Level 0x06

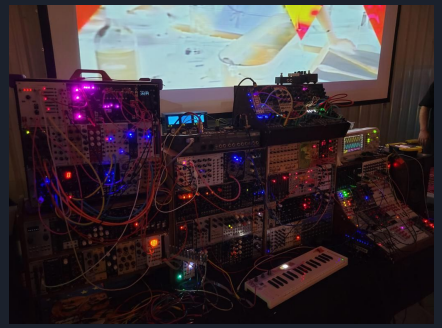
More C / The Stack

Topics

- Events
- Hacker History
- C Stuff
  - Branching
  - Looping
  - The Stack

Upcoming Events

- Maker Faire Orlando - “Greatest Show & Tell on Earth”
  - Saturday Nov 4th - Sunday Nov 5th, 2023
  - https://www.makerfaireorlando.com/
  - Orange County Fairgrounds
  - $25 Adults, $20 Students, $5 off if pre-purchased



rms
Richard Stallman

- Founder of GNU project (1983)
  - He got angry at a printer...
- GNU General Public License (GPL)
- GNU Compiler Collection (gcc)
- GNU Debugger (gdb)
- GNU Make
- GNU Emacs
- Insufferable / Low Charisma
  - GNU/Linux
  - Refuses to use anything non-free
  - Epstein controversy
  - Controversial political views

GPL

- Software License for Open Source Software
  - the freedom to use the software for any purpose,
  - the freedom to change the software to suit your needs,
  - the freedom to share the software with your friends and neighbors, and
  - the freedom to share the changes you make.
- If you *distribute* GPL software, you must *offer* source code for it
- *Derivative* works can't be more restricted than GPL
- GPL Virus (Microsoft):
  - Steve Ballmer declared that code released under GPL is useless to the commercial sector (since it can only be used if the resulting surrounding code becomes GPL)
  - "A cancer that attaches itself in an intellectual property sense to everything it touches".

If Conditionals

```
if (condition)
{
     // code block
}
```

```
if (condition)
{
     // code block
}
else
{
     // other code block
}
```

Switch Statements

```
switch (variable or expression)
{
case 1:
     // do stuff, fallthrough to next case
case 2:
     // so stuff, no fallthrough
     break;
case 3:
     {
          // code block variables
          // this is gross though
          // just call a function
     }
     break;
default:
     // if no other case was picked, do this
}
```

While loop

```
while(condition)
{
     // do this block of instructions over and over

     if (diff_condition)
     {
          // restart loop right now!
          continue;
     }

     if (some_other_condition)
     {
          // let's get out of this while loop,
          // regardless of loop condition
          break;
     }
}
```

Do While Loop

```
do
{
     // No matter what, I'm doing this code block
     // at least 1 time!

     // break and continue work in this too

} while(condition);  // don't forget this semicolon…
```

For loop

```
// My loop had a simple counting variable
for (int i = 0; i < 10; i++)
{
     printf("I've run this loop %d times\n", i);
}

// Initialization statement only runs 1 time

// Loop condition runs every loop, even before first loop

// Increment statement runs after every loop
```

Functions

```
int my_div_func(int arg1, int arg2)
{
      if (arg2 == 0)
      {
            printf("Don't divide by zero!\n");
            return 0;
      }

      return arg1 / arg2;
}

int main(int argc, char** argv)
{
      int topNum = atoi(argv[1]);
      int bottomNum = atoi(argv[2]);
      printf("Div result = %d\n", my_div_func(topNum, bottomNum) );
      return 0;
}

// Function must be declared before you can call them
```

Void function

```
void hexprint(int val)
{
     If (val < 0)
     {
          printf("-0x%08x", val * -1);
          return;
     }

     printf("0x%08", val);

     // Doesn't return anything, don't even need a return statement
}
```

Arrays

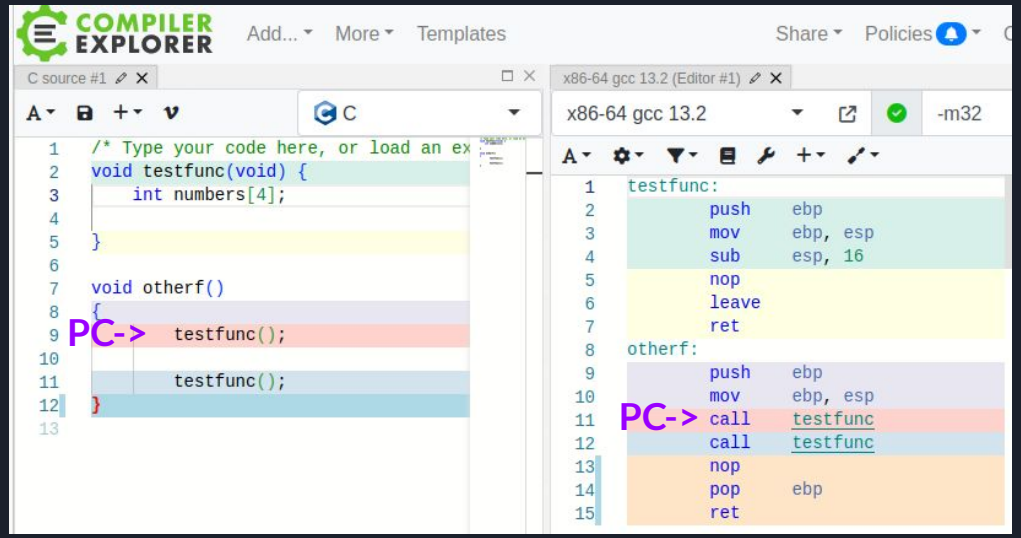
```
int listOfNums[4];

int differentList[] = { 1, 2, 3, 4};

// This won't work, size of array must be known
// at time of compilation
int badList[variable_from_user];

for(int i = 0; i < 4; i++)
{
     // Uninitialized arrays are filled with garbage data
     print("listOfNums index %d = %d\n", i, listOfNums[i]);
}
```

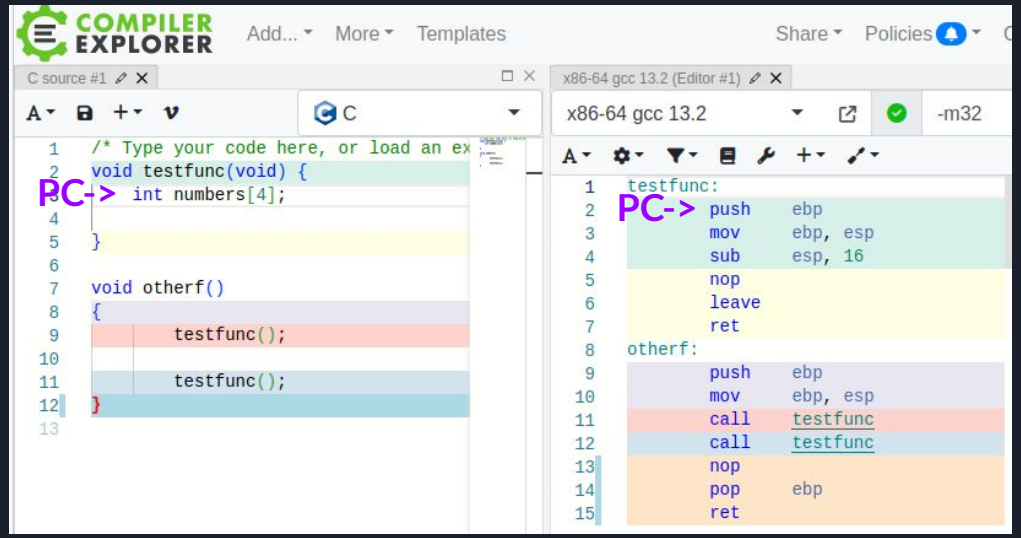
The Stack



Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	
0xf000 0040	
0xf000 0044	
0xf000 0048	
0xf000 004c	
0xf000 0050	

SP->
BP->

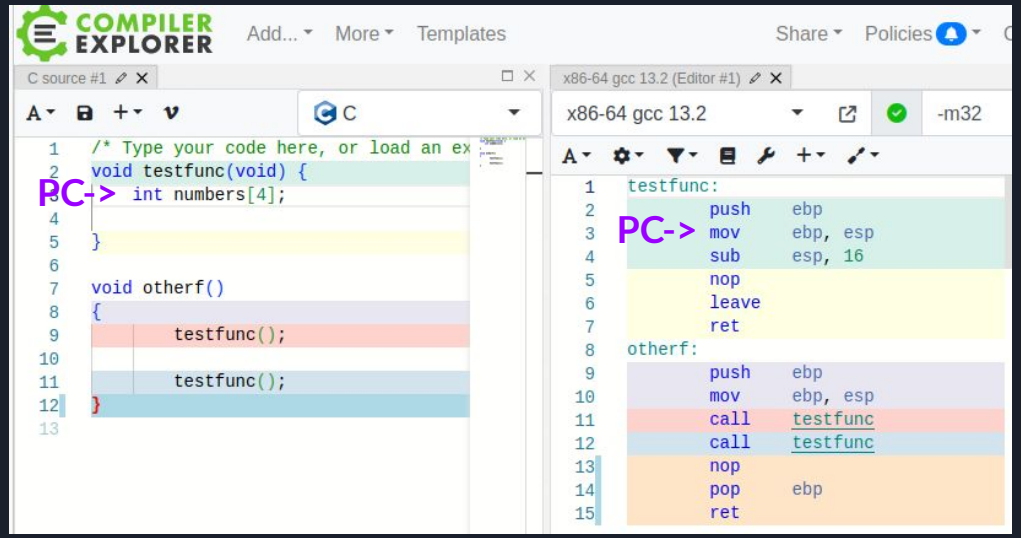
The Stack



	Addresses	Memory Vals
	0xf000 0030	
	0xf000 0034	
	0xf000 0038	
	0xf000 003c	
	0xf000 0040	
	0xf000 0044	
	0xf000 0048	
SP->	0xf000 004c	
BP->	0xf000 0050	12

Return address get pushed onto stack by call

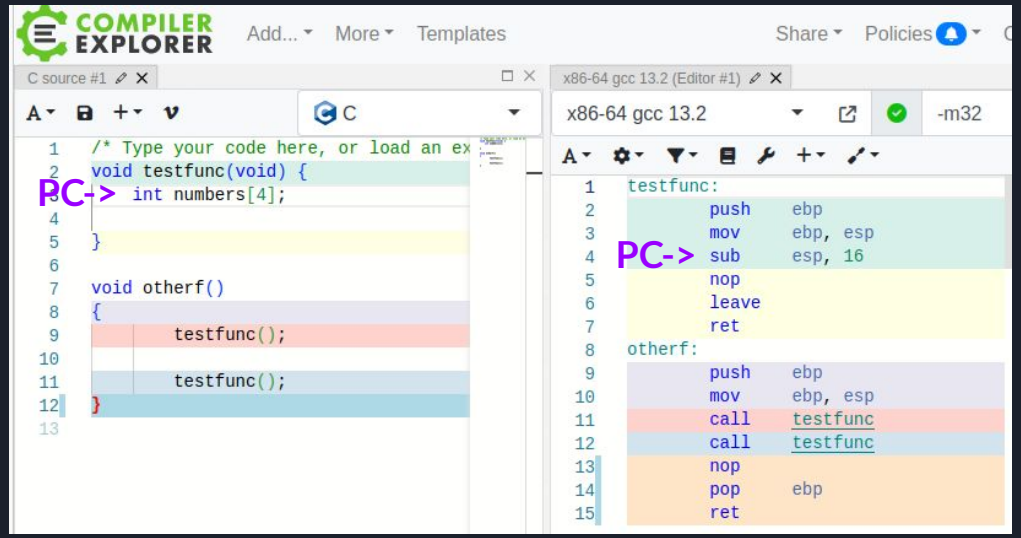
The Stack



	Addresses	Memory Vals
	0xf000 0030	
	0xf000 0034	
	0xf000 0038	
	0xf000 003c	
	0xf000 0040	
	0xf000 0044	
SP->	0xf000 0048	
	0xf000 004c	0xf000 0050
BP->	0xf000 0050	12

Pushed BP location onto stack

The Stack

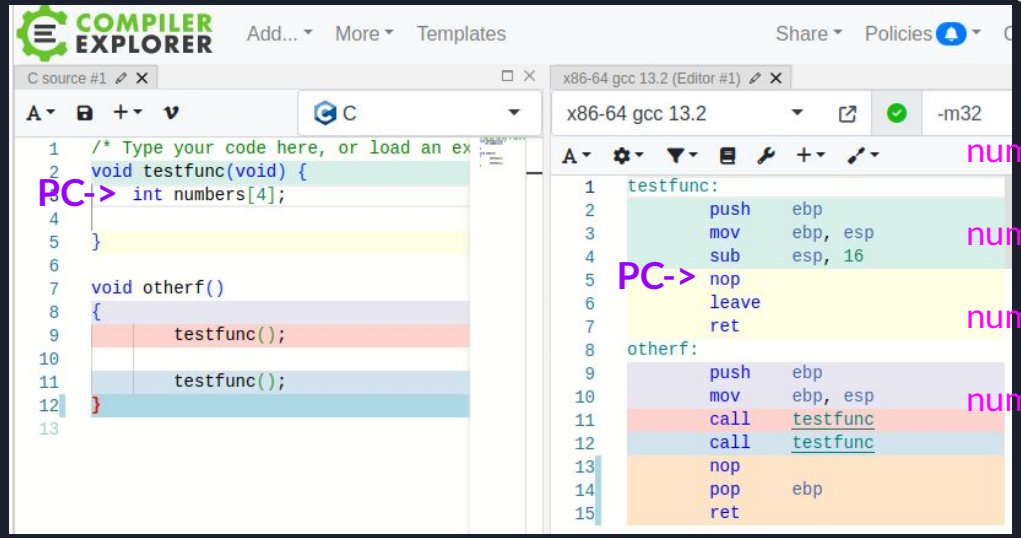


Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	
0xf000 0040	
0xf000 0044	
SP-> BP-> 0xf000 0048	
0xf000 004c	0xf000 0050
0xf000 0050	12

Moved BP to same spot as SP

The Stack

SP moved up 0x10 or 16 bytes



SP->

numbers[0]

numbers[1]

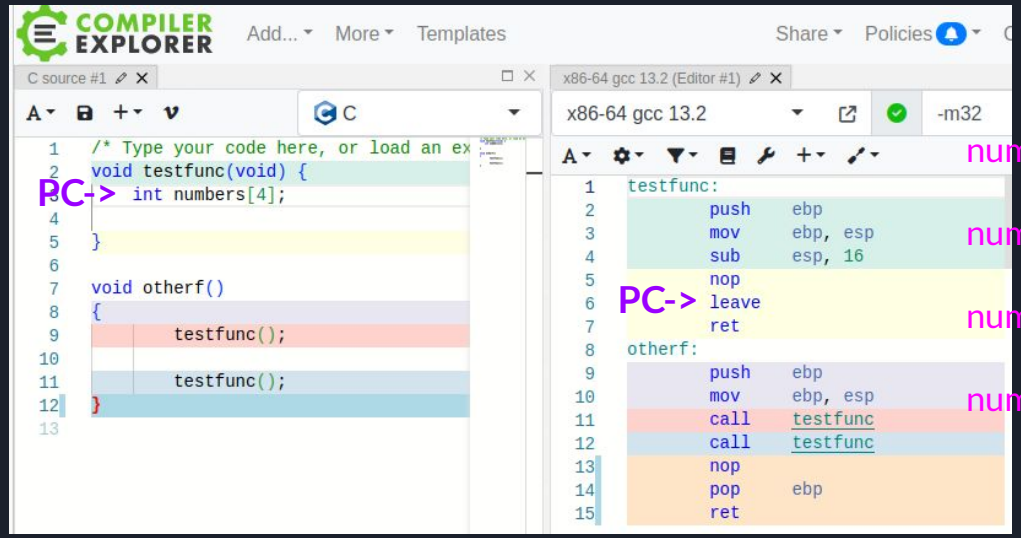
numbers[2]

numbers[3] BP->

Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

The Stack

NOP = no operations, does nothing



SP->

numbers[0]

numbers[1]

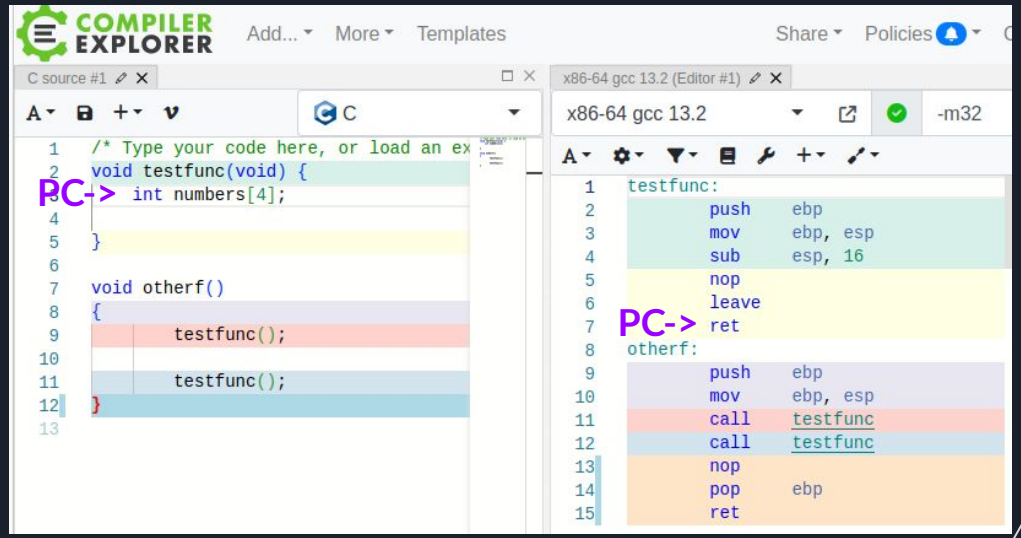
numbers[2]

numbers[3] BP->

Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

The Stack

leave is equiv to mov esp, ebp
pop ebp



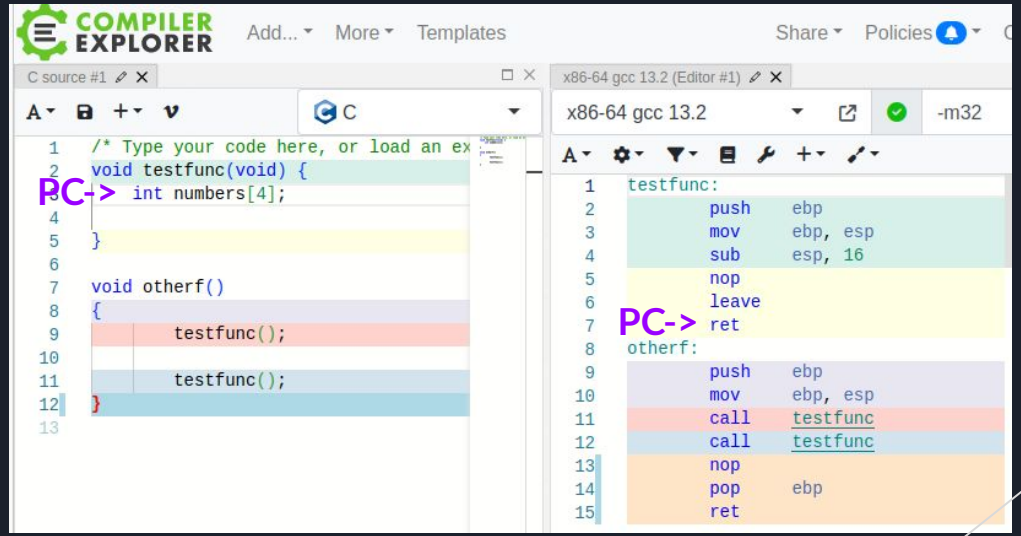
SP->
BP->

Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

We moved SP to where BP is

The Stack

leave is equiv to mov esp, ebp
pop ebp

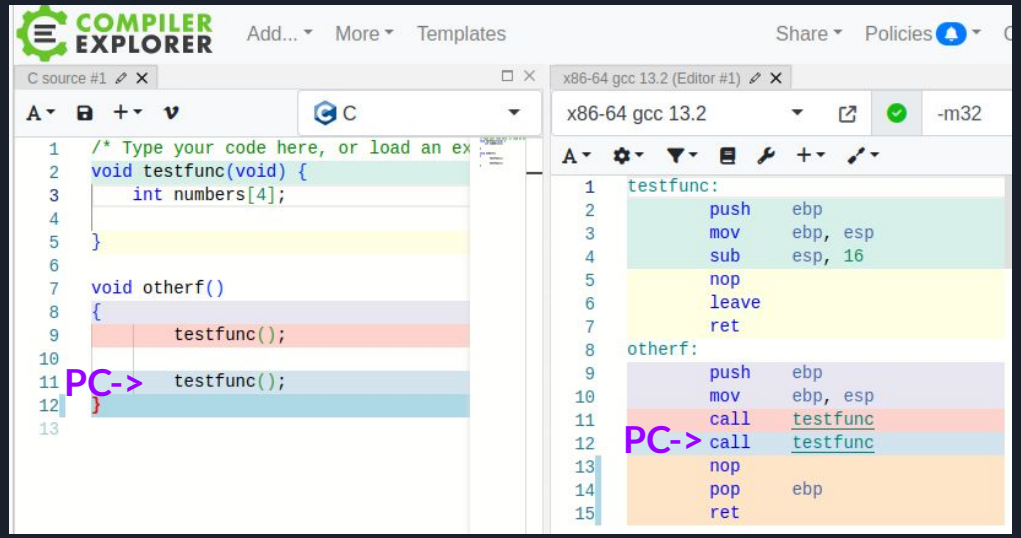


Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
SP-> 0xf000 004c	0xf000 0050
BP-> 0xf000 0050	12

SP moves down cause it was a pop.
BP moves to the value popped

The Stack

ret is equiv to pop PC



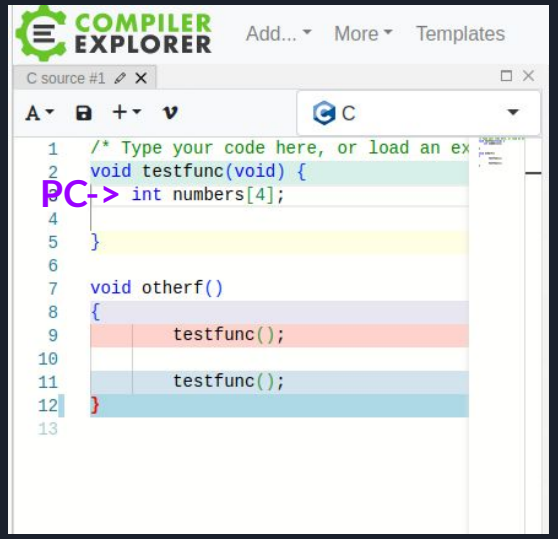
Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

SP->
BP->

We are now back in original function, about to call next instruction. SP and BP haven't moved.

Stack Variable Problems

- What data is stored right now in numbers[]?
- What happens if I wrote to numbers[5]?



	Addresses	Memory Vals
	0xf000 0030	
	0xf000 0034	
SP->	0xf000 0038	
numbers[0]	0xf000 003c	???
numbers[1]	0xf000 0040	???
numbers[2]	0xf000 0044	???
numbers[3]	0xf000 0048	???
	0xf000 004c	0xf000 0050
	0xf000 0050	12

Links

- Many graphics from Wikipedia / wikimedia.org (Creative Commons License)
  - https://en.wikipedia.org/wiki/Richard_Stallman
- https://www.gnu.org/licenses/quick-guide-gplv3.html
-
-