FreeBSD GSOC 2024 Proposal RISC-V

Getz Mikalsen

2024-04-02

Contents

Note: This proposal is also available in the following formats:
ASCII: `https://dflund.se/~getz/GSOC/RV_FreeBSD_proposal.txt`
HTML: `https://dflund.se/~getz/GSOC/RV_FreeBSD_proposal.html`
PDF: `https://dflund.se/~getz/GSOC/RV_FreeBSD_proposal.pdf`

1 General Information

```
Name Getz Mikalsen
Email getz@sdf.org
Phone +46737301192
IRC getz at libera.chat
Matrix @getz:envs.net
Timezone UTC+1
```

I can dedicate a significant amount of my time during the summer. With no other major obligations, I plan to work around 40 hours per week on the proposed tasks and objectives.

As for the timeline, I will align my schedule with the suggested Google Summer of Code program dates. The project will aim to commence on the official coding start date.

I am committed to maintaining open communication with my mentor and the FreeBSD community throughout the project's duration. Through the mailing lists, IRC and a weekly blog post. This will ensure that I stay on track, address any challenges promptly, and incorporate feedback effectively. Regular progress updates and discussions will be a priority to facilitate a smooth and successful collaboration as suggested by GSOC.

1.1 Biography

For the past four years, I have been an active FreeBSD user, running it on my personal laptop. Currently, I serve as a System Administrator at our University's Computer Society, Datorföreningen vid LU/LTH. In this role, I manage a diverse range of computer systems, including legacy machines like VAX, Sun, and SGI, as well as modern servers running FreeBSD, Debian, and OpenBSD. My responsibilities encompass system upgrades, security patch installation, troubleshooting, and backup and recovery procedures.

As an Embedded Electronics Engineering student pursuing a Master's Degree at Lund University (LTH) in Sweden [1], I have gained extensive knowledge in processor design, computer architecture, and I am currently implementing the RISC-V base ISA as a soft core on an FPGA. Furthermore, I have an interest in reverse engineering and actively participate at crackmes.one (previously crackmes.de). Reading and writing assembly code is fun and at times tedious, -O2 is a mess.

Robert Clausecker <fuz@FreeBSD.org> volunteers to mentor the project.

We have discussed the project on #freebsd-soc on libera.chat

2 Porting amd64 libc SIMD enhancements to riscv

2.1 Project Description

The proposed project aims to enhance the performance of the FreeBSD libc library by expanding its performance enhancements to the RISC-V architecture. This will be done by introducing Scalar and RVV support for optimized string manipulation functions. A dispatch framework will be implemented to dynamically select the best implementation based on hardware capabilities. Algorithms are already in place and written by fuz for the amd64 SIMD libc enhancement.

The porting will involve addressing the lack of pmovmskb instruction in RVV, an instruction heavily used in the existing amd64 implementation. Furthermore, the project will implement a dispatch framework for scalar support by introducing an `"riscv_archlevel.h"` file etc. to handle different architectural levels.

I dont currently have access to any RISC-V hardware but I am willing to purchase some devkits by myself and contact other FreeBSD developers regarding borrowing some time on their machines. I have already discussed this on #freebsd-riscv at libera.chat and with fuz.

2.1.1 libc Porting and Optimization

The initial phase of the project (before midterm evaluation) is suggested to concentrate on porting and optimizing smaller libc functions from the *src/lib/libc/amd64/string* directory, such as strlen. This to gather feedback and these functions will later serve as a foundation for optimizing other string manipulation routines, including strchrnul, strcat, and related functions.

2.2 Technical Approach

The majority of the effort will be dedicated to mapping existing SIMD algorithms to riscv counterparts and developing vectorized implementations of string manipulation algorithms.

While RVV draft 0.7.1 implementations are currently widespread, this project will exclusively target the ratified RISC-V Vector Extension 1.0.0 specification. This ensures long-term benefits of targeting the ratified standard to maximize the project's value for future RISC-V development.

Functions will be modeled after the approach used in the amd64 SIMD enhancements, the framework will allow runtime detection of RVV support and dynamic selection of the most efficient implementation (scalar or vectorized).

I will investigate the challenges of RISC-V's variable vector lengths and the potential lack of direct equivalents for some instructions. And develop strategies to overcome these obstacles and create efficient workarounds with help of the mentor.

2.3 Significance for FreeBSD

RISC-V adoption is growing rapidly. This project positions FreeBSD at the forefront of performance optimization for this emerging architecture. The dispatch framework ensures a future-proof solution that scales alongside RISC-V hardware advancements.

I am committed to becoming a long-term FreeBSD contributor, actively maintaining and further improving the optimized code and hope that this GSOC project will be a good starting point.

2.4 Applicant Background

I am currently pursuing a Master's Degree in Embedded Electronics Engineering at Lund University (LTH) in Sweden, I am passionate about performance optimization and low-power computing. Single Instruction Multiple Data (SIMD) technologies like RVV and NEON offer significant potential for improving the efficiency of software components, such as the libc library, on embedded systems.

I'm currently involved in a research project with a group at LTH creating an in-memory computing platform using memristor arrays. This project ends before the summer. I also help with teaching in our Computer Architecture course and helped a bit with our own simple visual 32-bit RISC-V simulator. [2]

I previously maintained of a branch of GerdaOS [3], for the JioPhone1, a featurephone released in India. Until I broke the device while debugging. :D I have completed relevant courses in computer architecture, digital design, operating systems and effective C programming. I am confident in my ability to contribute effectively to the proposed project.

I have a strong background in programming languages such as C, C++, Rust, and shell scripting (z{sh}, sed, awk). Additionally, I have hands-on experience with patching FreeBSD and am familiar with the workflow, "git pull && make buildworld && make buildkernel".

Im familiar with x86-64 and MIPS assembly which positions me well to grasp RISC-V concepts quickly.

3 Deliverables

Introduce a new ARCHLEVEL flag for the riscv architecture, allowing selection between base, Scalar, and RVV extensions. This will involve creating an riscv_archlevel.h file etc. to manage the different levels of architectural support.

Update the simd(7) manual page to document the new SIMD-accelerated string functions for riscv, including details on the ARCHLEVEL flag and its usage.

And finally port the functions with already existing SIMD implementations for amd64 to riscv.

All development and contributions will be hosted in a publicly accessible fork of the FreeBSD source code repository (freebsd-src git repository). The primary deliverable functions for this project are the following:

```
Functions to be Ported
------------------------

HEADER    FUNCTION  NOTES
string.h  stpcpy    String copy functions
          stpncpy
          strcat
          strncat
          strcpy
          strncpy
          strlcpy
          strlcat
          strchrnul
          strrchr
          strcspn
          strspn
          strpbrk
          strsep    String tokenisation functions
          strcmp    String comparison functions
          strncmp
          memcpy    Memory copy functions
          memccpy
          memset    Memory initialisation functions
          memchr    Memory search functions
          memrchr
          memcmp    Memory comparison function
          strlen    String length
          timingsafe_bcmp Constant time functions
          timingsafe_memcmp
```

```
Additional functions that will benefit from these enhancements:

strings.h index           *legacy, through strchr
          rindex         *legacy, through strrchr
          bcmp            *legacy, through memcmp
          bcopy           *legacy, through memmove
          bzero           *legacy, through memset
          explicit_bzero  through memset
          memset_s        through memset
          mempcpy         through memmove
          memcpy          through memmove
          strdup          through memcpy, strlen
          strtok          through strtok_r
```

3.1 Testing

Performance benchmarking will be carried out similarly to how fuz did when he implemented the libc SIMD enhancements for amd64 using his strfperf [4] utility. Functionality tests are located in *src/lib/libc/tests/string*, if further tests are needed they will be written.

For example `/src/lib/libc/tests/string/memcmp_test.c` has support for custom memcmp function. Helping development tremendously.

4 Project Schedule

The submitter is based in Copenhagen, Denmark.

I intend to continue studying the RISC-V Vector extension ahead of the start date and provide a proof of concept SIMD port for strlen(3) for the first week. Then gather feedback and investigate the need for additional unit tests for the ported functions.

This work will build from the same algorithmic ideas developed in the SIMD enhancened functions provided by fuz last year [5].

4.1 Timeline

350 hours are allocated for the project as per the GSOC guidelines for a large project.

I request the coding start date to be moved up from May 27 to May 31st due to collisions with an University exam on the 30th

May 1-26 Community bonding

May 31 Coding start

July 8 Midterm goals reached

July 12 Midterm eval deadline

August 26 - September 2 Final GSoC contributor evaluations

[1] `https://www.lunduniversity.lu.se/lubas/i-uoh-lu-TAEEE`

[2] `https://github.com/soppelmann/Masimulator`

[3] `https://archive.luxferre.top/gerdaos`

[4] `https://github.com/clausecker/strperf`

[5] `https://lists.freebsd.org/archives/freebsd-current/2023-December/005195.html`