Assembly AVR

Trabalhando com os registradores

Prof. Roberto de Matos

roberto.matos@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Objetivos

Objetivo

- Introduzir arquitetura "*Load-Store*"
- Entender e praticar com instruções que manipulam exclusivamente os registradores
- Utilizar as diretivas .DEF e .EQU

Referências:

- Datasheet do ATmega328p
- Manual online do montador
- Manual do conjunto de instruções do AVR
- AVR e Arduino: Técnicas de Projeto, 2ª ed.[1]

[1] Alguns exemplos e imagens dessa apresentação são retirados desse livro.

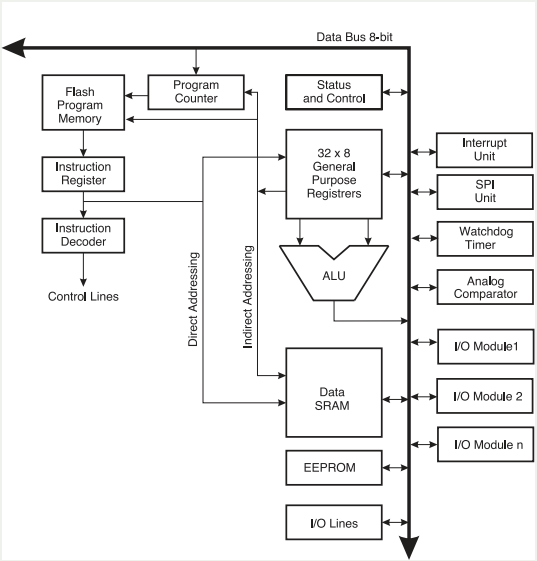
Motivação

Instruções Aritméticas do ATmega328p

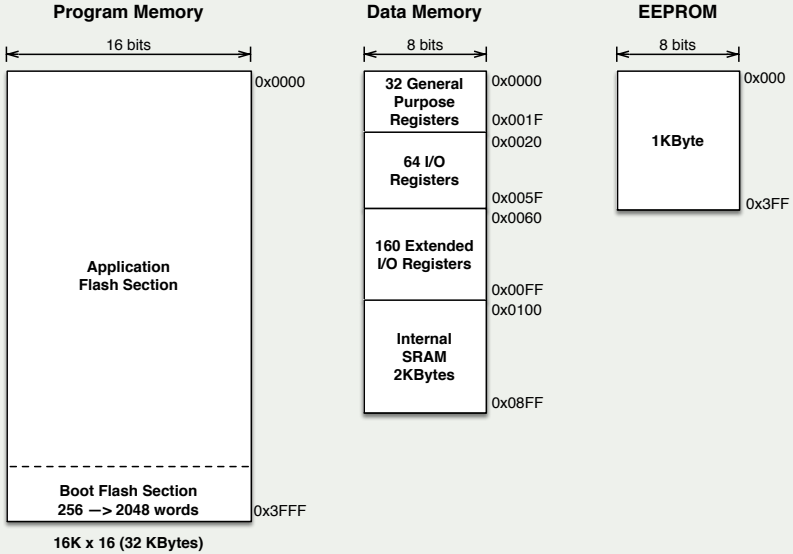
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2

Relembrando ...

Core do AVR

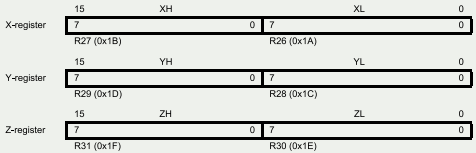

Data Bus 8-bit
Flash Program Memory
Program Counter
Status and Control
Instruction Register
Instruction Decoder
Control Lines
32 x 8 General Purpose Registrers
Direct Addressing
Indirect Addressing
ALU
Data SRAM
EEPROM
I/O Lines
Interrupt Unit
SPI Unit
Watchdog Timer
Analog Comparator
I/O Module1
I/O Module 2
I/O Module n

Memórias ATmega328p



Registradores de Uso Geral

7 … 0	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
…		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
…		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte



Endereçamento Imediato

Imediato: Registrador ← Constante

- Uma constante é carregada (*load*) para um registrador.
- A constante está gravada na memória de programa (*Flash*), codificada como o operando da instrução.
- Instrução: ***ldi Rd, K***
- Opcode:

1110	KKKK	dddd	KKKK

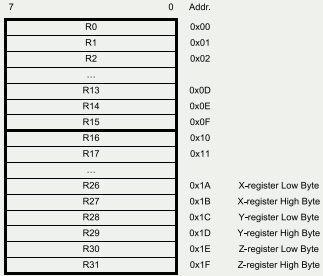
- $Rd \leftarrow K, onde : \mathbf{16} \leq \mathbf{d} \leq \mathbf{31}, 0 \leq K \leq 255$

Imediato: Registrador ← Constante

- Uma constante é carregada (*load*) para um registrador.
- A constante está gravada na memória de programa (*Flash*), codificada como o operando da instrução.
- Instrução: ***ldi Rd, K***
- Opcode:

1110	KKKK	dddd	KKKK

- $Rd \leftarrow K, onde: \mathbf{16} \leq \mathbf{d} \leq \mathbf{31}, 0 \leq K \leq 255$



- Exemplo:

```
1 LDI R16, 0x14   //opcode = 1110 0001 0000 0100
```

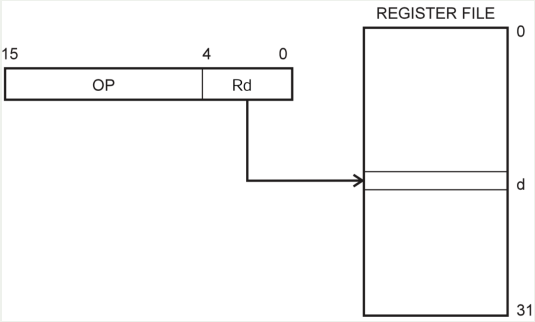

Instruções Modo Imediato

- **Transferência:** *ldi*
- **Lógica e Aritmética:** *andi, ori, addi, adiw, subi, sbci, sbiw*
- **Desvio:** *cpi*

Endereçamento Registrador Direto

Registrador Único

- O operando da instrução é o conteúdo de *Rd*. Após a operação *Rd* recebe o resultado.



- *<instrução>* Rd
  $0 \leq d \leq 31$

- Exemplo:

```
1 INC r0
2 CLR r0
```

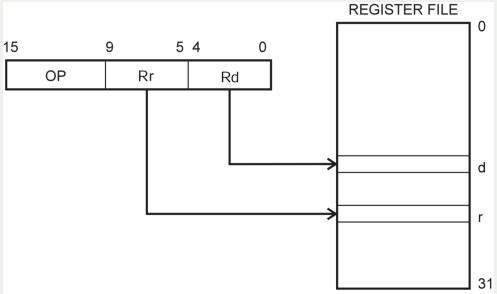
Instruções Registrador Único

- **Transferência:** *push, pop*.
- **Lógica e Aritmética:** *com, neg, inc, dec, tst, clr, ser*.
- **Bit e Teste de Bit:** *lsl, lsr, asr, rol, ror, swap, bld, bst*.

Dois Registradores

- Os operandos estão contidos nos registradores *Rr* e *Rd*. O resultado é armazenado no registro *Rd*.

- **mov Rd, Rr**
  $Rd \leftarrow Rr\ 0 \leq r/d \leq 31$



- **movw Rd, Rr**
  $Rd + 1 : Rd \leftarrow Rr + 1 : Rr$
  $r/d \in \{0, 2, \cdots, 30\}$

- Exemplo:

```
1 INC r0
2 MOV r1, r0
3 INC r0
4 MOVW r2,r0
```

Instruções Dois Registradores

- **Transferência:** *mov, movw*.
- **Lógica e Aritmética:** *add, adc, sub, sbc, and, or, eor, mul, muls, mulsu, fmul, fmuls, fmulsu*.

Diretivas do montador

Diretivas **.DEF** e **.EQU**

- **.DEF** define um nome simbólico para um **registrador**.
- Sintaxe:
  .DEF nome = registrador

Diretivas **.DEF** e **.EQU**

- **.DEF** define um nome simbólico para um **registrador**.
- Sintaxe:
  .DEF nome = registrador

- **.EQU** seta um símbolo a partir de uma **expressão**.
- Sintaxe:
  .EQU label = expressão

- Exemplo:

```
1 .DEF temp=r16
2 .EQU valor=0x23+5
3 start:
4     LDI temp, valor ; inicialize o reg. temp com valor
5     INC temp        ; incrementa o reg. temp
6     RJMP start
```


Experimentos

Experimentos no MPLAB IDE

- Simule todos os trechos de códigos apresentados.
- Faça um programa que some dois números de 8 bits armazenados em registradores (R16 e R17), mais uma constante (22) e copie para um terceiro registrador (R18). Modifique os valores dos registradores no simulador para verificar o funcionamento.

FIM!