Processor

**Memory:** Do we have to fully design memory? I assume not, so then we can just use:

reg[15:0] my\_memory [0:65535]

In conjunction with $readmemb with output from the assembler for I-mem

It could be useful to have clk, enable, rst\_n, read/write, etc. for the memory, but these can be implemented easily

**Caches:** I reviewed our group’s architecture slides on caches and tried to understand the format.

Memory will have 2^16 = 65,536 entries of 16 bits each, word addressable

Data stored will obviously have 16 bits per entry, because that is how our processor works.

I-cache:

I-cache will be much smaller. 2KB with 16B blocks = 128 entries, direct mapped

128 metadata entries with tag => 3 offset bits, 7 index bits, 6 tag bits

I-cache:

[2:0] offset

[9:3] index

[15:10] tag

D-cache:

D-cache is slightly larger. 2560B with 20B blocks = 128 entries, direct mapped

128 metadata entries with tag => 4 offset bits, 7 index bits, 5 tag bits

D-cache:

[3:0] offset

[10:4] index

[15:11] tag

**Register File:**

For this module we will probably have a module for the whole register file from which you can specify the two source and one destination register in binary, provide input data to write (if writing), then have two tri-state data outputs for both read register values.

The data outputs are tri-state to only read one register from each at a time. We don’t want to sample all 8 outputs for valid data.

Compiler