The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

COMPUTER SYSTEM PROGRAMMING
ECE454 – FALL 2023
ASSIGNMENT-2

Dues are defined at the end of this guideline.

Objectives

- Reviewing/Working with the source code optimization techniques (Loop Invariant Code Motion-LICM, Reducing unnecessary Statements and Memory Access, Loop Unrolling, Code Inclining, Minimizing the impact of Pointers, etc)
- Reviewing/Working with the System Functions for Time in the source code, measuring the execution times

2 Applying Optimization on Source Code

In this Lab, you will use *"A2-code.c"*, which contains various parts, and will apply different Program Optimization techniques on the source code (different parts) to analyse the code execution times.
After applying each optimization technique, changing the source code and finding the execution time (before and after optimization), record all the execution times in a proper table for further comparisons.

2.1 Loop Invariant Code Motion

In this section, you will use *"A2-code.c"*, partial code-1, processing vectors.

1- Use the function clock( ) before and after the original code-1 to find the execution time of this process
2- Change the code-1 using "Loop Invariant Code Motion (LICM) technique"
Note – At this step only change the invariant part of the loop
3- Use the function clock( ) again before and after the changed code-1 to find the execution time of its process
4- Repeat step3for 5 times, every time with a new compilation and execution process from the beginning, finding the average of various execution times for code-1 after change; enter all times in the result table
[4 mark]

Note – Feel free to increase the Vector size in the code (from step1), if this produces more clear execution time.

2.2 Reducing unnecessary Statements and Memory Access

In this section, you will use *"A2-code.c"*, partial code-2, processing vectors.

1- Use the function clock( ) before and after the original code-2 to find the execution time of this process
2- Change the code-2 using "Reducing unnecessary Statements and Memory Access technique"
Note - At this step only change the access to Vector elements; note that each access by V[index] syntax is equal to several instructions to execute
3- Use the function clock( ) again before and after the changed code-2 to find the execution time of its process
4- Repeat step3for 5 times, every time with a new compilation and execution, finding the average of execution time for code-2 after change; enter all times in the result table
[4 mark]

Note – Feel free to increase the Vector size in the code (from step1), if this produces more clear execution time.

2.3 Loop Unrolling

In this section, you will use "*A2-code.c*", partial code-2, processing vectors.

1- Use the function clock( ) before and after the original code-2 to find the execution time of this process
2- Change the code-2 using "Loop Unrolling"
   Note - At this step only change the loop statements preferably into two versions: accessing to 3 and 4 elements of the Vector in each loop iteration
3- Use the function clock( ) again before and after the changed code-2 (for both versions) to find the execution time of its process
4- Repeat step3for 5 times (for both versions), every time with a new compilation and execution, finding the average of execution time for code-2 after change; enter all times in the result table
   [4 mark]

Note – Feel free to increase the Vector size in the code (from step1), if this produces more clear execution time.

2.4 Code Inlining

In this section, you will use "*A2-code.c*", partial code-3, processing vectors.

1- Use the function clock( ) before and after the original code-3 to find the execution time of this process
2- Change the code-3 using "Code Inlining"
   Note - At this step only change the access to function *func1( )* result; each access to function syntax is equal to several instructions to execute
3- Use the function *clock( )* again before and after the changed code-3 to find the execution time of its process
4- Repeat step3for 5 times, every time with a new compilation and execution, finding the average of execution time for code-3 after change; enter all times in the result table
   [4 mark]

Note – Feel free to increase/decrease the Vector size in the code (from step1), if this produces more clear execution time.

Deliverables & Submission for Assignment-1 [Total 20 Marks = weight 5%]

-The teams should submit their final report through the submission folder on Quercus by:
Final Report due on Thursday Oct5, at 8:59am [4 Marks]
-In addition to the Final Report, A2 will be evaluated by the TAs through Q/A in the following Labs:
Thursday Oct5 (final work evaluation) [16 Marks]
-Make sure to add the names and student numbers of your Team members, and the Lab section your team attends at the top part of your final report. Only ONE final submission is required from each team.
-The teams should save all their answers in a final report (plain text format) under this naming rule:
"PRAxx_Tyy_Az.txt" where "xx" is the Lab section (01 for 12-3pm, and 02 for 9am-12pm), "yy is the Team number and "z" is the Assignment number, e.g. "PRA01_T06_A2.txt" is the name for submission from Team06 for A2 working in the Lab on Thursday 12-3pm.

Good Luck.

Useful Resources:

-All C programming language references