School of Computing: assessment brief

Module title	Computer Processors
Module code	COMP1212
Assignment title	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. Also provides the opportunity to understand 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	12/5/23
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	Andy Bulpitt
Other Staff contact	Samson Fabiyi

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 both a high level language and 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 $i = 0, \dots, n$,

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

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

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

   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,

$$\begin{aligned} K_0 &= b_7b_6b_5b_4b_3b_2b_1b_0 \\ K_1 &= b_6b_5b_4b_3b_2b_1b_0b_7 \\ K_2 &= b_5b_4b_3b_2b_1b_0b_7b_6 \\ K_3 &= b_4b_3b_2b_1b_0b_7b_6b_5 \end{aligned}$$

2. **Assessment tasks**

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

(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. 10101111 rotated left 3 times would be 01111101 where the MSb is used to replace the LSb.

(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 16-bit plaintext will be stored in RAM[2]. The result of the encryption should be stored in RAM[0].

3. **General guidance and study support**

Tools required to simulate the 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 all programs work with the test files provided and use the filenames provided in this sheet. **Do not alter the format of test files in any way.** Ensure the files you upload pass the submission tests provided on Gradescope. These are **not** necessarily the same tests as those that will be used to grade your submission.

4. **Assessment criteria and marking process**

This coursework will be automatically marked using Gradescope. Feedback will be provided through Gradescope and example solutions discussed in class.

Marks are awarded for passing the automated tests on the submitted programs.

5. **Presentation and referencing**

Submitted code should provide suitable comments where possible.

6. **Submission requirements**

You **must** submit your work via Gradescope as a single zip file. Ensure you use only the filenames provided in this 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 grid**

   - Part a) is graded using 3 tests, each worth 2 marks. [max 6 marks]
   - Part b) is graded using 3 tests, each worth 2 marks. [max 6 marks]
   - Part c) is graded using 4 tests, each worth 2 marks. [max 8 marks]

   [Total 20 marks]