Assignment 3B (50 marks) – Lab Week Ten

Due: End of your Week Twelve's Lab Period – Week of 8 – 14 April 2018 (see exception for Task Two's code submission).

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

Note: Your mark on Blackboard will be normalized to a mark /5

Any code submitted without a Name and Student Number affixed receives a mark of Zero.

This lab exercise may be optionally performed by up to **THREE** students **in the same lab section** working as a group (students pick their own partners). Also, 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). **There is only ONE submission required per group of students.**

Task One – Working with Arrays and Pointers (15 marks)

Write a complete program in Assembly Language (***Exchange_Elements.asm***) that effectively uses iteration and pointers to swap the first element with the last element of an array, the second element with the second-to-last element, and so on. The array has twenty-five 8-bit elements and is stored at $1000.

Some of the HCS12 Instruction set that you will likely use in this assignment, in the IMM, EXT and IDX addressing modes are: **ldx, ldy, ldaa, ldab, staa, stab, cpx, bne**. As such, you should review their use as taught in class and in the Almy Text and the S12CPUV2 Reference Manual.

Pre-Program Run Memory Map – Original Array

ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	21	6e	75	46	20	73	69	20	65	67	61	75	67	6e	61	4c
1010	20	79	6c	62	6d	65	73	73	41							

Post-Program Run Memory Map – Elements Exchanged

ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	41	73	73	65	6d	62	6c	79	20	4c	61	6e	67	75	61	67
1010	65	20	69	73	20	46	75	6e	21							

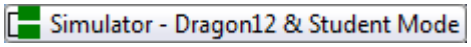
Constraints:

- The following code must be used in your solution to create the array elements and to dynamically determine the length of the array.

```
        org     $1000
Elements        db 033, 110, 117, 070, 032, 115, 105, 032, 101, 103, 097
                db 117, 103, 110, 097, 076, 032, 121, 108, 098, 109, 101
                db 115, 115, 065
EndElements
Len     equ     EndElements-Elements   ; sizeof() = Length of the array
```

- Your solution must be implemented in HCS12 Assembly Language using structured programming methods
- Create a constant called STACK that has the value of $2000
- Your program code must commence at $2000, and the first line of code should initialize the stack to a value of $2000 using lds #STACK.
- You may **NOT** use the Stack to simply push the values onto the Stack and then pull them off backwards
- You may **NOT** use another array in your solution
- Except for **org** and **equ** statements, do not hard-code any addresses – use Labels as appropriate
- Since the length of the destination array has been dynamically calculated, there is no requirement to hardcode any of the array lengths, nor is there any requirement to have a counter for iteration
- Use other equate (equ) statements for any other dynamically created values you may need
- Use the provided array Labels for iteration purposes – no absolute memory addresses are to be used in any comparison
- Iteration and pointers must be used
- Use appropriate CONSTANTS and Labels
- Code meets course standards – structured programming, header, documentation, indentation, comments, printed from AsmIDE

Demonstration:

- Your program run must be demonstrated using 

during your lab period 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

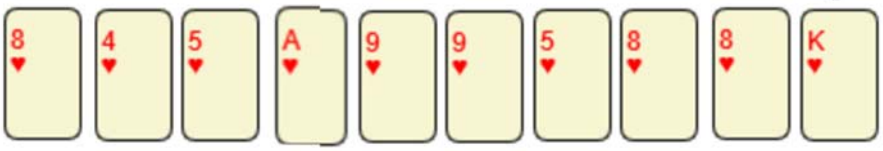
Code Submission

- You are required to submit a **printed copy** of ***Exchange_Elements.asm*** into your portfolio folder as per assignment due dates/times.

Task Two – The Selection Sort – Sorting Data Using Pointers and the Indexed Addressing Mode (25 marks)

Write a complete program in Assembly Language (**Unsigned_*Selection_Sort.asm***) that effectively uses iteration, pointers and the Indexed Addressing Mode to sort the following playing cards using the provided Selection Sort Algorithm and Pseudocode.

Pre-Sort Arrangement of Cards

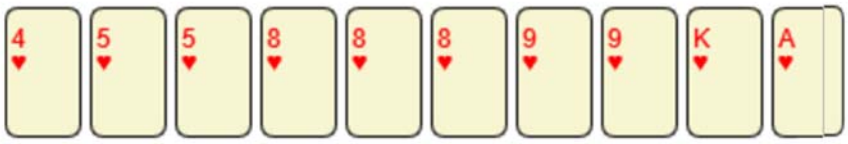


represented in memory as 8-bit **unsigned** values as illustrated in the screen capture of the program run.

Pre-Program Run Memory Map – Array Unsorted

ADDR	0	1	2	3	4	5	6	7	8	9
1000	08	04	05	0e	09	09	05	08	08	0d

Post-Sort Arrangement of Cards



Post-Program Run Memory Map – Array Sorted

ADDR	0	1	2	3	4	5	6	7	8	9
1000	04	05	05	08	08	08	09	09	0d	0e

A video illustrating the step-by-step sorting of these cards is on Blackboard (within SelectionSortCards.zip).

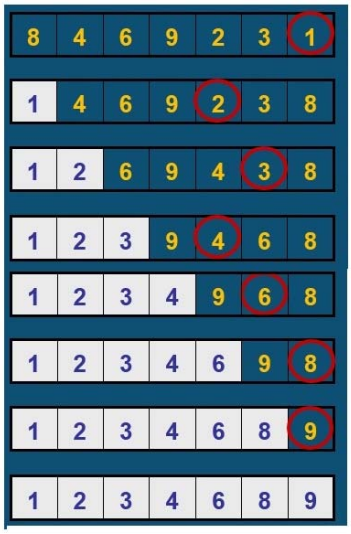
Once uncompressed, click on SectionSortCard.html to play the video in your browser. This video provides an excellent mechanism to see if your solution is working, following along in the simulator by setting a breakpoint at the point where you begin to exchange the two values in the deck of cards, and debugging your solution as you go along.

Selection Sort Algorithm

Source: http://freefeast.info/general-it-articles/selection-sort-pseudo-code-of-selection-sort-selection-sort-in-data-structure/

Idea of a Selection Sort Algorithm, with an example shown to the right:

- First it finds the smallest element in the array.
- Exchange that smallest element with the element at the first position.
- Then find the second smallest element and exchange that element with the element at the second position.
- This process continues until the complete array is sorted.



Pseudocode of Selection Sort

```
SELECTION-SORT(A)
    for j ← 1 to n-1
        smallest ← j
         for i ← j + 1 to n
               if A[ i ] < A[ smallest ]
                     smallest ← i
          Exchange A[ j ] ↔ A[ smallest ]
```

Some of the HCS12 Instruction set that you will likely use in this assignment, in the IMM, EXT and IDX addressing modes are: **ldx, ldy, ldaa, ldab, staa, stab, tfr, cpx, cpy, jsr, iny, inx, bne, unsigned branch statements,** As such, you should review their use in the Almy Text and the S12CPUV2 Reference Manual.

Constraints:

- All Constraints from Task One MUST be followed (**Review those**)
- Modified from Task One Constraints – You may **NOT** use another array in your solution
- You **must** use the provided Selection Sort Algorithm to sort the data within the **same** array
- Well-formed code that includes Iteration and Indexed Addressing Modes **must** be used. "Spaghetti code" will receive a mark of zero.
- Your program **must not** make us of stack pushes and pulls to sort the data.
- **Undocumented code will receive a mark of zero**.

Important Notes for Creating the Solution and Asking for Assistance, if required

- ***Solve the problem "on paper" first*** – understanding and keeping track of your pointer(s) and what you need to do with the data is paramount. I spent about an hour and a half solving the problem on paper and then 30 minutes coding and debugging the software.
- It is highly recommended that you create a flow chart that illustrates procedurally what steps must be taken to solve this problem and then write code beside the flowchart symbols to assist you in the implementation of your problem solution; otherwise, you may wish to transform the Pseudocode into a form that would assist you in Assembly Language. E.G. any reference to an Array Index in the Pseudocode is simply an address in Assembly Language.
- If you are having difficulty with the code, create small snippets of code outside of the program to see how the instruction set works and then incorporate the correct code back into your program
- If you run into trouble, I can assist you in understanding of the problem you are trying to solve; however, you must be able to explain **what** you are attempting to do, what you have tried (with examples) and where your problem area(s) are in implementing the solution. You must show me either your flowchart or Pseudocode
- I can explain the instruction set; however, I cannot code for you.
- The only "code checking" I will perform if you are having problems is a quick parse to see if any of the addressing modes are incorrect. Other than that, you must use the simulator to debug any faulty code

Demonstration:

- Your program run must be demonstrated using 

during your lab period 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 Submission

- You are required to submit a soft copy of **Unsigned_*Selection_Sort.asm*** via the Assignment 3B Submission link on Blackboard **Friday**, **April 13, 2018 @12:59 pm**. Otherwise, your code submission is late, and I cannot perform Post-Lab Code Analysis.

- Ensure all group members' names are included in the HEADER of your code submission. Otherwise, no credit will be given to omitted group members.

There is only ONE submission required per group of students.

Task Three – Understanding the Stack (10 marks)

Complete the program tracing exercise on page 3 of the Hand-In Sheet.

Assignment 3B (50 marks) – Lab Week Ten – Hand-In Sheet (Page 1 of 3)

Due: End of your Week Twelve's Lab Period – Week of 8 – 14 April 2018 (see exception for Task Two's code submission).

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 **in the same lab section** working as a group (students pick their own partners). Also, 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: _____ Student Number: _____

Name: _____ Student Number: _____

Name: _____ Student Number: _____

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

Indicate Your Lab Period

_____

Task One – Working with Arrays and Pointers (15 marks)

A. **In-Lab Demo** of your solution in the Simulator (Only **one** demo permitted) Credit for the demonstration will only be given to students in your group who are present for the demonstration Professors Initials: _____ /5

- ✓ Swapped the first element with the last element of an array, the second element with the second-to-last element, and so on. The array has sixteen 25-bit elements and is stored at $1000.

Pre-Program Run Memory Map – Original Array

```
ADDR   0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
1000  21 6e 75 46 20 73 69 20  65 67 61 75 67 6e 61 4c
1010  20 79 6c 62 6d 65 73 73  41
```

Post-Program Run Memory Map – Elements Exchanged

```
ADDR   0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
1000  41 73 73 65 6d 62 6c 79  20 4c 61 6e 67 75 61 67
1010  65 20 69 73 20 46 75 6e  21
```

B. **Post-Lab Code Analysis** – hand in a hard copy (printed copy) of your code ***Exchange_Elements.asm***. Ensure all group members' names are included in the HEADER of your code submission – typed, not handwritten). Otherwise, no credit will be given to omitted group members. Your submission will be evaluated as follows:

Any code submitted without a Name and Student Number affixed receives a mark of Zero.

- ✓ Filename: ***Exchange_Elements.asm***
- ✓ Your solution must be implemented in HCS12 Assembly Language using structured programming methods
- ✓ Create a constant called STACK that has the value of $2000
- ✓ Your program code must commence at $2000, and the first line of code should initialize the stack to a value of $2000 using lds #STACK.
- ✓ You may **NOT** use the Stack to simply push the values onto the Stack and then pull them off backwards **– 10 marks if you do**
- ✓ You may **NOT** use another array in your solution **– 10 marks if you do**
- ✓ Except for **org** and **equ** statements, do not hard-code any addresses – use Labels as appropriate
- ✓ Since the length of the destination array has been dynamically calculated, there is no requirement to hardcode any of the array lengths, nor is there any requirement to have a counter for iteration
- ✓ Use other equate (equ) statements for any other dynamically created values you may need
- ✓ Use the provided array Labels for iteration purposes – no absolute memory addresses are to be used in any comparison
- ✓ Iteration and pointers must be used
- ✓ Use appropriate CONSTANTS and Labels
- ✓ Code meets course standards – structured programming, header, documentation, indentation, comments, printed from AsmIDE

There is only ONE submission required per group of students. /10

Assignment 3B (50 marks) – Lab Week Ten – Hand-In Sheet (Page 2 of 3)

Due Dates/Times

Please staple the pages together.

Due: End of your Week Twelve's Lab Period – Week of 8 – 14 April 2018

Task Two Code Submission: by Friday, April 13, 2018 @11:59 p.m.

Late submissions will not be accepted

Task Two – The Selection Sort – Sorting Data Using Pointers and the Indexed Addressing Mode (25 marks)

C. **In-Lab Demo** of your solution in the Simulator (Only **one** demo permitted). Credit for the demonstration will only be given to students in your group who are present for the demonstration Professors Initials: _____ /5

- ✓ All array values are sorted and reside in the same memory space originally occupied by the original array.

Pre-Program Run Memory Map – Array Unsorted

ADDR	0	1	2	3	4	5	6	7	8	9
1000	08	04	05	0e	09	09	05	08	08	0d

Post-Program Run Memory Map – Array Sorted

ADDR	0	1	2	3	4	5	6	7	8	9
1000	04	05	05	08	08	08	09	09	0d	0e

A. **Post-Lab Code Analysis** – submit a **soft** copy of ***Unsigned_Selection_Sort.asm*** **via the Assignment 3B Submission link on Blackboard by Friday, April 13, 2018 @11:59 p.m. Otherwise, your submission is late, and you will receive a mark of zero for the Post-Lab Code Analysis.** Ensure all group members' names are included in the HEADER of your code submission. Otherwise, no credit will be given to omitted group members.

Any code submitted without a Name and Student Number affixed receives a mark of Zero.

- ✓ Filename: ***Unsigned_Selection_Sort.asm***
- ✓ Adhered to Task One's constraints for this task
- ✓ The given array was sorted "in place" – no use of a second array **(-20 marks if data sorted into a second array)**
- ✓ Code conforms to assignment instructions – use of iteration, pointer(s) and Indexed Addressing Mode **(spaghetti code -20 marks)**
- ✓ Code meets course standards – structured programming, header, documentation, indentation, comments, printed from AsmIDE
- ✓ Code is functional post-lab
- ✓ Undocumented code will receive a mark of zero. This means that you must include documentation throughout your program, not just the bare minimum header information

/20

Bonus Marks

Assembly Language program can be a challenging, but rewarding undertaking. As such, there are 5 bonus marks possible to the one best program (each Lab Group) in each of the following categories:

- Best solution – least number of bytes of code that is well organized and completely solves the problem. My solution has 62 bytes of code).

- Best solution – program that is the easiest program to maintain – excellent level of relevant documentation. My solution has about 30 lines of code, but it also includes the complete algorithm and line by line comments so that the solution could be implemented in any programming language.

Note that you should ensure that you have a functional program **and** save it first before attempting to optimize it for Bonus Marks.

Assignment 3B (50 marks) – Lab Week Ten – Hand-In Sheet (Page 3 of 3)

Please staple the pages together.

Due Dates/Times

Task Three is due end of your Week Twelve's Lab Period – Week of 8 – 14 April 2018

Task Three – Understanding the Stack (10 marks)

Answer all questions in the space provided on this page.

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

```
1
2  asl2, an absolute assembler for Motorola MCU's, version 1.2h
3
4                          ; The_Stack.asm
5  2000                    Stack   equ     $2000
6
7  1000                            org     $1000
8  1000 19 fe 32 c4 65     Data    db      $19, $FE, $32, $C4, $65
9
10 2000                            org     $2000
11 2000 cf 20 00                   lds     #Stack
12
13 2003 cd 10 00                   ldy     #Data
14 2006 a6 41                      ldaa    1,y
15 2008 e6 70                      ldab    1,y+
16 200a 37                         pshb
17 200b 36                         psha
18 200c 35                         pshy
19 200d 16 20 14                   jsr     Here
20 2010 31                         puly
21 2011 3a                         puld
22 2012 20 05                      bra     Finish
23 2014 f6 10 03           Here    ldab    Data+3
24 2017 03                         dey
25 2018 3d                         rts
26 2019 3f                 Finish  swi
27                                 end
28
```

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	Y	PC	SP

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

A	B	Y	PC	SP