Assembly and Memory

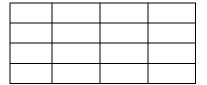
The current state of memory is given to the right.

FF	FF	00	00
00	00	00	80
СВ	10	11	12
00	11	22	33

Assume the memory resets to the above state after every set of operations. Feel free to leave a box blank if it does not change for a particular problem.

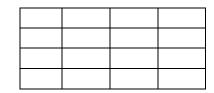
1. What is the state of memory after these operations?

lw \$s0, 0(\$0) addi \$t0, \$s0, 5 sw \$t0, 0(\$0)



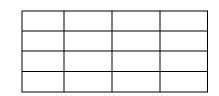
2. What is the state of memory after these operations?

lw \$t1, 12(\$0) lw \$t2, 4(\$0) and \$s0, \$t1, \$t2 sw \$s0, 0(\$0)



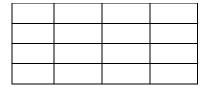
3. What is the state of memory after these operations?

Iw \$t3, 8(\$0) Iw \$t4, 4(\$0) sw \$t4, 0(\$t3) sw \$t3, -4(\$t3)

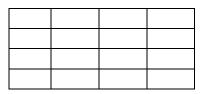


4. Assume we have big-endian memory. What is the state of memory after these operations?

lb \$t0, 11(\$0) lb \$t1, 4(\$0) sb \$t1, 0(\$0)



5. What is the result of the same set of operations using little-endian memory?



6.	Add 0xCB101112 and 0x00112233. Store the result in \$s0.
7.	Multiply the data at Word 0 by 4 and store it back in word 0.
8.	Store the value 0x00FF10CB in Word 1. (This one is a little more open ended. There are several different ways to go about it.)
9.	Bonus question: Can you complete question 8 using just one sw operation?

Given the memory layout from the previous page, write assembly code to accomplish the following

tasks.