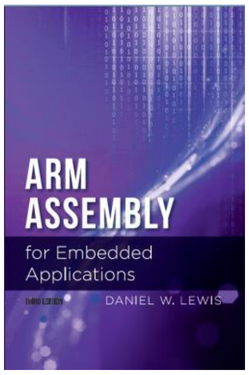


Programming Lab #6

Reversing Bits and Bytes

Prerequisite Reading: Chapters 1-7
Revised: November 1, 2017

Background: 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.

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.

Create an assembly language file containing three functions. The first is used for run-time performance comparisons and is to be implemented exactly as shown below:

```
CallReturnOverhead:   BX   LR
```

You are to code the other two without using the RBIT, REV, REV16 or REVSH instructions:

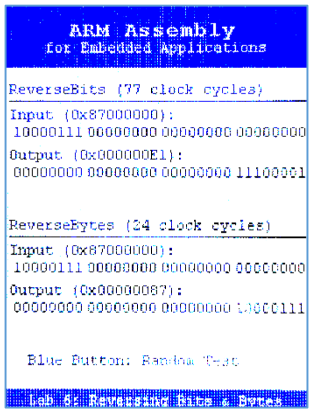
`uint32_t ReverseBits(uint32_t word) ;`

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 program that can be downloaded here. If your code is correct, the display should look like the following image, although your cycle counts may differ. Incorrect values will be displayed using white text on a red background.



```
ARM Assembly
for Embedded Applications

ReverseBits (77 clock cycles)
Input (0x87000000):
 10000111 00000000 00000000 00000000
Output (0x000000E1):
 00000000 00000000 00000000 11100001

ReverseBytes (24 clock cycles)
Input (0x87000000):
 10000111 00000000 00000000 00000000
Output (0x00000087):
 00000000 00000000 00000000 10000111

  Blue Button: Random Test

Lab 6: Reversing Bits & Bytes
```