Lab 1. Introduction to Cortex M Assembly (Spring 2025)

All students do Lab 1 by themselves (no partner for Lab 1)

Preparation
Purpose
System Requirements
Procedure
- Part a - Verify CCS Project for Lab1 is present and runs
- Part b - Draw Flowchart
- Part c - Write Pseudocode
- Part d - Write Assembly
- Part e - Debug
- Part f - Grade

Before checkout
Demonstration during checkout
Deliverables
FAQ

Preparation

Read Chapter 1 in textbook or ebook, and bookmark these resources

- Appendix 3 of the textbook provides a detailed description of all assembly instructions.
- Addendum 1 is used for exams and provides a short description of all assembly instructions.
- Class Website contains links to all lab assignments and their due dates.
- Arm_Architecture_v6m_Reference_Manual.pdf

Setup instructions **Steps 1-5** ECE319K CCS Installation Step By Step

Step 1: Download and install version 20.0.0 of CCS on your personal computer (MSPM0G3507)
https://www.ti.com/tool/CCSTUDIO
Choose "single file (offline)", not on-demand (web)
Does run on Windows, MacOS, or Linux
Choose **Custom install** (not Full install)
Choose **MSPM0 32-bit Cortex M0+** as your microcontroller
Watch Install Texas Instruments CCS 12 4 (old, needs rerecording)

Step 2: Download and install latest version of MSPM0-SDK — MSPM0 Software Development Kit (SDK)
https://www.ti.com/tool/MSPM0-SDK
Put it in the same folder as you installed CCS 20.0
Watch Install SDK (old, needs rerecording)

Step 3: Download and unzip the projects for ECE319K called **MSPM0_ValvanoWare**
ECE319K CCS Installation Step By Step has the link for the Spring 2025 install zip file
Create and put a **MSPM0_ValvanoWare** folder somewhere it is easy for you to find and edit

Step 4: Open CCS and import ECE319K projects into CCS

Step 5: Plug in the LaunchPad, build, debug, open a terminal window, and run **ECE319K_Lab1** project

Watch Install ECE319K Lab Projects (old, needs rerecording)

Step 6: How to open a Terminal Window

Understanding the structure of labs in this course:

Lab documents in this course will continuously become longer and contain more parts. Please read the entire document and understand the problem before attempting to write your solution. The document will contain useful hints and pseudo-code that you can use to develop your code.

All lab assignments will have a checkout associated with them. During these checkouts you will be asked questions to verify your understanding of the lab assignment and the code you wrote. Please be prepared and come on time to these checkouts.

If you are struggling with the lab assignment, each lab will have its own lab lecture associated with it. Within the lab lecture, a TA will explain what is expected and general tips on how to approach the lab. It is highly recommended to watch the recording/attend lab lecture if you have any questions about what the lab is asking of you.

Make sure J21 and J22 are configured to select XDS for the UART channel (not BP). Move jumpers on J21 and J22 so they are up (when holding the board so the silk screen is right side up), as shown in the Figure 1.0 below. Each project has a readme.html file that shows how to configure the jumpers.

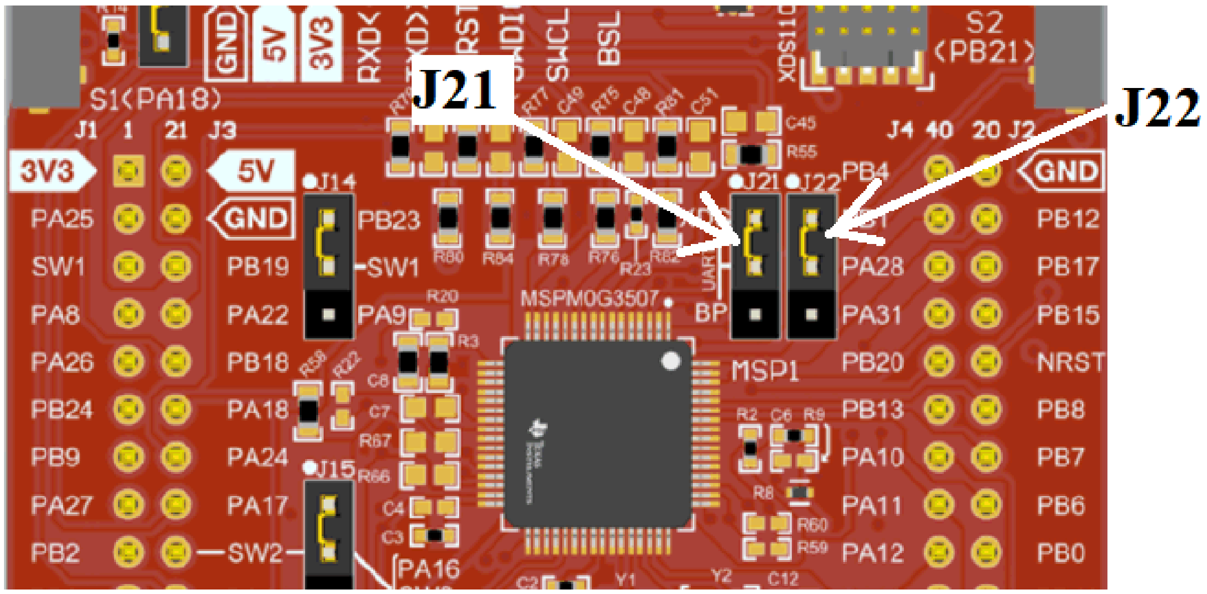


Figure 1.0. Set the J21 J22 jumps to select XDS for UART function.

Purpose

The general purpose of this laboratory is to familiarize you with the software development steps using the **CCS** IDE. In Lab 1, you will learn how to write assembly code for the Cortex M0+. The specific objectives for Lab 1 include

- Edit/build/load/run/debug cycle on CCS
- Using registers as pointers
- Accessing 32-bit and 8-bit values from memory
- Performing arithmetic and logical operations

System Requirements

The data structure defines the EIDs and Lab 1 grades for the class. Figure 1.1 shows an example with three students. Each student has an **EID** and a **Score.** The EID parameter is a pointer to a string. The strings are variable length arrays of 8-bit ASCII characters terminated with a null (0). The ASCII '0' (0x30) is different from the null (0). All EIDs have two or three upper case letters followed by 1 to 6 decimal digits. The **Score** is a 32-bit signed integer. The list of students is terminated with a null pointer (0) in the **EID** field. When your Lab 1 program is called, R0 is passed with a pointer to the list of students. Your function will search the list and return the result in R0 as specified by your specific assignment.

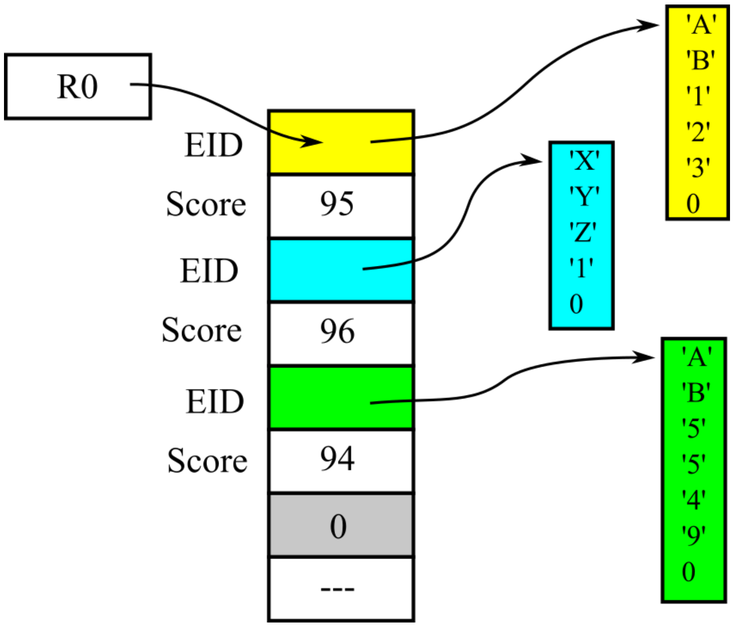


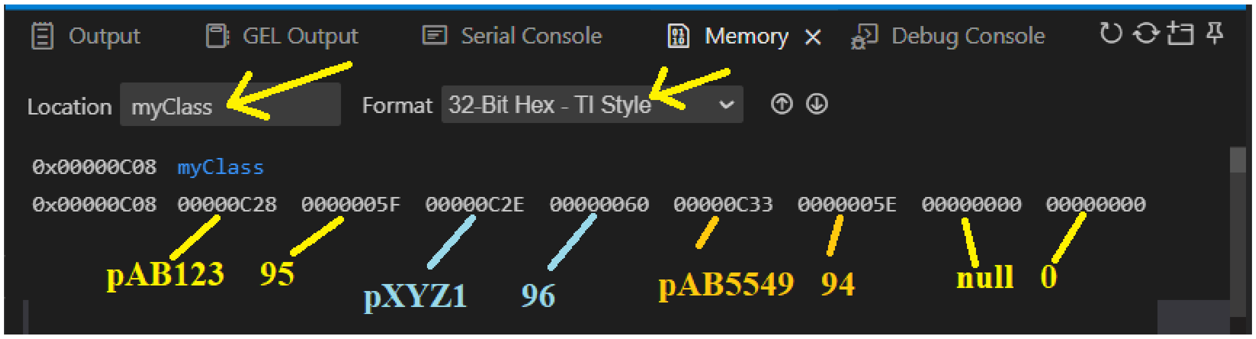
Figure 1.1. Data structure with three students.

The data structure shown in Figure 1.1 could have been created with Program 1.1.

```
         .text
         .align 2
myClass: .long pAB123  // pointer to EID
         .long 95      // Score
         .long pXYZ1   // pointer to EID
         .long 96      // Score
         .long pAB5549 // pointer to EID
         .long 94      // Score
         .long 0       // null pointer means end of list
         .long 0
pAB123:  .string "AB123"
pXYZ1:   .string "XYZ1"
pAB5549: .string "AB5549"
```

Program 1.1. Data structure with three students.

Open a memory window, execute View->Memory. In the address field, type **myClass**. Set the format to 32-bit hex. You should be able to see the list, 3 elements, null-terminated, each element as a pointer to a string and a grade.

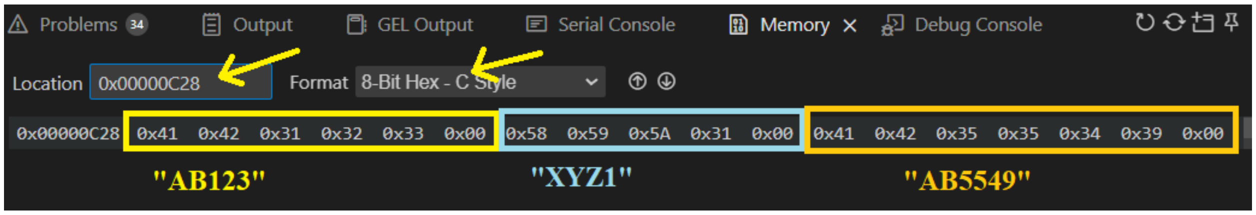


```
Location  myClass        Format  32-Bit Hex - TI Style

0x00000C08  myClass
0x00000C08  00000C28  0000005F  00000C2E  00000060  00000C33  0000005E  00000000  00000000
```

Figure 1.2. Memory browser with three students defined in ECE319K_Lab1.s.

Next, see the strings, in the address field, type the hex address of the first string, in this example it is **0x00000C28**. Your compiler might create a different place. Set the format to 8-bit hex. You should be able to see the list, 3 elements, null-terminated, each element as a pointer to a string and a grade.



```
Location  0x00000C28     Format  8-Bit Hex - C Style

0x00000C28  0x41  0x42  0x31  0x32  0x33  0x00  0x58  0x59  0x5A  0x31  0x00  0x41  0x42  0x35  0x35  0x34  0x39  0x00
```

Figure 1.3. Memory browser with three strings defined in ECE319K_Lab1.s.

To find out the exact requirements for your lab, enter your EID into the specified location in the ECE319K_Lab1.s file. After entering your EID, build, and run the system with Phase=0. View the serial terminal to see your assignment.

You will be randomly assigned to solve one of the many Lab 1 assignments based on your EID. Double check you have entered your correct EID before starting to solve the problem.

Option 1: Return R0 equal to the **Score** based on your EID, return R0 equal to -1 if your EID is not in the list. For example, assuming Figures 1.1 and 1.2, if your EID is "XYZ1", then return 96 (the score for "XYZ1" is 96). For example, if your EID is "XYZ12", then return -1 (because your EID is not in the list).

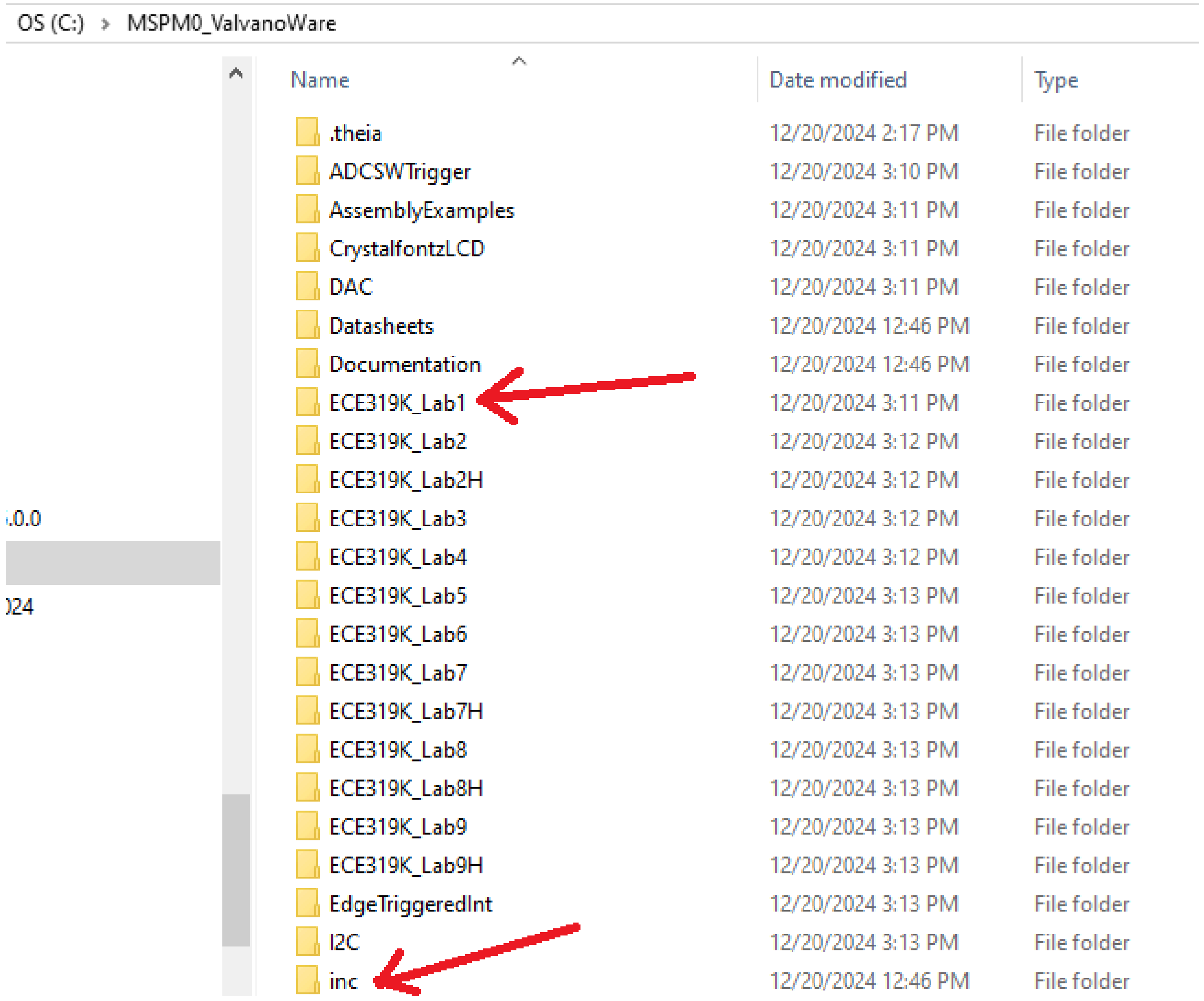
Option 2: Return R0 equal to the index value of your EID, return R0 equal to -1 if your EID is not in the list. For example, assuming Figures 1.1 and 1.2, if your EID were "AB123" you would return 0 (AB123 is index 0). For example, assuming Figures 1.1 and 1.2, if your EID were "AB4459" you would return 2 (AB4459 is index 2). For example, if your EID were "JV999" you would return -1 (your EID is not in the list).

Procedure

The basic approach to this lab will be to develop and debug your system using the MSPM0 Launchpad. Your lab assignment is automatically graded and your current grade can be seen in the terminal after running your code. There is no external hardware required for Lab 1, just run the code on the Launchpad.

Part a - Verify CCS Project for Lab1 is present and runs

To begin working on your lab assignment, perform the following tasks. Find a place on your hard drive to save all your MSPM0 software for this course. In Figure 1.4, it will be called **MSPM0_ValvanoWare**, created when you downloaded and unzipped the starter project in ECE319K CCS Installation Step By Step . Please ensure that all of your lab assignments for this course are located in the same folder as many of these assignments will attempt to access other folders present in **MSPM0_ValvanoWare**. Notice the projects you use for the labs will be folders that have the format of "ECE319K_LabX" where X will be the number of the lab.



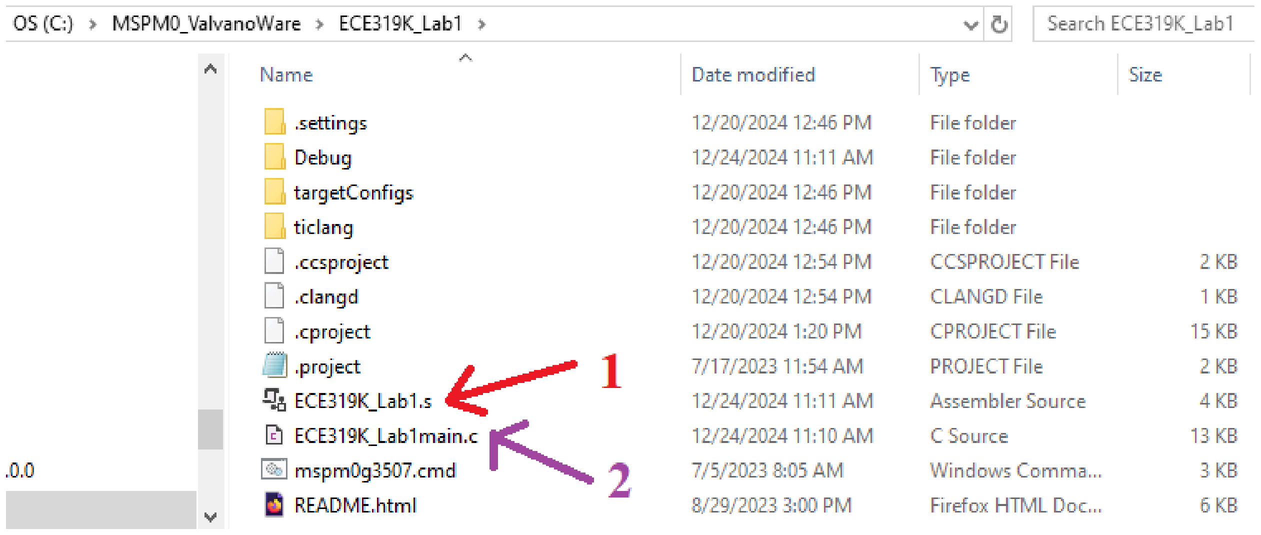
Name	Date modified	Type
.theia	12/20/2024 2:17 PM	File folder
ADCSWTrigger	12/20/2024 3:10 PM	File folder
AssemblyExamples	12/20/2024 3:11 PM	File folder
CrystalfontzLCD	12/20/2024 3:11 PM	File folder
DAC	12/20/2024 3:11 PM	File folder
Datasheets	12/20/2024 12:46 PM	File folder
Documentation	12/20/2024 12:46 PM	File folder
ECE319K_Lab1	12/20/2024 3:11 PM	File folder
ECE319K_Lab2	12/20/2024 3:12 PM	File folder
ECE319K_Lab2H	12/20/2024 3:12 PM	File folder
ECE319K_Lab3	12/20/2024 3:12 PM	File folder
ECE319K_Lab4	12/20/2024 3:12 PM	File folder
ECE319K_Lab5	12/20/2024 3:13 PM	File folder
ECE319K_Lab6	12/20/2024 3:13 PM	File folder
ECE319K_Lab7	12/20/2024 3:13 PM	File folder
ECE319K_Lab7H	12/20/2024 3:13 PM	File folder
ECE319K_Lab8	12/20/2024 3:13 PM	File folder
ECE319K_Lab8H	12/20/2024 3:13 PM	File folder
ECE319K_Lab9	12/20/2024 3:13 PM	File folder
ECE319K_Lab9H	12/20/2024 3:13 PM	File folder
EdgeTriggeredInt	12/20/2024 3:13 PM	File folder
I2C	12/20/2024 3:13 PM	File folder
inc	12/20/2024 12:46 PM	File folder

Figure 1.4. Directory structure with your Lab1.

It is important for the directory structure to look like Figure 1.4. Notice the directory relationship between the lab folders and the **inc** (include) folder. The inc folder will contain shared files used by all the projects in this class. Begin with the **ECE319K_Lab1** project.

Make sure you can compile it and run on the LaunchPad. Running the system with your EID will reveal the exact requirements for your Lab 1. Please contact your TA if the starter project does not compile or run on the LaunchPad.

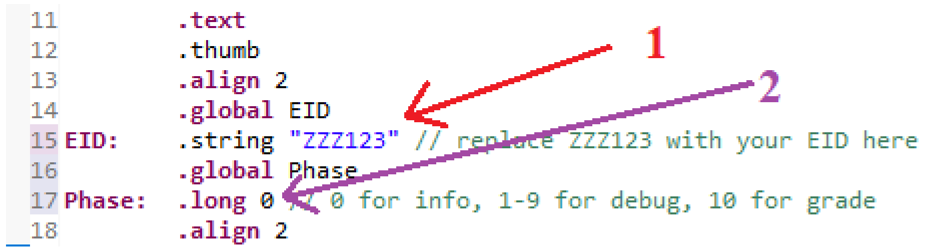
You will edit the **ECE319K_Lab1.s** file 1) red arrow). The **ECE319K_Lab1main.c** is the grader **2) purple arrow**. You should not edit the grader file. However, when the graders are provided, feel free to see how the graders work.



Name	Date modified	Type	Size
.settings	12/20/2024 12:46 PM	File folder	
Debug	12/24/2024 11:11 AM	File folder	
targetConfigs	12/20/2024 12:46 PM	File folder	
ticlang	12/20/2024 12:46 PM	File folder	
.ccsproject	12/20/2024 12:54 PM	CCSPROJECT File	2 KB
.clangd	12/20/2024 12:54 PM	CLANGD File	1 KB
.cproject	12/20/2024 1:20 PM	CPROJECT File	15 KB
.project	7/17/2023 11:54 AM	PROJECT File	2 KB
ECE319K_Lab1.s	12/24/2024 11:11 AM	Assembler Source	4 KB
ECE319K_Lab1main.c	12/24/2024 11:10 AM	C Source	13 KB
mspm0g3507.cmd	7/5/2023 8:05 AM	Windows Comma...	3 KB
README.html	8/29/2023 3:00 PM	Firefox HTML Doc...	6 KB

Figure 1.5. Start CCS and open the ECE319K_Lab1 project.

Please enter your EID into **ECE319K_Lab1.s** as shown with the red arrow in Figure 1.6. Your EID is used by a random number generator to select the assignment you need to implement. Please double check that you have entered your EID correctly. Initially, leave the **Phase** flag as 0 so the grader will show you examples of input lists and output expectations (purple arrow).



```
        .text
        .thumb
        .align 2
        .global EID
EID:    .string "ZZZ123" // replace ZZZ123 with your EID here
        .global Phase
Phase:  .long 0 // 0 for info, 1-9 for debug, 10 for grade
        .align 2
```

Figure 1.6. Put your EID into your ECE319K_Lab1.s file.

Run the system with your EID and Phase=0. Copy the text of the Terminal window and paste it into a text editor (such as Notepad). You will study the input/output expectations to fully understand your particular assignment.

Part b - Draw Flowchart

Write a flowchart for your solution. We expect 5 to 15 symbols in the flowchart. A flowchart describes the algorithm used to solve the problem and is a visual equivalent of pseudocode. See Section 1.7 in the textbook for examples of how a flowchart may be structured.

Part c - Write Pseudocode

While flowcharts are two dimensional, pseudo code is linear flowing from two to bottom. Write pseudocode for this program. We expect 5 to 15 steps in the pseudocode. You may use any syntax you wish, but the algorithm should be clear. Note, pseudocode ought to embody the algorithm and therefore be language blind. The pseudocode will become comments when developing the solution in any language.

Part d - Write Assembly

You will write assembly code for the Lab1 assembly function.

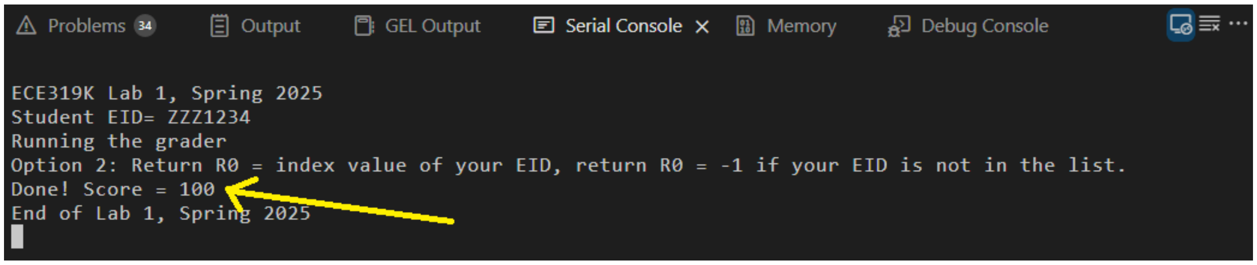
Part e - Debug

When developing your code, incrementally set the value of the **Phase** variable to test a variety of test cases with your program. Set the **Phase** flag to 1 so the grader will call your function with the first test case. Put a breakpoint in your code and single step through your solution do see whether your function calculates and returns the proper result. Please see the TAs for debugging techniques and tips that are available within CCS.

Once your function operates for the first example, change the **Phase** flag to 2 so the grader will call your function with the second test case. Repeat the debugging for Phases 3 - 7.

Part f - Grade

Once you have passed all individual test cases, run with the **Phase** set to **10**. This will test your program with the actual grader that will call your function several times repeatedly.



```
ECE319K Lab 1, Spring 2025
Student EID= ZZZ1234
Running the grader
Option 2: Return R0 = index value of your EID, return R0 = -1 if your EID is not in the list.
Done! Score = 100
End of Lab 1, Spring 2025
```

Figure 1.7. Running with Phase=10, grade results in Serial Console.

*NOTE: You can run the grader with **Phase** set to **10** as many times as you would like. Please continue developing your code until the grader returns full credit for the assignment.*

Before checkout

For all labs, please complete the following <u>**BEFORE**</u> checkout.

1. Sign up for a checkout time with your TA.
2. Upload all files you modified to Canvas, make sure your name/EID is in the comments.
3. Upload your one PDF with all deliverables to Canvas.

Demonstration during checkout

All lab assignments will follow a similar grading rubric:

- **20% Deliverables** - A collection of questions and code that will be submitted to Canvas (described below)
- **25% Performance** - The result returned by the grader. Does your code handle all situations correctly?
- **5% Code Standard Adherence** - The code you develop should be readable and well structured.

- **50% Demonstration** - The TA will ask you questions during your lab checkout about the assignment and your implementation. This also includes the visual correct execution of the lab assignment.

During your demonstration for this lab assignment, you will be asked to run your program for proper operation. You should be able to single-step your program, explain what is happening and why. You need to understand the various debugging techniques available in CCS and show the terminal output for your program.

For all labs, please have the following ready **DURING** checkout.

1. Have your one PDF with deliverables open and **ALREADY SUBMITTED PRIOR TO CHECKOUT**.
2. Have CCS with your lab assignment open, so the TA can ask you about your code.
3. Be on time to your lab checkout, being late to your checkout will result in a late penalty.
4. Demonstrate the lab to the TA.
5. Answer questions from the TA individually to determine your understanding.
6. Your score will be determined by the grading scheme above (TA will upload these to Canvas).

Deliverables

Upload your **ECE319K_Lab1.s** file to Canvas.

Combine the following components into one PDF file and upload this file to Canvas as well.

1) Your name, professor, and EID
2) Flowchart of the system
3) Pseudo-code for the algorithm
4) A screenshot of the Serial Console window, like Figure 1.7, showing your EID and score.

Optional Feedback : http://goo.gl/forms/rBsP9NTxSy

FAQ

Frequently asked questions relating to the lab assignment. This will be updated throughout the semester so please check back regularly.

1) How do I open the terminal window to see my assignment?
    a) How to open a Terminal Window
2) My MSPM0G3507 will no longer allow me to flash the device, how can I fix this?
    a) Press-and-hold the BSL_Invoke button (near LED) while pressing and releasing the Reset button.
    b) The device should go to BSL and stay in Active mode for ~10secs.
    c) Attempt to program immediately after releasing reset.

3) Errors encountered, ECE319K_Lab1.out not built?
    a) You have an error in assembly, scroll up in your build output to see the error
        i) Common errors: illegal instruction
            (1) Symbol error

4) How do I go about debugging my assembly function?
    a) Place a breakpoint at the top, run the grader, then view what you want
        i) Register view, memory browser, step, etc.

5) My makefile has an error, how do I fix it?

    a) Your makefile isn’t actually what has an error, your code failed to build. Check the build output to see the error.

6) Type ‘int32_t’ could not be resolved?
    a) Open a new workspace and try to rebuild and compile, contact TAs if that does not fix the issue
        i) Will need to reimport projects doing this, but no code will be lost

    uint32_t could not be resolved error