Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação de Sistemas Computacionais
Teste global de Época de Recurso, Verão de 2020/2021

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

1. [2,5] Implemente, em linguagem C, a função `setbits` que substitui, no valor apontado por `pvalue`, a sequência de *bits* entre a posição definida por `position` e a posição definida por `position` + `nbits`, pelo valor do parâmetro `new_value`. Admita que `nbits` pode assumir valores entre 0 - caso em que não há nenhuma substituição - e o número de *bits* do tipo `int` - caso em que há substituição total. Programe de forma genérica, sem se comprometer com uma dimensão concreta para o tipo `int`. (Atenção a situações de *overflow*.)

   ```
   void setbits(unsigned *pvalue, int position, int nbits, unsigned new_value);
   ```

2. [2,5] Implemente, em linguagem C, a função `parse_uri` que suprime os caracters presentes no início da *string* passada no parâmetro `uri`, até encontrar o caractere **'?'**, que também deve ser suprimido. Na restante *string* deve substituir as sequências **"%20"** pelo caractere espaço **' '**. Exemplo: se for passada em `uri` a *string* **"/cgi-bin/age?ana%20maria"**, a função retorna a *string* com o conteúdo **"ana maria"**. A *string* retornada deve usar o mesmo suporte de memória da *string* `uri`.

   ```
   char *parse_uri(char *uri);
   ```

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

   ```
   typedef struct buffer { size_t size; char *data; } Buffer;
   typedef struct stream { size_t n_buffers; Buffer **buffers; } Stream;

   char readchar(Stream *stream, size_t offset, char *datap) {
       size_t offset_end = 0;
       for (size_t i = 0; i < stream->n_buffers; ++i) {
           size_t offset_begin = offset_end;
           offset_end += stream->buffers[i]->size;
           if (offset < offset_end) {
               *datap = stream->buffers[i]->data[offset - offset_begin];
               return 1;
           }
       }
       return 0;
   }
   ```

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

   ```
   void vector_conditional_insert(void **vector, size_t size, void *new,
       int (*compar)(const void *, const void *)) {
       void **last = vector + size;
       for (void **current = vector ; current < last; current++)
           if ((*compar)(*current, new)) {
               memmove(current + 1, current, (last - current) * sizeof *current);
               *current = new;
               return;
        }
   }
   ```

   a. [2,5] Implemente a função `vector_conditional_insert` em *assembly* x86-64.

b. [2,5] Para efeitos de teste, escreva em linguagem C, um programa que, utilizando a função `vector_conditional_insert`, insira uma nova string num array de ponteiros para *strings*, ordenado por ordem alfabética do conteúdo das *strings* apontadas.

   Defina um *array* com pelo menos três posições, a função de comparação adequada a passar como argumento em `compar` e a função **`main`**, contendo a invocação de `vector_conditional_insert`.

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

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

extern char *incr(char *p, unsigned off);

static unsigned char data[] =
    {3, 0, 0, 0, 1, 0, 2, 0, 3, 0};

int main() {
    printf("%d\n", proc(data));
}
```

```c
/* f2.c */
#define OFFSET 1000

static struct vector {
    unsigned len; short data[16];
} Vector;

unsigned int data = OFFSET;

int proc(struct vector *pv) {
    int res = 0;
    for (int i = 0; i < pv->len; i++)
        res += pv->data[i];
    return res + data;
}
```

   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). Pode usar as convenções do utilitário **`nm`**.

   b. [1] Se considerar que os ficheiros objecto f1.o e f2.o podem ser combinados pelo *linker* com sucesso para gerar um executável, diga o que é mostrado na consola quando se executa o executável produzido. Se, pelo contrário, considerar que o *linker* falhará a combinação, diga qual a razão ou razões porque isso acontece.

6. [1] Considere um processador que usa uma *cache* com organização *set-associative* de 8 vias, 8 MiB de capacidade e 8192 *sets*. Quais os *bits* do endereço usados para endereçar no interior do bloco.

7. [4] Considere a composição de pacotes de dados baseada em fragmentos de dados de dimensão diversa. Os fragmentos são representados por uma *struct* do tipo `Fragment`, que contém um ponteiro para os dados e a dimensão desses dados. Um pacote é representado por uma lista ligada de nós do tipo `PackNode`. Cada nó indica um fragmento de dados e aponta para o nó seguinte. Em abstrato o pacote é formado pela concatenação de todos os fragmentos, pela ordem da lista ligada, e cuja dimensão é a soma das dimensões de todos os fragmentos que o compõem.

   A função `append` recebe em `packet` a representação de um pacote e acrescenta no fim, um fragmento com os dados definidos pelos parâmetros `data` e `size`. Esta função devolve um ponteiro para a lista resultante.

   A função `triam_ahead`, recebe em `packet` a representação um pacote e extrai e elimina, do início do pacote, copiando para `buffer`, uma porção de dados de dimensão `buffer_size`. A função devolve como valor o ponteiro para a representação da lista resultante. Através do ponterio `size` a função dá conhecimento da porção de dados efetivamente extraídos, que pode ser inferior a `buffer_size`.

   `Implemente em linguagem C as funções descritas.`

```c
typedef struct fragment { size_t size; unsigned char *data; } Fragment;
typedef struct pack_node { struct pack_node *next; Fragment *frag; } PackNode;
PackNode *append(PackNode **packet, const char *data, size_t size)
PackNode *trim_ahead(PackNode *packet, char *buffer, size_t buffer_size, size_t *size);
```

Duração: 2 horas e 30 minutos

ISEL, 21 de julho de 2021