Your names: Ben Smith, Jackson Schuetzle, Shreyhan Lakhina
Email: bxs566@case.edu
ID: 3559750

Problem Set - 5
Course: CSDS 337 - Compiler Design
Term: Spring 2024
Instructor: Dr. Vipin Chaudhary
Due Date: 15^{th} April, 2024

Number of hours delay for this Problem Set: 4.
Cumulative number of hours delay so far: 56.

I discussed this homework with: My group.

Problem 1 - 10 points
The C code to compute Fibonacci numbers recursively is shown below. Suppose that the activation record for f includes the following elements in order: (return value, argument n, local s, local t); there will normally be other elements in the activation record as well. The questions below assume that the initial call is $f(5)$.

```
int f(int n) {
    int t, s;
    if (n < 2) return 1;
    s = f(n-1);
    t = f(n-2);
    return s+t;
}
```

a Show the complete activation tree.

a What does the stack and its activation records look like the first time $f(1)$ is about to return?

Solution:

a

a

Problem 2 - 10 points
In a language that passes parameters by reference, there is a function $f(x; y)$ that does the following:
$x = x + 1; \quad y = y + 2; \quad return \quad x + y;$
If a is assigned the value 3, and then $f(a; a)$ is called, what is returned?

Solution: 12

Problem 3 - 10 points
The C function f is defined by:

```
int f(int x, *py, **ppz) {
    **ppz += 1; *py += 2; x += 3; return x+*py+**ppz;
}
```

Variable a is a pointer to b ; variable b is a pointer to c , and c is an integer currently with value 4. If

we call $f(c;b;a)$, what is returned?

Solution: $5 + 6 + 7 = 18$

Problem 4 - 70 points
Write a C or C++ program to showcase some of the features of Clang and LLVM. You want to write a C program that will best showcase the features of various optimizations. Use time function inside the code to measure program time for performance measurements. Use well known algorithms (sorting, searching, numerical computation, etc.) and state where they are used if the code is more obscure.

1. For a given architecture, compare the time it takes for different types of compiler optimization. Also use compiler option to minimize code size. Compare the codes (target assembly code) for all cases to isolate the types of optimizations implemented.

2. Show at least three different optimizations in your code that are affected by compiler optimizations. Use Transform (most important), and Utility Passes (https://llvm.org/docs/Passes.html) that are applicable to your example with complete makefile (or command line script) for the passes and appropriate documentation of results. You may also want to refer to clang.llvm.org

---

Deliverables: A zip file containing

- **File with your code**
- **README text file with directions to run the various programs**
- **Report showing original code and optimized code snippets and the particular optimization used; results (table of performance). Similarily for the bonus portion.**
  - **Full names and Case IDs**
  - **(not required) any special notes about your implementation the grader should be aware of**
