PUC-Rio – Software Básico – INF1612
Prova 1 – 3/10/02

1. Faça um programa assembly que tenha a seguinte declaração de matriz 5x4 como dado global:

```
        .data
chrs    byte 'a', 'A', 'D', ... ; uma lista de 20 caracteres
```

   O programa deve percorrer a matriz e contar o número de letras maiúsculas em cada linha, imprimindo os índices das linhas que contiverem pelo menos duas letras maiúsculas.

   Para imprimir um número, coloque-o em `eax` e chame a função *printnum*, como nos laboratórios. Não se preocupe com a definição de *printnum*.

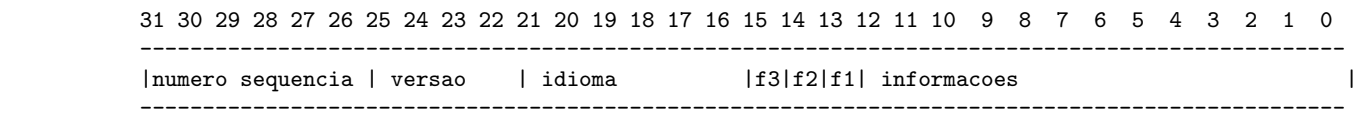
   obs: O código ASCII de 'A' é 65.

2. Considere o programa C a seguir:

```
#include <stdio.h>

void dump (void *p, int n) {
  unsigned char *p1 = p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}

struct sbobo{
  short s;
  int i;
  char c;
  };

int main (void) {
  struct sbobo bobo[2] = { {-3, 5, 'a'}, {5, -2500, 'C'}};
  dump (bobo, 2*sizeof(struct sbobo));
  return 0;
}
```

   Diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos. Suponha que a máquina de execução é *little-endian*.

3. Considere um protocolo que use o seguinte formato para os seus pacotes:



   Escreva funções, em C ou em assembly, para realizar as seguintes manipulações em valores com esse formato (todas as funções a seguir recebem como parâmetro um valor com esse formato):

   (a) alterar o valor do campo *versão* para que ele passe a conter o valor recebido como parâmetro (a função retorna o novo valor do pacote):

   ```
   int preencheVersao ( int pacote, short versao);
   ```

   (b) fazer o campo de nome 'f3' ficar com valor 1 (a função retorna o novo valor do pacote):

   ```
   int setaPrioridade (int pacote);
   ```

   (c) retornar um valor de 32 bits contendo o valor 'informacoes' nos bits de 0 a 12 e 0 nos demais bits.

   ```
   int informacoes (int pacote);
   ```

   Caso você resolva escrever em assembly, suas funções podem supor que o parâmetro `pacote` está inicialmente em `eax` e o parâmetro `versao`, no caso da função `prrencheVersao`, em ebx.

4. A instrução CDQ (convert doubleword to quadword) dobra o tamanho do operando contido no registrador `eax`, fazendo extensão de sinal. O resultado é gravado no par `edx:eax` (palavra mais significativa em edx). Ou seja, a instrução copia o sinal (bit 31) do valor contido em `eax` para todos os bits do registrador edx.

   Suponha que um número inteiro (em complemento a 2) foi colocado no registrador `eax`. Considere o seguinte trecho de programa assembly.

   ```
   cdq
   xor eax, edx
   sub eax, edx
   ```

   O que ele faz? (não é para simplesmente descrever o que cada instrução faz, e sim relacionar o conteúdo final de `eax` com o inicial)

   SUGESTÃO: Experimente iniciar com um valor inteiro positivo em `eax` e depois com um negativo. Veja o que acontece em cada caso. (trabalhe com menos bits, por exemplo estendendo de 4 para 8, para fazer suas experimentações!)