NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Arquitetura de Computadores 2022/23

Ficha 9

Tópicos: *Manipulação direta dos dispositivos de Entrada/Saída (IN/OUT) usando espera ativa. Exemplo da porta série. Uso de máquina virtual qemu e MS-DOS.*

Introdução

Em computadores com uma arquitetura compatível com a família dos processadores *Intel 80x86,* o acesso às interfaces dos periféricos existentes é realizado através do *mapa de Input/Output* (*I/O* ou seja, *entrada/saída*). Este mapa pode corresponder a um espaço de endereçamento separado da memória central, no qual cada periférico ocupa um conjunto de endereços (portos ou portas). Estes conjuntos de endereços (portos) estão mapeados em registos internos do *hardware* do dispositivo periférico e permitem transferir dados, dar comandos e consultar o estado desse dispositivo.

A família de processadores do *Intel 80x86* dispõe de instruções específicas para aceder aos portos e, portanto, aos registos dos periféricos. Estas operações são chamadas de *Input/Output* e designam-se, em *assembly*, pelas mnemónicas **`in`** e **`out`**, respectivamente. Nos processadores que suportam modos de execução utilizador e supervisor, estas operações estão reservadas apenas ao modo supervisor usado pelo SO, pelo que não podem ser usadas pelas aplicações normais.

Para se poderem usar essas operações, vamos recorrer a um SO antigo para processadores que não dispõem desses modos, como seja o Intel 8086. Podemos então construir programas normais que acedem diretamente a todo o hardware, sem qualquer restrição do CPU ou do SO. No caso deste trabalho, será usada um SO que é uma versão cópia do MS-DOS, o Free-DOS. Para a sua execução basta usar uma máquina virtual simples como o QEMU, que é a opção recomendada. Esta máquina virtual é capaz de emular o processador Intel em modo real como o original 8086 e restante *hardware* de uma arquitetura de tipo IBM/PC. Tal inclui a interface série e a simulação das respetivas entradas e saídas de dados. Mais informação é fornecida mais à frente, incluindo sobre o compilador de C já instalado na imagem Free-DOS que é fornecida.

As portas série RS-232 são uma interface para troca de dados em modo série (transmitidos como uma série de bits, um de cada vez), que era norma e estava presente em todos os computadores PC e compatíveis até há alguns anos. Atualmente está em claro desuso nos computadores pessoais (em favor das interfaces USB) mas é ainda usada por alguns dispositivos e sistemas de controlo. Neste trabalho esta interface vai servir de pretexto para manipular diretamente um dispositivo de entrada/saída.

Trabalho a realizar

Execute os dois objetivos seguintes, pela ordem indicada. Todo o programa deve ser implementado no sistema DOS em C (versão TurboC). A informação necessária para a sua realização segue em anexo, incluindo:

- como usar o qemu para emular um computador a executar o sistema Free-DOS;
- como verificar as entradas e saídas de bytes pela porta série virtual desse computador.

Campus'de'Caparica'
2829O516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'

1. Enviar bytes pela porta série COM1.

Deve programar a seguinte função:

```
void sendSerial( unsigned char byte );
```

A função deve utilizar o método de espera ativa para aguardar que seja possível enviar um byte através da porta série (COM1). Quando tal for possível, deve enviar o byte indicado no parâmetro.

O programa principal deve recorrer a esta função para enviar uma sequência de alguns caracteres. Para testar experimente enviar os caracteres de uma *string* à sua escolha. A saída pela porta série deste computador virtual deve aparecer no terminal de onde lançou o qemu. **Veja o código já fornecido na diretoria `work`**.

Complete o programa para que receba na linha de comando o nome de um ficheiro de texto e que envie todos os seus caracteres pela porta série, um de cada vez, recorrendo à função anterior. Veja os manuais das funções `fopen`, `fgetc/getchar`, `fclose` do *`stdio.h`*

2. Receber bytes pela porta série COM1

Desenvolva agora um novo programa, começando pela seguinte função:

```
unsigned char receiveSerial( void );
```

Esta função deve utilizar o método de espera ativa para aguardar a chegada de um byte através da porta série (COM1). Quando tal acontece, deve receber o byte e devolvê-lo como retorno da função.

O programa principal deve recorrer a esta função para ficar em ciclo, a receber uma sequência de alguns caracteres, afixando-os no ecrã (*stdout*). A simulação da entrada de bytes pela porta série será conseguida escrevendo caracteres no terminal Linux de onde lançou o qemu. O programa termina quando se receber a letra 'q'.

ANEXO

O Qemu, o FreeDOS e o TurboC

Como sabe, as instruções máquina de entrada/saída são privilegiadas e, portanto, apenas acessíveis em modo supervisor. Para contornar este problema que inevitavelmente surgiria se o desenvolvimento do trabalho fosse realizado sobre o sistema Linux, vamos executar os nossos programas sobre um simulador (máquina virtual) chamado **qemu**[1] e com um SO semelhante ao MS-DOS. Uma imagem de um disco com todos os ficheiros e software necessário é fornecido com este enunciado. Esta máquina virtual simula fielmente a arquitetura de um PC com um processador Intel e todo o *hardware* que o rodeia - portas série e paralela, controlador de interrupções, etc. Esta simulação é, em muitos casos, feita apenas por *software* pelo que não é de esperar um grande desempenho.

O sistema de operação **Free-DOS** que, como o nome indica, é uma re-implementação do **MS-DOS**, será o sistema operativo onde realizaremos o nosso trabalho. Estes sistemas de operação foram desenvolvidos para processadores 8086 (de 16 bits), sendo que os atuais processadores i386 e x86-64 continuam compatíveis com aquele processador. O Free-DOS não usa o modo privilegiado dos processadores modernos (pois não existia no processador 8086) e deixa que qualquer programa tenha total controlo da máquina, i.e., que execute qualquer instrução.

[1] Se não tem este comando instale-o. Por exemplo: sudo `apt install qemu-system-x386`

Campus'de'Caparica'
2829Ø516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'

TurboC

Como compilador de C, vamos usar a versão de TurboC já instalado na imagem fornecida. Lembre-se que está numa arquitetura Intel de 16 bits, produzindo código desse tipo, mesmo que execute a máquina virtual numa máquina de 32 ou 64 bits. O TurboC inclui, para além do compilador, um editor que funciona como um IDE em modo de texto, permitindo, para além de editar o seu código, compilar, executar, depurar e obter ajuda sobre a linguagem e suas bibliotecas.

Este ambiente disponibiliza nas suas bibliotecas as seguintes funções para aceder, a partir do C, aos portos de I/O utilizando as instruções máquina **`in`** e **`out`**, dispensando o uso de *assembly* no seu programa. Estas estão disponíveis através do ficheiro de cabeçalho **<dos.h>**.

```
/*le um byte do porto de I/O com endereço port*/
unsigned char inportb( int port );

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

Teste do qemu com o "disco" Free-DOS

Existe uma imagem comprimida de um disco do **Free-DOS** online. Descarregue a imagem e descompacte-a com o comando:

```
unzip fdos.img.zip
```

Agora já pode invocar o emulador, indicando como disco de *boot* o ficheiro com a respetiva imagem:

```
qemu fdos.img
```
*(em vez de qemu o comando pode ser **qemu-system-i386**)*

Execute alguns comandos do DOS (cd, dir, etc). Experimente também invocar o editor do TurboC: **`tc`**. Para terminar a simulação feche a janela do simulador (o que corresponderia a desligar o PC).

Para emular a porta série, simulando a sua ligação ao terminal onde lançou o qemu, execute o seguinte comando:

```
qemu-system-i386 -serial stdio -drive format=raw,file=fdos.img
```

Com este comando, o emulador irá **imprimir no terminal** tudo o que os seus programas internamente enviarem para a porta série. Reciprocamente, tudo o que for escrito pelo utilizador nesse terminal irá ser enviado para a porta série da máquina virtual e será recebido pelo programa que, dentro do Free-DOS, leia a porta série.

Nota:

Se achar necessário trocar ficheiros entre a imagem Free-Dos e o Linux, sem executar o qemu, efetue o "mount" do disco a partir do explorador de ficheiros do seu ambiente gráfico Linux[2], ou use os comandos **mtools** (veja o respectivo manual). **Nunca** execute a imagem no qemu quando esta está montada no seu sistema de ficheiros, para uso a partir do Linux (e *vice-versa*). Lembre-se também

[2] Se no browser de ficheiros não poder fazer o mount da imagem com o botão direito, execute:
`sudo apt install gnome-disk-utility`

Campus'de'Caparica'
2829C516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'

que a convenção para representar uma mudança de linha em ficheiros de texto é diferente no Linux/Unix e no DOS. Se necessário instale o comando **dos2unix**.

Porta série dos IBM/PC e compatíveis

Na comunicação série os bits são transmitidos sequencialmente (i.e., uma-a-um), através de uma linha de dados. A ligação pode ser SIMPLEX (apenas num sentido), HALF-DUPLEX (em ambos os sentidos mas de forma alternada) ou FULL-DUPLEX (em ambos os sentidos em simultâneo).

Os vários bits (símbolos 0 e 1) são convertidos em sinais elétricos e enviados de acordo com norma RS-232C. Esta norma define como efetuar este tipo de ligações entre dois equipamentos, por exemplo, como ligar dois computadores entre si, ou como ligar um computador a um MODEM (MOdulator/DEModulator) ou a qualquer outro periférico (como por exemplo outro computador, uma impressora, um rato, etc).

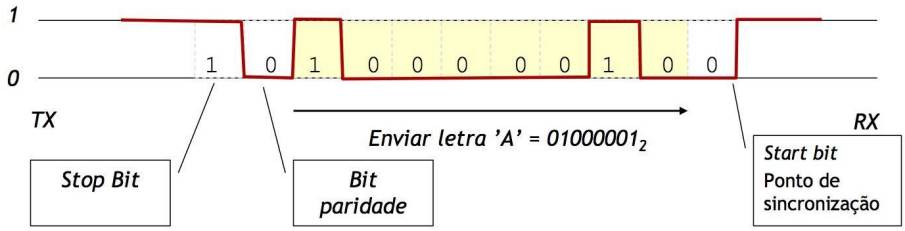
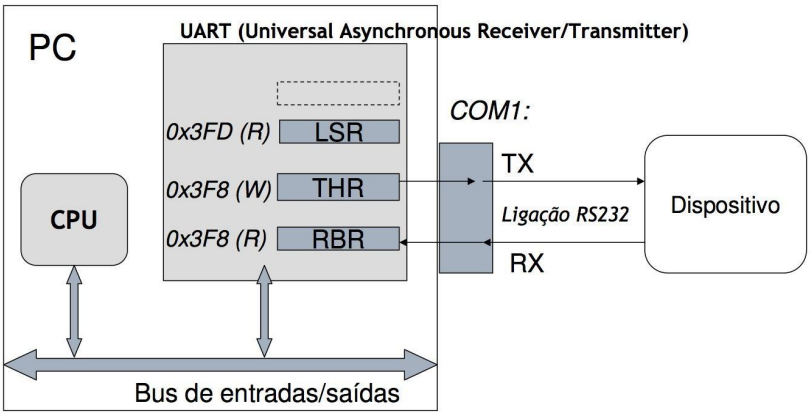


Figura 2: Exemplo de transmissão de $10000010_{(2)}$

A transmissão dos dados pode ser em unidades entre 5 a 8 bits, sendo cada unidade precedida por um *start* bit. Após os bits de dados são também enviados um *bit de controlo de paridade* (opcional) e um ou mais *stop bits* (ver figura 2). Antes da troca de dados, ambos os extremos da linha série têm de ser configurados para a mesma forma de comunicação e velocidade de transferência, para que se possam entender. A velocidade de transmissão é dada em bits/s (bits por segundo). Note que a taxa útil de transferência de informação é sempre menor que a velocidade máxima, mesmo que não ocorram erros, visto que é necessário enviar os bits *start*, *stop* e opcionalmente o de paridade.

Controlador da porta série: a UART

Cada porta série nos PCs é **controlada** por uma **UART** (Universal Asynchronous Receiver/Transmiter) integrado *NS 8250* ou compatível (por exemplo: 16450, 16550, 16C552). A porta série é operada através de um conjunto de **registos** (ver figura 3), acessíveis por ***portos de I/O*** colocados a partir do endereço inicial definido pelo respectivo descodificador de endereços. Geralmente, o **endereço inicial** da **primeira porta série** é **3F8H** e o da **segunda** é **2F8H**. Os nomes atribuídos às portas série pelo sistema de operação MS-DOS e MS-Windows são, *COM1*: e *COM2:*.



Campus'de'Caparica'
2829Ø516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'

Figura 3: Interação entre o CPU, a UART e a linha série

Acesso aos registos da COM*n*

Na tabela seguinte são apresentados alguns dos registos existentes e indicada a sua posição relativamente ao endereço inicial. Por exemplo: na ***primeira porta série*** (*COM1)*, o registo THR encontra-se no endereço **3F8H** (3F8+0), o registo LSR em 3FDH (3F8+5), etc; na ***segunda porta série*** (*COM2*), o respetivo registo LSR já se encontra no endereço **2FDH** (2F8+5).

Registo	**Abrev.**	**Acesso**	**Posição**
Dados – Transmissão (TX)	THR	Escrita	0
Dados – Recepção (RX)	RBR	Leitura	0
Estado da Linha	LSR	Leitura	5

De notar que os registos **RBR** e **THR** partilham o mesmo endereço de I/O — o primeiro é acedido através da instrução ***in*** e o segundo da instrução ***out***. Note que, quando o software envia um byte para registo de dados de transmissão (**THR**), o controlador da porta automaticamente trata do seu envio. Tal consiste em passar o byte, logo que possível, para o *shift register* de envio (chamado TSR) e depois enviando cada bit pela linha série ao ritmo correto.

O registo de estado, **LSR**, dá informação sobre a recepção e transmissão dos dados. Cada um dos seus bits fica a 1 caso a condição enunciada seja verdadeira e a 0 se falsa:

bit 0	Chegou um caráter que está disponível no registo RBR
bits 1-4	Erros na recepção. Respectivamente: sobreposição, erro de paridade, erro de transferência, break.
bit 5	O registo de transmissão (THR) está livre. (Podemos pedir o envio de um novo byte)
bit 6	Acabou de ser realmente enviado o último bit pela porta série (TSR vazio)

Quando o bit 0 do registo de estado é colocado a 1, podemos ler um byte do registo de dados de recepção (**RBR**). O bit 0 do registo de estado passará a zero assim que essa leitura for efetuada. Um programa só deve enviar um byte para o registo de envido (THR), quando o bit 5 está a 1.

Se necessário poderá usar o comando **MODE** do MS-DOS para alterar os parâmetros de funcionamento da porta série pretendida. Resumidamente:

```
MODE COMn: baud_rate[[[, paridade], bits_dados], stop_bits]
```

Em que:

n	número da porta série (de 1 a 4)
baud_rate	velocidade de transmissão (ex: 150, 300, 600, 1200, 2400, 4800, 9600, ...)
paridade	n = sem paridade; o = paridade ímpar; e = paridade par
bits_dados	dimensão dos dados: de 5 a 8 bits

Campus'de'Caparica'
2829O516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'

stop_bits	1 ou 2 bits

Existem outros registos que não são relevantes para este trabalho.

Campus'de'Caparica'
28290516'CAPARICA'
Tel:'+351'212'948'536'
Fax:'+351'212'948'541'
di.secretariado@fct.unl.pt'
'
www.fct.unl.pt'