Curso: LMCC **Exame 1ª Chamada** 14/Jun/06
Disciplina: Arquitectura de Computadores **Duração:** 2h30m

Nota: Apresente sempre o raciocínio ou os cálculos que efectuar; o não cumprimento desta regra equivale à não resolução do exercício. Use o verso das folhas do enunciado do exame como papel de rascunho.

1. **(A)** Considere o seguinte logotipo antigo da UM (o fundo é azul escuro). Indique, justificando, qual o formato de representação de informação mais adequado para o codificar em binário (GIF, JPEG, …), sabendo que se pretende mostrar esta imagem na Internet.

2. **(A)** Considere a operação de premir na tecla "**A**" no seu computador de secretária, com ele ligado e a funcionar normalmente. Identifique e caracterize sumariamente todas os blocos funcionais do computador por onde passam os bits que representam a letra "**A**", desde que a letra sai do teclado até ser desenhada no monitor.

3. **(R)** Imagine que foi convidada/o para uma festa de aniversário em casa do/a seu/sua namorado/a, que tem um primo de 11 anos fanático por computadores. Ele sabe que é estudante de informática, e faz-lhe uma pergunta sobre uma sigla que ele leu e não compreendeu, e que é um dos tópicos seguintes (escolha apenas um):

BIOS ** *clock* do CPU ** *cache* L2 ** *chipset* ** monitor WSXGA ** disco IDE 6200rpm ** USB2 ** *flash memory* ** CPU dual-core ** RAM DDR2 400MHz

Apresente a explicação que lhe daria (deve ser sucinta e usar uma linguagem clara e rigorosa).

Nº	**Nome**

4. Considere que o registo `%esp` dum processador IA-32 (*little endian*) contém o valor `0x8a62004`, e que as células de memória a partir de `0x8a62000` contêm os seguintes valores (representados em vários sistemas de numeração e separados por vírgulas):

`33, 107`$_{11}$`, 63, 377`$_{8}$`, 0x4d, 96, 164`$_{8}$`, 1201`$_{4}$`, 0x6d, 0`$_{6}$`, 100`$_{2}$`, 10, 2`$_{7}$`, 86`$_{9}$`, 12, 0x2a`

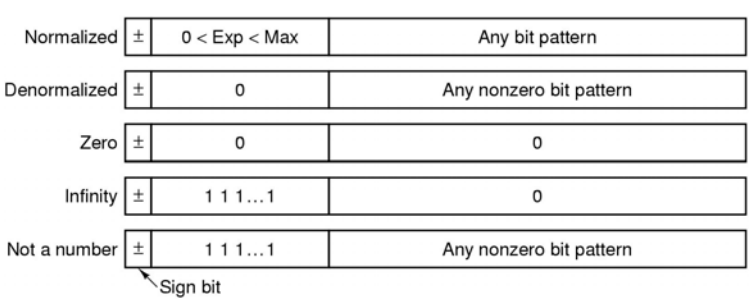
a) **(A)** **Represente** em numeração binária o conteúdo do registo `%esp` (apontador para o topo da pilha), e **indique, justificando**, se o valor desse endereço é maior ou menor que 10^8.

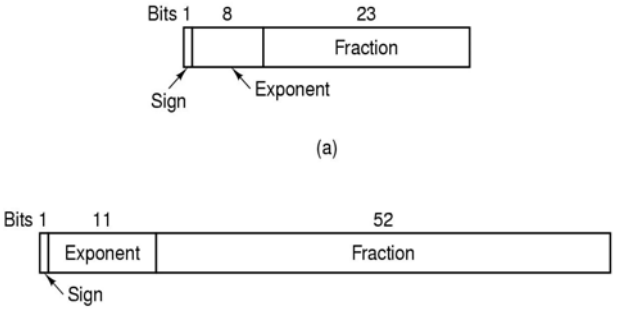
b) Considere que o CPU tinha acabado de executar uma instrução de `push` de um registo, quando se fez essa análise das células de memória. **Indique**, **justificando**, o valor em decimal, que foi colocado no topo da pilha, consoante se considere que a variável que se encontrava no registo **(i)** era do tipo `short int`, ou **(ii)** era do tipo `float`.
Nota: o resultado pode ser apresentado sob a forma de uma expressão aritmética na base 10.

(i) **(A/R)**

(ii) **(A/R)**

Notas de apoio (norma IEEE 754)





Valor decimal de um fp em binário:
precisão simples, normalizado: $V = (-1)^S * (1.F) * 2^{E-127}$
precisão simples, desnormalizado: $V = (-1)^S * (0.F) * 2^{-126}$

Nº	Nome

c) **(B)** A Unidade Astronómica (**UA**) é uma métrica para distâncias, e tem o valor da distância média entre a Terra e o Sol. O valor de UA que é normalmente utilizado é de `1.49597870691x10`^{11}`m`. **Indique, justificando**, se um valor com esta precisão pode ser representado numa variável do tipo `float`.

5. Considere o seguinte excerto de uma função em C (binarização de uma imagem):

```
int *bf;              // array dos pixeis da imagem
int Xdim; int Ydim;          // dimensões da imagem
...
int getXY(int x, int y) {
    return( bf[x + y*Xdim] );
}

void Imagem_bin2(){
  int tot, media, pixeis_mask,;
  int i, j, i1,j1;
  int sob=15;                // janela para o cálculo dos desvios
...
  for (i1=-sob; i1<=sob; i1++)
    for (j1=-sob; j1<=sob; j1++)
       tot += getXY(i+i1,j+j1);
  media = tot / pixeis_mask
...
```

Nota: esta parte sombreada é para a alínea f)

Parte desta função compilada (apenas) com `-O2` produziu o seguinte código *assembly*: De notar que, sendo `Xdim` e `bf` variáveis globais, o código *assembly* gerado não pode ainda indicar de modo explícito o endereço absoluto dessas variáveis em memória, pelo que os valores "`_Xdim`" e "`_bf`" irão ser posteriormente substituídos por um endereço (número positivo).

```
_getXY:
       pushl %ebp                ;
       movl  %esp, %ebp          ;
       movl  _Xdim, %eax         ;
       movl  12(%ebp), %ecx      ;
       imull %ecx, %eax          ;
       movl  8(%ebp), %edx       ;
       addl  %edx, %eax          ;
       movl  _bf, %edx           ;
       movl  (%edx,%eax,4), %eax ;
       popl  %ebp                ;
       ret                       ;
(...)
```

Nº	Nome

a) **(A)** A função `getXY` recebe 2 argumentos e devolve um valor inteiro. **Descreva** como é que é feita a passagem de argumentos e a devolução de valores escalares no código gerado por um compilador, a partir do C para um processador IA-32.

b) **(A)** **Introduza comentários** ao código em *assembly* apresentado, de modo a ficar claro a maneira como é feita a tradução do código C para *assembly*, e **identificando** explicitamente o corpo da função `getXY`.
Nota: coloque os comentários diante do código, na folha anterior.

c) **(A/R)** Considere neste código a instrução em *assembly* `movl 12(%ebp),%ecx`. **Identifique e caracterize** sumariamente o conjunto de comandos emitidos pela Unidade de Controlo do CPU (e a quem são destinados) para que esta instrução possa ser devidamente executada (incluindo a sua busca da memória, e considerando que ela vem toda de uma só vez da memória).
Sugestão: imagine que está a fazer o papel de UC numa peça de teatro que mostre o funcionamento de um CPU na execução de instruções.

d) **(R)** **Represente** a estrutura da *stack frame* desta função (`getXY`) quando estiver a executar a multiplicação, com indicação da **dimensão_e_descrição** de cada um dos valores nela presente.

Nº	**Nome**

e) **(R/B)** Uma das opções de optimização de desempenho mais óbvias neste exemplo é a substituição da invocação de uma função no interior do ciclo, pelo código da própria função (*function inlining*). Com esta operação (em baixo, a zona sombreada do código inicial, re-escrita) reduzem-se os acessos à memória.

```
for (i1=-sob; i1<=sob; i1++)
   for (j1=-sob; j1<=sob; j1++)
     tot += bf[(i+i1) + (j+j1)*Xdim];
media = tot / pixeis_mask
```

Identifique e **contabilize <u>todos</u>** os acessos à memória que são evitados com esta substituição.

f) **(A)** Com a optimização referida na alínea anterior, um excerto do código *assembly* contendo o interior do ciclo vem a seguir representado:

```
(...)
  movl    $-15, %edx
  movl    -16(%ebp), %esi
  movl    -24(%ebp), %eax
  leal    (%eax,%esi), %ecx
L48:
  movl    -20(%ebp), %eax
  addl    %edx, %eax
  imull   %edi, %eax
  leal    (%eax,%ecx), %eax
  movl    -28(%ebp), %esi
  addl    (%esi,%eax,4), %ebx
  incl    %edx
  cmpl    $15, %edx
  jle     L48
(...)
```

Identifique neste código *assembly* a implementação das expressões de controlo do ciclo `for` interior.

g) Considerando a execução da instrução a sombreado (após a sua descodificação), e que:

- os registos `%eax`, `%ebx`, `%esi`, `%ebp` e `%eip` contêm respectivamente os seguintes valores em hexadecimal: `0x2`, `0x202`, `0x8401228`, `0x8f04802`, `0x8c12004`
- o conteúdo de cada célula de memória é igual ao *byte* menos significativo do seu endereço;

Mostre:

Nº	**Nome**

(i) **(A)** O **tráfego** que circula no barramento de endereços, em hexadecimal.

(ii) **(R/B)** A localização e conteúdo das células que vão conter a instrução (com 5 *bytes*) que invoca esta função (**explicite** o raciocínio seguido).

(iii) **(B)** As alterações que deveriam existir em relação a esta instrução em *assembly* para que ela pudesse ser executada por uma arquitectura típica RISC (use a sintaxe do *assembler* do IA-32 da Gnu).

6. **(B)** Na medição de tempos de execução de programas para avaliação de desempenho, usam-se normalmente contadores de intervalos ou contadores de ciclos de *clock*. **Caracterize** sumariamente cada um destes, **mostrando** os prós e contras de cada.

Nº	**Nome**