Assignment 2B (50 marks) – Lab Week Nine - Introductory Assembly Language Analysis and Coding

Due Dates*:*

Part A – Quiz Questions – By 11:30 PM of Friday of Week 10
Part B – Zoom: Demos and Code Inspections of Flash_PB0.asm and Counter_0-F.asm -- before end of your Lab Week 10 lab period.

Summary of Assignment Marks:

Item	Marks
Zoom: Demo and Code Inspection of Flash_PB0.asm	10
Zoom: Demo of and Code Inspection Counter_0-F.asm	15
Quiz: Assignment 2B Lab Week Nine Quiz	25
Total Marks:	**50**

Purpose of the LAB:

The purpose of this lab is a follow-on to confirm your knowledge of the past Hybrid lectures.

- Week 6 – Flowchart Lecture
- Week 7 – Addressing Modes and Iteration

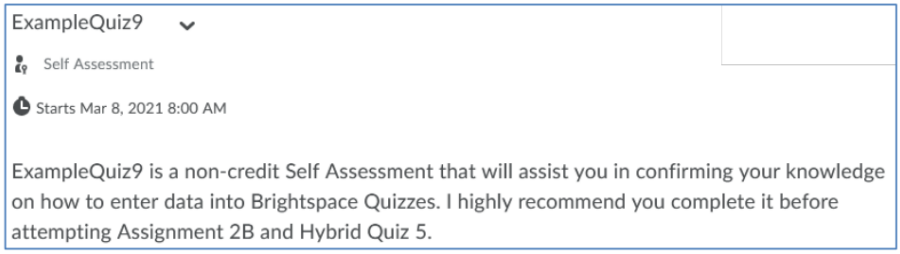
Additionally, understanding of Week 9's topics will be confirmed.

- Flowchart/Branch Statement Correlation
- Review of Iteration
- Review of Addressing Modes
- Using Arrays with AsmIDE

Part A (25 marks) – Lab Week 9 - Assignment 2B Quiz Questions – 25 Marks

Questions 1 – 10 of this part of the assignment tests your understanding of tracing through an Assembly Language program and evaluating the contents of Registers and Memory Address locations as was illustrated in the Flowcharting Hybrid Lecture
Question 11 tests your understanding of simple Assembly Language instructions as we have discussed in various lectures.
A reminder to use "$" and "%" signs in your answers in Questions 1 – 10, signifying a HEX value or a BINARY value as appropriate. Note that 16-bit HEX values take the form of "$1300" as an example, without the quotes in all cases.

In the same Brightspace module used for this assignment, I have included a non-credit Self-Assessment Quiz that will re-emphasize the use of "$" and "%" signs for the entry of required values. There is also a question that illustrates how to use Multiple Selection questions in Brightspace.
Note that the answers will indicate bracket around them e.g. [$12] – do NOT include the bracket in your answers, as the correct input would be $12.



Part B (25 marks) : Structured Programming in Assembly Language

This lab exercise provides the essence of structured programming in Assembly Language where programs' structure leads to software re-use and an easier understanding of how to solve problems using Assembly Language. Using a structured programming methodology will greatly assist you in not only the assignments in this course, but in industry as well where you will likely be tasked with developing portions of a software program that must be integrated into a larger problem solution. To that extent, you will be provided with Library files that will greatly assist you in Configuring the hardware/simulator that is used in this assignment.

Programming Task One – (10 marks) – A Simple Hardware Programming Exercise: Flash_PB0.asm

The purpose of the supplied assembly language is to flash an LED on and off every 250 ms. We will use the simulator to observe the correct program run.

a. Download the compressed Library Files package (Library_Files.zip). Create a new folder called Lib in your C:\68HCS12 folder (this was the folder where you originally installed the assembly language software). Unzip Library_Files.zip into C:\68HCS12\Lib. The figure at the right illustrates what you should have after unzipping the files. Ensure that you did not created a subfolder inside the Lib folder when you unzipped the files.

b. Download the corrupted Flash_PB0.asm into your Week Nine source directory (e.g. CST8216\Lab9) and correct the assembly language code source code listing so that the labels, opcodes, operands and comments are in the correct columns. You will also have to add the two missing lines of code that will include the two library files found in C:\68HCS12\Lib (read those files for instructions on how to use them – ensure that their "include" statements are placed just before the "end" statement in your code). Note that if you receive the following error: Fatal error -- Can't open #include file C:\68HCS12\registers.inc, then you incorrectly installed the software package and that you must de-install the software and then reinstall it into the correct path: C:\68HCS12.

c. Once you have the previous steps complete, assemble Flash_PB0.asm ensuring that there are no errors or warnings.

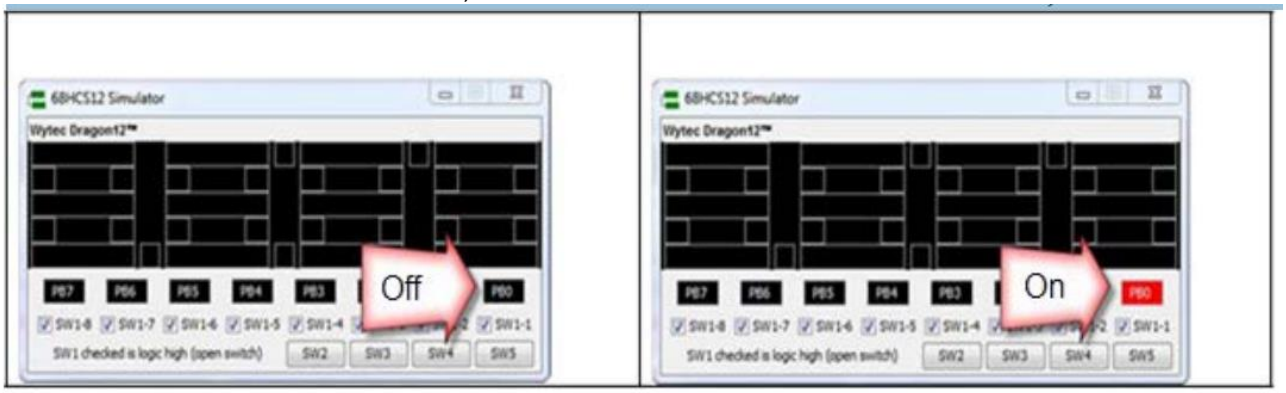
d. Load the .s19 file into the simulator and take the following action to run it:

i. Ensure "Tracing" is unchecked; otherwise, the LED will flash on and off only after several minutes of program run:

ii. On the main menu, click on View than Parallel Ports

iii. For your convenience, I have included a short video presentation on the expected behaviour of this program. Observe that the simulator and parallel ports are identical to the ones in that video. If they are not, then you are using the wrong simulator and you will have to select the correct one before proceeding any further

iv. Next, click the "Go" button on the simulator and observe that the Port B LED (PB0) on the simulator Parallel Port flashes on and off at a rate of about 250 ms, or ¼ of a second:



Your solution should very closely match it. Once the program run is correct, you will be ready to demonstrate it.
Note: Ensure that your Name, Student Number, the Modification Date is included in your code.

Your code will be assessed on both its functionality during the demo, its correct alignment of code and the inclusion of the information in the previous note.

Programming Task Two – (15 marks) – A Simple Counter: Counter_$0-$F.asm

Using a structured programming approach, write an Assembly Language program (Counter_$0-$F.asm) that implements a counter that continually counts in base sixteen from $0-$F, then starts over at $0 and then again counts to $F and so on until STOP is pressed on the Simulator. Each count will be displayed on PORT B LEDs as a binary number. Use the simulator to demonstrate your solution.

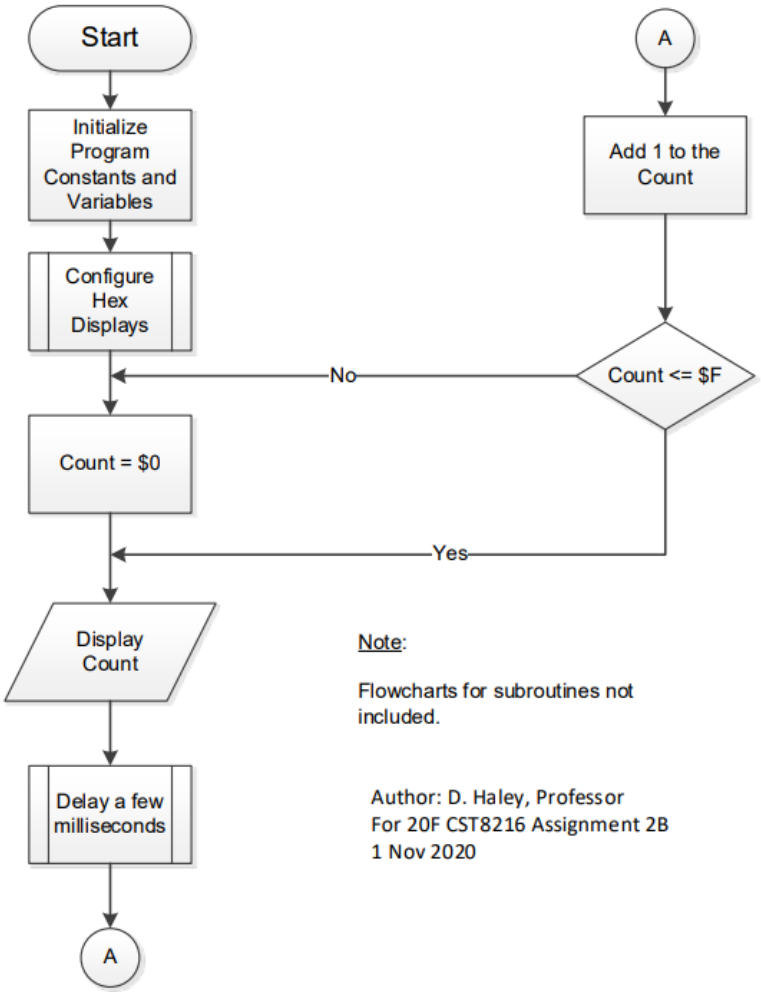
To assist you, download the skeleton code (partial code listing) for Counter_$0-$F.asm into your Week Nine source directory (e.g. CST8216\Lab9) and view the provided video: Counter_$-$F.mp4¸which provides the expected behaviour of your solution.

Then, referring to the following flowchart, implement your solution incorporating the Program Constraints into your solution.

Program Constraints:

Your solution must incorporate the following in your code:

a. Correct use of constant DELAY_VALUE to delay the indicated number of milliseconds (JSR Delay_ms) to do so.
b. Correct use of constant END_COUNT
c. Inclusion of the two Library files (same ones used in Task One's code). NO changes permitted to those files.
d. Documentation that describes what the code does, not the instruction Set (e.g. lds #$2000; stack location NOT lds #$2000 ; load the stack with address $2000
e. Full implementation of Flowchart
f. Fully functional program.



Note: Your Demonstrations and Code Inspection will take all of the above into consideration for determining your mark on the coding portion of this assignment

Rubrics:

Programming Task One – (10 marks) – A Simple Hardware Programming Exercise: Flash_PB0.asm

Criteria	Meets Expectations (2) 3 points	Below Expections (1) 1.5 points	Missing / Poor (0) 0 points	Criterion Score
Program Demo	Demo shows fully functional simulation	Demo shows that simulation is not fully functional	No Demo	/ 3

Criteria	Meets All Expectations (9) 7 points	Meets Some Expectations (6) 5 points	Below Expections (3) 3 points	Missing / Poor (0) 0 points	Criterion Score
Off-Line Testing of Code	Submitted code: • Assembles without intervention from Professor • Correct alignment of code • Inclusion of student information • Documentation that describes what the code does, • Fully functional program	Submitted code is missing one of the requirement listed in column1	Submitted code is missing two or more requirement listed in column1	Submitted code does not assemble and/or code is deemed non-functional	/ 7

Total	/ 10

Programming Task Two – (15 marks) – A Simple Counter: Counter_$0-$F.asm

Criteria	Meets Expectations (2) 3 points	Below Expections (1) 1.5 points	Missing / Poor (0) 0 points	Criterion Score
Program Demo	Demo shows fully functional simulation	Demo shows that the simulation is not fully functional	No Demo	/ 3

Criteria	Meets All Expectations (9) 12 points	Meets Some Expectations (6) 10 points	Below Expections (3) 6 points	Missing / Poor (0) 0 points	Criterion Score
Off-Line Testing of Code	Submitted code: • Assembles without intervention from Professor • Correct use of constant DELAY_VALUE to delay the indicated number of milliseconds (JSR Delay_ms) to do so. • Correct use of constant END_COUNT • Inclusion of the two Library files . • Documentation that describes what the code does, • Full implementation of Flowchart • Fully functional program • Student indormation included	Submitted code is missing one of the requirement listed in column1	Submitted code is missing two or more requirement listed in column1	Submitted code does not assemble and/or code is deemed non-functional	/ 12