# Contributing to McSema

Creating an executable lifter like McSema is no easy task. Even after the initial design is complete, there is still a huge linear effort to implement instruction semantics for every instruction. "Semantics" are a C++ description of what the microcode is doing for a given native-code instruction: as each instruction is fed to the lifter, it translates it to LLVM bitcode using the appropriate semantic function.

The Intel x86-64 architecture alone has around 1,000 instruction mnemonics, most of which have multiple operand types and operand widths. We try to use as much [metaprogramming](https://en.wikipedia.org/wiki/Template\_(C%2B%2B)) as possible to reduce the amount of work required to support new instructions, but there is still a lot of work to do. Contributions are greatly appreciated!

## Get in Touch

We do our best to respond to McSema users and potential contributors in our Slack instance, [Empire Hacking](https://empirehacking.slack.com/). Stop into `#binary-lifting` and say hello.

## Taking on a Task

We have many issues already defined. Currently, the majority of them have to do with instruction support. Adding support for a new instruction is a great place to begin if you are a new contributor, to get familiar with the McSema codebase.

Before you start, it would be best if you communicate that you are working on an issue, and if there is not already an issue for what you want to work on, to file an issue first. We use GitHub Issues to track the ongoing progress on the project, deconflict what everyone is doing, and record our accomplishments.

\*\*Note:\*\* If you find a \*\*Closed\*\* issue that seems like it is the same thing that you're experiencing, open a new issue and include a link to the original issue in the body of your new one.

To get started, use git to clone the `remill` repo, and start a new branch that you name according to the issue you are working on. If you are working on "Issue #1234: Missing FPU flag on ARM" then name your branch `issue\_1234\_missing\_fpu\_flag\_on\_arm`.

## Adding a New Instruction

To add support for lifting a new instruction, you will be extending the `remill` library. That is where the instruction semantics are implemented. We have created a whole guide to adding an instruction, which [you can find here](https://github.com/lifting-bits/remill/blob/master/docs/ADD\_AN\_INSTRUCTION.md).

## Creating Your GitHub Pull Request

If you submit a pull request we will do our best to review it and suggest changes in a timely manner. It helps if you constrain your pull request to just one issue fix or one enhancement. Pull requests that change tens of files and make thousands of lines of diff are much harder to approve and merge.

We have GitHub set up to run Travis continuous integration tests. Watch your pull request to see if all of the tests are passing. We will request that all tests pass before we review or merge your changes.

### Coding Style

Before you create your pull request, we ask that you conform your code to our preferred style, by running `clang-format` on your staged source files before you `git commit`.

### Git Commit Messages

\* Use the present tense ("Add feature" not "Added feature")

\* Use the imperative mood ("Move cursor to..." not "Moves cursor to...")

\* Limit the first line to 72 characters or less

\* Reference issues and pull requests liberally after the first line

### Documentation Styleguide

\* Use [Markdown](https://daringfireball.net/projects/markdown).

\* Consider using spellcheck and a Markdown linter, because you'd be surprised what you miss.

\* In documentation, the preferred spelling is \*McSema\*, not \*mcsema\* or \*Mcsema\*.

## Advice and Guidance for New Contributors

Please take a look at our McSema documentation! We try to keep it updated with

[debugging tips](https://github.com/lifting-bits/mcsema/blob/master/docs/DebuggingTips.md), [common errors you might encounter](https://github.com/lifting-bits/mcsema/blob/master/docs/CommonErrors.md), and [how to get acquainted with the codebase](https://github.com/lifting-bits/mcsema/blob/master/docs/NavigatingTheCode.md).

## Useful External Links

\* [Intel architecture software developer manuals](http://www.intel.com/sdm)

\* [Intel XED](https://software.intel.com/sites/landingpage/xed/ref-manual/html/index.html)

\* [ARM AARCH64 specifications](https://developer.arm.com/products/architecture/a-profile/docs)

\* [ARM AARCH64 specs in HTML (unofficial)](https://meriac.github.io/A64\_v83A\_ISA/)

\* [Intel Intrinsics Guide](https://software.intel.com/sites/landingpage/IntrinsicsGuide/)

\* [Setting up an ARM64 test environment](https://gist.github.com/george-hawkins/16ee37063213f348a17717a7007d2c79)