Nand2tetris

HDL file format:

This file represents the circuit using other HDL files which represent primitive gates, as well as, more complex circuits (e.g. Adders, Mux) that already come built into the **nand2tetris** tool which you may use to build your own custom circuit, with some exceptions.

HDL syntax:

A CHIP is divided into 3 sections: IN, OUT, and PARTS. The IN, and OUT are practically the same, here you declare values that will behave as the input and output of the chip. Each value is separated by a comma ending on a semi-colon. You can also specify the size of a variable, this is what the 16 besides b specifies. **Heads up: You can't use OUT variables as input *within their own circuit*.**

Ex:

```
      CHIP NameOfCircuit {
            IN a, b[16];
            OUT out, otherOut[16];

            PARTS:
            // logic section
}
```

The PARTS section of an HDL file is where we use both custom circuits and built-in gates. When a circuit runs, it searches for other HDL files exclusively within its own directory and the built-ins folder.

Temporary variables can be used and do not need to be declared in the **IN** or **OUT** sections. Instead, they are declared within the **OUT** part of a gate used in the **PARTS** section and automatically inherit the size of the returned value. The example provided illustrates how variables can be utilized. Note that, to be able to use CustomCircuit an hdl file with the same name has to exist on the same dir.

You can also use, false and true as constants

Ex:

```
      CHIP NameOfCircuit {
            // in and out...
            IN a, b[16], c[16] ;
            OUT out, otherOut[16];

            PARTS:
            CustomCircuit(a=a, b=b[0],c=c[0..3] , out=temporaryVar);

}
```

Running a circuit

To run a circuit, you can load the file through the HardwareSimulator GUI ... [or run the test with the java file]

### Adding HardwareSimulator to PATH

To run the HardwareSimulator from any directory, you'll need to add it to your system PATH:

Windows

1. For Command Prompt (temporary):
   cmd
   > set PATH=%PATH%;C:\path\to\nand2tetris\tools

2. For PowerShell (temporary):
   powershell
   > $env:PATH += ";C:\path\to\nand2tetris\tools"

3. For permanent changes:
   cmd
   > EXPORT PATH "%PATH%;C:\path\to\nand2tetris\tools"

   Note: You'll need to restart your command prompt after using setx.

macOS/Linux

1. Open your shell profile file (`~/.bashrc`, `~/.zshrc`, or similar)

2. Add: `export PATH="$PATH:/path/to/nand2tetris/tools"`
3. Run `source ~/.bashrc` (or your profile file) to apply changes

Verify installation
Test by running `HardwareSimulator` from any directory.

Built-in chips

Here's a list of the built-in chips in Nand2tetris with their inputs and outputs:

note: If you have a chip with the same name as the gates below it will be overridden by the one you made.

Basic Logic Gates
- **Nand**: IN a, b; OUT out
- **Not**: IN in; OUT out
- **And**: IN a, b; OUT out
- **Or**: IN a, b; OUT out
- **Xor**: IN a, b; OUT out

Multiplexers and Demultiplexers
- **Mux**: IN a, b, sel; OUT out
- **DMux**: IN in, sel; OUT a, b
- **DMux4Way**: IN in, sel[2]; OUT a, b, c, d
- **DMux8Way**: IN in, sel[3]; OUT a, b, c, d, e, f, g, h

Multi-bit Gates
- **Not16**: IN in[16]; OUT out[16]
- **And16**: IN a[16], b[16]; OUT out[16]
- **Or16**: IN a[16], b[16]; OUT out[16]
- **Mux16**: IN a[16], b[16], sel; OUT out[16]
- **Or8Way**: IN in[8]; OUT out
- **Mux4Way16**: IN a[16], b[16], c[16], d[16], sel[2]; OUT out[16]
- **Mux8Way16**: IN a[16], b[16], c[16], d[16], e[16], f[16], g[16], h[16], sel[3]; OUT out[16]

Arithmetic Chips
- **HalfAdder**: IN a, b; OUT sum, carry
- **FullAdder**: IN a, b, c; OUT sum, carry
- **Add16**: IN a[16], b[16]; OUT out[16]
- **Inc16**: IN in[16]; OUT out[16]
- **ALU**: IN x[16], y[16], zx, nx, zy, ny, f, no; OUT out[16], zr, ng