EECS 388: Embedded System
Assignment: Buffer Overflow Attack Total Points: 2

In this assignment, you will analyze a C program build for Intel 8051 architecture using Keil software and report the vulnerabilities. Refer to the lecture slides and recording on how to obtain and use the tool.

An organization needs a program that reports an SSN/secret number based on the person's date of birth and their father's date of birth. The requirement of the program is strict such that only the person associated with the date of birth can only get their SSN/secret data. Unfortunately, the library that is used to accept the user's input and compare it with the saved database has a buffer overflow vulnerability. As an attacker, you aim to collect all the possible secret data in the program by applying various input patterns in the "input_char."

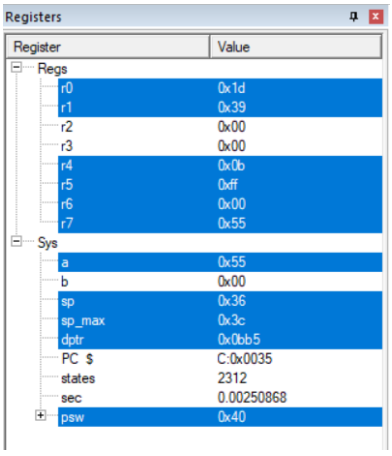
The target device for which the binary is generated is made by a vendor named "**Microchip**," and the model of the device is "**AT89C51**". The C program is attached with the homework.

Tasks:

1. Develop a malicious input based on the buffer size to leak maximum number of SSN/ secret number from the program. The attacker knows the length of the SSN/ secret number and it returns in the "r7" register during program execution. Attach the snapshot of "register" window for every leaked SSN/ secret number. For example, in the attached snapshot below we observed value "0x55" in the "r7" register. Also describe the procedure of creating malicious input step by step and finally show the malicious input.

   Points distribution:

   a) Report total characters of malicious input. For example, if the malicious input is "\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x50\x0B", there are total 13 characters. (0.5)
   b) If the malicious input can leak the first secret number 0x58 in r7 register (0.3)
   c) If the malicious input can leak the first and second secret number 0x58 & 0x57 in r7 register (0.3)
   d) If the malicious input can leak more than two secret number (0.3)



Hints:

- The location of "`return list[i].value`" might change based on the number of characters you are using in your malicious input.

- Once jump to the "`return list[i].value`" location try to understand how it's returning the data from RAM to r7 register. Try to utilize "RET" call as it will take you to the location which you just overflowed.
- Use "step" and "step over" button carefully. Always click on the "disassembly" window first then "step" through it.

2. What are the possible solutions for overcoming the buffer overflow issue in the prior code? Modify the C program according to your solution and describe how it will prevent the buffer overflow. (0.6 points)

Submit your answer as typed using an MS Word or PDF file.