

Arquitectura de Computadores
Licenciatura em Engenharia Informática
Teste 3 (A) – 2009/06/06 – Duração: 1h30m

Nome: _____ Número: _____

Teste sem consulta.

A interpretação do enunciado faz parte da avaliação. Explícite nas suas respostas todas as hipóteses assumidas.

Por favor, tente focalizar a sua respostas para que estas se enquadrem na zona delimitada.

1 Teórica - Escolha Múltipla

Indique o número da resposta que achar correcta no quadrado definido para o efeito. Cada resposta errada desconta 25% da cotação da pergunta.

Q-1 [1.0 val.] Numa arquitectura com memory-mapped I/O:

1. Existe um único espaço de endereçamento que engloba a memória e os controladores de periféricos
2. É necessário usar instruções dedicadas (por exemplo **in** e **out**) para interagir com controladores de periféricos
3. Não há interrupções
4. Existem um espaço de endereçamento para cada controlador de periférico

Opção correcta ☐

Q-2 [1.0 val.] A programação de entradas/saídas por interrupções:

1. É menos eficiente do que a espera activa porque usa buffers
2. Consiste na transmissão de blocos de bytes de/para memória sem intervenção do CPU
3. É mais eficiente do que a espera activa porque é executada em modo supervisor
4. Gere pedidos simultâneos a múltiplos periféricos de forma mais eficaz do que a espera activa

Opção correcta ☐

Q-3 [1.0 val.] No contexto da programação de entradas/saídas por interrupções seja R uma subrotina de atendimento de uma interrupção com prioridade i . Caso seja recebida uma interrupção de nível j :

1. A rotina R é sempre interrompida para atender a nova interrupção
2. Se $j < i$ então a rotina R é sempre interrompida para atender a nova interrupção
3. A rotina R só pode ser interrompida se tiver ligado explicitamente as interrupções
4. A rotina R nunca é interrompida para atender a nova interrupção

Opção correcta ☐

Q-4 [1.0 val.] Comparando os volumes de discos RAID 1 e RAID 5.

1. Uma operação de escrita no RAID 5 é mais rápida no que no RAID 1
2. O espaço gasto em redundância no RAID 5 é superior ao gasto no RAID 1
3. Ao contrário do RAID 1, operações de leitura no RAID 5 tornam-se mais rápidas com o aumento do número de discos
4. O RAID 5 tem suporte para tolerância a falhas e o RAID 1 não

Opção correcta ☐

Q-5 [1.0 val.] Nos processadores multi-core:

1. O motor de execução (ALUs, FPU, etc) é partilhado por todos os cores
2. Cada core tem a sua própria memória central
3. Cada core tem o seu próprio conjunto de registos de uso geral (no IA-32: EAX, EBX, ...)
4. O processador no seu todo segue uma arquitectura SISD (Single Instruction Single Data)

Opção correcta ☐

2 Prática

IMPORTANTE: Em todas as perguntas considere que está a usar o ambiente de programação usado nas práticas, isto é:

- PC x86 ou simulador
- Sistema de Operação FreeDOS
- Compilador TurboC
- Funções inByte e outByte para aceder aos periféricos

Consulte o anexo para saber as constantes e funções necessárias à resolução do teste que não são dadas no enunciado.

Deve sempre explicitar os valores de todas as constantes que utilizar (i.e. se usar nomes simbólicos tem de dizer quais são os valores a que correspondem)

Q-6 [2.00 val.] Considere o módulo *stack*, fornecido nas aulas praticas, consistindo nos ficheiros *stack.h* e *stack.c* (protótipos à direita). Complete a implementação da função *inverte_string* que inverte uma string usando as funções do módulo dado:

```
#include <string.h>
```

```
#include _____
```

```
void inverte_string( char *s ) {  
    int i;  
    _____ ;  
    _____ ;  
    while ( _____ ) {  
        _____ ;  
        _____ ;  
    }  
    _____ ;  
    while ( _____ ) {  
        _____ ;  
        _____ ;  
    }  
}
```

Protótipos das funções do módulo *stack*:

```
void stack_init( void );  
void stack_push( char c );  
char stack_pop( void );  
int stack_empty( void );  
int stack_full( void );
```

Q-7 Suponha que dispõe de um novo controlador que permite enviar e receber bytes para um dispositivo. Esse controlador está acessível pelos endereços de I/O seguintes:

- registo de dados de leitura 0xA00
- registo de dados de escrita 0xA01
- registo de estado 0xA02
 - bit 0 a 1 quando há um byte disponível
 - bit 1 a 1 quando é possível enviar um novo byte

a) Usando **espera activa** programe as seguintes funções que, respectivamente, enviam e recebem um byte para o controlador:

a1) [2.00 val]

```
void enviar_byte( unsigned char byte ) {
```

```
}
```

a2) [2.00 val]

```
unsigned char receber_byte() {
```

```
}
```

b) [3.00 val] Programe uma função que envia todos os bytes de um ficheiro para o dispositivo. Pode usar as funções programadas na alínea anterior. Verifique as situações de erro mais comuns.

```
void enviar_ficheiro ( char *nome_fich ) {
```

```
}
```

Q-8 Todas as alíneas desta questão operam sobre a porta série COM2 cujos endereços de I/O podem ser consultados na referência dada no fim do teste.

Neste contexto, considere o controlador de interrupções do PC, 8259A (PIC) cujo registo de máscara tem o endereço de I/O 0x21. Este registo tem 8 bits e cada um deles inibe um nível de interrupções quando está a 1.

a) [2.00 val] Sabendo que a porta série COM2 usa o nível de interrupções 3, programe em TurboC a seguinte função, que permite as interrupções da COM2 no PIC:

```
void ligar_interrupcoes_com2_PIC() {
```

```
}
```

b) [2.00 val] Nesta alínea e na seguinte considere que dispõe de um buffer como o usado nas aulas práticas.

Complete a rotina que trata uma interrupção para o envio de um carácter do buffer pela porta série COM2, sabendo que a função `desligar_ints()` desliga as interrupções da COM2 na UART e no PIC.

```
void _____ int_envio() {  
    unsigned char c;  
  
    enable();  
    c = _____;  
    if ( _____ )  
        desligar_ints();  
    _____;  
    _____;  
}
```

c) [2.00 val] Programe a rotina de atendimento de interrupções que coloca os bytes recebidos pela porta série COM2 no buffer.

```
void _____ int_recepcao() {
```

```
}
```

Referência:

Funções do TurboC:

```
void enable( void );  
void disable( void );  
void setvect( int numero_da_interrupcao , void* endereco_da_rotina_de_tratamento_da_interrupcao );
```

Funções disponibilizadas:

```
unsigned char inByte( int port );  
unsigned void outByte( int port , unsigned char byte );
```

```
void buf_init( void );  
void buf_put( char byte );  
char buf_get( void );  
int buf_empty( void );  
int buf_full( void );
```

Endereços de I/O:

- UART (COM2):

Registo de Dados de Escrita (THR):	0x2F8
Registo de Dados de Leitura (RBR):	0x2F8
Registo de Controlo das Interrupções (IER):	0x2F9
Registo de Controlo do Modem (MCR):	0x2FC
Registo de Estado (LSR):	0x2FD

- PIC:

Registo de Comandos:	0x20
Registo de Máscara:	0x21

Comandos:

- Fim de Interrupção (EOI): 0x20