Computer Systems COMP1071

2019/2020

LMC programming assignment

Submit your work on **DUO** before **08 November 2019, 14:00**. For any questions, contact the setter of the assignment Dr Ioannis Ivrissimtzis: ioannis.ivrissimtzis@durham.ac.uk

What to submit

You should submit assembly code. You can save a program as assembly code by selecting **Save** from the LMC Assembly Editor window of the LMC simulator.

Submit your programs as two separate plaintext files (.txt), with comments to indicate how it works. The filenames of the two program files should be:

basicCOMP1071.txt

advancedCOMP1071.txt

I should be able to open the files with the LMC Assembly Editor window and they should compile and run without any alteration.

Description of the basic task (80 marks / 100)

Given an integer f_0, consider the sequence described iteratively by

$$f_{i+1} = \begin{cases} f_i \,/\, 2 & f_i \ \ even \\ 3 * f_i + 1 & f_i \ \ odd \end{cases}$$

Create an LMC program with the following specifications:

i. The program accepts as input a positive integer, $f_0 > 0$.

ii. The program outputs the elements $f_0,\ f_1,\ f_2, \dots$ of the above sequence.

iii. If $f_i = 1$ for some i, the program terminates.

iv. If $f_i > 999$ for some i, the program outputs 0 and terminates.

In plain English, we construct the sequence by dividing even numbers by 2, and by multiplying odd numbers by 3 and then adding 1. We stop when we reach 1, or a number that is out of the bounds of the LMC calculator.

Some examples of inputs with the corresponding outputs:

1	1
6	6, 3, 10, 5, 16, 8, 4, 2, 1
25	25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
341	341, 0

Description of the advanced task (20 marks / 100)

Given an integer g_0, consider the sequence described iteratively by

$$g_{i+1} = \begin{cases} g_i \,/\, 2 & g_i \ \ even \\ (3 * g_i + 1)/2 & g_i \ \ odd \end{cases}$$

Create an LMC program with the following specifications:

i. The program accepts as input a positive integer, $g_0 > 0$.

ii. The program outputs the elements $g_0, g_1,\ g_2, \ldots$ of the above sequence.

iii. If $g_i = 1$ for some i, the program terminates.

iv. If $g_i > 999$ for some i, the program outputs 0 and terminates.

The sequence g can be obtained from f by deleting some of its elements. Some examples of inputs with the corresponding outputs:

1	1
6	6, 3, 5, 8, 4, 2, 1
25	25, 38, 19, 29, 44, 22, 11, 17, 26, 13, 20, 10, 5, 8, 4, 2, 1
341	341, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1

Discussion

The sequence f (or the sequence g) gives rise to an interesting mathematical question. Starting with any positive integer $f_0 > 0$, and assuming that we can handle arbitrarily large numbers, do we always reach 1?

This open problem is considered a prime example of a mathematical question witch is very easy to state and understand, but very difficult to answer.

The link below is to an article in New Scientist describing a recent development in this problem:

https://institutions.newscientist.com/article/2216259-baffling-maths-riddle-that-looks-like-a-pile-of-worms-almost-solved/

Marking scheme

Basic problem		
Test 1	On any input $2 < f_0 < 100$, the correct values of f_0 and f_1 are outputted.	**25**
Test 2	On any input $2 < f_0 < 100$, and for as long as the termination criteria have not been met, correct values are outputted.	**10**
Test 3	On any input $2 < f_0 < 100$, the program terminates correctly.	**20**
Test 4	The program handles appropriately all possible inputs.	**5**
Efficiency	Efficiency marks will be available only to programs that give correct answers to all tests. They will be awarded based upon the efficiency of the code, i.e. how many mailboxes are used (the fewer the better) and how many fetch-execute cycles are used for each calculation (again the fewer the better).	**20**

Advanced problem		
Test 1	On any input $2 < g_0 < 100$, and for as long as the termination criteria have not been met, correct values are outputted.	**10**
Test 2	The program handles appropriately all possible inputs, and terminates correctly.	**5**
Efficiency	Efficiency marks will be available only to programs that give correct answers to both tests. They will be awarded based upon the efficiency of the code, i.e. how many mailboxes are used (the fewer the better) and how many fetch-execute cycles are used for each calculation (again the fewer the better).	**5**