Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Verão de 2021/2022

Série de Exercícios 2

Nos exercícios seguintes é proposta a escrita de funções em *assembly* para a arquitetura x86-64, usando a variante de sintaxe AT&T, e seguindo os princípios básicos de geração de código do compilador de C da GNU. A resolução de cada exercício (código *assembly*) deve ser apresentado em conjunto com o respectivo programa de teste escrito em linguagem C. Tenha em consideração que, na discussão, os exercícios que não forem demonstrados a funcionar serão considerados como não tendo sido realizados. Não se esqueça de testar devidamente o código desenvolvido (invocando as funções escritas em *assembly* com pelo menos três parametrizações diferentes), bem como de o apresentar de forma cuidada, apropriadamente indentado e comentado. Não é necessário relatório. Contacte o docente se tiver dúvidas. Encoraja-se também a discussão de problemas e soluções com colegas de outros grupos, mas salienta-se que a partilha directa de soluções leva, no mínimo, à anulação das entregas de todos os envolvidos.

1. A função **`reverse`** inverte um valor inteiro sem sinal representado a 64 *bit,* criando um novo valor, em que qualquer *bit* de peso 2^n do novo valor, é igual ao bit 2^{63-n} do valor original. Escreva em *assembly* x86_64 a função **`reverse`**, cujo protótipo é apresentado a seguir.

   ```
   uint64_t reverse(uint64_t val);
   ```

2. Escreva em *assembly* x86_64 a função **`memmove`**, tal como está definida na biblioteca *standard* da linguagem C. Esta função move os **`len`** *bytes* de memória, a partir do endereço definido por **`src`**, para o endereço definido por **`dst`**. Tenha em consideração que a zona de memória de destino pode coincidir, no todo, ou em parte, com a zona de memória da fonte. Procure minimizar o número de acessos à memória, realizando, sempre que possível, acessos a palavras de 64 *bits* em endereços alinhados a 64 *bits* (i.e., endereços múltiplos de 8).

   ```
   void *memmove(void *dst, const void *src, size_t len);
   ```

3. Relativamente ao enunciado proposto no exercício 4 da 1º Série de Exercícios, reescreva a função **`stream_read`** em linguagem *assembly* x86-64.

4. Considere a função **`find_cars`** cuja implementação em C se apresenta abaixo. Esta função percorre o *array* **`cars`** com **`cars_size`** elementos e invoca, para cada um deles, a função **`predicate`** passada em parâmetro. Se **`predicate`** retornar um valor verdadeiro é guardado o ponteiro para essa posição de **`cars`** no *array* **`found_cars`**. A função **`find_cars`** termina quando for atingido o final de um dos *arrays*, retornando o número de elementos encontrados.

   ```
   struct car {char *brand; int year; char *color; }

   int find_cars(struct car cars[], size_t cars_size,
                   struct car *found_cars[], size_t found_cars_size,
                   int (*predicate)(struct car *car, void *context), void *context) {
       int j = 0;
       for (size_t i = 0; i < cars_size; ++i)
           if (predicate(&cars[i], context)) {
               found_cars[j] = &cars[i];
               if (++j == found_cars_size)
                     return j;
           }
       return j;
   }
   ```

   a. Traduza a função **`find_cars`** para linguagem assembly x86_64.

   b. Escreva um programa de teste que selecione de um conjunto de carros, os de um dado ano de fabrico.

Data recomendada para conclusão: 7 de Maio de 2022

ISEL, 1 de Abril de 2022