Project: Assembly Language II

COSC 216.001

Due Date: November 3, 2021, 11:59pm

1 Overview

Each student is asked to write an assembly language program that accesses and makes use of C-subroutines.

1.1 Objectives

- Understanding Assembly language code
- Interfacing C and Assembly code.

2 Detailed Description

2.1 Introduction

In this project you are asked to integrate pre-existing C-functions into your assembly language program. Your code, while written in assembly language, will have to pass parameters, call C functions, and utilize the return values. You will need to understand the C-function interface and call structure.

2.2 What to do

You are to implement a simple postfix to infix expression converter. You assembly language program should assume a string address is passed in via `R0` as a packed C-style string. You should then run the simplified postfix to infix algorithm specified below, and return a new string in infix notation via `R0`. Your code should call the following C-style functions, as necessary (these functions are provided in VisUAL compatible ARM format in util.s):

```
// returns the length of string s
int _STRLEN(char *s);

// allocates and returns a string that is the concatenation of strings s1 and s2.
char *_CONCAT(char *s1, char *s2);

// If possible, allocates bytes from the heap.  This is an all-or-nothing attempt.
```

```
void *_ALLOC(unsigned long bytes);

// returns the number of items stored in stack
int _STACK_SIZE();

// adds s to the top of the stack.
void _PUSH(char *s);

// removes and returns the string at the top of the stack.
char *_POP();

// returns the string at the top of the stack, without modifying the stack.
char *_PEEK();
```

2.2.1 Postfix to Infix Algorithm

Process each character of the input as follows:

1. if the character is a blank, skip it.
2. if the character is a digit, put it onto the stack.
3. otherwise, it is an operator
   (a) pop the term2 off the stack
   (b) pop the term1 off the stack
   (c) concatenate: '(' + term1 + operator + term2 + ')'
   (d) push the concatenated string onto the stack
4. At the end of the input string, if there is only one thing on the stack, it is the converted infix formulation of the expression.

2.3 Running your code

Please use the VisUAL ARM emulator to test your code. You can download VisUAL at: `https://salmanarif.bitbucket.io/visual/downloads.html`

3 Submission

Each of your source code files must have a comment near the top that contains your name, StudentID, and M-number.

The project should be submitted via Blackboard by **November 3, 2021, 11:59pm**. Follow these instructions to turn in your project.

You should submit the following files:

- post2infix.s
- *any other source files your project needs*

The following submission directions use the instructions that we will use for all projects this semester.

1. create a directory for your project:

   ```
   mkdir asm2
   ```
   2

2. create (or copy) all of your source files in this directory. Example: To copy a file called `example.s` into your `c2asm` directory:

   ```
   cp example.s asm2
   ```
   1

3. change parent directory of your project directory:

   ```
       cd asm2/..
   ```
   1

4. Create a tar file named `<user_name>.tar.gz`, where `<user_name>` is your studentID and `<proj_dir>` is the directory containing the code for your project, by typing:

   ```
   tar -czf <user_name>.tar asm2
   ```
   2

5. Finally, submit the compressed tar file to Blackboard in accordance with the class policies.

Late assignments will not be given credit.

4 Grading

While this rubric is subject to change based on class performance, the current grading rubric for this assignment is as follows:

component	value
Correctness	40
Completeness	40
Code Quality	20