## Filip Jeremic


**Contact Information**

filip@jeremic.ca

http://jeremic.ca
https://github.com/fjeremic


**Interests**

Compiler development (static and dynamic), programming languages (design and implementation), parallelization, computer graphics, software protection, reverse engineering, malware analysis, and cryptography.

**Technical Skills**

**C, C++, x86-64 Assembly, z/Architecture Assembly, C#, Java**
*Expert, 10+ years*

These technologies are the ones which I have used the most over the years. Because of my field of interest, low level programming languages such as various flavors of assembly, C, and C++ have been my primary languages of choice. Working on a just-in-time (JIT) compiler for Java has made me deeply familiar with these languages both from the angle of software engineering and performance optimization.

**Linux, Unix tools, Windows, z/OS**
*Advanced, 8+ years*

My primary development platforms is Linux. Most of my current work involves development on non-desktop architectures so working on a remote machine via SSH is second nature to me. I am very comfortable within a Unix environment carrying out tasks such as instruction level performance investigations (perf), assembly level debugging (gdb, IDA, Ollydgb, etc.), and remote development using the technologies listed above.

**Docker, Jenkins, CMake, Bash, Qt, Python, JavaScript, Perl, HTML, WPF, UWP**
*Experienced, 4+ years*

These technologies I have used in various ways to aid in my work, however they are not my primary tools and I do not consider myself an expert in any of them. However, I feel that I am proficient in all of them and often use them to my advantage when they fit the task at hand.

**Professional Experience**

**IBM**, Toronto, Canada

*Compiler development - Advisory Software Engineer* **2017 - Present**

A natural progression of my previous position into a team leadership role within the backend area of the JIT compiler technology I had been working on for several years. During this exciting time we worked on open sourcing our compiler technology as part of the Eclipse OpenJ9 and Eclipse OMR projects at GitHub. As the open source communities grew, my role has been evolving into being one of the focal points for cross-platform backend JIT development and community management within the realms of my expertise. I am currently the top code contributor across both projects as per the GitHub statistics and am heavily involved in reviewing others' contributions toboth projects.

*Compiler development - Staff Software Engineer* **2015 - 2017**

An extension of my previous role with a broader focus primarily on performance acceleration of Java workloads on Linux and z/OS. Among development items this role involved instruction level profiling of Java applications and identifying bottlenecks which can be optimized in the JIT compiler. The performance investigation is done at an instruction level but common intermediate language (IL) level optimizations were developed as part of the experimentation such that all supported backends (x86, Power, z/Architecture, ARM, AArch64, RISC-V) can benefit.

Part of my role also involved sharing the burden of some of the team leads responsibilities in technical management of work items triaged to the rest of the development team, and the delivery of performance improvement targets for the release of Java 9.

*Compiler development - Associate Software Engineer* **2013 - 2015**

Directly after graduation I took a position at IBM Canada's compiler group where I worked on the backend of the just-in-time (JIT) compiler for the IBM J9 Java Virtual Machine (JVM) in support of IBM platforms.

- Worked on a team developing the backend code generator for z/Architecture based processors
  - Designed and developed various compiler features ranging from intrinsics libraries, register allocation, instruction scheduling, and optimal instruction selection for the target processor
  - Carried out instruction level performance investigations on various workloads and implemented changes to the backend code generator to realize the performance gains
  - Postmortem core dump analysis of non-deterministic code generator bugs
- Developed a firm understanding the interaction between the various components of a dynamic runtime environment (VM, GC, JIT), particularly in the context of a JVM

## Patents

- **Multi-byte compressed string representation**

  Patent No. US10002010B2 **Issued 2018-06-19**

  Multi-byte compressed string representation embodiments define a String class control field identifying compression as enabled/disabled, and another control field, identifying a decompressed string created when compression enabled. Tests are noped based on null setting of the compression flag. When arguments to a String class constructor are not compressible, a decompressed String is created and stringCompressionFlag initialized. Endian-aware helper methods for reading/writing byte and character values are defined. Enhanced String class constructors, when characters are not compressible, create a decompressed String, and initialize stringCompressionFlag triggering class load assumptions, overwriting all nopable patch points. A String object sign bit is set to one for decompressed strings when compression enabled, and masking/testing this flag bit is noped. Alternative package protected string constructors and operations are provided. A predetermined location is checked to determine whether supplied arguments to a String class constructor are compressible is performed.

- **Object load introspection using guarded storage**

  Patent No. US20200218651A1 **Issued 2020-07-09**

  Systems and methods for object load introspection using guarded storage are disclosed. In embodiments, a computer- implemented method includes: determining objects of interest designated by a user; splitting a first subset of a predetermined memory heap into guarded regions based on a number of objects of interest; allocating each of the objects of interest to a respective one of the guarded regions and remaining objects to a second subset of the predetermined memory heap; executing a program; detecting one of the objects of interest is loaded from one of the guarded regions; generating a trap that transfers control of the executing the program to a signal handler, wherein the signal handler is designated to perform a user-defined task associated with the one of the objects of interest; and executing, by the signal handler of the computing device, the user- defined task.

## Publications

J. Carette, W. M. Farmer, F. Jeremic, V. Maccio, R. O'Connor, and Q. M. Tran, "The MathScheme Library: Some Preliminary Experiments", in: A. Asperti, J. H. Davenport, W. M. Farmer, F. Rabe, and J. Urban, eds., *Conference on Intelligent Computer Mathematics Work-in-Progress Papers Proceedings*, Technical Report UBLCS-2011-04, pp. 10–22, University of Bologna, 2011.

## Education

**McMaster University**, Hamilton, Ontario, Canada
*Master's Student* **2012 - 2013**
M.Eng., Computer Science

**McMaster University**, Hamilton, Ontario, Canada
*Undergraduate Student* **2008 - 2012**
B.Sc., Honours Mathematics and Computer Science

**References available upon request.**