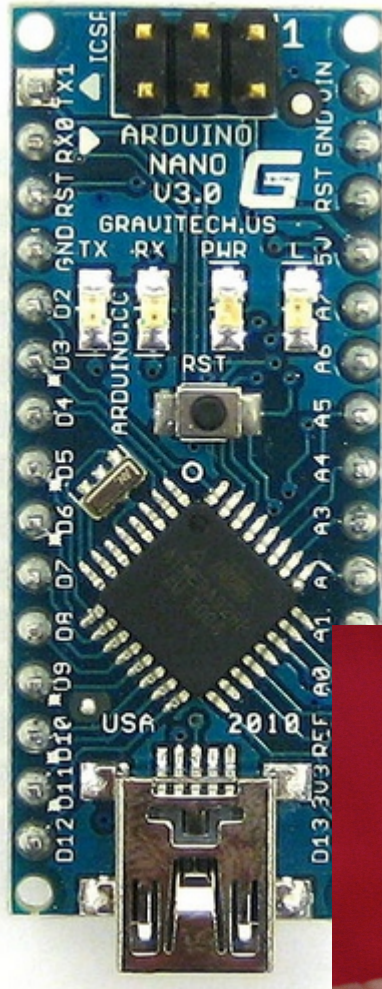
- Microcontrolador mostrando procesador (CPU) + periféricos



Microcontroladores

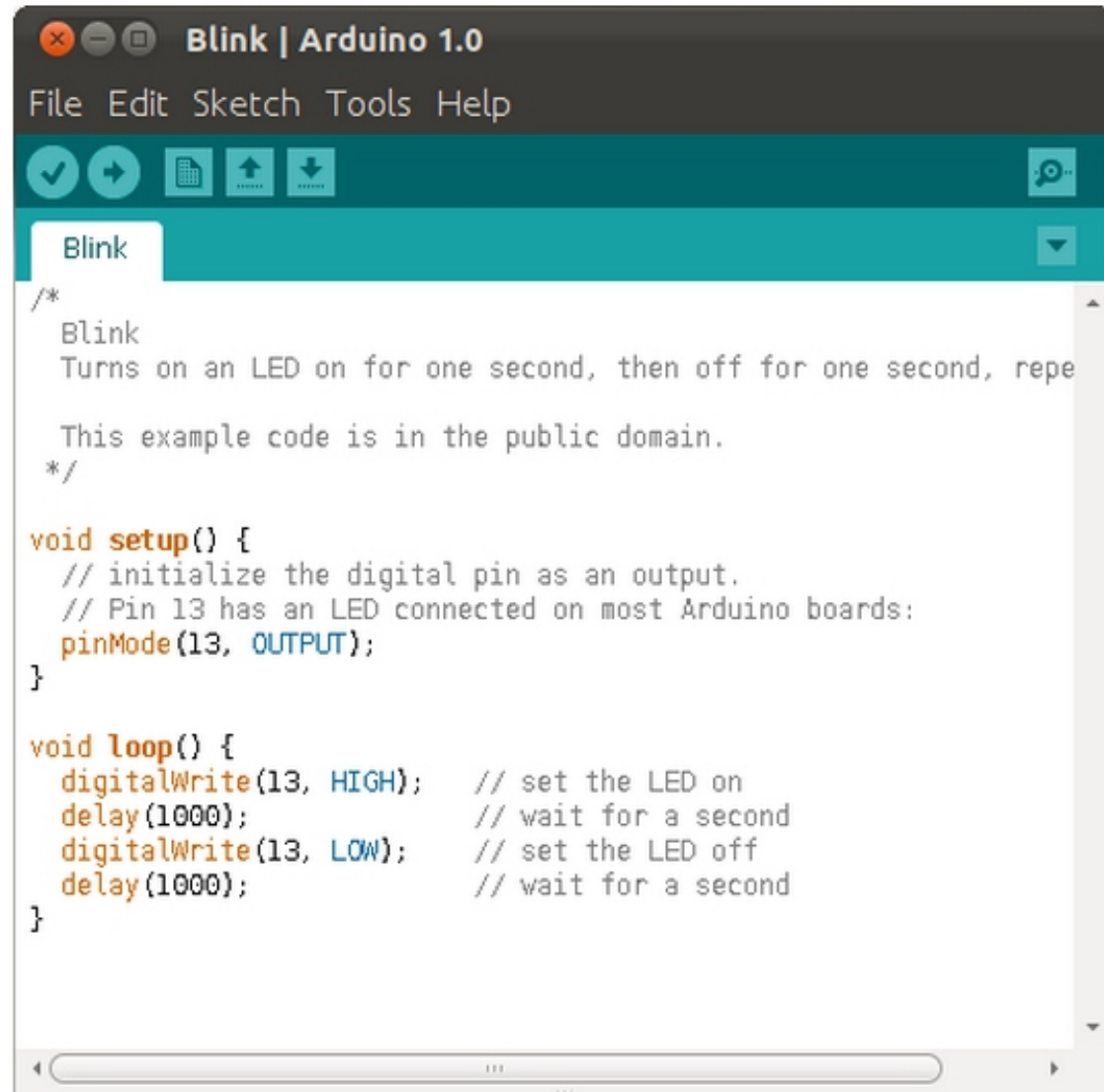
- O Arduino nada mais é que um microcontrolador + **adicionais** (para interface com outros dispositivos, gravação simplificada, etc)
- Basicamente, um microcontrolador que facilita a vida do desenvolvedor (*utilidade imediata*)
- O microcontrolador do Arduino Uno é o **Atmega328**





Microcontroladores

- A programação do **Atmega328** segue um protocolo complexo
- Graças ao Arduino e seus **drivers**, isso já está feito por nós
- Programamos o Arduino usando a Arduino IDE (*Integrated Development Environment*)

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, saving, and uploading. The main editor area shows the "Blink" sketch. The code is as follows:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

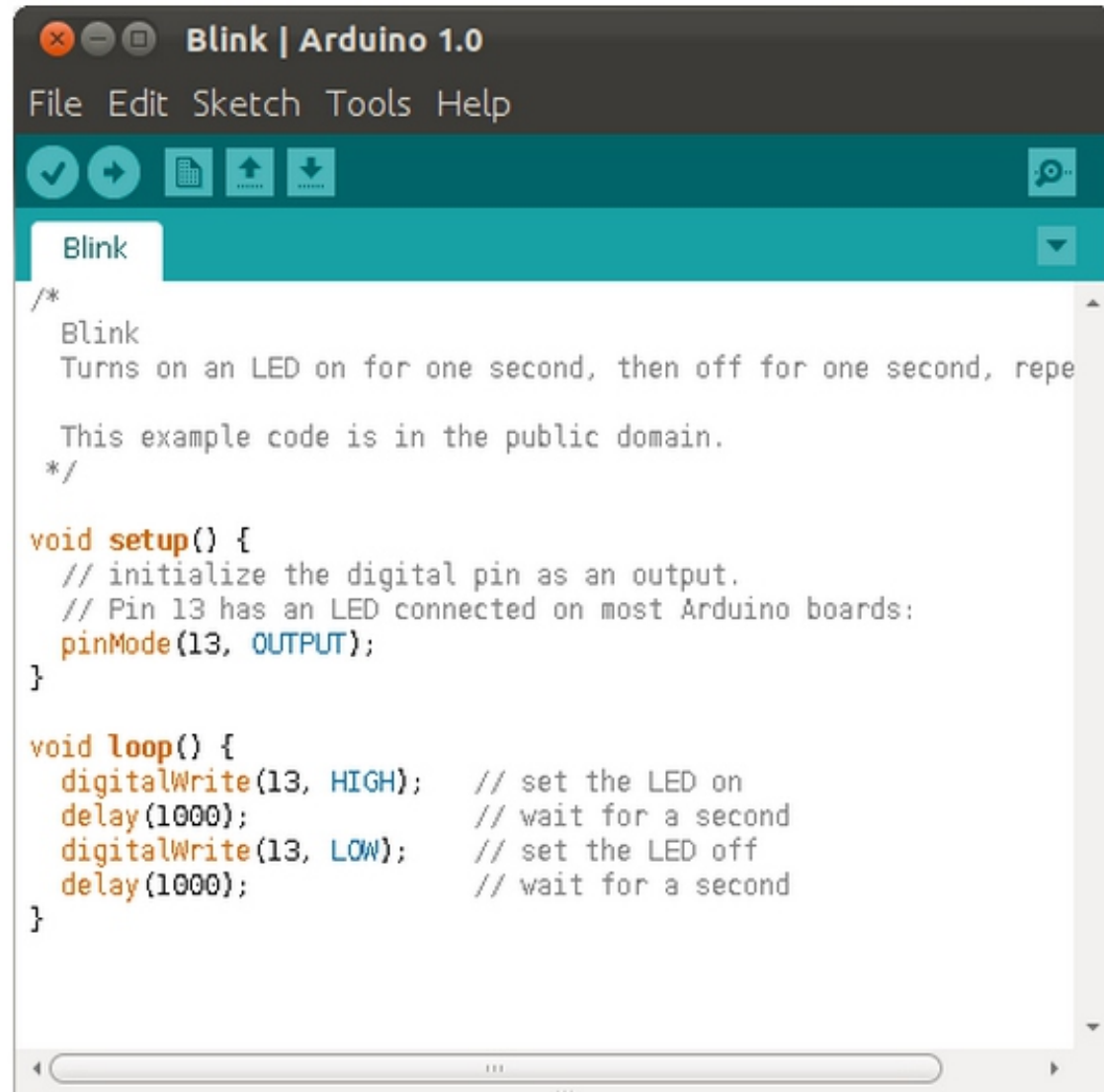
  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```


Microcontroladores

- Como qualquer processador, o controlador Atmega328 apresenta um **conjunto de instruções** específico
- Mas não precisamos programar em **Assembly!**
- A Arduino IDE possui **compilador** para o Atmega328 a partir de uma linguagem semelhante a **C/C++**

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, saving, and uploading. The main text area shows the "Blink" sketch. It starts with a multi-line comment: "/* Blink Turns on an LED on for one second, then off for one second, repeatedly. This example code is in the public domain. */". This is followed by the "void setup()" function, which initializes pin 13 as an output using "pinMode(13, OUTPUT);". Then, the "void loop()" function is shown, which sets the LED on with "digitalWrite(13, HIGH);", waits for one second with "delay(1000);", sets the LED off with "digitalWrite(13, LOW);", and waits for another second with "delay(1000);". The code is color-coded: keywords in orange, comments in grey, and literals in blue.

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

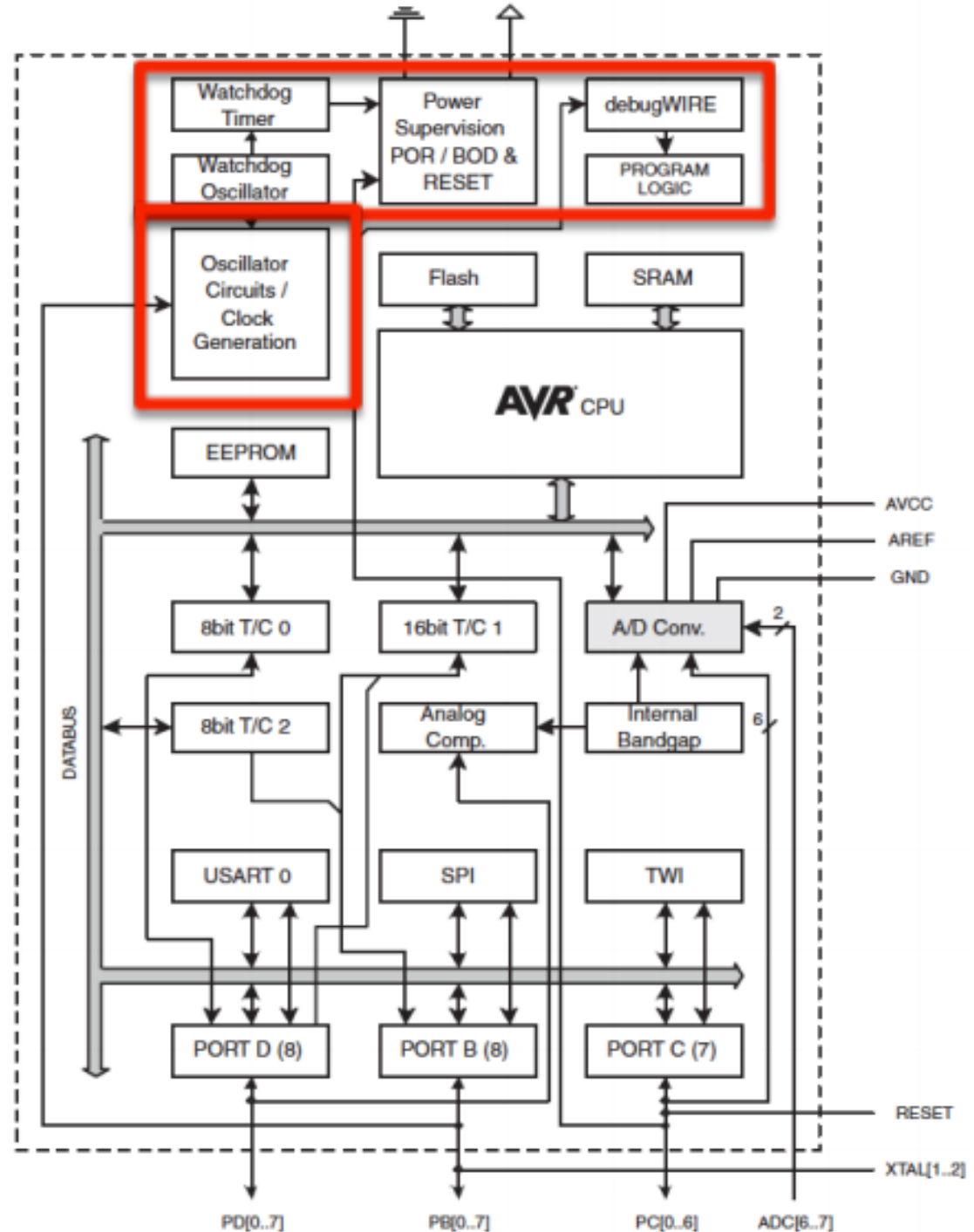
This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```


Microcontroladores

- Vejamos uma visão geral do Microcontrolador Atmega328, do Arduino Uno
- Potência e Clock (foge do escopo do curso)



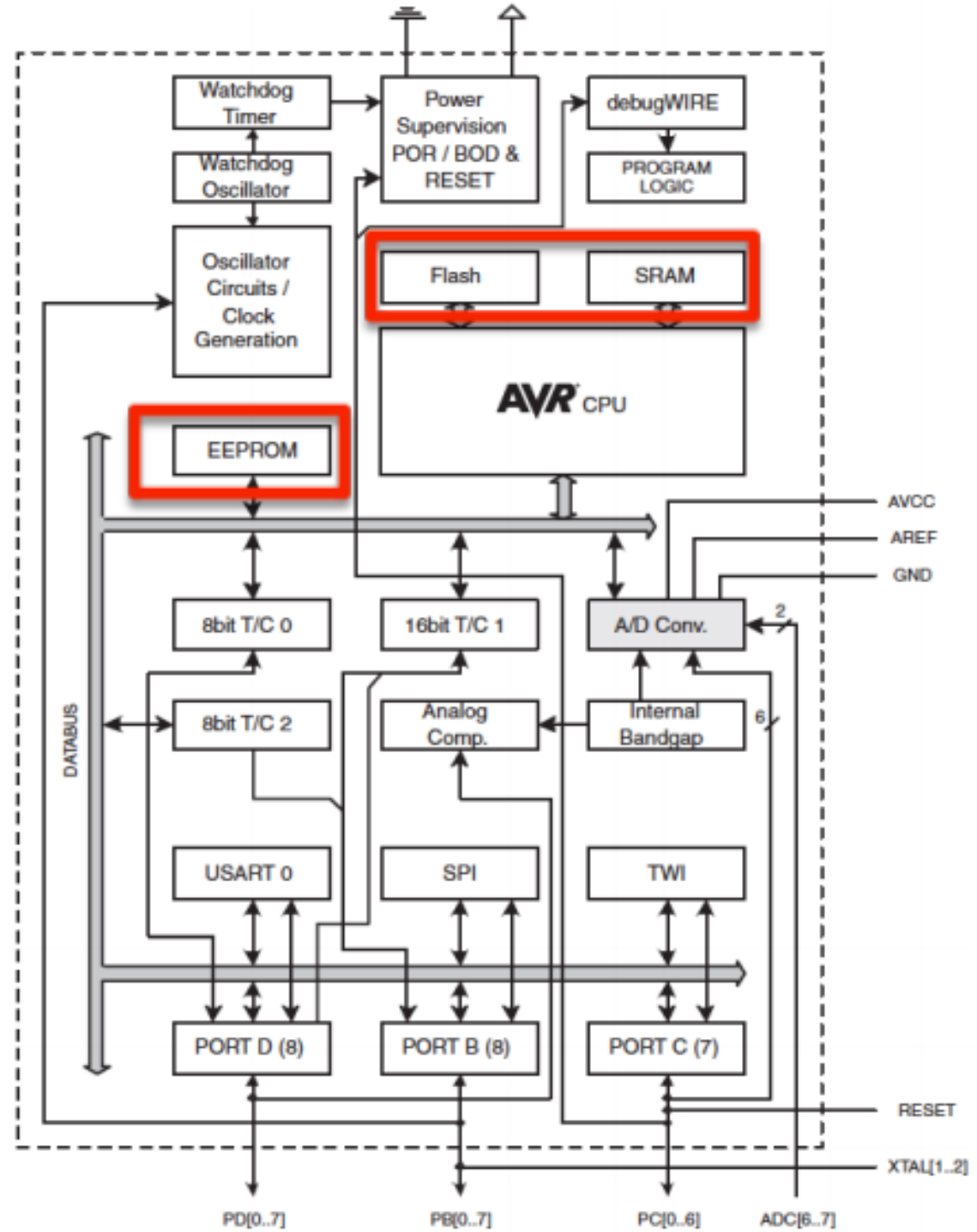
Microcontroladores

- Memórias

Flash: memória de programa

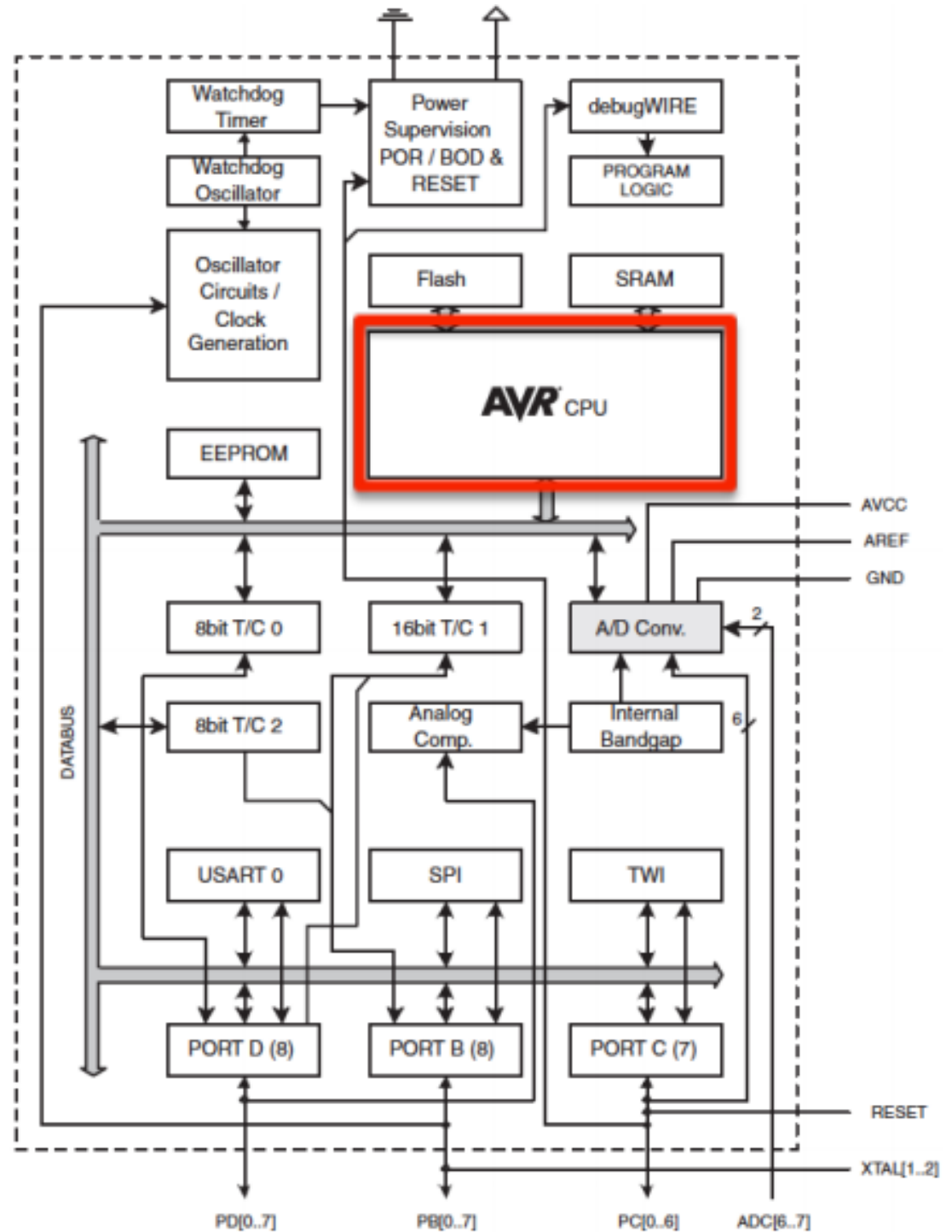
EEPROM: somente leitura

- SRAM: valores temporários, stack. Volátil (curto prazo)!



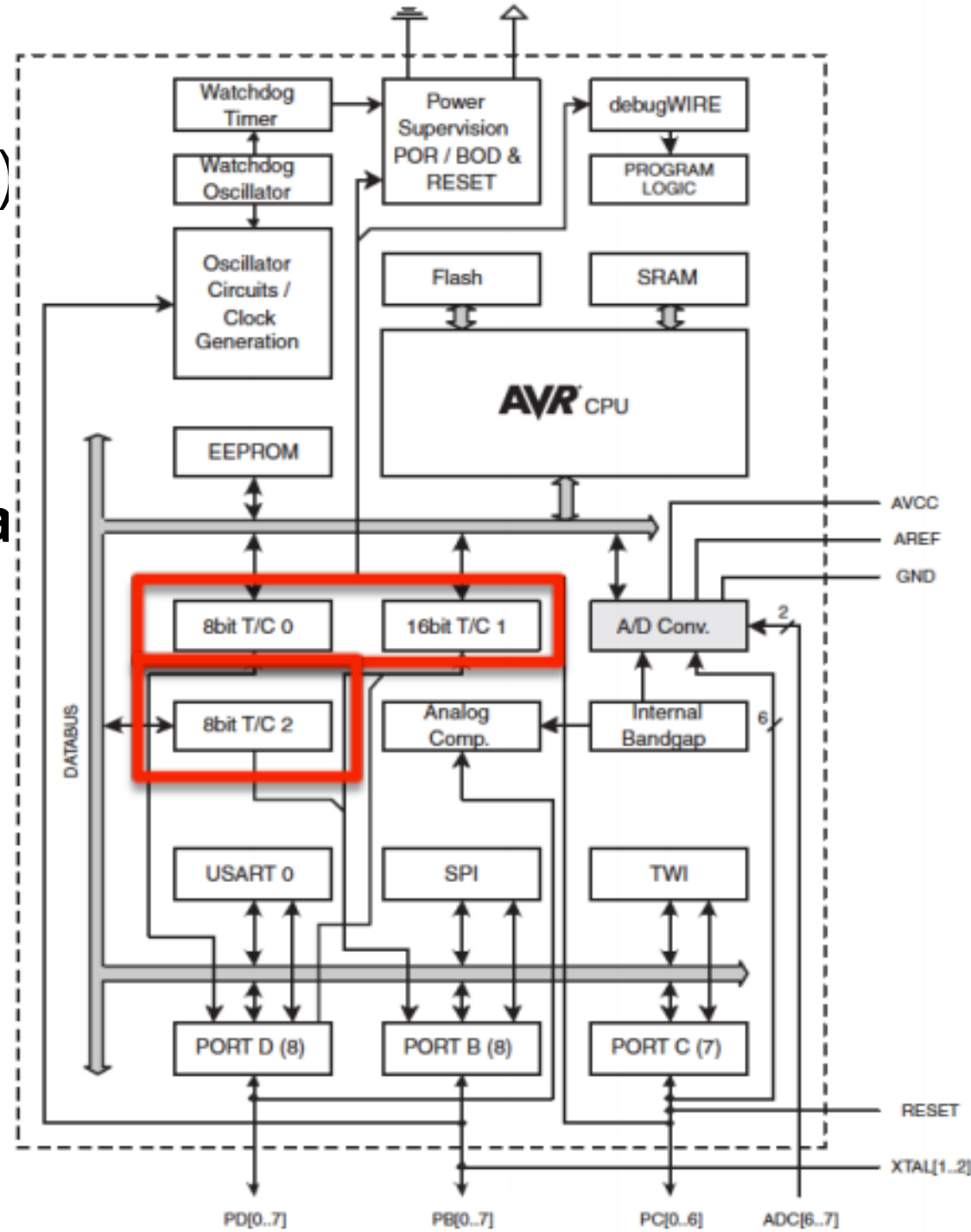
Microcontroladores

- CPU (discutimos anteriormente, mais detalhes adiante)



Microcontroladores

- Timers (temporizadores)
Escolhe taxa de clock
Gera sinal PWM (Pulse Width Modulation, ou **modulação por largura de pulso**)
- PWM usa um sinal digital para criar saída de potência variável



Microcontroladores

- Funções de timer do Arduino

`void delay(ms)`: espera tantos milissegundos

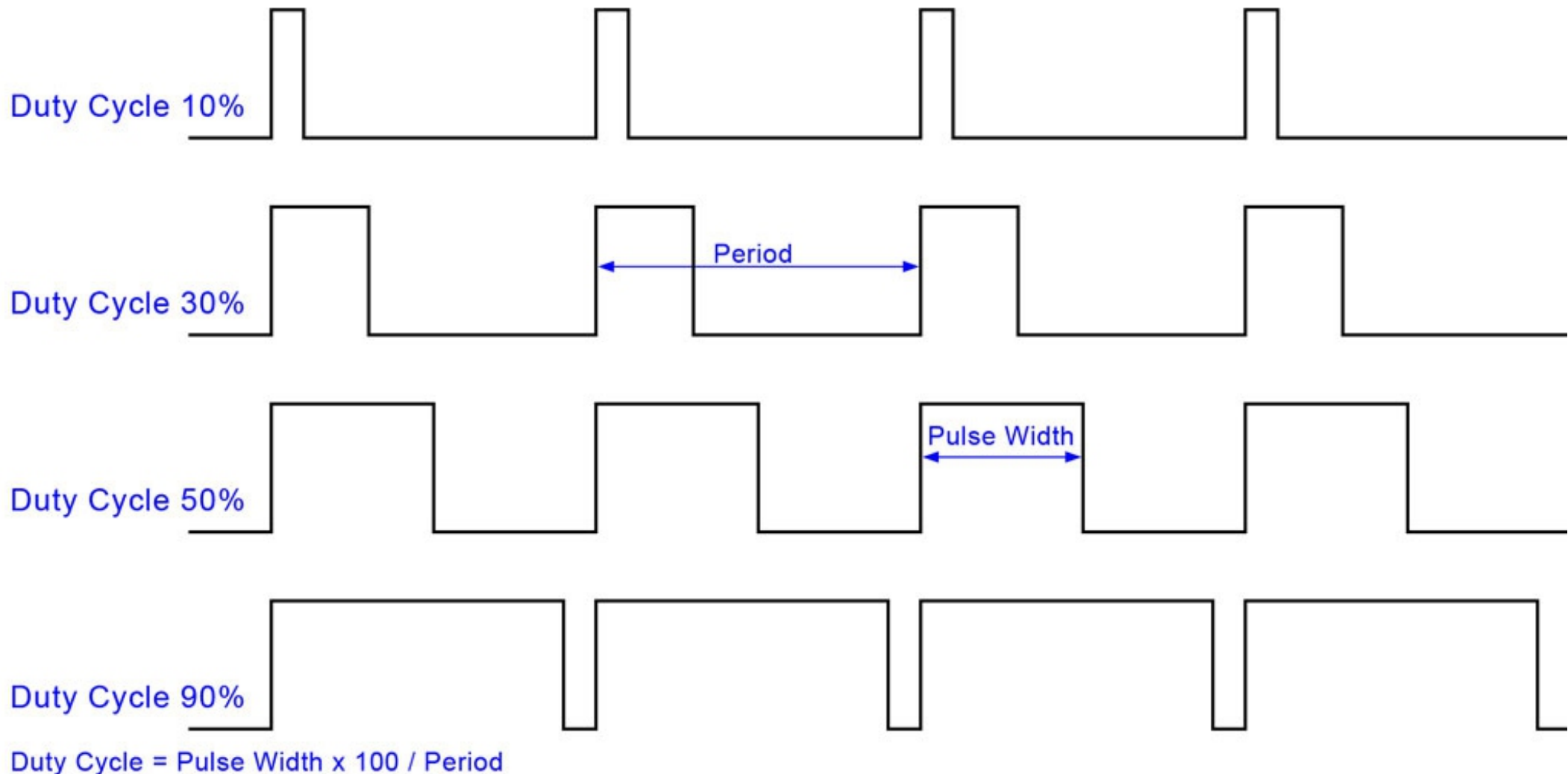
`void delay(us)`: espera tantos microssegundos

`unsigned long millis()`: retorna milissegundos
passados desde início do programa

`unsigned long micros()`: retorna microssegundos
passados desde início do programa

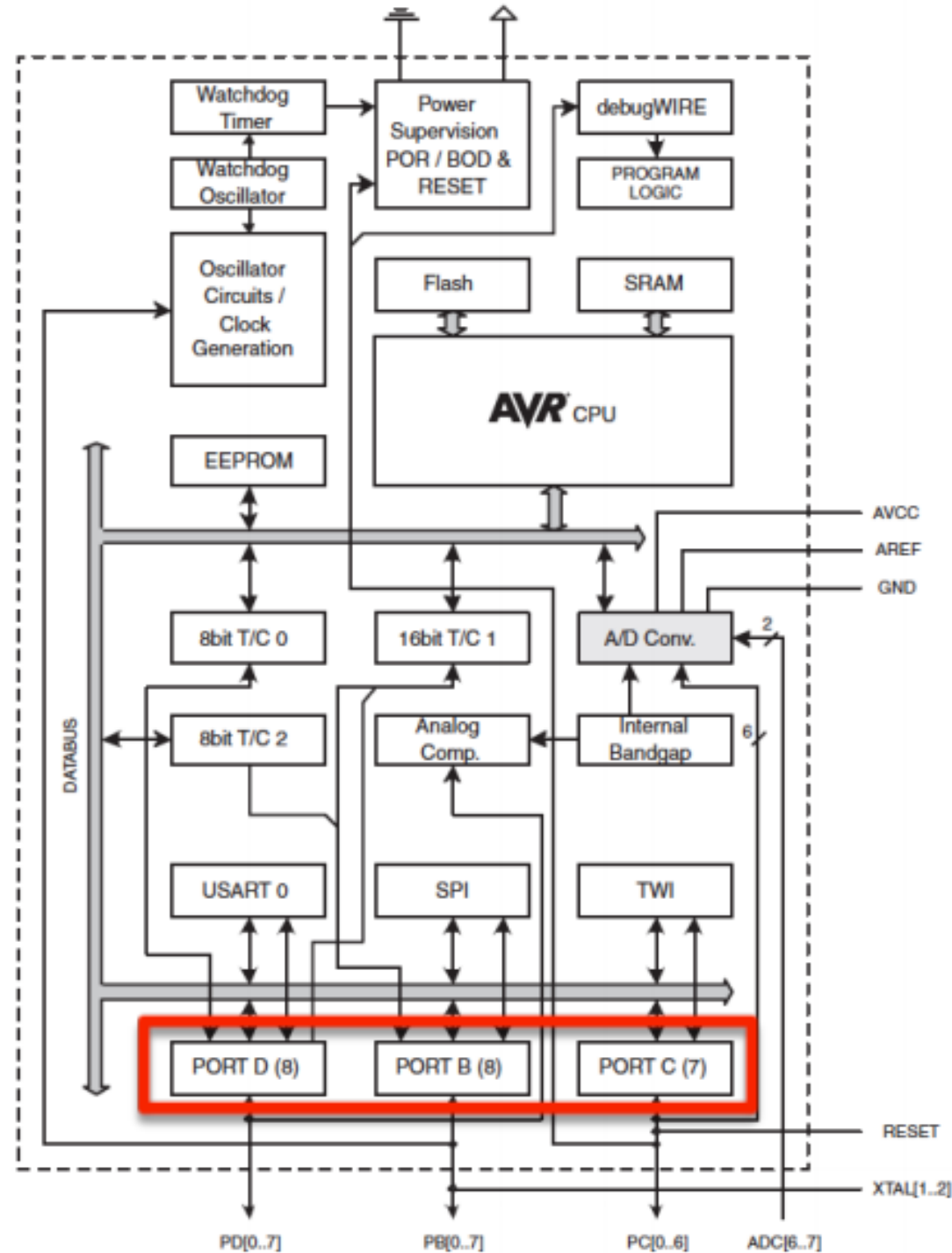
Microcontroladores

- PWM (variando potência através da largura do pulso)



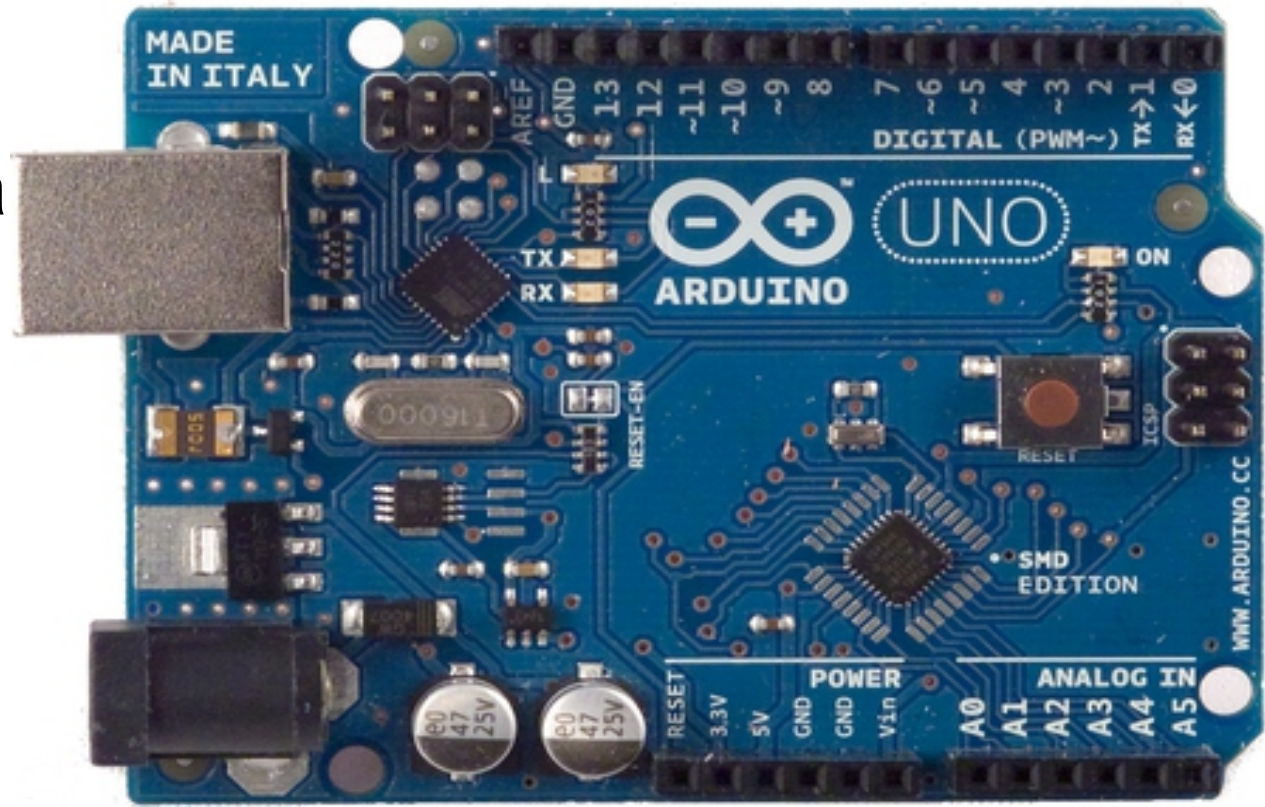
Microcontroladores

- Interface com os pinos
 - Cada pino é “programável”
- Alguns pinos são especiais
 - Analog vs Digital
 - Clocks
 - Reset



Microcontroladores

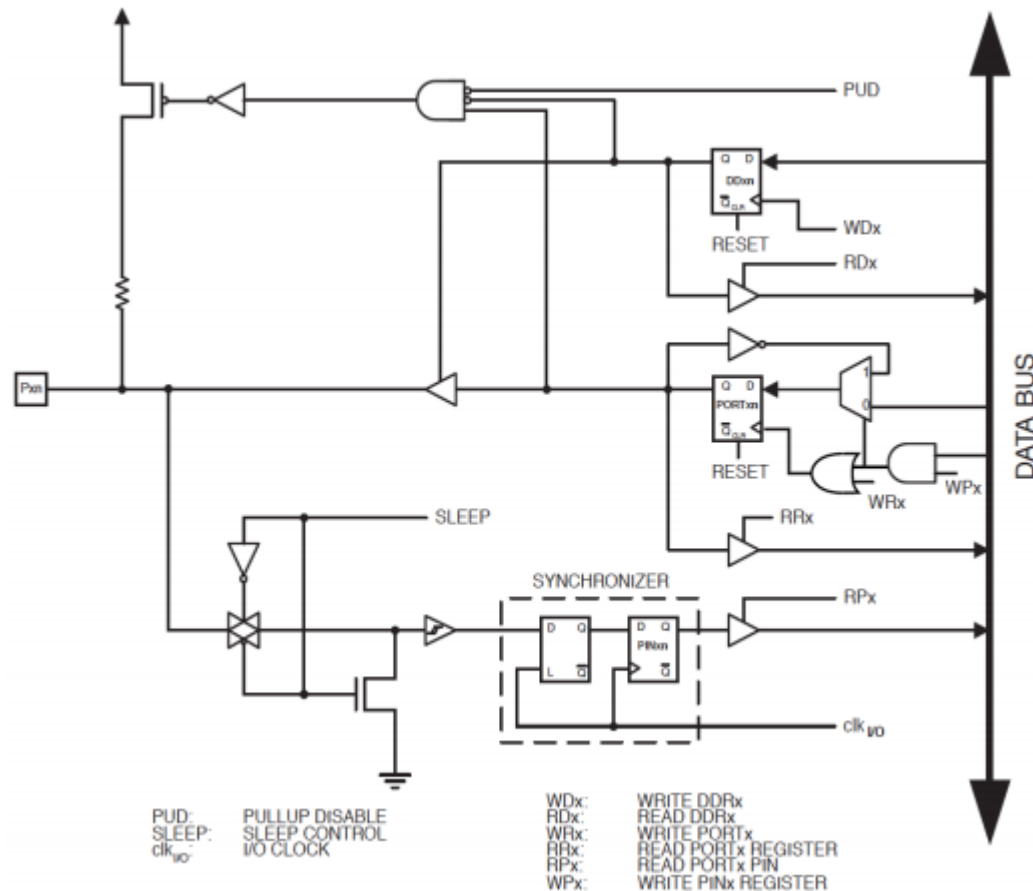
- Interface com os pinos
 - Cada pino é “programável”
 - Entrada ou Saída (E/S)
- Alguns pinos são especiais
 - Analog vs Digital
 - Clocks
 - Reset

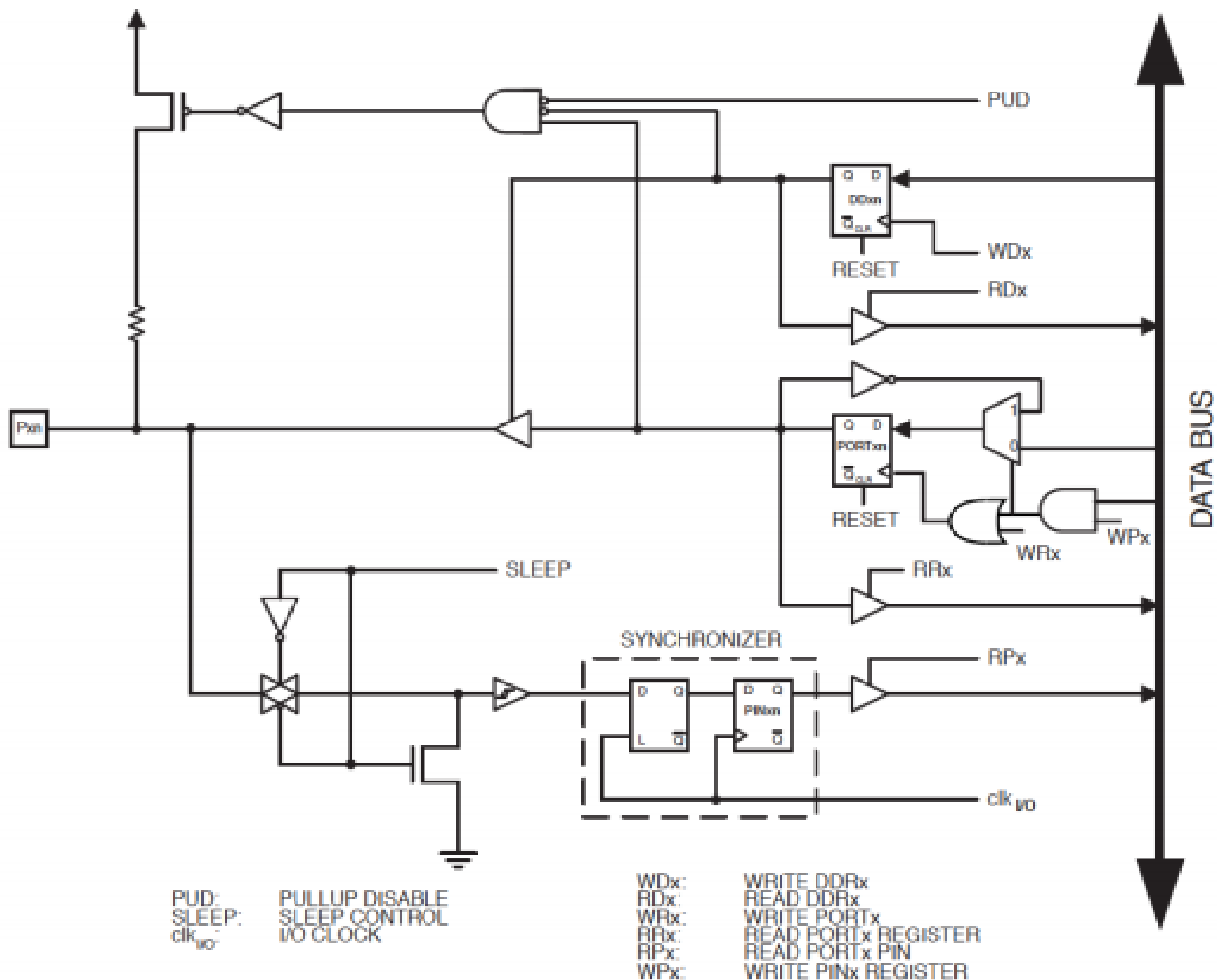


Microcontroladores

- Curiosidade:

Um pino por dentro. Os flip-flops controlam se pino é E ou S, e armazenam valor sendo escrito ou lido



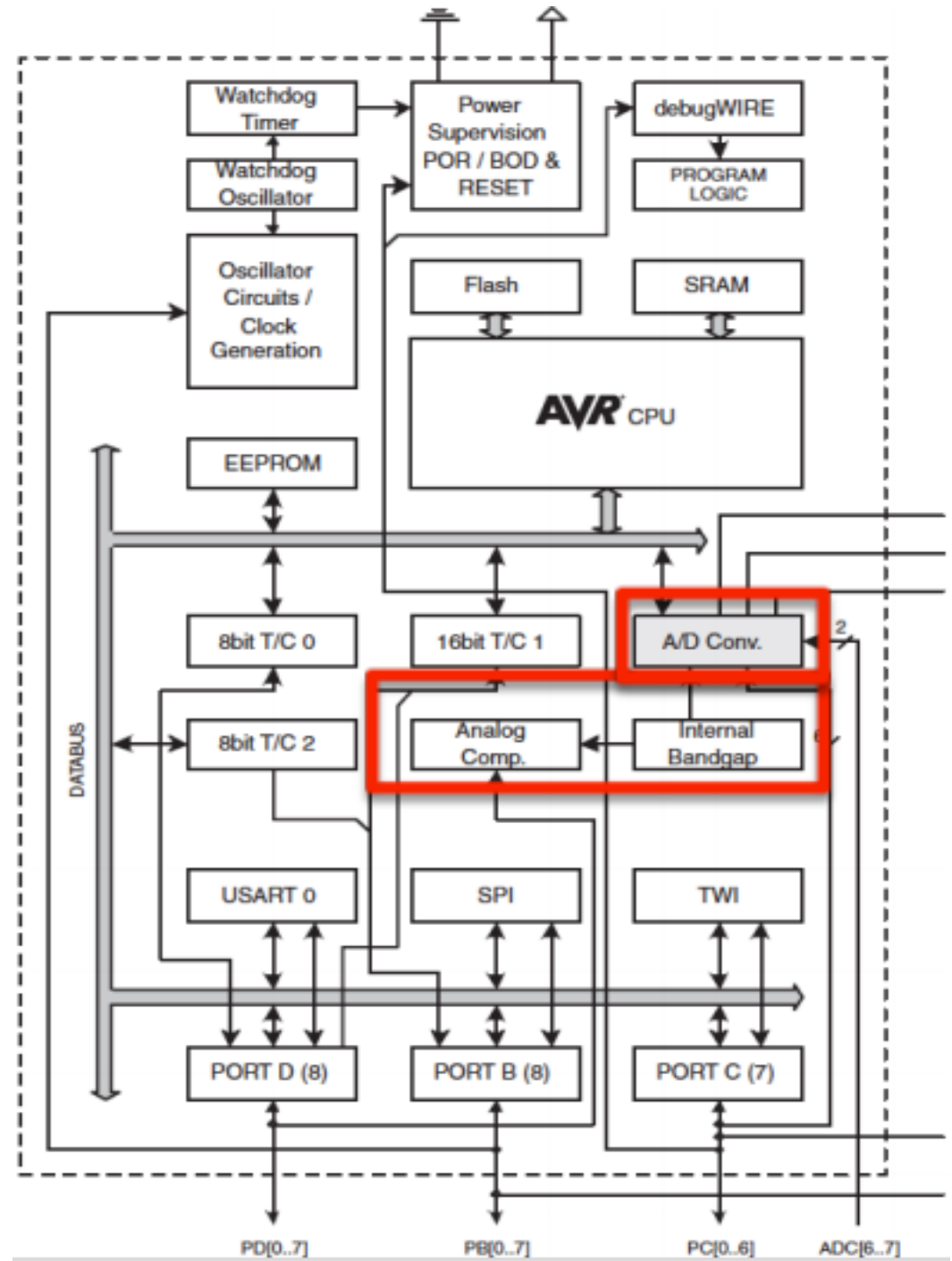


Microcontroladores

- Instruções pro Arduino lidar com pinos
 - `void pinMode(pin, mode)`: determina se pino “pin” é modo entrada ou saída (input ou output)
 - `void digitalWrite(pin, value)`: escreve valor “value” no pino “pin”, se pin é saída
 - `int digitalRead(pin)`: se “pin” é entrada, lê o valor do pino “pin”
 - `int analogRead(pin)`: lê valor de 0 a 5 volts e transforma em número de 0-1024. Bom para leitura de sensores
 - `void analogWrite (pin, value)`: escreve um valor value (0-255), gerando um sinal PWM na saída

Microcontroladores

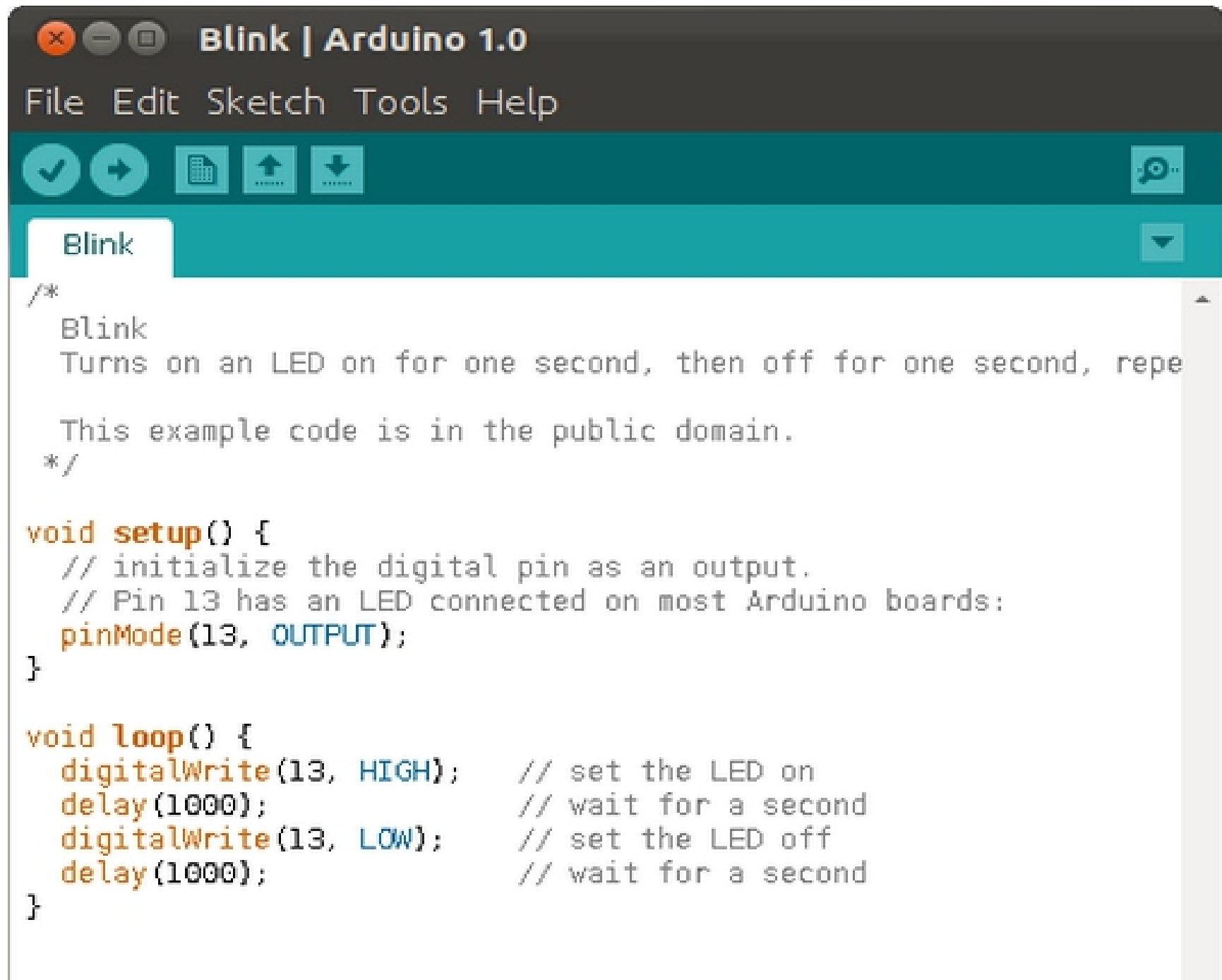
- Conversor das entradas analógicas



Microcontroladores

- Linguagem da Arduino IDE. Estrutura de um programa (*bare minimum*)
 - Cabeçalho: declarações, includes, etc.
 - **setup()**
 - Executa uma vez no início da execução
 - **loop()**
 - Executa constantemente, em laço
- Vejamos novamente o programa “blink” (pisca LED)

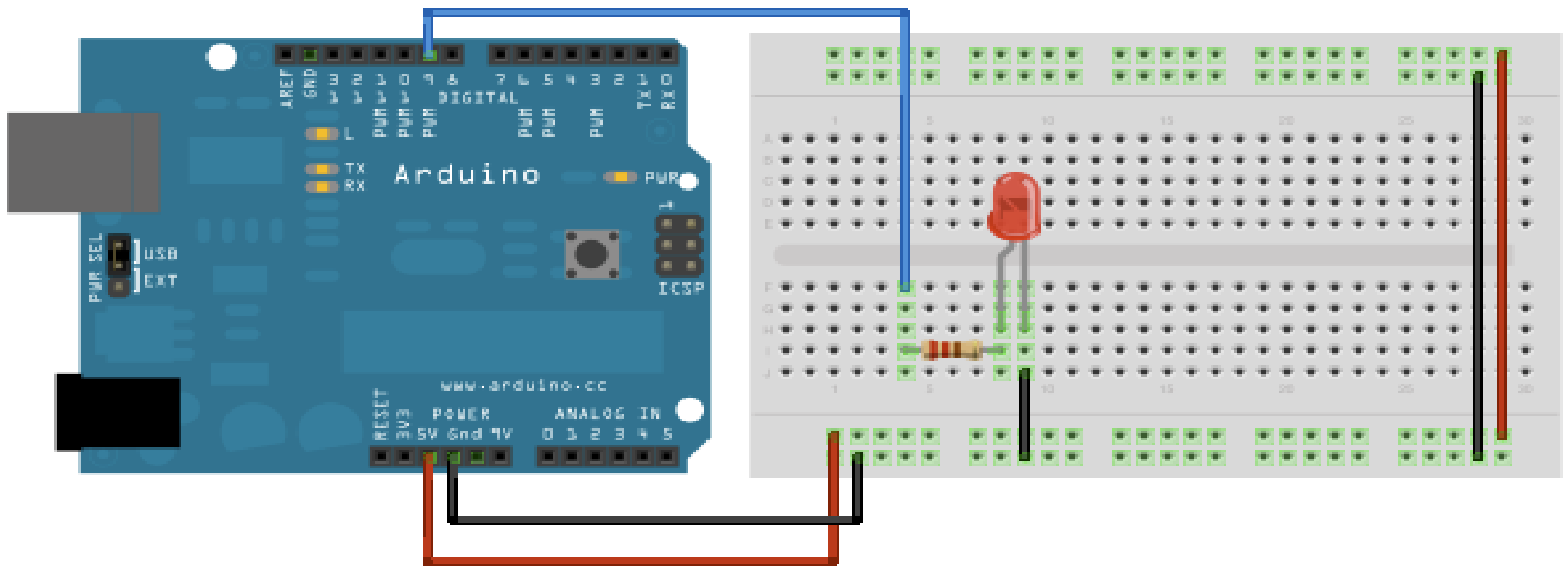
Microcontroladores

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for a checkmark, a right arrow, a document, an upload arrow, a download arrow, and a speech bubble. A tab labeled "Blink" is active. The code editor displays the following text:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repe  
  
  This example code is in the public domain.  
*/  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH);    // set the LED on  
  delay(1000);               // wait for a second  
  digitalWrite(13, LOW);    // set the LED off  
  delay(1000);               // wait for a second  
}
```

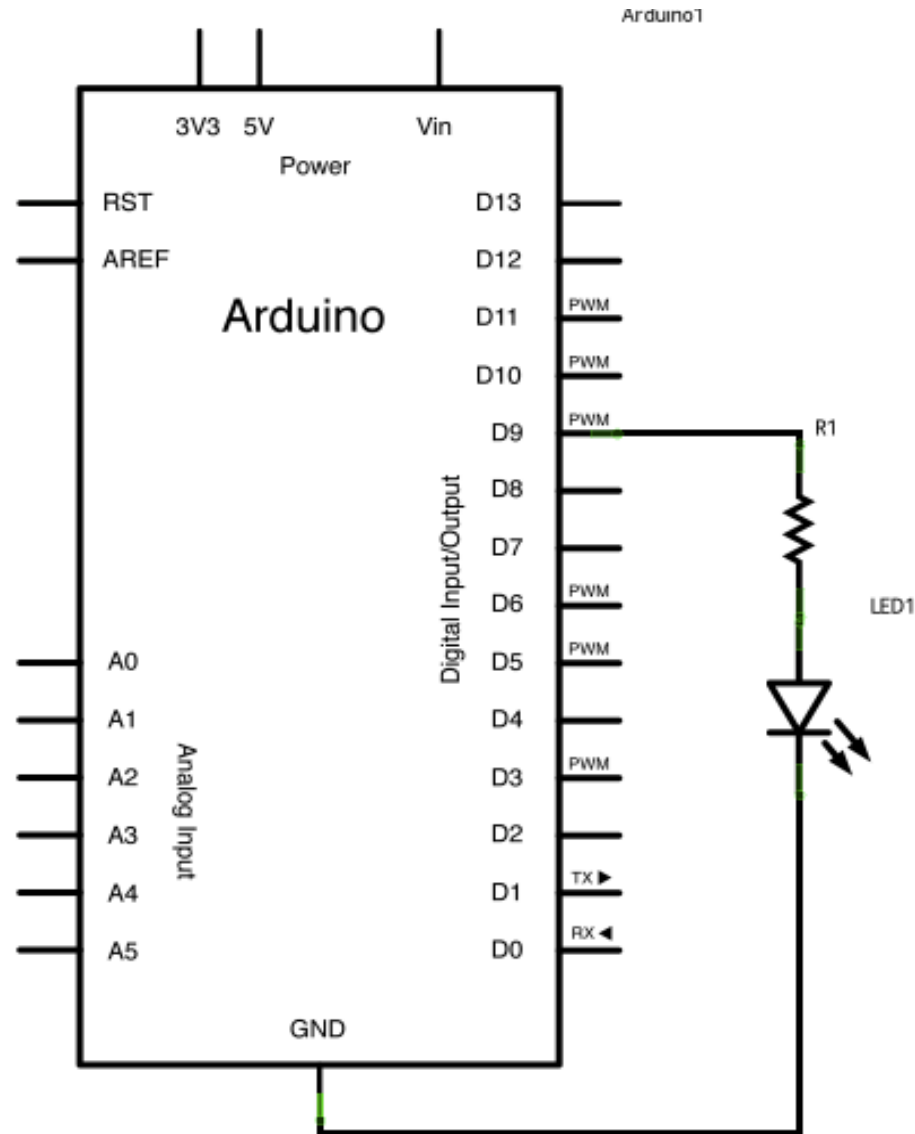
Microcontroladores

- Uma vez programado o Arduino, como executar?
Precisamos apenas ligar o LED no pino 13



Microcontroladores

- O mesmo desenho, como esquemático:



Microcontroladores

- Vejamos um código um pouco mais complexo, “Fade”
- Apaga e acende um LED lentamente
 - Note que o brilho do LED depende da quantidade de corrente chegando nele!

```
/*  
  Fade
```

*This example shows how to fade an LED on pin 9
using the analogWrite() function.*

*This example code is in the public domain.
/

```
int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

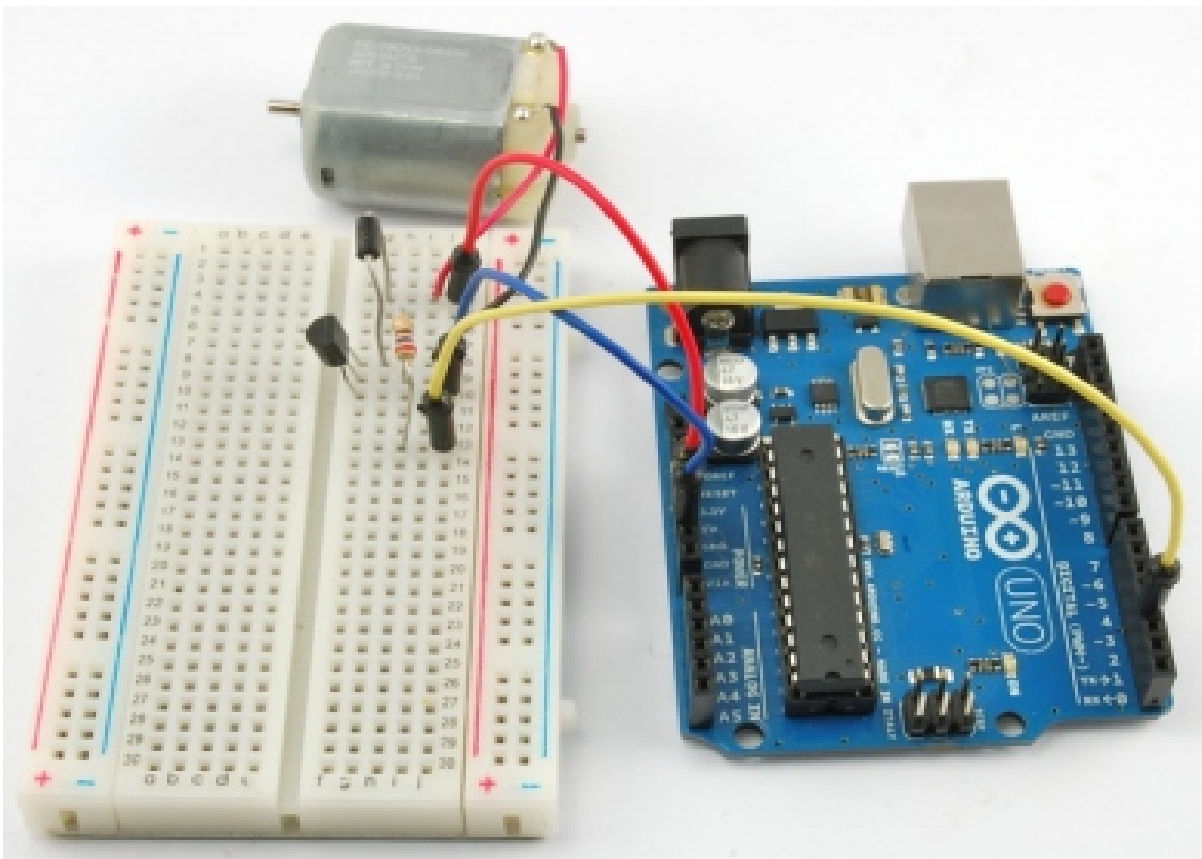
// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

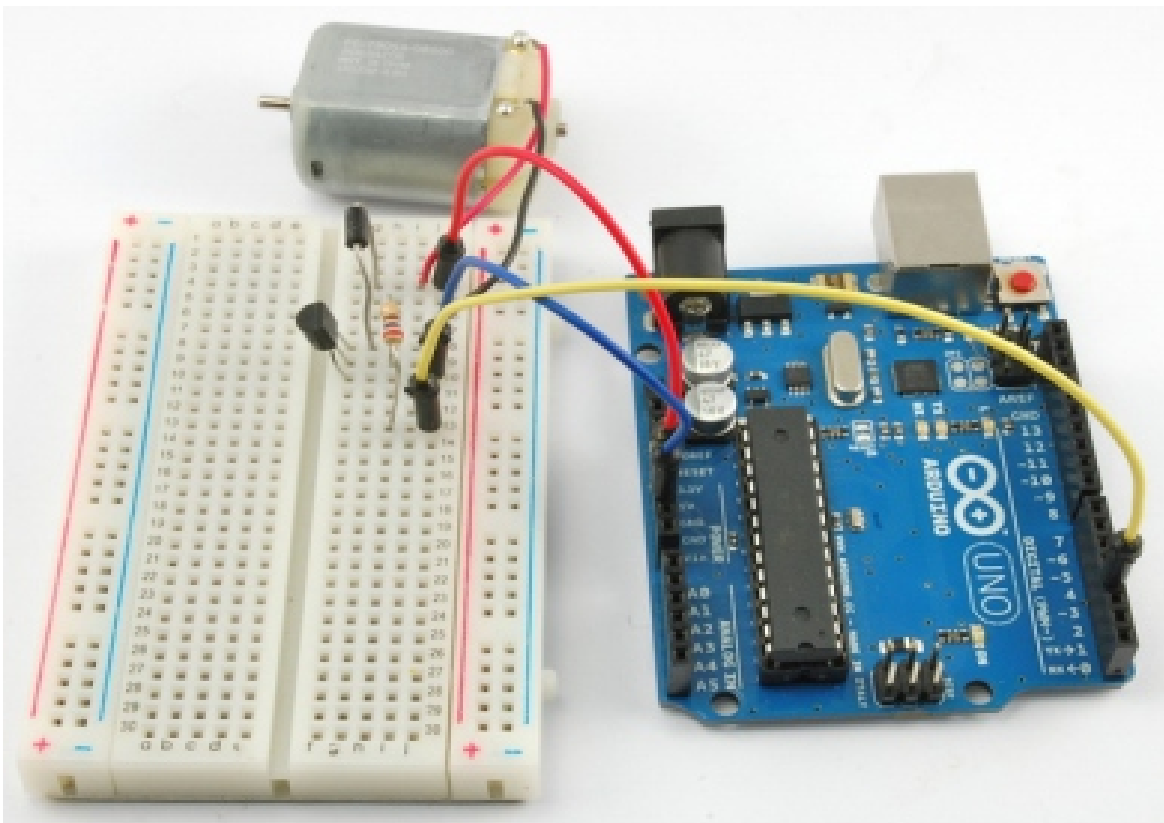
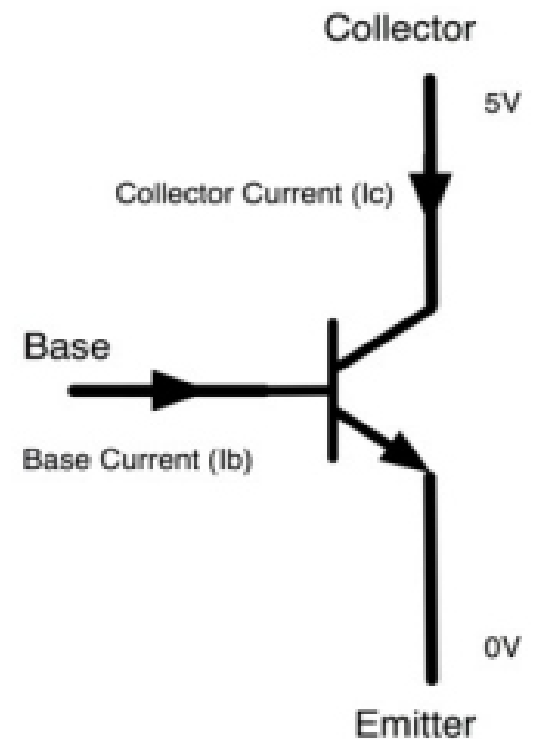
Microcontroladores

- Dica de design: A corrente que o Arduino provê é suficiente para acender um LED, mas não pra muito mais que isso. E se quisermos controlar um dispositivo mais potente? Como um motor robusto?



Microcontroladores

- E se quisermos controlar um dispositivo mais potente? Como um motor robusto?
- Solução: usar um transistor (como o PN2222)! (uma pequena corrente na base controla uma grande corrente do coletor p/ o emissor)



Microcontroladores

- Mais funções da “biblioteca” do Arduino

<http://arduino.cc/en/Reference/HomePage>

Microcontroladores

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Microcontroladores

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Microcontroladores

Math

- `min()`
- `max()`
- `abs()`
- `constrain()`
- `map()`
- `pow()`
- `sqrt()`

Trigonometry

- `sin()`
- `cos()`
- `tan()`

Bits and Bytes

- `lowByte()`
- `highByte()`
- `bitRead()`
- `bitWrite()`
- `bitSet()`
- `bitClear()`
- `bit()`

External Interrupts

- `attachInterrupt()`
- `detachInterrupt()`

Microcontroladores

- Interrupção:

Algum evento que muda o fluxo do sistema.
Interrompe execução atual, trata evento e volta.

Evita problemas de timing (porque?)

- Como você faria um programa que roda **até que o usuário aperte um botão?**
- E se estou fazendo um controle de air-bag e no lugar de **apertar um botão** é **detectar batida?**
Precisamos atender o problema imediatamente!

Microcontroladores

- Com interrupção:

`attachInterrupt(interrupt, ISR, mode)`

- Interrupt: número da interrupção (pino onde ocorreu)
- ISR: função para chamar quando acontecer a interrupção
- Mode: condição que ativa interrupção (LOW, HIGH, CHANGE, RISING, FALLING)
- Exemplo no slide a seguir

Microcontroladores

```
int pin = 13;
volatile int state = LOW;

void setup()
{
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
}

void loop()
{
    digitalWrite(pin, state);
}

void blink()
{
    state = !state;
}
```