Tecnologias e Arquitecturas de Computadores

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

Exame – **Época Normal**

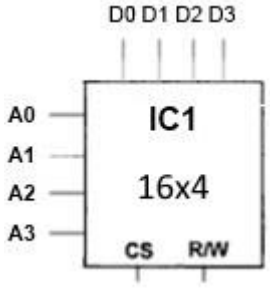
Duração Total Exame (T + P) : 2h:30m / **Sem consulta** 17 de Junho de 2015

Parte Teórica

1 Considere uma máquina com um espaço de endereçamento de 22 bits, uma cache de 32KB, mapeamento directo e blocos de 1 Byte.

a) Quantos bits são necessários para a *tag*? Justifique. ***(2 Val)***
b) Qual a capacidade total desta cache, contando com os bits da *tag* mais os *valid bits*? ***(1 Val)***

2 Considere o circuito integrado de memória RAM da figura, onde A_3, A_2, A_1, A_0 representam linhas de endereço, D_3, D_2, D_1, D_0 representam linhas de dados, R/W representa a linha de leitura/escrita e CS a linha de *Chip Selection*.



Faça um esboço associando múltiplos circuitos integrados iguais de forma a obter uma memória RAM com 32 endereços com 8 bits cada. Deverá ser indicada explicitamente a linha de CS (*Chip Selection*) da memória resultante. ***(2 Val)***

3 O Pipeline é uma tecnologia que permite melhorar substancialmente a performance da Unidade Central de Processamento. Faça uma descrição desta tecnologia fazendo referência à forma como poderá estar adequada a cada uma das arquitecturas RISC e CISC, respectivamente. ***(2 Val)***

Parte Prática

1. Realize um programa em Assembly que permita implementar multiplicações e divisões através de somas e subtracções, respectivamente. O programa deverá recorrer a 4 vetores: OPERANDO1 (vetor de bytes, onde constam os primeiros operandos das operações), OPERANDO2 (vetor de bytes, onde constam os segundos operandos das operações), MULTIPLIC (vetor onde se guardam os resultados da multiplicação dos valores de operando1 pelos valores em operando2 na mesma posição) e QUOCDIV (vetor onde constam os quocientes da divisão dos valores de operando1 pelos valores em operando2 na mesma posição). Todos os vetores têm o mesmo número de elementos, definido numa variável do tipo byte (NELEM). Deve ser assumido que os valores de OPERAND1 são sempre superiores aos valores na mesma posição de OPERAND2, pelo que não é necessário proceder a esta verificação. Soluções que não substituam a multiplicação por somas e a obtenção do quociente da divisão por subtracções não serão consideradas (por exemplo, 4*3=12 deverá ser realizado através de 12=4+4+4 e 7/2=3 (quociente), uma vez que é possível subtrair 3 vezes o valor 2 a 7).

   Exemplo:
   NELEM 10
   OPERAND1 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
   OPERAND2 1, 2, 3, 4, 5, 6, 7, 8 , 9, 10
   MULTIPLIC 10, 22, 36, 52, 70, 90, 112, 136, 162, 190
   QUOCDIV 10, 5, 4, 3, 2, 2, 2, 2, 2, 1 ***(2,5 Val.)***

2. Dado o programa abaixo, assinale cada uma das afirmações como verdadeiro ou falso: ***(2,5 Val.)***

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

4: dseg         segment para public 'data'
5:   posicao1        db      5,'B','L','P','N','R'
6:   posicao2        db      4,'A','E','O','U'
7:   posicao3        db      6,'R','M','N','S','R','L'
8: dseg         ends

9: cseg         segment para public 'code'
10:  assume  cs:cseg, ds:dseg

11: Main         Proc
12:  mov     ax,dseg
13:  mov     ds,ax

14:  mov     bx,0B800h
15:  mov     es,bx
```

```
16:  xor     bx,bx
17:  lea     si,posicao1
18:  mov     cl,[si]
19:  xor     ch,ch
20:  inc     si
21: ciclo1:
22:  mov     al,[si]
23:  inc     si
24:  push    si
25:  push    cx

26:  lea     si,posicao2
27:  mov     cl,[si]
28:  xor     ch,ch
29:  inc     si
30: ciclo2:
31:  mov     ah,[si]
32:  inc     si
33:  push    si
34:  push    cx

35:  lea     si,posicao3
36:  mov     cl,[si]
37:  xor     ch,ch
38:  inc     si
39: ciclo3:
40:  mov     dl,[si]
41:  inc     si

42:  mov     es:[bx],al
43:  add     bx,2
44:  mov     es:[bx],ah
45:  add     bx,2
46:  mov     es:[bx],dl
47:  add     bx,2
48:  mov     byte ptr es:[bx],','
49:  add     bx,2
50:  mov     byte ptr es:[bx],32 ; Caracter ESPAÇO
51:  add     bx,2
52:  loop    ciclo3

53:  pop     cx
54:  pop     si
55: loop    ciclo2

56:  pop     cx
57:  pop     si
58:  loop    ciclo1

59:sai: mov     ah,4ch
60:  int     21h
61:  Main    Endp
62: cseg          ends
63: end           Main
```

a) A variável posicao1 ocupa 5 bytes.

b) Para o objetivo do programa é necessário atribuir a ES o valor 0B800h.

c) A instrução da linha 22 poderia ser substituída pela instrução `mov  al,posicao1[si]`.

d) A intenção das instruções das linhas 18 e 19 é a de colocar CX com o número de letras de `posicao1`.

e) Da 1ª vez que é executada, a instrução da linha 24 guarda na pilha o 2º elemento de posicao1.

f) Na linha 31 é possível substituir `mov ah,[si]` por `mov ah,es:[si]` que o resultado seria o mesmo, pois são instruções equivalentes.

g) De cada vez que é executada, a instrução da linha 33 guarda na pilha cada um dos elementos de `posicao2`.

h) De cada vez que é executada, a instrução da linha 33 guarda na pilha posições/índices dos elementos de `posicao2`.

i) A instrução da linha 26 destrói o anterior valor de SI, não sendo contemplada no programa nenhuma forma de o recuperar.

j) As instruções das linhas 42, 44 e 46 escrevem no ecrã, de forma consecutiva, na posição dada por BX cada uma das letras de cada um dos vetores na seguinte ordem: BAR, LEM,PON,...

k) As instruções das linhas 42, 44 e 46 escrevem no ecrã, de forma consecutiva, na posição dada por BX cada uma das letras de cada um dos vetores na seguinte ordem: BAR, BAM, BAN,...

l) O objetivo da linha 48 e 50 é o de separar cada uma das palavras geradas por “,” e “espaço”.

BOA SORTE! ☺