School of Computing: assessment brief

Module title	Computer Processors
Module code	XJCO1212
Assignment title	**Assignment 2: Encryption using a Feistel Cipher**
Assignment type and description	In-course assessment. Requires design, implementation and testing of code written in assembly language
Rationale	Provides an opportunity to write assembly code including understanding the implementation of branching and functions and learn how a Feistel Cipher works for encryption.
Word limit and guidance	This coursework should take less than 15 hours to complete.
Weighting	60%
Submission deadline	**2/5/2024 10am UK time**
Submission method	Gradescope
Feedback provision	Feedback will be provided through Gradescope
Learning outcomes assessed	Explain how high level programming constructs, such as 'if' statements and 'for' loops, are implemented at a machine level
Module lead	Samson Fabiyi
Other Staff contact	Heng Lui

1. **Assignment guidance**

   The Feistel cipher is a symmetric block cipher encryption framework which is the basis of many modern day encryption algorithms. In this coursework you will implement a Feistel cipher system as a software implementation in Hack Assembly.

   In a Feistel cipher the plaintext, P, to be encrypted is split into two equal size parts L_0 and R_0 such that $P = L_0R_0$. A function F is applied to one half of the plaintext, combined with a key, and the result is XOR'd with the other half of the plaintext. Feistel ciphers often employ multiple rounds of this scheme. In general the scheme works as follows, for all $i = 0, \ldots, n$,

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

   To decrypt an encrypted message using this cipher we can apply the same procedure in reverse. For $i = n, n-1, \ldots, 0$,

$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

   For this coursework we are interested in the 16-bit Feistel cipher which uses 4 rounds. The function $F(A, B) = A \oplus \neg B$.

   The keys are derived from a single 8-bit key K_0 such that,

$$K_0 = b_7b_6b_5b_4b_3b_2b_1b_0$$
$$K_1 = b_6b_5b_4b_3b_2b_1b_0b_7$$
$$K_2 = b_5b_4b_3b_2b_1b_0b_7b_6$$
$$K_3 = b_4b_3b_2b_1b_0b_7b_6b_5$$

2. **Assessment tasks**

   (a) Write a program (XOR.asm) in HACK assembly that implements a bit-wise XOR function between two 16-bit values stored in RAM[3] and RAM[4] and stores the result in RAM[5].

   [4 marks]

(b) Write a program (Rotate.asm) in HACK assembly that implements an algorithm to rotate the bits of a 16-bit number left (Least Significant bit (LSb) to Most Significant bit (MSb)). The original number should be stored in RAM[3], the number of times to rotate the bits should be in RAM[4] and the result stored in RAM[5], i.e. 1010111100000000 rotated left 3 times would be 0111100000000101 where the MSb is used to replace the LSb on each rotation.

[8 marks]

(c) Write a program (FeistelEncryption.asm) in HACK assembly, that implements the described Feistel encryption system. The initial key, K_0, will be stored in RAM[1], and the plaintext to be encrypted will be represented by a 16-bit value stored in RAM[2]. The result of the encryption should be stored in RAM[0].

[10 marks]

[Total 22 marks]

3. **General guidance and study support**

Tools required to simulate the hardware and CPU are provided on Minerva under Learning resources: Software. You may find it easier to implement cipher in a high level language first. This will also allow you to test the results of your HACK program. Support will be available during lab classes. Please ensure the files you upload work with the test files provided and use the filenames provided in this sheet. **Do not alter the format of the lines of these test files in any way. The spacing in each line needs to be preserved** You are of course welcome to build your own test files in the same format or add to these files.

4. **Assessment criteria and marking process**

This coursework will be automatically marked using Gradescope. Feedback will be provided through Gradescope.

Marks are awarded for passing the automated tests on the submitted programs. These will not necessarily be the same tests that are provided to help you develop the solution. You should therefore test your solution thoroughly using other values for the plaintext and keys before your final submission.

5. **Presentation and referencing**

Submitted code should provide suitable comments where possible.

6. **Submission requirements**

Links to submit your work can be found on Minerva under Assessment and feedback/Submit my work. The HACK assembly (asm) files for each part must be uploaded individually. Ensure you use only the filenames provided in this specification sheet.

7. **Academic misconduct and plagiarism**

   Academic integrity means engaging in good academic practice. This involves essential academic skills, such as keeping track of where you find ideas and information and referencing these accurately in your work.

   By submitting this assignment you are confirming that the work is a true expression of your own work and ideas and that you have given credit to others where their work has contributed to yours.

8. **Assessment/marking criteria**

   No marks will be awarded for tests which fail

   - Part a) is graded using 4 tests, each worth 1 mark. [max 4 marks]
   - Part b) is graded using 4 tests, each worth 2 marks. [max 8 marks]
   - Part c) is graded using 4 tests, each worth 2 marks and a further 2 marks for optimised solutions that require a lower number of operations to complete the encryption [max 10 marks]

   [Total 22 marks]