

中国科学技术大学计算机学院  
《计算机系统详解》实验报告



实验题目：CSAPP\_labs

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020 年 6 月 24 日

计算机实验教学中心制

2019 年 09 月

## 实验目的

1. 结合课上所学，完成 CSAPP 课程实验 Data Lab，Bomb Lab，Cache Lab。

## 实验环境

1. PC 一台
2. Windows 系统
3. Ubuntu

## Data Lab

本实验内容为根据要求用有限的操作实现下列函数：

Name	Description	Rating	Max ops
bitXor(x, y)	$x \oplus y$ using only & and ~.	1	14
tmin()	Smallest two's complement integer	1	4
isTmax(x)	True only if x is largest two's comp. integer.	1	10
allOddBits(x)	True only if all odd-numbered bits in x set to 1.	2	12
negate(x)	Return -x with using - operator.	2	5
isAsciiDigit(x)	True if $0 \leq x \leq 127$ .	3	15
conditional	Same as $x ? y : z$	3	16
isLessOrEqual(x, y)	True if $x \leq y$ , false otherwise	3	24
logicalNeg(x)	Compute !x without using ! operator.	4	12
howManyBits(x)	Min. no. of bits to represent x in two's comp.	4	90
floatScale2(uf)	Return bit-level equiv. of $2 * f$ for f.p. arg. f.	4	30
floatFloat2Int(uf)	Return bit-level equiv. of (int) f for f.p. arg. f.	4	30
floatPower2(x)	Return bit-level equiv. of $2.0^x$ for integer x.	4	30

表 1

核心代码部分详见[链接](#)。

## Bomb Lab

根据实验要求，需要找到对应的 6 个字符串。而提供的源文件中并不包含所需的信息。因此，需要通过反汇编来实现。

首先输入 “`objdump -d bomb > bomb.txt`”，得到 bomb 的汇编代码。进而通过阅读汇编代码，并使用 gdb 进行调试，完成炸弹的解码过程。

第一个密码，通过下列汇编代码，及 strings\_not\_equal 的代码可知：密码为存在地址 0x402400 的字符串。

000000000400ee0 <phase\_1>:

```
400ee0: 48 83 ec 08      sub    $0x8,%rsp
400ee4: be 00 24 40 00   mov    $0x402400,%esi
400ee9: e8 4a 04 00 00   callq 401338 <strings_not_equal>
400eee: 85 c0            test   %eax,%eax
400ef0: 74 05            je     400ef7 <phase_1+0x17>
400ef2: e8 43 05 00 00   callq 40143a <explode_bomb>
400ef7: 48 83 c4 08      add    $0x8,%rsp
400efb: c3              retq
```

在 gdb 中输入 “`x/ls 0x402400`”，得到密码 “Border relations with Canada have never been better.”

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
BOOM!!!
The bomb has blown up.
Inferior 1 (process 78) exited with code 010]
(gdb)
(gdb) quit
hyx@DESKTOP-LP1A2G2:~/Github/CSAPP_labs/bomb$ gdb bomb
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) x/ls 0x402400
0x402400: "Border relations with Canada have never been better."
(gdb) run
Starting program: /mnt/c/Users/Asus/Documents/Github/CSAPP_labs/bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
```

图 1

第二个密码中，通过“read\_six\_numbers”可知密码为 6 个数字。再阅读下列代码后可知，密码为首项为 1，公比为 2 的等比数列。

400f0a:	83 3c 24 01	cmpl	\$0x1,(%rsp)
400f0e:	74 20	je	400f30 <phase_2+0x34>
400f10:	e8 25 05 00 00	callq	40143a <explode_bomb>
400f15:	eb 19	jmp	400f30 <phase_2+0x34>
400f17:	8b 43 fc	mov	-0x4(%rbx),%eax
400f1a:	01 c0	add	%eax,%eax
400f1c:	39 03	cmp	%eax,(%rbx)
400f1e:	74 05	je	400f25 <phase_2+0x29>
400f20:	e8 15 05 00 00	callq	40143a <explode_bomb>
400f25:	48 83 c3 04	add	\$0x4,%rbx
400f29:	48 39 eb	cmp	%rbp,%rbx
400f2c:	75 e9	jne	400f17 <phase_2+0x1b>
400f2e:	eb 0c	jmp	400f3c <phase_2+0x40>
400f30:	48 8d 5c 24 04	lea	0x4(%rsp),%rbx
400f35:	48 8d 6c 24 18	lea	0x18(%rsp),%rbp
400f3a:	eb db	jmp	400f17 <phase_2+0x1b>

故密码为 1 2 4 8 16 32.

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
The bomb has blown up.
hyx@DESKTOP-LP1A2G2:~/Github/CSAPP_labs/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
```

图 2

对于第三个密码，由下列代码知，输入的值个数应大于等于 2：

```
400f5b:  e8 90 fc ff ff      callq 400bf0 <__isoc99_sscanf@plt>
400f60:  83 f8 01            cmp    $0x1,%eax
400f63:  7f 05              jg     400f6a <phase_3+0x27>
400f65:  e8 d0 04 00 00      callq 40143a <explode_bomb>
```

再由下列代码知，第一个数应小于等于 7：

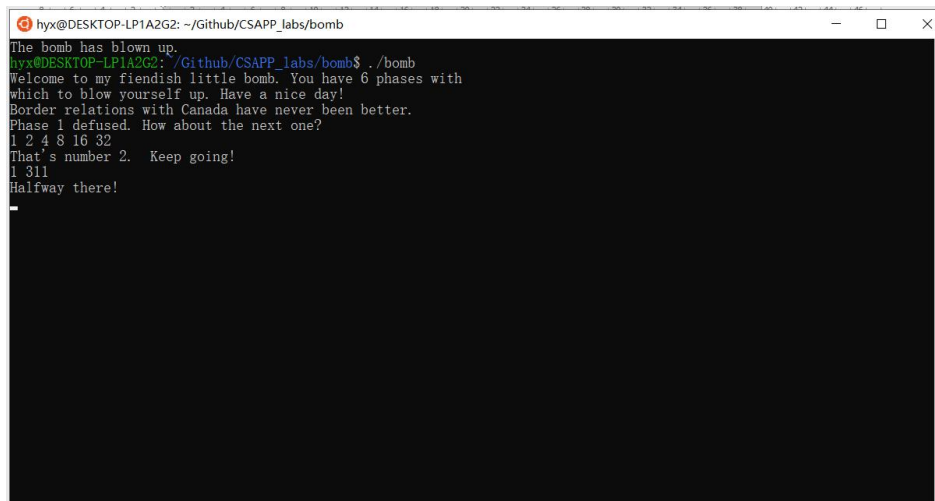
```
400f6a:  83 7c 24 08 07      cmpl   $0x7,0x8(%rsp)
400f6f:  77 3c              ja     400fad <phase_3+0x6a>
400f71:  8b 44 24 08         mov    0x8(%rsp),%eax
400f75:  ff 24 c5 70 24 40 00 jmpq    *0x402470(,%rax,8)
```

不妨令第一个数为 1，跳转到 0x400fb9：

```
400fb9:  b8 37 01 00 00      mov    $0x137,%eax
400fbe:  3b 44 24 0c         cmp    0xc(%rsp),%eax
400fc2:  74 05              je     400fc9 <phase_3+0x86>
400fc4:  e8 71 04 00 00      callq 40143a <explode_bomb>
400fc9:  48 83 c4 18         add    $0x18,%rsp
400fcd:  c3                retq
```

观察可知，须令第二个数为 0x137 即 311，即可返回。

故第三个密码为 1 311。



```

hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
The bomb has blown up.
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
1 311
Halfway there!

```

图 3

对于第四个密码，阅读下列代码知，须输入两个数：

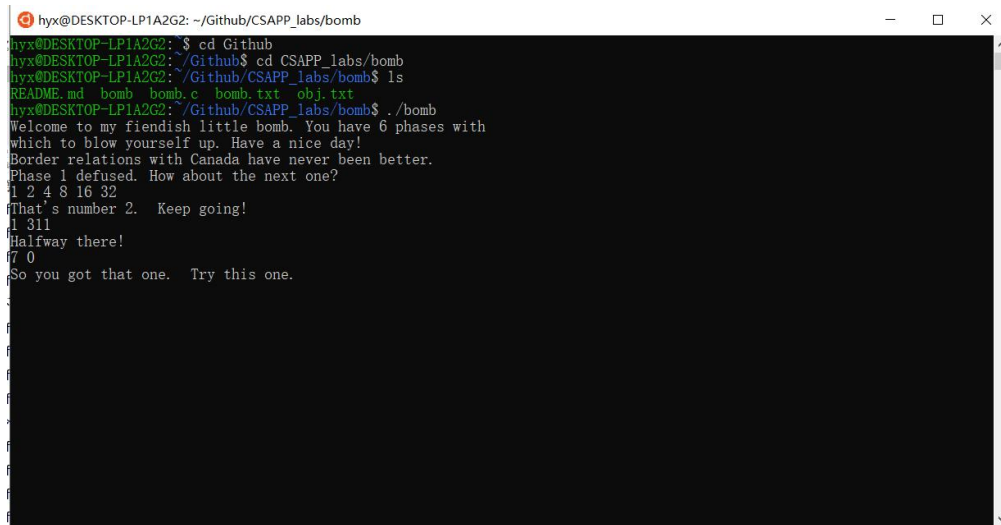
```
401024:  e8 c7 fb ff ff      callq 400bf0 <__isoc99_sscanf@plt>
401029:  83 f8 02            cmp    $0x2,%eax
40102c:  75 07              jne    401035 <phase_4+0x29>
```

而由如下这段知，第一个数应小于等于 14：

```
40102e:  83 7c 24 08 0e      cmpl   $0xe,0x8(%rsp)
401033:  76 05              jbe    40103a <phase_4+0x2e>
401035:  e8 00 04 00 00      callq 40143a <explode_bomb>
40103a:  ba 0e 00 00 00      mov    $0xe,%edx
40103f:  be 00 00 00 00      mov    $0x0,%esi
401044:  8b 7c 24 08         mov    0x8(%rsp),%edi
```

跳转后进入 func4，由 func4 内代码判断，输入的第一个数应为 7。而由 phase\_4 后半部分代码，知第二个数必须为 0：

40104d:	85 c0	test	%eax,%eax
40104f:	75 07	jne	401058 <phase_4+0x4c>
401051:	83 7c 24 0c 00	cmpl	\$0x0,0xc(%rsp)
401056:	74 05	je	40105d <phase_4+0x51>
401058:	e8 dd 03 00 00	callq	40143a <explode_bomb>
40105d:	48 83 c4 18	add	\$0x18,%rsp
401061:	c3	retq	



紧接着第 5 个密码，由下列指令知，字符串长度为 6:

此处为一个循环，内容为对字符串进行逐字符操作：

40108b:	0f b6 0c 03	movzbl (%rbx,%rax,1),%ecx
40108f:	88 0c 24	mov %cl,(%rsp)
401092:	48 8b 14 24	mov (%rsp),%rdx
401096:	83 e2 0f	and \$0xf,%edx
401099:	0f b6 92 b0 24 40 00	movzbl 0x4024b0(%rdx),%edx
4010a0:	88 54 04 10	mov %dl,0x10(%rsp,%rax,1)
4010a4:	48 83 c0 01	add \$0x1,%rax
4010a8:	48 83 f8 06	cmp \$0x6,%rax
4010ac:	75 dd	jne 40108b <phase_5+0x29>

```
4010bd:  e8 76 02 00 00    callq 401338 <strings_not_equal>
4010c2:  85 c0             test   %eax,%eax
4010c4:  74 13            je     4010d9 <phase_5+0x77>
4010c6:  e8 6f 03 00 00    callq 40143a <explode bomb>
```

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) break *0x400e49
Breakpoint 1 at 0x400e49: file /usr/include/x86_64-linux-gnu/bits/stdc2.h, line 105.
(gdb) run
Starting program: /mnt/c/Users/Asus/Documents/Github/CSAPP_labs/bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.

Breakpoint 1, 0x0000000000400e49 in printf (__fmt=<optimized out>) at /usr/include/x86_64-linux-gnu/bits/stdc2.h:105
warning: Source file is more recent than executable.
105
}
(gdb) print 0x40245e
$1 = 4203614
(gdb) x/1s 0x40245e
0x40245e: "flyers"
(gdb)
```

图 5

```
选择hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) break *0x400e49
Breakpoint 1 at 0x400e49: file /usr/include/x86_64-linux-gnu/bits/stdc2.h, line 105.
(gdb) run
Starting program: /mnt/c/Users/Asus/Documents/Github/CSAPP_labs/bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.

Breakpoint 1, 0x0000000000400e49 in printf (__fmt=<optimized out>) at /usr/include/x86_64-linux-gnu/bits/stdc2.h:105
warning: Source file is more recent than executable.
105
}
(gdb) print 0x40245e
$1 = 4203614
(gdb) x/1s 0x40245e
0x40245e: "flyers"
(gdb) print 0x4024b0
$2 = 4203696
(gdb) print (char*)0x4024b0
$3 = 0x4024b0 <array> "maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?"
(gdb) print (char*)0x4024b0
$4 = 0x4024b0 <array> "maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?"
(gdb) print (char*)0x4024b0
$5 = 0x4024b0 <array> "maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?"
(gdb)
```

图 6

目标字符串为 `flyers`，而操作后的字符串中每个值，为操作前的值与 `0xf` 与后得到的值 `a` 对应于首地址为 `0x4024b0` 的字符数组的第 `a` 个值。因此，输入的字符串的 ASCII 码后四位(二进制)分别应为 9, f, e, 5, 6, 7。故 `9?>567` 可作为第 5 个密码。

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
The bomb has blown up.
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
1 311
Halfway there!
7 0
So you got that one. Try this one.
9?>567
BOOM!!!
The bomb has blown up.
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
1 311
Halfway there!
7 0
So you got that one. Try this one.
9?>567
Good work! On to the next...
```

图 7

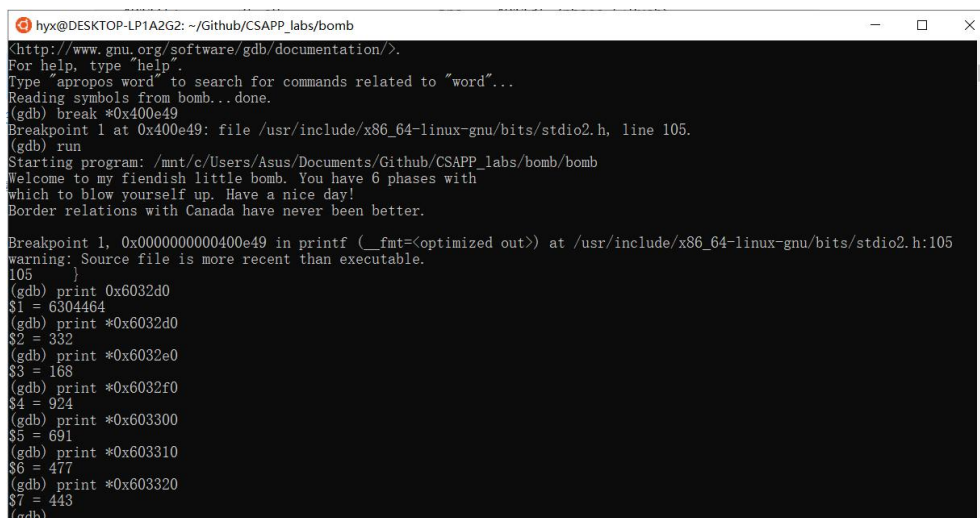
最后一个密码，代码量较大，从开头可看出，还是读取 6 个数字：  
401106: e8 51 03 00 00 callq 40145c <read\_six\_numbers>



读取完后，便是二重循环，判断输入的数范围为 1-6，且各不相同：

```
40110b: 49 89 e6      mov    %rsp,%r14
40110e: 41 bc 00 00 00 00 mov    $0x0,%r12d
401114: 4c 89 ed      mov    %r13,%rbp
401117: 41 8b 45 00    mov    0x0(%r13),%eax
40111b: 83 e8 01      sub    $0x1,%eax
40111e: 83 f8 05      cmp    $0x5,%eax
401121: 76 05        jbe    401128 <phase_6+0x34>
401123: e8 12 03 00 00 callq  40143a <explode_bomb>
401128: 41 83 c4 01    add    $0x1,%r12d
40112c: 41 83 fc 06    cmp    $0x6,%r12d
401130: 74 21        je     401153 <phase_6+0x5f>
401132: 44 89 e3      mov    %r12d,%ebx
401135: 48 63 c3      movslq %ebx,%rax
401138: 8b 04 84      mov    (%rsp,%rax,4),%eax
40113b: 39 45 00      cmp    %eax,0x0(%rbp)
40113e: 75 05        jne    401145 <phase_6+0x51>
401140: e8 f5 02 00 00 callq  40143a <explode_bomb>
401145: 83 c3 01      add    $0x1,%ebx
401148: 83 fb 05      cmp    $0x5,%ebx
40114b: 7e e8        jle    401135 <phase_6+0x41>
40114d: 49 83 c5 04    add    $0x4,%r13
401151: eb c1        jmp    401114 <phase_6+0x20>
```

之后对各个数进行 7-x 操作。



```
hyxx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) break *0x400e49
Breakpoint 1 at 0x400e49: file /usr/include/x86_64-linux-gnu/bits/stdio2.h, line 105.
(gdb) run
Starting program: /mnt/c/Users/Asus/Documents/Github/CSAPP_labs/bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.

Breakpoint 1, 0x0000000000400e49 in printf (_fmt=<optimized out>) at /usr/include/x86_64-linux-gnu/bits/stdio2.h:105
warning: Source file is more recent than executable.
105 }
(gdb) print 0x6032d0
$1 = 6304464
(gdb) print *0x6032d0
$2 = 332
(gdb) print *0x6032e0
$3 = 168
(gdb) print *0x6032f0
$4 = 924
(gdb) print *0x603300
$5 = 691
(gdb) print *0x603310
$6 = 477
(gdb) print *0x603320
$7 = 443
(gdb)
```

图 8

根据后半部分代码的要求，要使经 7-x 操作过后的数列对应到 0x6032d0 到 0x603320 的顺序为从大到小。而地址 0x6032d0 到 0x603320 上对应的数为，332，168，924，691，477，443。故从大到小顺序为 3，4，5，6，1，2。而 7-x 操作前的数列应为 4，3，2，1，6，5。炸弹拆解成功！

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
1 311
Halfway there!
7 0
So you got that one. Try this one.
9?>567
Good work! On to the next...
4 3 2 1 5 6

BOOM!!!
The bomb has blown up.
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2. Keep going!
1 311
Halfway there!
7 0
So you got that one. Try this one.
9?>567
Good work! On to the next...
4 3 2 1 5 6
Congratulations! You've defused the bomb!
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/bomb$
```

图 9

## Cache Lab

本实验的第一部分为设计一个 cache 的模拟器,并根据模拟参数( $s$ (索引位个数), $E$ (行数), $b$ (块的大小)),输出仿真结果。

实验的核心代码详见[链接](#)。

编译运行后,结果如下:

```
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ls
Makefile  cachelab.c  csim-ref  driver.py  test-trans.c  traces
README.md cachelab.h  csim.c    test-csim  tracegen.c    trans.c
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ make
gcc -g -Wall -Werror -std=c99 -m64 -o csim csim.c cachelab.c -lm
gcc -g -Wall -Werror -std=c99 -m64 -O0 -c trans.c
gcc -g -Wall -Werror -std=c99 -m64 -o test-trans test-trans.c cachelab.c trans.o
gcc -g -Wall -Werror -std=c99 -m64 -O0 -o tracegen tracegen.c trans.o cachelab.c
# Generate a handin tar file each time you compile
tar -cvf hyx-handin.tar csim.c trans.c
csim.c
trans.c
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ls
Makefile  cachelab.c  csim  csim.c  hyx-handin.tar  test-trans  tracegen  traces  trans.o
README.md cachelab.h  csim-ref  driver.py  test-csim    test-trans.c  tracegen.c  trans.c
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ./test-csim
Your simulator      Reference simulator
Points (s,E,b) Hits Misses Evicts Hits Misses Evicts
3 (1,1,1) 9 8 6 9 8 6 traces/yi2.trace
3 (4,2,4) 4 5 2 4 5 2 traces/yi.trace
3 (2,1,4) 2 3 1 2 3 1 traces/dave.trace
3 (2,1,3) 167 71 67 167 71 67 traces/trans.trace
3 (2,2,3) 201 37 29 201 37 29 traces/trans.trace
3 (2,4,3) 212 26 10 212 26 10 traces/trans.trace
3 (5,1,5) 231 7 0 231 7 0 traces/trans.trace
6 (5,1,5) 265189 21775 21743 265189 21775 21743 traces/long.trace
27
TEST_CSIM_RESULTS=27
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$
```

图 10

在进行第二部分实验前,需先安装 valgrind,输入 `sudo apt install valgrind` 后完成安装。

第二部分实验的核心代码详见[链接](#)。

之后便可输入测试命令,逐一测试,  $32*32$ ,  $64*64$ , 及  $61*67$  的矩阵转置,测试结果如下,基本符合实验预期:



```

hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout
hyx@DESKTOP-LP1A2G2: $ cd Github/CSAPP_labs/cachelab-handout
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ls
Makefile  cachelab.c  csim  csim.c  hyx-handin.tar  test-trans  trace.tmp  tracegen.c  trans.c
README.md  cachelab.h  csim-ref  driver.py  test-csim  test-trans.c  tracegen  trans.o
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ./test-trans -M 32 -N32

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:1766, misses:287, evictions:255

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:870, misses:1183, evictions:1151

Summary for official submission (func 0): correctness=1 misses=287

TEST_TRANS_RESULTS=1:287
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ./test-trans -M 64 -N 64

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:9026, misses:1219, evictions:1187

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:3474, misses:4723, evictions:4691

Summary for official submission (func 0): correctness=1 misses=1219

TEST_TRANS_RESULTS=1:1219
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$ ./test-trans -M 61 -N 67

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:6187, misses:1992, evictions:1960

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:3756, misses:4423, evictions:4391

Summary for official submission (func 0): correctness=1 misses=1992

TEST_TRANS_RESULTS=1:1992
hyx@DESKTOP-LP1A2G2: ~/Github/CSAPP_labs/cachelab-handout$

```

图 11

## 遇到的问题 and 解决方案说明

1. Data Lab 编译时出现报错，导致未能使用实验自带的测试程序，对实验结果进行测试。  
解决方案：未找到合适的解决办法，报错信息来自测试程序。
2. Cache Lab 运行第二个实验部分时，缺少 `valgrind`。  
解决方案：在 Ubuntu 中输入 `sudo apt install valgrind`，完善环境。

## 体会与总结

Data Lab 在中实现的函数，有的与 Leetcode 上的题类似，都是考察对数在计算机中的表示的理解。通过这个实验复习了课上的知识，但因为对操作的限制，实验本身的难度还是不小。

Bomb Lab 是这些实验中最有意思的实验。虽然，汇编代码比较难阅读，但拆弹的过程趣味性十足。而且，通过这种寓教于乐的方式，真正深入了解了汇编语言。相较于组成原理课程中，简单的冒泡排序汇编程序设计，这种形式的实验(引导式的学习)有助于掌握汇编，而非简单的一带而过。此外，这次实验还学会了 `gdb` 的基础操作。因此，我认为这一实验是我目前做过的实验中收获最多的。

Cache Lab 部分与组成原理所学内容比较接近，因此实验的过程并不算太复杂。当然，在矩阵转置优化过程中，也是需要有很多思考的内容。

总体而言，经过一学期这门课的学习，我学到了很多在计算机学院计划课程中所未能学到/未能学精的知识。当然，我也希望学校能更多地借鉴国外的类似 CSAPP 这样的课程，比如 MIT 的 [The Missing Semester of Your CS Education](#)。让我们能学到一些在日后科研/工作中应具备的基本技能，基本素养。

本次实验的所有文件都保存在了 [Github 仓库](#) 中。

实验过程中，我也遇到了不少问题，在 `csdn`，`stack overflow` 等社区上也学习到了不少

实验以外的内容。整体而言，通过这门课所学确实超过了我的预期，而且也不至于像某些专业课那样负担大。可以说，这是性价比极高的一门课。