Pilha, Ponteiro da Pilha e Sub-rotinas

AVR ATmega328p

Prof. Roberto de Matos

roberto.matos@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Objetivo

- Entender o que é e como funciona uma pilha de microcontrolador
- Praticar com as instruções que manipulam a pilha diretamente (`push` e `pop`)
- Criar Sub-rotinas (`rcall`, `call` e `ret`)
- Entender o que é salvar o contexto e passar parâmetros

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

Objetivo

- Entender o que é e como funciona uma pilha de microcontrolador
- Praticar com as instruções que manipulam a pilha diretamente (`push` e `pop`)
- Criar Sub-rotinas (`rcall`, `call` e `ret`)
- Entender o que é salvar o contexto e passar parâmetros

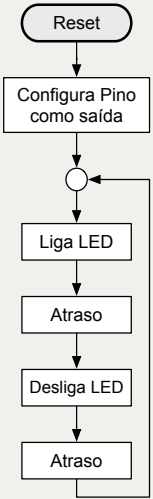
Referências:

- Datasheet do ATmega328p
- Manual online do montador
- Manual do conjunto de instruções do AVR
- AVR e Arduino: Técnicas de Projeto, 2ª ed.[1] (Leitura recomendada: Seção 2.1.2)

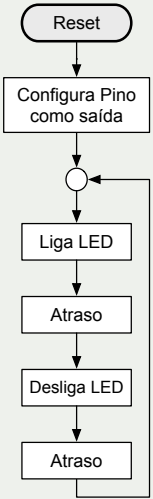
[1] Alguns exemplos e imagens dessa apresentação são retirados desse livro.

Motivação

Como piscar um LED?



Como piscar um LED?

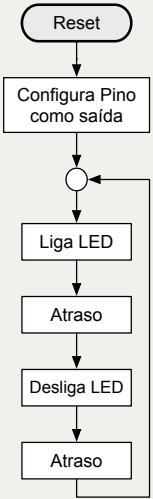


- Contagem do número de ciclos:

```
loop:
    dec   r16
    brne  loop
    dec   r17
    brne  loop
```

$$\begin{bmatrix} 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = 197119\ ciclos$$

Como piscar um LED?



- Contagem do número de ciclos:

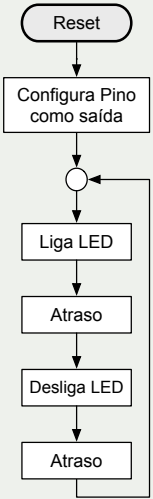
```
loop:
    dec  r16
    brne loop
    dec  r17
    brne loop
```

$$\begin{bmatrix} 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = 197119\ ciclos$$

- Cálculo do atraso a partir da frequência de operação da CPU:

$$atraso = num\_ciclos \times \frac{1}{freq\_CPU}$$

Como piscar um LED?



- Contagem do número de ciclos:

```
loop:
    dec  r16
    brne loop
    dec  r17
    brne loop
```

$$\begin{bmatrix} 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 2\ ciclos \end{bmatrix} \times 255 + \begin{bmatrix} 767\ ciclos \\ \\ +\ 1\ ciclo \\ +\ 1\ ciclo \end{bmatrix} = 197119\ ciclos$$

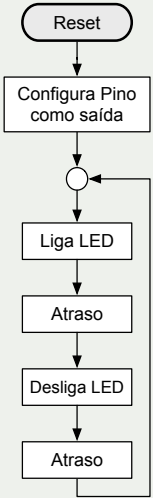
- Cálculo do atraso a partir da frequência de operação da CPU:

$$atraso = num\_ciclos \times \frac{1}{freq\_CPU}$$

- Considerando a frequência da CPU como 16MHz (período de 62,5 ns), temos:

$$atraso = 197119 \times 62,5ns = 12,32ms$$

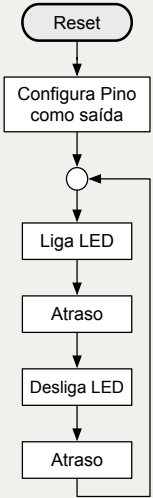
Fatoração do código




```
1  ;DEFINIÇÕES
2  .equ LED = PB5   ;LED é o substituto de PB5
3
4  start:
5    SBI DDRB, LED  ;configura pino LED como saída
6
7  main:
8    SBI PORTB, LED ;coloca o pino PB5 em 5V
9
10   ;atraso de aprox. 200ms
11   LDI R19,16
12 loop:
13   DEC  R17       ;decrementa R17, começa com 0x00
14   BRNE loop      ;se R17>0 fica decrementando R17
15   DEC  R18       ;decrementa R18, começa com 0x00
16   BRNE loop      ;se R18>0 volta decrementar R18
17   DEC  R19       ;decrementa R19
18   BRNE loop      ;se R19>0 vai para volta
19
20   CBI PORTB, LED ;coloca o pino PB5 em 0V
```

```
21
22   ;atraso de aprox. 200ms
23   LDI R19,16
24 loop2:
25   DEC  R17       ;decrementa R17, começa com 0x00
26   BRNE loop2     ;se R17>0 fica decrementando R17
27   DEC  R18       ;decrementa R18, começa com 0x00
28   BRNE loop2     ;se R18>0 volta decrementar R18
29   DEC  R19       ;decrementa R19
30   BRNE loop2     ;se R19>0 vai para volta
31
32   RJMP main      ;volta para main
```


Fatoração do código



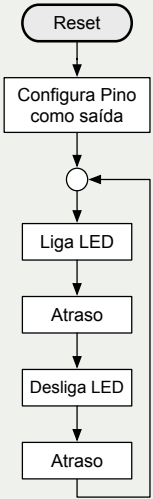

```
1  ;DEFINIÇÕES
2  .equ LED = PB5   ;LED é o substituto de PB5
3
4  start:
5    SBI DDRB, LED  ;configura pino LED como saída
6
7  main:
8    SBI PORTB, LED ;coloca o pino PB5 em 5V
9
10   ;atraso de aprox. 200ms
11   LDI R19,16
12 loop:
13   DEC  R17       ;decrementa R17, começa com 0x00
14   BRNE loop      ;se R17>0 fica decrementando R17
15   DEC  R18       ;decrementa R18, começa com 0x00
16   BRNE loop      ;se R18>0 volta decrementar R18
17   DEC  R19       ;decrementa R19
18   BRNE loop      ;se R19>0 vai para volta
19
20   CBI PORTB, LED ;coloca o pino PB5 em 0V
```

```
21
22   ;atraso de aprox. 200ms
23   LDI R19,16
24 loop2:
25   DEC  R17       ;decrementa R17, começa com 0x00
26   BRNE loop2     ;se R17>0 fica decrementando R17
27   DEC  R18       ;decrementa R18, começa com 0x00
28   BRNE loop2     ;se R18>0 volta decrementar R18
29   DEC  R19       ;decrementa R19
30   BRNE loop2     ;se R19>0 vai para volta
31
32   RJMP main      ;volta para main
```

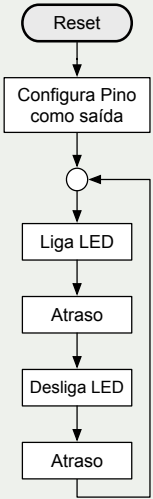
Solução Ruim!

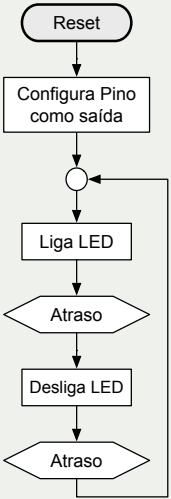
Repete trechos de código

Fatoração do código

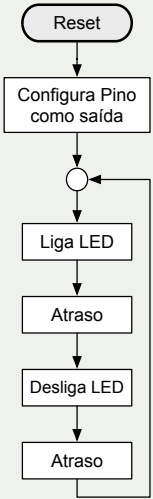


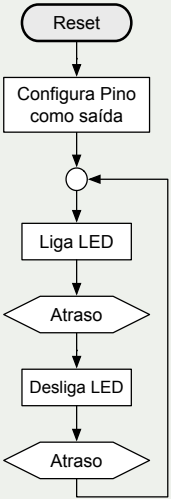
Fatoração do código

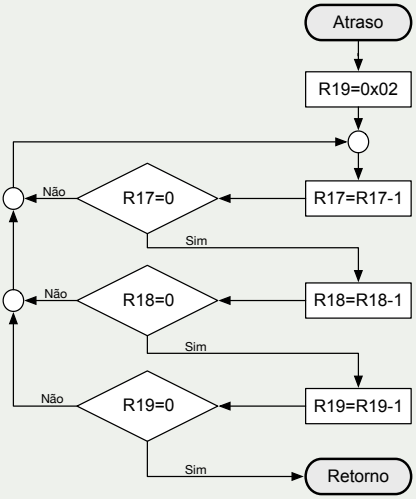




Fatoração do código

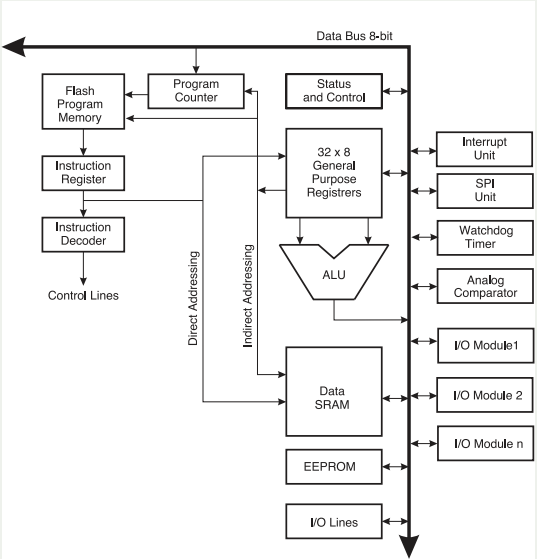




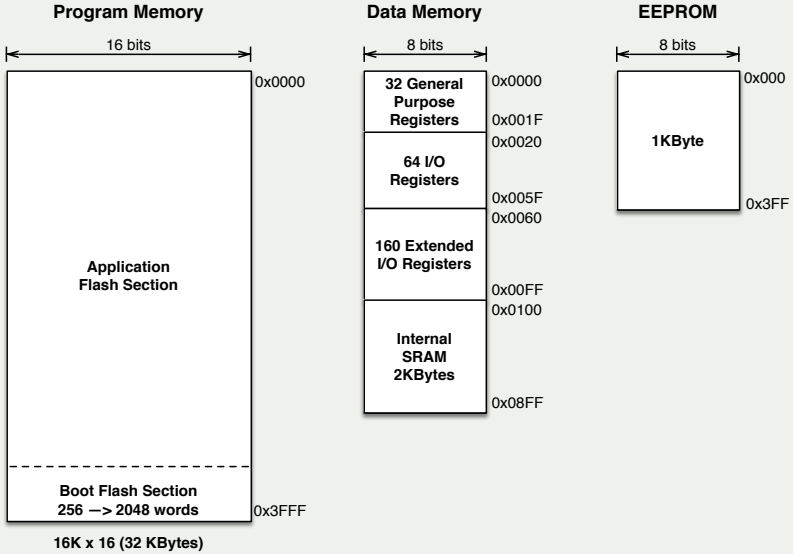


Relembrando ...

Núcleo do AVR



Memórias ATmega328p



Pilha e Ponteiro da Pilha

Pilha e Ponteiro da Pilha

- Pilha:
  - Utiliza parte da SRAM
  - Armazena temporariamente dados de variáveis locais e retorno de sub-rotinas/interrupções.
- Ponteiro de Pilha ou *Stack Pointer* (SP):
  - Pós-decrementado quando um dado é adicionado na pilha.
  - Pré-incrementado quando um dado é retirado da pilha
  - SP deve ser inicializado para o último endereço da RAM
  - O ATmega328 o endereço é 0x8FF e já é carregado durante o reset.
  - Alguns MCUs da família ATmega precisam ter o SP inicializado pelo programador.

Pilha e Ponteiro da Pilha

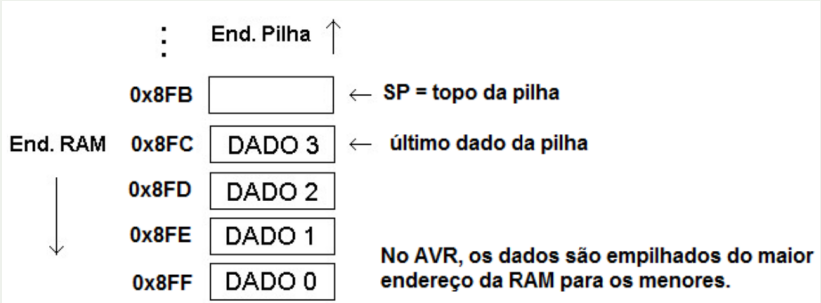


Fig. 2.7 – *Stack Pointer*.

Pilha e Ponteiro da Pilha

Bit		15	14	13	12	11	10	9	8
	SPH	-	-	-	-	SP11	SP10	SP9	SP8
	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Bit		7	6	5	4	3	2	1	0
Lê/Escreve		L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
		L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
Valor Inicial		-	-	-	-	1	0	0	0
		1	1	1	1	1	1	1	1

Fig. 2.8- Detalhamento dos registradores do *Stack Pointer* (valor inicial 0x8FF).

Instruções que manipulam a Pilha

Instrução	SP	Descrição
PUSH	Decrementa em 1	Um dado é colocado na pilha (1 byte).
CALL ICALL RCALL	Decrementa em 2	O endereço de retorno é colocado na pilha quando uma chamada de sub-rotina ou interrupção acontece (o endereço possui 2 bytes).
POP	Incrementa em 1	O dado do topo da pilha e retirado (1 byte).
RET RETI	Incrementa em 2	O endereço de retorno é retirado da pilha quando se retorna de uma sub-rotina ou interrupção (o endereço possui 2 bytes).

Exemplo: Armazenando de dados na Pilha

`push`, `pop`

Instruções `push` e `pop`

```
1 LDI r16,0x01  ; carrega r16 com 0x01
2 LDI r17,0x02  ; carrega r17 com 0x02
3
4 PUSH  r16     ; salva r16 na pilha
5 PUSH  r17     ; salva r17 na pilha
6
7 POP r17       ; restaura r17 da pilha
8 POP r16       ; restaura r16 da pilha
```


Sub-rotina

Visão Geral

- O mecanismo de sub-rotina permite:
  - Organizar o código em bloco modulares, inclusive bibliotecas
  - Continuar a execução de onde foi chamada sem rótulos de forma “automática”
  - Reutilização de código
- Exemplo:

```
1 main:
2   RCALL sub_rotina
3   RJMP main
4
5 sub_rotina:
6   ;executa sub-rotina
7   RET
```


Rótulo vs. Sub-rotina

- Rótulo:
  - Utilizado para identificar um ponto na memória de programa
  - Deve ser utilizado com instruções de desvio condicional e incondicional
  - Não tem mecanismo de retorno

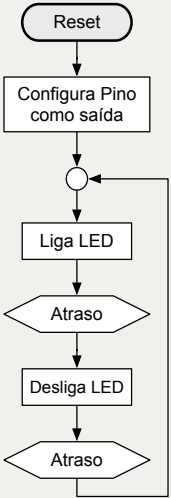
```
1 rótulo:
2   ;trecho de programa a ser executado a partir do Rótulo
3   ;continua sequencialmente até encontrar outro desvio
```

- Sub-rotina:
  - Começa com um rótulo e termina com um `ret`
  - Deve ser chamado com `rcall, icall e call`
  - Retorna de onde foi chamada
  - Utiliza a pilha para armazenar o endereço de retorno

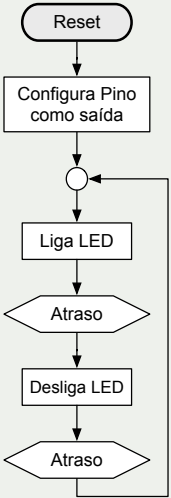
```
1 sub_rotina:
2   ;pode saltar para rótulos quantas vezes quiser
3   ;pode executar outras sub-rotinas (rcall, icall e call)
4   ;desde que a última instrução seja ret
5
6   RET
```


Exemplos: Sub-rotinas

Exemplo: Sub-rotina *delay*

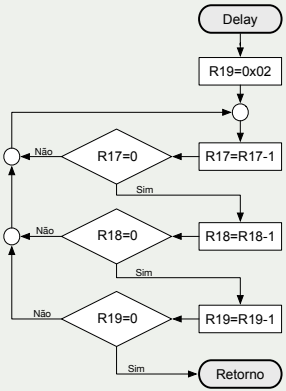


Exemplo: Sub-rotina *delay*



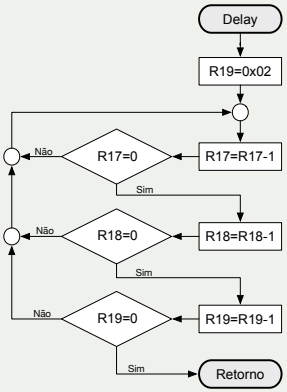

```
1  ;DEFINIÇÕES
2  .equ LED = PB5    ;LED é o substituto de PB5
3
4  start:
5    SBI DDRB, LED   ;configura pino LED como saída
6
7  main:
8    SBI PORTB, LED ;coloca o pino PB5 em 5V
9    RCALL delay    ;chama a sub-rotina de atraso
10   CBI PORTB, LED ;coloca o pino PB5 em 0V
11   RCALL delay    ;chama a sub-rotina de atraso
12   RJMP main      ;volta para main
13
14 delay:           ;atraso de aprox. 200ms
15   LDI R19,16
16 loop:
17   DEC  R17       ;decrementa R17, começa com 0x00
18   BRNE loop      ;enquanto R17 > 0 fica decrementando R17
19   DEC  R18       ;decrementa R18, começa com 0x00
20   BRNE loop      ;enquanto R18 > 0 volta decrementar R18
21   DEC  R19       ;decrementa R19
22   BRNE loop      ;enquanto R19 > 0 vai para volta
23   RET
```

Exemplo: Sub-rotina *Delay*




```
1  ;DEFINIÇÕES
2  .equ LED = PB5   ;LED é o substituto de PB5
3
4  start:
5    SBI DDRB, LED  ;configura pino LED como saída
6
7  main:
8    SBI PORTB, LED ;coloca o pino PB5 em 5V
9    RCALL delay    ;chama a sub-rotina de atraso
10   CBI PORTB, LED ;coloca o pino PB5 em 0V
11   RCALL delay    ;chama a sub-rotina de atraso
12   RJMP main      ;volta para main
13
14 delay:           ;atraso de aprox. 200ms
15   LDI R19,16
16 loop:
17   DEC  R17       ;decrementa R17, começa com 0x00
18   BRNE loop      ;enquanto R17 > 0 fica decrementando R17
19   DEC  R18       ;decrementa R18, começa com 0x00
20   BRNE loop      ;enquanto R18 > 0 volta decrementar R18
21   DEC  R19       ;decrementa R19
22   BRNE loop      ;enquanto R19 > 0 vai para volta
23   RET
```

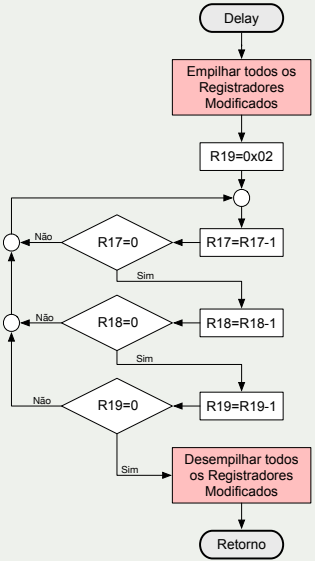

Exemplo: Sub-rotina *Delay*




```
1  ;DEFINIÇÕES
2  .equ LED = PB5   ;LED é o substituto de PB5
3
4  start:
5    SBI DDRB, LED  ;configura pino LED como saída
6
7  main:
8    SBI PORTB, LED ;coloca o pino PB5 em 5V
9    RCALL delay    ;chama a sub-rotina de atraso
10   CBI PORTB, LED ;coloca o pino PB5 em 0V
11   RCALL delay    ;chama a sub-rotina de atraso
12   RJMP main      ;volta para main
13
14 delay:           ;atraso de aprox. 200ms
15   LDI R19,16
16 loop:
17   DEC  R17       ;decrementa R17, começa com 0x00
18   BRNE loop      ;enquanto R17 > 0 fica decrementando R17
19   DEC  R18       ;decrementa R18, começa com 0x00
20   BRNE loop      ;enquanto R18 > 0 volta decrementar R18
21   DEC  R19       ;decrementa R19
22   BRNE loop      ;enquanto R19 > 0 vai para volta
23   RET
```

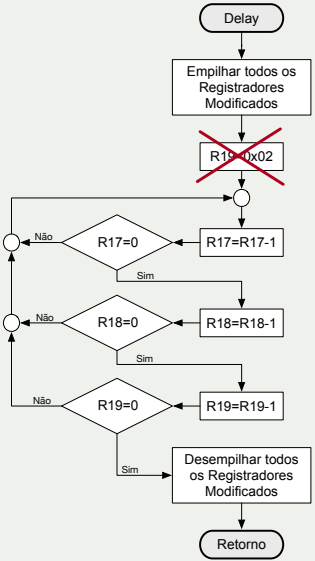
Como melhorar?

Melhorando: Salvando o contexto dos registradores modificados




```
1  delay:          ;atraso de aprox. 200ms
2
3    PUSH r17      ; Salva os valores de r17,
4    PUSH r18      ; ... r18,
5    PUSH r19      ; ... r19,
6    IN r17,SREG   ;
7    PUSH r17      ; ... e SREG na pilha.
8
9    ; Executa sub-rotina :
10   CLR r17
11   CLR r18
12   LDI R19,16
13
14 loop:
15   DEC  R17      ;decrementa R17, começa com 0x00
16   BRNE loop     ;enquanto R17 > 0 fica decrementando R17
17   DEC  R18      ;decrementa R18, começa com 0x00
18   BRNE loop     ;enquanto R18 > 0 volta decrementar R18
19   DEC  R19      ;decrementa R19
20   BRNE loop     ;enquanto R19 > 0 vai para volta
21
22   POP r17
23   OUT SREG, r17 ; Restaura os valores de SREG,
24   POP r19       ; ... r19
25   POP r18       ; ... r18
26   POP r17       ; ... r17 da pilha
27
28   RET
```

Melhorado mais: Usando os registradores para passar parâmetro




```
1  ;-------------------------------------------------------------
2  ;SUB-ROTINA DE ATRASO Programável
3  ; Depende do valor de R19 carregado antes da chamada.
4  ; Ex.: - R19 = 16 --> 200ms
5  ;      - R19 = 80 --> 1s
6  ;-------------------------------------------------------------
7  delay:
8    PUSH r17        ; Salva os valores de r17,
9    PUSH r18        ; ... r18,
10   IN r17,SREG     ; ...
11   PUSH r17        ; ... e SREG na pilha.
12
13   ; Executa sub-rotina :
14   CLR r17
15   CLR r18
16 loop:
17   DEC  R17        ;decrementa R17, começa com 0x00
18   BRNE loop       ;enquanto R17 > 0 fica decrementando R17
19   DEC  R18        ;decrementa R18, começa com 0x00
20   BRNE loop       ;enquanto R18 > 0 volta decrementar R18
21   DEC  R19        ;decrementa R19
22   BRNE loop       ;enquanto R19 > 0 vai para volta
23
24   POP r17
25   OUT SREG, r17   ; Restaura os valores de SREG,
26   POP r18         ; ... r18
27   POP r17         ; ... r17 da pilha
28
29   RET
```

Experimentos

Experimentos no MPLAB IDE

- Simule todos os trechos de códigos apresentados.
- Faça a atividade proposta no Moodle.

Fim!