Arquitectura de Computadores 2010/2011

Aula prática 11– Interrupções e dispositivos de I/O

Pretende-se realizar programas que enviem caracteres através da porta série da porta série do PC utilizando interrupções. Será utilizado o TurboC a correr no sistema operativo FreeDOS, emulado pelo software de virtualização *qemu*. Consulte o guia da aula prática 10 sobre como usar o *qemu*, como ter acesso às portas de entrada / saída e sobre como transferir ficheiros da máquina virtual para o computador pessoal

Na directoria **`c:\work\ea`** encontra um conjunto de ficheiros, já utilizados na aula anterior, e que o ajudarão a realizar o trabalho.

Consulte o ficheiro **serie.pdf**, na secção Problemas do CLIP, para saber os detalhes da programação da porta série do PC. Recordam-se aqui os endereços de IO do controlador da porta série COM1: ; o circuito integrado que é a principal parte do controlador é conhecido por UART (Universal Asynchronous Receiver/transmiter)

- Registo de Dados de Escrita (THR): 0x3F8
- Registo de Dados de Leitura (RBR): 0x3F8
- Registo de Controlo das Interrupções (IER): 0x3F9
- Registo de Controlo do Modem (MCR): 0x3FC
- Registo de Estado (LSR): 0x3FD

Interrupções e Turbo C

No TurboC é possível programar o sistema de interrupções, incluindo fazer o ficheiro de cabeçalho **dos.h**

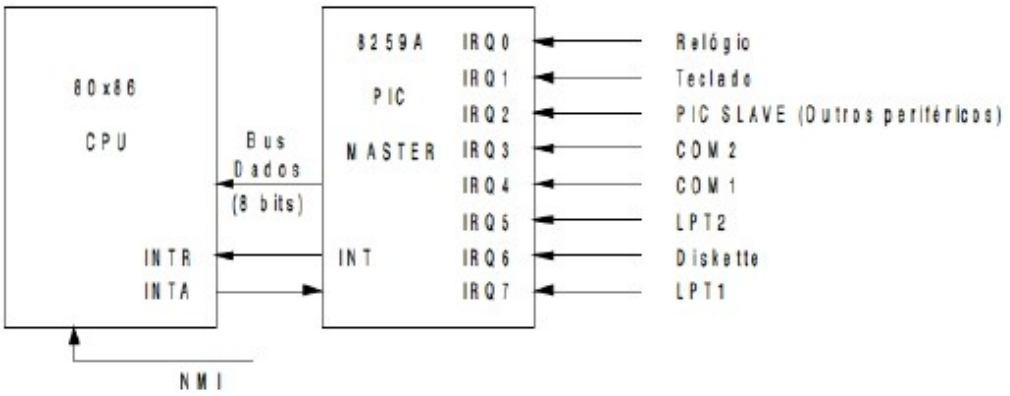
- **`enable()`** - função do TurboC para ligar as interrupções no CPU (coloca interrupt flag a 1);
- **`disable()`** - função do TurboC para desligar as interrupções no CPU (coloca interrupt flag a 0);
- Para colocar na mesma entrada o endereço da uma rotina de tratamento de interrupções **`setvect()`**

```
#include <dos.h>
…
void interrupt handlingRoutine(){
      …
}
…
void some_function() {
    …
    setvect(12, handlingRoutine);
    …
}
```

Arquitectura de Computadores 2010/2011

Aula prática 11– Interrupções e dispositivos de I/O

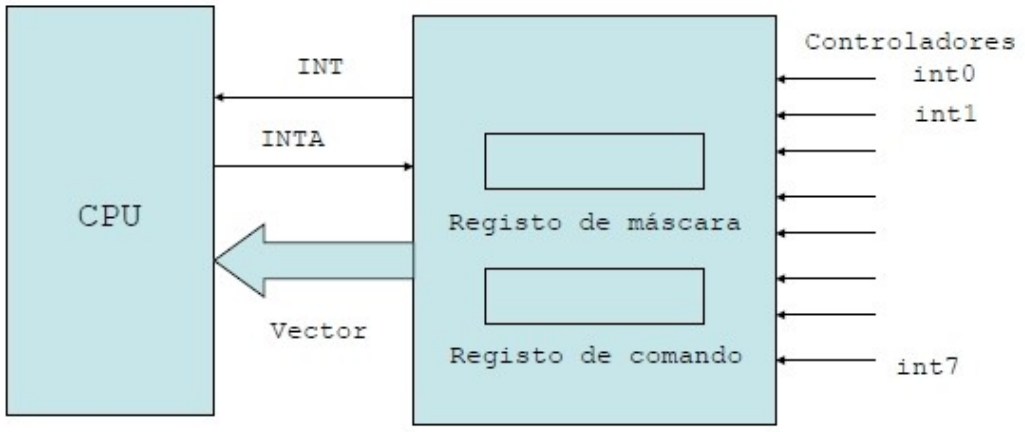
PIC



O PIC tem dois registos

- **Registo de comando** (endereço 0x20): é para este endereço que é enviado o comando EOI (End Of Interrupt), que corresponde a *out 0x20, 0x20*

- **Registo de mascara** (endereço 0x21): o bit i a 1 indica que o nível i está desligado; o bit i a 0 indica que a entrada i está desinibida



Registo de máscara: a 0 indica que a linha i está activa
Registo de comando: permite programar a forma de atendimento
End of Interrupt (EOI) informa o PIC que a rotina
de atendimento acabou

A manipulação do registo de máscara do PIC deve fazer-se com as interrupções desligadas. No nosso caso, deve apenas alterar o bit 4 (COM1:) deixando todos os outros bits inalterados. Apresentam-se a seguir dois exemplos em assembly e em TurboC.

- Desligar o nível 3

```
Cli                    ;disable();
```

```
in al, 0x21
or al, 0000 1000B
out 0x21,al        ;outport( 0x21, inport(0x21)|0x08)
sti                ;enable();
```

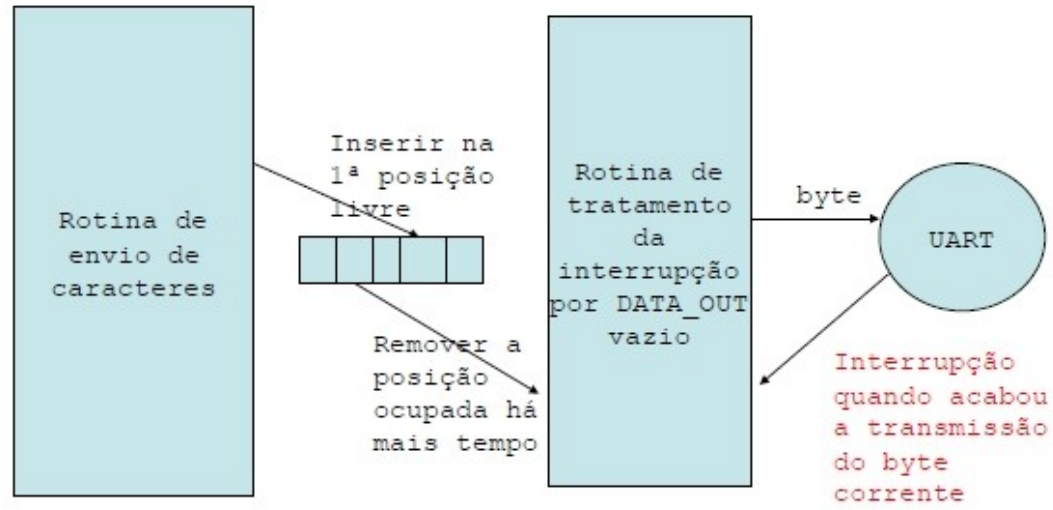
- Ligar o nível 2

```
Cli                ;disable()
in al, 0x21
and al, 1111 1011B
out 0x21,al        ;outport( 0x21, inport(0x21)& ~0x04)
sti                ;enable();
```

Consulte o documento *interrupcoes.pdf*, na secção Textos de Apoio do CLIP, para uma explicação detalhada do funcionamento das interrupções no PC, incluindo a UART (Controlador da Porta série) e o PIC (controlador de interrupções).

Trabalho a realizar:

Objectivo: Enviar um ficheiro para a porta série utilizando o mecanismo de interrupções dado nas aulas teóricas. Terá de implementar a função ***enviar_serie*** para realizar o envio de um carácter para a porta série. A figura seguinte resume a organização da solução a implementar

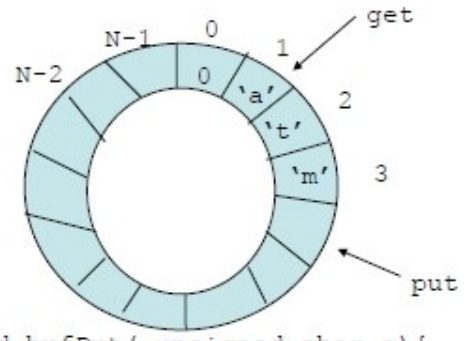


A função *enviar_serie()* deverá colocar os caracteres num *buffer* circular. Ao mesmo tempo a rotina de tratamento de interrupções vai retirar os caracteres desse buffer e enviá-los para a porta série. A rotina de tratamento de interrupções será chamada através mecanismo hardware das interrupções, sempre que as interrupções da porta série estiverem ligadas e o registo de transmissão da UART estiver livre.

O código que manipula o buffer circular está nos ficheiros **`bufcir.h`** e **`bufcir.c`** que estão disponíveis na secção “Problemas” do CLIP

A figura seguinte resume o funcionamento do *buffer* circular:

```
#define BUFSIZ 128

unsigned char buf[BUFSIZ];

int put;  // 1ª casa livre

int get;  // casa ocupada há mais
          // tempo

int nc;   // nº de bytes no buffer


unsigned char bufGet(void){
/* assume que buf não está
vazio */
  unsigned char x = buf[get];
  get = (get + 1) % BUFSIZ;
  nc --;
  return x;
}
int bufEmpty(){
  return (nc == 0);
}
```




```
void bufPut( unsigned char c){
/* assume que buf não está cheio */
  buf[put] = c;
  put = (put + 1) % BUFSIZ;
  nc ++;
}
int bufFull(){
  return (nc == BUFSIZ);
}
```

Supondo disponíveis as funções de gestão do *buffer* circular acima referidas, podemos esboçar as acções das rotina que envia caracteres e a rotina de tratamento de interrupções.

```
Enviar_serie(ch){
   if bufFull()
            esperar;
   if bufEmpty(){
       bufPut(ch)
       ligar interrupções TXEMPTY
   } else
       bufPut(ch)
}
```

```
Rotina de tratamento de
interrupções TX_EMPTY{
   ch = bufGet();
   outportb( DATA_OUT, ch );
   if bufEmpty()
      desligar as
      interrupções TXEMPTY

   outportb( 8259_COMMAND,
EOI)
}
```

Alguns pontos importantes no esboço apresentado são:

- a rotina **enviar_série** espera em ciclo que haja espaço no “buffer”
- A rotina de interrupções desliga as interrupções da UART se o “buffer” fica vazio
- A rotina **enviar_série** verifica se o “buffer” está vazio; se tal acontecer deposita o carácter no “buffer” e liga as interrupções na UART

Sugere-se que antes das rotinas *enviar_serie()* e de *tratamento de interrupções*, escreva o código das seguintes subrotinas que pode acrescentar ao módulo io.c usado na aula anterior

- *tx_int_on()*que permite ligar as interrupções da porta série e ligar e o atendimento de interrupções da linha IRQ4 do PIC.

- Na UART é necessário:
  - colocar o bit 3 do registo MCR – Model Control Register (porto 3FCH) a 1, para permitir as interrupções da UART;
  - colocar o bit 1 do registo IER – Interrupt Enable Register (porto 3F9H) a 1, para permitir gerar interrupções após o envio de um caracter.
- No PIC é necessário colocar o bit 4 do registo máscara (porto 21H) a 0, de modo a deixar passar as interrupções que vêm da porta série. A alteração do registo máscara do PIC deve ser efectuada com as interrupções do CPU desligadas (i.e. com a *interrupt flag* a 0).

- *tx_int_off()* que permite desligar as interrupções da porta série e desligar o atendimento de interrupções da linha IRQ4 do PIC.
  - Na UART é necessário colocar o bit 3 do registo MCR a 0 e o bit 1 do registo IER a 0.
  - No PIC é necessário colocar o bit 4 do registo máscara (21H) a 1.
- *init_tx_serie()* que deve
  - guardar o conteúdo da entrada do vector de interrupções (12)
  - instalar nessa entrada o endereço da função que é o nova ISR (Interrupt Service Routine) da porta série COM1:
- *reset_tx_serie()* restaurar a entrada salvaguarda em *init_tx_serie()*

Quando escrever o código das das rotinas *enviar_serie()* e de *tratamento de interrupções*, deve ter em atenção os seguintes pontos:

- como o buffer pode ser acedido em qualquer altura pela rotina de tratamento de interrupções, o programa principal deve acedê-lo com as interrupções do CPU desligadas (com a *interrupt flag* a 0).
- quando a rotina de tratamento de interrupções é chamada, as interrupções são desligadas. Devemos ligá-las o mais cedo possível, pois podem acontecer interrupções mais importantes (de nível mais elevado) que seja importante tratar imediatamente.