Binutils 를 이용한 Retargetable Assembler 와 Linker 의 개발[1]

윤종희, 김호균, 안민욱, 최영규, 김대호, 정지문, 백윤흥
서울대학교 공과대학 전기컴퓨터 공학부
e-mail : jhyoun,hkkim,mwahn,ykchoi,dhkim,jmjung,ypaek@optimizer.snu.ac.kr

Development of Retargetable Assembler & Linker based on Binutils

Jonghee Youn, HoKyun Kim, Minwook Ahn, YoungKyu Choi, Daeho Kim, Jimoon Jung,
Yunheung Paek
School of Electrical Engineering, Seoul National University

요 약

CE (Consumer Electronics) 시장에서 Embedded System 은 time-to-market 이라는 개념이 나날이 중요해지고 있다. 시스템의 중심인 core processor 에 대하여 지원하는 여러 가지 software toolkit 의 빠른 개발은 무엇보다 중요해지고 있다. 이 논문에서는 GNU Binutils 를 기반으로 ADL 을 이용하여 Embedded system 의 core processor 를 위하여 신속한 Assembler 와 Linker 를 개발하는 플랫폼을 개발하였다. 이 플랫폼은 서울대학교 소프트웨어 최적화 연구실에서 개발한 ADL (Architecture Description Language)[1] 을 이용하여 core processor 를 기술하면 자동으로 Assembler 와 Link 를 생성해주는 시스템이다.

1. 서론

Embedded system 에서 ASIP 이나 DSP 와 같은 core process 의 개발은 매우 중요하다. 전통적으로 DSP 와 ASIP 은 코드의 성능을 높이기 위하여 손으로 assembly 작성을 많이 해왔다. 그러나 어플리케이션의 복잡성과 사이즈가 증가함에 따라 embedded system 의 개발자들은 HLL (High Level Language) 와 컴파일러를 이용하여 코드를 생성함으로써 assembly 코드를 만드는 과정에서 오는 부담을 줄이기를 원하고 있다.

Embedded system 에서 다른 한가지 중요한 이슈는 time-to-market 이다. 새로운 processor 에 대하여 software toolkit (compiler, assembler, linker, debugger, simulator 등) 을 빠르게 개발하는 것은 새로운 제품의 출시에 중요한 역할을 하고 있다. 이러한 embedded system 시장에서의 경향에 맞추어 본 논문에서는 GNU Binary Utility (Binutils) 를 기반으로 한 Retargetable Assembler 와 Linker 를 개발하고자 한다.

본 논문의 2 장에서는 GNU Binutils 의 전체적인 구조와 Retargetable Assembler 와 Linker 를 개발하기 위해서 필요한 부분이 무엇인지에 대해서 설명하겠다. 그리고 3 장에서는 Retargetable Assembler 와 Linker[2] 에 필요한 파일을 자동으로 생성해주는 플랫폼에 대해서 설명하고 마지막으로 4 장에서는 결론을 내리겠다.

2. GNU Binary Utility (GNU Binutils)

GNU Binutils 는 GNU project 의 일부분으로 object 파일을 처리하기 위한 여러 가지 프로그래밍 툴들의 모음이라고 볼 수 있다. Binutils 에 대한 자세한 설명은 GNU project[3]에 나와있다. Binutils 에 포함된 프로그래밍 툴들은 다음과 같다.

1. AS : the GNU assembler
2. LD : the GNU linker
3. Addr2line : Converts addresses into filenames and line numbers
4. Ar : A utility for creating, modifying and extracting from archives
5. C++ filt : Filter to de-mangle encoded C++ symbols
6. Nm : Lists symbols from object files
7. Objcopy : Copy and translates object files
8. Objdump – Display information from object files
9. Ranlib : Generates an index to the contents of an archive
10. Readelf : Displays information from any ELF format

[1] 본 연구는 교육과학기술부/한국과학재단 우수연구센터육성사업(R11-2008-007-01001-0), 지식경제부 출연금으로 ETRI, SoC 산업진흥센터에서 수행한 ITSoC 핵심설계인력양성사업, 서울시 산학연 협력사업, 2008 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(R0A-2008-000-20110-0), 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업(IITA-2008-C1090-0801-0020), 지식경제부 및 정보통신연구진흥원의 IT 원천기술개발사업[과제관리번호: 2006-S-006-02, 과제명: 유비쿼터스 단말용 부품/모듈] 의 지원을 받아 수행되었습니다.

object files
11. Size : Lists the section sizes of an object or archive file
12. Strings : Lists printable strings from files
13. Strip : Discards symbols
14. Gprof : Displays profiling information

아래의 **그림 1** 은 GNU Binutils 의 전체적인 구조를 나타낸다. 그림에서와 같이 GNU Binutils 는 크게 4 개의 부분으로 나누어져 있다. 먼저 Opcode 부분은 instruction 의 encoding, decoding 등의 정보를 생성해준다. 이 부분은 CGEN 이라는 interpreter 를 사용하여 필요한 파일들을 생성할 수 있다. CGEN interpreter 는 cpu 파일을 입력으로 받아 새로운 processor 에 필요로 하는 파일을 생성한다. 그리고 BFD 부분은 여러 가지 object 파일을 처리하기 위한 중요한 부분이다. BFD 에서는 여러 가지 object file format 과 여러 종류의 프로세서 그리고 다양한 운영 체제를 지원하고 있

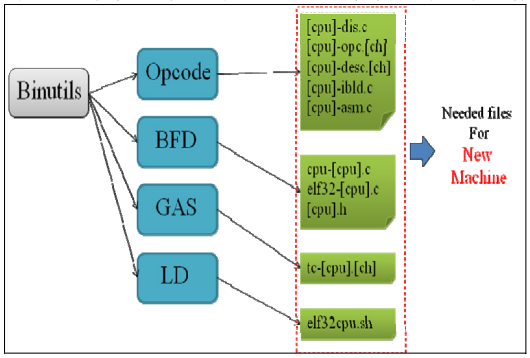

그림 1 GNU Binutils 의 구조

다. 마지막으로 GAS 와 LD 부분은 assembler 와 linker 에 관련된 부분이다.

위의 **그림 1** 에서 오른쪽에 있는 파일들은 새로운 프로세서를 대상으로 할 경우 각 부분에서 필요로 하는 파일들이다. 위의 파일 중에서 Opcode 부분에 필요한 파일은 CGEN interpreter 를 통하여 자동으로 생성될 수 있다. 따라서 BFD, GAS, LD 부분에서 필요한 파일들만을 자동으로 생성해 준다면 그 프로세서에 맞는 assembler 와 linker 를 얻을 수 있다.

3. Retargetable Assembler & Linker

아래의 **그림 2** 는 Retargetable assembler 와 linker 가 생성되는 과정에 대하여 나타내고 있다.

아래의 그림에서 SoarDL 은 processor 를 위한 서울대학교 소프트웨어 최적화 연구실에서 개발한 ADL 이다. SoarDL 로 새로운 processor 에 대하여 기술을 하면 Soar BINUTILS 에서 SoarDL 로 기술된 정보를 분석하여 assembler 와 linker 을 생성하는데 필요한 파일을 자동으로 생성한다. 여기에서 CGEN 은 interpreter 로 [cpu].cpu 파일을 해석하여 Binutils 의 Opcode 부분에 필요한 파일을 생성한다.

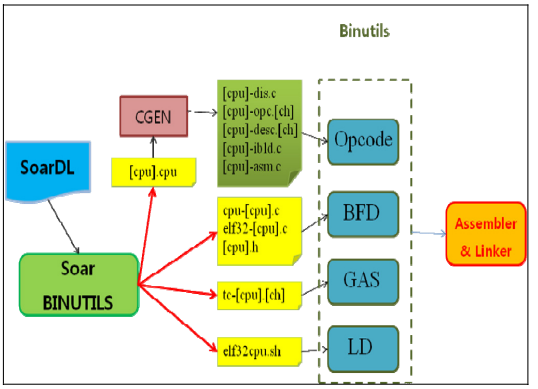

그림 2 Retargetable assembler & linker 생성 과정

결국 위의 **그림 2** 에서 보는 바와 같이 새로운 processor 에 대하여 SoarDL 로 기술하면 원하는 processor 에 대한 assembler 와 linker 를 생성할 수 있다.

SoarDL 로 기술된 간단한 명령어를 예로 들어 assembler 와 linker 가 생성되는 지에 대하여 알아 보겠다.

```
Instruction add : arith
{
        r rD, rA, rB;
        action {
                rD = ss_plus(rA, rB);
        }
        syntax {
                "l.add "::rD::", "::rA::", "::rB;
        }
        encoding {
        rD::rA::rB::RESERVED[1]::"00"::RESERVED[4]
:         :"0000";}
        cost(1);
}
```

그림 3 ADD instruction

위의 **그림 3** 은 Openrisc processor 의 ADD instruction 을 SoarDL 로 기술한 것이다. 위의 그림 3 에서 assembler 와 linker 에 필요한 정보는 encoding 부분에 기술되어 있다. 물론 이외에도 processor 의 hardware register, memory, relocation 등의 정보를 기술해 주어야 한다. 이러한 정보를 이용하여 Soar BINUTILS 에서 여러 파일을 생성한다. 이러한 파일을 Binutils 에 입력한 뒤 실행하게 되면 원하는 processor 의 assembler 와 linker 를 얻을 수 있다.

4. 결론

이 논문에서는 GNU Binutils 를 이용하여 새로운 processor 의 Assembler 와 Linker 를 생성하는 것에 대하여 알아 보았다. SoarDL 을 이용하여 processor 의

정보만을 기술한다면 편리하게 assembler 와 linker 를 생성할 수가 있다는 점에서 embedded system 에서 SoarBINUTILS 가 유용하게 사용될 수 있을 것이라 생각한다.

참고문헌

[1] 안민욱, 백윤흥, 조정훈 “응용 프로그램에 특화된 명령어를 통한 고정 소수점 오디오 코덱 최적화를 위한 ADL 기반 컴파일러 사용”, 정보처리학회논문지 A 제 13-A 권 제 4 호(2006.08)

[2] Maghsoud Abbaspour, Jianwen Zhu, “Retargetable Binary Utilities”, Design Automation Conference, June 10-14, 2002

[3] GNU project, http://www.gnu.org/software/binutils