

Re: New C compiler and analyzer lang/cparser in ports

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2008-11/msg00286.html>

- *From:* Christoph Mallon <christoph.mallon@xxxxxx>
 - *Date:* Thu, 27 Nov 2008 23:22:22 +0100
-

Eygene Ryabinkin schrieb:

Christoph, good day.

Thu, Nov 27, 2008 at 09:39:45PM +0100, Christoph Mallon wrote:

A few days ago libFIRM[1] and cparser were added to the ports tree. If you want to see, what other compilers besides GCC have to offer, this might be of interest for you. libFIRM is a modern optimizing intermediate representation (IR) library. cparser is a C compiler providing many useful warnings and uses libFIRM for optimization and code generation.

[...]

The whole description looks like that of LLVM GCC port, <http://llvm.org/>
Do you know something about that project and if yes, could you, please, provide brief comparison of these two?

Both LLVM and FIRM use SSA as an important aspect of their IR. FIRM uses it more extensively in backend. The whole FIRM backend operates on program graphs in SSA form. Spilling and register allocation are decoupled: First spilling is done and after that registers are allocated. There are several spilling algorithms available. The best so far is one based on the idea of Belady's algorithm and it produces very good results. Our novel approach to register allocation and copy coalescing was developed as PhD thesis by Sebastian Hack. LLVM uses a more classical approach. It has a combined register allocation and spilling phase using a modified linear scan algorithm. LLVM not only needs an instruction schedule, but also a block schedule. FIRM does block scheduling after register allocation is done.

The SSA based spilling of FIRM leads to better code (i.e. less spill and reloads) in most cases. A paper by Matthias Braun and Sebastian Hack submitted to the next CC elaborates on this subject.

In the middle end both systems are quite similar, but there are some notable differences. LLVM uses instructions scheduled in basic blocks whereas FIRM resembles the "sea of nodes" proposed by Click and has no schedule in the "middleend". Actually a schedule is introduced after instruction selection. This is necessary, so spilling and register allocation can operator. To be more precise our representation is called an "explicit dependency graph": All necessary dependencies (we use dependencies, not (data) flow) are modelled as edges in the program graph. We use no temporary variables, but every node in the graph represents a value. If a node produces several results, projection nodes are used to get the different aspects. Memory is modelled as SSA value, too. You can have a look at this in our online demo of the compiler. The link is on our website.

Re: New C compiler and analyzer lang/cparser in ports

You can also read a bit more about FIRM there, too.

You mentioned LLVM GCC. FIRM only has a stale GCC frontend, which is not maintained anymore.

Currently there are three frontends available: C and Java using the Edison Design Group frontend and our own C frontend called "cparser". The latter is probably the more interesting option for you.

Did you find the frontend examples interesting or are you mainly interested in optimization and backend aspects?

Regards

Christoph

freebsd-hackers@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>

To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@xxxxxxxxxxx"