Section 2 / Half Precision Floats

TL;DR - don't use these in C and C++ without being willing to wade through a great deal of muck. In assembly language, it is more straight forward.

Half Precision Formats in C and C++

Support for half precision (16 bit) floating point values does exist but there is no complete agreement on how different compilers support them. Indeed, there are not one but two competing half precision formats out there. These are the IEEE and GOOGLE types. Further still, many open source developers have created their own implementations with potentially clashing naming conventions.

Finally, as of this writing, there is a performance penalty to using half precision floating point values from C and C++ for ordinary math. See below:

```
__fp16 Foo(__fp16 g, __fp16 f) {
    return g + f;
}
```

compiles to:

```
    fcvt    s1, h1
    fcvt    s0, h0
    fadd    s0, s0, s1
    fcvt    h0, s0
    ret
```

Notice each half precision value is converted to single precision. So, from C and C++ working with half precision values can be inefficient.

On the other hand, if you are willing to use *intrinsics* and one of the SIMD instruction sets offered by ARM, then knock yourself out. Be aware that doing so ties your code to the ARM processor in ways which you might regret later.

Where are Half Precision Values Used

Use of half precision is rare. We've only seen half precision used in graphics code and for video processing.

Half Precision in Assembly Language

more to come