Chapter 5
Processor Designer

In this book, the Language for Instruction-Set Architectures (LISA) ADL is used and extended for automatic generation of C compilers. LISA is the key component of the Processor Designer ASIP design environment, formerly known as the LISA processor design platform (LPDP) [15, 16]. It was initially developed at the Institute for Integrated Signal-processing Systems at the RWTH Aachen University [119] and is now commercialized by CoWare Inc. [58]. The LISA design methodology can be considered as one of the most powerful and comprehensive ADL-based design platform available today and is also well recognized by academia and industry. It enables an efficient design space exploration to tailor a processor architecture to the intended range of applications. During the process, the micro-architecture, instruction-set, register, and memory configuration are investigated and optimized.

The LISA-based design space exploration and the related tools are briefly introduced in the following sections. Afterward, the LISA language as far as relevant to understand the compiler generation techniques presented in this book is introduced in the next section. A detailed overview about LISA and the generated software development tools is given in [15]. Finally, Section 5.3 describes the current tool flow for C compiler generation.

5.1 Design Space Exploration

As illustrated in Fig. 5.1, a single LISA processor description drives all ASIP design phases: architecture exploration, architecture implementation, software tools generation, and system integration (see Section 2.1). Using the LISA language, changes to the processor architecture can be quickly modeled. In this way, an efficient exploration of the architectural design space is ensured.

5.1.1 Software Tool Generation

The Processor Designer provides an integrated design environment (IDE) to support the manual creation and configuration of the LISA model. From the IDE the
so-called _LISA processor compiler_ is invoked. It parses the description and builds the software development tools based on a set of retargetable software development tools.

The assembler is retargeted to the specialized instruction-set. It processes assembly files and produces the object code for the target architecture. Additionally, a macro assembler is provided for user convenience. The automatically generated linker then combines several object files to a single executable in the ELF format [262]. The linker has to be adapted to the target-specific memory organization. More detailed information about the assembler, macro assembler, and linker can be found in [18].

The generated simulator basically can be split into frontend and backend. The former supports the typical functionality such as disassembly, loop and execution profiling, and pipeline utilization. It provides all profiling information as required for design space exploration. The backend supports various kinds of simulation techniques, such as interpretative simulation, compiled simulation [90], and just-in-time (JIT) [27, 28] simulation. As shown in [92], the performance of the generated simulators strongly depends on the abstraction level of the LISA model and the accuracy of the memory model.

A CoSy-based C compiler is manually retargeted via a graphical user interface (GUI) [168], called _Compiler Designer_ (see Section 5.3). Instruction schedulers, though, can already be automatically generated [195].