SI 413 Fall 2021: Homework 11

Due Monday, November 22

Your name:

Citations and collaborators:

Comments, suggestions, or questions for your instructor:

Fill out the first row of the table on a 0-5 scale before turning in.

This rubric is also available on the website under "Admin":

- **5**: Solution is completely correct, concisely presented, and neatly written.
- **4**: The solution is mostly correct, but one or two minor details were missed, or the presentation could be better.
- **3**: The main idea is correct, but there are some significant mistakes. The presentation is somewhat sloppy or confused.
- **2**: A complete effort was made, but the result is mostly incorrect.
- **1**: The beginning of an attempt was made, but the work is clearly incomplete.
- **0**: Not submitted.

Problem	1	2	3	Total
Self-assessment				
Final assessment				

1 Compile this

Consider the following C function (or download ex1.c)

```
int foo(int x) {
  if ((x+3)*4 == 20) {
    return 14;
  }
  else {
    return x*7;
  }
}
```

a) Draw the AST for the body of this function. Make sure each AST node is clearly labeled.

b) Write LLVM IR instructions for this C function. Use registers only; no load or store.

Label each line of LLVM IR with the corresponding AST node that should emit that line in a compilation.

Note, you should only need to use the following LLVM commands: add, mul, icmp eq, br, ret.

2 Decompile that

Take a look at this LLVM IR program (or download decomp.ll):

```
declare i32 @getchar()
declare i32 @putchar(i32)

define i32 @main() {
  %x1 = call i32 @getchar()
  %x2 = alloca i32, align 4
  store i32 0, i32* %x2
  br label %L1

L1:
  %x3 = load i32, i32* %x2
  %x4 = icmp slt i32 %x3, 10
  br i1 %x4, label %L2, label %L3

L2:
  %x5 = call i32 @putchar(i32 %x1)
  %x6 = add i32 %x3, 1
  store i32 %x6, i32* %x2
  br label %L1

L3:
  %x7 = call i32 @putchar(i32 10)
  ret i32 0
}
```

Write a C program which should do the same thing that this program does.

(Note that `getchar` and `putchar` are standard library functions found in `stdio.h`.)

3 Clanging

Here is a small C program, collatz.c:

```
#include <stdio.h>

int main() {
  int x;
  scanf("␣%d", &x);
  while (x > 1) {
    if (x % 2 == 0) {
      x /= 2;
    }
    else {
      x = 3*x + 1;
    }
    printf("%d\n", x);
  }
  return 0;
}
```

We can compile this to LLVM by running the command

```
clang -emit-llvm -S collatz.c
```

The resulting collatz.ll file is shown on the next page.

Your tasks are to investigate the "real" clang LLVM output and answer the following questions:

a) The variable `x` in the C program is stored in stack memory according to the LLVM instructions. **Which register** holds the address of `x`?

b) Circle the line or lines of LLVM IR that correspond to the if condition expression `x % 2 == 0`.

c) Draw the control flow graph (CFG) for this program.

```
; ModuleID = 'collatz.c'
source_filename = "collatz.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-n8:16:32:64-S128
target triple = "x86_64-pc-linux-gnu"

@.str = private unnamed_addr constant [4 x i8] c" %d\00", align 1
@.str.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1

; Function Attrs: noinline nounwind optnone uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %3 = call i32 (i8*, ...) @__isoc99_scanf(i8* getelementptr inbounds ([4 x i8],
[4 x i8]* @.str, i64 0, i64 0), i32* %2)
  br label %4

4:                                                ; preds = %18, %0
  %5 = load i32, i32* %2, align 4
  %6 = icmp sgt i32 %5, 1
  br i1 %6, label %7, label %21

7:                                                ; preds = %4
  %8 = load i32, i32* %2, align 4
  %9 = srem i32 %8, 2
  %10 = icmp eq i32 %9, 0
  br i1 %10, label %11, label %14

11:                                               ; preds = %7
  %12 = load i32, i32* %2, align 4
  %13 = sdiv i32 %12, 2
  store i32 %13, i32* %2, align 4
  br label %18

14:                                               ; preds = %7
  %15 = load i32, i32* %2, align 4
  %16 = mul nsw i32 3, %15
  %17 = add nsw i32 %16, 1
  store i32 %17, i32* %2, align 4
  br label %18

18:                                               ; preds = %14, %11
  %19 = load i32, i32* %2, align 4
  %20 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
[4 x i8]* @.str.1, i64 0, i64 0), i32 %19)
  br label %4

21:                                               ; preds = %4
  ret i32 0
}

declare dso_local i32 @__isoc99_scanf(i8*, ...) #1

declare dso_local i32 @printf(i8*, ...) #1

attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-divide-sqrt-fp-math"
attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-tail-calls"="fal

!llvm.module.flags = !{!0}
```

```
!llvm.ident = !{!1}

!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{!"clang version 10.0.0-4ubuntu1 "}
```