Arquitectura de Computadores 2009/2010

Aula práticas 11 e 12 – Programação de dispositivos de I/O

Pretende-se realizar programas que enviem caracteres através da porta série da porta série do PC. Será utilizado o TurboC a correr no sistema operativo FreeDOS, emulado pelo software de virtualização *qemu*. O ficheiro comprimido com a imagem da máquina virtual a utilizar encontra-se na secção Documentação de Apoio -> Outros no CLIP.

Simulação da porta série com o qemu

Para a emulação da porta série proceda aos seguintes passos:

i. Simular a porta série, executando o comando na forma:

```
qemu -serial stdio fdos-10meg.img
```

O simulador irá fazer surgir no terminal tudo o que os seus programas internamente enviarem para a porta série. Tudo o que for escrito pelo utilizador no terminal, vai surgir como estando a entrar pela porta série da máquina virtual.

ii. Configurar a porta série no FreeDOS:

```
mode com1:9600,n,8,1
```

O comando **mode** do DOS configura os parâmetros de funcionamento da porta série especificada, neste caso o **com1** (velocidade, paridade, tamanho da palavra, número de stop bits).

Rotinas de acesso aos portos I/O a partir do TurboC

Na directoria **c:\work\ea** encontra-se o ficheiro cabeçalho ***inout.h*** e o ficheiro assembly **inout.asm** que, respectivamente, definem os protótipos e forneçem a implementação de duas funções (**inByte** e **outByte**) que permitem aceder, a partir do TurboC, às instruções assembly **IN** e **OUT**.

Este código assembly utiliza a sintaxe do assembler TurboASM (tasm) e apenas instruções de 16 bits. Pode chamar estas funções do seu programa em C, de acordo com a seguinte interface:

```
/*le um byte no porto de I/O com endereço port*/
extern unsigned char inByte (unsigned int port);
/*escreve um byte no porto de I/O com endereço port */
extern void  outByte (unsigned int port, unsigned char byte);
```

Na mesma directoria pode encontrar a versão "assemblada" do ficheiro, **inout.obj**, que pode incluir nos projectos do TurboC.

Arquitectura de Computadores 2009/2010

Aula práticas 11 e 12 – Programação de dispositivos de I/O

Fase 1 - Espera Activa

Objectivo: Enviar um ficheiro para a porta série utilizando o mecanismo de espera activa.

A directoria **`c:\work\ea`** contem ainda a implementação parcial de um programa que apresenta uma linha de comando (**acio**) que permite três comandos:

- **`print "nome_do_ficheiro"`** envia o ficheiro com o nome dado para a porta série
- **`help`** mostra os comandos disponíveis
- **`exit`** termina o programa

O programa recorre aos seguintes ficheiros fonte:

- **`main.c`** programa principal e implementação dos comandos **`print`** e **`help`**
- **`comline.h`** e **`comline.c`** reconhecimento da linha de comando
- **`inout.h`** e **`inout.asm`** interacção com dispositivos de I/O acesso às instruções **`IN`** e **`OUT`**
- **`io.h`** e **`io.c`** envio de um carácter para a porta série utilizando espera activa
- **`ea.prj`** ficheiro projecto do TurboC que define que ficheiros compõem a aplicação, ou seja, que ficheiros devem dados ao *linker* para este produza o ficheiro executável final

Todos os ficheiros estão totalmente implementados à excepção do ficheiro **`io.c`**, nomeadamente a função

```
void send_serial(unsigned char b);
```

que envia o carácter recebido como argumento para a porta série.

O seu trabalho nesta fase será implementar a função **`send_serial`** utilizando o mecanismo de espera activa dado nas aulas teóricas.

Compile o programa e teste-o, utilizando o comando ***`print`*** para enviar ficheiros para a porta série.

Consulte o ficheiro serie.pdf, na secção *Documentação de Apoio → Textos de Apoio* do CLIP, para saber os detalhes da programação da porta série do PC.

Arquitectura de Computadores 2009/2010

Aula práticas 11 e 12 – Programação de dispositivos de I/O

Fase 2 - Interrupções

Objectivo: Enviar um ficheiro para a porta série utilizando o mecanismo de interrupções dado nas aulas teóricas.

Nesta fase deve implementar o mesmo programa da fase anterior, mas recorrendo ao mecanismo de interrupções para controlar o fluxo de dados enviado para a porta série.

A directoria **`c:\work\int`** contem a mesma implementação dada como base para a fase anterior, acrescida dos ficheiros:

- **`bufcir.h`** e **`bufcir.c`** — implementação de um buffer circular, como dado nas aulas teóricas

Mais uma vez terá de implementar a função ***`send_serial`*** para realizar o envio de um carácter para a porta série. A função deverá colocar os caracteres no 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.

Sugestão: adicione duas funções ao módulo ***`io`***

- uma para a configuração do sistema de forma a utilizar o mescanismo de interrupções no controlo do fluxo de dados para a porta série;
- uma segunda para a reposição do estado incial, aquando da terminação do programa.

Nota 1: 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).

Nota 2: 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.

Consulte o documento interrupcoes.pdf na secção *Documentação de Apoio → 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).

Este documento também contém informação sobre as funções disponíveis no TurboC necessárias para este trabalho.

Arquitectura de Computadores 2009/2010

Aula práticas 11 e 12 – Programação de dispositivos de I/O

Anexo A - Rotinas de Apoio

Sugerimos que implemente funções que permitam ligar e desligar as interrupções da porta série e ligar e desligar o atendimento de interrupções da linha IRQ4 do PIC.

1. Ligar interrupções da porta série

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.

  Nota: a alteração do registo máscara do PIC deve ser efectuada com as interrupções do CPU desligadas (i.e. com a *interrupt flag* a 0).

2. Desligar interrupções da porta série

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.

Arquitectura de Computadores 2009/2010

Aula práticas 11 e 12 – Programação de dispositivos de I/O

Anexo B - Programação de entradas/saídas em TurboC

Endereços de I/O:

- UART (COM1):
  - 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
- PIC:
  - Registo de Comandos: 0x20
  - Registo de Máscara: 0x21

Comandos do PIC:

- Fim de Interrupção (EOI): 0x20

Programação de interrupções no TurboC:

- Deve fazer o include do 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);
    …
}
```