PUC-Rio – Software Básico – INF1612
Prova 1 – 27/04/06

1. (2,5 pontos) Considere o programa C a seguir:

```
#include <stdio.h>
void dump (void *p, int n) {
  unsigned char *p1 = (unsigned char *) p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}
struct sprova {
  int questao; short valor; unsigned char n;
};
struct saluno {
  struct sprova *p;
  int nota;
};
int main (void) {
  struct sprova provas[] = {
    {1, 2, 9}, {1, 3, -6}
  };
  struct saluno alunos[] = {
    {&provas[0], 5}, {&provas[1], -32}
  };
  dump ((void *)provas, sizeof(struct sprova)*2); /* dump 1 */
  dump ((void *)alunos, sizeof(struct saluno)*2); /* dump 2 */
  return 0;
}
```

   Considerando que `provas` será alocado na posição de memória 0xbffff7b8 e `alunos` na posição de memória 0xbffff7a8, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (ATENÇÃO: valores sem contas e explicações NÃO valem ponto!!!). Suponha que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32.

2. Traduza as funções `copia` e `dup` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Use registradores para abrigar as variáveis locais que aparecem nessas funções. Comente seu código.

   (Não se preocupe se você não souber o que as funções fazem, apenas traduza-as literalmente! Também não se preocupe com a implementação de alocaVetor!)

   (a) (2,5 pontos)

```
int * copia (int *d, int *o, int n) {
  int i;
  for (i=0;i<n;i++)
    if (o[i]&1 != 0)
      d[i]=o[i];
    else
      d[i]=0;
  return d;
}
```

(b) (2,5 pontos)

```
int *alocaVetor (int n);

int *dup (int *a, int n) {
  int *p = alocaVetor(n);
  p = copia (p, a, n);
  return p;
}
```

3. (2,5 pontos) No formato UTF-8, usado para representar caracteres UNICODE, caracteres têm tamanho variável. Um caracter tem tamanho mínimo de 8 bits, porém, se seu código necessitar de mais espaço para ser representado, o formato UTF-8 pode utilizar mais de um byte para representá-lo (até um máximo de quatro bytes). Abaixo é apresentada a faixa dos números e o número de bytes necessários para representar cada faixa do código UNICODE (onde x é o valor real do bit):

Código UNICODE	Representação UTF-8
U+0000 - U+007F	0xxxxxxx
U+0080 - U+07FF	110xxxxx 10xxxxxx
U+0800 - U+FFFF	1110xxxx 10xxxxxx 10xxxxxx
U+10000 - U+10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Escreva, em C ou em assembly IA-32 usado nos laboratórios, uma função *utf8strlen*, com o protótipo:

```
int utf8strlen (char * utf8);
```

Essa função recebe um string com uma sequência de caracteres UTF8 e retorna seu comprimento *em caracteres*, ou seja, o número de caracteres que estão representados no string. O final do string dado como entrada para a função é marcado por um byte nulo (`0x00`).