CSD1100 Programming Assignment: Lab 11 (Assembler - Flow Control)

Topics and references

- Debugging using gdb
- Branching
- Looping

Task

In this assignment you have to implement 2 functions that take parameters and return result of calculations using basic arithmetic operators and flow control instructions.

Complete explanation how to control the flow by using instructions `jmp`, `cmp`, `je`, `jge` and so on **will be given in the class**.

You implementation must passes all given tests in order to get the full mark for the assignment.

Submission details

Please read the following details carefully and adhere to all requirements to avoid unnecessary deductions.

Source files

You have to submit your implementation of the functions in the source file `functions.asm`.

```
; ------------------------------------------------------------------------
; File: functions.asm
; Project: CSD1100 Assignment 11
; Author: Vadim Surov, vsurov@digipen.edu
; Co-Author: Your name, email
;
; Compile: nasm -f elf64 -g -F dwarf functions.asm -o functions.o
; Link: gcc main.o functions.o -o main.o -lm
; Run: ./main 0
; Debug: gdb main
;  (gdb) b f1
;  (gdb) run
;  ...
;  0
;  ...
;  (gdb) ...
;
; Copyright: 2021, Digipen Institute of Technology, Singapore
;
; Note: All functions use at most 6 parameters
;       p1, p2, p3, p4, p5, p6
;       located in registers
;       rdi, rsi, rdx, rcx, r8, r9
;       accordingly.
```

```
; ------------------------------------------------------------------

    section .text

    global f1
    global f2

f1:
; TODO: - Given two circles with central points at (p1,p2), (p3,p4),
;       and radii p5, p6. All values in p1, ..., p6 are integers.
;       - Create code that determins intersection
;       or non-intersection of the circles.
;       - You code must return 1 when intersecting or 0 otherwise
;       based on the result of calculation.
;       - Tip 1: use the method without square root calculations:
;       if (p1-p3)^2+(p2-p4)^2 <= (p5+p6)^2 then return 1
;       - Tip 2: do not use pow() function to calculate x^2, use
;       x*x instead.
;       - Note that a point is a circle with radius 0.

    ret    ; return rax;

f2:
; TODO: - Calculare (p1+0)*(p2+0) + (p1+1)*(p2+1) + (p1+2)*(p2+2) + .... +
;           (p1+p3)*(p2+p3)
;       - Tip: Accumulate the result in reverse order by decrementing
;           p3 or (p1+p3) and (p2+p3)

    ret    ; return rax;
```

Test cases are given in `main.c` file. Do not change it. It won't be submitted anyway.

```
/*------------------------------------------------------------------
  File: main.c
  Project: CSD1100 Assignment 11
  Author: Vadim Surov, vsurov@digipen.edu

  Compile: gcc -c -Wall -Wextra -Werror main.c -o main.o
  Link: gcc main.o functions.o -o main.o -lm

  Copyright: 2021, Digipen Institute of Technology, Singapore
------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>

// See the function description in functions.asm
int f1();
int f2();

void test1();
void test2();
void test3();
void test4();
void test5();
void test6();
void test7();
```

```
void test8();
void test9();
void test10();

int main(int argc, char* argv[])
{

    void (*f[])() = { test1, test2, test3, test4, test5, test6, test7, test8,
test9, test10 };
    const int SIZE = sizeof(f) / sizeof(f[0]);
    int id = -1;

    if (argc == 2)
    {
        if (argv[1][0] == 'i')
        {
            printf("Enter the test number or 0 to run all tests:\n");
            scanf("%d", &id);
        }
        else
            id = atoi(argv[1]);
    }
    else
        scanf("%d", &id);

    if (id == 0)
        for (int i = 0; i < SIZE; ++i)
            f[i]();
    else if (0 < id && id <= SIZE)
        f[id - 1]();
    else
        printf("Test %d not found.\n", id);

    return 0;
}

void test1()
{
    // Special point/point case
    int actual = f1(0,0,0,0,0,0);
    int expected = 1;

    if (actual == expected)
        printf("Test 1 : Pass\n");
    else
        printf("Test 1 : Failed (%d)\n", actual);
}

void test2()
{
    // Point/circle intersecting case
    int actual = f1(0,0,0,0,0,10);
    int expected = 1;

    if (actual == expected)
        printf("Test 2 : Pass\n");
    else
        printf("Test 2 : Failed (%d)\n", actual);
```

```
}

void test3()
{
    // Circle/point non-intersecting case
    int actual = f1(10,10,3,1,1,0);
    int expected = 0;

    if (actual == expected)
        printf("Test 3 : Pass\n");
    else
        printf("Test 3 : Failed (%d)\n", actual);
}

void test4()
{
    // Circle/circle non-intersecting case
    int actual = f1(0,1,3,5,1,1);
    int expected = 0;

    if (actual == expected)
        printf("Test 4 : Pass\n");
    else
        printf("Test 4 : Failed (%d)\n", actual);
}

void test5()
{
    // Circle/circle intersecting case
    int actual = f1(0,1,3,5,10,10);
    int expected = 1;

    if (actual == expected)
        printf("Test 5 : Pass\n");
    else
        printf("Test 5 : Failed (%d)\n", actual);
}

void test6()
{
    // Circle/circle intersecting-touching case 3*3 + 4*4 == 5*5
    int actual = f1(0,1,3,5,2,3);
    int expected = 1;

    if (actual == expected)
        printf("Test 6 : Pass\n");
    else
        printf("Test 6 : Failed (%d)\n", actual);
}

void test7()
{
    int actual = f2(0, 0, 0);
    int expected = 0; /*   = (0+0)*(0+0)   */

    if (actual == expected)
        printf("Test 7 : Pass\n");
    else
```

```
        printf("Test 7 : Failed (%d)\n", actual);
}

void test8()
{
    int actual = f2(10, 20, 0);
    int expected = 200;  /*   = (10+0)*(20+0)   */

    if (actual == expected)
        printf("Test 8 : Pass\n");
    else
        printf("Test 8 : Failed (%d)\n", actual);
}

void test9()
{
    int actual = f2(0, 0, 1);
    int expected = 1;  /*   = (0+0)*(0+0) + (0+1)*(0+1)   */

    if (actual == expected)
        printf("Test 9 : Pass\n");
    else
        printf("Test 9 : Failed (%d)\n", actual);
}

void test10()
{
    int actual = f2(1, 10, 2);
    int expected = 68;  /*   = (1+0)*(10+0) + (1+1)*(10+1) + (1+2)*(10+2) = 10 +
22 + 36   */

    if (actual == expected)
        printf("Test 10 : Pass\n");
    else
        printf("Test 10 : Failed (%d)\n", actual);
}
```

Compiling, executing, and testing

Run `make` with the default rule to bring program executable `main` up to date:

```
$ make
```

Or, directly test your implementation by running `make` with target `test`:

```
$ make test
```

If the `diff` command in the `test` rule is not silent, then one or more of your function definitions is incorrect and will require further work.

File-level documentation

Every **edited by student** source file *must* begin with a *file-level* documentation block. This documentation serves the purpose of providing a reader the purpose of this source file at some later point of time. It has simple format. This module will not use any documentation generator like `Doxygen`.

```
; ------------------------------------------------------------------
; File: functions.asm
; Project: CSD1100 Assignment 11
; Author: Vadim Surov, vsurov@digipen.edu
; Co-Author: Your name, email
; ...
; ------------------------------------------------------------------
```

Submission and automatic evaluation

1. In the course web page, click on the appropriate submission page to submit `functions.asm`.
2. Please read the following rubrics to maximize your grade. Your submission will receive:
    - `F` grade if your `functions.asm` doesn't compile with the given options.
    - `F` grade if your `functions.asm` doesn't link to create an executable.
    - Your implementation's output doesn't match correct output of the grader (you can see the inputs and outputs of the auto grader's tests). The auto grader will provide a proportional grade based on how many incorrect results were generated by your submission. `A+` grade if output of function matches correct output of auto grader.
    - A deduction of one letter grade for each incorrect, incomplete or missing documentation block in `functions.asm`. For example, if the automatic grader gave your submission an `A+` grade and one documentation block is missing, your grade will be later reduced from `A+` to `B+`.