“Uma imagem vale mais do que mil palavras, mas ocupa 3 mil vezes mais espaço em disco.”

Divisões e Debounce

Paulo Ricardo Lisboa de Almeida

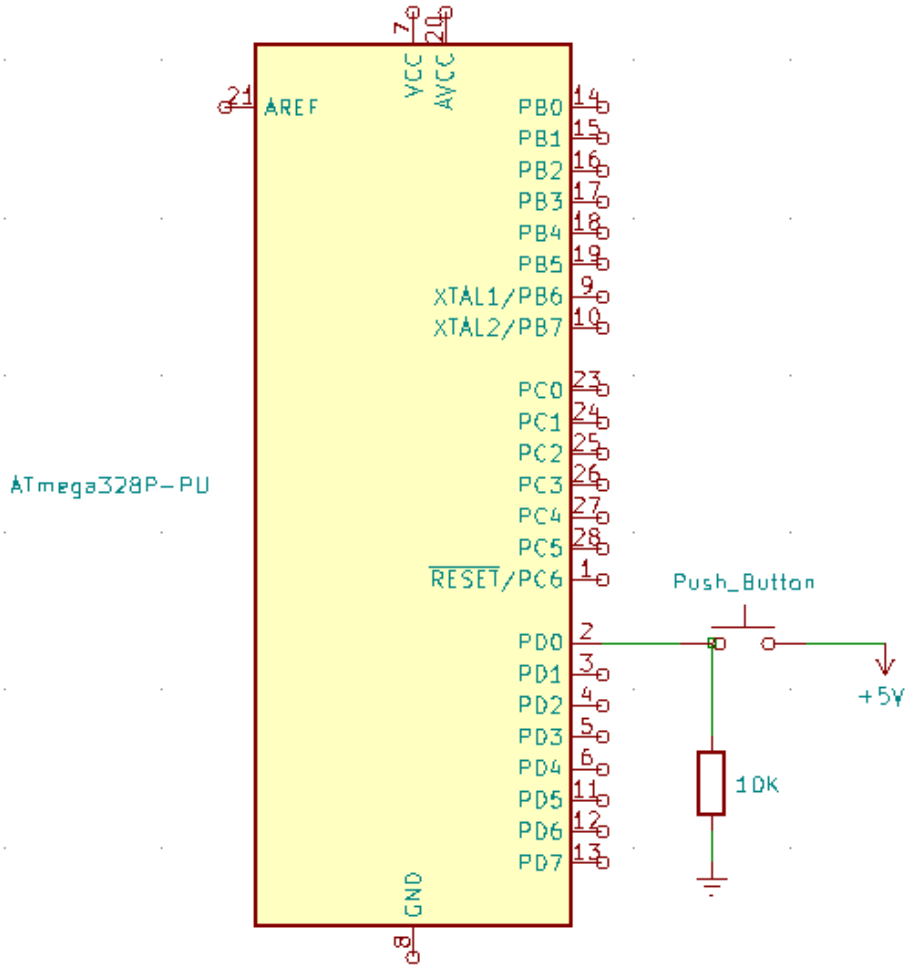
UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Continuando

- Vamos continuar com o problema da aula passada
  - Mantenha o display de 7 segmentos conectado na protoboard

Botão em PD0

- Adicione um botão em PD0
  - Não esqueça do pull-down



Configurando

- Constantes necessárias

```
.equ DDRB, 0x04
.equ DDRD, 0x0A
.equ PORTB, 0x05
.equ PIND, 0x09

.equ RAMEND, 0x8FF
.equ SPH, 0x3E
.equ SPL, 0x3D
```

Configuração inicial

```
;ajuste do stack pointer (por precaução)
ldi R16, hi8(RAMEND)
ldi R17, lo8(RAMEND)
out SPL, R17
out SPH, R16
;fim do ajuste do stack pointer

ldi r16,0b11111111 ;carrega 0 para r16
out DDRB,r16 ;Todos pinos de PB como saída

eor r16, r16 ; 0 em r16
out DDRD, r16 ; todos pinos de PORTD como entrada
```

Loop principal

```
    ldi r31, 0x0 ;inicializa com 0
LOOP:
    in r30, PIND
    andi r30, 0x1 ;and para ficar somente com o último bit
    add r31,r30
    mov r25, r31
    rcall calcula7segs ;salto curto relativo ao PC
    out PORTB,r25
    jmp LOOP
```

UDESC

Loop principal

Onde está o problema?

```
    ldi r31, 0x0 ;inicializa com 0
LOOP:
    in r30, PIND
    andi r30, 0x1 ;and para ficar somente com o último bit
    add r31,r30
    mov r25, r31
    rcall calcula7segs ;salto curto relativo ao PC
    out PORTB,r25
    jmp LOOP
```

UDESC

Loop principal

Verificamos a cada iteração se o botão está pressionado.
Isso pode ocorrer literalmente milhares de vezes por segundo.
Um “aperto” do usuário vai gerar um acréscimo gigantesco no contador.
Soluções?

```
    ldi r31, 0x0 ;inicializa com 0
LOOP:
    in r30, PIND
    andi r30, 0x1 ;and para ficar somente com o último bit
    add r31,r30
    mov r25, r31
    rcall calcula7segs ;salto curto relativo ao PC
    out PORTB,r25
    jmp LOOP
```

Soluções

- Podemos aguardar um tempo a cada iteração
  - Dar tempo do usuário liberar o push button
    - Delay de debounce
- Podemos utilizar interrupções
  - Veremos nas próximas aulas

Solução

Aguarde um pouco antes da próxima iteração.

```
        ldi r31, 0x0
LOOP:
        in r30, PIND
        andi r30, 0x1 ;and para ficar somente com o último bit
        add r31,r30
        mov r25, r31
        rcall calcula7segs ;salto curto relativo ao PC
        out PORTB,r25
        rcall delay_omic ;salto curto relativo ao PC
        jmp LOOP
```

Pooling

- Implementamos uma técnica de pooling
  - Revise suas aulas de sistemas operacionais
- Quais as vantagens e desvantagens?

Pooling

- Implementamos uma técnica de pooling
  - Revise suas aulas de sistemas operacionais
- Quais as vantagens e desvantagens?

  + Simples

  - Gastamos tempo com o delay e com o loop que verifica eternamente o estado do botão

  - Se logo após a verificação, por exemplo
    - O usuário pressionar o botão
    - Vamos precisar esperar o loop se repetir, para só então tomar a ação necessária
      - Especialmente ruim para loops que demoram muito tempo e que **cuidam de algo crítico**
      - Ex.: você não quer um atraso de 0,5 segundos entre pisar no freio do seu carro, e o sistema ABS ser acionado

Divisões

- Vamos pegar o resto da divisão por 10 para exibir no display
  - Evitar um overflow no display
- Procure pela instrução de divisão no PIC e no ATmega

Divisões

- Vamos pegar o resto da divisão por 10 para exibir no display
  - Evitar um overflow no display
- Procure pela instrução de divisão no PIC e no ATmega

Divisões

- O ATmega possui instruções para multiplicação, mas não para divisão
- O PIC não possui instruções de multiplicação, nem divisão
- Pesquise na internet soluções por solftware para fazer multiplicação e divisão
  - **Alguma boa solução?**

Divisões via software

- Boa parte das “soluções” propostas podem envolver um loop de somas/subtrações
  - Exemplo: 10*4 é o mesmo que 4 somado 10x
- **Péssima ideia**
  - Quem diria, o pessoal que manda suas opiniões/soluções na internet está errado!
- Soluções mais inteligentes envolvem algumas poucas somas/subtrações e shifts
  - Veja essas soluções no capítulo 3 de Patterson e Henessy (2017)
  - A solução trata de uma implementação via hardware, mas é facilmente simulada no PIC, ATMega via software

Para o PIC

- Algumas soluções relativamente otimizadas podem ser encontradas em
  - piclist.com/techref/microchip/math/index.htm
  - www.convict.lu/Jeunes/Math/Fast_operations2.htm
  - Note que não são “soluções oficiais”, e sim soluções propostas por desenvolvedores
    - **Podem conter erros**
    - Verifique se as soluções estão corretas
- Disponibilizei uma adaptação minha para o PIC16F628A no Moodle
  - divisao16por8PIC.asm
- Você também pode fazer a divisão em C, compilar e verificar o assembly gerado pelo compilador

ATMega

- Para o Atmega, vamos utilizar a solução disponibilizada em
  - T. S. Margush. **Some Assembly Required: Assembly Language Programming with the AVR Microcontroller.** 2016.
- Disponibilizada no Moodle
  - Toma um *unsigned int* (2 bytes) que é dividido por um byte
  - Retorna como resposta o resultado da divisão inteira e o resto

Exercício

- Adicione outro botão para decrementar o contador em PD1 (agora temos um botão para incrementar, e outro para decrementar). Utilize a rotina de divisão para sempre exibir o resto da divisão do contador no display de 7 segmentos.
  - Submeta
    - O código fonte do seu trabalho
    - Um vídeo do demonstrando o funcionamento do seu sistema

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.
- **ATmega328P Automotive - Complete Datasheet**. Microchip.
- **AVR Instruction Set Manual**. Microchip, 2016.
- D. Patterson; J. Henessy. **Organização e Projeto de Computadores**: **a Interface Hardware/Software.** 5a Edição. Elsevier Brasil, 2017.