“Responda: por que o tiranossauro não consegue coçar as costas?”

Interrupções

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Interrupções

- O que é uma interrupção (exceção)?

Interrupções

- Uma interrupção é um evento que pode interromper o fluxo normal de execução do programa
  - Acionadas por, por exemplo
    - Sinais externos
    - Pelo software em execução
      - **Exemplo?**
    - Pelo hardware

Interrupções

- Uma interrupção é um evento que pode interromper o fluxo normal de execução do programa
  - Acionadas por, por exemplo
    - Sinais externos
    - Pelo software em execução
      - **Exemplo: syscall**
    - Pelo hardware

Interrupções

- Ao receber uma interrupção
  - Suspendemos a execução do programa “normal”
  - Executamos alguma rotina para tratar a interrupção
  - Se possível, retornamos os programa principal quando concluímos

Fontes

- Algumas fontes externas de interrupção no ATMega
  - Sinal enviado para um pino
  - Timers
  - Watchdog

Status Register

- Status Register (SREG)
  - Seção 7.3.1 do manual

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Bit 7 (I)
    - Global Interrupt Enable
      - 0 → Todas interrupções desabilitadas
      - 1 → Interrupções habilitadas

Status Register

- Você pode carregar um valor para o S*tatus Register* manualmente, ou usar as instruções
  - *sei* → Global interrupt Enable
    - Coloca 1 no sétimo bit do *status register*
  - *cli* → Global interrupt Disable
    - Coloca 1 no sétimo bit do *status register*

Interrupções vetorizadas

- No ATMega328p as interrupções são vetorizadas
  - O que isso significa?

Interrupções vetorizadas

- No ATMega328p as interrupções são vetorizadas
  - O contador de programa é redirecionado para um endereço de memória específico para cada interrupção
    - No endereço de memória começa a rotina de tratamento
      - Mesma estratégia do x86-64

Problemas a se considerar

- Imagine que entre essas duas instruções

  cpi r16, 0
  breq algumLugar

- Ocorre uma interrupção
  - Que problemas podem acontecer?

Problemas a se considerar

- Imagine que entre essas duas instruções

  cpi r16, 0
  breq algumLugar

- Ocorre uma interrupção
  - A rotina de tratamento de interrupção pode precisar fazer alguma comparação
    - Vai “estragar” a comparação feita pelo CPI do programa “normal”
    - Ex.: Bit Z, registrador SREG
    - Idealmente
      - Sua rotina de tratamento de interrupções deve **salvar o estado de todos os registradores** para que o programa continue executando normalmente
        - Inclusive o **SREG**

Exemplo

- Exemplo de tratador de (Margush; 2016)

```
tratador:
    push r16
    in r16, SREG
    push r16

    ;faz o tratamento aqui

    pop r16
    out SREG, r16
    pop r16
    reti
```

Interrupções nos pinos INT0 e INT1

- Vamos usar o **pino INT0** para interrupção
  - Configurar no “registrador” EICRA
    - External Interrupt Control Register A
    - Seção 13.2 do manual

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 13-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

UDESC

Habilitando interrupções INT0 e INT1

- Registrador EIMSK
  - External Interrupt Mask Register
  - Seção 13.2.2 do manual

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Ao setar os bits 0 ou 1 (INT0/INT1) de EIMSK, habilitamos as interrupções nos pinos INT0 e INT1

Configurações

```
main:
     ;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

     ;configurando portas
     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
     ;fim da configuração de portas

     ;configuração inicial
     ldi r31, 0x0

     ;habilitando interrupções
     ldi r16,0b00000001
     sts EICRA,r16 ;interrupção por mudança em INT0
     ldi r16, 0b00000001
     out EIMSK, r16 ; interrupção em INT0 habilitada
     sei ;habilitando interrupções globais
     ;fim interrupções

LOOP:
     ; … programa principal
     jmp LOOP
```

UDESC

Tratador de interrupção

- O valor a ser exibido no display está em r31
  - O tratador de INT0 vai somar 1 em r31

Tratador de interrupção

- O valor a ser exibido no display está em r31
  - O tratador de INT0 vai somar 1 em r31

```
int0BtnSoma:
    push r16
    in r16, SREG
    push r16

    in r16, PIND
    andi r16, 0b00000100 ;r16 <- 0b100 se o INT0 estava ativo
    breq saidaInt0 ; salte para saída se deu 0 a comparação
    inc r31
saidaInt0:
    pop r16
    out SREG, r16
    pop r16
    reti
```

Tabela de interrupção

- Antes de inicializar o microcontrolador
  - Precisamos carregar os endereços dos tratadores de interrupção na tabela de endereços de interrupção
  - Veja os endereços na seção 12.4 do manual

Tabela de interrupção

- Antes de inicializar o microcontrolador
  - Precisamos carregar os endereços dos tratadores de interrupção na tabela de endereços de interrupção
  - Veja os endereços na seção 12.4 do manual
    - **Palavra** 0x0 → Endereço de interrupção de reset
      - Essa interrupção não pode ser desabilitada
    - **Palavra** 0x2 → Endereço de interrupção em INT0

Atenção

- O manual não deixa isso claro, mas nas tabelas de interrupção ***Program Address*** se refere ao endereço da palavra de instrução, e não endereço real da memória
  - No ATMega328P, cada palavra de instrução ocupa 2 bytes, então precisamos multiplicar por 2 os endereços para obter os endereços reais de memória
  - A única dica que ele dá quanto a isso é no início da Seção 12, onde consta:

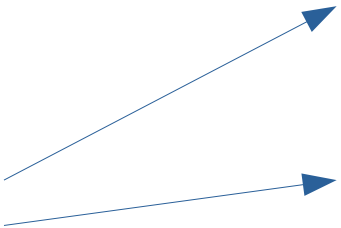
    - Each Interrupt Vector occupies two instruction words in ATmega168A/168PA and ATmega328/328P, and one instruction word in ATmega 48A/48PA and ATmega88A/88PA.

Tabela de interrupção

```
.org 0x0
    jmp main
    jmp int0BtnSoma

.org 0x34

int0BtnSoma:
    ; ...

    .global main
    .type main, @function
main:
    ;...
```

Tabela de interrupção

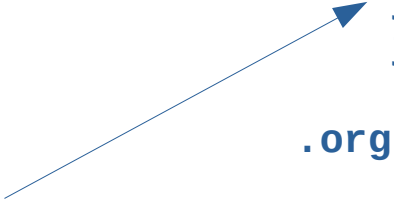
Diretivas .org instruem o montador a colocar as instruções exatamente a partir do endereço de memória especificado



```
.org 0x0
    jmp main
    jmp int0BtnSoma

.org 0x34

int0BtnSoma:
    ; ...

    .global main
    .type main, @function
main:
    ;...
```

Tabela de interrupção

Em caso de reset, vamos redirecionar para o main. Interrupções de reset não podem ser desabilitadas, e devemos redirecionar para alguma rotina **segura** que reinicie o microcontrolador (o ideal era termos uma rotina especializada em reset, mas vamos deixar assim por enquanto).



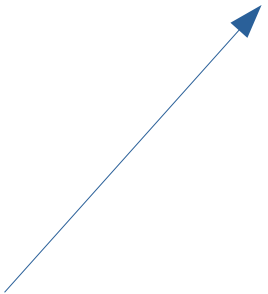
```
.org 0x0
    jmp main
    jmp int0BtnSoma

.org 0x34

int0BtnSoma:
    ; ...

    .global main
    .type main, @function
main:
    ;...
```

Compilando

- avr-gcc -Wall -Os -mmcu=atmega328p -o aula aula.s **-nostartfiles**

Compilando

- avr-gcc -Wall -Os -mmcu=atmega328p -o aula aula.s **-nostartfiles**



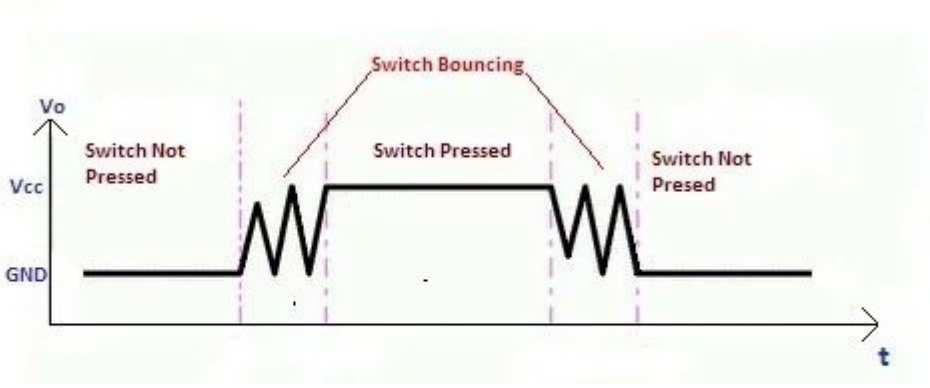
Perdi uma tarde pensando que meu programa estava incorreto, mas a maldição do avr-gcc estava sobrescrevendo a tabela de interrupções e adicionando valores padrão sem avisar. Para desabilitar esse comportamento, adicione -nostartfiles

Ainda não acabamos

- Note que ao pressionar o push button, o contador pode ser somado múltiplas vezes
  - O que há de errado agora?

Debouncing

- Para tratar a entrada, precisamos de um sistema de *debouncing* no botão
  - Botões mecânicos não são perfeitos e geram instabilidades quando pressionados/soltos
- Debouncing via hardware
  - Ex.: Utilização de Flip-Flops sincronizados e/ou capacitores
- Debouncing via software
  - **Incluir atrasos no software para esperar o sinal estabilizar**
  - Ou ainda solução via Timer
    - Veja Margush 2016



Fonte: electrosome.com/switch-debouncing

Delay Debounce

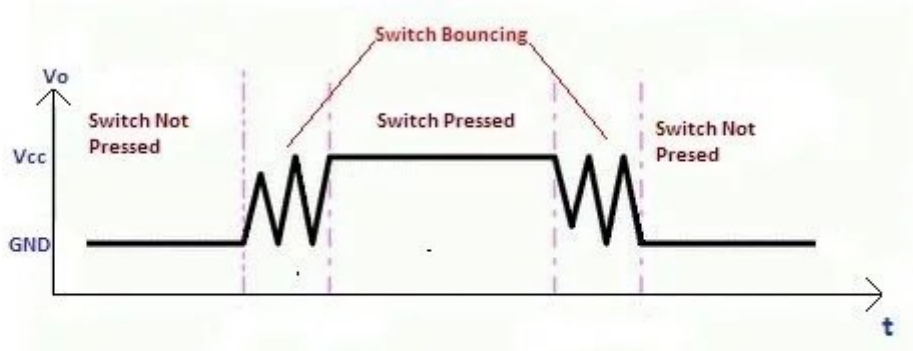
```
;ciclos de debounce: 3 + 2 + (r18-1)*3 + 1+2+4 = 774
;Com um clock de 1Mhz, esperamos 0,0008 segundos
delay_debounce: ;saltar aqui custa 3 ciclos
     ldi r18,0xFF ;custa 2 ciclos
loop_debounce: ;passa pelo loop R18-1 vezes
     dec r18 ;custa 1 ciclo
     brne loop_debounce ;custa 2 ciclos (salta), ou 1 ciclo
     ret ;custa 4 ciclos
```

```
int0BtnSoma:
     push r16
     in r16, SREG
     push r16
     ;aguardar o sinal estabilizar
     push r18 ;salvar para poder chamar delay debounce
     call delay_debounce
     pop r18
     in r16, PIND
     andi r16, 0b00000100 ;r16 <- 0b100 se o INT0 estava ativo
     breq saidaInt0 ; salte apra saída se deu 0 a comparação
     inc r31
saidaInt0:
     pop r16
     out SREG, r16
     pop r16
     reti
```

UDESC

Delay Debounce

```
;ciclos de debounce: 3 + 2 + (r18-1)*3 + 1+2+4 = 774
;Com um clock de 1Mhz, esperamos 0,0008 segundos
delay_debounce: ;saltar aqui custa 3 ciclos
     ldi r18,0xFF ;custa 2 ciclos
loop_debounce: ;passa pelo loop R18-1 vezes
     dec r18 ;custa 1 ciclo
     brne loop_debounce ;custa 2 ciclos (salta), ou 1 ciclo
     ret ;custa 4 ciclos
```



Esperar que o sinal estabilize!

```
int0BtnSoma:
     push r16
     in r16, SREG
     push r16
     ;aguardar o sinal estabilizar
     push r18 ;salvar para poder chamar delay debounce
     call delay_debounce
     pop r18
     in r16, PIND
     andi r16, 0b00000100 ;r16 <- 0b100 se o INT0 estava ativo
     breq saidaInt0 ; salte apra saída se deu 0 a comparação
     inc r31
saidaInt0:
     pop r16
     out SREG, r16
     pop r16
     reti
```

UDESC

Considerações

- As interrupções são automaticamente desabilitadas quando um tratador de interrupção é acionado
  - Elas podem ser reacionadas via *sei*, *reti*, ou alterando manualmente o registrador STATUS
    - reti retorna e aciona novamente as interrupções automaticamente antes de retornar
- O que acontece se uma interrupção ocorre enquanto estamos tratando de outra?

Exercício

- Adicione outro botão, agora em INT1
  - Ao pressionar o botão, o contador é decrementado
  - Utilize interrupções
- Submeta no Moodle o seu código fonte, e um vídeo demonstrando o funcionamento do circuito

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.