Tecnologias e Arquitecturas de Computadores

Licenciatura em Engenharia Informática
Licenciatura em Engenharia Informática – Pós Laboral
Licenciatura em Engenharia Informática – Curso Europeu

Exame – **Época de Recurso**

Duração Total Exame (T + P) : 2h:30m / **Sem consulta** 6 de Julho de 2015

Parte Teórica

1 O tipo mais comum de discos rígidos é o disco *Winchester* composto por uma unidade selada com um conjunto de pratos sobrepostos, dentro de uma caixa de metal com uma pequena separação entre eles. Considere um disco deste tipo com quatro pratos de dupla face, com dez mil cilindros, dezasseis sectores por cada pista, onde cada sector armazena 512 bytes em cada pista. Calcule:

a) A quantidade de bytes armazenada em cada pista. ***(0,5 Val)***
b) A quantidade de bytes armazenada em cada cilindro. ***(0,5 Val)***
c) A quantidade de bytes armazenada em cada prato. ***(1 Val)***
d) A capacidade do disco em bytes. ***(1 Val)***

2 Os Processadores apresentam barramentos internos e barramentos externos. Dê exemplos de barramentos destes dois tipos, indicando a função de cada um deles.

(2 Val)

3 Faça uma descrição da arquitectura de uma célula de memória dinâmica DRAM e descreva o mecanismo de escrita de informação.

(2 Val)

Parte Prática

1. Realize um programa em Assembly que permita calcular potências. O programa deverá recorrer a 3 vetores: BASE (vetor de bytes, onde constam os números que servirão de base das potências a calcular), EXPOENTE (vetor de bytes, onde constam os números que servirão de expoente das potências a calcular), RESULTADO (vetor onde se guardam os resultados do cálculo dos valores de BASE pelos valores em EXPOENTE na mesma posição). Todos os vetores têm o mesmo número de elementos, definido numa variável do tipo byte (NELEM). ***(2,5 Val.)***

   Exemplo:

   NELEM 5
   BASE 2, 3, 5, 10, 12
   EXPOENTE 10, 2, 3, 4, 2,
   RESULTADO 1024, 9, 125,10000, 144

2. Dado o programa abaixo, responda às seguintes questões: ***(2,5 Val.)***

   a) As variáveis var1, var2 e var3 ocupam cada uma 1 byte.
   b) A variável var4 ocupa 10 double words.
   c) As instruções das linhas 14 e 15 poderiam ser substituídas por uma única: `mov es,0b800h.`
   d) No contexto deste programa a instrução da linha 18 é desnecessária.
   e) A instrução "mul bl" (linha 21) multiplica o valor de AX por 160.
   f) As instruções compreendidas entre as linhas 19 e 22 colocam em DI a posição correspondente à linha do ecrã a partir da qual se desencadeiam ações no programa.
   g) A instrução da linha 28 coloca em BX a posição correspondente à linha e coluna do ecrã a partir da qual se desencadeiam ações no programa.
   h) O programa tem três ciclos um iniciado na linha 31, outro na linha 34 e outro na linha 41.
   i) O ciclo iniciado na linha 33 é executado 10 vezes.
   j) O ciclo compreendido entre as linhas 34 e 38 é executado o número de vezes presentes em var3.
   k) O objectivo das instruções "sub di,2" e "add si,2" (linhas 36 e 37) é o de recuar duas posições no ecrã e avançar duas posições no vetor var4, respetivamente.
   l) A instrução da linha 31 serve apenas para guardar o valor de cx, sendo esse valor reposto na linha 49.
   m) O ciclo compreendido entre as linhas 31 e 50 é realizado var3 vezes.
   n) O ciclo compreendido entre as linhas 34 e 38 é realizado 10 vezes, copiando 18 caracteres e respetivo atributo, a partir da posição bx na memória de vídeo.
   o) O ciclo compreendido entre as linhas 34 e 38 é realizado 10 vezes, copiando 10 caracteres e respetivo atributo, a partir de 18 endereços à frente de bx na memória de vídeo.
   p) A linha 48 faz avançar o índice correspondente à posição na memória de vídeo duas linhas no monitor.
   q) O programa troca a posição de 10 caracteres do ecrã, estando o 1º situado a partir da linha dada por var1 e coluna dada por var2.

```
1:     .8086
2:     .model small
3:     .stack 2048

4:     DATA_HERE   SEGMENT
5:           var1 db 6
6:           var2 db 12
7:           var3 db 14
8:           var4 dw 10 dup(0000h)
9:     DATA_HERE   ENDS

10:    CODE_HERE   SEGMENT
11:          ASSUME CS:CODE_HERE, DS:DATA_HERE
12:    START:mov  ax, DATA_HERE
13:          mov  ds, ax
14:          mov  bx,0b800h
15:          mov  es,bx
16:          xor  di,di
17:          xor  si,si
18:          xor  ax,ax
19:          mov  al,var1
20:          mov  bl,160
21:          mul  bl
22:          mov  di,ax
23:          xor  ax,ax
24:          mov  al,var2
25:          mov  bl,2
26:          mul  bl
27:          add  ax,di
28:          mov  bx,ax
29:          xor  ch,ch
30:          mov  cl,var3
31:    pa:   mov  dx,cx
32:          mov  cx,10
33:    pb:   mov  di,18
34:    pc:   mov  ax,es:[bx+di]
35:          mov  var4[si],ax
36:          sub  di,2
37:          add  si,2
38:          loop pc
39:          mov  cx,10
40:          xor  si,si
41:    pe:   mov  ax,var4[si]
42:          mov  es:[bx],ax
43:          add  bx,2
44:          add  si,2
45:          loop pe
46:          sub  bx,20
47:          mov  si,0
48:          add  bx,160
49:          mov  cx,dx
50:          loop pa
51: acabou: mov  ah,4ch
52:          int  21h
53:    CODE_HERE  ENDS
54:    END START
```

BOA SORTE! ☺