CO224 – Project I – ARM Assembly Programming: Speck Algorithm

Worth = 10%

Deadline = 11.55pm, 24 July 2016 [late submissions will attract 20% penalty per day they are late]

This is an individual project.

Description

Speck is a recently introduced encryption algorithm. Specialty is that, it only uses three simple operations namely add, xor and round. As the computational and memory requirements of this algorithm is really low when compared to algorithms like AES, this is very suitable for low end processors. The whole algorithm itself is less than 20 lines of C code, but yet it is said to provide a great level of security. More information about the Speck cipher can be found at https://en.wikipedia.org/wiki/Speck_(cipher).

Your task is to implement a variant of **Speck with 128-bit block size and key (C code is found in the Wiki page)** using ARM Assembly. You **should not use any library functions** except those for input/output (available in stdio.h).

Your program should behave like the following:

```
Enter the key:
0f0e0d0c0b0a0908 0706050403020100
Enter the plain text:
6c61766975716520 7469206564616d20
Cipher text is:
a65d985179783265 7860fedf5c570d18
```

All the inputs and outputs are in hexadecimal. 128-bit version of Speck uses 64-bit unsigned integers for calculation and hence observe that each input and output above has 2 separate numbers in hexadecimal.

Testing and Deliverables

By the deadline, you should submit the assembly source code in a **single file (.s file)** to FEeLS. Check the given sample input with the sample output, as follows. Make sure it works according to the specification as we are **auto marking**.

qemu-arm -L /usr/arm-linux-gnueabi ./speck < sampleinput.txt > myoutput.txt
diff sampleoutput.txt myoutput.txt

Further, we have provided you the binary file compiled for Intel x86_64 if you need to generate more test cases.

Hints

- Break down the algorithms into functions, implement each function and test each function separately.
- You don't have to implement the C code in the wiki page as is. Do whatever changes to it to make your life easier. How you implement or code is not a problem as long as you get the correct output.

- ARM has 32 bit registers but you are supposed to do operations on 64 bit unsigned integers. This is challenging but quite easy if you think carefully. Think how you can use instructions such as adds, adc, lsl, rls and orr from the ARM ISA.

Important

- This is a project and hence things are not straight forward. You will have to do some self-learning plus thinking as things are like that in real world.
- Assembly code debugging would take time and hence start early.
- Ask questions or doubts in the **Forums in FEeLS**. Don't ask them on Facebook chat or any social media as I won't answer.
- Don't try to become smart by using gcc to convert from C to assembly or using obj-dump to convert the binary to assembly. We know how to catch them and you will get 0 marks.
- Don't copy from anyone as if caught everyone involved would get 0 marks.

Happy coding!