Arquitectura de Computadores

Licenciatura em Engenharia Informática

Teste 1 (A) – 2009/03/25 – 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

Deve assinalar com um **X** a resposta correcta. Cada resposta errada desconta 25% da cotação da pergunta.

Q-1 [1.00 val.] Uma das características que diferencia as arquitecturas CISC e RISC é:

1. As arquitecturas CISC são load/store
2. As arquitecturas CISC possuem, por norma, mais registos do que as arquitecturas RISC
3. As arquitecturas CISC permitem a manipulação de posições de memória em instruções lógicas e aritméticas
4. As arquitecturas CISC executam aplicações mais complexas do que as RISC

Q-2 [1.00 val.] Considere os seguintes excertos de código IA-32/NASM, onde **`jg`** (Jump if Greater) opera sobre números com sinal e **`ja`** (Jump if Above) opera sobre números sem sinal:

```
mov al, 255                mov al, 255
cmp al, 0                  cmp al, 0
jg MaiorQueZero            ja MaiorQueZero
```

1. O salto para `MaiorQueZero` é efectuado em ambos os códigos
2. O salto para `MaiorQueZero` não é efectuado em ambos dos códigos
3. O salto para `MaiorQueZero` é apenas efectuado pela execução da instrução **`jg`**
4. O salto para `MaiorQueZero` é apenas efectuado pela execução da instrução **`ja`**

Q-3 [1.00 val.] Considere as seguintes declarações em C:

```
struct s {
   int a; // inteiro a 4 bytes
   int b; // inteiro a 4 bytes
};
struct s v[100];
```

Seleccione a opção certa para carregar o valor de `a[i].b` para o registo `eax`:

1.
```
mov ebx, [i]
mov eax, [v+ebx*4+4]
```
2.
```
mov ebx, [i]
mov eax, [v+ebx+4]
```
3.
```
mov ebx, [i]
mov eax, [v+ebx*8+4]
```
4.
```
mov ebx, [i]
mov eax, [v+ebx*8+8]
```

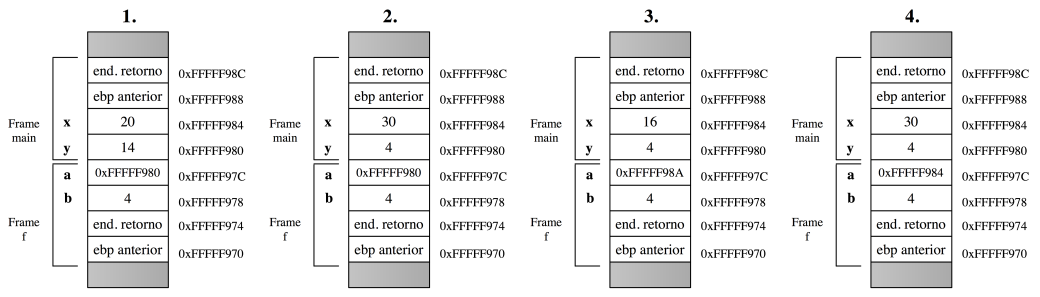
Q-4 [1.00 val.] Indique qual das seguintes afirmações é verdadeira:

1. Cada arquitectura tem apenas um assembly para a sua programação
2. Cada componente de um computador tem uma ligação (bus) dedicada ao CPU
3. A memória endereçável por um programa é limitada pelo número de bits que compõe um endereço
4. O modelo conceptual da máquina de von Neumann introduziu o conceito de registo

---

Q-5 [1.00 val.] Considere o seguinte código C: Indique qual das seguintes configurações da pilha de execução está correcta se pararmos a execução de programa antes da instrução apontada pela seta.

```
void f(int *a, int b) {
   a--;
   *a = b+10;
   return;    <==
}
int main() {
   int x = 20, y = 4;
   f(&x , y);
   return 0;
}
```



2 Prática

Q-6 [2.50 val.] Complete o programa seguinte que dado um número n, fornecido pelo utilizador, mostra todos os números ímpares compreendidos no intervalo $[1,n]$. Para tal recorra à função `impar` que verifica se um dado número é ímpar.

```
#include ____________________

_________ impar( int x ) {
  return ____________________________________ ;
}

_________ main() {
  int i, n;

  scanf( _________________, _________________ );
  for ( i = 1 ; i <= n; i++ )
    if ( ____________________________________ )
      printf( _________________, _____________________ );
  ____________________________ ;
}
```

Q-7 a) [2.50] Complete o seguinte programa que lê uma cadeia de caracteres e a apresenta invertida no ecrã. O programa escreve três linhas, através de três chamadas à função `printf`, a última das quais mostra a cadeia invertida.

```
#include ____________________________
#include <string.h>

void inverteString(char str[]) {
    
}

_____________ main() {
  char s[100];

  printf("Insira a string: ");
  scanf( _________________ , ____________________ );
  inverteString( _________________ );
  printf(_____________________________, strlen(s) );
  printf(_____________________________, sizeof(s) );
  printf(_____________________________, ____________________ );
  _____________________________ ;
}
```

b) [1.00] Dado o programa anterior e assumindo que foi efectuada a leitura do vector de caracteres "XPTO", qual será o resultado da execução do seu programa (o que irá surgir no ecrã).

Q-8 [3.00 val.] Programe o seguinte conjunto de funções onde:

- `bit0a0` coloca a 0 o bit 0 do carácter dado como argumento (`c`)
- `bit7a1` coloca a 1 o bit 7 do carácter dado como argumento (`c`)
- `bitNa1` coloca a 1 o bit `n` do carácter dado como argumento (`c`) caso `n` seja um número positivo e menor do que o tamanho de um inteiro em bits. Caso `n` não cumpra a condições a função deve retornar -1.
  Sugestão: use deslocamentos (shifts) para definir o bit a ligar
- `pot2` calcula 2^n para `n` dado como argumento

```
unsigned char bit0a0(unsigned char c) {
  return c ____________________________;
}
unsigned char bit7a1(unsigned char c) {
  return c ____________________________;
}
int bitNa1(unsigned int c, int n) {
  if (_____________________________________)
     return -1;
  else
     return c ______ ( _____________________ ) ;
}
int pot2(unsigned int n) {
  return bitNa1(__________, _________);
}
```

Q-9 [3.00 val.] Dadas a seguinte declaração e afectações:

```
int x[2];
x[0] = 0; x[1] = 2;
```

Indique quais são os valores retornados pela avaliação de cada uma das seguintes expressões:

`~~x[1]`	_____	`*(x + 1)`	_____
`x[0] == x[1]`	_____	`x[1]++`	_____
`!x[1]`	_____	`x[1] = x[0]`	_____
`!x[0] && x[1]`	_____	`&(x[1]) - &(x[0])`	_____
`*x + 1`	_____	`sizeof(x)/sizeof(*x)`	_____

Q-10 [3.00 val.] **Para ter a cotação máxima nesta pergunta deve usar apontadores e desreferenciação**
Programe a função:

```
int converter_decimal(char s[], unsigned int b );
```

que dada uma cadeia de caracteres `s` com a representação de um número na base `b`, também dada como argumento, converte esse número para decimal. Caso o número não seja válido na base indicada a função deve retornar -1.
Assuma que $2 \leq b \leq 16$ e que o resultado cabe num inteiro.