Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2022/2023

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.

Deve submeter a sua realização de cada exercício aos testes anexos a este enunciado. As respetivas instruções de utilização estão incluídas no próprio pacote de testes.

Tenha o cuidado de apresentar o código 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. Tenha consciência que os exercícios só são benéficos na aprendizagem se forem realizados com honestidade académica.

1. Escreva em *assembly* a função **`rotate_right`** que desloca para a direita (no sentido de maior peso para o de menor peso) o valor a 128 *bit*, que recebe no parâmetro **`value`**, o número de posições indicadas no parâmetro **`n`**. O valor numérico de 128 *bit* é formado pela concatenação de dois valores a 64 *bit* armazenados num *array* com duas posições, segundo o formato *little-endian*. Os *bits* que saem da posição de menor peso entram, pela mesma ordem, na posição de maior peso.

   ```
   void rotate_right ( unsigned long value[], size_t n );
   ```

2. Programe em *assembly* a função `my_strlen` segundo a definição da função `strlen` na biblioteca normalizada da linguagem C. Procure minimizar o número de acessos à memória efetuando acessos alinhados a palavras com múltiplos *bytes*.

   ```
   size_t my_strlen ( const char *str );
   ```

3. Considere a função `compare_data_value`, cuja definição em linguagem C se apresenta a seguir. Implemente esta função em *assembly* em duas versões, uma para cada definição da *struct* `Dataset`.

   a) `typedef struct { const int *id; unsigned length; Data **data; } Dataset;`

   b) `typedef struct { const int *id; unsigned length; Data *data[]; } Dataset;`

   ```
   typedef struct { char label[7]; short value; } Data;


   int compare_data_value ( Dataset *set1, Dataset *set2, unsigned index ) {
       if (index >= set1->length || index >= set2->length ||
              NULL == set1->data[index] || NULL == set2->data[index])
              return -1;
       return set1->data[index]->value == set2->data[index]->value;
   }
   ```

4. Considere a função genérica `find` que realiza uma pesquisa sequencial num *array*. Programe esta função em *assembly x86-64*.

```
size_t find (void *array, size_t array_size, size_t elem_size,
           int (*predicate)(const void *, const void *), const void *context,
           void *result[], size_t result_size ) {
    char *iter, *last = (char *)array + array_size * elem_size;
    void **result_iter = result;
    for (iter = array; iter < last ; iter += elem_size) {
           if (predicate(iter, context) == 0) {
                  *result_iter++ = iter;
                  if (--result_size == 0)
                         break;
           }
    }
    return result_iter - result;
}
```

Data recomendada para conclusão: 27 de Novembro de 2023

ISEL, 24 de Outubro de 2023