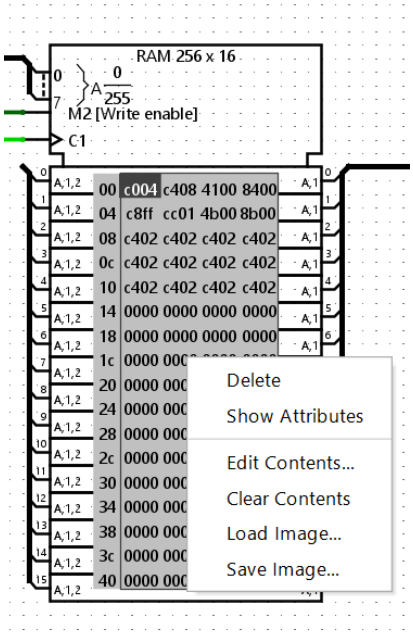
Ryan Lee CS 382-LD

I pledge my honor that I have abided by the Stevens Honor System.

My CPU is called the *Trident*, and I worked on this by myself.
I made an assembler with Node.js to be used on assembly .s files.
The command to use in the terminal to run the assembler is:
node assembler.mjs assemblyFile.s

The “assemblyFile.s” argument is the file name of the assembly file in the same folder. The user would change this to the corresponding assembly file that they made.
I provided an example assembly file called example.s in the zip folder.
Once the assembler program finishes, it writes instructImage.txt into the folder, and this is the instruction memory. To load the instruction memory, right click the RAM, click load image, attach instructImage.txt, and click open. The RAM should be updated according to instructImage.txt like below:



My CPU has 4 general purpose registers and they are referred to as X0, X1, X2, and X3. In the circuit diagram, they are labeled as Register0 to Register3.
The Trident CPU can perform 3 operations: adding 2 registers, subtracting 2 registers, and loading an 8-bit immediate into a register. The output of addition and subtraction are shown at the far right with LED lights, lighting up depending on whether addition or subtraction was performed. Note that the outputs of adding/subtracting are unsigned decimals.
The RAM can store up to 256 instructions and each general purpose register stores up to 255D and as low as 0D.

The binary encoding for the assembly files is:
0000 0000 0000 0000

1. I[15:14] = opcode, the leftmost 2 bits where 00B = nothing, 01B = addition, 10B = subtraction, and 11B = move immediate to register Rm. This is 2 bits to represent the 4 possible operations.
2. I[13:12] = dummy bits, always 00
3. I[11:10] = Rm, the 1st register and this is 2 bits to represent the 4 general purpose registers.
4. I[9:8] = Rn, the 2nd register and this is 2 bits to represent the 4 general purpose registers.
5. I[7:0] = 8-bit immediate to be loaded into the 8-bit registers (rightmost 8 bits).

The syntax for the assembly .s files is:

1. ADD Xm Xn → add the values stored in registers Xm and Xn, outputting the sum in the pin labeled Sum. The binary encoding is I[15:14] = 0100B, I[13:12] = 00B, I[11:10] dependent on m, I[9:8] dependent on n, and I[7:0] = 0000B.
2. SUB Xm Xn → subtract the values stored in registers Xm and Xn, outputting the difference in the pin labeled Difference. The binary encoding is I[15:14] = 1000B, I[13:12] = 00B, I[11:10] dependent on m, I[9:8] dependent on n, and I[7:0] = 0000B.
3. MOV Xm imm8 → load the 8-bit immediate (imm8) into register Xm where imm8 is **read as hexadecimal**. The binary encoding is I[15:14] = 1100B, I[13:12] = 00B, I[11:10] dependent on m, I[9:8] dependent on n, and I[7:0] = imm8 in binary. **Note: imm8** is read as hexadecimal by the assembler like in example.s: when imm8 is ff, that means 0xff.