第一次实验 “温故知新”

在实验开始之前，有三句话送给大家：

- Stay Hungry, Stay Foolish. （不要满足现状）
- Do not set limit for yourself. （不要停止优化，任何程序都有重整优化的空间）
- Courage is going from failure to failure without losing enthusiasm. （不要害怕 debug）

实验内容

1. 众所周知，LC3 机器是没有可以直接使用的乘法指令的。本次实验任务是实现乘法，写出对应程序机器码。两个运算数分别放置于 R0 和 R1，结果需要存储到 R7，其他寄存器状态我们不做限制（即不限结束状态）。初始状态：R0和R1存放待计算数，其余寄存器全部为0。
2. 请评估自己程序的代码行数、完成实验功能所需要执行的指令数，并将统计方法写在报告中。
3. 要求提交两个版本的代码。L版本尽量编写更少的代码行数，P版本尽量让程序执行更少的指令。

评分标准

1）程序正确性 （60%）

- ！！！ 请严格按照实验要求将结果存储在R7！！！
- L版本程序通过所有测试可得30%
- P版本程序通过所有测试可得30%
- 要求与c语言short型整数乘法结果相同

2）实验报告（20%），要求图文并茂

- L程序和P程序是如何设计的?
- L版本程序最初用了多少行？最终版本用了多少行?
- P版本程序最初用了多少条指令？最终完成版本用了多少条指令?
- ... .

3）L版本的代码行数小于14行（10%），否则分数按照下列公式计算：

- $Score_{vesion\ l} = 10 \times min(2 \times \frac{\texttt{Lines of TA's program}}{\texttt{Lines of your program}}, 1)$
- 其中，**Lines of TA's program = 7**。
- 程序的指令行数（注意是行号代表的行数，而非程序执行的指令数）在TA's program的两倍以内即可得满分。

4）P版本在所有测试用例上执行的平均指令条数需要小于130条（10%），否则分数按照下列公式计算：

- $Score_{vesion\ p} = 10 \times min(2 \times \frac{\texttt{Average time of TA's program}}{\texttt{Average time of your program}}, 1)$
- 其中，Time of TA's program在**下方测试样例**上执行的平均指令数为70，在**全部测试样例上** (包括未给出的)执行的平均指令数为65。
- 我们采用“执行过的指令数”作为时间衡量标准。程序执行的指令数在TA's program的两倍以内即可得满分。

测试样例

- 需要对下列case的测试：
  - 计算 1 * 1
  - 计算 5 * 4000
  - 计算 4000 * 5
  - 计算 -500 * 433 （刻意溢出）
  - 计算 -114 * -233
- 评估程序正确性、计算P时，会有其他的样例

注意事项

- 必须使用LC3 tool完成实验
- 自己使用LC3 tool时，程序的第一条指令需要指定在内存中的位置，比如0x3000，并以HALT指令结束。以下是一个例子，计算R7 = R0 + R1：

```
; start the program at location x3000
0011 0000 0000 0000

;your own program
0101 001 001 1 00000 ;R1 = 0, clear R1
0101 000 000 1 00000 ;R0 = 0, clear R0
0001 000 000 1 00001 ;R0 = 1
0001 001 001 1 00001 ;R1 = 1
0001 111 000 000 001 ;R7 = R0 + R1

; halt
1111 0000 00100101
```

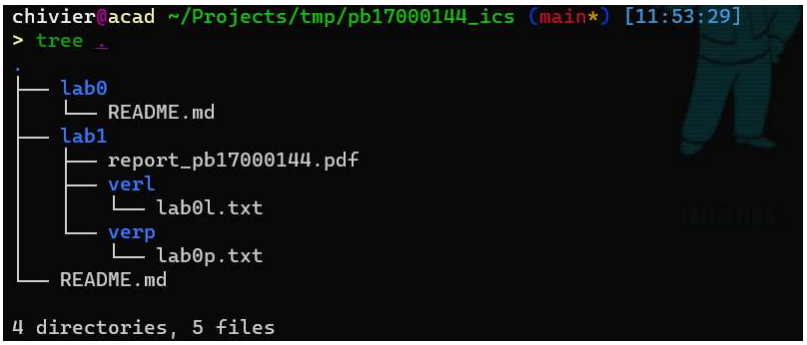
- ！！！提交的代码只需要保留your own program的部分，并去掉对寄存器初始化的过程！！！比如上述程序，你提交的代码只需要：

```
0001 111 000 000 001
```

此时，统计的代码行数为1行。评分标准中提到Lines of TA's program = 7就是掐头去尾统计得到的。

提交说明

完成的程序应当在目录下呈现如图所示结构：



lab1目录为本次实验内容，其中 report_pbxxxxxxxx.pdf 为报告，限制pdf格式，其他格式默认不予接受。

verl目录存放L版本代码，verp目录存放P版本代码。

提交方式

安老师班

Git提交

苗老师、张老师班

请将lab1文件夹打包，压缩包改为 姓名_学号_lab1.zip/tar/rar/...

之后上传到坚果云，链接会放在课程主页上。

截止日期：待定（期中考试之后），具体时间后续在课程主页上更新

Lab2 Lending Your Name

From this lab, Prof. An ask me to write guide in English.

Task

In lab 2, the task is very simple. Store the n-th number of a sequence into register R7.

> In mathematics, the Fibonacci numbers, commonly denoted Fn, form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2).
>
> -- From Wikipedia

As the Fibonacci sequence grows quite rapidly, we made a little difference with the origin Fibonacci sequence:

F(0) = 1

F(1) = 1

F(2) = 2

F(n) = (F(n-1) + 2 * F(n-3)) mod 1024 (1 <= n <= 16384)

In the beginning, n is stored in R0. All other registers' initial values are 0.

Your tasks are:

- store F(n) in register R7
- Divide your student number into four equal segments, labelling them with a, b, c and d. For example, TA's student ID is PB17000144, so a = 17, b = 0, c = 1, d=44. Store value of F(a), F(b), F(c) and F(d) at the end of your code with '.FILL' pseudo command.

Score

1) Corectness (60%)

If L-version passes all the test cases, you can get 30%

2) Report (20%)

- Same as lab1

3) L-Score(20%)

- The number of lines of instructions in the program (note that the line number represents the number of lines, not the number of instructions executed in the program) is within twice the number of TA's program for full credit.
- If the number of lines is larger than Lines of TA's program, this part of score is given by the following formula:
  $Score_{vesion\ l} = 10 \times min(2 \times \frac{\texttt{Lines of TA's program}}{\texttt{Lines of your program}}, 1)$
- **Lines of TA's program = 18**

Tips

From this experiment, each experiment requires using assembly code. Here are some notifications.

- Capitalize keywords (For example, use "AND, ADD..." instead of "and, add...")
- Comma division of operands
- Decimal constants start with #, hexadecimal with lowercase x
- Write comments when necessary
- Do not to modify the value(x3000) in ORIG section unless there are special needs

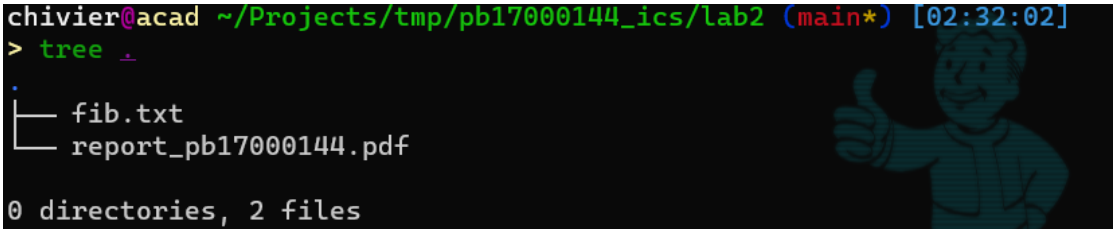
Here is an example of adding two numbers together,

```
.ORIG x3000
LD R1, NUMBER1
LD R2, NUMBER2
ADD R0, R1, R2 ; Add 2 numbers together
ST R0, NUMBERA
NUMBERA .FILL #0
NUMBER1 .FILL #13
NUMBER2 .FILL x1C
.END
```

ORIG and END pseudo commands will not be counted. The lines of the program above is 7.

Submission

The completed program should be structured in the directory as shown in the figure,



Prof. An

Use git to submit your program.

Prof. Miao & Zhang

Please pack the lab2 folder and zip it to Name_ID_lab2.zip/tar/rar/...

After that upload to the nut cloud and the link will be placed on the course homepage.
Please pack the lab2 folder and zip it to

2021.12.12 21:00 (UTC+8 China Standard Time)

(Since it is lab2, the deadline onely consists of number 1 and number 2).

Lab3 Better Angels

Task

task 1: read and understand

In lab 3, you will get a piece of machine code in 'foo.txt'. Your first task is to translate machine code into assembly code.

Store your program in 'translate.txt'.

task 2: guess

This code is some other's program in lab2. Guess the owner of the program by the last 4 lines of the program.

Write down the owner's student id in 'id.txt'.

task 3: optimize

The code in lab2 is a L-version program. Of course it's performance is not very well. In this part, you need to optimize other's program.

(Rewriting is also a legitimate optimization method)

Store the optimized code in 'optimized.txt'.

task 4(optional): feedback

Contact the owner, say anything you like. (Don't swear. If the other party's program is too difficult to understand, please also keep polite.)

Score

read (20%)

After finishing part 1, you will get 20% score.

Guess (20%)

Getting correct student id will earn 20% score.

Optimizeing(40%)

In this section, a score of 40% can be achieved if optimization is completed.

Here we will test the following 10 data sets.

N = 24, 144, 456, 1088, 1092, 2096, 4200, 8192, 12000, 14000

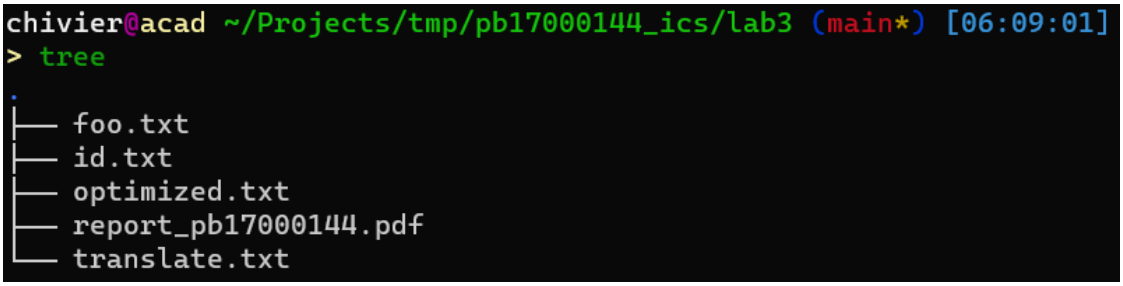
The ratio of the average number of execution cycles before and after these 10 sets of optimizations will be used as a prize(the ratio has nothing to do with your score), and the student with the largest optimization will receive a surprise gift.

Report（20%）

- Same as lab1

Submission

The completed program should be structured in the directory as shown in the figure,



Prof. An

Use git to submit your program.

Prof. Miao & Zhang

Please pack the lab3 folder and zip it to Name_ID_lab3.zip/tar/rar/...

After that upload to the nut cloud and the link will be placed on the course homepage.

2021.12.18 23:00 (UTC+8 China Standard Time)

Lab4 Reveal Yourself

Task

Have you ever met such a situation that your disk is broken and data is lost?

OK, in this lab, your task is to find out the missing bit in a program.

task 1

In the first part, 4 bits in the code are missing. You need to find out the missing bits. Before the program starts, values in the registers(except PC) are 0. At the end of the program, the register status is as follow.

```
R0 = 5, R1 = 0, R2 = 300f, R3 = 0
R4 = 0, R5 = 0, R6 = 0, R7 = 3003
```

The program is stored in task1.txt.

Store the fixed program in 'recur.txt'.

task 2

The second part of the program is used to calculate the remainder. 15 bits int the code are missing. For 2,4,6,8 the remainder is easy to calculate, but for 7 the remainder needs some skills. Here is a quicker way to do the remainder for 7.

Hint: The program uses the term "divide by 8".

The program is stored in task2.txt.

Store the fixed program in 'mod.txt'.

Score

read (40%)

After finishing part 1, you will get 40% score.

Guess (40%)

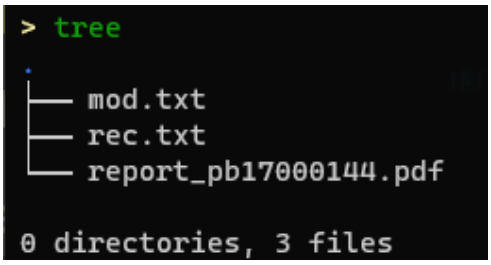
After finishing part 2, you will get 40% score.

Report （20%)

- Same as lab1

Submission

The completed program should be structured in the directory as shown in the figure,



Prof. An

Use git to submit your program.

Prof. Miao & Zhang

Please pack the lab4 folder and zip it to Name_ID_lab4.zip/tar/rar/...

After that upload to the nut cloud and the link will be placed on the course homepage.

2021.12.24 23:00 (UTC+8 China Standard Time)

Merry Christma

Lab5 Riddle

Task

For this experiment you need to functionally reproduce a c++ programs.

Program A:

```
int judge(int r0) {
    int i = 2;
    r1 = 1;
    while (i * i <= r0) {
        if (r0 % i == 0) {
            r1 = 0;
            break;
        }
        i++;
    }
    return r1;
}
```

Your program should follow a specific framework:

```
.ORIG x3000
... ; TO BE DONE
HALT
JUDGE ... ; TO BE DONE
... ; TO BE DONE
RET
... ; TO BE DONE
.END
```

r0(an integer, $0 \leq r0 \leq 10000$) is given before program executes(just like lab1), and store the final result in r1. (no need to print out with TRAP)

Score

1) Corectness (60%)

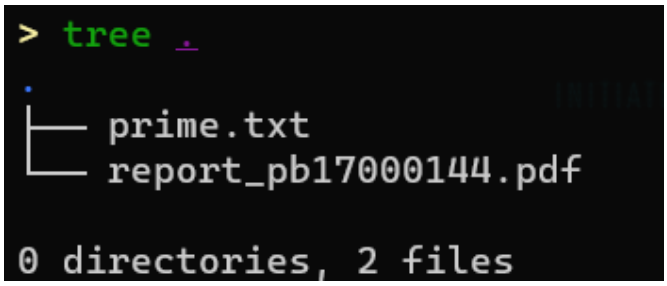
If your program passes all the test cases, you can get 60%

2) Report (40%)

Same as lab1

Submission

The completed program should be structured in the directory as shown in the figure,



Prof. An

Use git to submit your program.

Prof. Miao & Zhang

Please pack the lab5 folder and zip it to Name_ID_lab5.zip/tar/rar/...

After that upload to the nut cloud and the link will be placed on the course homepage.

2021.12.31 21:00 (UTC+8 China Standard Time)

Happy New Year

Lab6 Learn from the past

The last lab might be the simplest one. Use a high-level programming language (e.g. C, Python, C++) to implement all the code that has been written before. The algorithm needs to be consistent with what was used before, e.g. a replication of the first experiment cannot be implemented with just one line of multiplication.

Program list:

- lab0l (lab1 L version)
- lab0p (lab1 P version)
- fib (lab2 fibonacci)
- fib-opt (lab3 fibonacci)
- rec (lab4 task1 rec)
- mod (lab4 task2 mod)
- prime (lab5 prime)

For this experiment, you should think about the following questions:

1. how to evaluate the performance of your own high-level language programs
2. why is a high-level language easier to write than LC3 assembly
3. what instructions do you think need to be added to LC3? (You can think about the previous experiments and what instructions could be added to greatly simplify the previous programming)
4. is there anything you need to learn from LC3 for the high-level language you use?

Score

1) Corectness (50%)

Your program accounts for half the score.

2) Report (50%)

Your report accounts for another half the score.

2022.1.7 23:00 (UTC+8 China Standard Time)

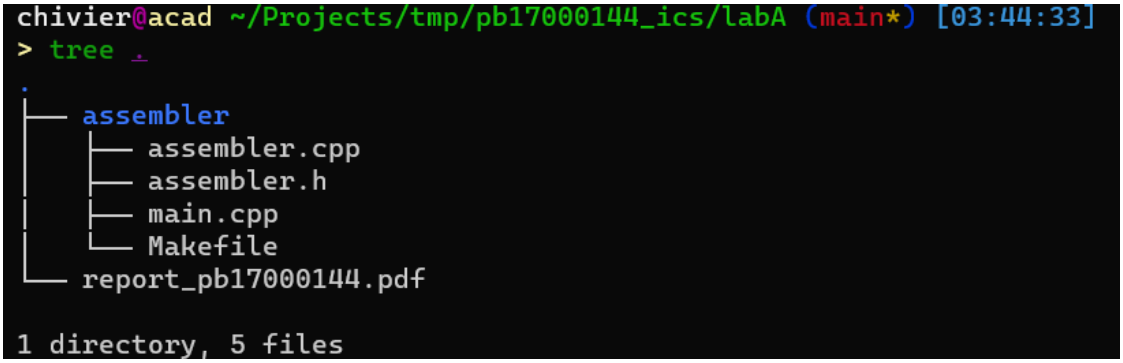
Thank you for taking this course

<!--

- @Author : Chivier Humber
- @Date : 2021-10-22 12:37:32
- @LastEditors : Chivier Humber
- @LastEditTime : 2021-11-23 15:41:13
- @Description : file content
  -->

Assembler Lab

Framework



Tasks

1. learn how to use Makefile
2. read the code, understand the framework of the program, and train the ability to read the program
3. replace all `TO BE DONE` in the code with the correct code

Score

The correct code will get 95% of the marks for this experiment.

Report accounts for 5% of the score.

<!--

- @Author : Chivier Humber
- @Date : 2021-10-22 19:10:10
- @LastEditors : Chivier Humber
- @LastEditTime : 2021-11-23 15:46:58
- @Description : file content
  -->

Simulation lab

Framework

```
chivier@acad ~/Projects/ICS2021-Lab/labS/simulator (main*) [03:50:36]
> tree .
.
├── CMakeLists.txt
├── include
│   ├── common.h
│   ├── memory.h
│   ├── register.h
│   └── simulator.h
└── src
    ├── main.cpp
    ├── memory.cpp
    ├── register.cpp
    └── simulator.cpp

2 directories, 9 files
```

Tasks

1. learn how to use Cmake
2. read the code, understand the framework of the program, and train the ability to read the program
3. replace all `TO BE DONE` in the code with the correct code

Score

The correct code will get 95% of the marks for this experiment.

Report accounts for 5% of the score.