Arquitectura de Computadores

Licenciatura em Engenharia Informática

Exame de Recurso (A) – 2008/07/09 – Duração: 2h30m

Nome:_____ Número:_____

Teste sem consulta.
A interpretação do enunciado faz parte da avaliação. Explicite 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

Deve assinalar com um **X** a resposta correcta. Cada resposta errada desconta 20% da cotação da pergunta.

Q-1 [0.625 val.] Na programação de entradas/saídas por espera activa:

1. O PIC serve como intermediário entre o controlador do dispositivo e o CPU
2. A rotina de atendimento da interrupção tem de esperar até que o controlador do dispositivo detenha o byte a ser lido
3. O CPU tem de perguntar ao controlador do dispositivo se este detém o byte a ser lido
4. O ciclo de espera activa é despoletado por uma interrupção

---

Q-2 [0.625 val.] A representação de dados "little endian" significa que:

1. O byte menos significativo é sempre guardado no endereço de memória que tem o valor mais baixo
2. O byte mais significativo é sempre guardado no endereço de memória que tem o valor mais baixo
3. Se usa a norma IEEE para guardar os números reais na memória
4. Os registos do CPU podem ter 8, 16 ou 32 bits

Q-3 [0.625 val.] Considere o seguinte pedaço de código:

```
int x, *y, z;
y = &x;
...
```

Sendo que no resto do programa `y` não é alterado. Para afectarmos o valor de `x` a `z` podemos fazer:

1. z = &x;
2. z = &y;
3. z = *x;
4. z = *y;

Q-4 [0.625 val.] Diga qual das instruções está errada (não compila) em NASM IA/32:

1. `add ah, bh`
2. `sub ecx, ecx`
3. `cmp edx, eex`
4. `and ecx, edi`

Q-5 [0.625 val.] A noção de segmentação permite

1. Dividir o espaço de endereçamento de um programa em secções de tamanhos eventualmente diferentes
2. Dividir o espaço de endereçamento de um programa em secções de tamanhos sempre iguais
3. Colocar um programa na cache de forma mais eficiente
4. Suportar as tabelas de páginas na MMU

Q-6 [0.625 val.] Qual dos seguintes periféricos **não** é orientado ao byte?

1. Teclado
2. Porta série
3. Leitor de DVD
4. Impressora

Q-7 [0.625 val.] A tecnologia hyperthreading diferencia-se da multicore por:

1. Usar pipelines
2. Não duplicar as unidades de execução
3. Não usar pipelines
4. Não ser super-escalar

Q-8 [0.625 val.] A técnica de *"branch prediction"* pretende

1. Minimizar o impacto das dependência de dados
2. Maximizar o número de instruções executadas por unidade de tempo
3. Permitir que os saltos executados para trás sejam executados
4. Evitar ciclos

2 Teórica - Desenvolvimento

Q-9 [1.50 val.] Indique quais são as grandes vantagens do uso de memória virtual?

Q-10 Considere o seguinte código assembly NASM/IA-32

a) [1.00 val.] Preencha o rectângulo abaixo com o código correspondente à seguinte chamada à função `incN`:

```
incN(&x, n)
```

b) [1.00 val.] Assuma que pára a execução nos ponto assinalado. Considerando que os endereços das variáveis globais `x` e `n` são, respectivamente, 0x00001000 e 0x0001004 complete a figura à direita, escrevendo qual é o conteúdo da pilha de execução (*stack*) no ponto da interrupção. Deve também indicar para onde apontam, nessa altura, os registos ESP e EBP.

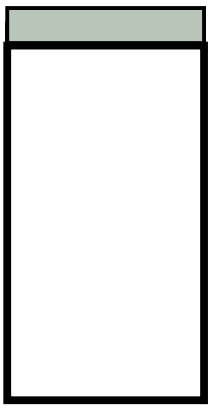
```
section .data
x:   dd   10
n:   dd   4

section .text
_start:
```

```
   mov eax, 1 ; SYS_EXIT
   mov ebx, 0
   int 0x80 ; LINUX_SYSCALL
```

```
incN:
   push ebp
   mov ebx, [ebp+12]   ⟵  ●
   mov eax, [ebp+8]
   add [ebx], eax
   pop ebp
   ret
```

Frame da subrotina incN



c) [0.50 val.] Que tipos de endereçamento é que são aplicados nas seguintes instruções?

```
mov eax, [ebp+8]
add [ebx], eax
```

Q-11 [1.50 val.] Faça um esquema que ilustre o processo de transferência por DMA. Deve incluir os passos efectuados pelo CPU e a transferência em si. Anote o esquema com a enumeração dos passos e use essa enumeração para elaborar a legenda.

Legenda:

Q-12 Admita uma arquitectura de um computador com as seguintes características: endereçamento de 24 bits, cache com mapeamento directo de 2 MBytes, onde cada linha tem 128 Bytes.

a) [0.50 val.] Indique para o endereço seguinte qual é a chave do bloco de memória e qual o byte referenciado dentro desse bloco:

1001 1001 0011 1001 0010 0101

chave: _____ linha: _____ byte dentro do bloco: _____

b) [1.00 val.] Suponha o seguinte cenário:

- a cache está vazia;
- a cache usa uma política de *write-back*;
- o acesso que se está a efectuar é de escrita.

Explique que passos são efectuados até que os bytes sejam escritos na cache.

c) [0.50 val.] Considere que a cache tem um tempo de acesso de 6 ns, que memória central tem um tempo de acesso de 50 ns e que a taxa de sucesso (*hit ratio*) no acesso à cache é de 90%. Indique o tempo médio de acesso à memória. Inclua os cálculos realizados.

d) [0.50 val.] Considere ainda que em média 20% do processamento é passado em acessos a memória. Calcule qual é o impacto da cache da alínea anterior na performance global do sistema. Indique os cálculos.

3 Prática

Q-13 Seja multiplica(x,y) uma função inteira de variáveis **inteiras, positivas, não nulas**, tal que

$$\texttt{multiplica}(x,y) = x \times y$$

a) [0.75 val.] Escreva um programa em C que ao ser executado pede ao utilizador que digite os valores de x e de y, calcula o seu produto usando a função multiplica e imprime o resultado.

Agora, decida se vai responder a b-1) ou a b-2). **Responda apenas a uma das alíneas!**
b-1) [0.50 val.] Implemente a função multiplica recorrendo a um ciclo de somas sucessivas.
b-2) [1.00 val.] Implemente a função multiplica recorrendo à seguinte definicão recursiva:

$$\texttt{multiplica}(x,y) = \begin{cases} y & \text{se } x = 1 \\ y + \texttt{multiplica}(x-1,y) & \text{se } x > 1 \end{cases}$$

Indique a que alínea está a responder.

Q-14 Abaixo encontrará as alíneas ordenadas por ordem crescente de dificuldade e, consequentemente, de cotação. **Responda apenas a uma das alíneas**; não se aventure tentando responder a alíneas de cotação mais elevada, responda àquela que julga ajustar-se melhor aos seus conhecimentos!

Considere a função $\texttt{multiplica}(x,y)$ descrita na alínea anterior. Implemente-a em assembly IA-32,
a-1) [0.75 val.] Baseada numa implementação iterativa (ciclo), sendo x, y variáveis globais (não se esqueça de as definir).

a-2) [1.25 val.] Baseada numa implementação iterativa (ciclo), sendo x, y parâmetros armazenados na pilha (stack). Aceda à pilha usando a convenção "standard"(que foi dada nas aulas).

a-3) [1.75 val.] Baseada na definicão recursiva acima descrita, sendo x, y parâmetros armazenados na pilha (stack). Aceda à pilha usando a convenção "standard"(que foi dada nas aulas).

Indique a que alínea está a responder.

Q-15 Considere que se pretende usar a segunda porta série do PC (COM2) para receber dados.
Os endereços de IO (portos) da COM2 são:

2F8H - THR e RBR - Registos de dados de escrita e de leitura

2F9H - IER - Registo de controlo das interrupções.
O bit 0 quando está a 1 liga as interrupções para leitura
O bit 1 quando está a 1 liga as interrupções para escrita

2FCH - MCR - Registo de controlo
O bit 3 quando está a 1 liga as interrupções da porta série

2FDH - LSR - Registo de estado
O bit 0 quando está a 1 indica que há um byte disponível para ler
O bit 5 quando está a 1 indica que é possível enviar um novo byte

O controlador de interrupções (PIC) tem os seguintes registos:

20H - PIC_COMMAND - Registo de comandos - serve para enviar comandos para o PIC. O único comando que é necessário usar é o EOI (End Of Interrupt) que tem o valor 20H

21H - PIC_MASK - Registo de máscara de interrupções Este registo possui um bit por cada nível de interrupções - a 0 a interrupção é permitida, a 1 é inibida. O bit 3 corresponde à interrupção gerada pela COM2

A posição do vector de interrupções usada para a COM2 é a 11.

Responda às seguintes questões com código C. Assuma que está a usar o ambiente FreeDOS e o Turbo C, dispondo das funções `inByte` e `outByte` pré-programadas (tal como nas práticas). **Tem de definir todas as *constantes* que utilizar.**

a) [1.00 val.] Programe a função `receber_byte_com2()` usando espera activa.

b) [1.00 val.] Programe a função `ligar_int_com2()` que programa todos os bits necessários para que se gerem interrupções sempre que chega um byte à porta série 2.

c) [1.00 val.] Programe a rotina de tratamento de interrupções, `IntReceive_com2()`, para tratar as interrupções causadas pela chegada de bytes à porta série. Assuma que tem disponível o módulo do buffer circular usado nas práticas.

d) [0.50 val.] Programe a função `init_int_com2()` que faz todas as inicializações necessárias para preparar a COM2 para receber bytes usando interrupções. Pode usar funções feitas nas alíneas anteriores.