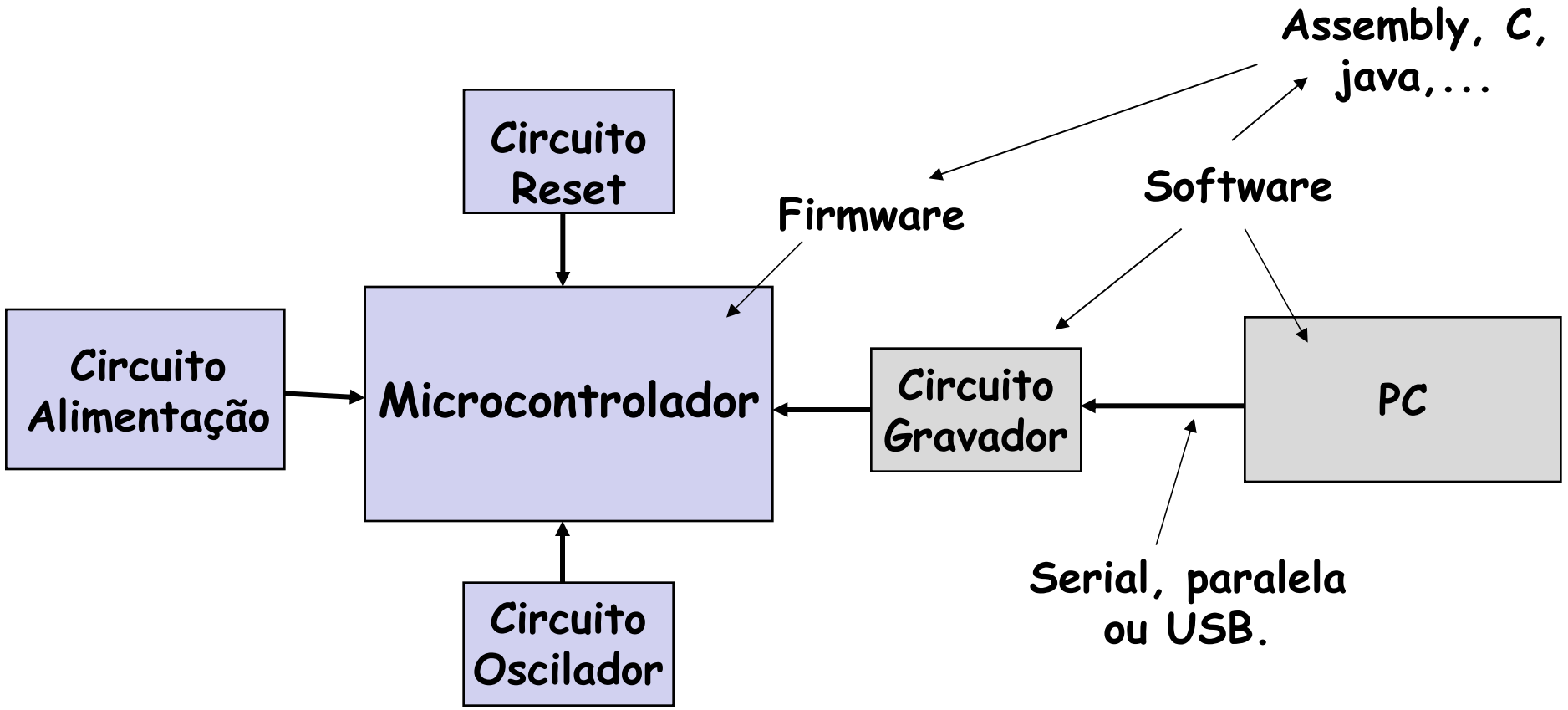
Introdução aos Sistemas Microcontrolados

Jadsonlee da Silva Sá

Jadsonlee.sa@univasf.edu.br
www.univasf.edu.br/~jadsonlee.sa

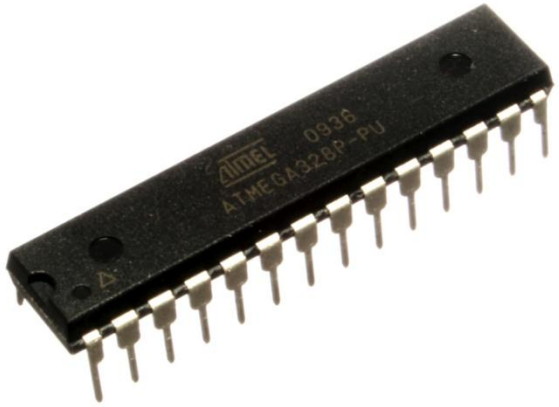
Introdução aos Sistemas Microcontrolados



Microcontrolador AVR da ATMEL

- ✓ AVR – Não é um acrônimo, segundo a ATMEL.
- ✓ É um microcontrolador RISC com uma arquitetura Harvard modificada de 8 bits.
- ✓ Na família de uC AVR, há quatro grupos:
  - ✓ tinyAVR (série ATtiny);
  - ✓ **megaAVR (série ATmega);**
  - ✓ XMEGA (série ATxmega);
  - ✓ Atmel At94k (Série FPSLIC - FPGA).

ATmega328P



Características

- ✓ **Arquitetura RISC avançada.**
  - Conjunto de 131 instruções - a maioria com execução em um ciclo de clock;
  - Conjunto de 32 registradores de propósito geral de 8 bits;
  - Frequência de operação de até 20 MHz.

Características

- ✓ **Memórias.**
  - Memória de programa flash – 32 Kbytes;
  - Memória de dados SRAM – 2 Kbytes;
  - Memória EEPROM – 1 Kbyte.

Características

- ✓ **Periféricos.**
  - Dois timer/counters de 8 bits com prescaler separado e modo de comparação.
  - Um timer/counter de 16 bits com prescaler separado, modo de comparação e captura.
  - Um contador em tempo real com oscilador separado.
  - Seis canais PWM (Pulse Width Modulation).
  - ADC com 6 canais e 10 bits de resolução (PDIP).

Características

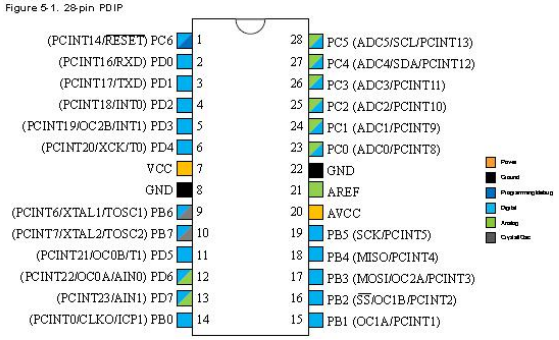
- ✓ **Periféricos.**
  - Duas SPI (Serial Peripheral Interface) mestre/escravo.
  - Uma USART programável.
  - Uma interface serial 2-wire (I2C da Phillips).
  - Watchdog timer programável com oscilador on-chip separado.
  - Um comparador analógico on-chip.
  - Interrupção e wake-up na mudança de pino.

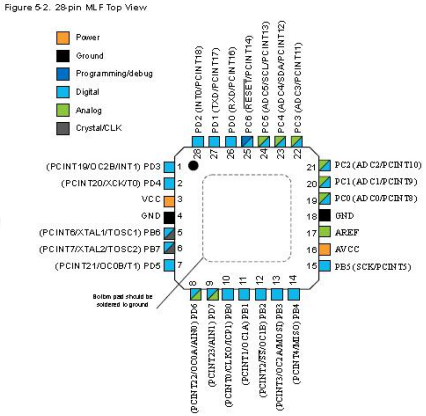
Características

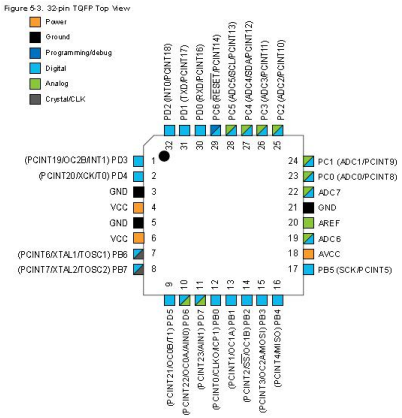
- ✓ **Especiais.**
  - Power-on reset e detecção de brown-out programável.
  - Oscilador interno calibrado.
  - Fontes de interrupção externa e interna.
  - Seis modos sleep: idle, redução de ruido do ADC, power-save, power-down, standby e standby estendido.

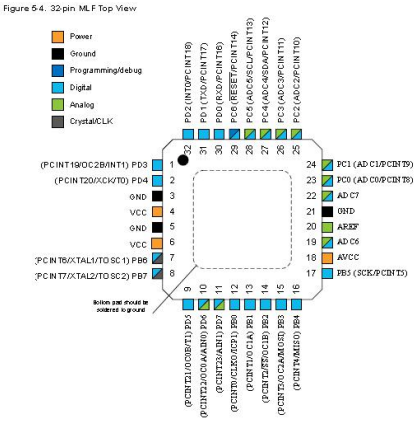
Características

- ✓ E/S e packages.
  - Conjunto com 23 pinos de entrada/saída programáveis.
  - Quatro packages.









Características

- ✓ **Tensão elétrica de operação.**
  - 1,8 a 5,5 V
- ✓ **Faixa de temperatura.**
  - -40 a 105 °C
- ✓ **Speed grade.**

0 - 4MHz @ 1.8 - 5.5V
0 - 10MHz @ 2.7 - 5.5V
0 - 20MHz @ 4.5 - 5.5V

Características

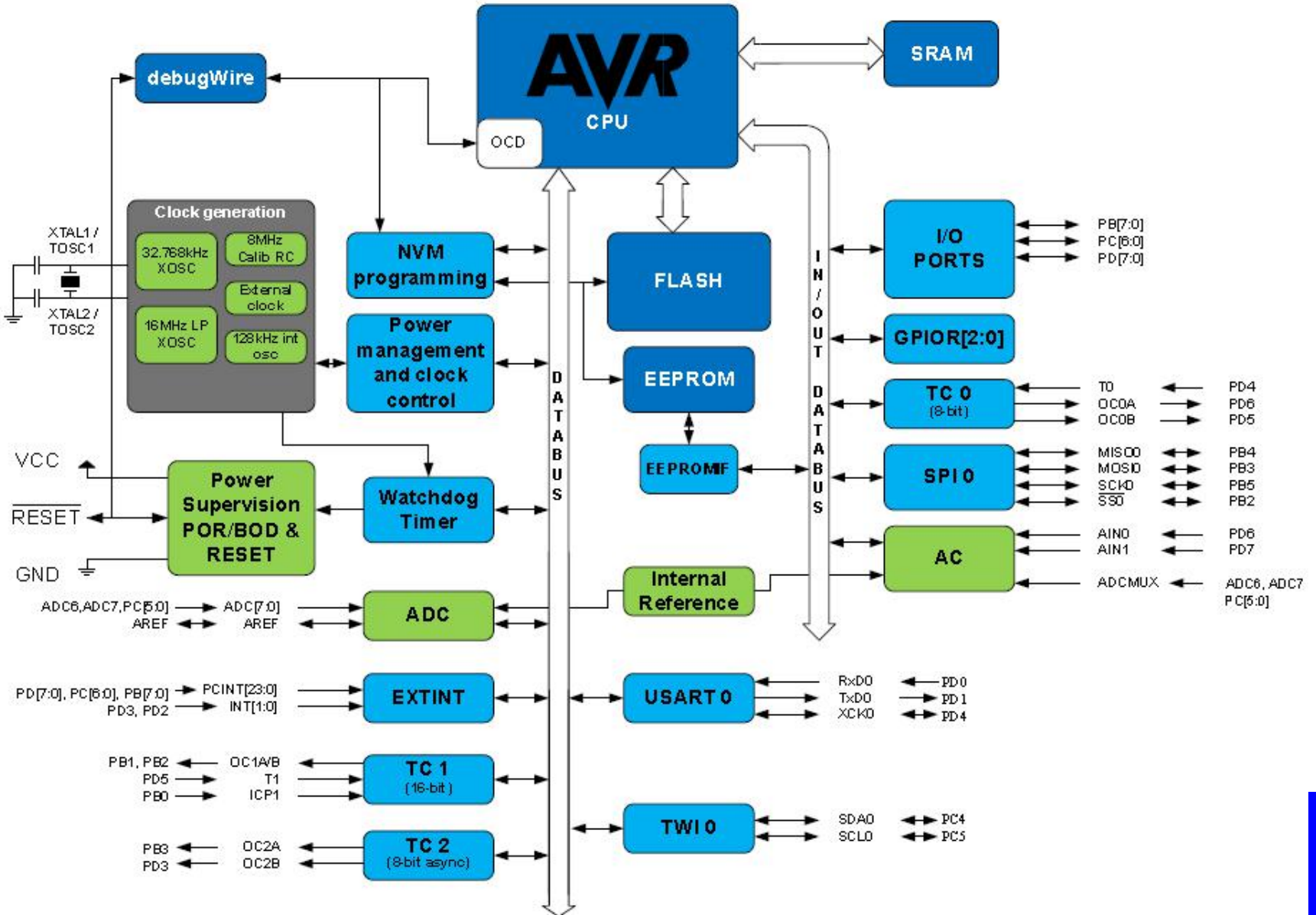
- ✓ **Consumo em 1 MHz, 1,8 V e 25 °C.**

Active Mode: 0.2mA

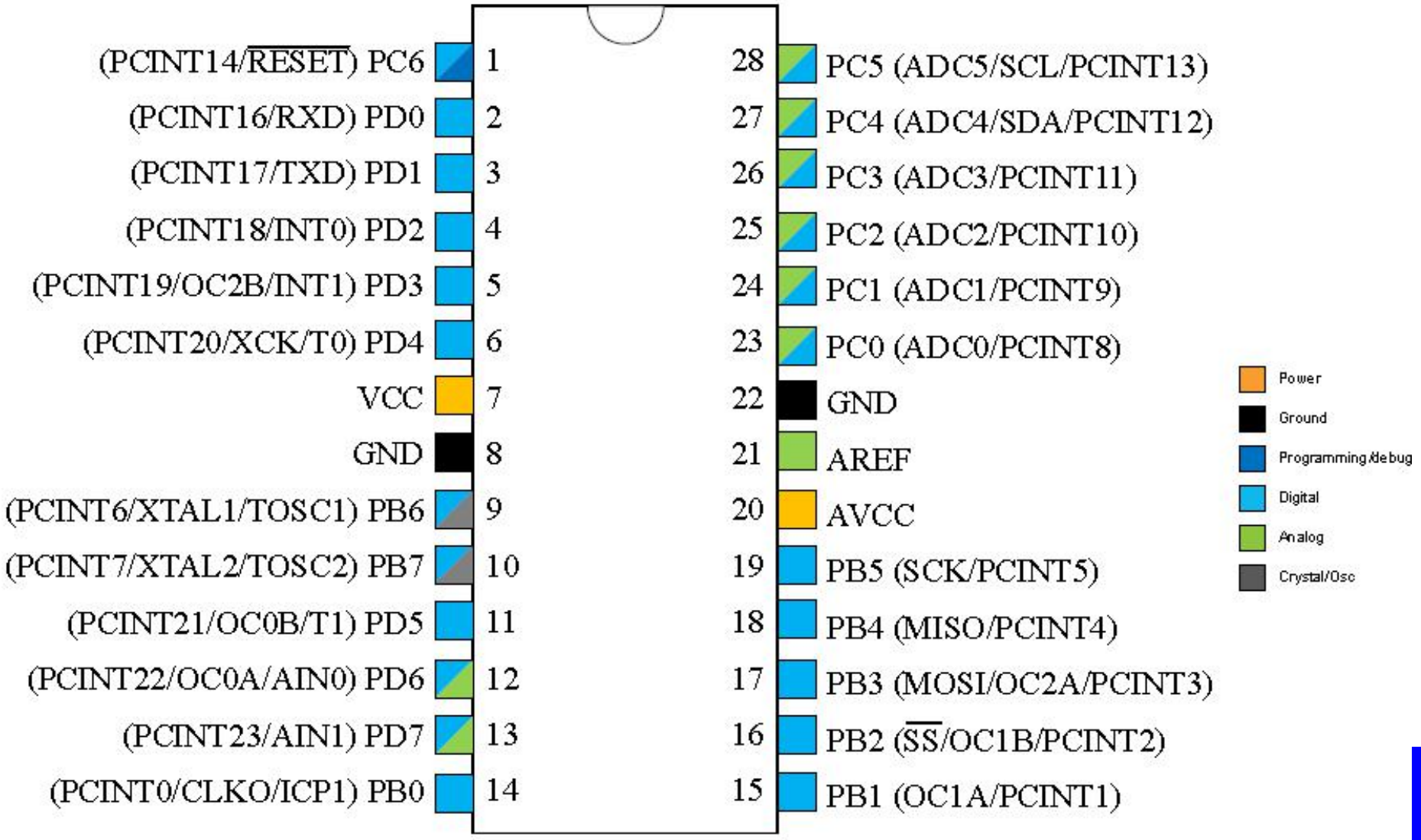
Power-down Mode: 0.1µA

Power-save Mode: 0.75µA (Including 32kHz RTC)

Diagrama de Blocos

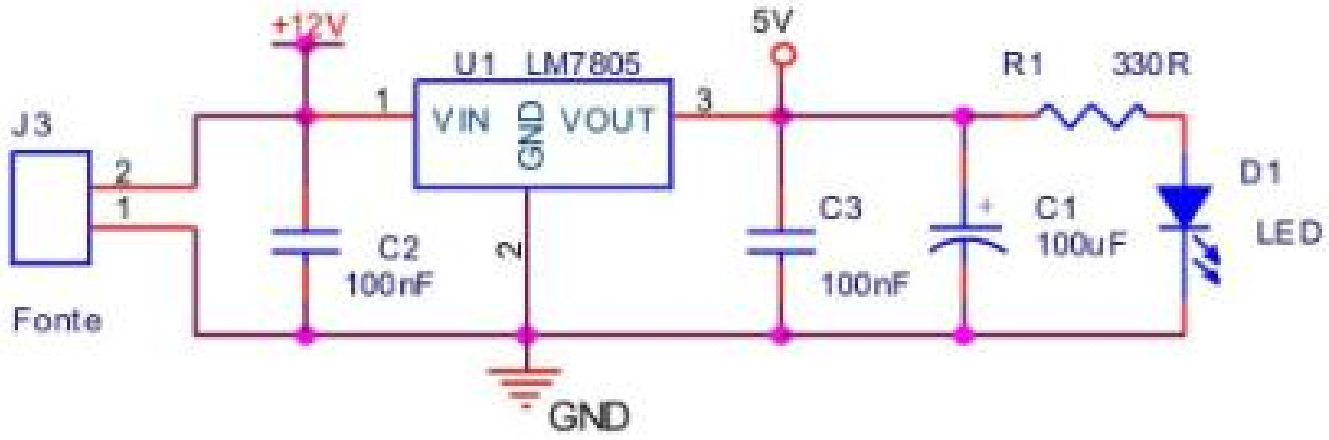


Configurações dos Pinos



Circuito Tensão de Alimentação

- ✓ Normalmente, utiliza-se um circuito regulador de tensão, para fornecer a tensão de alimentação ao microcontrolador.
  - Ex.: circuito para µC 5 V.

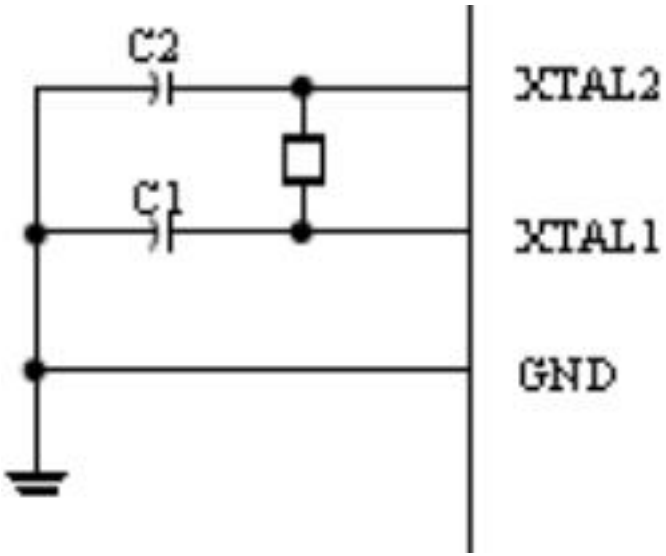


  - Conexão dos pinos Vcc, AVcc e GND – De Vcc para GND coloque um capacitor de 100 nF.

Fontes de Clock

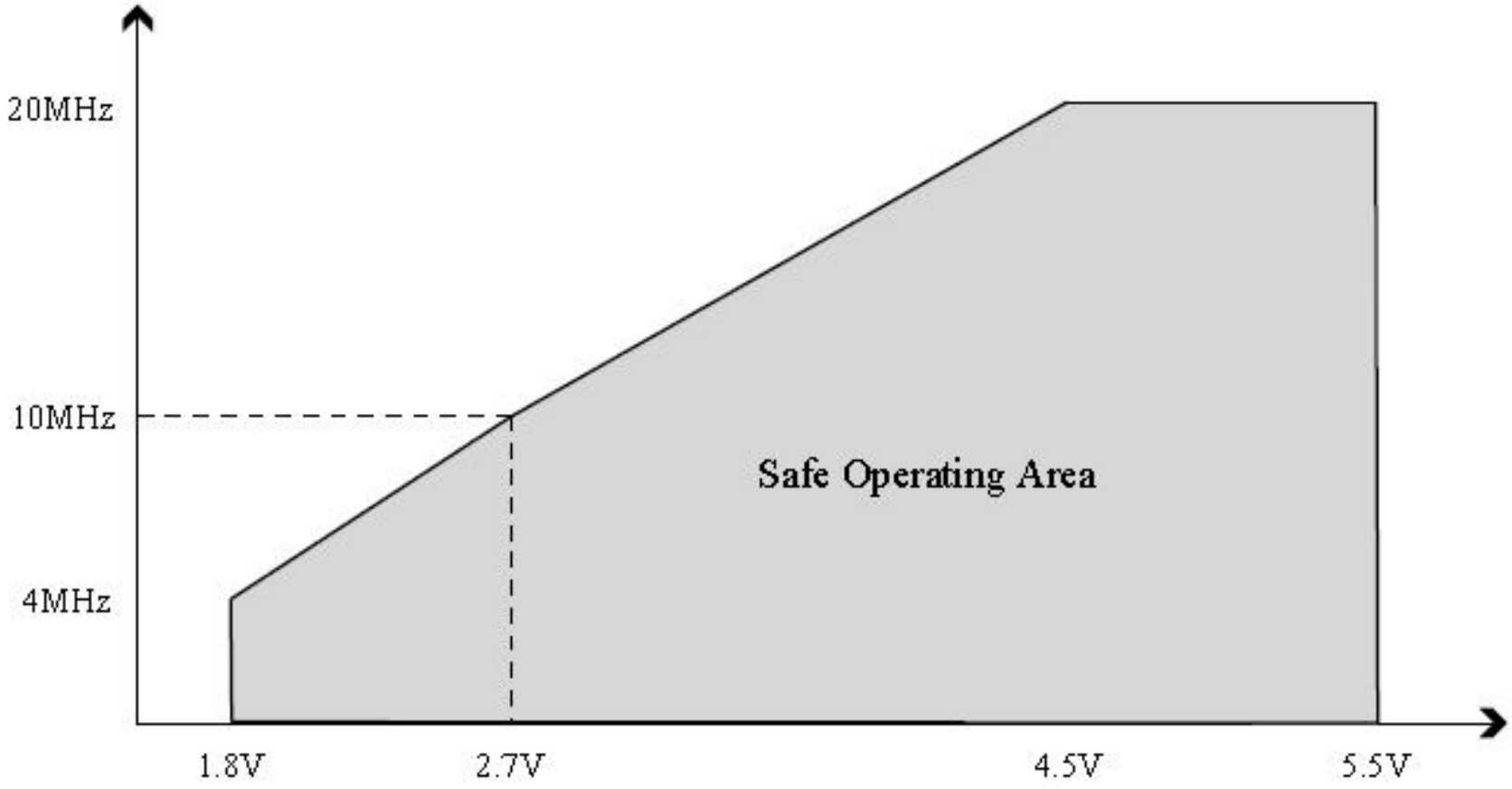
- ✓ Há várias possíveis fontes de clock no ATmega328P.
  - Circuito oscilador externo com cristal;
  - Circuito RC interno;
  - Clock externo.
- ✓ Geralmente, circuitos osciladores externos são usados para gerar o sinal de relógio.

Circuito Oscilador Externo com Cristal



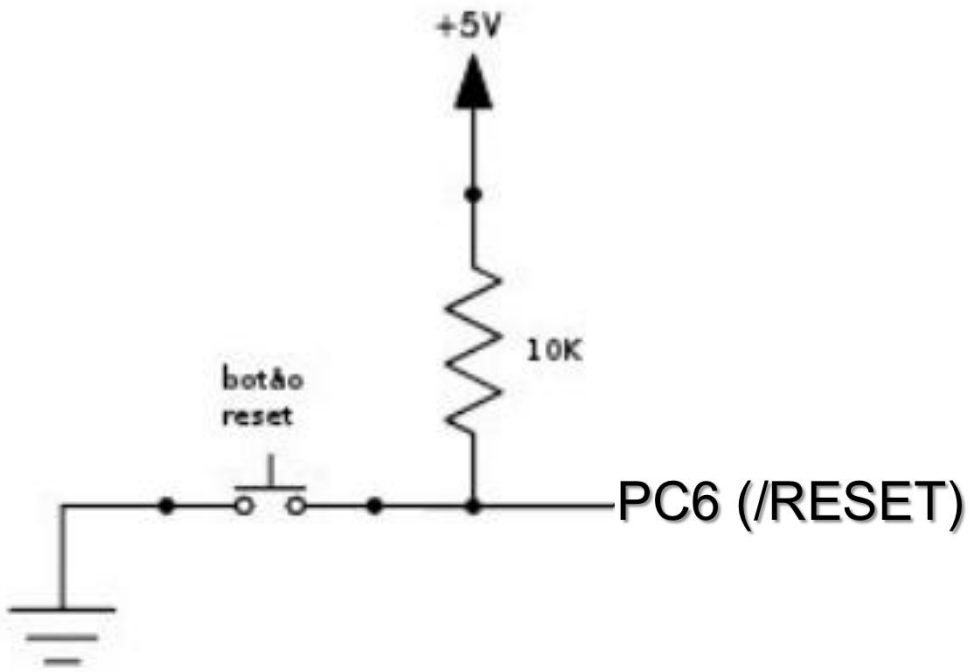
- ✓ Os valores do cristal e dos capacitores dependem do modo de operação da fonte de clock.

Tensão x Frequência

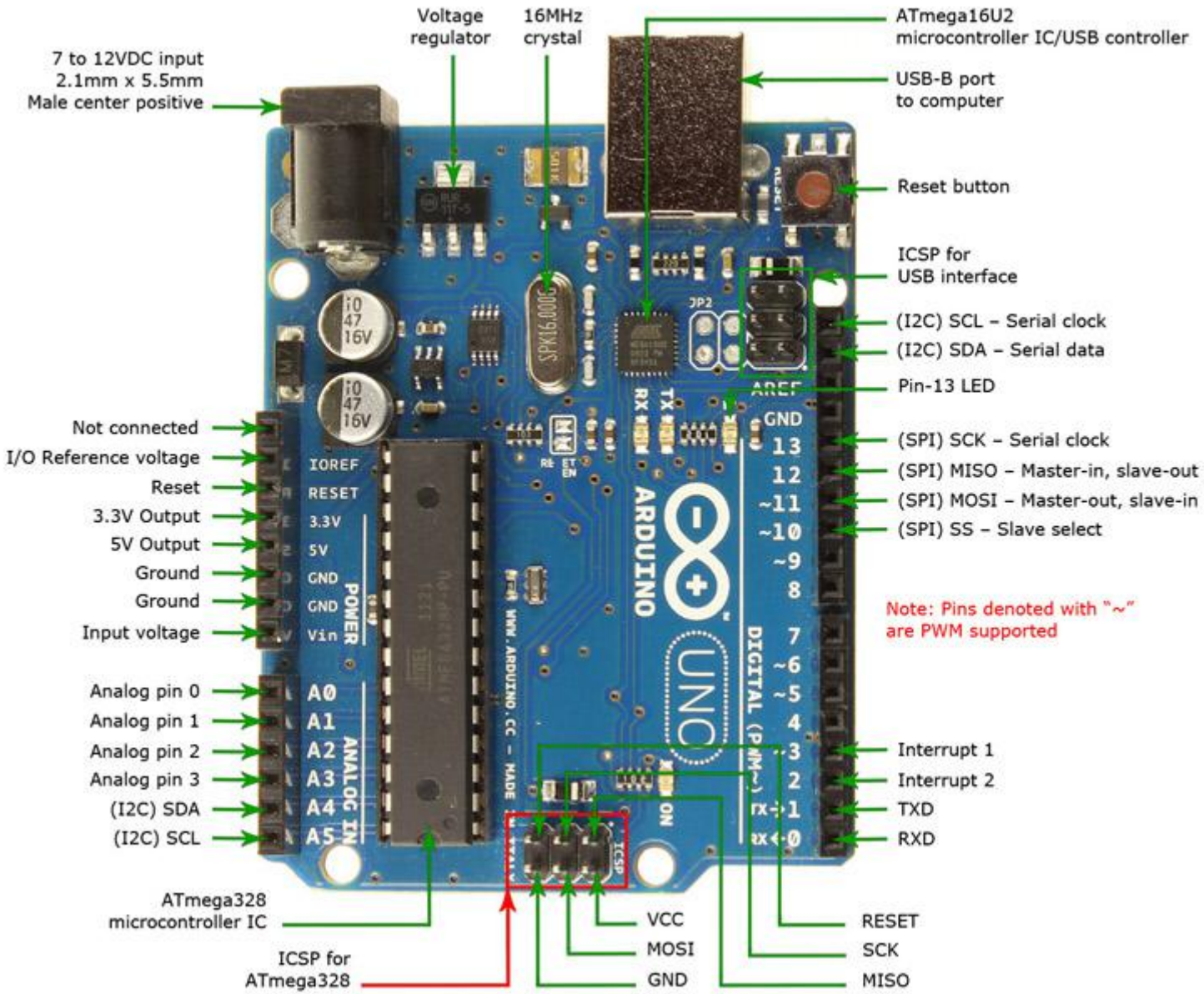


Circuito Reset

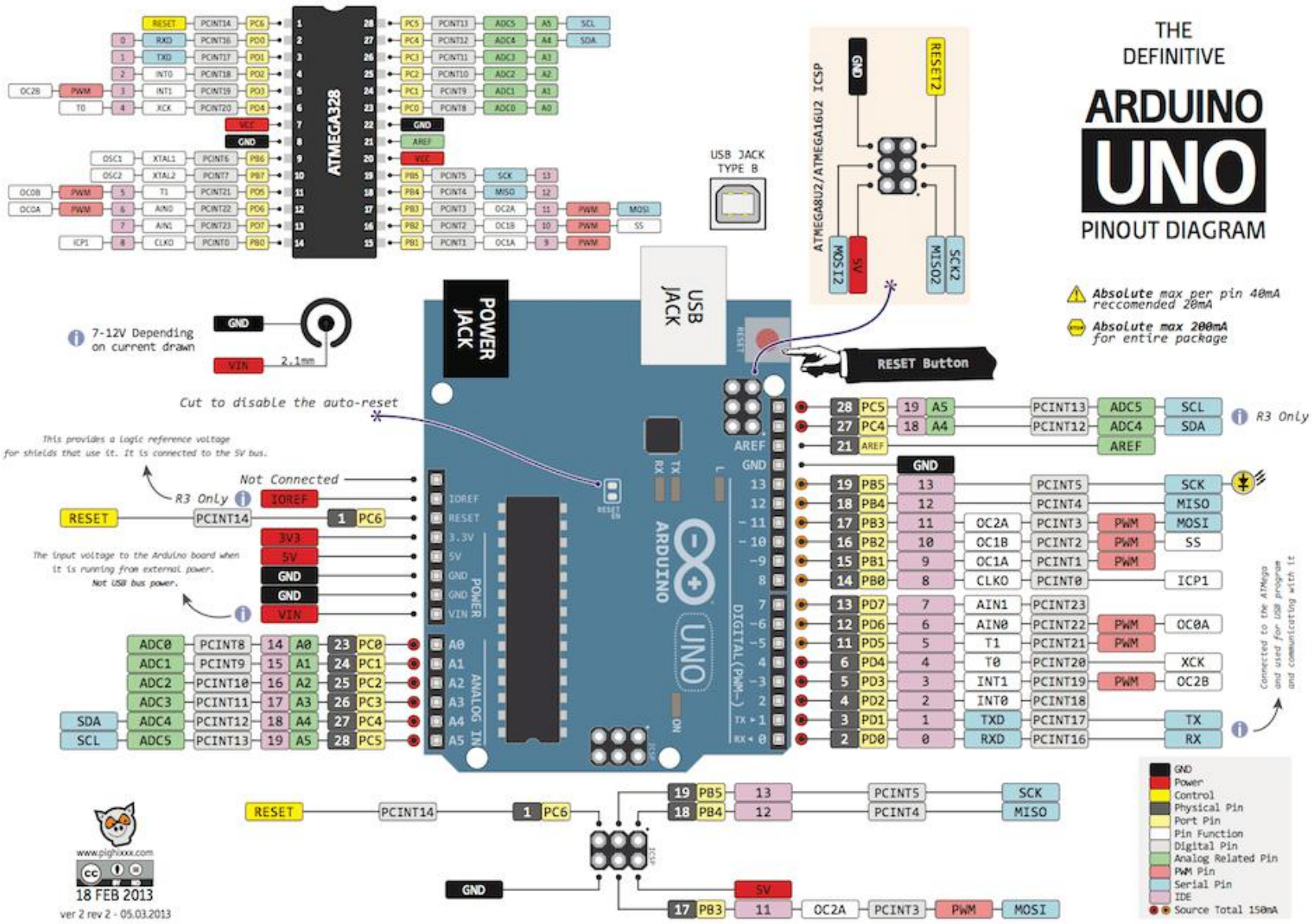
- ✓ O µC é colocado em um estado conhecido.



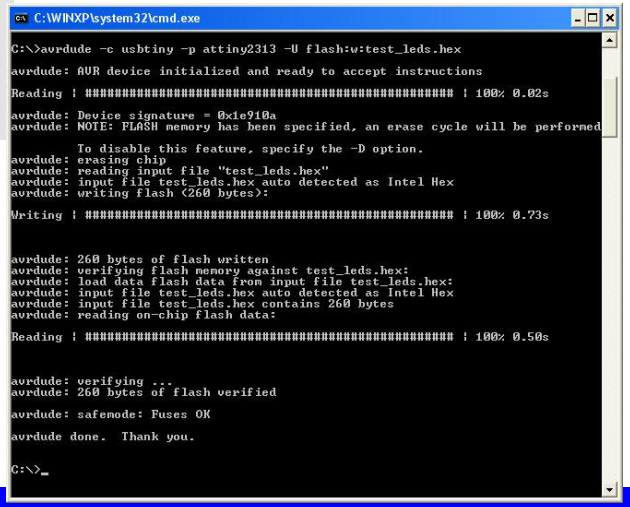
Placa Arduino Uno



Placa Arduino Uno



Atmel Studio IDE, AVRDude e Arduino IDE



- ✓ Passos para instalação e transferência do .hex para o Atmega 328P usando a placa Arduino Uno.

1. Instale o Atmel Studio IDE (www.atmel.com).
2. Instale o Arduino IDE (www.arduino.cc).
3. Conecte uma placa do arduino uno e verifique qual a porta COM.
4. Inicie o atmel studio, depois clique em tools --> external tools e digite:

Title: Programador atmega328

Command: C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe

Arguments: -C "C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -p atmega328p -c arduino -P COM4 -b 115200 -U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i"

5. Em seguida, selecione use output window, clique em apply e depois em ok.

6. Faça o seu código e clique em **F7** ou **Build→ Build Project**.

7. Para carregar o executável, clique em Tools e depois em Programador atmega328.

Programação em Assembly e C no Atmel Studio

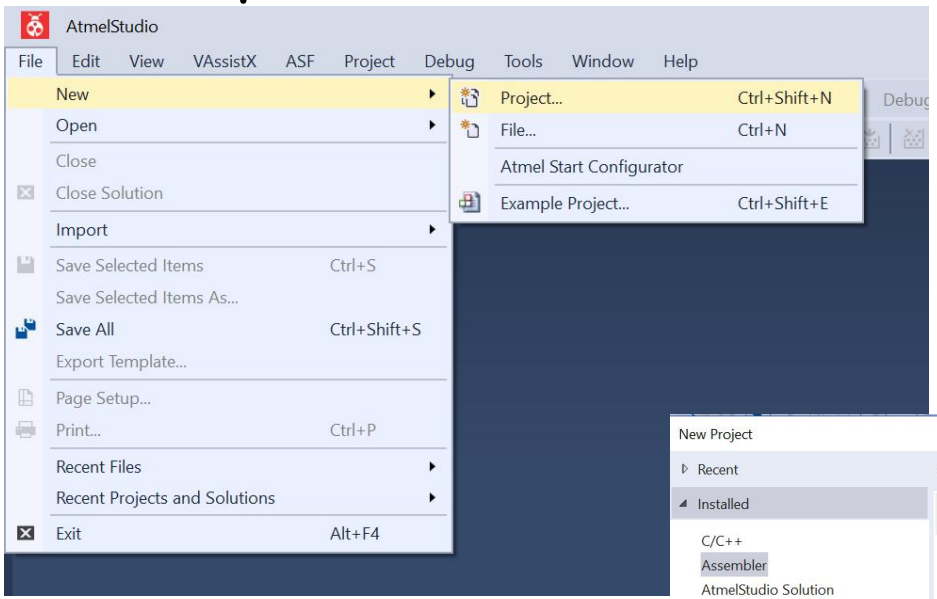
Universidade Federal do Vale do São Francisco - UNIVASF
Colegiado de Engenharia da Computação – CECOMP

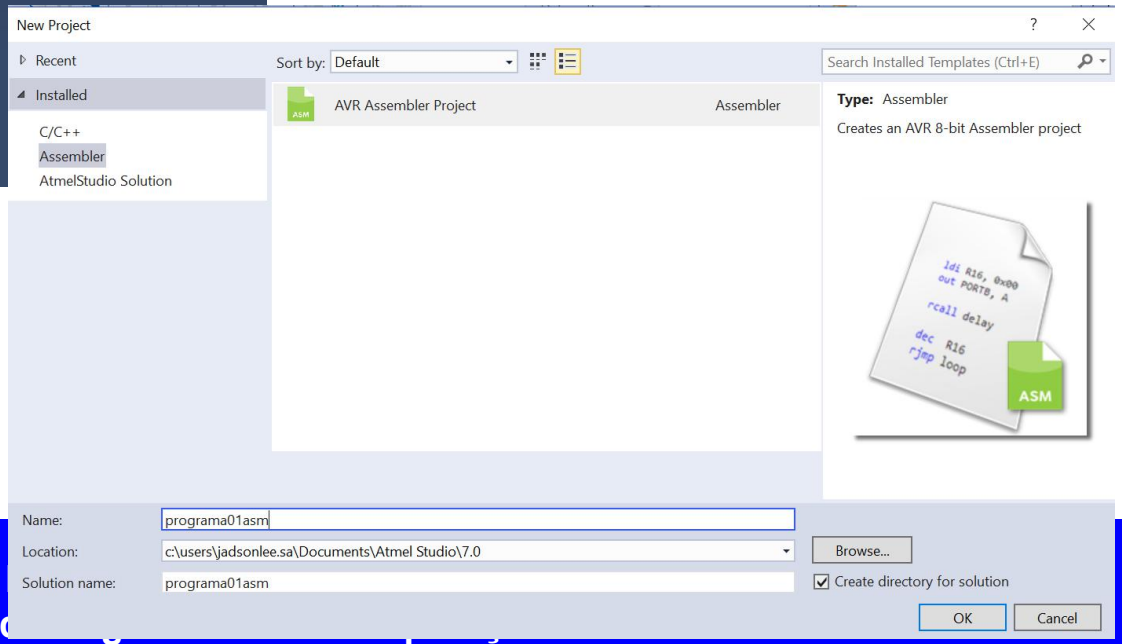
Criando um Projeto em Assembly

- ✓ Abra o Atmel Studio 7.0.

Criando um Projeto em Assembly

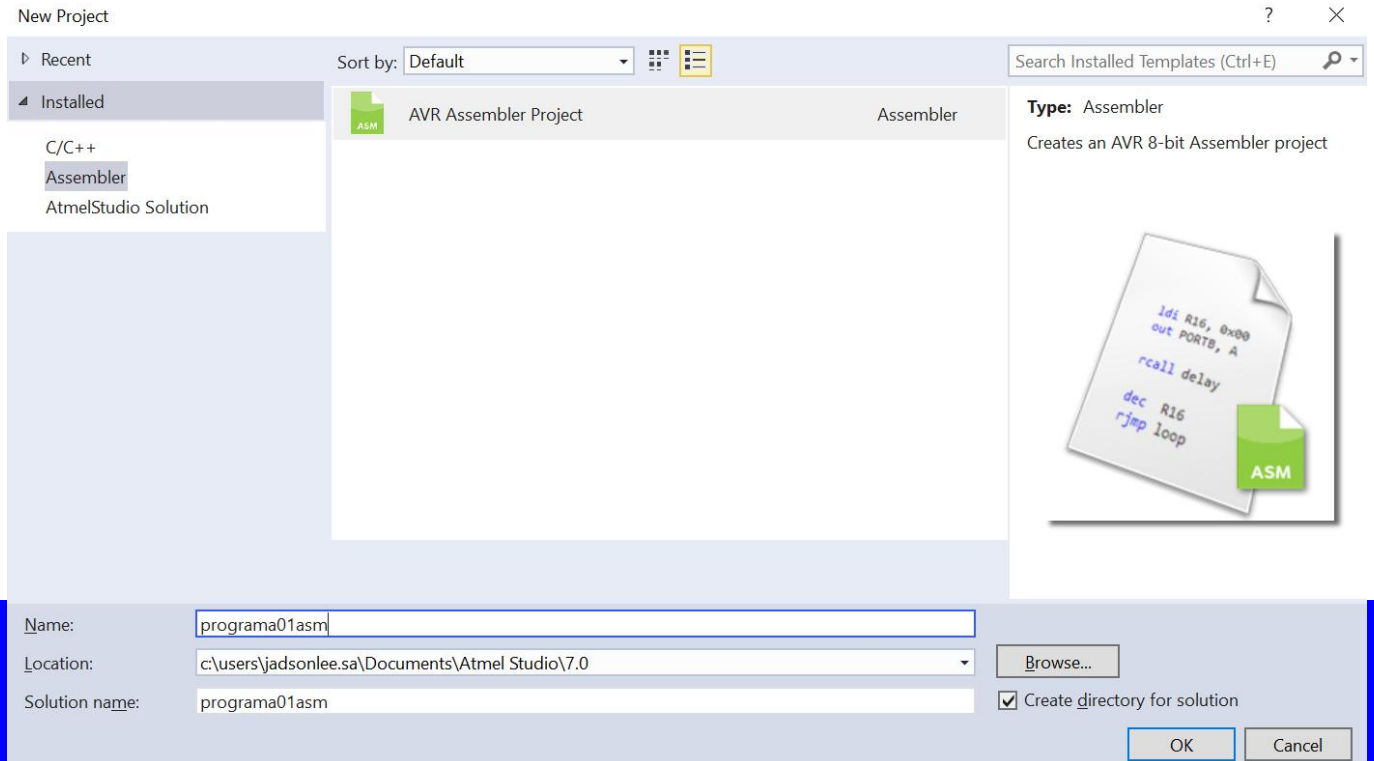
✓ Clique em File → New → Project.





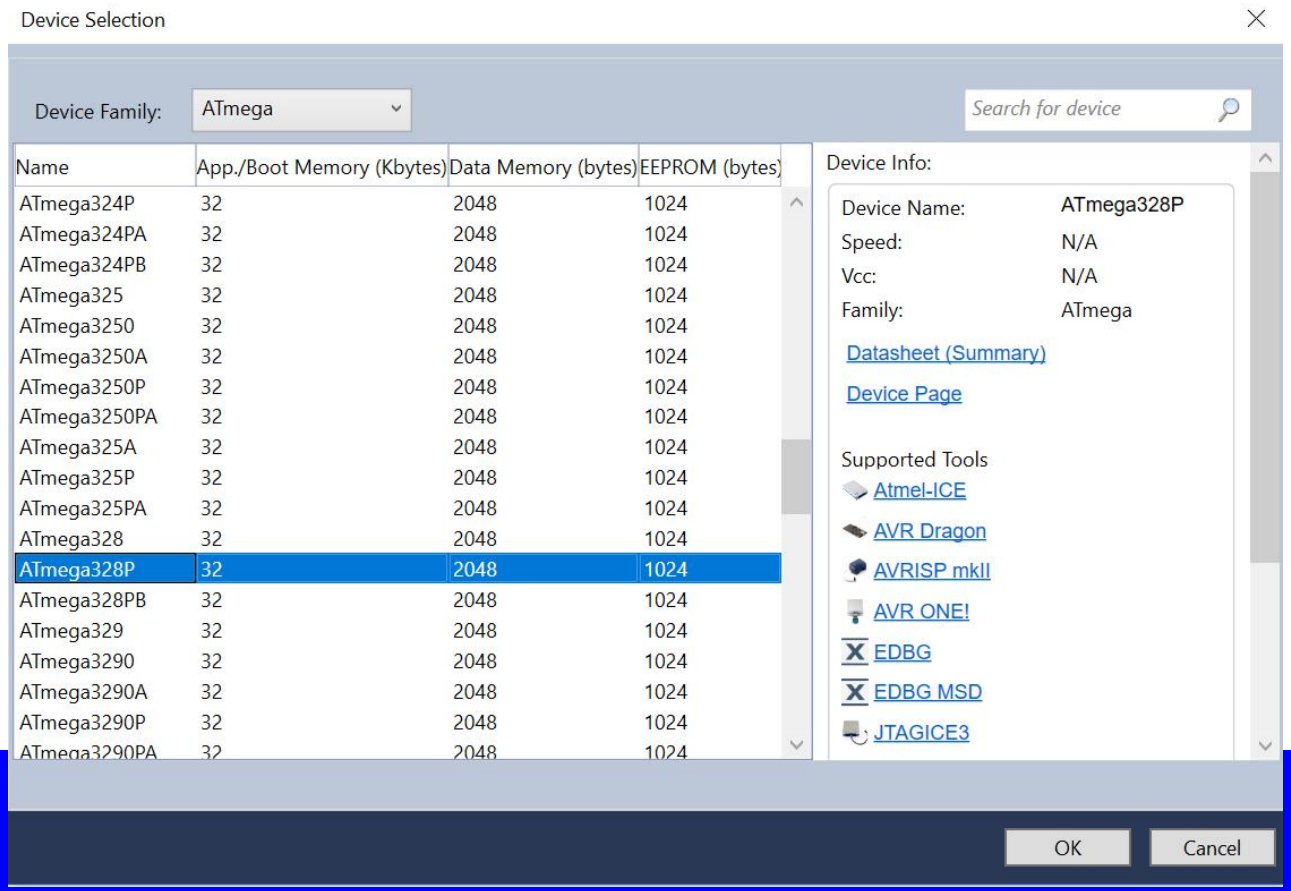
Criando um Projeto em Assembly

- ✓ Selecione Assembler e, em seguida, **AVR Assembler Project**. Dê um nome ao projeto (**Name: Programa01asm**). O mesmo nome aparecerá em **Solution Name**. Observe onde o projeto será salvo (**Location**). Por fim, clique em **OK**.

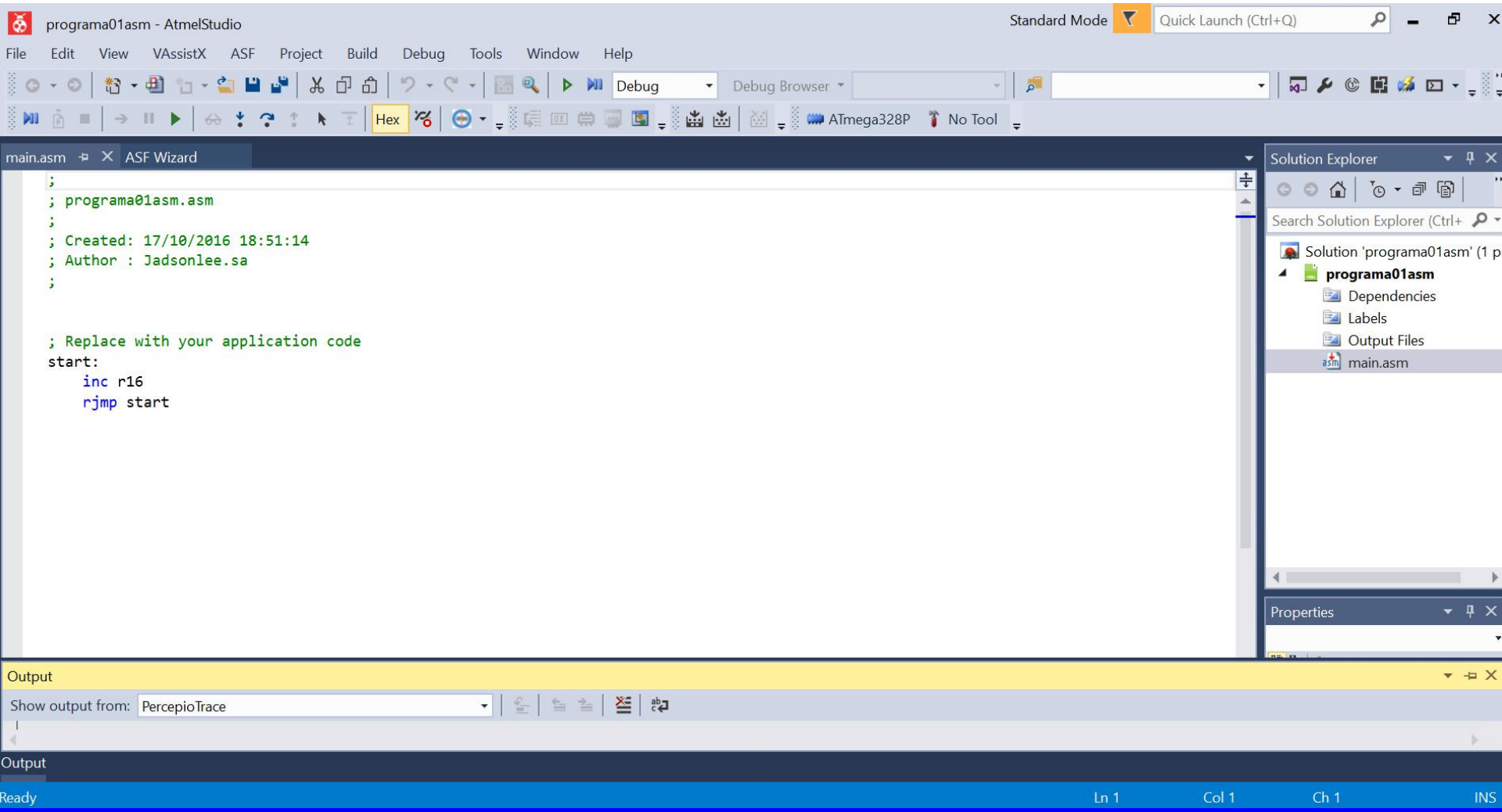


Criando um Projeto em Assembly

✓ Na próxima tela, selecione o **ATmega328P** e clique em OK para o projeto ser criado.

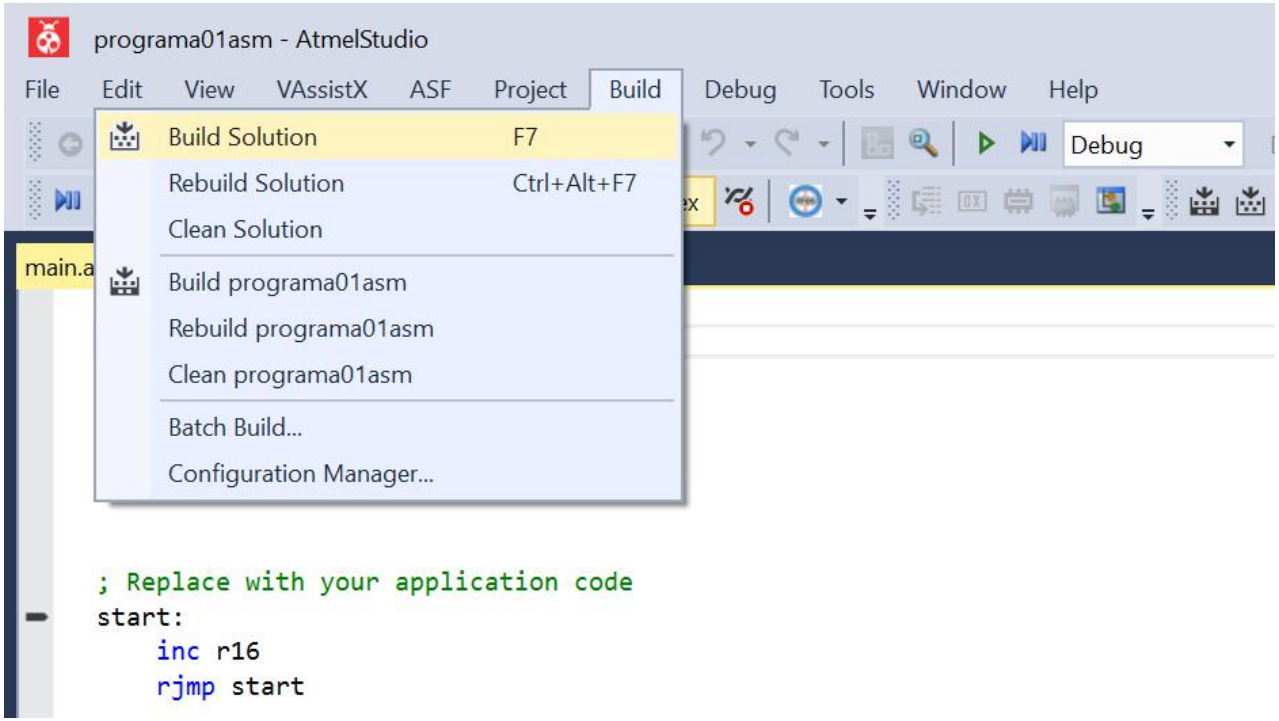


Criando um Projeto em Assembly



Criando um Projeto em Assembly

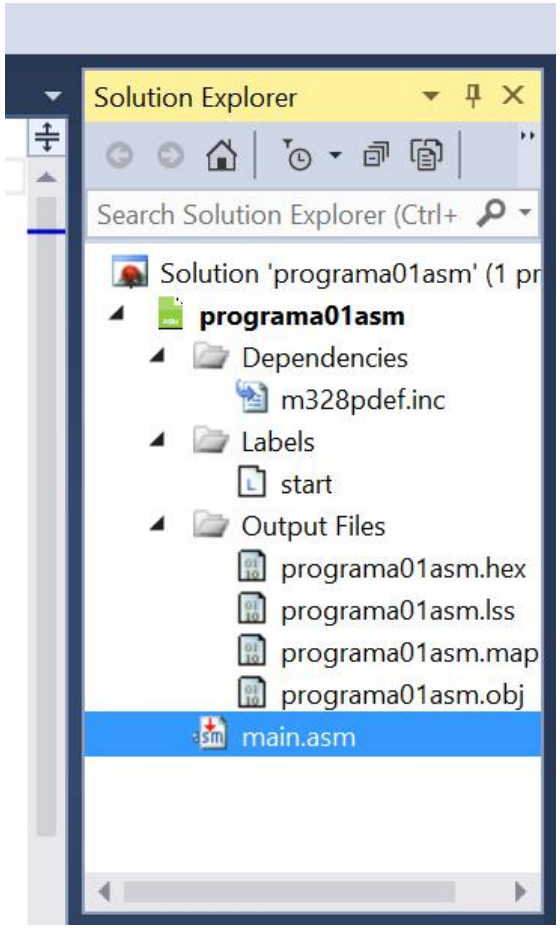
- ✓ Para compilar o projeto clique em **F7** ou **Build → Build Solution**.



Criando um Projeto em Assembly

- ✓ Após o build project.

m328pdef.inc contém a definição dos registradores, bits, locks, FUSES, declarações da memória de dados, vetores de interrupção entre outros.



Criando um Projeto em Assembly

✓ Exemplo básico.

```
;
; Programa01asm.asm
;
; Created: 27/10/2016 16:34:46
; Author : Jadsonlee.sa
;


; Replace with your application code
start:
    inc r16
    rjmp start
```

Status Register (SREG)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

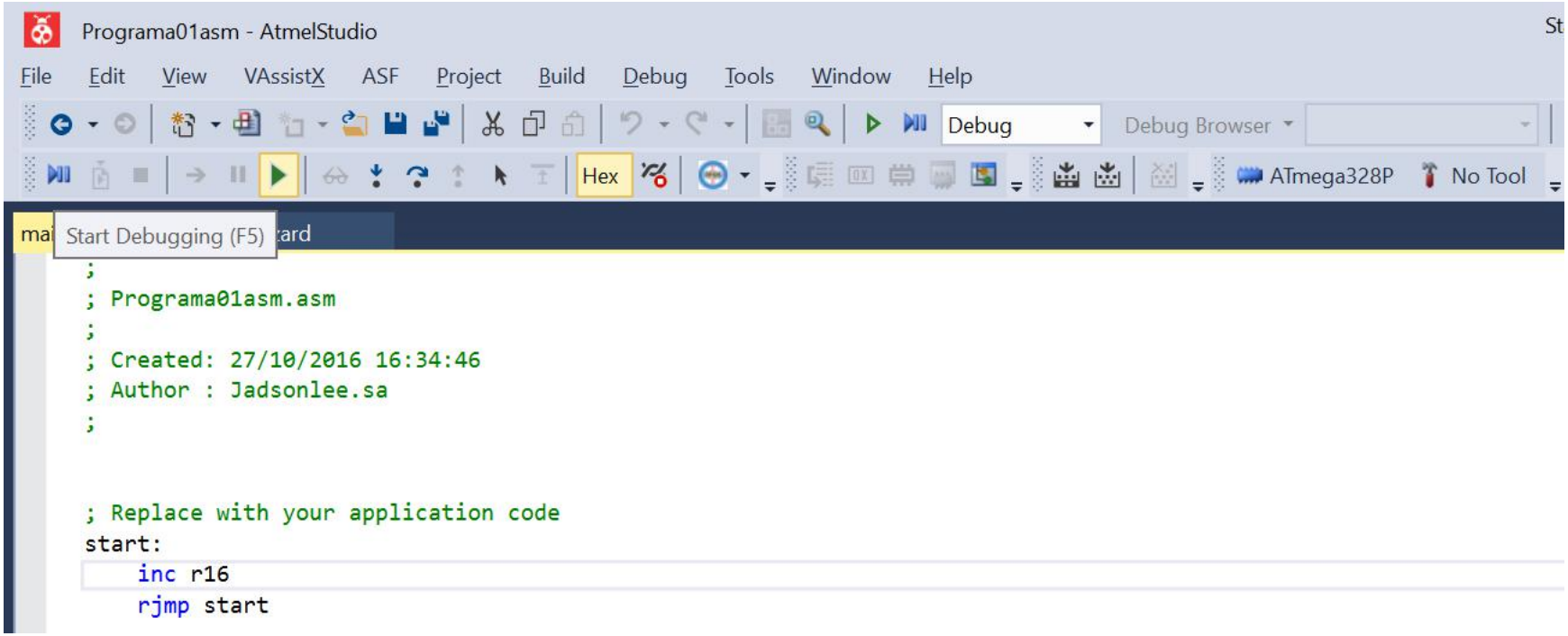
Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1	
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2	

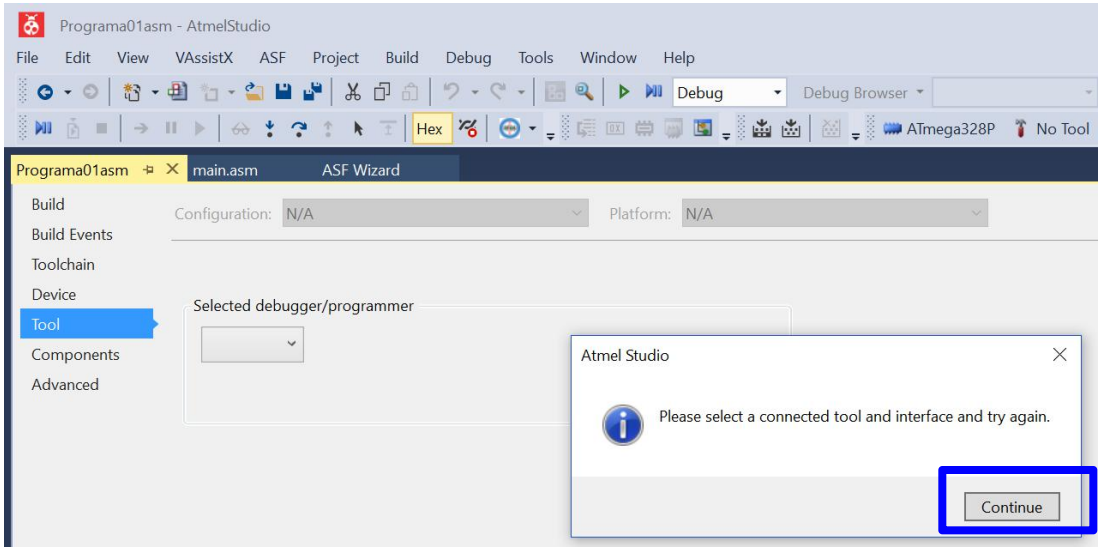
Criando um Projeto em Assembly

- ✓ **Simulando o exemplo.**
  - ✓ Clique em start debugging (F5).

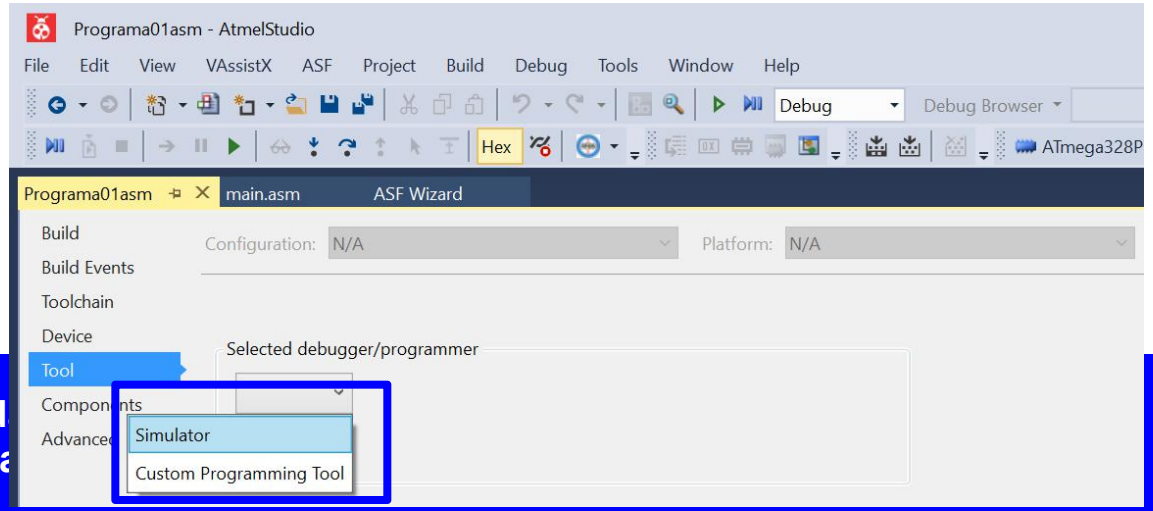


Criando um Projeto em Assembly

- ✓ Simulando o exemplo.

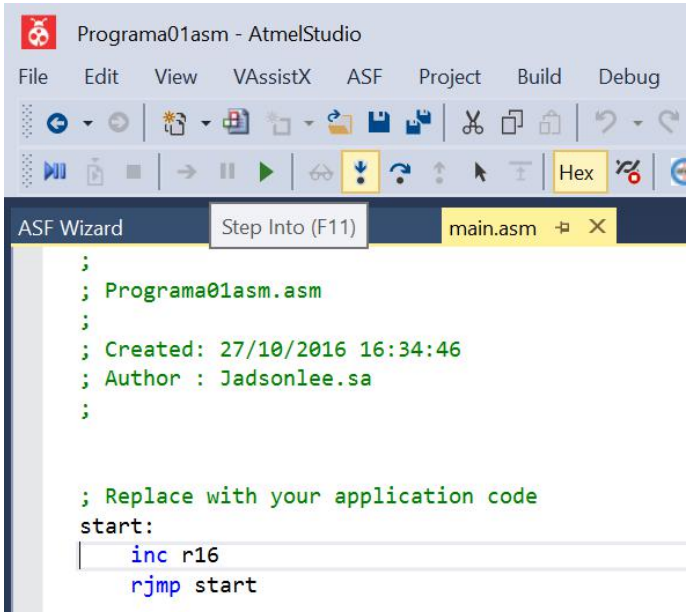


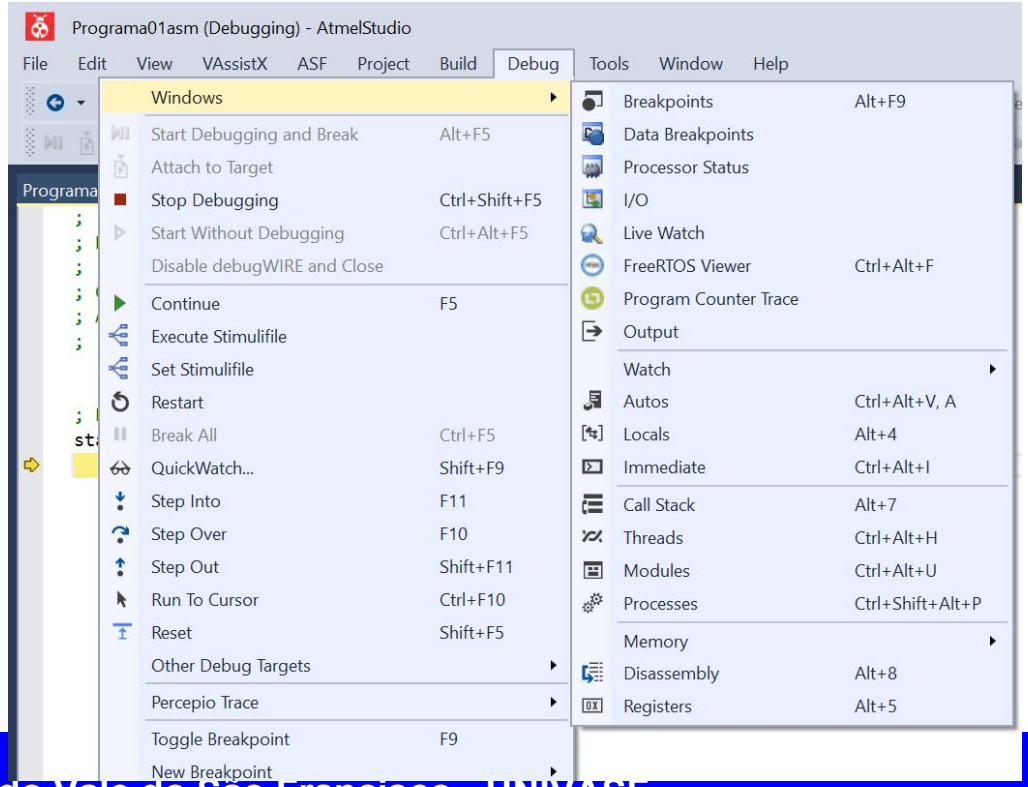
Selecione a aba main.asm e, em seguida, Clique em start debugging.



Criando um Projeto em Assembly

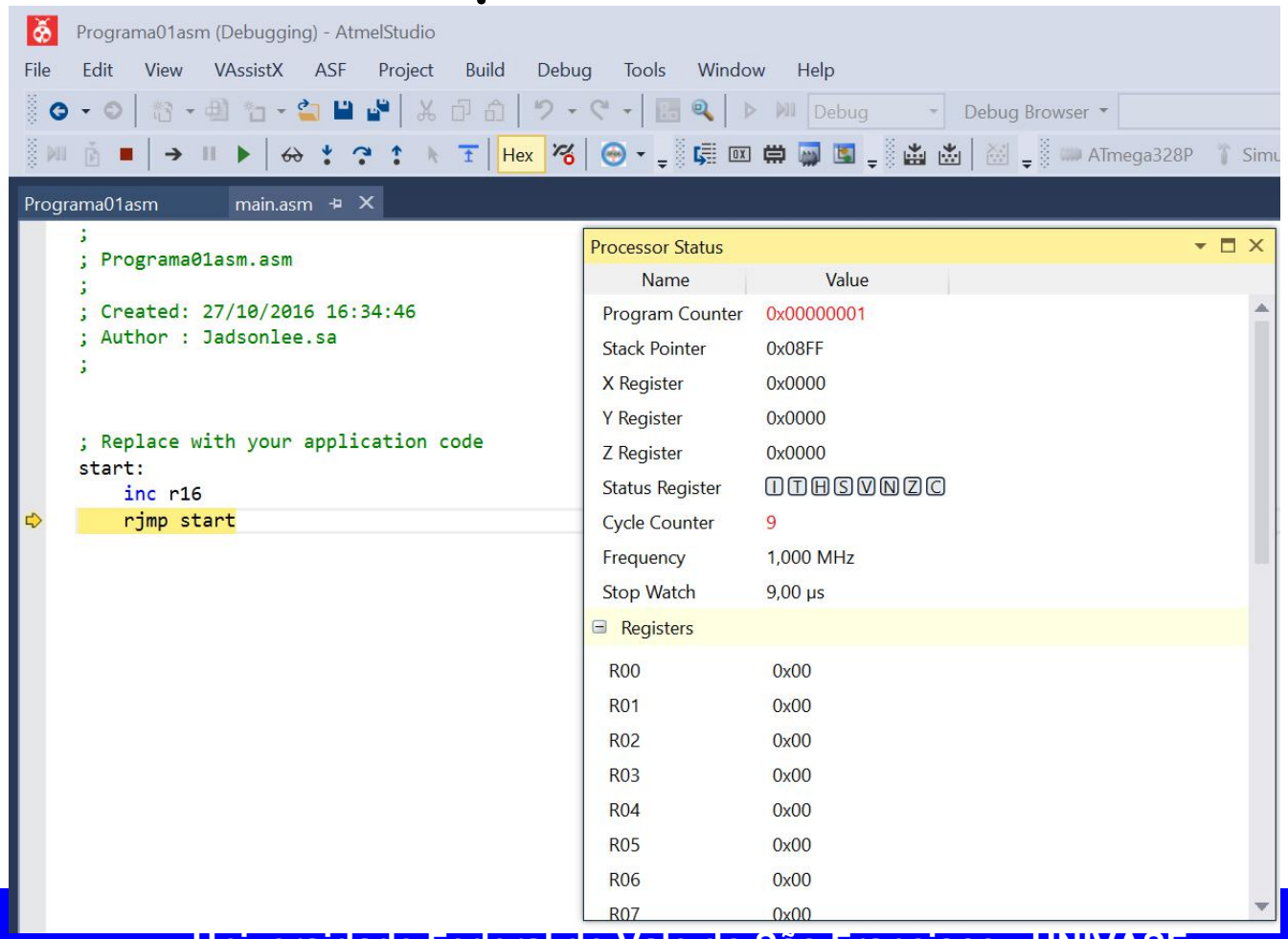
- ✓ **Simulando o exemplo.**
  - ✓ Clique em step into (F11) para executar por instrução.





Criando um Projeto em Assembly

- ✓ Simulando o exemplo.



"Hello World"

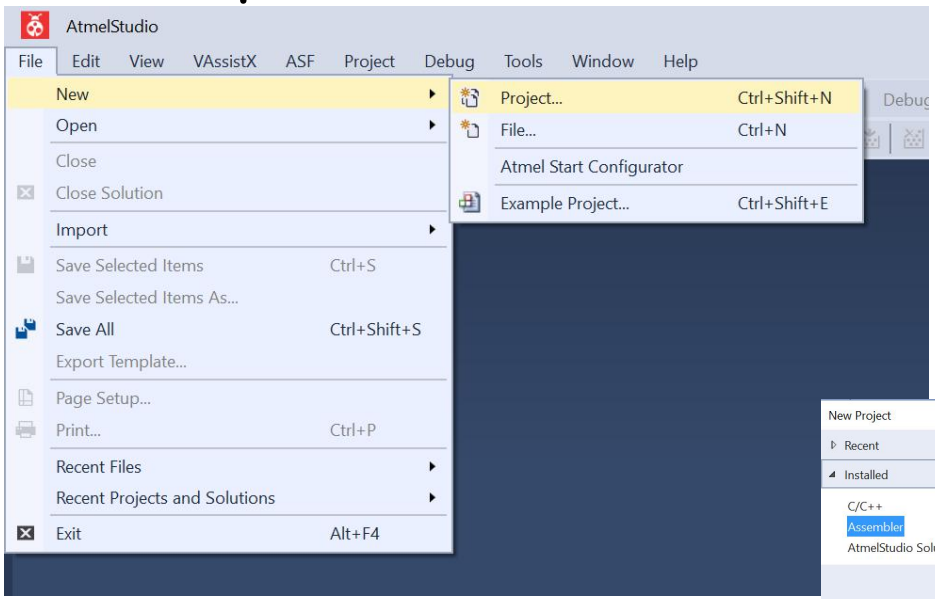
```
start:

    ldi r16, 0b00100000
    out DDRB, r16

again:
    ldi r16, 0b00100000
    out PORTB, r16

    ldi r16, 0b00000000
    out PORTB, r16

    rjmp again
```

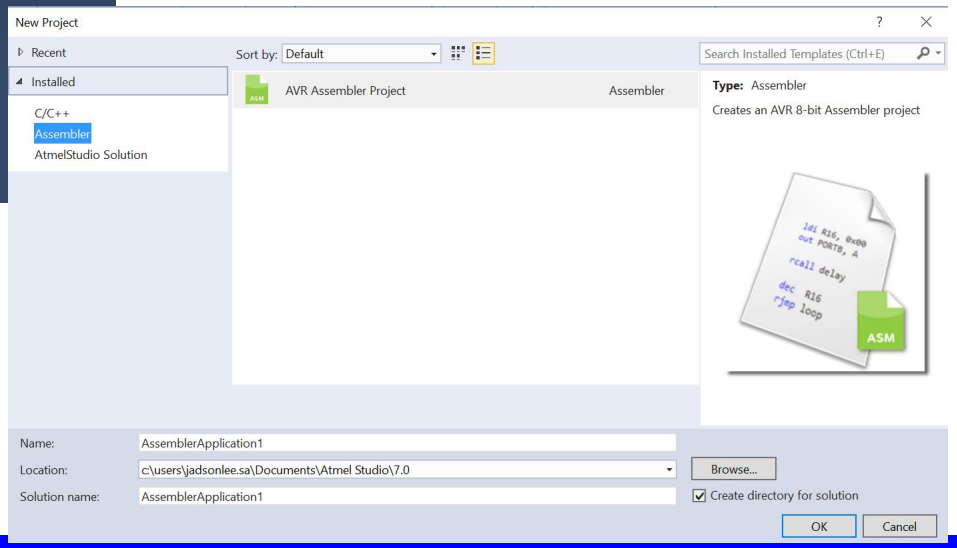
Criando um Projeto em C

- ✓ Abra o Atmel Studio 7.0.

Criando um Projeto em C

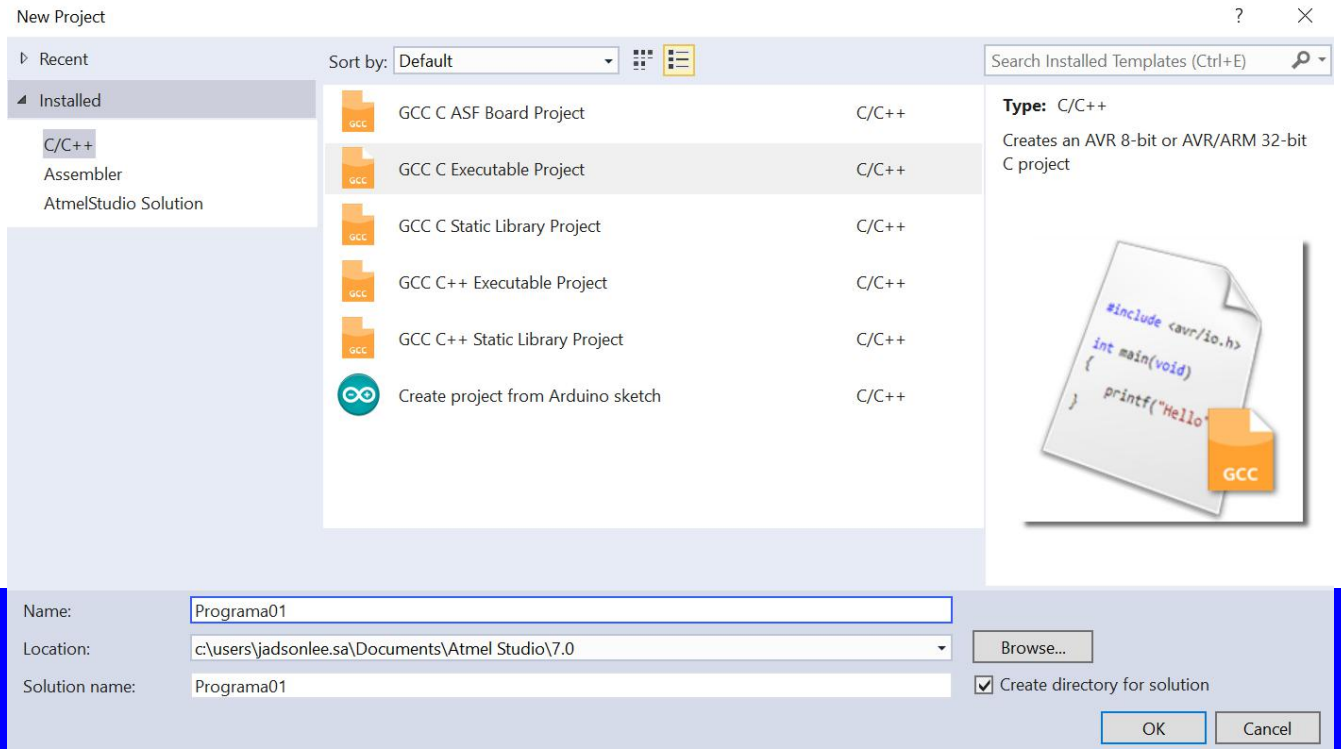
✓ Clique em File → New → Project.





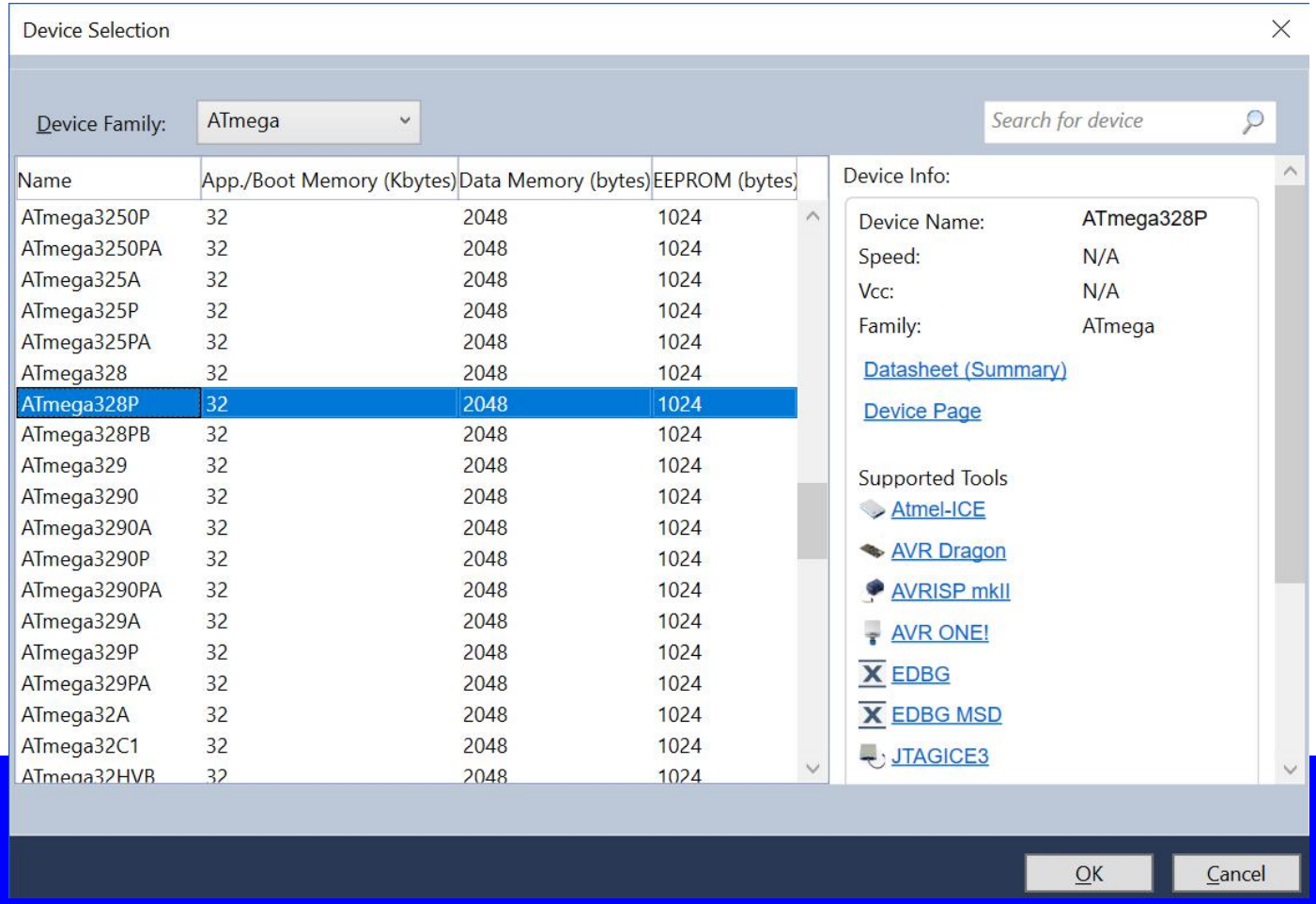
Criando um Projeto em C

- ✓ Selecione **C/C++** e, em seguida, **GCC C Executable Project**. Dê um nome ao projeto (**Name: Programa01**). O mesmo nome aparecerá em **Solution Name**. Observe onde o projeto será salvo (**Location**). Por fim, clique em **OK**.



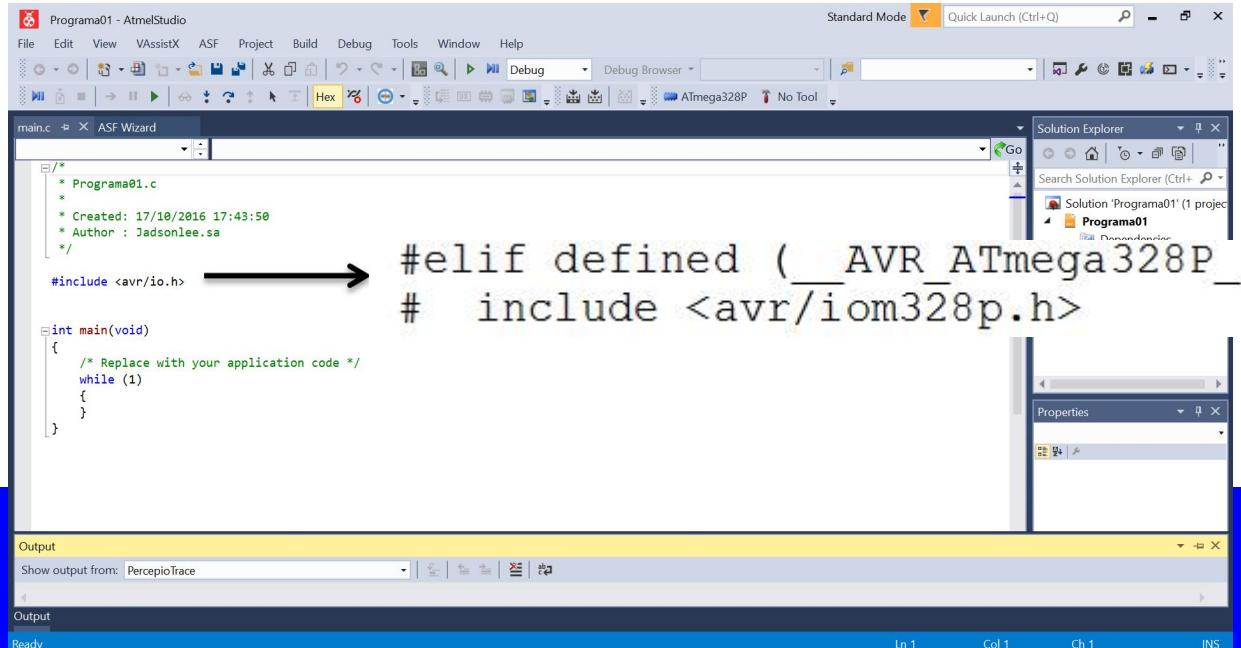
Criando um Projeto em C

- ✓ Na próxima tela, selecione o **ATmega328P** e clique em OK para o projeto ser criado.



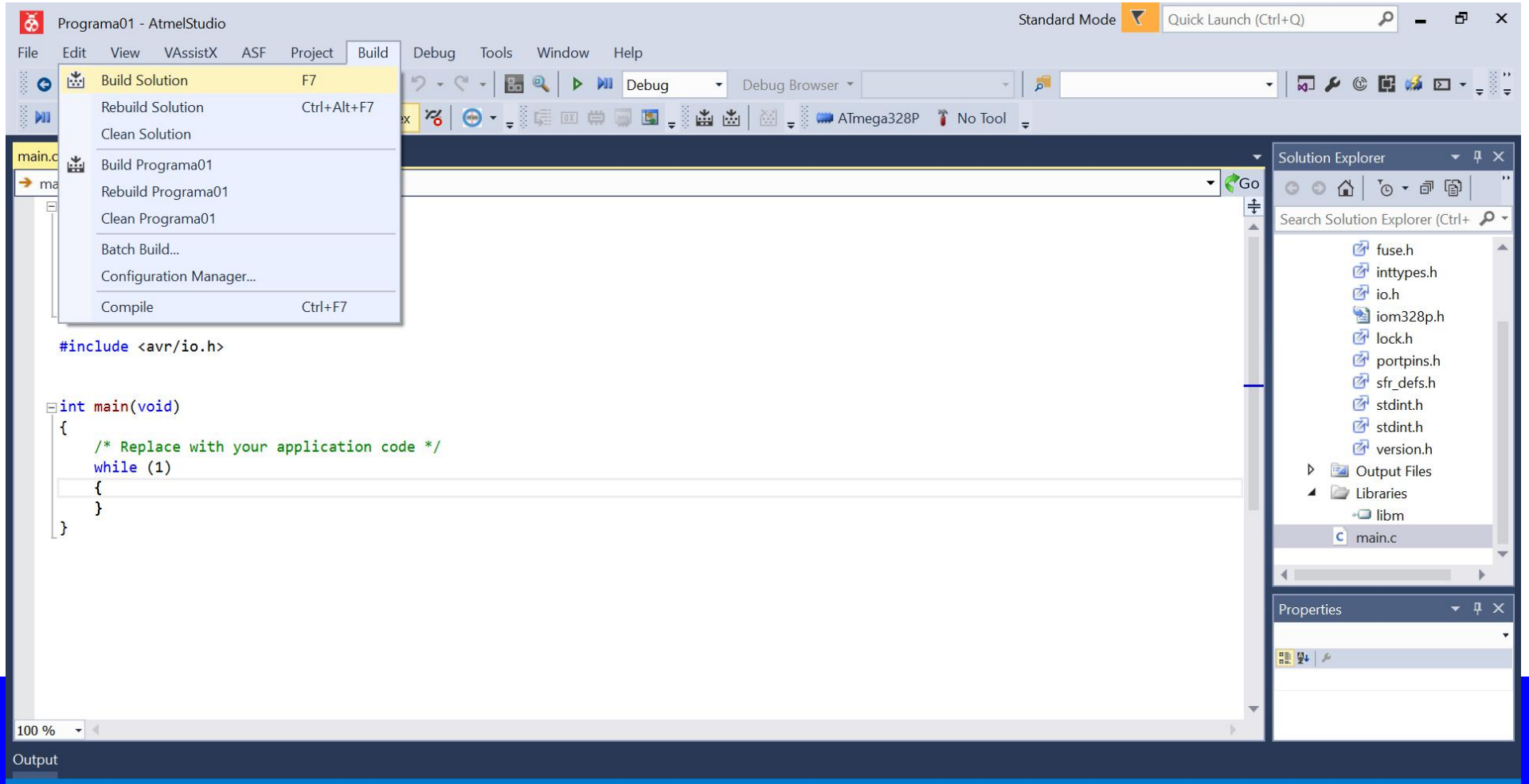
Criando um Projeto em C

- ✓ O arquivo io.h direciona para arquivos que contém as definições dos registradores, vetores de interrupção, constantes, FUSES entre outros do ATmega328P. Esse arquivo está em C:\Program Files x86)\Atmel\Studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\avr.

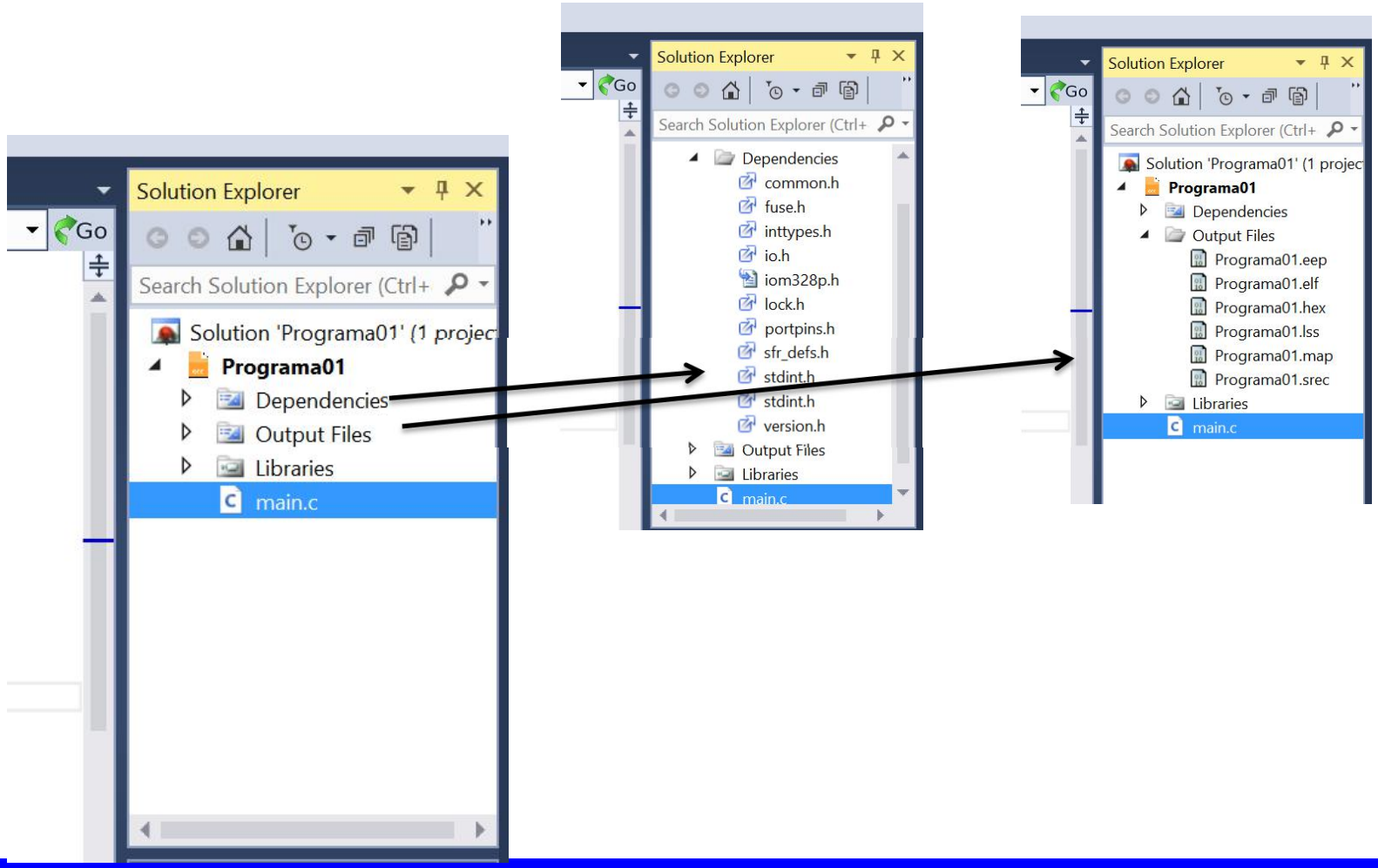


Criando um Projeto em C

- ✓ Para compilar o projeto clique em **F7** ou **Build → Build Solution**.



Criando um Projeto em C



"Hello World"

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB = 0b00100000;//PB5 COMO SAÍDA E OS DEMAIS COMO ENTRADA
    PORTB = 0x20;//PB5 = 1

    _delay_ms(1000);

    while (1)
    {
        PORTB = 0x00;//PB5 = 0
        _delay_ms(1000);
        PORTB = 0x20;//PB5 = 1
        _delay_ms(1000);
    }
}
```

"Hello World" - API Arduino

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```