Interrupções

o que são interrupções

- interrupção do programa em execução
  - síncronas
  - assíncronas
    - forma do hardware chamar atenção!

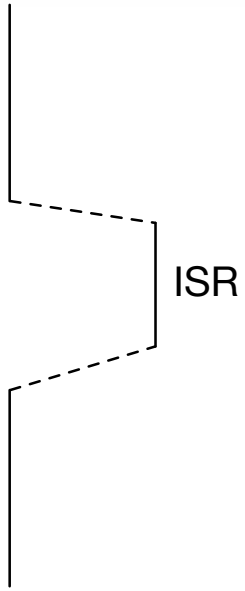
o que são interrupções

- interrupção do programa em execução
  - síncronas
  - assíncronas
    - forma do hardware chamar atenção!

similaridade com uma callback gerada pelo hardware!
→ mas programa em execução é interrompido!

tratamento de interrupções

- hardware desvia controle para tratador
  - configuração específica de cada CPU



interrupções - utilidade

- alternativa ao polling: avisos sobre operações completas
- tratamento de falhas (page faults, etc)
- preempção de processos e threads
- ...
- chamadas ao sistema operacional
- ...

similaridade com uma callback gerada pelo hardware!

interrupções - tipos

- transmissão/recebimento de dados
  - chegada de dados
  - erro na transmissão
  - ...
- erros de execução
  - segfault (acesso a endereço inválido)
  - código de máquina inválido
- syscall

interrupções - cuidados

- troca de contexto
  - registradores
  - pilha de execução
  - ...
- tratamento bloqueia outros acontecimentos
  - interrupção da interrupção?
  - tratamento tem que ser rápido

interrupções - avr

- suporte do compilador
  - registradores são automaticamente salvos e restaurados
- tratadores de novas interrupções executados apenas depois que tratador em execução terminar
  - em ordem de prioridades
- tratadores por conjuntos de pinos
  - `ISR (PCINT0_vect) pin change interrupt for D8 to D13`
  - `ISR (PCINT1_vect) pin change interrupt for A0 to A5`
  - `ISR (PCINT2_vect) pin change interrupt for D0 to D7`

```
 1  Reset
 2  External Interrupt Request 0  (pin D2)           (INT0_vect)
 3  External Interrupt Request 1  (pin D3)           (INT1_vect)
 4  Pin Change Interrupt Request 0 (pins D8 to D13)  (PCINT0_vect)
 5  Pin Change Interrupt Request 1 (pins A0 to A5)   (PCINT1_vect)
 6  Pin Change Interrupt Request 2 (pins D0 to D7)   (PCINT2_vect)
 7  Watchdog Time-out Interrupt                      (WDT_vect)
 8  Timer/Counter2 Compare Match A                   (TIMER2_COMPA_vect)
 9  Timer/Counter2 Compare Match B                   (TIMER2_COMPB_vect)
10  Timer/Counter2 Overflow                          (TIMER2_OVF_vect)
11  Timer/Counter1 Capture Event                     (TIMER1_CAPT_vect)
12  Timer/Counter1 Compare Match A                   (TIMER1_COMPA_vect)
13  Timer/Counter1 Compare Match B                   (TIMER1_COMPB_vect)
14  Timer/Counter1 Overflow                          (TIMER1_OVF_vect)
15  Timer/Counter0 Compare Match A                   (TIMER0_COMPA_vect)
16  Timer/Counter0 Compare Match B                   (TIMER0_COMPB_vect)
17  Timer/Counter0 Overflow                          (TIMER0_OVF_vect)
18  SPI Serial Transfer Complete                     (SPI_STC_vect)
19  USART Rx Complete                                (USART_RX_vect)
20  USART, Data Register Empty                       (USART_UDRE_vect)
21  USART, Tx Complete                               (USART_TX_vect)
22  ADC Conversion Complete                          (ADC_vect)
23  EEPROM Ready                                     (EE_READY_vect)
24  Analog Comparator                                (ANALOG_COMP_vect)
25  2-wire Serial Interface  (I2C)                   (TWI_vect)
26  Store Program Memory Ready                       (SPM_READY_vect)
```

interrupções – como tratar

```
#include "pindefs.h"

// globais

volatile int buttonChanged = 0;

void pciSetup(byte pin) {
    *digitalPinToPCMSK(pin) |= bit (digitalPinToPCMSKbit(pin));  // enable pin
    PCIFR  |= bit (digitalPinToPCICRbit(pin)); // clear any outstanding interrupt
    PCICR  |= bit (digitalPinToPCICRbit(pin)); // enable interrupt for the group
}

ISR (PCINT1_vect) { // handle pin change interrupt for A0 to A5 here
   buttonChanged=1;
 }

void setup() {

  pinMode(LED1, OUTPUT); pinMode(LED2, OUTPUT);
  digitalWrite(LED1,HIGH); digitalWrite(LED2,HIGH);
  pinMode (KEY1, INPUT_PULLUP);
  pinMode (KEY2, INPUT_PULLUP);
  pinMode(KEY3, INPUT_PULLUP);

  pciSetup(KEY1); pciSetup(KEY2); pciSetup(KEY3);
}


void loop() {
  if (buttonChanged) {
     digitalWrite(LED1,digitalRead(KEY1));
     digitalWrite(LED2,digitalRead(KEY2));
     buttonChanged = 0;
  }
}
```

interrupções – quando usar

- tempo de resposta
- compatibilidade com tarefas de usuário de longa duração

- dificuldades:
  - problemas de concorrência: não podemos usar locks comuns
  - tratamento tem que ser rápido

interrupções – outro uso

- dispositivos podem ser colocados em "modo sleep" para economizar energia
  - no arduíno, economia é um pouco artificial
  - conceito é importante em vários contextos

interrupções – outro uso

```
#include <avr/sleep.h>
#include <avr/power.h>
...
void enterSleep(void){
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_mode();     /* The program will continue from here. */
  /* First thing to do is disable sleep. */
  sleep_disable();
}
```

interrupções – exercícios

- Colocar chamada a enterSleep() dentro do loop() da aplicação mostrada, acordando por interrupções geradas pelas chaves
  - aplicação pronta: led iluminado quando chave apertada
  - led1 para chave 1, led 2 para chave 2

http://www.inf.puc-rio.br/~noemi/sr-19/code/arduino/interrupcaochaves.ino

(uso de Serial.print("!"); delay(10); no início de loop para comparar as versões com e sem sleep)

- Sem usar sleep: fazer programa que usa interrupção do timer para piscar e parar qdo 2 chaves apertadas com menos de 500ms de diferença (não precisa acelerar e desacelerar)

http://www.inf.puc-rio.br/~noemi/sr-19/code/arduino/timer.ino

interrupções – timer

timer | pindefs.h

```
#include "pindefs.h"
byte state = HIGH;
volatile int counter = 0;

void timerSetup () {
   TIMSK2 = (TIMSK2 & B11111110) | 0x01;
   TCCR2B = (TCCR2B & B11111000) | 0x07;
}

void setup() {
   pinMode(LED1, OUTPUT); digitalWrite(LED1, state);
   pinMode(LED2, OUTPUT); digitalWrite(LED2, state);
   pinMode(LED3, OUTPUT); digitalWrite(LED3, state);
   pinMode(LED4, OUTPUT); digitalWrite(LED4, state);
   timerSetup();
}

void loop() {
  if (counter==50) {
    state = !state;
    digitalWrite(LED1, state);
    counter = 0;
  }
}
```

PUC RIO