1. Create a new directory called HW09 under your PennSim directory.
2. Download this file [[hw09.asm](https://canvas.wisc.edu/courses/201484/files/13117441/download?wrap=1)] to your HW09 directory (right-click the link and choose "Save As" or something similar from the context menu). It will be the starting point for your programming assignment.
3. Modify the comment at the top of the **hw09.asm** file to replace "NAME" with your full name as it appears in Canvas.
4. If you are working with a partner, the submitting partner's name should be listed first, and the non-submitting partner's name should be second.

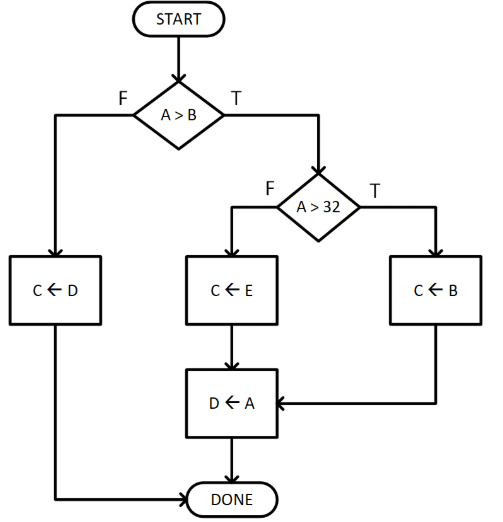
Please read the entire remainder of this assignment before starting any of the work described below! You will need to assemble and test (and probably debug) your code in PennSim, so be sure to budget enough time for this!

All code submitted for this assignment is expected to assemble, function correctly, and follow the course Programming Style Guide (available on the course website). Be sure to follow the directions below! Failing to follow directions will result in a significant penalty (possibly a zero score!).

**Assignment Description**

The purpose of this assignment is to give you practice with and assess your ability to write programs using LC-3 Control Flow instructions.

In this assignment, you will add code to the provided file to implement the below flowchart, where A, B, C, D, and E are all memory locations.  Each memory location is indicated by the comments found near the VARS array.



**Writing and Testing Your Code**

All code that you write (apart from the modifications to the header comments) must be written in the indicated location of the provided file. Do not modify any other part of the file, including the variable values in the .asm file.

You will need to add labels in your part of the code. Be sure to use useful label names that help document your code! Overly generic label names (e.g., "LABEL") will be penalized for not conforming to the Code Style Guidelines.

1. Look through the downloaded .asm file to see what is in it.
2. Insert instructions to implement the above flowchart. **Document (comment) your code as you write it.** Do not wait until the end!
   * If you need temporary storage for a variable in a register, use the below mapping. However, you do not need to put all of the variables into registers to solve this problem! You may only use R2  and R3 for intermediate value(s) you need.  **You are NOT allowed to use R4-R15.**

                             R0: Address of VARS

                             R1: Value of A

* + Remember, to update the array, you need to write to its  **memory location**. Registers only hold temporary copies of the variable values.
  + If you have trouble figuring out where to start, first implement **one** of the decision boxes, test that, and make sure you have that part of the flowchart working before you add code to implement the other decision box.
  + Remember that in Ex09, we looked at how to compare two values... if you don't remember how, review that exercise!

1. Assemble your code in PennSim. If PennSim indicates that there are any warnings or errors, go back and fix them before proceeding.
2. Load your code in PennSim. ***STEP***  through your code to verify that your program works correctly by checking whether your branches are taken or not, and compare that to what should happen based on the flowchart and the values of the A, B, C, D, and E variables given in your .asm file. If your code does not behave correctly, go back and fix it until it does.
3. Once your code is working for the variable values given in your file, you need to complete the required testing. Rather than test that your code works for all possible values of A, B, C, D, and E (216×216×216×216×216 test cases!), you'll instead test that the code works for  **all possible paths through the flowchart** (only 3 test cases, one of which you already checked!).

**Test the case where (A <= B) :**

* + - This is the default operation of the program.

**Test the case where (A > B) && (A > 32):**

* + - Edit the PC to set it back to x0200 (or reset PennSim).
    - **Reload your object file** to reset the variables to their original values.
    - In the memory window, double-click the value in the **Value** column in the row where the initial A value is loaded, and change it to be 34.  **Be sure to hit ENTER (or RETURN) so that the value actually updates!**
      * **Warning: do not modify any variable values in the .asm file; only edit the memory values in PennSim. You will need to submit a screenshot of your program running using the original provided variable values.**
    - Use the original values you were given for the other variables. **Notice that they are different from one another so that you are able to observe if one is copied to another!**
    - ***STEP***  through your code to check if its behavior matches the behavior described by the flowchart.

**Test the case where (A > B) && (A < 32):**

* + - Edit the PC to set it back to x0200 (or reset PennSim).
    - **Reload your object file** to reset the variables to their original values.
    - In the memory window, double-click the value in the **Value** column in the row where the initial A value is loaded, and change it to be 25.  **Be sure to hit ENTER (or RETURN) so that the value actually updates!**
      * **Warning: do not modify any variable values in the .asm file; only edit the memory values in PennSim. You will need to submit a screenshot of your program running using the original provided variable values.**
    - Use the original values you were given for the other variables. **Notice that they are different from one another so that you are able to observe if one is copied to another!**
    - ***STEP***  through your code to check if its behavior matches the behavior described by the flowchart.
* If you added fewer than 18 instructions to the supplied code, you have most likely overlooked something. Review the requirements, fix your code, and re-test all paths through the flowchart using the method described above to make sure it has the correct behavior.
* If you added more than 28 instructions to the supplied code:
  + Make a backup copy of your code so that you have a working solution even if you are unable to improve the efficiency.  **It is most important to submit working code.** In this course, efficiency is less critical.
  + Try to improve your code to make it more efficient (i.e., have fewer instructions), but still function correctly.
    - Remember the statement earlier in this assignment about BR and JMP instructions not changing the condition codes.
    - Review the questions on the first page of this assignment to see if you're using branch instructions inefficiently.
  + After making any changes to your code, be sure to re-test it using the same methodology as before. **Any changes to code have the potential to introduce bugs**, so don't assume that the new version works just because the previous version worked! If it does not work, fix it and re-test all the possible paths through the flowchart until you have verified they work.
  + If you are unable to get the optimized version working correctly, revert to your saved un-optimized code for the submission.

**Submitting Your Assignment**

**Please follow these directions carefully.**

1. Upload the hw09.asm file into Canvas
2. Make sure that you have the assignment open in a single tab in a single browser window (e.g., make sure you don't have this page showing in multiple browser windows).
3. Either close and reopen PennSim or reset it (using the **reset** command in the command window). **Note: the screenshot you will take must show your PennSim window at its original size, so if you re-sized it earlier, close and re-open it!**
4. Reload your working hw09.obj file.
5. Set a breakpoint at the **BR DONE** instruction provided at the end of the program (but before the program's data).
6. Press ***CONTINUE*** to execute until **BR DONE** is highlighted in yellow. Do not execute **BR DONE**!
7. **Scroll the PennSim memory window** so that memory address x0201 is shown at the top of the memory window, and your entire program and the program data are both visible.
8. Make sure that the command window is scrolled to the bottom so that its most recent text is visible.
9. Take a  **SCREENSHOT**  showing your **entire PennSim window** (it doesn't matter if it shows the rest of your desktop or not). **Your screenshot MUST be a .png or .jpg**. Do not upload a .doc or .docx file! If you do not know how to take a screenshot, follow the instructions here:  [https://kb.wisc.edu/helpdesk/search.php?q=screenshot&cat=0Links to an external site.](https://kb.wisc.edu/helpdesk/search.php?q=screenshot&cat=0)
   * **This does not mean a screenshot of your text editor. This is a screenshot of your code in  PennSim!**
10. Attach your screenshot to your submission
11. After you submit the homework, verify that your code and screenshot are still attached.

**To receive any credit for this programming assignment:**

* Your code must assemble in PennSim without errors.
* Your code must be commented and formatted according to the course Programming Style Guide.
* You must attach a screenshot of your code in PennSim that demonstrates the behavior of your code (following the directions above).

If you do not meet all of these requirements, you will receive a 0 for your program.

**If you are unable to get the program to work correctly by the deadline, but would like it evaluated for partial credit:**

* **You must still include a screenshot**at the point of execution described above that shows what your code does. This is to demonstrate that you did, in fact, attempt to run it in PennSim.
* **Add a comment**to the top of your code explaining what parts do and do not work correctly, and how you know that to be the case.