Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2020/2021

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 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, mas salienta-se que a partilha direta de soluções leva, no mínimo, à anulação das entregas de todos os estudantes envolvidos.

1. Escreva em assembly x86-64 a função `rotate_left` que roda para a esquerda 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*. Procure tirar partido das instruções *double precision shift* do processador.

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

   Nota: os bits deslocados para a esquerda da posição 127 devem ser inseridos, pela mesma ordem, na posição 0.

2. Programe em *assembly* x86-64 a função `my_memcpy` segundo a definição da função `memcpy` na biblioteca normalizada da linguagem C. Esta função copia o conteúdo da zona de memória definida pelo ponteiro `source` e dimensão `num` para a zona de memória a partir de `destination`. Procure minimizar o número de acessos à memória efetuando acessos alinhados a palavras com múltiplos bytes.

   ```
   void *memcpy(void *destination, const void *source, size_t num);
   ```

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

   ```
   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;
   }
   ```

   Teste a função `get_val_ptr` escrita em *assembly* invocando-a de uma função escrita em C, com diversas combinações de argumentos.

4. Apresenta-se abaixo uma implementação do algoritmo *bubblesort* de forma recursiva.

   a. Implemente as funções `bubble_sort` e `memswap` em linguagem *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 nel, size_t width,
    int (*compar)(const void *, const void *)) {
    if (nel <= 1)
          return;
    char *limit = (char *)base + (nel - 1) * width;
    for (char *ptr = base; ptr < limit; ptr += width)
          if ((*compar)(ptr, ptr + width) > 0)
                memswap(ptr, ptr + width, width);
    bubble_sort(base, nel - 1, width, compar);
}
```

   b. Escreva, em linguagem C, um programa de teste da função `bubble_sort` que ordena um array de ponteiros para instâncias do tipo `Student` , por ordem crescente dos nomes. No programa, deve explicitar a inicialização do array de ponteiros, a definição da função de comparação, assim como o código para mostrar na consola a informação sobre os estudantes ordenada.

```
typedef struct { int number; const char *name; } Student;
```

Data recomendada para conclusão: 6 de Dezembro de 2020

ISEL, 9 de Novembro de 2020