ENSF 614

Advanced System Analysis and Software Design

LAB 2

Author:

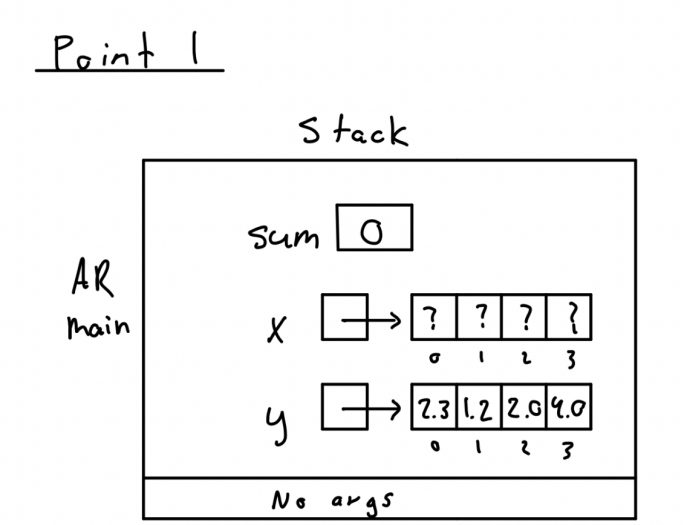
Steven Duong
(30022492)

Affiliation
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Alberta

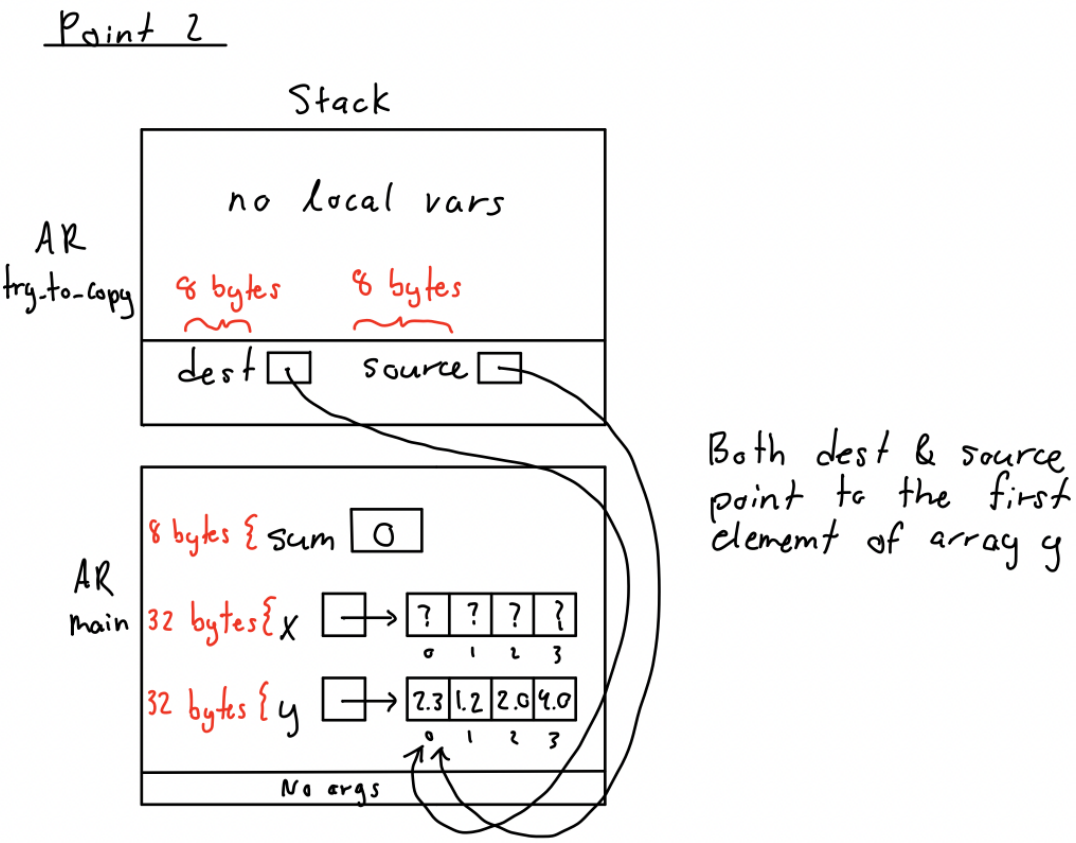
Lab Block: B01
Date of Report: Jan 23, 2023

Exercise A

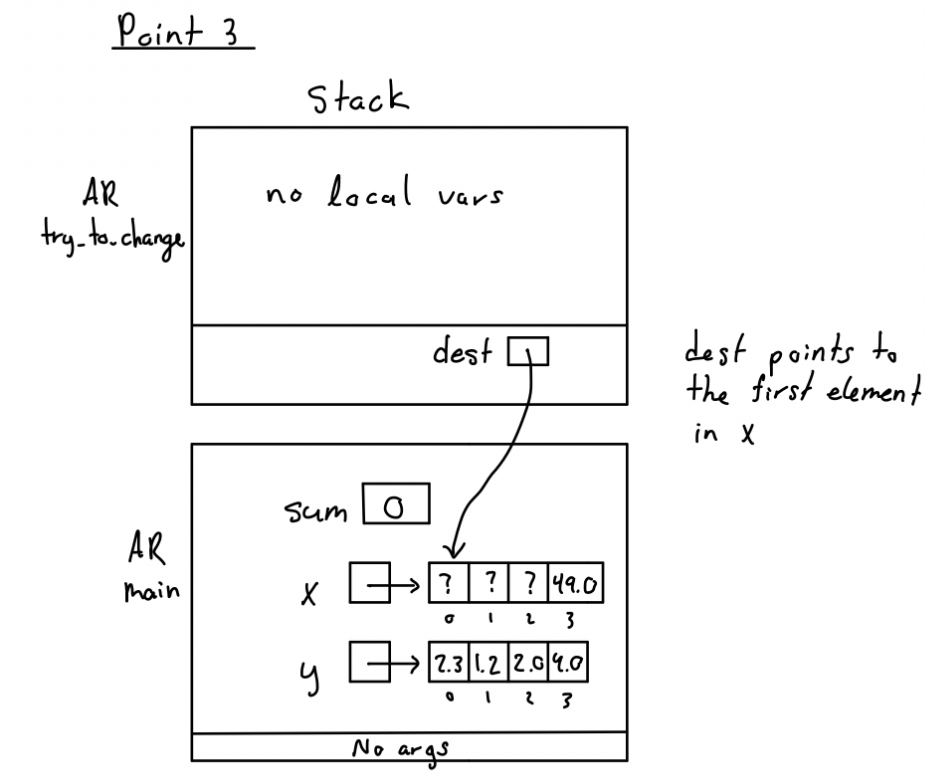
AR Memory Diagram for Point One:



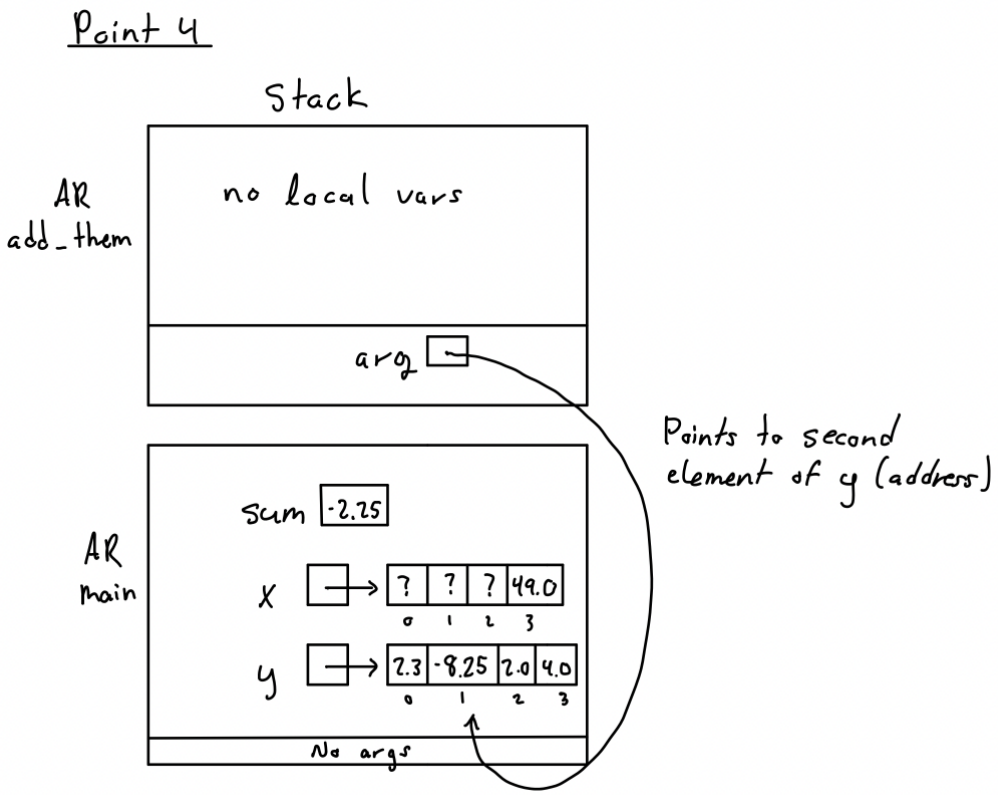
AR Memory Diagram for Point Two:



AR Memory Diagram for Point Three:



AR Memory Diagram for Point Four:



Exercise B

C++ Code:

```
/*
 *  File Name: lab2exe_B.cpp
 *  Assignment: ENSF 614 Lab 2 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Jan 23, 2023
 */

int my_strlen(const char * s);
/*  Duplicates strlen from <cstring>, except return type is int.
 *  REQUIRES
 *     s points to the beginning of a string.
 *  PROMISES
 *     Returns the number of chars in the string, not including the
 *     terminating null.
 */

void my_strncat(char * dest,
  const char * source, int num);
/*  Duplicates strncat from <cstring>, except return type is void.
 */

int my_strcmp(const char * str1,
  const char * str2);
/*
 * REQUIRES
 *    string comparison between str1 and str2 by subtracting the ASCII
 *    values of the first two different characters that appear.
 * PROMISES
 *    1) Returns 0 if two strings are identical
 *    2) Returns a positive number if str1 is greater than str2
 *    3) Returns a negative number if str1 is less than str2
 */

#include <iostream>

#include <cstring>

using namespace std;

int main(void) {
  char str1[7] = "banana";
  const char str2[] = "-tacit";
  const char * str3 = "-toe";
```

```
  /* point 1 */
  char str5[] = "ticket";
  char my_string[100] = "";
  int bytes;
  int length;

  /* using strlen library function */
  length = (int) my_strlen(my_string);
  cout << "\nLine 1: my_string length is " << length;

  /* using sizeof operator */
  bytes = sizeof(my_string);
  cout << "\nLine 2: my_string size is " << bytes << " bytes.";

  /* using strcpy library function */
  strcpy(my_string, str1);
  cout << "\nLine 3: my_string contains: " << my_string;

  length = (int) my_strlen(my_string);
  cout << "\nLine 4: my_string length is " << length << ".";

  my_string[0] = '\0';
  cout << "\nLine 5: my_string contains:\"" << my_string << "\"";

  length = (int) my_strlen(my_string);
  cout << "\nLine 6: my_string length is " << length << ".";

  bytes = sizeof(my_string);
  cout << "\nLine 7: my_string size is still " << bytes << " bytes.";

  /* strncat append the first 3 characters of str5 to the end of
my_string */
  my_strncat(my_string, str5, 3);
  cout << "\nLine 8: my_string contains:\"" << my_string << "\"";

  length = (int) my_strlen(my_string);
  cout << "\nLine 9: my_string length is " << length << ".";

  my_strncat(my_string, str2, 4);
  cout << "\nLine 10: my_string contains:\"" << my_string << "\"";

  /* strncat append ONLY up to '\0' character from str3 -- not 6
characters */
  my_strncat(my_string, str3, 6);
  cout << "\nLine 11: my_string contains:\"" << my_string << "\"";

  length = (int) my_strlen(my_string);
  cout << "\nLine 12; my_string has " << length << " characters.";

  cout << "\n\nUsing strcmp - C library function: ";
```

```
  cout << "\n\"ABCD\" is less than \"ABCDE\" ... strcmp returns: " <<
    my_strcmp("ABCD", "ABCDE");

  cout << "\n\"ABCD\" is less than \"ABND\" ... strcmp returns: " <<
    my_strcmp("ABCD", "ABND");

  cout << "\n\"ABCD\" is equal to \"ABCD\" ... strcmp returns: " <<
    my_strcmp("ABCD", "ABCD");

  cout << "\n\"ABCD\" is less than \"ABCd\" ... strcmp returns: " <<
    my_strcmp("ABCD", "ABCd");

  cout << "\n\"Orange\" is greater than \"Apple\" ... strcmp returns:
" <<
    my_strcmp("Orange", "Apple") << endl;

  return 0;
}

int my_strlen(const char * s) {
  // creating a counter variable
  int count = 0;

  // looping the pointer in the string starting from the first
character
  while ( * s != '\0') {
    count++;
    s++;
  }

  // returns the string length, excluding '\0'
  return count;
}

void my_strncat(char * dest,
  const char * source, int num) {

  // Checks if initial string is null
  if ( * dest == NULL) {
    for (int i = 0; i < num; i++) {
      dest[i] = source[i];
      dest[i + 1] = '\0';
    }
  } else {

    // Puts the pointer at location of '\0'
    int counter = 0;
    while ( * dest != '\0') {
      counter++;
```

```
      dest++;
    }

    // reset the pointer back to the first element.
    dest -= counter;

    int templen = counter + num;
    char temp[templen];

    // Appends dest string into temp
    for (int i = 0; i < counter; i++) {
      temp[i] = dest[i];
      temp[i + 1] = '\0';
    }

    // Appends source string into temp
    for (int i = counter, j = 0; i < templen; i++, j++) {
      temp[i] = source[j];
      temp[i + 1] = '\0';
    }

    // Appends final temp string into dest string
    for (int i = 0; i < templen; i++) {
      dest[i] = temp[i];
    }
  }
}

int my_strcmp(const char * str1,
  const char * str2) {

  bool equal = true;

  // Checks to see if str1 and str2 are equal. Returns 0 if true.
  if (strlen(str1) == strlen(str2)) {
    for (int i = 0; i < strlen(str1); i++) {
      if (str1[i] != str2[i]) {
        equal = false;
      }
    }
    if (equal) {
      return 0;
    }
  }

  // Finds the maximum length between the two strings that are
compared.
  int max;
  if (strlen(str1) > strlen(str2)) {
    max = (int) strlen(str1);
```

```
  } else {
    max = (int) strlen(str2);
  }

  // Compares each letter in str1 and str2. Returns the ASCII value
differences
  // of the first 2 characters that are different.
  int count = 0;
  int res;
  for (int i = 0; i < max; i++) {
    if (str1[i] != str2[i] && count < 1) {
      res = str1[i] - str2[i];
      count++;
    }
  }

  // Returns the result of the ASCII values
  return res;
}
```

Program Output for Exercise B:

```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 2/cmake-build-debug/Lab_2"

Line 1: my_string length is 0
Line 2: my_string size is 100 bytes.
Line 3: my_string contains: banana
Line 4: my_string length is 6.
Line 5: my_string contains:""
Line 6: my_string length is 0.
Line 7: my_string size is still 100 bytes.
Line 8: my_string contains:"tic"
Line 9: my_string length is 3.
Line 10: my_string contains:"tic-tac"
Line 11: my_string contains:"tic-tac-toe"
Line 12; my_string has 11 characters.

Using strcmp - C library function:
"ABCD" is less than "ABCDE" ... strcmp returns: -69
"ABCD" is less than "ABND" ... strcmp returns: -11
"ABCD" is equal to "ABCD" ... strcmp returns: 0
"ABCD" is less than "ABCd" ... strcmp returns: -32
"Orange" is greater than "Apple" ... strcmp returns: 14

Process finished with exit code 0
```

Exercise E

C++ Code:

```cpp
/*
 *  File Name: lab2exe_E.cpp
 *  Assignment: ENSF 614 Lab 2 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Jan 23, 2023
 */

#include "lab2exe_E.h"

cplx cplx_add(cplx z1, cplx z2) {
  cplx result;

  result.real = z1.real + z2.real;
  result.imag = z1.imag + z2.imag;
  return result;
}

void cplx_subtract(cplx z1, cplx z2, cplx * difference) {
  ( * difference).real = z1.real - z2.real;
  ( * difference).imag = z1.imag - z2.imag;
}

void cplx_multiply(const cplx * pz1,
  const cplx * pz2, cplx * product) {
  double a = ( * pz1).real;
  double b = ( * pz1).imag;
  double c = ( * pz2).real;
  double d = ( * pz2).imag;

  ( * product).real = (a * c - b * d);
  ( * product).imag = (a * d + b * c);
}
```