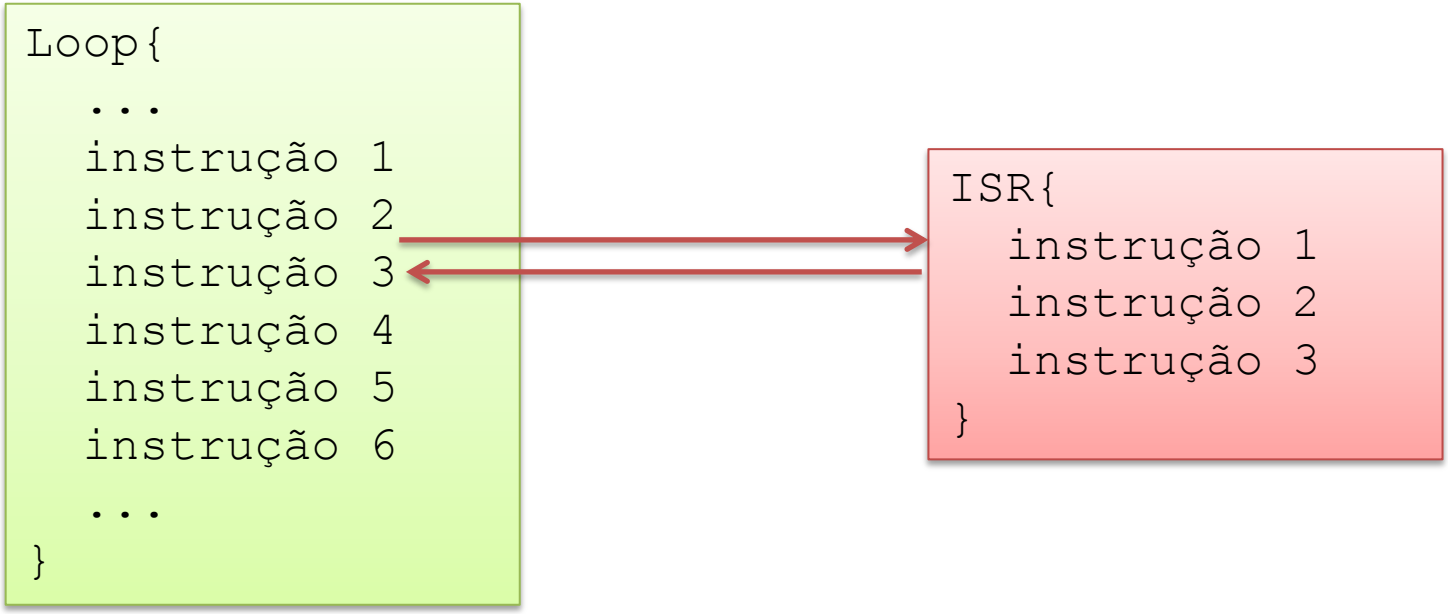
Interrupções e Processamento Concorrente

Prof. Leandro Israel Pinto

Interrupção

- Interrupção é um sinal emitido por hardware ou software indicando um evento que necessita atenção imediata;
  - Interrompe-se a função atual para processar uma função que pode ser chamada de *interrupt handler*.
    - *Ou Interrupt Service Routine (ISR);*
  - Ao término, retorna-se a função anterior;

Interrupção

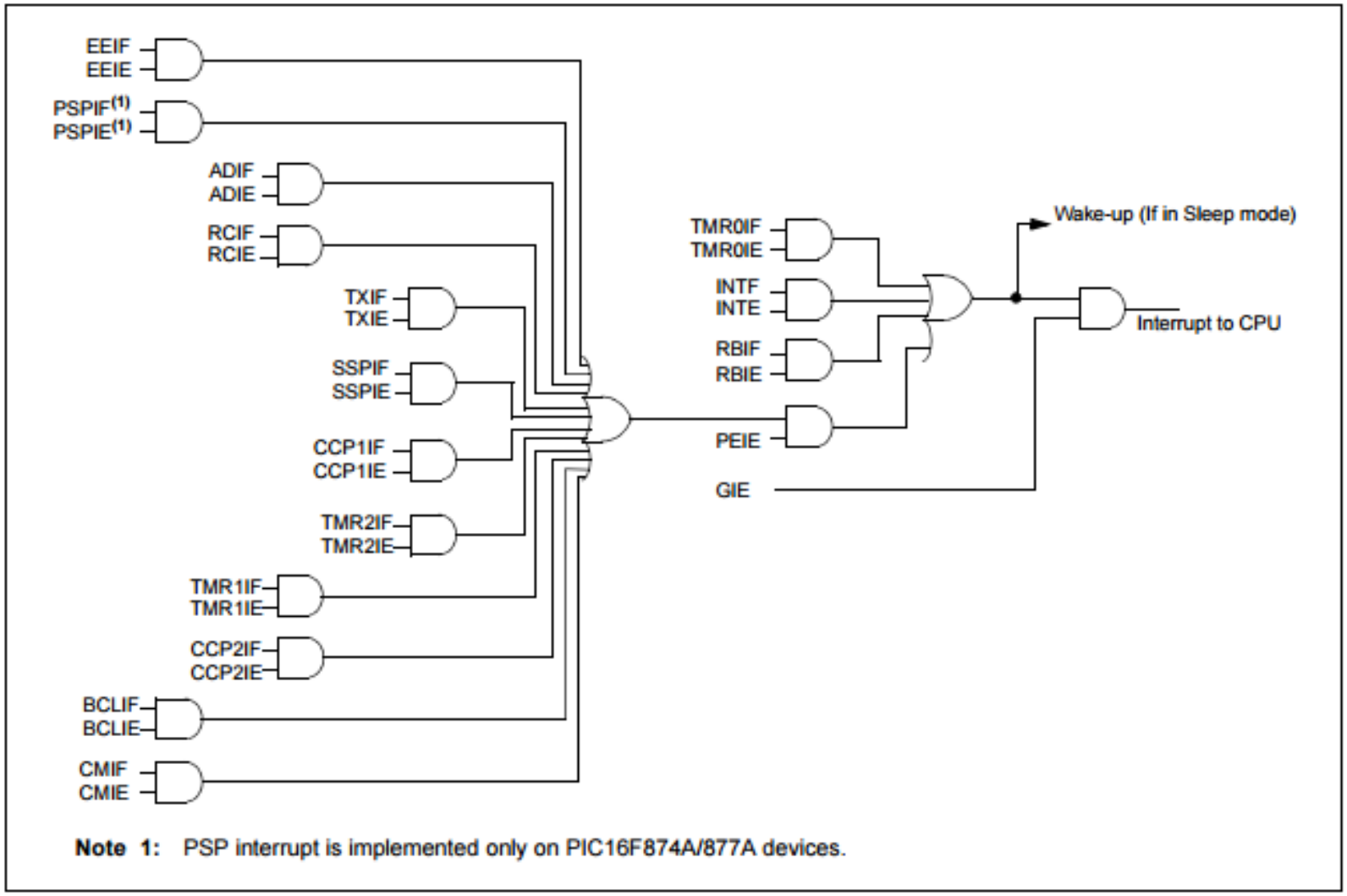


Interrupção

- Por Hardware
  - Periféricos podem gerar interrupções
    - Recebimento de dados;
    - Término de conversão AD;
    - Estouro de Timers;
  - Internamente, são implementadas utilizando sinais elétricos que são enviados ao processador;
  - Economiza tempo
- Por Software
  - Exceção (divisão por zero)
  - Contadores
  - Drivers

Interrupção

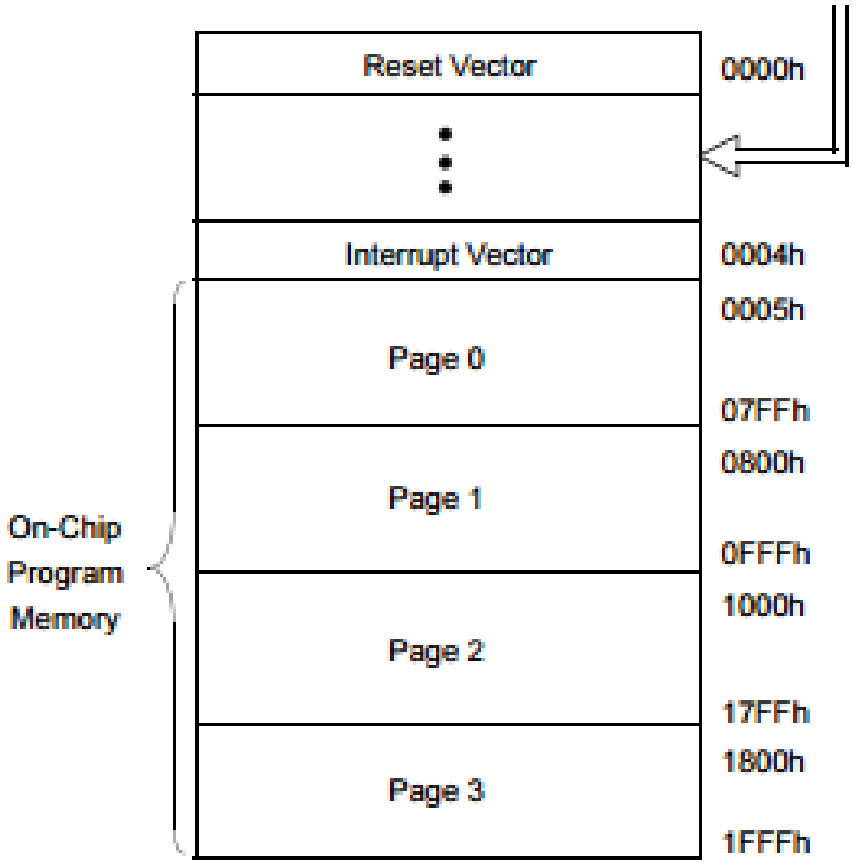
FIGURE 14-10: INTERRUPT LOGIC



Note 1: PSP interrupt is implemented only on PIC16F874A/877A devices.

Interrupção

- PIC



Interrupção

Table 12-6. Reset and Interrupt Vectors in ATmega328 and ATmega328P

VectorNo.	Program Address[2]	Source	Interrupt Definition
1	0x0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete

Processamento Concorrente

- As interrupções são utilizadas para executar código que tratam de eventos específicos;
- Sistemas embarcados geralmente precisam controlar vários processos;
- Porém, nos microcontroladores mais simples não há suporte a threads ou processamento paralelo;
- Em geral, processos precisam ser cooperativos;
  - Liberam o processador por vontade própria;

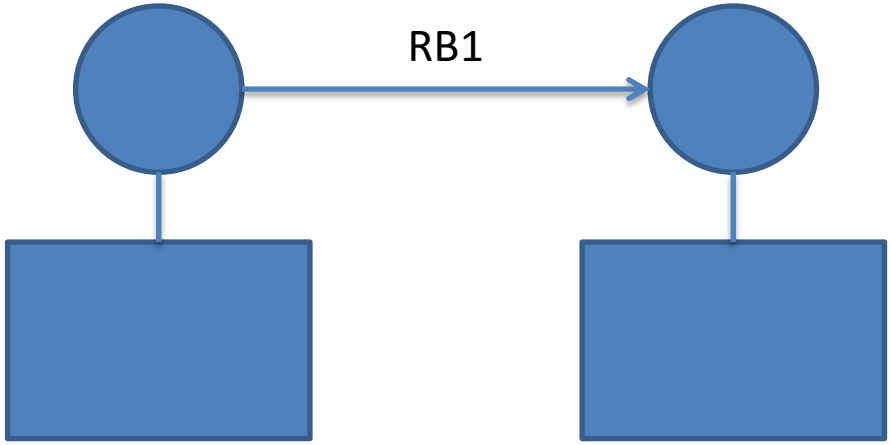
Processamento Concorrente

- Várias técnicas são possíveis;
- Em geral reflete na forma de programar;
- Utilização de estrutura de dados;
- Avaliar que tipo de processos a aplicação permite:
  - É melhor implementar um processo importante numa interrupção e os demais sem se preocupar?
  - Todos devem ser colaborativos?
  - Algum processo pode eventualmente ocupar de forma indefinida o processador?

Processamento Concorrente

- Para controle de um processo em passos pode-se utilizar autômatos

```
...
if(estado == 0){
    ...
    if(RB1) estado = 1;
}
if(estado == 1){
   ...
}
...
```



Processamento Concorrente

- Exemplo temporizador

```
void TON_init(st_TON *st){
        st->Q=0;
        st->ET=0;
}
void TON(st_TON *st){
        if(st->IN == 0){
                st->Q = 0;
                st->ET = 0;
        }else{
                st->ET++;
                if(st->ET >= st->PT){
                        st->Q = 1;
                        st->ET--;
                }
        }
}
```

```
st_TON var1;

...

var1.IN = (RB1);
var1.PT = (5000);
TON(&var1);
RB4 = var1.Q;

...
Delay_ms(1);
...
```

Exercício 1 de 2

- Implementar e simular:
  - Piscar 1 led em intervalos de 1 segundo;
    - 100ms aceso, 1 segundo apagado.
  - Piscar 1 led em intervalos de 3 segundos;
    - 100ms aceso, 3 segundos apagados
  - Acionar uma saída se uma entrada se manter acionada por 5 segundos. Se a entrada desativar, desativar também a saída;
  - Acionar e manter acionado S1 se E1 for acionado. Desativar S1 se E2 foi acionado;

Exercício 2 de 2

- Piscar 1 led em intervalos de 1 segundo (usar timer0);
  - 100ms aceso, 1 segundo apagado.
- Piscar 1 led em intervalos de 3 segundos (usar timer0);
  - 100ms aceso, 3 segundos apagados
- Acionar uma saída se uma entrada se manter acionada por 5 segundos. Se a entrada desativar, desativar também a saída;
- Acionar e manter acionado S1 se E1 for acionado por interrupção. Desativar S1 se E2 foi acionado;