ECE353 In Class Exercise

ARM Assembly – Load/Store Instructions

Problem 2A

Background Information

In computer science, a **lookup table** is an array that replaces runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is often faster than undergoing an 'expensive' computation or inut/output operation. (Wikipedia)

Requirements

1. Create symbolic constants for BYTE, HALF_WORD, and WORD that represent the number of bytes that each data type reserves in memory. Make sure there are no spaces prior to defining the constant. A constant in the ARM assembler is similar to #define in C.

```
BYTE        EQU   1
HALF_WORD   EQU   2
WORD        EQU   4
```

2. Create an 8 entry lookup table (**LTABLE**) in FLASH that contains the cubed of 'a' where $0 \leq a \leq 7$. Each value in **LTABLE** is an unsigned 16-bit number

```
(LTABLE[0] = 0, LTABLE[1] = 1, LTABLE[2] = 8,…)
```

3. Create two half word arrays in **SRAM** called **ARRAY1**, **ARRAY2**. Each array should have 8 entries
4. Load the address of **LTABLE** into **R0**
5. Load the address of **ARRAY1** into **R1**
6. Load the address of **ARRAY2** into **R2**
7. Place the the value of 6^3 into R10 using **LTABLE**.
8. Copy the contents of **LTABLE** into **ARRAY1** using the half word versions of **LDR/STR**. Use *pre-indexed* load/stores
9. Copy the contents of **LTABLE** into **ARRAY2** using the half word versions of **LDR/STR**. Use *post-indexed* load/stores
10. Verify the contents of **ARRAY1** and **ARRAY2** in the memory window.

Problem 2B

Requirements

1. Add a byte array in **FLASH** called **ASCII_ARRAY**. It should contain the ASCII string "ECE353".
2. Allocate a **WORD** sized global variable in **SRAM** called **RESULT**
3. Load the address of **ASCII_ARRAY** into **R0**. Do not modify **R0** after this instruction.
4. Load the address of **RESULT** into **R1**. Do not modify **R1** after instruction.
5. Write a while loop in ARM assembly that will add the binary value(s) of any ASCII characters that represent decimal numbers. For the string listed above, the total would be 3+5+3 = 11. The while loop will terminate once the end of the string is reached. Write the while loop as if you did not know the length of the string.

Notes

1. The ASCII characters that represent numbers are in the range of 48-57. Any values that fall outside of that range are discarded. To get the binary value of a valid number, subtract 0x30 from the ASCII value.
2. A string will end with a NULL character (0x0). Exit the while loop when the current character equals 0.
3. You will need to use labels, branches, and conditional instructions to implement a while loop. You can make a branch conditional by appending a condition code at the end of the branch instruction.

What to Turn in

You are encouraged to work with your classmates to complete this exercise. Each student will in their version of ece353_main.s for problem 2A. Submissions are evaluated for overall understanding of the concepts. They are not going to be evaluated based on if the code is 100% correct.