PUC-Rio – Software Básico – INF1612
Prova 1 – 9/10/03

1. (2,5 pontos) Considerando a seguinte declaração de variáveis:

```
struct {
  int nota;
  int idade;
  char sit;
} turma[10];
```

suponha que a estrutura `turma` é um dado global em um programa assembly e construa um trecho desse programa (em assembly) que imprima a menor nota da turma (as notas são números inteiros entre 0 e 10). (Não se preocupe com a declaração de `turma` no programa assembly, apenas suponha que o nome `turma` se refere ao início dessa estrutura de dados)

atenção: Você pode chamar a função `printnum`, usada nos laboratórios, para imprimir o resultado.

2. (3,0 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[] = {
    {&p[0], 5}, {&p[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 carregado na posição de memória 0x0013F420, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (mostre suas contas aqui ou no rascunho!!). Suponha que a máquina de execução é *little-endian*.

3. (2,5 pontos) O *código BCD*, para representação de inteiros usando números binários, é utilizado em alguns contextos por facilitar a leitura de números em decimal. Ele usa grupos de quatro bits para representar os algarismos de 0 a 9, ou seja de 0000 a 1001. Usando BCD, um byte representa apenas números sem sinal de 0 a 99.

Seguem a seguir alguns exemplos.

- 120 (decimal) em dois bytes

  ```
  BCD:     0000 0001 0010 1000
  binário: 0000 0000 1000 0000
  ```

- 99 (decimal) em dois bytes

  ```
  BCD:     0000 0000 1001 1001
  binário: 0000 0000 0110 0011
  ```

Implemente uma função C ou assembly que converta um número de BCD para binário, conforme o protótipo a seguir:

```
unsigned int bcd2binary (unsigned int BCDvalue);
```

obs: Caso você faça em *assembly*, como ainda não vimos funções, escreva um trecho de código que suponha que inicialmente o número BCD está em `eax` e termine colocando o valor final, binário, também em `eax`.

4. (2,0 pontos) Faça um "chinês"do código a seguir e diga o que ele vai imprimir.

```
_DATA   SEGMENT
value   DB 'Peguei um onibus sei la onde e para onde', 00H
_DATA   ENDS
PUBLIC  _main
_TEXT   SEGMENT
printstr:
  ; imprime o string cujo endereco inicial esta em eax
_main:
       mov ebx, 0
volta: cmp value[ebx], 0
       je ok
       inc ebx
       jmp volta
ok:    dec ebx
       mov eax, 0
L1:    mov cl, BYTE PTR value[eax]
       mov ch, BYTE PTR value[ebx]
       mov BYTE PTR value[ebx], cl
       mov BYTE PTR value[eax], ch
       inc eax
       dec ebx
       cmp eax, ebx
       jl L1
       mov eax, offset value
       call printstr
       ret
_TEXT  ENDS
END
```

5. (0,5 *ponto extra*) A instrução `SBB op1,op2` subtrai a soma de `op2` com o CF (carry flag) de `op1`, gravando o resultado em `op1`. Tipicamente, o estado de CF reflete o resultado de uma operação de subtração anterior (o CF indica um "toma emprestado").

   Suponha que dois números inteiros foram colocados nos registradores `eax` e `ebx`. Considere o seguinte trecho de programa assembly.

   ```
   sub ebx, eax
   sbb ecx, ecx
   and ecx, ebx
   add eax, ecx
   ```

   O que ele faz? (não é para simplesmente descrever o que cada instrução faz, e sim relacionar o conteúdo final de `eax` com os valores iniciais em `eax` e `ebx`).