TCSS 372 Fall 2018   HW5

Due: by midnight Friday Dec. 7
      or with 10% late penalty, Sunday Dec. 9

Using the starting code posted on our Canvas course website, complete the MicMac microcode that emulates (imitates) MIPS16, our slightly modifed version of the MIPS architecture. As outlined in class, MIPS16 is just like MIPS but the 32 general purpose registers are only half as big. Any calculations such as addition supply two 16-bit values to a 16-bit ALU to produce a 16-bit result and the LW and SW instructions move only 2 bytes of data between the processor and memory, not 4.

We have decided to use the MicMac's Mac memory to store, in this order:

- the 32 general purpose registers (2 bytes each)
- the instructions of whatever MIPS program we're running (4 bytes each)
- the data of the MIPS program (4 bytes for each variable)

In class, we will try to implement microcode for the LW instruction and this will become the starter code for the assignment. You must complete the SW, BEQ, ADD, and SLT instructions. You may skip the AND, OR, and SUB instructions.

Two test programs will be available. To run them, you have to paste them into the bottom of a MicMac .sim file where the Mac code would normally be located. When finished, submit your MicMac microcode together with test1 copied into the bottom.

Remember that addresses generated by a MIPS program are generally divisible by 4. For example, the instructions, since they follow 64 bytes of general purpose registers, will be located at MIPS addresses 64, 68, 72 etc. These addresses cannot be used when accessing the MicMac's mac memory because that memory assigns address to each pair of bytes, not to each byte. Therefore the instructions at MIPS address 64, 68, and 72 will be located in the MicMac's mac memory at MicMac addresses 32, 34, and 36. <u>MicMac addresses are half as big as MIPS addresses.</u> This is important because you will be using MicMac addresses whenever you put a value in the MicMac's MAR. The PC should hold normal MIPS addresses such as 64, 68, 72 and in fact, our microcode will initialize the PC to 64. Likewise, our test programs use $at as the base register and that register is therefore initialized with the MIPS address of the start of the data section.

To accomodate the "shrinkage" of our MIPS variables from four bytes to two bytes, when we prepare test programs we manually move data sitting in the bottom half of a MIPS 4-byte word into the top half. That way, LW and SW can use the same addresses (offsets) generated by Mars and will still find and modify values created by the original MARS program even though LW and SW now fetch and return just 2 bytes and not 4 as MIPS would.