Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Teste Global da Época Especial, Verão de 2018/2019

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]] Implemente, em linguagem C, num sistema que não suporte operações em vírgula flutuante, a função `double_cmp_exp`, que compara os expoentes dos *doubles* `d1` e `d2`, devolvendo como resultado da comparação: zero se os expoentes forem iguais, um valor maior do que zero se o expoente de `d1` for maior do que o de `d2` e um valor menor que zero, no caso contrário. Relembra-se a codificação do *double* segundo a norma IEEE 754: s_{63} | $exp_{62..52}$ | $frac_{51..0}$.

   ```
   int double_cmp_exponent(double d1, double d2);
   ```

2. [3] Escreva em linguagem C a função `string_match`, que percorre a string C, recebida em `text`, e procura ocorrências da sequência de caracteres recebida em `string`. As ocorrências são registadas no *array* `match`, cuja dimensão é definida por `match_len`, através do ponteiro para o início de cada sequência encontrada. A função devolve o número de ocorrências.

   ```
   size_t string_match(const char *text, const char *string, const char *match[],
                       size_t match_len);
   ```

3. [3,5] Implemente em *assembly* x86-64 a função `get_val_ptr`, cuja definição em linguagem C se apresenta a seguir.

   ```
   typedef struct data { unsigned id; size_t length; short *vals; } Data;
   typedef struct info { double ref; Data *data[16]; int valid; } Info;
   short *get_val_ptr(Info items[], size_t item_idx, size_t data_idx, size_t val_idx) {
     return items[item_idx].valid && val_idx < items[item_idx].data[data_idx]->length
            ? &(items[item_idx].data[data_idx]->vals[val_idx])
            : NULL;
   }
   ```

4. [3,5] Implemente em *assembly* x86-64 a função `bsearch` (*binary search*), cuja definição em linguagem C se apresenta a seguir.

   ```
   void *bsearch(const void *key, const void *base, size_t nitems, size_t size,
                 int(*compar)(const void *, const void *)) {
     int low = 0, high = nitems - 1, cond;
     while (low <= high) {
       int mid = (low + high) >> 1;
       char *midp = (char *)base + mid * size;
       if ((cond = (*cmp)(key, midp)) < 0)
         high = mid - 1;
       else if (cond > 0)
         low = mid + 1;
       else
         return midp;
     }
     return NULL;
   }
   ```

5. [3,5] Considere o conteúdo dos ficheiros fonte `f.h` , `f1.c` e `f2.c`.

```
/* f.h */

int print(char *str);

enum state {
    STATE0, STATE1
};

enum state state;
```

```
/* f1.c */

#include <ctype.h>
#include "f.h"

char *ccase(char *str) {
  char *ptr = str;
  int c;
  for (; c = *str; *str++ = c)
    c = (state == STATE0)
        ? tolower(c)
        : toupper(c);
  return ptr;
}
```

```
/* f2.c */

#include "f.h"

const char msg[] = "aaAAaaAA";
char *ccase(const char *str);

int main() {

  state = STATE0;
  print(ccase(msg));
  state = STATE1;
  print(ccase("bbbbbb"));
}
```

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

   b. [1] Considere que o código da função `print` se encontra no ficheiro com o nome `libprint.so`, localizado na directoria corrente. Escreva a sequência de comandos para compilar `f1.c` e `f2.c` em separado e por fim gerar o ficheiro objecto executável.

   c. [1] Entre a definição e a declaração de `ccase` existe uma inconsistência não detectável no processo de geração do executável. Diga qual é a inconsistência, porque é que é indetectável e qual é consequência.

6. [1] Considere uma *cache* para endereços físicos de 32 *bit*, com uma organização *8 way set associative*, uma capacidade de 512 KiB e com 1024 *sets*. A presentando os cálculos apropriados, indique quais os *bits* do endereço físico (e.g., A_0 a A_{31}) que são usados para definir o índice do *set*.

7. [3,5] Considere que se pretende converter uma lista de cartões bancários organizada em lista ligada para uma organização em *array* de ponteiros. O tipo `Card` contém os dados de um cartão: número, estado e respectivo titular. O tipo `NodeCard` é um tipo auxiliar usado para construir listas simplesmente ligadas de cartões; o campo `next` indica o elemento seguinte e o campo `card` aponta uma instância de `Card` com os dados do cartão. O tipo `ArrayCard` armazena um *array* de ponteiros para os dados dos cartões acompanhado da respetiva dimensão (`ncard`).

   Programe em linguagem C as funções `list_to_array`, que converte uma lista simplesmente ligada de cartões num array de cartões e `array_to_list` que realiza a operação contrária. Durante o processo de conversão, as a memória utilizada pelas estruturas de dados originais deve ser libertada.

   Nota: Quando se pretende usar estruturas com um campo do tipo *array* com dimensão variável, o compilador de C permite declarar o *array* sem se 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 na determinação de `sizeof` da estrutura.

```
typedef struct {
  unsigned long number: 50;
  unsigned long state: 3;
  unsigned long hlen: 8;
  char holder[];
} Card;

typedef struct node_card { struct node_card *next; Card *card; } NodeCard;

typedef struct { unsigned int ncard; Card *array[]; } ArrayCard;

ArrayCard *list_to_array(NodeCard *list);

NodeCard *array_to_list(ArrayCard *array);
```

Duração: 2 horas e 30 minutos
ISEL, 17 de Julho de 2019