GDB

Compile option

```
-ggdb or -g
```

optimized out

On high optimization levels, the compiler can eliminate intermediate values, as you have seen here. There are a number of options:

- You can reduce the optimization level to make it easier for the debugger to keep track of things. `-O0` is certain to work (but will be quite a lot slower), `-O1` might work okay as well.
- You can add some explicit print statements to log the output value.
- You can also usually force the compiler to retain this specific value by making it volatile (but remember to un-make it volatile when you're done!). Note, however, that since control flow is also subject to alteration in optimized code, even if you can see the value of the variable, it may not be entirely clear what point in the code you're at when you're looking at the variable in question.

Continue, stepping over and in – gdb commands

There are three kind of gdb operations you can choose when the program stops at a break point. They are continuing until the next break point, stepping in, or stepping over the next program lines.

- c or continue: Debugger will continue executing until the next break point.
- n or next: Debugger will execute the next line as single instruction.
- s or step: Same as next, but does not treats function as a single instruction, instead goes into the function and executes it line by line.

By continuing or stepping through you could have found that the issue is because we have not used the <= in the 'for loop' condition checking. So changing that from < to <= will solve the issue.

gdb command shortcuts

Use following shortcuts for most of the frequent gdb operations.

- run [args] - start running the program using the run command in the gdb debugger.
- x [address] - can be thought of as being short for "examine"
- l – list
- p – print
- c – continue
- s – step
- ENTER: pressing enter key would execute the previously executed command again.

Miscellaneous gdb commands

- **l command:** Use gdb command l or list to print the source code in the debug mode. Use l line-number to view a specific line number (or) l function to view a specific function.
- **bt: backtrack** – Print backtrace of all stack frames, or innermost COUNT frames.
- **help** – View help for a particular gdb topic — help TOPICNAME.
- **quit** – Exit from the gdb debugger.

add-symbol-file

```
add-symbol-file ~/hzhen/pcie/misc/hkr_endpoint_test.ko
```

https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch04.html