Arquitectura de Computadores

Licenciatura em Engenharia Informática

Teste 2 (A) – 2009/05/13 – Duração: 1h30m

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

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.] Considere a FPU Intel. Qual é a diferença entre as instruções da família F, por exemplo FSTP, e as da família FI, por exemplo, FISTP?

1. Nas segundas está implícita uma conversão de vírgula flutuante para inteiro ou vice-versa
2. As primeiras operam sobre números com sinal e as segundas sobre números sem sinal
3. As segundas permitem o uso de endereçamento imediato
4. Nenhuma

Opção correcta ☐

Q-2 [1.0 val.] A estruturação do motor de execução de um processador num pipeline:

1. Diminui o tempo de execução de cada instrução
2. Aumenta o número de instruções executadas por unidade de tempo
3. Aumenta o número de acessos a memória
4. Diminui o número de acessos a memória

Opção correcta ☐

Q-3 [1.0 val.] As arquitecturas VLIW (Very Large Instruction Word):

1. Delegam a responsabilidade do escalonamento das instruções a executar nos compiladores
2. Reordenam as instruções em tempo de execução para evitar as dependências de dados
3. Dividem as instruções em vários pequenos estágios
4. Não podem executar várias instruções ao mesmo tempo

Opção correcta ☐

Q-4 [1.0 val.] Considere uma arquitectura com endereços de 32 bits e uma cache de mapeamento directo com 1024 linhas, cada um com um bloco de 64 bytes. Divida o endereço 0100 1111 0010 1011 0111 1110 1011 0100 nas suas componentes.

1.	Deslocamento: 11 0100	Linha: 01 1111 1010	Chave: 0100 1111 0010 1011
2.	Deslocamento: 10 1011 0100	Linha: 01 1111	Chave: 0100 1111 0010 1011
3.	Deslocamento: 11 0100	Linha: 1010 1101 1111 1010	Chave: 01 0011 1100
4.	Deslocamento: 10 1011 0100	Linha: 1100 1010 1101 1111	Chave: 01 0011

Opção correcta ☐

Q-5 [1.0 val.] No conceito de paginação utilizado na gestão da memória virtual:

1. Cada processo tem associada uma tabela de páginas
2. Cada tabela de página tem associado um TLB (Translation Look-aside Buffer)
3. Todos os processos partilham a mesma tabela de páginas
4. Cada processo tem associado uma TLB (Translation Look-aside Buffer)

Opção correcta ☐

2 Prática

Q-6 [2.5 val.] Sabendo que o perímetro do círculo é dado por $2.\pi.r$, complete o programa que se segue. Coloque o resultado do cálculo na variável **perimetro**. Pode utilizar as variáveis auxiliares que achar necessárias.

```
global _start
section .data
raio:       dd      5.1
perimetro:  dd      0.0
```

```
section .text
_start: nop
```

```
        mov eax, 1
        mov ebx, 0
        int 0x80
```

Q-7 [3.0 val.] Considere as seguintes declarações de variáveis inteiras sem sinal:

```
section .data
res:    dd      0
x:      dd      10
y:      dd      20
```

Preencha os espaços de forma a completar o código em NASM/IA-32 que executa as acções especificadas na linguagem C.

```
res = 2*(x-y)-5;
x = 0;
y++;
```

```
    mov eax, _________
    ______ eax, ______      ; faz x - y

    ______ eax, ______      ; multiplica por 2

    ______ _____, ____      ; subtrai 5

    mov ________, eax       ; guarda em res

    sub eax, ________
    mov ______ , eax        ; põe x a 0

    ______ dword ______     ; soma 1 a y
```

```
if ( x != 100 )
   res = res + y;
else
   res = res - y;
```

```
        mov eax, ________       ; teste da condição
        ________ eax, 100
        ______ __________

        mov eax, ________       ; bloco do if
        _____ ______, eax
        _____ __________

_else:  mov eax, ________       ; bloco do else
        _____ ______, eax
_endif:
```

Q-8 Considere a função conta_pares que dado um vector de inteiros a 32 bits sem sinal e o seu número de elementos, conta quantos destes elementos são pares. O protótipo é o seguinte:

```
int conta_pares( unsigned int v[], int size );
```

a) [4.5 val.] Programe conta_pares como uma subrotina em assembler NASM/IA-32. Deve utilizar as convenções de passagem de parâmetros e retorno de valores do C.

b) [1.0 val.] Traduza para NASM/IA-32 a declaração e inicialização das variáveis vec e x apresentadas no seguinte código em C:

```
int vec[] = { 1, 2, 3, 4, 5, 6 };
int x = 0;
```

section .data

c) [1.5 val.] Considerando a função conta_pares da alínea a) e as variáveis da alínea b), apresente o código NASM/IA-32 que efectua a seguinte chamada: x = conta_pares(vec, 6);

Q-9 [2.5 val.] Considere a função:

```
unsigned int log2(unsigned int n);
```

que calcula o logaritmo base 2 de n, **sem recorrer à FPU**, sendo $n \geq 1$. Por exemplo: `log2(8) = 3` (visto que $2^3 = 8$).
Se n não for uma potência de 2, o resultado deve ser dado por defeito (arredondado para baixo), por exemplo: `log2(9) = 3`.

Hipótese 1: Programe a função como uma subrotina em assembler, utilizando as convenções de passagem de parâmetros e retorno de valores do C.

Hipótese 2: Se não conseguir programar o cálculo do logaritmo base 2 (Hipótese 1), considere $log_2(x) = \frac{x}{2}$. Neste caso a sua resposta será cotada para **metade** da cotação total da pergunta.

Principais instruções da IA-32 e directivas do NASM

Movimento de dados
mov *op1*, *op2*

Aritméticas
inc *op*
dec *op*
add *op1*, *op2*
cmp *op1*, *op2*
neg *op1*, *op2*

mul *op*
imul *op*
div *op*
idiv *op*

Lógicas e de bits
or *op1*, *op2*
and *op1*, *op2*
test *op1*, *op2*
xor *op1*, *op2*
not *op*

shl *op*, *n*
shr *op*, *n*
sal *op*, *n*
sar *op*, *n*
rol *op*, *n*
ror *op*, *n*

Saltos
jmp *label*

jz *label*
jnz *label*
jc *label*
jnc *label*
jo *label*
jno *label*
js *label*
jns *label*

je *label*
jne *label*

ja *label*
jae *label*
jb *label*
jbe *label*

jg *label*
jge *label*
jl *label*
jle *label*

Pilha
push *op*
push *op*

Subrotinas
call *label*
ret

Unidade de Reais
fld *op*
fild *op*
fldz
fld1
fldpi

fstp *op*
fistp *op*

fld st0
ffreep *st0*
fxch *op1*, *op2*

faddp st1
fsubp st1
fsubrp st1
fmulp st1
fdivp st1
fdivrp st1

fcomi st1

fchs
fabs

fsin
fcos
fptan
fpatan
fsqrt

Várias
int *n*
nop

Directivas nasm
db
dw
dd
dq
dt

Qualificadores de memória
byte
word
dword
qword
tword