Arquitectura de Computadores
Teste nº 2 – versão A – 12 de Junho de 2015

Nome Nº

Teste sem consulta e sem esclarecimento de dúvidas.
Duração: 2h 15m

Questão 1 (1 valor)

O bit M do registo *status word* do processador (PSW) indica se o CPU pode executar instruções privilegiadas (se o bit M é 1) ou se não pode (se o bit M é 0). Indique o valor desse bit M em cada uma das seguintes situações:

a) o processo executa o código do programa do utilizador: M = _____

b) o processo efetuou uma chamada ao sistema e executa código do kernel. M = _____

c) o código que atende a interrupção do relógio (*timer*) é executado. M = _____

Questão 2 (2 valores) Para suportar o que um dos livros usados na cadeira "Operating Systems: Three Easy Pieces" (OSTEP) chama *limited direct execution*, a arquitectura hardware deve suportar os três seguintes mecanismos:

- instruções privilegiadas,
- proteção de memória, e
- interrupção de relógio (*timer*).

Explique porque é que basta que um deles não exista para que o sistema operativo não possa garantir que um dado processo efetua a computação que lhe está atribuída sem interferência dos outros processos.

Questão 3 (2.5 valores) Considere a forma como são suportadas as chamadas ao sistema no sistema operativo Linux sobre um CPU como a do Pentium.

a) Para realizar uma chamada ao sistema, os registos do CPU são carregados com valores que especificam qual é a chamada ao sistema e quais são os parâmetros. Por exemplo, a biblioteca do C implementa a chamada ao sistema *write(1, msg, 14)* através da execução das seguintes instruções máquina

```
movl     $14, %edx  # numero de bytes a escrever (14)
movl     $msg, %ecx # endereço onde se encontram os bytes (msg)
movl     $1, %ebx   # canal 1
movl     $4, %eax   # 4 é o código para indicar que se quer fazer um write
int      $0x80
…
.data
msg: .ascii "hello, world \n"
```

Explique em detalhe o que se passa após a execução da instrução ***int $0x80*** nomeadamente quanto ao modo em que o CPU fica, valor no PC e o estado da pilha.

b) O Pentium tem uma instrução máquina chamada *return from interrupt*, cuja mnemónica é ***iret***. Esta instrução, entre outras coisas, muda o modo de operação do CPU de modo sistema para modo utilizador. Explique porque é que a instrução *iret* existe e em que circunstâncias é usada.

Nome Nº

Questão 4 (5.5 valores)

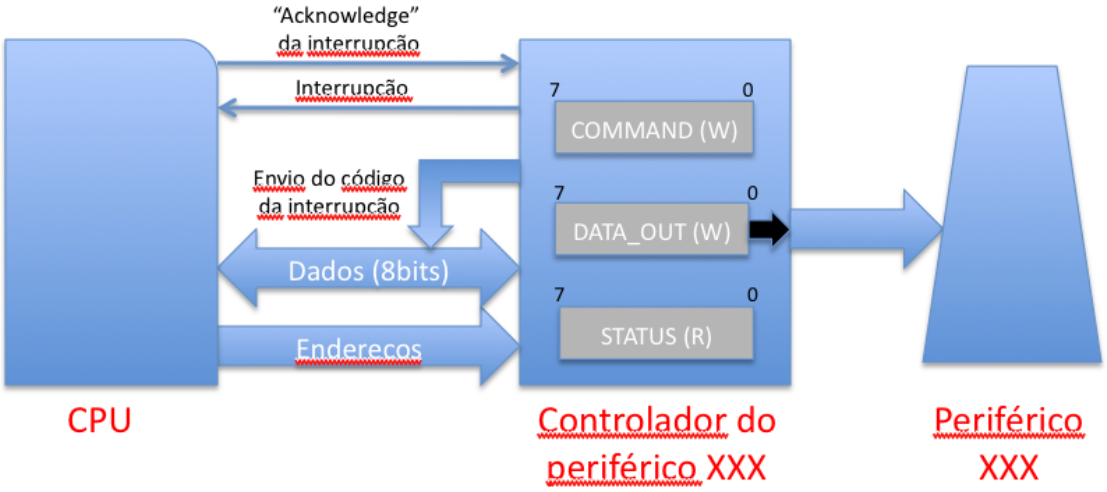
As várias alíneas desta pergunta supõem um ambiente, do ponto de vista do software, semelhante ao usado nas aulas práticas e no projecto:

- sistema operativo *Freedos*
- compilador *Turbo C* com as seguintes funções disponíveis
    - unsigned char inportb (unsigned int portid)
    - void outportb(unsigned int portid, unsigned char value)
    - void enable(void)
    - void disable(void)

Supondo a existência de uma rotina de tratamento de interrupções void my_isr() o endereço dessa subrotina é colocada na entrada 5 do vector de interrupções através da invocação da função setvect(5, my_isr).
Está ainda disponível um módulo de software que manipula um *buffer circular* e que disponibiliza as seguintes operações

- void buffer_init(): inicializa a estrutura de dados que descreve o *buffer*
- int buffer_full(): retorna 1 se não há espaço e 0 se há
- int buffer_empty(): retorna 1 se o *buffer* está vazio e 0 se não está
- void buffer_put(unsigned char c): coloca c na 1ª posição livre do *buffer*; assume que há espaço
- unsigned char buffer_get(): retorna o elemento que está há mais tempo no *buffer*; assume que há pelo menos um elemento no buffer

Do ponto de vista do hardware, o controlador do periférico XXX é o único controlador ligado à linha de interrupção do CPU. Depois do controlador accionar a linha de interrupção do CPU, aguarda que este active a linha de "Acknowledge" da interrupção; quando isto acontece o controlador coloca o código 0x11 no bus de dados e desactiva a linha de interrupção.



Todos os registos do controlador têm 8 bits e o bit 7 é o mais significativo. Os registos são os seguintes:

- **endereço 0x30 DATA_OUT**: registo que só pode ser escrito; o valor escrito neste registo é transferido para o periférico XXX
- **endereço 0x31 STATUS**: registo só de leitura; se o bit 1 está a 1 isso indica que o controlador está pronto a enviar mais um byte. Quando o CPU escreve no registo DATA_OUT este bit fica a 0.
- **endereço 0x32 COMMAND**: registo só de escrita; Quando o CPU escreve neste registo o valor 0x00 o controlador não envia interrupções para o CPU; quando o CPU escreve neste registo 0xFF o controlador acciona a linha de interrupção quando o bit 1 do registo STATUS está a 1

a) Escreva, usando o Turbo C, uma rotina com o protótipo
`void out_XXX( unsigned char c)`
que envia o byte c para o periférico XXX usando espera activa

b) Escreva, usando o Turbo C, uma rotina com o protótipo
void out_XXX(unsigned char c que envia o byte *c* para o periférico XXX usando interrupções. Escreva também a rotina de atendimento de interrupções out_XXX_int() cujo endereço está na entrada 0x11 do vector de interrupções.

c) Apresente também o código que deve ser executado antes da ocorrência da 1ªinterrupção proveniente do periférico XXX, relativamente à inicialização do periférico e do vector de interrupções.

Nome Nº

Questão 5 (1.5 valores) Considere a seguinte sequência de comandos no shell que se supõem serem executados sem erros. Os dois primeiros geram ficheiros objecto a partir de códigos fonte em C; o terceiro comando gera um ficheiro executável.

```
gcc –c –o p1.o  p1.c
gcc –c –o p2.o  p2.c
ld –o prog p1.o p2.o
```

Explique como é que o ligador *(ld)* gera o ficheiro executável.

Questão 6 (3 valores) Um dado CPU emite endereços com 24 bits. Entre o CPU e a memória central existe uma cache associativa por grupos com as seguintes características:

- uma linha tem 8 bytes
- cada conjunto ou grupo tem 4 linhas
- Há 1024 conjuntos

a) Qual é a capacidade da cache?

endereços de 24 bits - 2^24 bytes
associativo - grupos, linhas
1 linha - 8 bytes
grupo - 4 linhas
1024 conjuntos = 2^10
capa cache= nGrupos*nLinhaporGrupo*capa linha

capacidade de cache = 2^10 * 2^4 * 2^3 = 2^17

b) O CPU emite o endereço E com 24 bits. Como é que o hardware verifica que o conteúdo de E está ou não na cache? Indique claramente como é que o endereço é dividido, indicando para que servem cada um dos 24 bits.

CPU -> address de 24 bits

offset => 2^3 => offset tem 3 bits

index => 2^10 conjuntos => index tem 10 bits

tag => bits totais - bits index -bits do offset = 24 -10 -3 =11

offset - 0 a 2
index => 3 a 12
tag => 13 a 24

Nome Nº

Questão 7 (3 valores) Um sistema em que os endereços virtuais têm 28 bits, usa uma MMU que suporta páginas com dimensão 64 KBytes (2^{16}).

a) Diga como é interpretado um endereço virtual. Justifique

address tem 28 bits
capacidade das páginas é de 2^16 bytes

offset tem 16 bits => 0 a 15
nºpágina tem 28 -16 = 12 => 16 a 28

b) Quantas entradas tem a tabela de páginas de cada processo? Justifique.

nºde página é escirta até 12 bitas => 2^12 linhas diferentes

nº de entradas da tabela de páginas é 2^12 (1024*2*2)

c) Para um dado processo em execução, as primeiras entradas da tabela de páginas são as seguintes:

Nº página virtual	Nº página física (em base 16)
0	Inválida
1	0xAFE
2	0xADA
3	0xEAD

27 - 24; 23 - 20; 19 - 16; 15 - 12; 11 - 8; 7 - 4; 3 - 0

As outras entradas são todas inválidas. Indique, justificando, os endereços físicos que correspondem aos seguintes endereços virtuais. Responda "Endereço Inválido" se o endereço virtual for inválido.

2*16^4 = 2*(2^4)^4=2*2^16 = 2^17

b1) 0x0022000 0000 0000 0010 = 2^1 => página 2 => 0xADA2000

1*16^5 = 2^20 1*16^4 = 2^16

b2) 0x0110100 0000 0001 0001 = 2^4+2^0 = 17 => página inválida

3*16^4 = 3*2^16 = 2^17 + 2^16

b3) 0x0030100 0000 0000 0011 = 3 => pág 3 => 0xEAD0100

b4) 0x0121100 0x012 > 3 => página inválida

Questão 8 (1.5 valores) Explique porque é que um sistema de gestão de memória baseado em páginas necessita de um TLB (*Translation Lookaside Buffer*)