Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação de Sistemas Computacionais
Teste Global da Época Normal, Inverno de 2019/2020

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] Desenhe a ocupação de memória produzida por cada uma das seguintes definições:

   a. `int a = 3;`
   b. `short b = 2;`
   c. `char c[] = "1.1";`
   d. `struct {int x; long y;} d = {10};`
   e. `union {int z; long w;} e = {32};`

   Para cada caso indique, em hexadecimal, o conteúdo de cada posição de memória ocupada pela variável.

2. [2,5] Escreva a função `strndelc`, que elimina até `n` ocorrências do carácter `c` da *string* `str`. Por exemplo, se `char str[] = "Vies-te, vis-te e vences-te. Alegra-te.";` a instrução `strndelc(str, 3, '-');` deixa em `str` a versão devidamente corrigida, `"Vieste, viste e venceste. Alegra-te."`, enquanto `strndelc(str, 1, 'x');` não tem qualquer efeito.

   ```
   void strndelc(char *str, size_t n, char c);
   ```

3. [2,5] Implemente em *assembly* x86-64 a função `get_last_event`, cuja definição em linguagem C se apresenta a seguir.

   ```
   typedef struct appointment {
       char *description;
       char *schedule;
   } Appointment;

   typedef struct appointments {
       size_t length;
       Appointment *events;
   } Appointments;

   char *get_last_event(Appointments *calendar[], int day) {
       if (calendar[day] != NULL) {
           int last_index = calendar[day]->length - 1;
           return calendar[day]->events[last_index].description;
       }
       return NULL;
   }
   ```

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

   ```
   int filter_objs(void *objs, size_t n, size_t dim, void *ptrs[],
                   void *ctx, int (*eval)(void *ptr, void *ctx)) {
       int i = 0;
       for ( ; n; n--) {
           if (eval(objs, ctx)) ptrs[i++] = objs;
           objs = ((char *)objs) + dim;
       }
       return i;
   }
   ```

   a. [3] Implemente a função `filter_objs` em *assembly* x86-64.
   b. [1,5] Escreva um programa em C que apresente uma chamada à função `filter_objs` com `n` >= 3 e `dim` >= 24. Defina os tipos e declare as variáveis necessárias para a invocação de exemplo, bem como a função de avaliação a passar como 6º argumento.

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

```c
/* f1.c */
#include <stdio.h>

void inc_counter();
char counter[] = "2000";

int main() {
    inc_counter();
    printf("counter = %s\n", counter);
}
```

```c
/* f2.c */
extern char counter;
const float d = 1.0;
long f = 0x7fffffffff00;
static char c;

void inc_counter() {
    static int s = 0;
    counter++;
}
```

   a. [1] 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. [1,5] Existe inconsistência entre a definição e a declaração de `counter`, não detectável no processo de geração do executável. Qual é a inconsistência? Porque não é detectável? Qual é o *output* produzido pelo executável?

   c. [1] Descreva sucintamente como procede o *linker* quando encontra a definição de símbolos com o mesmo nome em diferentes módulos.

6. [1] Considere dois sistemas de *cache N-way set-associative*, A e B, com a mesma capacidade de armazenamento de dados. O sistema de *cache* A usa 4 vias e armazena linhas de 32 *bytes*, enquanto o sistema B tem 2 vias e armazena linhas de 16 *bytes*. Qual a diferença no número de *bits* do endereço que os sistemas A e B usam para determinar o *set*? Apresente os cálculos que justificam a sua resposta.

7. [4] Considere um repositório de informação, suportado numa lista simplesmente ligada, formada por nós do tipo `DataNode`. Cada nó referencia um elemento de informação, apontado pelo campo `data` e com dimensão `length`.

   A função `list_copy` cria uma cópia da lista tomada como argumento e devolve o ponteiro para a nova lista. A nova lista deve ser independente da lista original, isto é, os elementos de dados também devem ser copiados.

   A função `list_destroy` liberta toda a memória alocada.

   Implemente as funções `list_copy e list_destroy` em linguagem C. A alocação da memória necessária para suportar a lista deve ser gerida dinamicamente.

```c
typedef struct data_node {
  struct data_node *next;
  size_t length;
  void *data;
} DataNode;

DataNode *list_copy(DataNode *list);

void list_destroy(DataNode *list);
```

Duração: 2 horas e 30 minutos
ISEL, 17 de Janeiro de 2020