CS244, STELLENBOSCH UNIVERSITY

TUT 5: CALL-BY-REFERENCE PARAMETERS, POINTERS, AND STRUCTURED DATA TYPES IN 32-BIT INTEL ASSEMBLY LANGUAGE

22 OCTOBER 2021

I really hate this darn machine;
I wish that they would sell it.
It won't do what I want it to,
but only what I tell it.

—Programmer's lament

OVERVIEW

The goal of this tutorial is to gain more experience in writing recursive functions in assembly language. Most routines use call-by-reference parameters and involve the manipulation of structured data types. Arrays and binary trees are explored as data types. After successfully completing this tutorial, you should be able to:

- translate C functions to Intel assembly language by hand;
- understand how function results, and call-by-value and call-by-reference parameters work in C;
- understand how structures are manipulated, specifically arrays and `structs`; and
- use the Intel assembly language instructions `add`, `call`, `cld`, `cmp`, `cmps`*c*, `dec`, `inc`, `j`*cc*, `jmp`, `lea`, `lods`*c*, `mov`, `movs`*c*, `pop`, `push`, `rcl`, `rep`*cc*, `ret`, `sar`, `scas`*c*, `set`*c*, `std`, `stos`*c*, `sub`, and `xor`.

A NOTE ON 32-BIT V. 64-BIT EXECUTABLES

Remember that we create 32-bit executables. If, for example, you use a C program to work out the alignment of data structures, you must use the `-m32` when compiling. The makefile included in the repository already has this switch included in the compilation directive.

INSTRUCTIONS

1. Clone your repository for Tutorial 5 at

   `rw244-2021@gitolite.cs.sun.ac.za:`⟨*US number*⟩`/tut5`

2. The repository contains a number of files, include a makefile and a C program that must be used for testing. The C header file `tut5.h` contains the following three prototypes:

   - `void binary_sort(Item *list, int n);`
   - `void delete_node(Node **root, char *name);`

- `void bin_to_string(int n, char *s);`

The first two functions are defined using the following user-defined types, respectively:

```
typedef struct item_s {
    int  number;
    char name[32];
} Item;

typedef struct node_s Node;
typedef struct node_s {
    char  name[32];
    Node *left;
    Node *right;
} Node;
```

The C source code for all the functions are in the file `test5.c`; they are called `binary_sort_c`, `delete_node_c`, and `bin_to_string_c`, repectively.

Note that the `delete_node` function performs deletion on a binary search tree (BST). The only reference to the BST is a pointer to its root, and the functions `create_node` and `insert_node` are provided for BST construction; they do not have to implemented in assembly language. In addition, the function `delete_node_c` calls the function `remove_max_c` to locate and unlink the maximum node in the left subtree of the node that will be deleted, so that it can be used as the root node of the subtree where the deleted node is now missing. You have to provide an assembly language version of this function, called `remove_max`, for use by `delete_node`.

3. You have to complete `remove_max` and `delete_node` for submission via Git. The last commit will be assessed as your attendance assignment for next week.

4. You MAY NOT use any of the functions provided by the C standard string library. Instead, use the string instructions of the processor: `cmps`, `lods`, `movs`, `scas`, and `stos`.

5. **OPTIMISATION EXERCISE:** The `bin_to_string` function is an exercise in code optimisation. (It is not for submission.) You have to implement the function based on the following constraints:

   (a) You MAY NOT use the arithmetic instructions `div` and `idiv`.

   (b) Only general instructions supported by the 80386 processor may be used; refer to the Intel manual if you are unsure.

   (c) No local variables may be used.

   (d) Implement your solution with the minimum number of instructions. It is possible to implement this routine with only ten instructions. (This excludes the instructions used to create and destroy the stack frame, as well as the instructions used to preserve any registers.)

6. You MUST code your solutions by hand, and you MAY NOT use any tools for disassembly or decompilation. You MUST complete your plagiarism form in the repository.

7. There are no marks for style.