RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Arthur Heimbrecht

Bachelor Thesis

HD-KIP-TODOTODOTODO

KIRCHHOFF-INSTITUT FÜR PHYSIK

Department of Physics and Astronomy
University of Heidelberg

Bachelor Thesis
in Physics
submitted by
Arthur Heimbrecht
born in Speyer

TODO 2123

Bachelor Thesis

This Bachelor Thesis has been carried out by Arthur Heimbrecht at the

Kirchhoff Institute for Physics

Ruprecht-Karls-Universität Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Bachelor Thesis

As part of the Human Brain Project, BrainScaleS is a unique project on many levels. This includes a processor solely used by the HICANN-DLS, which manages synaptic weights for every neuron built into one of the many wafers. To accelerate the speed at which this so called plasticity processor unit (PPU) computes all synaptic weights of every neuron used, the processor has an extended instruction set architecture (ISA) that supports vector registers and single input multiple data (SIMD). This report deals with the task of adding built-in functions to an existing backend of GCC, specifically the one used by the PPU, in order to extend the already implemented set of functions according to the users needs.

Contents

1 Compiler Structure 1

2 Discussion 3

3 Outlook 4

Appendix 5

Bibliography 7

1 Compiler Structure

Most Compilers that are used nowadays are built of three basic components which handle different steps in the process of converting human-readable programming language to machine-readable machine code. As does the GNU Compiler collection (GCC) which also can be seen as made of three main parts. The so called front-end, middle-end and back-end. All three parts work more or less independently from each other and communicate over a compiler-specific „language“, which is described a the Intermediate Representation (IR). It is typically never seen by the user and exists for a fact in many different forms [reference] one of which is Register Transfer Language (RTL), which is the lowest-level IR used by GCC. It is the most interesting IR when working on a back-end and will get more attention later in this report. But in before that we need to understand the structure an functioning of a compiler in general.

The first mentioned Front-end resembles the main interaction point between the human programmer and the machine. Front-ends are usually divided by their respective programming languages such as C, C++, Java... and have the main task of converting any programming language into unified IR, that can be passed to the middle-end in GIMPLE or GENERIC language. Therefore no matter which language you prefer, in an ideal case code, that is syntactically identical, should not differ after it is processed by the front-end. This is due to the goal of compilers such as GCC and LLVM to support as many languages and machines as reasonably possible while offering the equally good optimization and saving themselves overall work. It obvious that a single compiler for every combination of language and machine would simply not be practical, especially as the optimization taking place in the middle-end follows the same rules for pretty much any architecture. The middle-end main task is basically this sort of optimization, and makes for the main difference between compilers as most compilers offer the same range of front-ends and back-ends. (Part about the optimizations taking place in the middle-end). After all optimizations are through, the middle-end passes IR in form of RTL to the back-end. As you can see the middle-end rarely needs to be modified except for fundamental changes in the compilers architecture such as new kinds of optimizations and „multiple memory handling“ (also Harvard-Architecture (vielleicht)) The back-end is responsible for the final steps of the compilation process as it translates the general RTL IR into specific Assembly commands. It uses some sort of table of available assembly instructions, that is provided, and finds the best fitting instructions. GCC for example uses a Lisp-like language (is this RTL?) that uses something called insns. These combine different properties with the Assembly commands like an equivalent set of actions that are executed, the operands and the constraints it must satisfy. These will be further explained later. The back-end also contains the the code which implements processor-specific built-in functions. Depending on the Compiler architecture the back-

end it finally emits IR to the assembler which emits the machine code in assembly or emits assembly itself. Then finally the linker links the assembly code of all program files together and substitutes the offset addresses with absolut addresses to generate the final machine code. reload register spilling register handling endianess wordsize

2 Discussion

Discussion...

3 Outlook

Outlook...

Appendix

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, January 22, 2017

.....
(signature)