Arquitectura de Computadores

Licenciatura em Engenharia Informática

Exame de Época Normal (A) – 2007/06/25 – Duração: 2h00m + 15m tolerância

Nome:_____ Número:_____

Total de páginas: 6+___ *páginas* Classificação:_____

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.
Tenha em atenção estes dois pontos:

1. A aprovação à disciplina está sujeita a uma nota mínima de 7.5/20 em ambas as partes do exame (teórica e prática). Neste exame em concreto essas notas são 4.69 na parte teórica e 2.81 na parte prática.
2. Na pergunta 10a) responda apenas a uma das opções, indicando com uma cruz a sua escolha.

Resumo da classificação:

Q-1 a) [0.75] = _____	Q-6 [1.00] = _____	**Teórica [12.50]** = _____
Q-1 b) [0.75] = _____	Q-7 [2.00] = _____	Q-10 a) [2.00] = _____
Q-1 c) [0.25] = _____	Q-8 a) [0.50] = _____	Q-10 b) [2.00] = _____
Q-1 d) [0.75] = _____	Q-8 b) [0.50] = _____	Q-11 a) [1.00] = _____
Q-2 [1.00] = _____	Q-8 c) [0.75] = _____	Q-11 b) [1.50] = _____
Q-3 [1.00] = _____	Q-8 d) [0.50] = _____	Q-11 c) [1.00] = _____
Q-4 [1.25] = _____	Q-8 e) [0.50] = _____	**Prática [7.50]** = _____
Q-5 [1.00] = _____	Q-9 [1.00] = _____	**Total [20.00]** = _____

Q-1 Considere o ISA IA-32 utilizado nas arquitecturas Intel 80x86.

a) [0.75 val.] Estamos perante uma arquitectura do tipo CISC ou RISC? Justifique.

b) [0.75 val.] Comente a afirmação: *Os modos de endereçamento oferecidos por uma dada arquitectura não contribuem para a complexidade do conjunto de instruções.*

c) [0.25 val.] Coloque a seguinte palavra de 32 bits (0F AE 33 3F) no endereço 0FFF da memória abaixo. Escreva o novo valor na coluna *Conteúdo Actualizado*.

Endereço	Conteúdo	Conteúdo Actualizado
00000FFC:	2F	_____
00000FFD:	10	_____
00000FFE:	04	_____
00000FFF:	FF	_____
00001000:	FC	_____
00001001:	00	_____
00001002:	10	_____
00001003:	A3	_____
00001004:	01	_____

d) [0.75 val.] Considerando a mesma zona de memória, diga qual é o resultado das seguintes operações. *Considere que todas as posições de memória não indicadas na tabela acima contêm o valor zero.*

`mov ax, [1000H]`	ax = _____	Lembre-se que o H denota que o número está em hexadecimal.
`mov ax, 1000`	ax = _____	
`mov ax, bx`	ax = _____	O conteúdo de bx é OFFCH.
`mov ax, [ebx]`	ax = _____	O conteúdo de ebx é OFFCH.
`mov ax, {1001H}`	ax = _____	Suponha que a notação { } denota endereçamento indirecto por memória. Algo que não existe neste assembly.

Q-2 [1.00 val.] Explique a diferença entre o deslocamento (shift) lógico e aritmético.

Q-3 [1.00 val.] Escreva o código NASM/Intel (IA-32) que efectua a chamada à função `count` ilustrada no seguinte código C: `n = count('a')`. O protótipo da função é: `int count(char)`.

Q-4 [1.25 val.] Explique porque é que se usa a pilha para passar os valores para os parâmetros das subrotinas.

Q-5 [1.00 val.] Construa a *codeword* com o código de Hamming para a seguinte palavra de memória (*word*):

0010 1110 0000 1011

Note que tendo a palavra 16 bits são precisos pelo menos 5 bits de paridade. **Nota:** os bits podem ser numerados da esquerda para a direita (como no livro), ou da direita para a esquerda (como no acetatos). Indique a sua escolha.

Q-6 [1.00 val.] Explique o conceito de volume (de discos) RAID e quais as suas vantagens.

Q-7 [2.00 val.] Faça um esquema que ilustre o funcionamento de um periférico que usa interrupções - por exemplo, recepção de dados numa UART. Anote o esquema com a enumeração dos passos necessários à recepção dos dados e use essa enumeração para elaborar a legenda do esquema.

Legenda:

Q-8 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.] Qual é o tamanho máximo para o espaço de endereçamento de um processo?

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

0110 1111 0000 0001 0000 1011

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

c) [0.75 val.] Suponha o seguinte cenário:

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

Explique que passos são efectuados até que os bytes sejam escritos nos endereços pretendidos.

d) [0.50 val.] Considere que a cache tem um tempo de acesso de 5 ns, que memória central tem um tempo de acesso de 30 ns e que a taxa de sucesso (*hit ratio*) no acesso à cache é de 80%. Indique o tempo médio de acesso à memória.

e) [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.

Q-9 [1.00 val.] Diga o que entende por, e em que cenários se recorre à técnica de *branch prediction*.

Q-10 Para a função que se segue, em assembly, poderá fazer uso das instruções que desejar; contudo, fica aqui uma sugestão: as instruções `SHL` ou `SHR` (shift left ou right), deslocam o registo alvo para a esquerda ou direita, colocando o bit que "salta fora"no carry bit.

a) [de 1.00 a 2.00 val.] Implemente em assembly uma função, que vamos designar por `func`, que recebe como parâmetro um byte e devolve como resultado (em EAX) um inteiro que indica o número de bits a 1 encontrado no byte (Exemplo: se o byte passado como argumento for 01010111, devolve 5). **Implemente apenas uma das opções, indique a sua escolha com uma cruz.**

- **[1.00 val.]** O argumento é passado para a função num registo à sua escolha.
- **[1.50 val.]** O argumento é passado para a função na pilha, e acedido na função via ESP.
- **[2.00 val.]** O argumento é passado para a função na pilha, e acedido na função via EBP.

`func:`

b) [2.00 val.] Implemente em assembly uma função, de nome `setParity`, que tem como parâmetros um byte e um inteiro. Para simplificar a leitura do enunciado denotemos esses parâmetros por `byte` e `parmode`.

`setParity` deve recorrer à função `func` da alínea anterior para determinar o número de bits a 1 de `byte` e deve usar esse valor, mais o de `parmode`, para definir o valor do bit mais significativo de `byte` (a paridade).

- Se `parmode` = 0 e `func(byte)` é par então o bit mais significativo de `byte` será 0.
- Se `parmode` = 0 e `func(byte)` é ímpar então o bit mais significativo de `byte` será 1.
- Se `parmode` = 1 então o bit mais significativo de `byte` será 0.
- Se `parmode` = 2 então o bit mais significativo de `byte` será 1.

Sugestão: Considere a divisão com sinal e use a instrução `IDIV`. Esta instrução recebe apenas um operando `src`, realizando a seguinte operação `EAX` ← `EDX:EAX/src`, sendo que em `EDX` fica o resto da divisão.

```
setParity:
```

Q-11 Relembre a recepção de dados da porta série por via de interrupções. Considere que tem à sua disposição as funções:

- `ligaPIC`/`desligaPIC` que configuram o PIC para aceitar/ignorar interrupções da UART;
- `ligaUART`/`desligaUART` que ligam/desligam as interrupções na UART;
- `confVI`/`repoeVI` que configuram/repõem o vector de interrupções para associar a função `trataByte` (a implementar na alínea b)) às interrupções provenientes da UART;
- `bufGet`/`bufPut`/`bufFull`/`bufEmpty` que permitem operar sobre um buffer circular.

a) [1.00 val.] Complemente a implementação de um programa principal que: configura o sistema para a recepção de bytes da UART via interrupções; para cada byte recebido invoca `func` e usa o resultado para acumular em `count` o número total de bits recebidos a 1; repõe a configuração inicial do sistema (desliga as interrupções); por fim; imprime o número de bits contabilizados.

```
#define RBR 0x3F8
#define PIC_COMMAND 0x20
#define EOI 0x20
#define EOT 0x04

int main() {
    unsigned char c = 0;
    int count;
    
    while (c != EOT) {
        
    }
    
    printf("Numero de bits a 1: %  ", count);
    return 0;
}
```

b) [1.50 val.] Implemente a função que trata a recepção do byte da porta série.

```
void trataByte() {
    unsigned char c;
    
    if (                              )
        
    else
        
    
}
```

c) [1.00 val.] Admitindo que está a usar o ambiente Linux (não o DOS/Turbo C), indique quais seriam os passos necessários para obter o executável a partir das linguagens fontes assembly e C. Considere que o programa se chama *prog.c* e a função *func.asm*.