Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina
Departamento Acadêmico de Eletrônica
Graduação em Engenharia Eletrônica

Prof. Hugo Marcondes (hugo.marcondes@ifsc.edu.br)

Exercícios Linguagem Assembly MIPS

1. Implemente as seguintes chamadas de funções para encapsular as principais chamadas de sistemas que utilizamos no MARS:

```
int get_int();
float get_float();
double get_double();
char get_char();
int get_string(char * str, int max_size);

void print_int(int v);
void print_float(float v);
void print_double(double v);
void print_char(char v);
void print_string(char * str);

void exit();
void exitv(int v);
```

2. Implemente os seguintes procedimentos abaixo descritos em linguagem C, utilizando a convenção de chamada de procedimentos adequada.

(a)
```
// Memcopy
void memcpy(char * src, char * dst, int bytes){
   while(bytes--)
    *dst++ = *src++;
}
```

(b)
```
// Calcula o somatorio de um vetor de inteiros
int sum(int* v, int size) {
  int sum = 0;
  while(size--)
     sum += *v++;
  return sum;
}
```

(c)
```
// Calcula a raiz quadrada (inteiro)
int isqrt(int num) {
    int res = 0;
    int bit = 1 << 30;

    while (bit > num)
        bit >>= 2;

    while (bit != 0) {
        if (num >= res + bit) {
            num -= res + bit;
            res = (res >> 1) + bit;
        }
        else
            res >>= 1;
        bit >>= 2;
    }
    return res;
}
```

(d)
```
// Algoritmo de ordenacao BubbleSort
void bubble(int* v, int size) {
  int i; # t0
  int j; # t1
```

```
        int aux;
        int k = size - 1 ; # t2

        for(i = 0; i < size; i++) {
            for(j = 0; j < k; j++) {
                if(v[j] > v[j+1]) {
                    aux = v[j];
                    v[j] = v[j+1];
                    v[j+1] = aux;
                }
            }
            k--;
        }
    }
```

(e)
```
    void print_vector(int* v, int size) {
        while(size--){
          print_int(*v++);
          print_string(", ");
        }
        print_string("\n");
    }
```

3. Implemente as funções abaixo para a manipulação de matrizes, assim como o programa de teste apresentado abaixo:

```
// Definição da estrutura de Matrix
struct Matriz {
  int linhas;
  int colunas;
  float dados[linhas][colunas];
};

void le_matriz(struct Matriz * dst);
void imprime_matriz(struct Matriz * src);
float determinante(struct Matriz * src);
float soma(struct Matriz * dst, struct Matriz * a, struct Matriz * b);
void oposta(struct Matriz * dst, struct Matriz * src);
void transposta(struct Matriz * dst, struct Matriz * src);


void main(int* v, int size) {

  struct Matriz * a; // 3x3
  struct Matriz * b; // 3x3
  struct Matriz * c; // 3x3

  le_matriz(a);
  le_matriz(b);

  imprime_matriz(a);
  imprime_matriz(b);

  soma(c, a, b);

  imprime_matriz(c);

  oposta(b, b);
  imprime_matriz(b);

  soma(c, a, b);
  imprime_matriz(c);

  transposta(c, a);
  imprime_matriz(c);

  int detA = determinante(a);

  print_int(detA);

}
```