Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação em Sistemas Computacionais

Teste Global de 1ª Época, Inverno de 2016/2017

Nas questões em que não se indiquem explicitamente outras condições, considere as características do ambiente de referência usado na unidade curricular neste semestre.

1. [2,5] Implemente em linguagem C a função `find_bit_sequence` que pesquisa no *array* de 64 *bits* especificado pelo argumento data a primeira ocorrência da sequência de *bits* especificada pelos argumentos `seq` e `seq_len`. A função devolve o índice do primeiro *bit* da sequência (0..63) em `data` ou -1 no caso da sequência de *bits* especificada não ser encontrada.

   ```
   int find_bit_sequence(unsigned long data, unsigned int seq,  size_t seq_len);
   ```

2. [3] Implemente em *assembly* x86-64 a função `find_by_value`, cuja definição em linguagem C se apresenta a seguir.

   ```
   typedef struct data_item { const char *label; short value; } DataItem;

   const char *find_by_value_(DataItem *pitems[], short val) {
     for (size_t i = 0; pitems[i] != NULL; i++)
       if (pitems[i]->value == val)
         return pitems[i]->label;
     return NULL;
   }
   ```

3. [4,5] Considere a função `select_if`, cuja definição em linguagem C se apresenta a seguir.

   ```
   size_t select_if(void *items[], size_t nelems, int (*select)(const void *)) {
     size_t selected;
     void **src, **dst, **last = &items[nelems - 1];
     for (selected = 0, src = dst = items; src <= last; src++)
       if ((*select)(*src) != 0) {
         *dst = *src; selected++; dst++;
       }
     return selected;
   }
   ```

   a) [3] Implemente a função `select_if` em *assembly* x86-64.

   b) [1,5] Escreva, em linguagem C, um programa que utilize a função `select_if` para selecionar de entre os elementos de um *array* de ponteiros para *strings* C, aqueles cujas *strings* contenham pelo menos uma letra maiúscula. No programa deve explicitar a iniciação do *array* de ponteiros, a definição da função de selecção, a chamada à função `select_if`, assim como o código para mostrar as *strings* seleccionadas na consola.

4. [2,5] Considere o conteúdo dos ficheiro fonte `f1.c` e `f2.c`.

   ```
   /* f1.c */
   extern size_t TAB_SIZE;
   int h(void);
   const unsigned char x = 0;
   int g(int val) {
       if (TAB_SIZE > val)
           return h();
   }
   int main(int argc, char **argv) {
       return g(argc) + x;
   }
   ```

   ```
   /* f2.c */
   extern int x;
   char *tab[] = {"a", "b", "c"};
   #define TAB_SIZE  (sizeof(tab)/sizeof(*tab))
   #define TAB(z)    (tab[z][0])
   static int main(int argc, char **argv) {
       for (int i = 0; i < argc; ++i)
           x += TAB(i);
   }
   int h() { return main(TAB_SIZE, tab); }
   ```

a) [1,5] Indique o conteúdo das tabelas de símbolos dos ficheiros objecto relocáveis resultantes da compilação de `f1.c e f2.c`. Para cada símbolo, indique o nome, a secção e o respectivo âmbito (local ou global).

b) [1] Identifique, justificando, quais os erros que ocorrem na ligação entre os módulos f1.o e f2.o.

5. [1,5] Considere uma *cache* com uma organização *16-way set associative*, 32 *bytes* de dimensão do bloco e que utiliza 9 *bits* do endereço para seleccionar o *set*. Apresentando os cálculos apropriados, indique qual é a capacidade da *cache* e qual o número de *bits* usados para armazenar a *tag* em cada linha de *cache*, tendo em consideração que os endereços físicos são definidos a 48 *bits*.

6. [3,5] O tipo `Chunk` representa um bloco de dados identificados por uma chave e composto por um número arbitrário de valores do tipo `long`; os valores são armazenados no *array* `values`, sendo o número de elementos deste *array* definido pelo campo `length`. A função `compact_sparse_array` recebe, através dos parâmetros `sppchunk` e `in_length`, um *array* de ponteiros para instâncias de `Chunk` esparsamente preenchido (i.e., tem elementos não definidos cujo valor é `NULL`) e devolve um *array* compacto (através do valor da função e do parâmetro de saída `out_length`) que refere todas as instâncias de `Chunk` referidas pelo *array* esparso. Toda a memória utilizada para construir o *array* compacto deve ser alocada dinamicamente. A função `free_array` liberta toda a memória alocada na construção do *array* devolvido pela função `compact_sparse_array`. Implemente estas duas funções em linguagem C.

   Nota: Quando se pretende usar estruturas com um campo do tipo *array* com dimensão variável, o compilador de C permite declarar o *array* sem especificar a dimensão desde que o *array* seja o último campo da estrutura. Este campo do tipo *array* pode ser referido normalmente no código, contudo não é tido em consideração no cálculo do `sizeof` da estrutura.

```
typedef struct chunk { const char *key; size_t length; long values[]; } Chunk;
Chunk **compact_sparse_array(Chunk **sppchunk, size_t in_length, size_t *out_lenght);
void free_array(Chunk **ppchunk, size_t length);
```

7. [2,5] O seguinte código *Java* implementa parte de uma hierarquia de tipos que realiza operações sobre imagens com diferentes formatos (JPEG, BMP, etc.). Apresente uma versão em linguagem C das classes `Image` e `BMPImage` que permita integrar novos tipos na hierarquia sem alterar o código base. (Tenha em consideração que o método `move` deve invocado com despacho estático enquanto que o método `select` tem que ser invocado com despacho dinâmico.)

   Nota: Na implementação em C apenas é necessário declarar os protótipos das funções correspondentes aos métodos `Image.move` e `BMPImage.select`.

```
public abstract class Image {
  private FILE file;
  public Image(FILE f) { file = f; }
  public abstract boolean select();
  public final void move() { ... }
  public static mvSelectedImgs(Image[] images) {
    for (Image image : images)
      if (image.select())
       image.move();
  }
}
```

```
class JPEGImage extends Image {
  public JPEGImage(FILE *f) { super(f); }
  public boolean select() { ... }
}
class BMPImage extends Image {
  public BMPImage(FILE f) { super(f); }
  public boolean select() { ... }
}
```

Duração: 2 horas e 30 minutos
ISEL, 24 de Janeiro de 2017