Assignment 4 (100 marks) – Lab Week Eleven – Final Assignment

Due:

Demos: By Week 13: 1 – 4 Dec 2015 during the lecture period or ANY lab period (without penalty)
Code Submission: On Blackboard by 1159 hrs on Friday, 4 Dec 2015 (see hand-in sheet)

APPROVED
By D. H. Haley at 9:49 pm, Nov 02, 2015

Lab periods in WP-214 are Tue 10 – 12, Wed 1 – 3, Wed 3 – 5, and Fri 2 – 4

See the Hand-In Sheet for details about program demonstration and documentation submissions.

Early Demos and Submissions are Most Welcome!

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

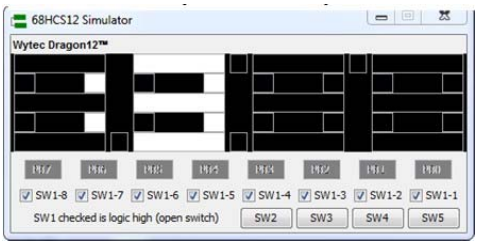
Note that if you missed your demo deadline, you can still get partial marks for the assignment by meeting the Documentation Submission deadline.

This lab exercise may be optionally performed by 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 – The BCD Counter – Demo Solution on the Wytec HCS12 Dragon12-Plus board (40 marks)

For this lab assignment, you are required to code a BCD counter using Assembly Language that continually counts from 00 → 99 → 00 → 99, etc. and correctly displays the count on the HEX Displays as illustrated below. Note that you must use **only one** of the Accumulators to hold your count; the ideal one to use based upon lecture discussions. Then, on an as-required basis, you will push and pull its value to and from the stack in order to save the count at the appropriate time(s) in your program.

To assist in your learning, I have included, a flowchart for the "main" Assembly Language program, some "skeleton code", which you may use to build upon for this assignment, and a couple of videos that illustrate the problem solution.



Instead of using the LEDs to display our count from 00→99→00→99 etc, we display the results on the 7-Segment HEX Displays of the Simulator to confirm that the display portion of our solution is correct. (LEDs' colour adjusted for printing purposes only).

Finally, once we have debugged our software and have confirmed that is FULLY functional, we will download it to the Dragon12-Plus HCS12 Trainer Board and run it there for demonstrations purposes.
You may wish to change the delay from 250 ms to 125 ms for this demonstration.

Development Requirements and Constraints

In order to be considered for full credit for this portion of the assignment, you must write a "main" Assembly Language program called ***Counter_$00_$99_BCD_HEX_Display.asm*** that solves the problem specification detailed in this document and illustrated in ***Counter_00_99_BCD_HEX_Display Flowchart***, which is located on Blackboard. (20 marks)

Here are the constraints for ***Counter_$00_$99_BCD_HEX_Display.asm***

- ✓ Use the supplied "skeleton code" for the assignment as a starting point for your coding solution. Note the "(DO NOT CHANGE ANY OF THE FOLLOWING LINES OF CODE)" statement in the skeleton
- ✓ Make use of CONSTANTS ***DIGIT3_PP0*** *and* ***DIGIT2_PP1*** as well as ***DELAY_VALUE***.
- ✓ **You MUST use ONLY one Accumulator as the count within this BCD Counter; otherwise little credit will be given for your solution.**
- ✓ **You MUST use the *stack* to temporarily store and retrieve the value of the BCD count. YOU MAY NOT USE any other methods to temporarily store and retrieve values; otherwise, little credit will be given for your solution.**
- ✓ Use AsmIDE to develop the program code and the Simulator to debug your solution.
- ✓ Use the Wytec Dragon12 Plus board to demonstrate your solution.
- ✓ Re-use library subroutine ***Delay_ms****, which should be found in the C:\68HCS12\Lib folder,* in your solution. This file was supplied to you in a previous lab. Do not change any code in that file.
- ✓ Appropriately use the supplied subroutine ***Config_HEX_Displays****, placing Config_Hex_Displays.asm in* C:\68HCS12\Lib. Do not change any code in that file.
- ✓ Fully document ***Counter_$00_$99_BCD_HEX_Displays.asm*** as per other previously supplied documented programs in this course.

In support of ***Counter_$00_$99_BCD_HEX_Displays.asm***, write the following Assembly Language subroutines:

- ***Extract_MSB*** – Subroutine to extract MSB (the upper nibble of the Accumulator) from an 8-bit Accumulator as per the following illustration: (5 marks)

- ***Extract_LSB*** – Subroutine to extract LSB (the lower nibble of the Accumulator) from an 8-bit Accumulator as per the following illustration: (5 marks)

Notes:

Each subroutine is to be in a separate file (**use the subroutine name + .asm as the filename – e.g. Extract_MSB.asm)** and fully document your subroutines as per the documentation standards contained in the supplied Config_Hex_Displays.asm file.

In support of ***Counter_$00_$99_BCD_HEX_Displays.asm***, use the supplied ***HEX_Display*** subroutine to display the MSB and LSB of the BCD count on the HEX displays, following the procedure outlined in the supplied flowchart. **Do NOT change any of the code in ANY supplied subroutine.** You should note that the Wytec hardware board's HEX Displays are "dumb" devices. As such, ***HEX_Display*** does not contain ***any*** iterative and/or decision-making statements (e.g. no loops or if-then-else type of statements) within the subroutine.

IMPORTANT PROGRAMMING CONCEPT

Subroutines NEVER call other subroutines. That is, a subroutine has one and only one function – e.g. the HEX_Display subroutine **cannot** call the provided Delay_ms subroutine because this subroutine knows nothing about other subroutines. Rather, a subroutine only accepts parameters (as applicable) from the calling "main" routine, which provides it arguments (as applicable) and returns values (as applicable) just like in any other higher-level programming language. "main" is the only portion of a program that may call subroutines.

See the Hand-In sheet for particulars of demos and submissions.

TASK Two – Pass-Fail Calculator – Demo Solution on the Wytec HCS12 Dragon12-Plus board (60 marks)

Congratulations! You have just received the contract to provide an assembly language program (***Pass_Fail_Calc.asm***) that evaluates a class of six students' CST816 marks and displays the results on the Wytec Dragon12+ Demo board.

In part, the course outline for CST8216 stipulates the following:

> *In order to pass this course, the student must have a grade of at least at least 50% or 'D- on Lab Exercises/Assignments/Hybrid activities (25/50) and at least 50% or 'D-' on Term Tests/Final Exam (25/50).*

You must write a fully documented solution that determines if each of the six students have passed CST8216 or not.

Part A – The Test Plan (6 marks)

The first thing you must do before writing any code, is to use the marks file appropriate to your lab section (located on Blackboard), and complete the test plan on the hand-in sheet so that you will know WHAT results your software solution should realize. Ensure you record which test plan you used (e.g. Wed 3 -5) so that your expected results can be compared against the correct marks file during your program demo.

Once the test plan has been created, take one student and "walk" their marks through the supplied program logic so that you know what all intermediate results as well as the final result should be when you actually code the solution

Once you have one student's results calculated, go through all of the other students' values so that you know what they should be as well. This will greatly assist you in the program debugging process!

Part B – The Program Logic

Using the following information, you are to write a fully documented "main" HCS12 Assembly Language program **(Pass_Fail_Calc.asm)** that implements the following program flow (in this EXACT order) for solving this problem. You may optionally create a high-level Visio flowchart if that would assist you in better understanding the problem solution

a. Configuring program constants;
b. Reading the file of marks, which contains marks for six students;
c. Calling subroutine **Calculate_Average** that calculates the integer average out of 50 for one student's Practical marks;
d. Calling subroutine **Pass_Fail** to determine from c. above if that student has passed or failed the Practical;
e. Calling subroutine **Calculate_Average** that calculates the integer average out of 50 for one student's Theory marks;
f. Calling subroutine **Pass_Fail** to determine from e. above if that student has passed or failed the Theory;
g. Looping back to c. to do the remaining students' marks calculations;
h. Calling subroutine **Config_HEX_Displays,** a subroutine to configure the Wytec Dragon12+ Demo board hardware to use the 7-segment Hex Displays;
i. Calling subroutine **PF_HEX_Display** that causes the Wytec Dragon12+ Demo board hardware to display **P** (for an overall Pass of the Course – there is only **one** output *per student*) or **F** (for an overall Fail of the Course – there is only **one** output *per student*) on the right-most 7-segment Hex Display. Additionally, display the **P** or **F** values for **1 sec**, then **blank** the display for **1 sec**;
j. Looping back to i. to display the remaining results; and
k. Continually looping back to this statement (k.) to maintain the **blank**ed Display.

Part C – Coding the Implementation (Total of 44 marks)

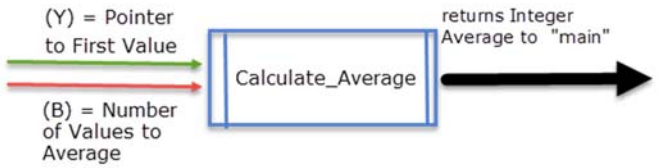
What you are to do for this part:

I. Write a fully documented "main" HCS12 Assembly Language program **(Pass_Fail_Calc.asm)** that implements the given **mandatory** Program Logic to this problem. **(20 marks)**
- Use the supplied skeleton code **Pass_Fail_Calc.asm** as your starting point, noting the instructions provided in that code
- Make maximum use of CONSTANTS in your program.
- Ensure that you use iteration in your solution. The **mandatory** Program Logic indicates **3 separate loops** within "main." Note that the last loop is just a line of code that always branches to itself.
- The library file Config_HEX_Displays.asm has been provided to you – see the instructions in that file for its use
- Your solution must be ***structured*** – e.g. **Pass_Fail_Calc.asm** MUST use the identified subroutines rather than having ALL of the code in **Pass_Fail_Calc.asm**. For example, it is **not** the responsibility of "main" to average values or determine a Pass or Fail. Rather, "main" calls subroutines to do its work by passing the subroutines values and receiving a result (where applicable) back from the subroutine.
- **Hints**: The result received by "main" could then be directly passed to another subroutine, used as part of a calculation, stored in memory or used for some other action.
- Ensure that all non-library files are written as separate files in the same folder as **Pass_Fail_Calc.asm**

Helpful Hints:

a. When you are ready to code and I recommend that you code the subroutines one-by-one and test them with some of the student data. Once you know that the subroutine is functional, incorporate it into the main program;
b. Continue on with this approach until you complete all of the subroutines that calculate the results; and
c. Now, work on the program output.

II. Write a fully documented subroutine **Calculate_Average** (using the skeleton code provide to you), that accepts two values as per the following illustration: **(14 marks)**
- a pointer (*use Index Register Y*) to the first value to use; and
- the number of values (in Accumulator B) to average



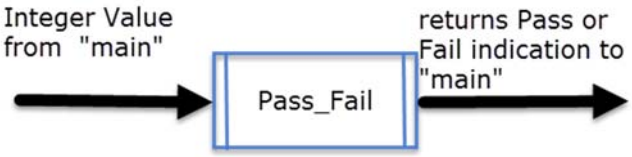
then adds up the values by using iteration, then divides the total by 5 and returns the integer average.

Make maximum use of CONSTANTS within the subroutine, especially the DIVISOR constant in the supplied source code.

Do **not** use an if-then-else structure in this subroutine as it is meant to serve **any** number of values, not just the three or five values in this application.

Note that your solution for **Calculate_Average** must work regardless of the number of values passed to the subroutine to average.

III. Write a fully documented subroutine **Pass_Fail** (using the skeleton provided to you that determines if the supplied integer average is a passing or failing average and returns the results **(7 marks)**



IV. Make maximum use of CONSTANTS within the subroutine.

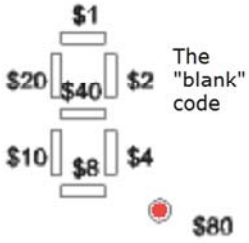
V. Complete the supplied skeleton code for subroutine **PF_HEX_Display** that converts the supplied value to a 'P' or 'F' segment value or **blanks** the selected display. (There is only one line of code that you must complete). **(3 marks)**

Helpful Information

Here are the 7-segment illustrations for 'P' and 'F' and the "blank."

7-segment Hex Display segment illuminating for 'P':

7-segment Hex Display segment illuminating for 'F':



IMPORTANT PROGRAMMING CONCEPT

Subroutines NEVER call other subroutines. That is, a subroutine has one and only one function – e.g. the Calculate_Average subroutine **cannot** call the Pass_Fail subroutine because they know nothing about each other. Rather, a subroutine only accepts parameters (as applicable) from the calling "main" routine, which provides it arguments (as applicable) and returns values (as applicable) just like in any other higher-level programming language. "main" is the only portion of a program that may call subroutines.

Final Note: Do not change any of the supplied Library files that I have provided to you.

See the Hand-In sheet for particulars of demos and submissions.

Assignment 4 (100 marks) – Lab Week Eleven – Hand-In Sheet

Due:

Please staple the pages together.

Early Demos and Submissions are most welcome!

Demos: By Week 13: 1 – 4 Dec 2015 during the lecture period or ANY lab period (without penalty)
Code Submission: On Blackboard by 1159 hrs on Friday, 4 Dec 2015 (see below)

Lab periods in WP-214 are Tue 10 – 12, Wed 1 – 3, Wed 3 – 5, and Fri 2 – 4

This sheet contains details about program demonstration and documentation submissions. Ensure you have this sheet with you when you demonstrate your solutions and that the sheet is submitted to your portfolio folder once all demonstrations have been completed or during Friday's lab period if you haven't demo'd.

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

Note that you do not have to demonstrate Task One and Task Two at the same time and that if you missed your demo deadline, you can still get partial marks for the assignment by meeting the Documentation Submission deadline.

This lab exercise may be optionally performed by 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 – The BCD Counter – Demo Solution on the Wytec HCS12 Dragon12-Plus board (40 marks)

A. Demo of your solution on the Hardware Board Professors Initials: _____ /10

Notes: For full demo marks, your solution must correctly implement the assignment instructions, specifically using the correct Digits on the Hardware Board and the display of only **valid** BCD counts.

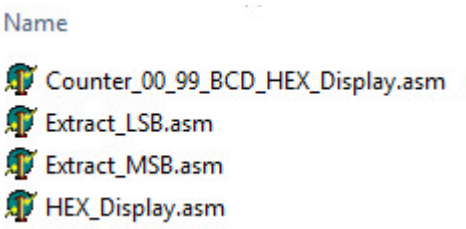
Credit for the demonstration will only be given to students in your group who are present for the demonstration

I recommend that you set the delay time for your hardware demonstration to 125 ms **AND** that you test your solution in the student version of the simulator (before demoing on the hardware board) using a delay of 250 ms.

Remember that there is only one demo permitted, so test your software in the Simulator first; then, test it on **the** Wytec HCS12 Dragon 12-Plus board to ensure that it works there **before calling me over to demo the software**. If it doesn't work on the hardware board, then load the software back into the simulator and check to ensure that you initialized all of your constants/variables and memory storage addresses at the beginning of the program.

B. Submit the following files to the ***Assignment Four – BCD Counter Link*** on Blackboard ***by 2359 hrs on Friday, 4 Dec 2015***

Place following files depicted to the right of this text into a single compressed .zip file using the following Naming Convention for the file name:



BCD_<Lab_Day_Time>_<Student_Last_Names>.zip

e.g. ***BCD_Wed_1_3_Gosling_Hooper_Kernighan.zip***

All hand-in sheets will either be returned to you Week 13 in your portfolios or they will be emailed back to you with your mark and my comments.

C. Post-Lab Code Inspection – your submitted code will be evaluated as follows:

I. Counter_$00_$99_BCD_HEX_Display.asm / 20

II. Extract_MSB.asm / 5

III. Extract_LSB.asm / 5

All code will be assessed as follows in descending priority:

a. Functionality – does the code correctly solve the problem according to the assignment instructions; wise use of iteration; have the registers been correctly used?
b. Maintainability – formatting of code; use of assembler directives, labels and CONSTANTS versus hardcoding values other than 0, 1, -1
c. Documentation: program title, header; meaningful comments that do not merely explain the instruction set
d. Optimal use of the instruction set

D. Post-Lab Program Run – I will assemble your files using the library files from their required locations to ensure that your solution works on the hardware board when independently assembled, downloaded and run.

/5

Note that the above 5 marks are intended to supplement your mark if you did not have time to demonstrate the program run by the deadlines. These marks, however, may be counted regardless of the status of your previous demonstration. So, if your original demo was correct, incorrect or not performed and the program runs correctly Post-Lab, then you will be awarded the five marks.

Your Assignment 4 – Task One Mark /40

TASK Two – Pass-Fail Calculator – Demo Solution on the Wytec HCS12 Dragon12-Plus board (60 marks)

A. Demo of your solution on the Hardware Board Professors Initials: _____ /10

Note: For full demo marks, your solution must correctly implement the assignment instructions, specifically using the correct Digits on the Hardware Board and display the Pass and Fail results for the set of six students' marks you have been assigned by lab period. Ensure that you have completed the Test Plan before your demon and that the display timing is as per the assignment instructions

Remember that there is only one demo permitted, so test your software in the Simulator first; then, test it on **the** Wytec HCS12 Dragon 12-Plus board to ensure that it works there **before calling me over to demo the software**. If it doesn't work on the hardware board, then load the software back into the simulator and check to ensure that you initialized all of your constants/variables and memory storage addresses at the beginning of the program.

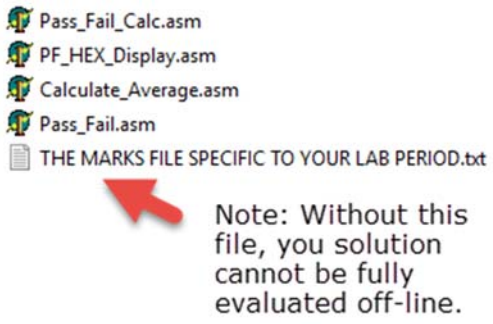
B. Submit the following files to the ***Assignment Four – Pass-Fail Calculator Link*** on Blackboard ***by 23:59 hrs on Friday, 4 Dec 2015***

Place following files depicted to the right of this text into a single compressed .zip file using the following Naming Convention for the file name:

PF_<Lab_Day_Time>_<Student_Last_Names>.zip

e.g. **PF_*Wed_1_3_Gosling_Hooper_Kernighan.zip***

All hand-in sheets will either be returned to you Week 13 in your portfolios or they will be emailed back to you with your mark and my comments.



C. Complete the following Test Plan for the set of six students' marks you have been assigned by lab period /6

Marks File Name: _____

Students	Passed Practical (Y/N)	Passed Theory (Y/N)	Passed Course (Y/N)
Student A			
Student B			
Student C			
Student D			
Student E			
Student F			

D. Post-Lab Code Inspection – your submitted code will be evaluated as follows:

I. Pass_Fail_Calc.asm / 20

II. Calculate_Average.asm / 14

III. Pass_Fail.asm / 7

IV. PF_Hex_Display.asm / 3

All code will be assessed as follows in descending priority:

a. Functionality – does the code correctly solve the problem according to the assignment instructions; wise use of iteration; have the registers been correctly used?
b. Maintainability – formatting of code; use of assembler directives, labels and CONSTANTS versus hardcoding values other than 0, 1, -1
c. Documentation: program title, header; meaningful comments that do not merely explain the instruction set
d. Optimal use of the instruction set

E. Post-Lab Program Run – I will assemble your files using the library files from their required locations to ensure that your solution works on the hardware board when independently assembled, downloaded and run.

/5

Note that the above 5 marks are intended to supplement your mark if you did not have time to demonstrate the program run by the deadlines. These marks, however, may be counted regardless of the status of your previous demonstration. So, if your original demo was correct, incorrect or not performed and the program runs correctly Post-Lab, then you will be awarded the five marks.

Your Assignment 4 – Task Two Mark /60