Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2023/2024

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 `next_pow2` que retorna a potência de dois igual ou superior ao argumento passado em parâmetro. Retorna igual se o valor do argumento corresponder a uma potência inteira de 2. Utilize a instrução BSR.

   ```
   size_t next_pow2(size_t);
   ```

2. Programe em *assembly* a função `my_strtok` segundo a definição da função `strtok` na biblioteca normalizada da linguagem C.

   ```
   char *my_strtok(char *str, const char *delim);
   ```

3. Considere a função `compare_offset`, cuja definição em linguagem C se apresenta abaixo. Implemente esta função em *assembly* x86-64.

   ```
   struct values {
       char len;
       short base;
       short *offset;
   };

   int compare_offset(struct values *values[], int i, int j, int k, struct values *avalue) {
       return k > values[i][j].len &&
              values[i][j].offset[k] == avalue->offset[k];
   }
   ```

4. Considere a função genérica `bubble_sort` apresentada abaixo assim como a função auxiliar `memswap`. Programe ambas as funções em *assembly x86-64*.

```
static void memswap(void *one, void *other, size_t width){
    char tmp[width];
    memcpy(tmp, one, width);
    memcpy(one, other, width);
    memcpy(other, tmp, width);
}

void bubble_sort(void *base, size_t nelements, size_t width,
    int (*compare)(const void *, const void *)) {

    int swap_flg = 0;
    if (nelements <= 1)
          return;
    char *limit = (char *)base + (nelements - 1) * width;
    for (char *ptr = base; ptr < limit; ptr += width)
          if (compare(ptr, ptr + width) > 0) {
                memswap(ptr, ptr + width, width);
                swap_flg = 1;
           }
    if (swap_flg == 0)
         return;
    bubble_sort(base, nelements - 1, width, compare);
}
```

Data recomendada para conclusão: 19 de Novembro de 2023

ISEL, 16 de Outubro de 2023