
Introduction to Datalab

— Tang Tang. Feb 22 —

Recap: operations in C

- And
 - Use 1 to extract, 0 to unset one bit
 - Extract the ith bit: $x \& (1 \ll (i - 1))$
 - Unset the ith bit: $x \& \sim(1 \ll (i - 1))$
- Or
 - Use 1 to set, 0 to extract one bit
 - Set the ith bit: $x | (1 \ll (i - 1))$
- Xor
 - 0 if equal, otherwise 1
 - Flip the ith bit: $x \wedge (1 \ll (i - 1))$
- Not
 - Flip all bits

$$\begin{array}{r} 01101001 \\ \& 01010101 \\ \hline 01000001 \end{array}$$

$$\begin{array}{r} 01101001 \\ | 01010101 \\ \hline 01111101 \end{array}$$

$$\begin{array}{r} 01101001 \\ \wedge 01010101 \\ \hline 00111100 \end{array}$$

$$\begin{array}{r} \sim 01010101 \\ \hline 10101010 \end{array}$$

And

$A \& B = 1$ when both $A=1$ and $B=1$

$\&$	0	1
0	0	0
1	0	1

Or

$A | B = 1$ when either $A=1$ or $B=1$ or both

$ $	0	1
0	0	1
1	1	1

Exclusive-Or (Xor)

$A \wedge B = 1$ when $A=1$ or $B=1$, but not both

\wedge	0	1
0	0	1
1	1	0

Not

$\sim A = 1$ when $A=0$

\sim	0	1
	1	0

Recap: operations in C

- Logic Not
 - View 0 as **false**, otherwise **true**
 - **Always return 0 or 1**
 - **Whether an integer is zero: !x**
 - Whether an integer is not zero: **!!x**
- Add
 - Some magic between x, x + 1 and ~
 - Lowbit(x): **x & (~x+1)**
- << and >>
 - Replace * with <<
 - Multiple x by 8: **x << 3**
 - Combined with |
 - Cat: **(0xFF << 8) | 0xFF = 0xFFFF**

x	z10..0
~x	~z01..1
x + 1	z10..1
~x + 1	~z10..0

Argument x	01100010
<< 3	00010000
Log. >> 2	00011000
Arith. >> 2	00011000

Argument x	10100010
<< 3	00010000
Log. >> 2	00101000
Arith. >> 2	11101000

Datalab: Reminders

- Type
 - **int** and **long** are available
 - cast between int and long is ok, in either direction: **(long)x, (1L << 63)**
- Constant
 - Constant between **0x0 ~ 0xFF (0 ~ 255)** is valid
 - It is ok to concatenate such as **0xFFFF = (0xFF << 8 | 0xFF)**, uses 2 ops (<< and |)
- Operators
 - One operand
 - ! and ~
 - Two operands
 - &, |, ^, +, << and >>
 - **Other operators is invalid**
- Unlimited
 - (), =, and local variables **are always valid and not counted**
- **Important**
 - **Declare all local variables at the beginning of the function**



Some puzzles may have more strict limitations

Datalab: Getting Started

- Method 1:
 - Download [datalab-sp25.tar](#) from [course website](#)
 - **cd /path/to/your/download**
 - Upload tar to remote server (if using ics server)
 - **scp datalab-sp25.tar username@x86.ics.xjtu-ants.net:~/datalab-sp25.tar**
 - Decompress
 - **tar -xvf datalab-sp25.tar**
- Method 2 (recommended):
 - Clone from [Github repo](#)
 - **git clone [-o remote_name] https://github.com/xjtu-ics/datalab-sp25.git (HTTPS)**
 - **git clone [-o remote_name] git@github.com:xjtu-ics/datalab-sp25.git (SSH)**
- Start
 - **cd /path/to/your/datalab-sp25**
 - **make**

Datalab: Check your code

- **btest**
 - runs your solutions on random values
- **dlc**
 - a modified C parser
- **driver.pl**
 - Runs **./dlc -z** to identify coding rules
 - Runs **./btest -g** to check correctness
 - Runs **./dlc -Z** to check operation counts
 - Print your grade
- **Grade**
 - Correctness: pass **./btest** check and **./dlc -z** check
 - Performance: pass **./dlc -Z** check (use **./dlc -e** to count your operations)
 - We will use **driver.pl** to test your solution

Datalab: Submission

- Upload **bits.c** in a **zip** file to [here](#) for submission
 - **make submit**

Datalab: Some tools

- ishow

- Usage: `./ishow val1 val2 ...`
- Values may be given in **hex** or **decimal**
- Examples:

```
ttang in ~/cs/xjtu-ics/ics-labs/datalab/datalab-sp25 on main ● ● λ ./ishow 0x88 88
Hex = 0x00000088,      Signed = 136,   Unsigned = 136
Hex = 0x00000058,      Signed = 88,    Unsigned = 88
```

- printf()

- Directly used in bits.c (may be some warnings, but it is ok in this lab)
- **Do not include** `<stdio.h>` in bits.c, or you will not pass the dlc checker

Datalab: Example

- **Objective:** swap odd and even bits in an integer, where bits are numbered from 0 (least significant) to 31 (most significant)
- **Legal ops:** `&`, `|`, `^`, `~`, `!`, `+`, `<<`, `>>`
- **10 ops max**

```
1  /*
2   * swapOddEvenBits - Swap odd and even bits
3   *   where bits are numbered from 0 (least significant) to 31 (most significant)
4   *   Examples: swapOddEvenBits(0b101110101) -> 0b01111010
5   *   Legal ops: & | ^ + << >> ~ !
6   *   Max ops: 10
7   *   Rating: 3
8   */
9  int swapOddEvenBits(int x) {
10     return 2;
11 }
```

Example:

- `0b101110101`
 - odd: `0b10100000`
 - even: `0b0010101`
- `0b01111010`

Datalab: Example

correct?	Valid op?	Ops <= 10?

- First attempt:

```
1  /*
2   * swapOddEvenBits - Swap odd and even bits
3   *   where bits are numbered from 0 (least significant) to 31 (most significant)
4   *   Examples: swapOddEvenBits(0b10110101) -> 0b01111010
5   *   Legal ops: & | ^ + << >> ~ !
6   *   Max ops: 10
7   *   Rating: 3
8   */
9  int swapOddEvenBits(int x) {
10     int mask1 = 0x55555555;
11     int mask2 = 0xAAAAAAAA;
12     return ((x & mask1) << 1) | ((x & mask2) >> 1);
13 }
```

Datalab: Example

correct?	Valid op?	Ops <= 10?
✗		

- First attempt:
 - WTF ?

```
ERROR: Test swapOddEvenBits(-2147483648[0x80000000]) failed...
...Gives -1073741824[0xc0000000]. Should be 1073741824[0x40000000]
```

```
1  /*
2   * swapOddEvenBits - Swap odd and even bits
3   *   where bits are numbered from 0 (least significant) to 31 (most significant)
4   *   Examples: swapOddEvenBits(0b10110101) -> 0b01111010
5   *   Legal ops: & | ^ + << >> ~ !
6   *   Max ops: 10
7   *   Rating: 3
8   */
9  int swapOddEvenBits(int x) {
10     int mask1 = 0x55555555;
11     int mask2 = 0xAAAAAAAA;
12     return ((x & mask1) << 1) | ((x & mask2) >> 1);
13 }
```

Example:

- $x = 0b\ 1000\dots000$
- $x \& \text{mask1}: 0b_0\dots_0_$
- $x \& \text{mask2}: 0b1_0\dots_0_$
- **What if $(x \& \text{mask2}) \gg 1$?**
 - $0b\ \mathbf{1}1_0\dots_0_$
- **Finally**
- $0b\ \mathbf{1}100\dots00\ (0xc0000000)$

Datalab: Example

correct?	Valid op?	Ops <= 10?
		

- Second attempt:
 - Correct but invalid

```
10 10 0 swapOddEvenBits
dlc:bits.c:173:swapOddEvenBits: Illegal constant (0x55555555) (only 0x0 - 0xff allowed)
dlc:bits.c:174:swapOddEvenBits: Illegal constant (0xAAAAAAAA) (only 0x0 - 0xff allowed)
dlc:bits.c:177:swapOddEvenBits: Zapping function body!
```

```
1 /*
2  * swapOddEvenBits - Swap odd and even bits
3  *   where bits are numbered from 0 (least significant) to 31 (most significant)
4  *   Examples: swapOddEvenBits(0b10110101) -> 0b01111010
5  *   Legal ops: & | ^ + << >> ~ !
6  *   Max ops: 10
7  *   Rating: 3
8  */
9 int swapOddEvenBits(int x) {
10     int mask1 = 0x55555555;
11     int mask2 = 0xAAAAAAAA;
12     int mask3 = ~(1 << 31);
13     return ((x & mask1) << 1) | (((x & mask2) >> 1) & mask3);
14 }
```



Tips:

- Use 0x7FFFFFFF and &
- Unset sign bit
- Keep others unchanged

How to get mask?

- **Use << and |**
- 0x55555555
 - (0x55 << 8) | 0x55
 - (0x5555 << 16) | 0x5555
- **0xAAAAAAAA = ~0x55555555**

Datalab: Example

correct?	Valid op?	Ops <= 10?
		




- Third attempt:
 - Ops < 10?

```
1  /*
2   * swapOddEvenBits - Swap odd and even bits
3   *   where bits are numbered from 0 (least significant) to 31 (most significant)
4   *   Examples: swapOddEvenBits(0b10110101) -> 0b01111010
5   *   Legal ops: & | ^ + << >> ~ !
6   *   Max ops: 10
7   *   Rating: 3
8   */
9  int swapOddEvenBits(int x) {
10     int mask = 0x55;
11     int mask3 = ~(1 << 31);
12     mask |= mask << 8;
13     mask |= mask << 16;
14     return ((x & mask) << 1) | (((x & (~mask)) >> 1) & mask3);
15 }
```

Do we really need mask3?

- Why mask3
 - Unset the sign bit
 - Keep others unchanged
- Why not mask3
 - Use >> first
 - Mask can help us to unset the sign bit

Datalab: Example

correct?	Valid op?	Ops <= 10?
		

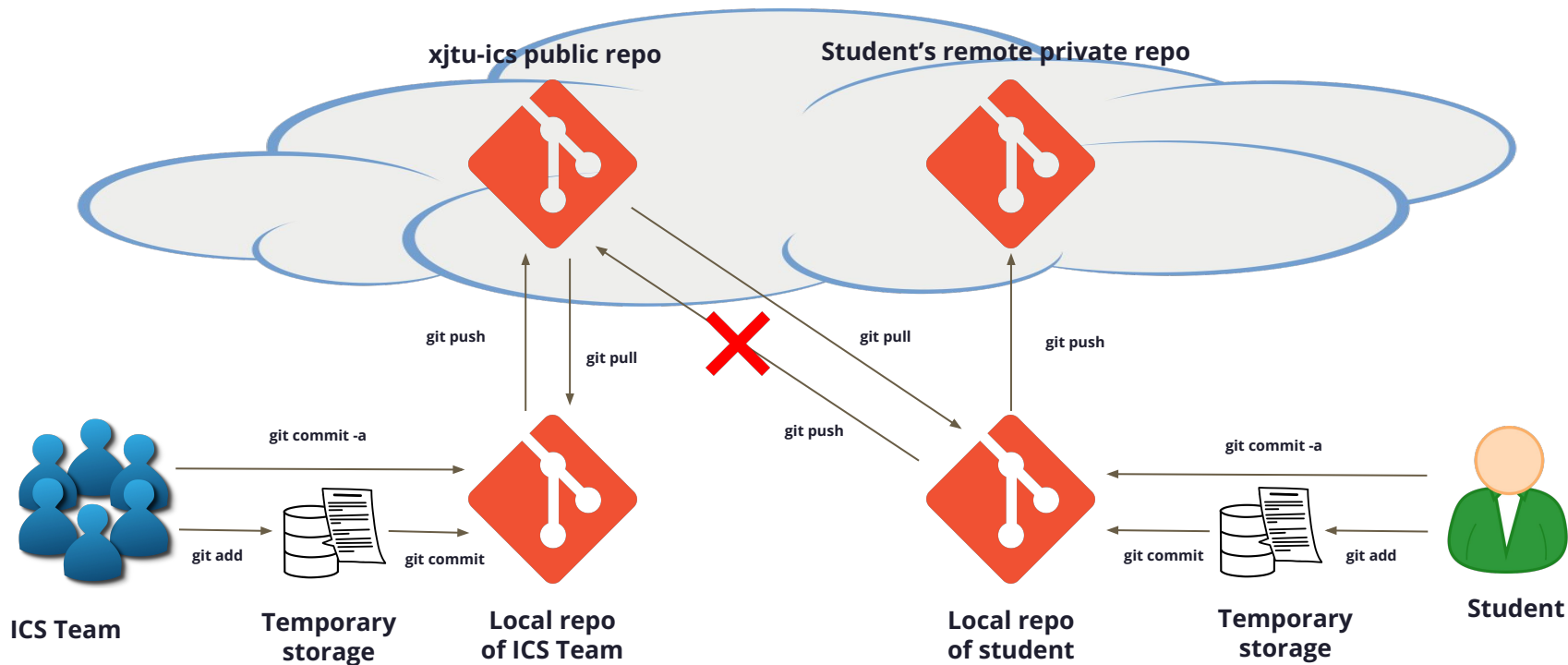
- Final attempt:

10 10 0 2 9 swapOddEvenBits

```
1  /*
2   * swapOddEvenBits - Swap odd and even bits
3   *   where bits are numbered from 0 (least significant) to 31 (most significant)
4   *   Examples: swapOddEvenBits(0b10110101) -> 0b01111010
5   *   Legal ops: & | ^ + << >> ~ !
6   *   Max ops: 10
7   *   Rating: 3
8   */
9  int swapOddEvenBits(int x) {
10     int mask = 0x55;
11     mask |= mask << 8;
12     mask |= mask << 16;
13     return ((x & mask) << 1) | ((x >> 1) & mask);
14 }
```

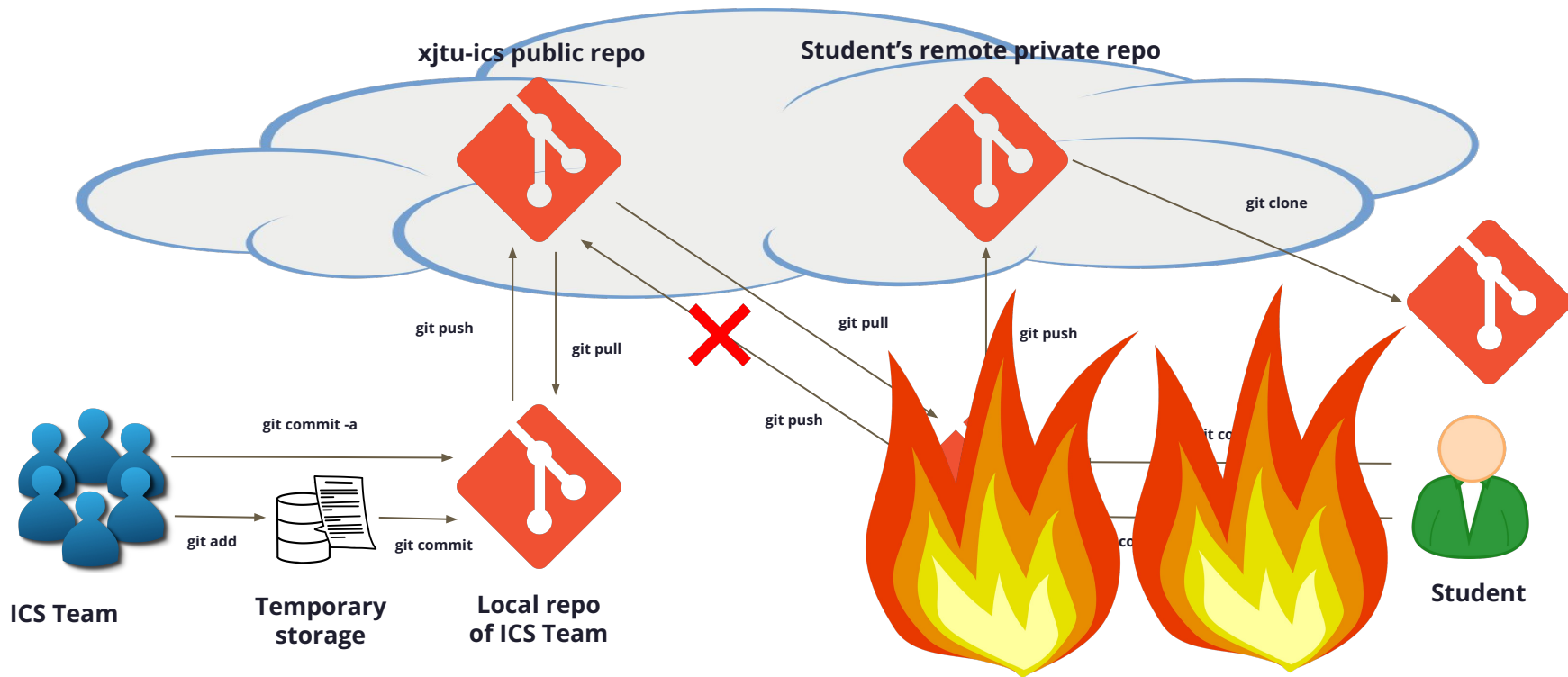
Using Github

- Use your own **private** repo to backup



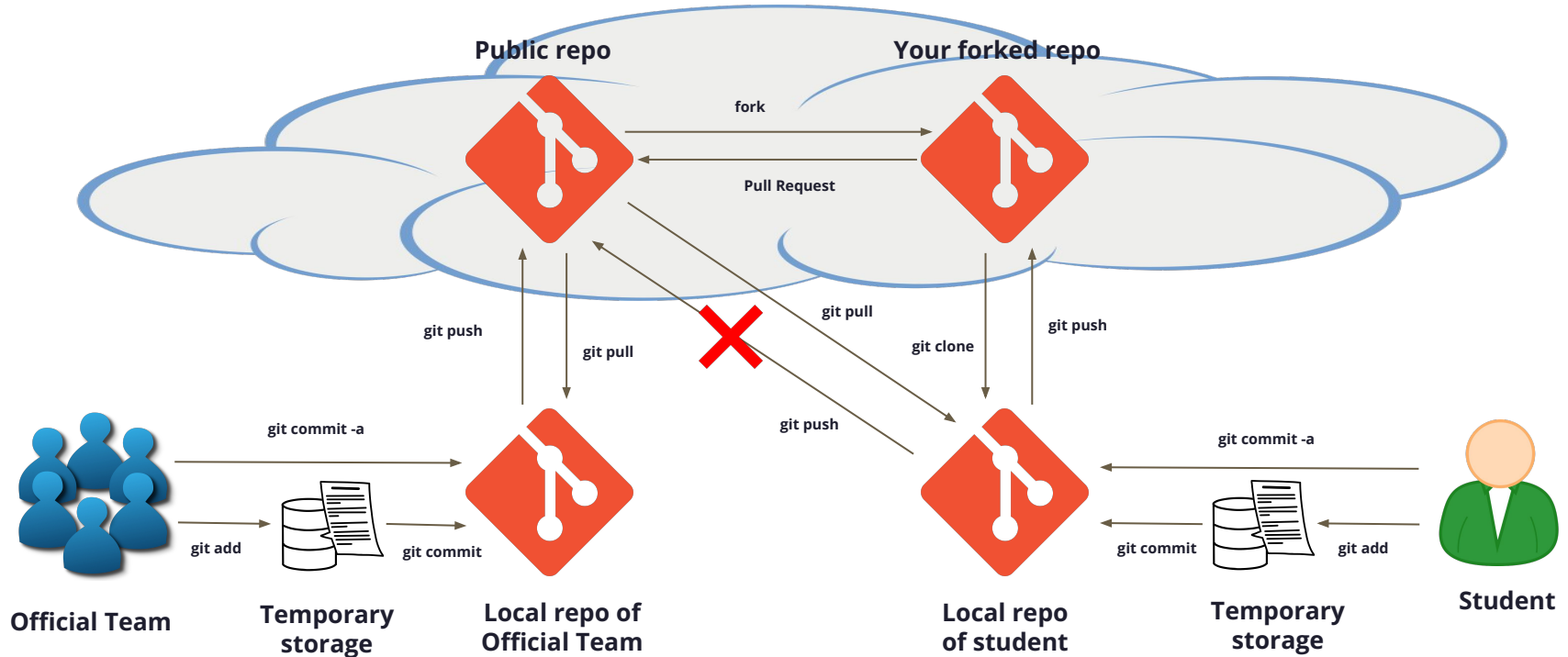
Using Github

- When crashed...



Using Github

- Contributing to open source project



Using Github

- [xjtu-ics/textbook](https://github.com/xjtu-ics/textbook) is waiting for you

textbook Public template
generated from [lzzsG/mdBook-pages-template](#)

Edit Pins Watch 0 Fork 4 Star 0 Use this template

main 1 Branch 0 Tags

Go to file Add file Code

Orion-zhen fix typos 8fd2e34 · 4 days ago 86 Commits

File/Folder	Commit Message	Time
.github/workflows	fix mdbook-admonish version for CI	last month
recommendation	add rec books	last month
src	fix typos	4 days ago
.gitignore	Initial commit	last month
CONTRIBUTING.md	add contributing format	last month
LICENSE	GPL v3 license	last month
NOTICE	::page_facing_up: add apache-2.0 notice	last month
README.md	add rec books	last month
book.toml	remove pdf-gen for env-err	last month
mdbook-admonish.css	add pdf-gen and ics homepage	last month
ref-template.md	framework is done	last month

About

mdBook for ICS course in Xi'an Jiaotong University

xjtu-ics.github.io/textbook/

javascript markdown rust mdbook xjtu google-style-guide

Readme GPL-3.0 license Activity Custom properties 0 stars 0 watching 4 forks Report repository

Releases

No releases published
[Create a new release](#)