Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação de Sistemas Computacionais
Teste Global de Época de Recurso, Inverno de 2021/2022

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

1. [2,0] Implemente em linguagem C, a função `reverse_bits` que retorna a palavra resultante da inversão de posição dos *bits* de `value`. Inverter a posição de um bit corresponde a colocá-lo na posição oposta, tomando como referência o centro da palavra. Por exemplo: se a palavra tiver 8 *bits*, o *bit* da posição 0 passa para a posição 7 e o da posição 7 passa para a posição 0; o *bit* da posição 1 passa para a posição 6 e o da posição 6 passa para a posição 1; etc.

   ```
   int reverse_bits( int value );
   ```

2. [2,5] Implemente em linguagem C, a função `strrstr` que procura na *string* `str1` a última ocorrência da *string* `str2`. Devolve o ponteiro para o primeiro caráter da última ocorrência ou NULL se `str2` não fizer parte de `str1`.

   ```
   char * strrstr(char *str1, char *str2);
   ```

3. [4] Implemente em *assembly* x86-64 a função `get_val_ptr`, cuja definição em linguagem C se apresenta a seguir.

   ```
   typedef struct data { short flags:6; short length:10; 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, short mask) {
       return (items[item_idx].valid && val_idx < items[item_idx].data[data_idx]->length)
              && (items[item_idx].data[data_idx]->flags & mask)
                     ? &(items[item_idx].data[data_idx]->vals[val_idx])
                     : NULL;
   }
   ```

4. [4] Considere a função `merge_sort_lists` e o tipo `Node`, cujas definições em linguagem C se apresentam a seguir.

   ```
   typedef struct node { struct node *next; void *data; } Node;

   Node *merge_sorted_lists(Node *list_a, Node *list_b, int (*comparator)(void *, void *)) {
       if (list_a == NULL)
              return list_b;
       else if (list_b == NULL)
              return list_a;
       Node *result;
       if (comparator(list_a->data, list_b->data) <= 0) {
              result = list_a;
              result->next = merge_sort_lists(list_a->next, list_b, comparator);
       } else {
              result = list_b;
              result->next = merge_sort_lists(list_a, list_b->next, comparator);
       }
       return result;
   }
   ```

   a. [2] Implemente a função `merge_sort_lists` em *assembly* x86-64.

   b. [2] Escreva, em linguagem C, um programa de teste da função `merge_sorted_lists`, que junta duas lista de informações pessoais, por ordem alfabética do nome das pessoas. A informação pessoal é guardada em `struct person {char *name, int age}`. No programa, deve explicitar a formação das listas com informação de pelo menos duas pessoas, a definição da função de comparação e a invocação da função `merge_sort_lists`.

5. [3,0] Considere o conteúdo dos ficheiros fonte `f1.c` e `f2.c`.

```
/* f1.c */

#define    DIM    3

float a = 10.5;

const int b = 3.5;

void f(int);

extern int table[DIM];

int main() {
    f(b);
}
```

```
/* f2.c */

#include <stdio.h>

int a;

int *p = &a;

int table[] = {1, 2, 3};

void f(float c) {
    printf("%x %f", a, c);
}
```

   a. [1,5] Indique o conteúdo da tabela de símbolos dos ficheiros objecto relocáveis `f1.o` e `f2.o`, resultantes da compilação de `f1.c` e `f2.c`. Para cada símbolo indique o nome, a secção e o respectivo âmbito (local ou global). Propõe-se que use a notação do utilitário `nm`, em que a classificação dos símbolos com minúscula (`t`, `d` ou `b`) indica âmbito local.

   b. [1,5] O programa resultante da ligação dos módulos `f1.c` e `f2.c` escreve **`41280000 0.000000`** na consola, ao ser executado. Justifique.

6. [1] Considere o projeto de uma *cache* com organização N-*way set associative*, em que se pretende modificar a organização para duplicar o número de vias, mantendo a capacidade total de dados e a dimensão dos blocos. Indique se, com esta modificação, a quantidade total de memória necessária para *tags* se mantém, aumenta ou reduz. Se considerar que é diferente, descreva a relação dessa diferença com os outros parâmetros de dimensionamento da *cache*. Justifique a resposta; se considerar conveniente para a justificação, pode exemplificar com valores concretos e os respetivos cálculos.

7. [3,5] Uma lista simplesmente ligada de nós com o tipo `Item`, contém elementos de informação, que têm uma *string* (apontada por `elem_id`), como identificador, e um ponteiro para dados genéricos (`void *data`). A lista é ordenada, alfabeticamente crescente, pelos identificadores. Pretende-se melhorar o tempo de pesquisa dos elementos, separando a lista em sublistas que são grupos de elementos. Cada grupo é uma sublista de elementos cujos identificadores têm o primeiro caráter igual. Para acesso aos grupos, é criada uma nova lista com nós do tipo `Group`; cada um contém o ponteiro para uma sublista (`sublist`) e regista o caráter que a identifica (`group_id`). A função `build_groups` separa a lista original (`orig_list`) em sublistas, constrói a lista de nós `Group` para as referenciar e devolve o ponteiro para o primeiro elemento da lista criada. Esta deve ter os elementos ordenados crescentemente por `group_id`. As sublistas devem manter a ordem da lista original. A função `find_in_groups` procura a sublista que pode conter o identificador `elem_id` e dentro desta o respetivo elemento; devolve o valor do ponteiro de dados encontrado (campo `data`) ou NULL se o identificador pesquisado não existir. Escreva as funções especificadas. Valoriza-se a eficiência.

```
typedef struct item { char *elem_id; void *data; struct item *next; } Item;
typedef struct group { char group_id; Item *sublist; struct group *next; } Group;

Group *build_groups( Item *orig_list );
void *find_in_groups( Group *group_list, char *elem_id );
```

Duração: 2 horas e 30 minutos
ISEL, 19 de Fevereiro de 2022