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

COMPUTER SYSTEM PROGRAMMING
ECE454 – FALL 2023
ASSIGNMENT-1(V2)

Dues are defined at the end of this guideline.

Objectives

- Reviewing/Working with the Compilation Flags (-g, -O0, -O1, -O2, -O3, ...)
- Reviewing/Working with the Profiling Tools (time, gprof, gcov)
- Reviewing/Working with the Inspection Tool (objdump)

NOTE- This document is version2 of A1 with the same total marks. You can submit either solution for A1(v1) or A1(v2).

Update Note about Profiling (A1)

Profiling with gprof: Use the provided file *gprof_test.c* and the following commands in order to:
1) compile it with –gp option (enable profiling), 2) execute it to generate *gmon.out* and then 3) execute it with *gmon.out* as parameter to generate *analysis.txt* file that contains the required profiling information (flat profile and call graph, you may want to use –a or –b to suppress the information):

```
$ ls
gprof_test.c

$ gcc -Wall -pg gprof_test.c -o gprof_test

$ ls
gprof_test  gprof_test.c

$ ./gprof_test
...

$ ls
gmon.out gprof_test  gprof_test.c

$ gprof gprof_test gmon.out > analysis.txt
$ gprof –a gprof_test gmon.out > analysis.txt
$ gprof –b gprof_test gmon.out > analysis.txt
```

Profiling with gcov: Use the provided file *gcov_test1.c* and the following commands in order to: 1) Instrument the code with the test information using -ftest-coverage and -fprofile-arcs parameters and 2) run it to see the result

```
$ ls
gcov_test1.c

$ gcc -Wall    -fprofile-arcs     -ftest-coverage    gcov_test1.c

$ ls
a.out gcov_test1.c      gcov_test1.gcno

$ ./a.out
...
$ gcov gcov_test1.c
...
```

Update Note about Questions (A1)

1.1 Profiling

Q1-The previous question has been changed to the following:
Using the Extension-Note about Profiling, go through the instructions, create profiling information using gprof and gcov, and finally display/discuss about the results. (describe the generated report for each prof and gcov options briefly). [3 mark]

From Q2 to Q14, measure the compilation time using the gcc command with these compilation flags at the command line: 1) -g, 2) -O2, 3) -O3, 4) –Os, then answer the following questions based on the results:

1.2 Measuring Compilation Time

Find the compilation-time with all four flags used in the previous section:
Q2-Report the 4 compilation time measurements. [1 mark]
Q3-Which compilation time is the slowest and why (describe briefly)? [1 mark]
Q4-Which compilation time is the fastest and why (describe briefly)? [1 mark]
Q5-Based on the results in Q1, which of gprof and gcov is faster and why (describe briefly)? [1 mark]

1.3 Measuring Program Size

Use "ls -l" to measure the size of all four versions of binary files generated in the previous section:
Q6- Report the 4 size measurements for the generated binary files. [1 mark]
Q7-Which size is the smallest and why (describe briefly)? [1 mark]
Q8-Which size is the largest and why (describe briefly)? [1 mark]
Q9-Based on the results in Q1, which of gprof and gcov generate smaller binary file and why (describe briefly)? [1 mark]

1.4 Measuring Performance

Find the run-time with all four flags used in the previous section:
Q10-Report the 4 execution time measurements. [1 mark]
Q11-Which version is the slowest and why (describe briefly)? [1 mark]
Q12-Which version is the fastest and why (describe briefly)? [1 mark]
Q13-Based on the results in Q1, which of grof and gcov generates faster binary file and why (describe briefly)? [1 mark]

1.5 Inspect Assembly

Run the object files created by the compilation flags –g and –O3 to list their assembly instructions for main.c using this command:

"objdump -d OBJ/main.o"

Q14- Report the results created by each flag and compare them with each other based on the number of instructions (describe briefly)? [1 mark]

(Optional Question)-Name the shortest GCC compiler flag (i.e, -xx) to enable a compiler optimization that requires memory alignment. How many bytes does the data need to be aligned? [1 bonus Mark]

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

-The teams should submit their final report through the submission folder on Quercus by:
Thursday Oc5, at 8:59am (final Report) [3 Marks]
-In addition to the Final Report, A1will be evaluated by the TAs through Q/A in the following Labs:
Thursday Oct5 (final work evaluation) [16 Marks]

Notes:
-Sample Makefiles were already posted in the folder of A1-code, but using a Makefile is not mandatory.
-Extra C programs (test-1.c and test-2.c are added in the folder of A1-code. Feel free to use them with various inputs.
-If you already attended the Lab session on Sep21, your mark for Q1 in A1(v1) will be considered for this A1(v2).

-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 or word format) under this naming rule:
"PRAxx_Tyy_Az" 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_A1" is the name for submission from Team06 for A1 working in the Lab on Thursday 12-3pm.

Good Luck.

Useful Resources:

GCC Man Page:

https://linux.die.net/man/
https://linux.die.net/man/1/gcc
https://man7.org/linux/man-pages/man1/gcc.1.html

Gprof:

https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_mono/gprof.html
https://www.tutorialspoint.com/unix_commands/gprof.htm

Gcov:

https://gcc.gnu.org/onlinedocs/gcc-11.1.0/gcc/Invoking-Gcov.html
https://github.com/gcc-mirror/gcc/blob/releases/gcc-11.1.0/gcc/gcov.c