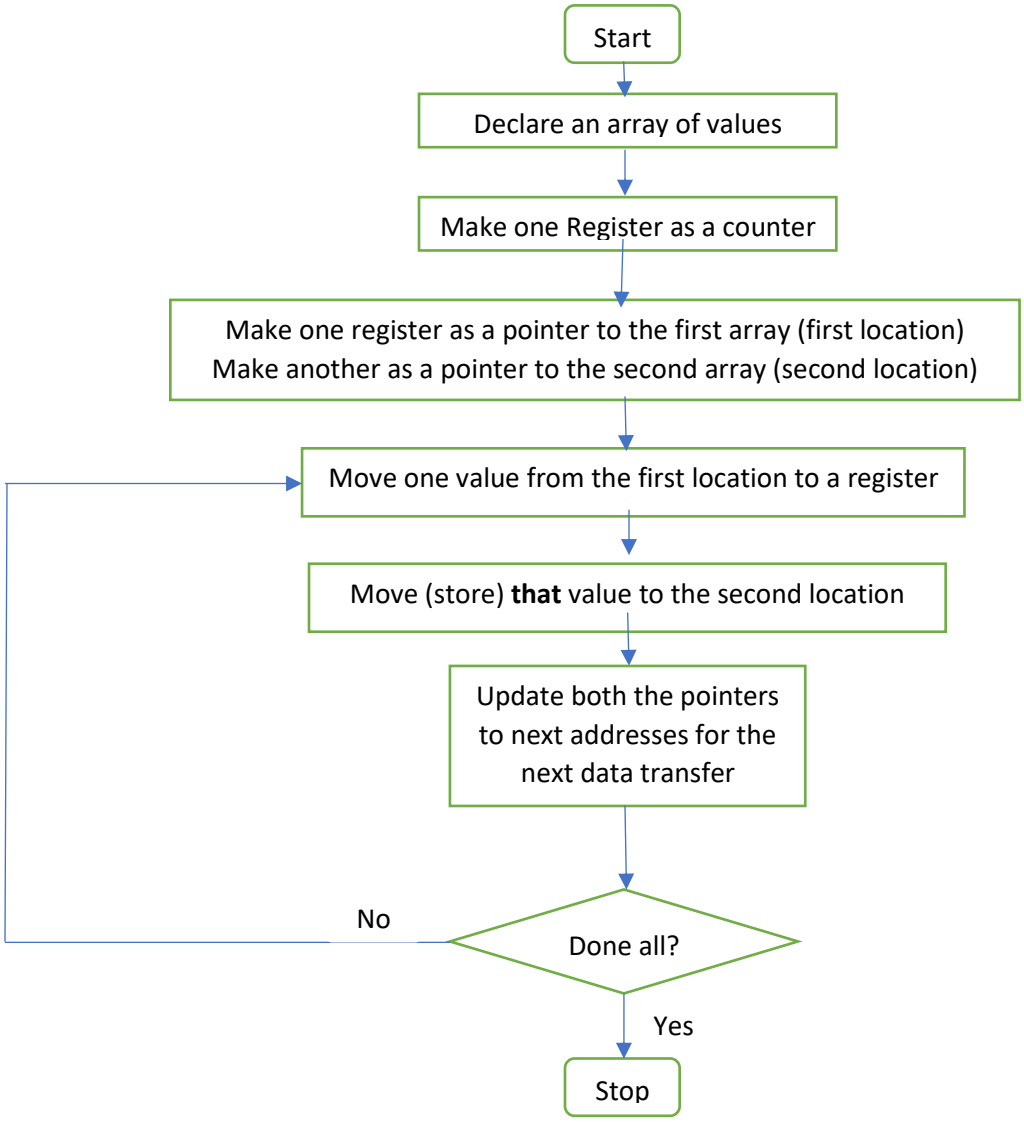
Department of Engineering Technology and Industrial Distribution

ESET 349 Microcontroller Architecture

Lab 2: Programming for data transfer between registers and memory locations and to find a particular value in the memory

Lab Introduction

This lab works with a simple program on data transfer between registers and memory locations. The concept of this program can be described using the following flowchart.



Objective

The objectives of this lab are to:

a) Understand how to declare data outside the code segment
b) Repeat a program segment using a loop (like do-while loop in high level language)
c) Move data from one location to a second location
d) Investigate data in the memory locations by debugging
e) Understand the padding of zeros to smaller-sized values in the case of instructors operating with 32-bit data.
f) Find the largest and the smallest values from the array

Program provided at the bottom of the lab manual.

Your Tasks:

<u>Task 1</u>: 1-Byte array

I. You can declare an array of data according to the following example:

```
                area lab2, code, readonly
byteData        dcb 0x23, 0xAB, 48, 0x9F, 0xFF
                export __main
__main          proc
```

> DC**D** means Define Constant Double (4-byte value) and DC**B** is Define Constant Byte (1-byte value). Example shows five 8-bit (1 byte) values declared at a memory with identifier, **byteData**. byteData is now a pointer to the address of the first value.

II. To move a value larger than 12-bit size value, use LDR statement instead of a MOV statement, example:
- ldr r4, =0x12345678; notice the '=' sign instead of '#' sign. R4 now has a 4-byte value. This could be a value or an address depending on how we use it.
- ldr r8, =0xABCD7856

III. To move a value from a memory location to a register, look at the following example,

```
ldr r9, [r4]
```

This means the value in the memory, pointed to by register r4, is moved (loaded) to register r9

IV. Note the following instruction carefully:

str r8, [r4]

r4 is inside a pair of square brackets. Thus, r4 has taken the role as a pointer to address, 0x12345678, in the memory. The value in register r8 has been stored (str) to the memory location 0x12345678 as pointed to by r4.

Using the 5 given hex numbers, move the values from the source address (original location in memory) to the target address (new location in memory). For each iteration, document the values in **Table 1** as shown below. Add extra lines if needed.

Target address: starting from 0x20002000

Table 1

Count	Value to be moved	Source Address	Target Address

Task 2: Memory map for 1-byte array

Now you have the given data at two locations as per your program. Create two tables (memory maps) for location 1 (source) and location 2 (target) as given below in **Table 2**.

Table 2 (a)

Source Address (location 1)

Memory Address	Byte Stored

Table 2 (b)

Target Address (location 2)

Memory Address	Byte Stored

Task 3: Memory map for 4-byte array

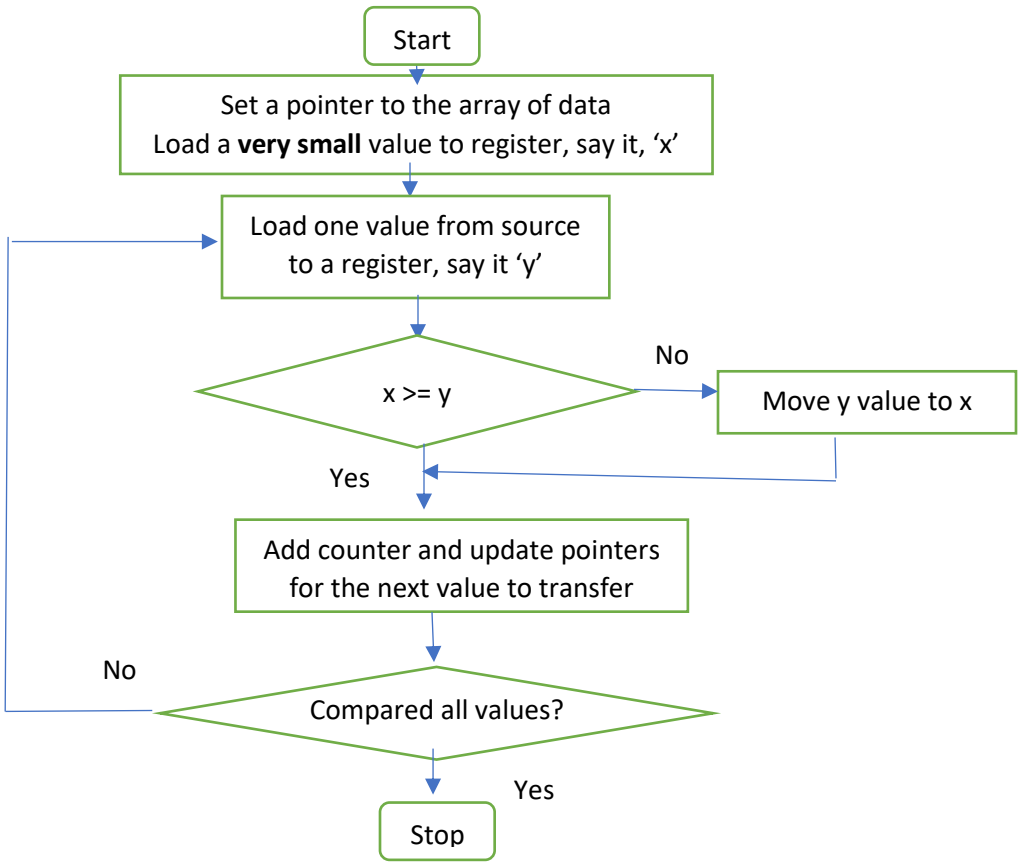
Create a memory map (similar to Task 2) for the source & target address of the 4-byte array. For the 5 values in the array, the total length of each table would be 20 (=5 x 4 bytes) rows. However, show the tables for only the first three data movements (3 x 4 bytes = 12 rows)

Target address: starting from 0x20004000

Values to be moved: 0x12345678, 0x2BCD1234, 0x1234ABCD, 0x4FAABBCC, 8

Task 4: Largest value of an array

The following flowchart shows the process of finding the largest value from an array.



Observe the data movement between memory locations and in registers during Debug session while finding the largest value. The following program finds the largest value. Make appropriate modifications as needed.

Lab 2 Code

```
            area lab2code, code, readonly
byteData    dcb 0x23, 0xAB, 48, 0x9F, 0xFF  ; data can be declared in 'code' area
byteDataLen equ 5
wordDataLen equ 5

            export __main
__main      proc

; TASKS 1&2
; moving (dealing) one-byte (8-bit) data
            ldr r0, =0x20002000     ; r0 is the pointer to new location
            ldr r1, =byteData       ; r1 is the pointer to byteData
            mov r12, #0             ; r12 is used as a counter
moreBytes   ldrb r3, [r1]           ; load one value from array in memory
            strb r3, [r0]           ; store to new location
            add r12, #1             ; moved one, so increase counter
            add r0, #1              ; set pointer to new locations for both
            add r1, #1
            cmp r12, #byteDataLen
            bne moreBytes

; TASK 3
; moving (dealing) with 4-byte data
            ldr r0, =0x20004000
            ldr r1, =wordData
            mov r12, #0
moreWords   ldr r3, [r1]
            str r3, [r0]
            add r12, #1
            add r0, #4              ; pointers added by four for 4-byte data
            add r1, #4
            cmp r12, #wordDataLen
            bne moreWords

; TASK 4
; Find the largest value from the word data array
            mov r8, #0          ; we start with a small value to find the largest
                                ; start with a very big value to find the smallest
            ldr r1, = wordData
            mov r12, #0
moreToGo    ldr r3, [r1]
            cmp r3, r8
            blt skip            ; bgt for finding the smallest value

            mov r8, r3          ; keep the latest large value in r8
skip        add r12, #1         ; go for next value
            add r0, #4
            add r1, #4
            cmp r12, #wordDataLen
            bne moreToGo

            endp


            area lab2data, data, readonly   ; data can be declared in 'data' area too
wordData    dcd 0x12345678, 0x2BCD1234, 0x1234ABCD, 0x4FAABBCC, 8

            end
```

Note: At checkout, show all your activities to your TA.