O FORMATO HEX

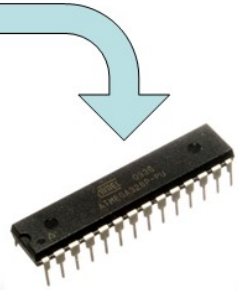
© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com
Revisão em setembro 2026

Montagem

Compilador
MikroC for AVR

Programa
Fonte
(Texto em C)

Programa
executável
(Hex)



```
/*-------------------------------------------------------------
 Programa : Um.c
 Data     : 25/05/2020
 Autor    : Prof. Vargas
 Descrição:
    Como exercício inicial vamos criar um programa que faz os
    LEDs D1 a D6 piscarem alternadamente com 0.5S de tempo.
    Sendo que os mesmos estão conetados respectivamente em
    PB0 a PB5.
 -------------------------------------------------------------*/
void main()
{
   PORTB = 0b11010101; //dado que impede qualquer LED acender
   DDRB  = 0b00111111; //define quem é entrada e quem é saida
   for(;;)
   {
      PORTB.B0 = !PORTB.B0; //muda estado do LED D1
      PORTB.B1 = !PORTB.B1; //muda estado do LED D2
      PORTB.B2 = !PORTB.B2; //muda estado do LED D3
      Delay_ms(500);  //temporiza 0.5S
   }//final do loop permanente
}
```

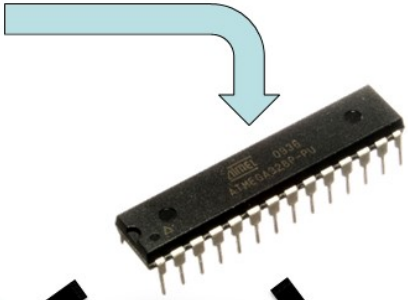
```
:1000000035C00000FDCF0000FBCF0000F9CF00009D
:10001000F7CF0000F5CF0000F3CF0000F1CF0000D4
:10002000EFCF0000EDCF0000EBCF0000E9CF0000E4
:10003000E7CF0000E5CF0000E3CF0000E1CF0000F4
:10004000DFCF0000DDCF0000DBCF0000D9CF000004
:10005000D7CF0000D5CF0000D3CF0000D1CF000014
:06006000CFCF0000CDCF60
:10006C00BFEFBDBFB8E0BEBFB5EDB5B9BFE3B4B926
:10007C0005B0B1E00B2605B805B0B2E00B2605B80B
:10008C0005B0B4E00B2605B805B0B8E00B2605B8F2
:10009C0005B0B0E10B2605B805B0B0E20B2605B8EB
:1000AC0029E216E900E80A95F1F71A95E1F72A9585
:0600BC00D1F7DECFFFCFFB
:00000001FF
```

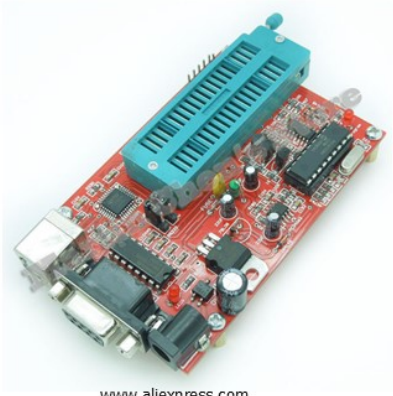
© Prof. Vargas

Processo

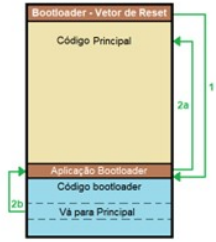
```
:10003000E7CF0000E5CF0000E3CF0000E1CF0000F4
:10004000DFCF0000DDCF0000DBCF0000D9CF000004
:06006000CFCF0000CDCF60
:0600BC00D1F7DECFFFCFFB
:00000001FF
```

Formato Intel





www.aliexpress.com



© Prof. Vargas

A partir do HEX

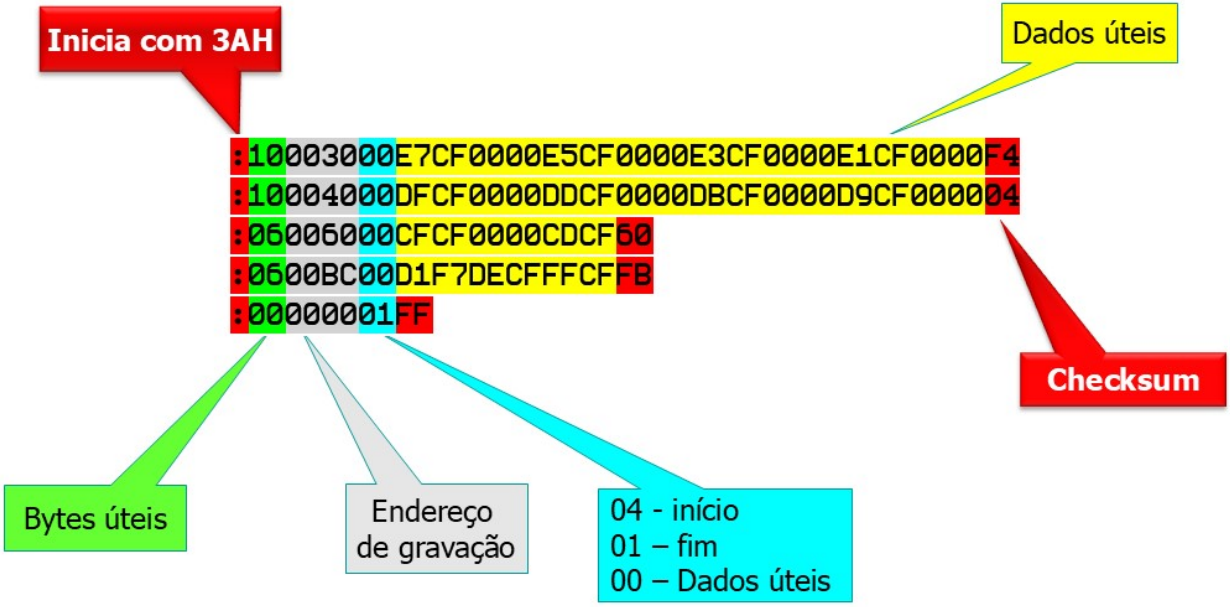
- O formato HEX proposto pela INTEL® e usado até os dias de hoje, é conforme o exemplo seguinte:

```
:10003000E7CF0000E5CF0000E3CF0000E1CF0000F4
:10004000DFCF0000DDCF0000DBCF0000D9CF000004
:06006000CFCF0000CDCF60
:0600BC00D1F7DECFFFCFFB
:00000001FF
```

© Prof. Vargas

HEX

- O qual, podemos enxergar melhor colorindo:



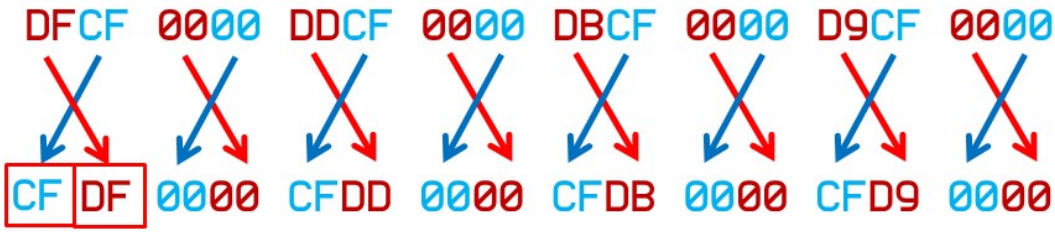
© Prof. Vargas

Extraindo os dados

```
:10003000E7CF0000E5CF0000E3CF0000E1CF0000F4
:10004000DFCF0000DDCF0000DBCF0000D9CF000004
:06006000CFCF0000CDCF60
:0600BC00D1F7DECFFFCFFB
:00000001FF
```

E podemos extrair partes do programa:

DFCF0000DDCF0000DBCF0000D9CF0000



© Prof. Vargas

Identificando

CFDF = 1100 1111 1101 1111

16 bits

Formato de comando

1100.1111.1101.1111

Procurando no set de instruções encontramos:

16-bit Opcode:

1100	kkkk	kkkk	kkkk

Por comparação, concluímos que é um jump relativo:

Syntax:	**Operands:**
RJMP k	$-2K \leq k < 2K$

© Prof. Vargas

Bit Assignments

- rrrrr = Source register
- rrrr = Source register (R16-R31)
- rrr = Source register (R16-R23)
- RRRR = Source register pair (R0:R1 ... R30:R31)
- ddddd = Destination register
- dddd = Destination register (R16-R31)
- ddd = Destination register (R16-R23)
- DDDD = Destination register pair (R0:R1 ... R30:R31)
- pp = Register pair, W, X, Y or Z
- y = Y/Z register pair bit (0=Z, 1=Y)
- u = FMUL(S(U)) signed with 0=signed or 1=unsigned
- s = Store/load bit (0=load, 1=store)
- c = Call/jump (0=jump, 1=call)
- cy = With carry (0=without carry 1=with carry)
- e = Extend indirect jump/call address with EIND (0=0:Z, 1=EIND:Z)
- q = Extend program memory address with RAMPZ (0=0:Z, 1=RAMPZ:Z)
- aaaaaa = I/O space address
- aaaaa = I/O space address (first 32 only)
- bbb = Bit number
- B = Bit value
- kkkkkk = 6-bit unsigned constant
- KKKKKKKK = 8-bit constant

1	1	0	c	12 bit signed offset	Relative jump/call to PC ± 2×simm12

© Prof. Vargas

RJMP – Relative Jump

From Datasheet

94.1. Description

Relative jump to an address within PC - 2K +1 and PC + 2K (words). For AVR microcontrollers with Program memory not exceeding 4K words (8KB) this instruction can address the entire memory from every address location. See also JMP.

Operation:

(i) PC ← PC + k + 1

	Syntax:	Operands:	Program Counter:	Stack:
(i)	RJMP	k -2K ≤ k < 2K	PC ← PC + k + 1	Unchanged

16-bit Opcode:

1100	kkkk	kkkk	kkkk

Sinal +

1100.1111.1101.1111

RJMP 2015

© Prof. Vargas

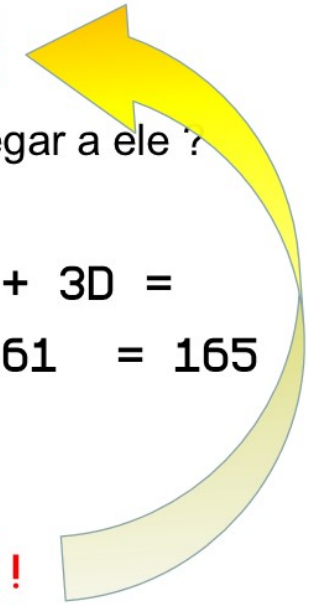
Interpretando o CHKSUM

- Considere uma linha qualquer:

```
:02400E00183D5B
```

- Onde o CHKSUM é **5B.** Como chegar a ele ?
- Somamos todos os Bytes:

```
02 + 40 + 0E + 00 + 18 + 3D =
2 + 64 + 14 + 0 + 24 + 61   = 165
165    = 1010.0101
Em C1 = 0101.1010
Em C2 = 0101.1011 = 5B !
```



© Prof. Vargas

© Prof. Vargas