Debugging Using GDB and Valgrind

Learning Objectives

The learning objectives in this tutorial are:

- To study and debug programs line-by-line and/or instruction-by-instruction
- To simulate programs' interaction with the machine's memory and identify bad and good memory access patterns.

Tools

- GDB
- Valgrind

Contents

Debugging with GDB 1: Tutorial (20 min) 1

Assessing memory access patterns with Valgrind 2: Tutorial (20 min) 2

Optional: Assembly Instructions 3: Tutorial (5 min) 2

In this tutorial we use GNU debugger (gdb) and Valgrind to study the state of two given programs during execution. The programs are simple matrix initialization codes written in C. The first program, matrix_init_rw.c, initializes each row, and the second, matrix_init_cn.c initializes each column. Due to the differences in the initializations, the programs will use the memory differently resulting to different run-times.

Debugging with GDB 1: Tutorial (20 min)

Use GDB to view the assembly code of the two given programs and print some of the registers used.

Steps

1. Compile the two source codes using gcc with -g option
2. Use GDB's help command to find and use different Text User Interface (TUI) layouts
3. Change the default GDB's assembly syntax to Intel
4. Introduce breakpoints at various positions and run the programs
5. Print contents of general purpose registers.

Assessing memory access patterns with Valgrind 2: Tutorial (20 min)

Analyse the memory patterns when the two given programs are executed. Check whether cache is used effectively by the programs.

Steps

1. Compile the two source codes using gcc with -g option
2. Run and time their executions using shell built-in `time` utility
3. Assess and compare cache misses in each program using Valgrind
4. Write results to output files "cache_misses_1.txt" and "cache_misses_2.txt"

Optional: Assembly Instructions 3: Tutorial (5 min)

Using Intel Architecture Software Developer's Manual, find out the meaning of different instructions and registers shown in tutorial 1 above.

Hints

- Create `$HOME/.gdbinit` file and add the following line, `set disassembly-flavor intel`
- Use Valgrind's `Cachegrind` tool to simulate the program's interaction with the machine's memory hierarchy.
- Write output files using the `--cachegrind-out-file` option.

Further Reading

- I. Zhirkov (2017) Low-Level Programming, DOI 10.1007/978-1-4842-2403-8_11
- Intel Architecture Software Developer's Manual - https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
- Valgrind User Manual - https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
- GDB User Manual - https://sourceware.org/gdb/current/onlinedocs/gdb/