Assignment 3B (50 marks) – Lab Week Ten

Due: End of your week Twelve's lab period: Week of 24 – 27 Nov 2015

Late submissions will not be accepted and will receive a mark of zero (0).

APPROVED
By David Haley at 12:42 pm, Oct 30, 2015

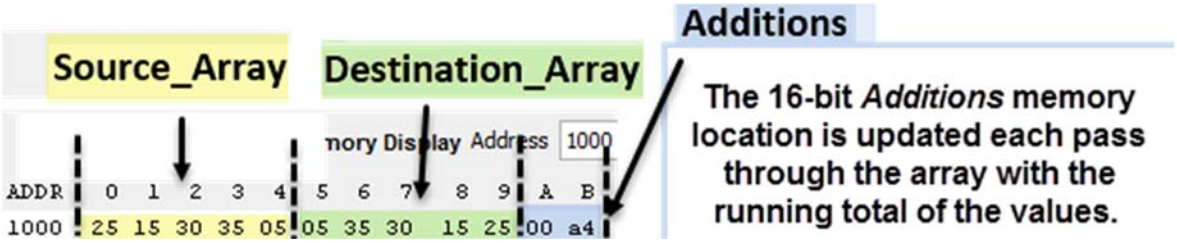
This lab exercise may be optionally performed by up to THREE students working as a group (students pick their own partners from the SAME lab section). If you are in a multi-student group, ensure all student names/numbers are on all program listings and documentation in order for all students to receive credit for the work (No name, no credit).

Task One – Working with Pointers, Arrays and Arithmetic (20 marks)

Write a complete program in Assembly Language (***A3B_Array.asm***) that effectively uses iteration and pointers to copy the values from **Source_Array** to **Destination_Array** and places the copied values backwards in memory. e.g. the last element from **Source_Array** is the first element in **Destination_Array** and so on.

Inside the same loop, you are to total the values (they are all unsigned – do you know why?) in the **Source_Array** *as you iterate through the loop*. Think carefully what registers to use in your solution. Zeroize the 16-bit value of **Additions** before your loop and store all of the intermediate addition results at **Additions** during each loop.

Here is an example program run using an **example** data set. Note that this is NOT the data set you are to use in your solution to this problem.



The following code must be used within your solution to create the S**ource_Array, Destination_Array,** and reserved memory space for **Additions**

```
                org     $1000
Source_Array
                db      $06, $55, $2C, $7A, $07
Destination_Array ds    Destination_Array-Source_Array    ; auto calculate
                                                          ; Array Size
Additions       ds.w    1                                 ; store 16 bit
                                                          ; additions here
```

Your program code should commence at $2000, and the first line of code should initialize the stack to a value of $2000 using **lds** #STACK.

Your solution should realize the correct value in **Additions** for any number of program runs (without closing the simulator and starting it again) – e.g. if **Additions** = $00A4 the first program run, then if we run the program for any number of times, **Additions** must = $00A4 again. Create a Test Plan that illustrates each operation of the program and the intermediate values of the **A**, **B** and **D** Registers and the Memory designated by **Additions**. Ensure you understand how the data is added by manually going through the data before you code the solution.

Demonstration:

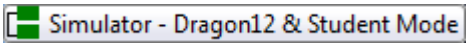
- Your program run must be demonstrated using the 

and credit for the demonstration will only be given to students in your group who are present for the demonstration
- Only **one** demonstration per group is permitted – the software is either 100% functional or it is defective, in which case you will receive minimal marks for your demonstration
- Note that only students in **your** lab period may be members of the group – no exceptions!

Code and Test Plan Submission

- You are required to submit a **hard copy** of ***A3B_Array.asm*** and your ***Test Plan*** at the time of your demonstration
- **Ensure all group members' names are included in the HEADER of your code submission and the Test Plan. Otherwise, no credit will be given to omitted group members.**

Task Two – Sorting Data Using Pointers (20 marks)

Given the documentation (on Blackboard) for the Bubble Sort Algorithm, you are to sort the following unsigned data in ascending (smallest to largest) order, coding your solution (***Bubble_Sort.asm***) in HCS12 Assembly Language using AsmIDE and the 

$FF, $FE, $03, $04, $80, $01, $FE, $02, $00

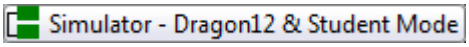
Constraints:

- Your solution must be implemented in HCS12 Assembly Language using the supplied Bubble Sort Algorithm
- Except for **org** statements, do not hard-code any addresses – use Labels as appropriate
- Load the array into memory from a file as taught in lecture
- Create a copy of the original array and correctly sort the values from lowest to highest. In both cases, you must use iteration to do so.
- Do not "hard code" the length of the array. Rather, use the method taught in lecture to dynamically calculate the length of the Destination Array
- Use appropriate CONSTANTS, Labels and Comments
- Create a Memory Map of the Pre and Post Memory contents of both arrays. Have it printed out for your demonstration.

Extra Credit: (No assistance given for these bonus marks – don't spend too much time on this)

- Optimize the code so that as a value is "bubbled" up to the top of the array (e.g. the right-most position), the next loop omits checking that value, effectively "shrinking the size of the array" each loop.

Demonstration:

- Your program run must be demonstrated using the 

and credit for the demonstration will only be given to students in your group who are present for the demonstration
- The resulting program run should clearly illustrate the original array's contents unchanged and the sorted array's contents as per the Memory Map you will have printed out prior to the demonstration
- Only **one** demonstration per group is permitted – the software is either 100% functional or it is defective, in which case you will receive minimal marks for your demonstration
- Note that only students in **your** lab period may be members of the group – no exceptions!

Code and Memory Map Submission

- You are required to submit a **hard copy** of **Bubble_Sort.asm** and your **Memory Map** at the time of your demonstration.
- **Ensure all group members' names are included in the HEADER of your code submission and your Memory Map. Otherwise, no credit will be given to omitted group members.**

Task Three – Understanding The Stack (10 marks)

See page 2 of the Assignment Hand-In Sheet.

Assignment 3B (50 marks) – Lab Week Ten (Page 1 of 2)

Due: End of your week Twelve's lab period: Week of 24 – 27 Nov 2015

Late submissions will not be accepted and will receive a mark of zero (0).

Please staple the pages together.

This lab exercise may be optionally performed by up to THREE students working as a group (students pick their own partners from the SAME lab section). If you are in a multi-student group, ensure all student names/numbers are on all program listings and documentation in order for all students to receive credit for the work (No name, no credit).

Name: _____ Name: _____

Student Number: _____ Student Number: _____

Name: _____

Student Number: _____

Circle Your Lab Period/Time

Tue: 10 – 12 Wed: 1 – 3 Wed 3 – 5 Fri 2 – 4

Assessment – It is recommended that you check your solution against the following marking rubric BEFORE the lab demo.

Task One – Working with Pointers, Arrays and Arithmetic (20 marks)

A. Demo of your solution in the Simulator Professors Initials: _____ /6
Note: For full demo marks, your solution must correctly implement the assignment instructions, including the correction addition of the values for any number of program runs.

B. Post-Lab Code Inspection – hand in a hard copy of your code ***A3B_Array.asm*** and your ***Test Plan*** at the time of your demonstration. Evaluated as follows:

- ✓ Code conforms to assignment instructions – use of iteration, correctly adding numbers in the same loop, use of given Labels, etc. /6
- ✓ Test Plan illustrates each operation of the program and the intermediate values of the **A**, **B** and **D** Registers and the memory designated by **Additions**. /4
- ✓ Code meets course standards – header, documentation, indentation, comments, printed from AsmIDE /4

Task Two – Sorting Data Using Pointers (20 marks)

A. Demo of your solution in the Simulator Professors Initials: _____ /6
Note: For full demo marks, the resulting program run should clearly illustrate the original array's contents unchanged and the sorted array's contents as per the Memory Map you will have printed out prior to the demonstration.

A. Post-Lab **Code** Inspection – hand in a hard copy of your code ***Bubble_Sort.asm*** and your **Memory Map** at the time of your demonstration. These will be evaluated as follows:

/14

- ✓ Your solution must be implemented in HCS12 Assembly Language using the supplied Bubble Sort Algorithm
- ✓ Except for org statements, no hard-coded addresses – used Labels as appropriate
- ✓ Loaded the array into memory from a file
- ✓ Created a copy of the original array and correctly sorted it using iteration to copy and sort
- ✓ Dynamically calculated the length of the Destination Array
- ✓ Used appropriate CONSTANTS, Labels and Comments
- ✓ Memory Map contains the Pre and Post Memory contents of both arrays
- ✓ Code conforms to assignment instructions – use of iteration, correctly adding numbers in the same loop, use of given Labels, etc.
- ✓ Code meets course standards – header, documentation, indentation, comments, printed from AsmIDE

Extra Credit: (No assistance given for bonus marks – don't spend too much time on this) up to 5 bonus marks
Optimize the code so that as a value is "bubbled" up to the top of the array (e.g. the right-most position), the next loop omits checking that value, effectively "shrinking the size of the array" each loop.

Assignment 3B (50 marks) – Lab Week Ten (Page 2 of 2)

Due: End of your week Twelve's lab period: Week of 24 – 27 Nov 2015

Late submissions will not be accepted and will receive a mark of zero (0).

Please staple the pages together.

This lab exercise may be optionally performed by up to THREE students working as a group (students pick their own partners from the SAME lab section). If you are in a multi-student group, ensure all student names/numbers are on all program listings and documentation in order for all students to receive credit for the work (No name, no credit).

Task Three – Understanding The Stack (10 marks)

Answer all questions in the space provided on this page.

a. Given the following .lst file, answer the following questions ***by manually tracing through the code***. **(10 marks)**

```
4                                   ; The_Stack.asm
5  2000                     Stack   equ     $2000
6
7  1000                             org     $1000
8  1000 93 53 21 11 10      Data    db      $93, $53, $21, $11, $10
9
10 2000                             org     $2000
11 2000 cf 20 00                    lds     #Stack
12
13 2003 ce 10 00                    ldx     #Data
14 2006 a6 00                       ldaa    0,x
15 2008 e6 30                       ldab    1,x+
16 200a 36                          psha
17 200b 37                          pshb
18 200c 34                          pshx
19 200d 16 20 14                    jsr     Here
20 2010 30                          pulx
21 2011 3a                          puld
22 2012 20 05                       bra     Finish
23 2014 b6 10 02            Here    ldaa    Data+2
24 2017 09                          dex
25 2018 3d                          rts
26 2019 3f                  Finish  swi
27                                  end
```

A. Complete the following table of memory contents after the execution of line 19. Where the value is unknown, enter " - -"

$1FFA	*$1FFB*	*$1FFC*	*$1FFD*	*$1FFE*	*$1FFF*

B. Complete the following table of values for the contents of the registers after the execution of line 25.

A	B	X	PC	SP

C. Complete the following table of values for the contents of the registers after the execution of line 21.

A	B	X	PC	SP

Your Mark /10