Program with password protection

Introduction:
Areas covered in this part:
ASCII Characters, I/O, DEBUG One – line Assembler, Branching, NOP operation, DEBUG commands "a" and "p", Instructions INT 10 service 0E, INT 16 Service 00, INT 21 Function 09 and 0A.

In this laboratory assignment, you will add text input and output to your program from lab 1B. Examples of using interrupt-based services are attached and will serve as a starting point.

Part 1: Password Protection Module

Develop a program that requires that you use a two-character password and then display on the screen the message, “New Balance” (or a message of your choice, like, "Welcome to <insert your name> Banking program"). The program should have the following features:

1. The password will be your personal initials (for example, the password would be “JH” for John Hancock).
2. Initially use INT 10 service 0E to echo each character of the password to the screen immediately after it is typed.
3. If anything but the correct password is entered, display the message, “Access Denied”.

Use the DEBUG command “p” (proceed) for the INT instructions (the “p” command will properly execute the instructions; recall what happens when using the "t" command with INT 20 from lab 1). Assemble sections 1, 2, and 3, entering in the program using the DEBUG assemble command “a”.

Part 2: Combining Password Protection and Calculation Modules:
Add a numeric input routine and combined programs for Part 1B and Part 2A
1. After you have been checked off for Part 1 (pre-lab flow chart and demo), remove the password “echo” to the screen by using the NOP instruction (op-code 90h). The NOP (90h) will effectively “rub out” the password characters without changing the byte count of the program. If you just delete the code instead of using NOPs you will have to recalculate your jumps.
2. Write a routine that uses a string input INT21 function 0A to input the value of the check into the debit location (memory address location 0202). Note: you will need to remove the “ASCII bias” from the inputted characters. Include an input prompt message of your choice (for example: “Welcome to JN Bank” next line “Please enter the check amount”). Use the One-line Assembler in DEBUG, command “a”.
3. Relocate your Hand Assembly lab code (this is your calculation module). The code from your Hand Assembly program (written in the previous lab) will perform the calculations. You will probably need to change your memory data addresses and the start of your calculation module program (because part 1 password protection code will need to start at IP 0100h), but you can still use the original registers. Note the combined program must start at location 0100h. You will want to use the One-line Assembler in DEBUG, command “a”. You may want to check out using a text file for saving and pasting into DEBUG (see Uffenbeck page 189 in the 3rd edition) for creating the combined program. Turn the calculation module into a sub routine and create a method by which the program will keep count how many times overdraft was used. Modify the program to print a title that includes your name, and displays program results in BCD (hint; see program 5.2 to display BCD).

Pre-Lab Work:

1. Complete flow chart for Part 1 before the lab; show your lab instructor at the beginning of the lab.
2. Complete the flow chart for Part 2 before the lab. The flow chart will show the numeric input routine and combined programs for Part 1 and Part 2, and show your lab instructor at the beginning of the lab.

Laboratory Work:

1. Demonstrate Part 1: Your updated pre-lab Flow Chart showing the machine instructions for each symbol on the chart, commented code, and be prepared to discuss your program.
2. Demonstrate Part 2: Your updated pre-lab Flow Chart showing the machine instructions for each symbol on the chart, commented code, and be prepared to discuss your program.

Lab Report Due: Week 7
Please enter current balance and then check amount