“Para quem só sabe usar martelo, todo problema é um prego”
(Abraham Maslow).

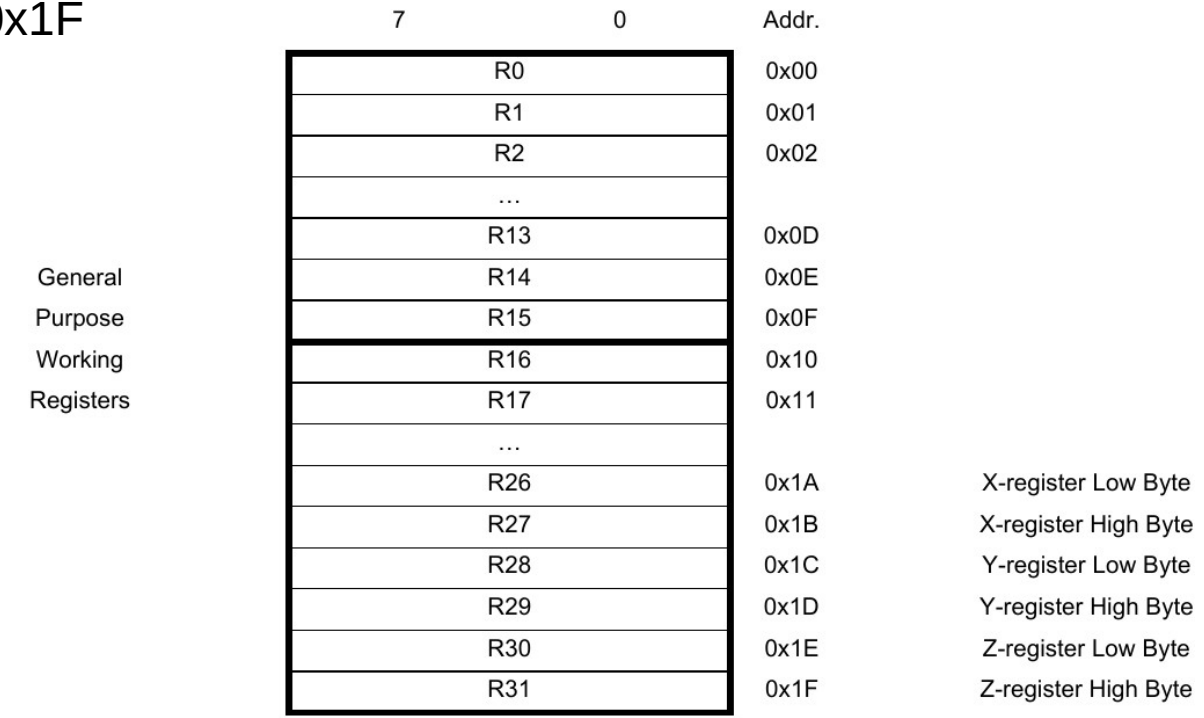
Registradores, Memória e Portas do ATmega328P

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Registradores

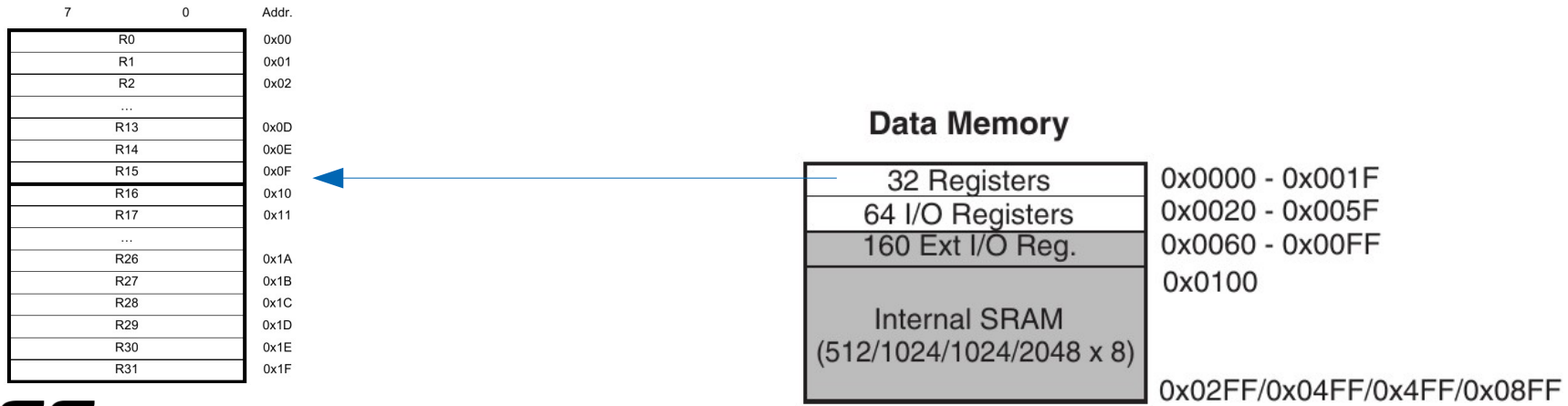
- O ATMega328P possui 32 registradores de uso geral
  - Registradores de **8 bits**
  - Mapeados em memória SRAM
    - Endereços entre 0x00 e 0x1F
- Registradores R0 a R31
- **Seção 7.4 do manual**



UDESC

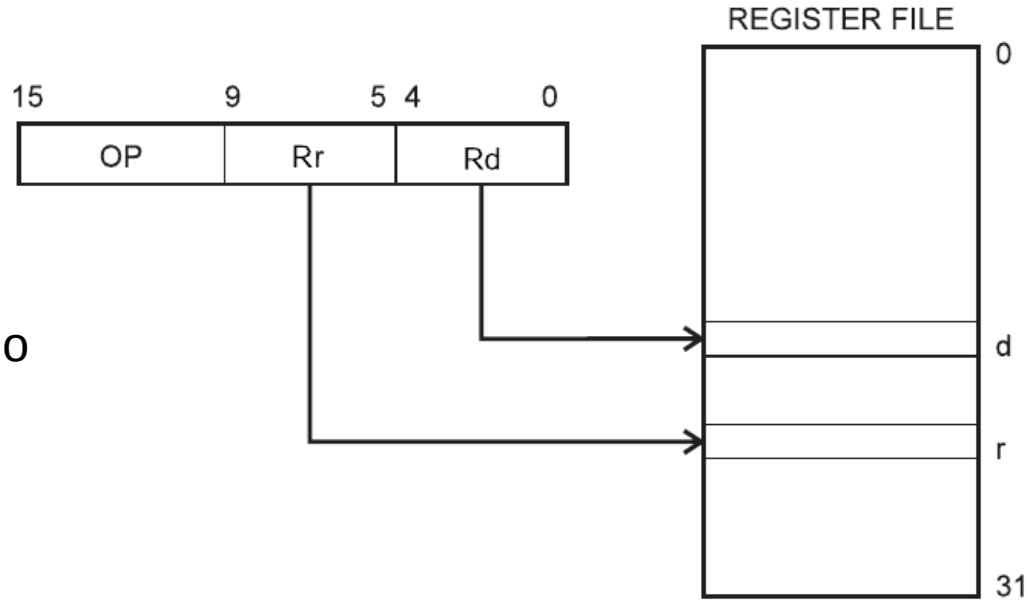
SRAM

- ATMega328P possui 2 Kbytes de memória principal implementada com SRAM
  - Alguns segmentos são reservados
    - 0x00 a 0x1F são os 32 registradores
    - 0x20 a 0x5F são registradores de I/O
      - I/O mapeada em memória



Formato de instruções

- A maioria das instruções ocupam 16 bits
- Existem diversos formatos de instrução
  - Veja no *AVR Instruction Set Manual*
- OPCodes de tamanho variável
- Exemplo
  - Register Direct
    - OP → Operação
    - Rr → Registrador fonte 1
    - Rd → Registrador fonte 2 e destino



Instruções assembly

- As instruções assembly seguem a mesma ordem utilizada no MIPS
- OPCODE destino, fonte
  - Exemplo
    - ldi r16,0xF
    - Carregue o imediato 0xF para o registrador r16
- O número de operandos varia de acordo com a operação

Olá mundo (mais uma vez)

- Crie um arquivo led.s em um diretório de sua preferência
- Adicione o seguinte

```
.global    main
        .type      main, @function
main:
        ldi r16,0b00000001
        out 0x4,r16
LOOP:
        out 0x5,r16
        jmp LOOP
```

Olá mundo (mais uma vez)

- Crie um arquivo led.s em um diretório de sua preferência
- Adicione o seguinte

```
.global    main
        .type      main, @function
main:
        ldi r16,0b00000001
        out 0x4,r16
LOOP:
        out 0x5,r16
        jmp LOOP
```

ldi carrega um imediato para registradores **16 a 31.**
Usando para carregar a constante em r16.

Olá mundo (mais uma vez)

- Crie um arquivo led.s em um diretório de sua preferência
- Adicione o seguinte

```
.global    main
        .type      main, @function
main:
        ldi r16,0b00000001
        out 0x4,r16
LOOP:
        out 0x5,r16
        jmp LOOP
```

O “registrador” **DDRB** (Seção 14.4 do manual), que fica no endereço **0x4**, configura os pinos da porta B são para entrada (0) ou saída(1).

out carrega para o endereço de I/O o valor que está armazenado no registrador

Estamos carregando 0b00000001 para o endereço 0x4, fazendo com que o o pino 0 da Porta B seja usado como saída, e os demais como entrada

Olá mundo (mais uma vez)

- Crie um arquivo led.s em um diretório de sua preferência
- Adicione o seguinte

```
.global    main
        .type      main, @function
main:
        ldi r16,0b00000001
        out 0x4,r16
LOOP:
        out 0x5,r16
        jmp LOOP
```

Registrador PORTB (endereço 0x5) define se um sinal alto ou baixo é enviado para os pinos de **saída** da porta B. Vamos enviar 1 lógico (5 volts) em PB0.

Olá mundo (mais uma vez)

- Crie um arquivo led.s em um diretório de sua preferência
- Adicione o seguinte

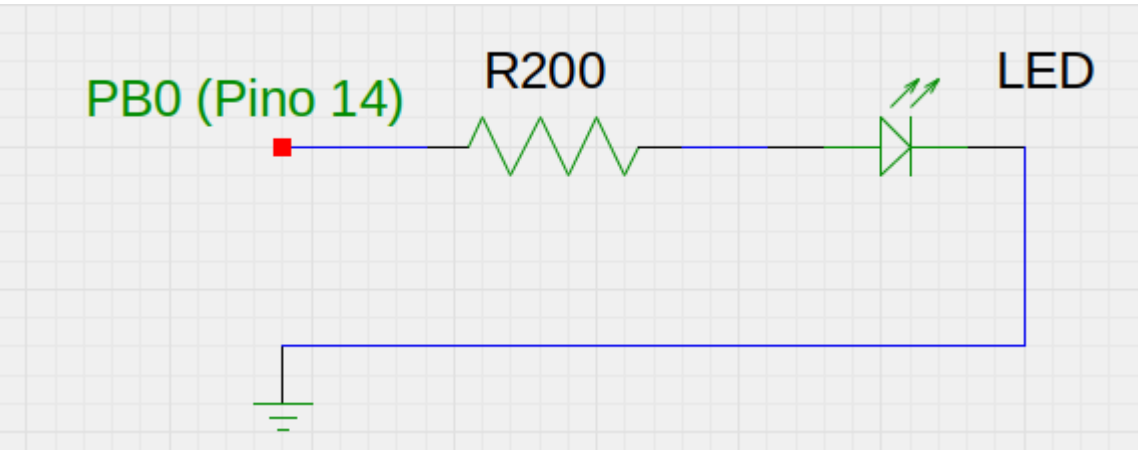
```
.global    main
        .type      main, @function
main:
        ldi r16,0b00000001
        out 0x4,r16
LOOP:
        out 0x5,r16
        jmp LOOP
```

Salte para o rótulo LOOP – jmp faz um salto incondicional e de endereçamento direto

Criamos um loop infinito

Ligação Física

- Utilize o mesmo circuito da aula passada
  - LED ligado em PB0



Montando e carregando

- Vamos usar o *avr-gcc* para **montar o assembly**
  - Poderíamos usar o *avr-as* e o *avr-ld* (como feito com o x86 e o *as* com *ld*)
    - Mas precisamos passar detalhes extras para o ld, como o início do endereçamento de programa na memória do nosso microcontrolador
    - Vamos evitar esses detalhes (pelo menos por enquanto)

Montando e carregando

- Para montar
  - *avr-gcc -Wall -Os -mmcu=atmega328p -o led led.s*
- Para carregar na memória (mesmo procedimento da aula anterior)
  - *avrdude -b 9600 -p m328p -c usbasp -e -U flash:w:led*

Exercício

1. Ligue outro LED, agora em PB1
   - Não esqueça do **resistor**
   - Não esqueça de configurar o pino correto da PORTA B como saída
   - Os dois LEDS (o de PB0 e o de PB1) devem estar sempre ligados

Referências

- T. S. Margush. **Some Assembly Required: Assembly Language Programming with the AVR Microcontroller.** 2016.
- S. Naimi, S. Naimi, M. Mazidi. **The Avr Microcontroller and Embedded Systems Using Assembly and C.** 2010**.**
- **megaAVR® Data Sheet.** Microchip, 2018.
- **AVR Instruction Set Manual**. Microchip, 2016.
- D. Patterson; J. Henessy. **Organização e Projeto de Computadores**: **a Interface Hardware/Software.** 5a Edição. Elsevier Brasil, 2017.