Arquitectura de Computadores

Licenciatura em Engenharia Informática

1º teste (A) – 2006/05/07 – Duração: 2h00m

Nome:_____

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.

Resumo da classificação:

Q-1 [0.50] = _____	Q-4 a) [0.50] = _____	**Teórica [12.00]** = _____
Q-2 a) [0.75] = _____	Q-4 b) [0.50] = _____	
Q-2 b) [0.25] = _____	Q-4 c) [0.50] = _____	Q-10 a) [1.00] = _____
Q-2 c) [0.75] = _____	Q-5 [0.75] = _____	Q-10 b) [3.00] = _____
Q-2 d) [0.25] = _____	Q-6 [1.50] = _____	Q-11 a) [1.00] = _____
Q-2 e) [0.25] = _____	Q-7 [0.50] = _____	Q-11 b) [3.00] = _____
Q-3 f) [0.75] = _____	Q-8 a) [0.50] = _____	**Prática [8.00]** = _____
Q-3 a) [1.25] = _____	Q-8 b) [1.00] = _____	
Q-3 b) [0.50] = _____	Q-9 [1.00] = _____	**Total [20.00]** = _____

Q-1 [0.50 val.] Diga qual é o papel do nível ISA (Instruction Set Architecture) na arquitectura de um computador e indique que ISA é que se tem estudado na disciplina.

Q-2 Considere uma máquina máquina Intel que utiliza o ISA que indicou na questão anterior.

a) [0.75 val.] Apesar desta máquina operar a 32 bits, pode-se aceder a registos de tamanho inferior. Estes são na realidade subconjuntos dos registos de carácter geral, como o EAX. Diga o porquê da necessidade desta arquitectura ter registos de tamanhos inferiores a 32 bits?

b) [0.25 val.] No caso particular do registo EAX que nomes têm esses registos e qual é o seu respectivo tamanho (em bits)?

c) [0.75 val.] Nesta máquina a memória é endereçada ao byte, a 16 bits ou a 32 bits? O que é que isso significa?

d) [0.25 val.] Que convenção é utilizada para ordenar os bytes dentro de uma palavra (*word*) de memória?

e) [0.25 val.] Coloque a seguinte palavra de 32 bits (FF FE C3 0F) no endereço 1001 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:	01	_____
00001003:	A3	_____
00001004:	01	_____

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

Q-3 Considere o seguinte fragmento de um programa em C.

```
int f(int a, int b) {
  int i, x = 0;
  for (i = 0; i < 10; i++) {
    x = x + g(a-b)
    b--;
  }
  return x;
}
int g(int a) {
  return a*a - a;
}
int main() {
 printf("O valor e' %d", f(20,10));
  return 0;
}
```

a) [1.25 val.] Faça um esquema das frames de activação das funções f e g.

b) [0.50 val.] Indique nos esquemas acima para onde deve apontar o registo EBP. Qual é a utilidade deste registo?

Q-4 Considere o seguinte programa em NASM/Intel (IA-32):

```
mov eax, 1000
add eax,  1000
mul dword [1000]
```

e o seguinte estado: `Overflow Flag (OF):` 1 `Carry Flag (CF):` 0 `Zero Flag (ZF):` 1

a) [0.5 val.] Qual é o valor das flags após a primeira instrução. `OF:` _____ `CF:` _____ `ZF:` _____

b) [0.5 val.] Qual é o valor das flags após a segunda instrução. `OF:` _____ `CF:` _____ `ZF:` _____

c) [0.5 val.] Explique o porquê de, na terceira instrução, se ter de anotar a quantidade de informação a ler de memória, quando não necessário fazê-lo nas duas primeiras.

Q-5 [0.75 val.] Escreva o código NASM/Intel (IA-32) que efectua a chamada à função `raiz` ilustrada no seguinte código C: `x = raiz(5.0)`. O protótipo da função é: `float raiz(float x)`.

Q-6 [1.50 val.] Faça um esquema do ciclo de compilação de um ficheiro escrito na linguagem C até se obter o executável. De seguida explique-o sucintamente.

Q-7 [0.50 val.] Diga quais são e em que consistem os dois factores que determinam do tempo de acesso a um bloco em disco.

Q-8

a) [0.50 val.] Descreva de que forma está organizado o BUS de sistema e qual a sua utilidade de cada um dos seus componentes.

b) [1.00 val.] Indique os passos que o CPU tem de percorrer para efectuar um pedido de leitura a um dado endereço (memória ou registo de um periférico).

Q-9 [1.00 val.] Em dispositivos de entrada/saída explique em que consiste a programação por espera activa e quais os seus inconvenientes.

Q-10 Implemente um programa em assembly NASM/Intel (IA-32) que calcula a função de Fibonacci do valor colocado na posição de memória com etiqueta *N*. O resultado da função deve ser colocado na posição com etiqueta *fibN*. **As soluções das alíneas devem ser escritas no espaço reservado da página seguinte.**

a) [1.0 val.] Declarar a variável *N*, com tamanho de 32bits e com valor inicial 10 e a variável *fibN*, com tamanho de 32 bits e sem valor inical.

b) [de 1.50 a 3.00 val.] Implementação do cálculo da função. **Implemente apenas uma das opções, indique a sua escolha com uma cruz.**

- **[1.50 val.]** Implementação iterativa no programa principal, sem recorrer a uma subrotina.
- **[2.00 val.]** Implementação iterativa usando uma subrotina. Deve implementar o programa principal que chama a rotina, passando o argumento *N* por pilha, e guardar o resultado, retornado no registo EAX, em *fibN*.
  O algoritmo iterativo é o seguinte:

```
int fib(int n) {
  int pen = 0, ult = 1, res = 1, i;
  for (i = 1; i < n; i++) {
    res = pen + ult;
    pen = ult;
    ult = res;
  }
  return res;
}
```

- **[2.50 val.]** Implementação recursiva, recorrendo ao ESP como base para os argumentos. Deve implementar o programa principal que chama a rotina, passando o argumento *N* por pilha, e guardar o resultado, retornado no registo EAX, em *fibN*.
- **[3.00 val.]** Implementação recursiva, recorrendo ao EBP como base para os argumentos. Deve implementar o programa principal que chama a rotina, passando o argumento *N* por pilha, e guardar o resultado, retornado no registo EAX, em *fibN*.
  O algoritmo recursivo é o seguinte:

```
int fib(int n) {
  if (n <= 1) return 1;
  else return fib(n-1) + fib(n-2);
}
```

ESCREVA AQUI A SOLUÇÃO. Não se esqueça do código para terminar o programa:

```
mov ax, SYS_EXIT
mov bx, 0
int LINUX_SYSCALL
```

```
SYS_EXIT equ 1
LINUX_SYSCALL equ 80H
global _start
section .data
N:
section .bss
fibN:
section .text
_start:
```

Q-11 Suponha que tem à sua disposição as seguintes funções que operam sobre strings:
`int stringLength(char* src)`: Calcula o tamanho da string dada.
`char* stringInvert(char* src)`: Retorna uma string cujo conteúdo é a inversão da string dada.
`void stringInvert2(char* dest, char* src)`: Faz o mesmo processamento que a função anterior, mas a string a conter o resultado é passada por referência. **Nota:** a função pressupõe que a memória para a string destino foi reservada antes da chamada.

a) [1.00 val.] Implemente a função `main` de um programa em C que dada uma string em `argv[1]` calcula e imprime o seu tamanho e a sua string inversa. Deve usar as funções `stringLength` e `stringInvert` acima definidas. **Nota:** o programa só recebe o argumento que contem a string a ser processada.

```
int main(int argc, char* argv[]) {
```

```
}
```

b) [de 1.50 a 3.00 val.] Implementação da rotina para o cálculo da função. **Implemente apenas uma das opções, indique a sua escolha com uma cruz.**

- **[1.50 val.]** Implemente a função `stringLength`.
- **[2.00 val.]** Implemente a função `stringInvert2`. **Não se esqueça:** a função pressupõe que a memória para a string destino foi reservada antes da chamada.
- **[2.50 val.]** Implemente a função `stringInvert`. **Não pode usar variáveis globais.**
- **[3.00 val.]** Implemente a função `stringInvert` usando apontadores para percorrer as strings. **Não pode usar variáveis globais.**