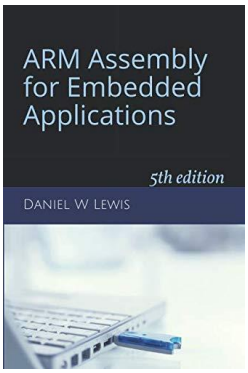


Programming Lab 7A

Reversing Bits & Bytes

Topics: Bit and byte manipulation, shift instructions, bitfields.

Click to download Lab7A-Main.c

Prerequisite Reading: Chapters 1-7

Revised: June 22, 2021

***Background*[1]:** Reversing the order of bits is needed when computing a Cyclic Redundancy Check (CRC) to detect errors in digital communications and data storage. Reversing the order of bytes is needed when converting between the Little Endian byte order of the ARM processor and the Big Endian byte order used in Internet packets. These operations are used so pervasively that they must be implemented as efficiently as possible; that's why ARM created the REV and RBIT instructions that perform these operations in a single clock cycle.

Assignment: Your task is to implement these same operations without using the REV and RBIT instructions. Your solutions should execute as fast as possible, so implementing them with loops is not acceptable. Instead, you need to find the shortest possible straight-line sequence of instructions that will do the reversal. Hint: You may find the `.rept` directive to be useful.

The main program will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following two functions found in the C main program without using the RBIT, REV, REV16 or REVSH instructions. The original C versions have been defined as "weak" so that the linker will automatically replace them in the executable image by those you create in assembly; you do not need to remove the C versions. This feature allows you to write and test one assembly language function at a time.

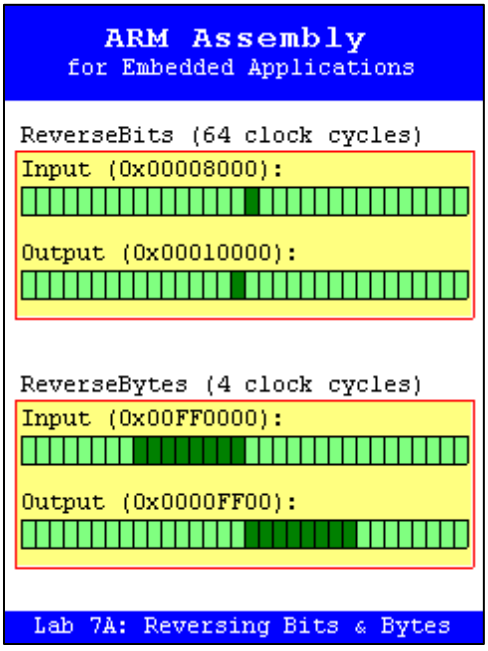
`uint32_t ReverseBits(uint32_t word) ;`

Returns a result that corresponds to reversing the order of all the bits in its input.

`uint32_t ReverseBytes(uint32_t word) ;`

Returns a result that corresponds to reversing the order of all the bytes in its input.

Test your functions using the C main. If your code is correct, the display should look similar to the following image, with the dark green cells representing 1's that rotate through the bit pattern. An incorrect result will display the incorrect bit pattern in red, pause the program, and wait for you to press the pushbutton to proceed. The execution time of each of your functions is displayed in clock cycles.



[1] https://en.wikipedia.org/wiki/Endianness