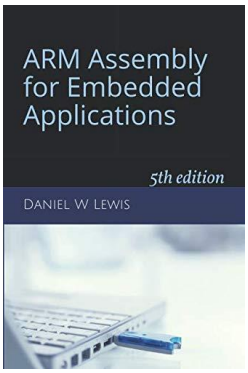


Programming Lab #3

Copying Data Quickly

Topics: Load and store instructions, unrolling loops, the .rept directive, comparing execution times.

Prerequisite Reading: Chapters 1-4
Revised: March 22, 2020

Create five functions in ARM Cortex-M4 assembly all contained in a single source code file. Each function copies 512 bytes of data from one array to another.

Each function should use the .REPT and .ENDR directives shown in Listing 4-1 to copy the data without a loop using a straight-line sequence of instructions. The main program (download here) will display the relative execution time of the three functions.

`void UseLDRB(void *dst, void *src)`

Copy 1 byte at a time using `LDRB` and `STRB`, and optimize the execution time by updating the address using the Post-Indexed addressing mode shown in Table 4-6.

`void UseLDRH(void *dst, void *src)`

Copy 2 bytes at a time using `LDRH` and `STRH`, and optimize the execution time by updating the address using the Post-Indexed addressing mode shown in Table 4-6.

`void UseLDR(void *dst, void *src)`

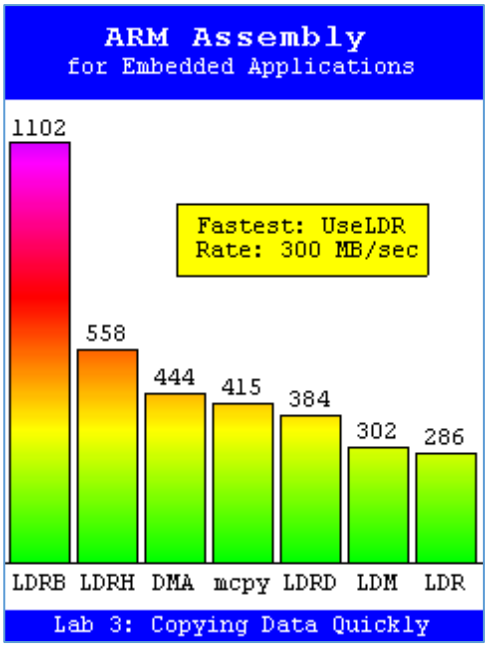
Copy 4 bytes at a time using `LDR` and `STR`, and optimize the execution time by updating the address using the Post-Indexed addressing mode shown in Table 4-6.

`void UseLDRD(void *dst, void *src)`

Copy 8 bytes at a time using `LDRD` and `STRD`, and optimize the execution time by updating the address using the Post-Indexed addressing mode shown in Table 4-6.

`void UseLDM(void *dst, void *src)`

Copy 32 bytes at a time using `LDMIA` and `STMIA`, and optimize the execution time by updating the address using the write-back flag (!) shown in Table 4-7.



If your code is correct, the display should look similar to the image at right with each function's execution time shown in clock cycles at the top of each bar graph. (Your numbers may differ, and the bar graph of an incorrect copy will be displayed in solid red.)

The bar graph labeled “mcpy” shows the clock cycle count for the library function `memcpy`, and the graph labeled “DMA” is the clock cycle count for a function provided in the main program that uses direct memory access. Note that once initialized by software, DMA transfers have the advantage of being independent of instruction execution so that both can continue concurrently.