Arquitectura de Computadores

Exame de recurso - 05/Julho/2010 - Duração: 2h30m

Nº:_____ Nome:_____

- Sem esclarecimentos de dúvidas. Explicite nas suas respostas todas as hipóteses assumidas.
- Sem consulta
- Não é permitido o uso de calculadoras nem de telemóveis.
- A detecção de fraude implica a reprovação na cadeira de **todos** os envolvidos.

Perguntas de escolha múltipla

Q-1 [0.5 val.] Uma das características das arquitecturas RISC é:

1. ter mais memória RAM do que as máquinas CISC
2. ter menos memória RAM do que as máquinas CISC
3. ter mais registos do que as máquinas CISC
4. ter menos registos do que as máquinas CISC

Resposta correcta _____

Q-2 [0.5 val.] Um bus é um meio

1. interno ao CPU que pode interligar vários componentes
2. externo ao CPU que pode interligar vários componentes
3. que interliga apenas três componentes (CPU, memória e periférico)
4. que pode interligar vários componentes, quer dentro do CPU quer fora do CPU

Resposta correcta _____

Q-3 [0.5 val.] Uma instrução máquina de salto incondicional, por exemplo *jmp endereço* no *assembly* NASM/IA-32, altera o conteúdo do registo

1. Program Counter, passando este a conter o endereço calculado durante a execução da instrução
2. Program Counter, passando este a conter o o endereço da instrução seguinte ao salto
3. Instruction Register, passando este a conter o endereço calculado durante a execução da instrução
4. Stack Pointer, passando este a conter o endereço da instrução seguinte ao salto

Resposta correcta _____

Q-4 [0.5 val.] No contexto da linguagem C, um erro de compilação é um erro que ocorre aquando da

1. ligação dos vários ficheiros objecto (.o) para gerar o executável
2. execução do programa
3. tradução de código fonte (.c) para código assembly (ou código máquina)
4. gravação do ficheiro fonte (.c) em disco

Resposta correcta _____

Q-5 [0.5 val.] No contexto da arquitectura Intel IA-32 a instrução *pop ebx*

1. coloca o conteúdo do registo ebx no topo da pilha de execução do programa
2. coloca o conteúdo da posição de memória apontada pelo registo ebx no topo da pilha de execução do programa
3. coloca o conteúdo do registo ebx no topo da pilha da FPU
4. retira o conteúdo do topo da pilha de execução do programa e coloca-o no registo ebx

Resposta correcta _____

Q-6 [0.5 val.] Considere uma cache de mapa directo com a capacidade total de 64 Kbytes e com linhas de 64 bytes. Suponha que o CPU emitiu o endereço E; a parte de E que escolhe a linha da cache em que poderá estar o conteúdo de E tem

1. 8 bits
2. 9 bits
3. 10 bits
4. 11 bits

Resposta correcta _____

Q-7 [0.5 val.] Durante a execução de um dado programa foi medido um "miss rate"na cache de 10 %. Sabendo que o tempo de acesso à cache é de 10 ns e o tempo de acesso à RAM é de 100 ns, qual é o tempo médio de acesso à memória do programa?

1. 15 ns
2. 19 ns
3. 91 ns
4. 85 ns

Resposta correcta _____

Q-8 [0.5 val.] Um dado CPU emite um endereço virtual com 34 bits e cada página tem 64 Kbytes. Supondo os bits do endereço numerados de 33 a 0 (em que o bit 0 é o menos significativo), a parte do endereço que identifica a página virtual

1. vai do bit 31 ao bit 16
2. vai do bit 0 ao bit 17
3. vai do bit 33 ao bit 16
4. vai do bit 33 ao bit 17

Resposta correcta _____

Q-9 [0.5 val.] São usadas interrupções para efectuar efectuar operações de entrada-saída porque assim

1. as operações de entrada-saída são mais fiáveis
2. o código de controlo do periférico é mais fácil de desenvolver
3. o hardware do controlador pode utilizar *DMA (Direct Mempry Access)*
4. o CPU e os periféricos são usados de forma mais eficiente

Resposta correcta _____

Q-10 [0.5 val.] Num sistema de interrupções que usa um vector de interrupções, ao periférico HD1 corresponde a entrada 10 no vector de interrupções em memória. A entrada 10 do vector de interrupções contém

1. o endereço do registo de controlo do periférico HD1
2. o endereço de memória correspondente ao início da rotina que trata as interrupções do periférico HD1
3. o endereço do registo de dados do periférico HD1
4. o endereço da rotina de inicialização do periférico HD1

Resposta correcta _____

Perguntas que não são de escolha múltipla

Q-11 [1.25 val.] Considere as seguintes instruções *assembly* do Pentium IA-32 em que se usou a sintax do NASM. No quadro seguinte, assinale quais são os modos de endereçamento que são usados em cada uma das instruções.

	Imediato	Registo	Directo	Indirecto por registo	Indirecto baseado
add [ebp+4], 5					
add [ebx], 5					
add eax, [1000]					
add ebx, eax					

Q-12 [1.25 val.] Explique porque é que um programa escrito na linguagem C pode ser facilmente transportado para um CPU diferente, enquanto um escrito em *assembly* não.

Q-13 [1.0 val.] Um dado sistema gere a memória através de paginação. A tabela de páginas está em RAM e é usado um TLB (Translation Lookaside Buffer). Descreva o processo de conversão de endereços virtuais em endereços físicos.

Q-14 [1.0 val.] A memória central de um dado sistema é gerida através de paginação a pedido e um endereço virtual tem 32 bits. O sistema possui 1 Gbyte de RAM. Nestas condições, qual é a dimensão (em bytes) da maior imagem de um programa que pode ser executado? Justifique.

Q-15 [0.5 val.] Explique porque é que se usa uma instrução máquina *int XX* para invocar os serviços do sistema operativo.

Perguntas práticas

Q-16 Na linguagem C

a) [1.5 val.] Implemente a função *strcount* que conta o número de ocorrências de um carácter *c* numa dada string *str*.

```
int strcount (_____________ str[], char c) {




}
```

b) [1.0 val.] Implemente a função *inc* que incrementa o valor de uma variável inteira *var* cujo endereço é um parâmetro de entrada.

```
________ inc(int *var) {

 _____________________________;

}
```

c) [0.5 val.] Utilize a função *inc* da alínea anterior para incrementar o conteúdo de uma variável inteira *x*.

```
inc(________)
```

Q-17 [2.0 val.] Pretende-se escrever um pedaço de código em *assembly* do Pentium que, dado um vector de inteiros *v* e a respectiva dimensão *dim*, some a todos os elementos do vector um valor inteiro *val*. Complete o código seguinte, que usa as convenções do NASM.

```
section .bss
v:       resd   1000
dim:     resd   1
val:     resd   1

section .text
  ...
; código que inicializa v, val e dim
  ...
; código pretendido

            mov eax, _______________ ; eax recebe val

            mov ecx, _______________ ; ecx recebe dim

            mov ebx, _______________ ; ebx recebe o endereço de v[0]

ciclo: add  ___________________,  ___________________

            ________________________

            ________________________
```

Q-18 Considere a função *f* definida da seguinte forma:

```
f(x) =  2x + 1   se     x >= 0
       -x        se     x < 0
```

a) [1.0 val.] Preencha o seguinte código *assembly NASM/IA-32* que chama a função *f* com o argumento *7*, ou seja *f(7)*, e coloca o resultado na variável global *x*

```
_________ dword _____________

call ____________________

add _____________, _________

mov [x], _________
```

b) [1.0 val.] Implemente a função *f* em assembly NASM/IA-32. Para obter a cotação completa o argumento da função deve ser passado através da pilha (stack). A utilização da passagem por registo implicará uma penalização de 50%.

```
f:




      ret
```

Q-19 Considere um dispositivo XPTO que transmite um byte de cada vez e que é gerido por um controlador hardware com os seguintes registos, todos com 8 bits, em que o bit 7 é o mais significativo.

Endereço	Função	Utilização
0x2d0	Dados (só de escrita)	Quando é escrito um valor neste registo, desencadeia-se automaticamente o seu envio pelo dispositivo
0x2d1	Estado (só de leitura)	Se o bit 6 deste registo está a 1 isso significa que o dispositivo está ocupado a transmitir o último byte colocado no registo de dados; se o bit 6 está a 0 então o dispositivo está livre e pronto a transferir mais um byte. Assim que é escrito um valor no registo de dados este bit passa a 1
0x2d2	Controlo (leitura e escrita)	O bit 0 deste registo indica se o controlador do dispositivo envia ou não interrupções ao CPU. Se o bit 0 está a 1, o controlador interrompe o CPU sempre que o bit 6 do registo de estado passa a 0; se o bit 0 está a 0, o controlador não envia interrupções ao CPU. Quando o controlador recebe energia este bit 0 está a 0.

Suponha disponíveis as funções

```
unsigned char inByte (unsigned short address)
void outByte( unsigned short address, unsigned char value)
```

que permitem o acesso aos espaço de endereços dos controladores de entrada/saída.

a) [1.0 val.] Complete a código da função *void sendXPTO(unsigned char v)* que transmite o valor v usando o dispositivo XPTO; suponha que o controlador do dispositivo XPTO não efectua interrupções, tendo portanto de usar a técnica da espera activa.

```
void sendXPTO( unsigned char v){




}
```

b) [2.0 val.] Pretende-se usar interrupções para efectuar a transmissão de bytes através do dispositivo XPTO. Para esse efeito usa-se um "buffer"circular que é manipulado pelas operações

void bufInit(void)	Inicializa a estrutura de dados que representa o buffer circular
void bufPut(unsigned char v)	Coloca *v* no buffer circular na 1ª posição livre Assume que há pelo menos uma casa livre.
unsigned char bufGet(void)	Retorna o elemento que está há mais tempo no buffer circular. Assume que existe pelo menos uma casa ocupada.
int bufEmpty(void)	Retorna um valor diferente de 0, se não há nenhum elemento no buffer; 0 se existe pelo menos um
int bufFull(void)	Retorna um valor diferente de 0, se o buffer está cheio; 0 se existe pelo menos uma casa livre

Está ainda disponível a rotina

void endOfInterruptXpto( )	permite interrupções de todos os periféricos com prioridade igual ou inferior a XPTO
void setvecXpto(void interrupt (*func)())	preenche a entrada do vector de interrupções atribuída ao periférico XPTO com o endereço da função *func*

Complete o código abaixo

```
/* Inicialização das operações de transmissão usando interrupções */
void initXptoInt(){




}
/* Rotina de envio de bytes */
void sendXptoInt( unsigned char v){









}
```

```
/* Rotina de tratamento de interrupções */
void interrupt XptoIntHandler ( void){









}
```

Principais instruções do IA-32 e directivas do NASM

Movimento de dados
mov *op1, op2*

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

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*
pop *op*

Subrotinas
call *label*
ret

Várias
int *n*
nop

Directivas para reserva de espaço sem e com inicialização
resb
resw
resd
resq
db
dw
dd
dq

Qualificadores de memória
byte
word
dword
qword
tword