Arquitectura de Computadores

Licenciatura em Engenharia Informática

Exame de Época Normal (A) – 2008/06/18 – 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.] Suponha que num processo a correr numa arquitectura IA-32 a subrotina `S` começa no endereço 0x0000FFFF e que o valor corrente do program counter (EIP) é 0x00007777 e do ESP é 0xFFFF0004. Qual das seguintes configurações está correcta depois da execução de `call S`?

1. ESP = 0xFFFF0000 e EIP = 0x0000FFFF
2. ESP = 0xFFFF0008 e EIP = 0x0000777B
3. ESP = 0xFFFF0002 e EIP = 0x0000FFFF
4. ESP = 0xFFFF0006 e EIP = 0x0000777B

Q-2 [0.625 val.] No contexto da unidade de vírgula flutuante

1. Os registos ST0 a ST7 guardam valores representados numa extensão da norma IEEE-754 a 80 bits
2. O estado da última operação efectuada é guardado no registo FLAGS
3. Existem instruções da família **`jump`** para operar sobre o registo FSW
4. Os registos ST0 a ST7 podem ter 64 ou 80 bits

Q-3 [0.625 val.] Qual é o comportamento do seguinte código C?

```
void f() {
 char * a = (char *) malloc (sizeof(char)*20);
 char b[20];
 ...
}
```

1. Reserva espaço para os vectores apontados por `a` e `b` na pilha
2. Reserva espaço para os vectores apontados por `a` e `b` no heap
3. Reserva espaço para o vector apontado por `b` no heap e para o vector apontado por `a` na pilha
4. Reserva espaço para o vector apontado por `a` no heap e para o vector apontado por `b` na pilha

---

Q-4 [0.625 val.] Considere a arquitectura IA-32 e a memória ao lado. Qual é o resultado da seguinte instrução? **`mov`** `eax, [0xFFD]`

1. eax = FC FF 04 10
2. eax = 10 04 FF FC
3. eax = 10
4. eax = CF FF 40 01

Endereço	Conteúdo
00000FFD:	10
00000FFE:	04
00000FFF:	FF
00001000:	FC

Q-5 [0.625 val.] As arquitecturas Intel utilizam um conjunto de instruções (**`in`** e **`out`**) para interagir com periféricos porque:

1. Um endereço por si só não é suficiente para distinguir se o acesso é a uma posição de memória ou a um periférico
2. Utiliza memory-mapped IO
3. As operações de troca de dados com os periféricos são fundamentalmente diferentes das operações de troca de dados com a memória, porque não usam o conceito de endereço
4. São necessárias para o controlador de DMA escrever dados directamente na memoria

---

Q-6 [0.625 val.] Dada uma arquitectura com apenas um nível de cache, com um tempo de acesso de 5ns e uma taxa de acerto de 90%, e uma memória central com tempo de acesso de 60ns. Qual é o tempo médio de acesso a memória?

1. 10.5 ns
2. 11.3 ns
3. 16.0 ns
4. 15.6 ns

Q-7 [0.625 val.] Numa arquitectura super-escalar o aumento dos estágios do pipeline

1. Resulta sempre no aumento da performance do CPU
2. Pode não resultar num aumento de performance devido, por exemplo, ao aumento da complexidade na gestão das dependências de dados
3. Implica duplicar os circuitos que mantêm o estado do CPU (por exemplo, os registos)
4. Pode não resultar num aumento de performance devido à sobrecarga dos registos

---

Q-8 [0.625 val.] O conceito de paginação no contexto de memória virtual consiste em:

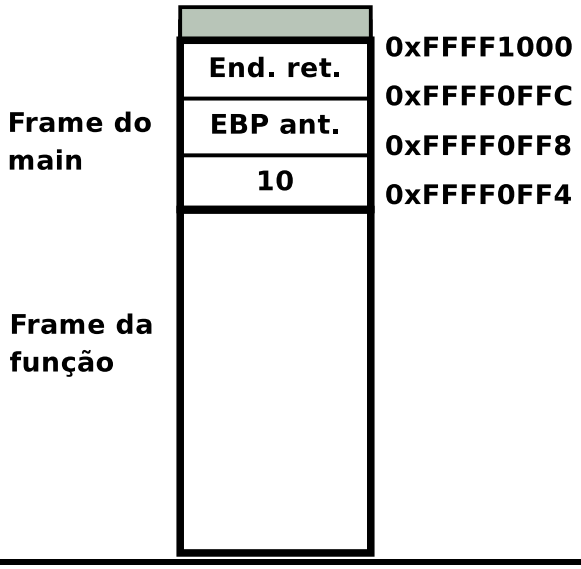
1. Colocar o espaço de endereçamento de um programa todo na memória central ou todo em disco
2. Dividir o espaço de endereçamento de um programa em secções para código e dados
3. Desfragmentar a memória
4. Dividir o espaço de endereçamento de um programa em várias páginas de tamanho fixo

2 Teórica - Desenvolvimento

Q-9 [1.50 val.] Considere o seguinte código C e assuma que pára a execução no ponto assinalado. Complete a figura à direita, escrevendo qual é o conteúdo da pilha de execução (*stack*). Deve também indicar para onde apontam, nessa altura, os registos ESP e EBP.

```
void soma(int *a, int b) {
    int temp = *a;
    return temp+b; ⟵   ●
}

int main() {
    int x = 10;
    soma(&x, 20);
    return 0;
}
```



Q-10 [1.50 val.] Descreva que operações são efectuadas por um processador Intel para efectuar a seguinte sequência de instruções:

```
cmp eax, [1000]
jz fim
```

Q-11 [1.50 val.] Considere o periférico porta série e o seu controlador UART. Faça uma figura que ilustre os passos necessários para que o CPU leia um byte do periférico, via **espera activa**, e o escreva em memória. Anote a figura com a enumeração dos passos e use essa enumeração para elaborar a legenda correspondente.

Legenda:

Q-12 Admita uma arquitectura de um computador com endereçamento de 20 bits e com uma cache associativa por grupos (conjuntos).
Considere o seguinte endereço 1101 0110 0001 0000 1011, onde:

- o deslocamento (byte dentro do bloco) é dado por 0 1011 (os últimos 5 bits - do bit 0 ao bit 4);
- o número do conjunto é dado por 0001 000 (do bit 5 ao bit 11);
- a chave é dada por 1101 0110 (os primeiros 8 bits - do bit 12 ao bit 19).

a) [0.50 val.] Quantos conjuntos tem a cache? _____
b) [0.50 val.] Qual é o tamanho em Bytes de um bloco de memória? _____
c) [0.50 val.] Qual é o tamanho da cache em KBytes? _____
d) [1.00 val.] Qual é a desvantagem das memórias cache associativas puras? De que forma essa desvantagem é colmatada pelas memórias cache associativas por conjunto?

Q-13 [1.00 val.] Qual é o papel do TLB numa arquitectura com memória virtual gerida através de paginação?

3 Prática

Q-14 Como sabe, em "C"as strings são representadas por vectores terminados por um carácter `'\0'`.
a) [1.50 val.] Escreva em "C"a função:

int mystrcmp(**char** s1[], **char** s2[])

A função tem como argumento duas strings e devolve 0 se as strings forem iguais e um valor diferente de 0 se forem diferentes.
ATENÇÃO: Para ter a cotação máxima deve aceder aos caracteres da string usando apontadores em "C". Se usar vectores e índices a resposta será cotada para 75%.

b) [2.00 val.] Traduza a para assembler NASM IA/32 sua função em "C"da alínea anterior ou faça uma função equivalente.

ATENÇÃO:

- Para a resposta ter a cotação máxima deve usar as convenções de passagem de parâmetros e retorno de resultados dadas na teórica.
- Se usar uma convenção de passagem de parametros/retorno de resultados definida por si a resposta será cotada para 80%.
- Se fizer um pedaço de código equivalente que não seja uma função a resposta será cotada para 60%.

Q-15 Suponha que tem um computador capaz de ler e escrever dados num dispositivo **D**, orientado ao byte; o controlador de I/O do dispositivo é programado/acedido usando registos de 8 bits com os seguintes endereços (portos):

0x400 - Registo de dados de leitura
0x401 - Registo de dados de escrita
0x402 - Registo de estado
    bit 0 - a 1, indica que existe um byte pronto para ler
    bit 1 - a 1, indica é possível enviar um novo byte para o dispositivo
0x403 - Registo de comando
    bit 0 - a 1 liga as interrupções para a recepção de bytes; a 0 desliga-as
    bit 1 - a 1 liga as interrupções para o envio de bytes; a 0 desliga-as

Caso exista, uma rotina para tratamento de interrupções de **D** deverá ser instalada na posição 15 do vector de interrupções. Não é necessário recorrer a qualquer outro dispositivo (por exemplo, PIC) para a operação de **D**, quer por espera activa, quer por interrupções.

Considere que esse computador corre o sistema FreeDOS, e tem disponíveis o Turbo C e o Turbo Assembler (tal como no ambiente usado nas aulas praticas). Considere ainda que dispõe das mesmas funções `inByte` e `outByte` que usou nas aulas práticas. **Nota:** Deve responder as seguintes perguntas com código em C.

a) [1.25 val.] Assumindo que o dispositivo já está programado para trabalhar em espera activa, programe a função `escrever_byte(unsigned char b)`, que envia um byte para o dispositivo **D** (usando espera activa).

b) [0.50 val.] Agora vamos usar interrupções para enviar bytes para o dispositivo **D**. Nesta alínea e na próxima assuma que está disponível o módulo "buffer circular" usado na prática.
Programe todas as inicializações necessárias para preparar **D** para o envio de bytes usando interrupções.

c) [1.75 val.] No contexto da alínea anterior (envio de bytes por interrupções para **D**), programe a rotina de serviço da interrupção, `IntSend`, que trata uma interrupção para escrita de um carácter em **D**.