산업기술연구(강원대학교 산업기술연구소 논문집), 제30권 B호, 2010.
Journal of Industrial Technology, Kangwon Natl. Univ., Korea, No. 30 B, 2010.

LC-3 프로세서를 위한 어셈블러 및 시뮬레이터의 구현

An Implementation of Assembler and Simulator for LC-3 Processor

이 기 민* 이 민 우** 김 용 석***
Lee, Ki-Min Lee, Min-Woo Kim, Yong-Seok

Abstract

LC-3 is an educational microprocessor for processor logic design, instruction execution fundamentals, and assembly language programming. This paper presents an IDE software, LIDE (LC-3 Integrated Development Environment), which integrates assembly program editor, LC-3 assembler, and LC-3 simulator.

키워드 : *마이크로프로세서, 통합개발환경, 어셈블러, 시뮬레이터*
Keywords : *microprocessor, IDE, LC-3, assembler, simulator*

1. 서론

프로세서를 이해하는데 있어서 어셈블리 언어는 핵심적인 역할을 한다. 프로세서의 프로그래밍 모델과 이를 이용하여 어셈블리 언어로 적절히 프로그램을 작성하면서 프로세서가 어떻게 동작하는지를 들여다 볼수 있고, 이를 바탕으로 하여 C 언어와 같은 고급 언어로 작성된 프로그램이 어떻게 어셈블리 언어 및 기계어 프로그램으로 변환되어 프로그래머가 원하는 대로 동작되는 지를 이해할 수 있게 된다. 또한 어셈블리 언어는 고급 프로그래머가 능숙하게 사용할 수 있어야 하는 이유가 있는데, C 언어로는 표현이 불가능한 내용을 표현하는 데에도 필수적이고, 실행속도나 메모리 용량을 최적화해야 하는 경우에도 어셈블리 언어로 직접 작성하면 컴파일러가 자동으로 생성하는 코드보다 우수한 결과를 얻을 수 있다.

* 강원대학교 컴퓨터학부 컴퓨터정보통신공학전공 재학
** 강원대학교 컴퓨터학부 컴퓨터정보통신공학전공 학사
*** 강원대학교 컴퓨터학부 교수 (교신저자)

본 논문에서는 교육용으로 널리 사용되는 프로세서인 LC-3 (Little Computer 3)를 근간으로 하여, 에디터와 어셈블러 및 시뮬레이터가 하나로 통합된 IDE[5] (Integrated Development Environment) 형태의 LIDE (LC-3 IDE) 소프트웨어 개발에 대해서 기술한다. LC-3 는 마이크로프로세서의 동작원리와 어셈블리 언어 및 기계어를 교육하는데 많이 사용되고 있는 어셈블리 언어를 포함한 마이크로프로세서이다[1][6]. LC-3 어셈블리 언어는 비교적 간단한 명령어 집합을 가지고 있지만, C 언어의 컴파일도 가능한 기능들을 보유하고 있다. LIDE는 어셈블리 프로그램 에디터, LC-3 어셈블러, 및 LC-3 시뮬레이터를 하나의 통합된 소프트웨어로 구현하였다.

2. LC-3 프로세서와 LIDE의 구성

2.1 LC-3 프로세서

LC-3는 마이크로프로세서의 동작원리와 어셈블리 언어 및 기계어를 교육하는데 많이 사용되고 있는 어셈블리 언어를 포함한 마이크로프로세서이다. LC-3 어셈블리 언어는 비교적 간단한 명령어 집합을 가지고 있지만, C 언어의 컴파일도 가능한

기능들을 보유하고 있다.

LC-3는 16비트의 레지스터와 16비트의 주소, 64KB의 메모리 공간을 갖는 간단한 프로세서이다. 범용 레지스터는 R0부터 R7까지 8개를 가지고 있고 자유로이 사용할 수 있으며 16비트로 된 명령어는 상위 4비트는 OPCODE 나머지 비트는 명령어에 따라 OPERAND로 사용된다. 명령어에는 표 1에서 볼 수 있듯이 덧셈이나 AND, NOT연산 같은 기능을 수행하는 연산명령어, 레지스터의 내용을 메모리에 저장하는 STORE (ST) 관련 명령어, 메모리의 내용을 레지스터에 저장하는 LOAD (LD) 관련 명령어, 조건에 따라 분기를 제어하는 BRANCH (BR) 및 JUMP (JMP) 명령어, 함수 호출 및 복귀 관련 명령어 (JSR, RET), 인터럽트로부터 복귀하는 명령어 (RTI), 그리고 입출력 및 종료와 관련하여 시스템 호출 기능을 담당하는 TRAP 명령어 등이 있다.

표 1. LC-3 명령어

명 령 어	문 법
ADD	ADD DR, SR1, SR2 ADD DR, SR1, imm5
AND	AND DR, SR1, SR2 AND DR, SR1, imm5
NOT	NOT DR, SR
ST	ST SR, LABEL
STI	STI SR, LABEL
STR	STR SR, BaseR, offset6
LD	LD DR, LABEL
LDI	LDI DR, LABEL
LDR	LDR DR, BaseR, offset6
LEA	LEA DR,LABEL
BR	BR LABEL BRn LABEL BRp LABEL BRz LABEL BRzp LABEL BRnp LABEL BRnz LABEL BRnzp LABEL
JMP	JMP BaseR
JSR	JSR LABEL
RET	RET
RTI	RTI
TRAP	TRAP trapvector8

2.2 LIDE의 구성

LIDE는 어셈블리 프로그램 에디터, LC-3 어셈블러 및 LC-3 시뮬레이터가 하나로 통합된 개발환경 형태를 갖는다. 그림 1에서와 같이 사용자가 에디터를 이용하여 프로그램을 작성하면 *.asm의 어셈블리 프로그램 파일이 생성된다. 어셈블러는 이 파일에 대하여 문법 오류를 검사하고 적절한 오류 메시지를 로그파일을 통해 에디터의 하단부에 출력한다. 문법 오류가 없으면 어셈블링을 거쳐서 *.obj 의 목적파일을 생성한다. 시뮬레이터는 생성된 목적파일을 사용하여 시뮬레이션을 한다. 시뮬레이션은 매 명령어 단위로 실행할 수도 있고 끝까지 한꺼번에 실행할 수도 있도록 한다.

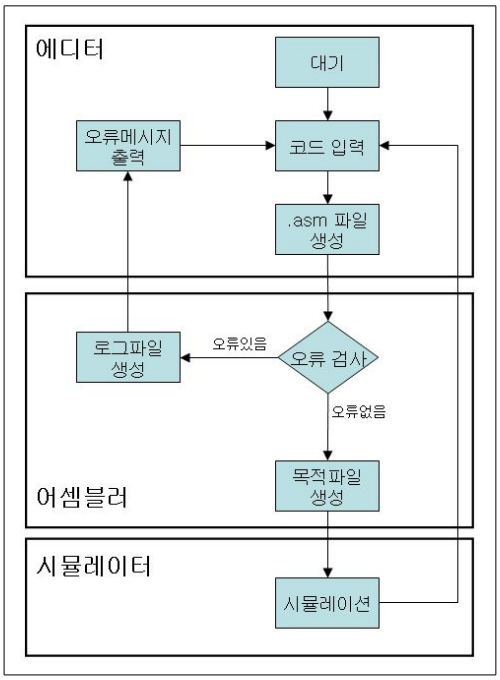

그림 1. LIDE의 순서도

3. 어셈블리 프로그램 에디터

표 2와 같이 LC-3 어셈블리 프로그램 에디터에서는 가독성과 코드의 정확한 편집을 위해 16개의 명령어에 대해서 파란색으로 표시하고, 레지스터나 특정 키워드에 대해서는 굵은 글씨체로 표시하고 숫자와 스트링은 분홍색, 주석은 초록색으로 표시된다. 프로그램은 윈도즈 환경에서 MFC로 작성하였다[2].

표 2. 입력된 문자에 대한 에디터의 표현 색상

	입력문자의 색상
명령어	파 랑 색
명령어, 지시어	검 정 색
상수형 입력 값	분 홍 색
주 석	초 록 색

줄 번호를 에디터 좌측에 붙여 에러 발생 시 빠르게 에러 부분을 찾을 수 있도록 하였고, 그림 2의 아래 부분에 위치한 결과 창은 툴바 형식으로 만들어 위치를 옮길 수 있고 숨길 수 있으며, 어셈블링 결과를 보여준다. 또한 사용자가 명령어의 사용법을 모를 경우 참고 할 수 있도록 에디터 좌측에 명령어 도움말 버튼을 넣어서 해당되는 명령어의 기능 및 문법을 설명하는 창을 열어 참고 하면서 코딩을 할 수 있도록 하였다.

LC-3 어셈블리 언어 에디터를 이용하여 어셈블리 프로그램을 작성하면 텍스트 형태의 파일에 저장된다. 다른 개발 도구와 마찬가지로 최근 저장 이후에 수정된 내용이 있으면 어셈블링 전에 지금까지 작성된 내용이 자동 저장된다. 사용자의 편의를 위해서 단축키를 모두 지정해 놓았으며 아이콘의 그림도 알아보기 쉽도록하여 편리한 사용환경을 제공하도록 하였다.

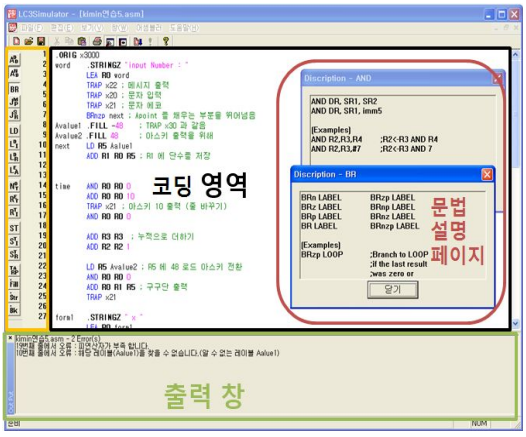

그림 2. 어셈블리 프로그램 에디터의 화면

4. LC-3 어셈블러

4.1 어셈블러의 구성

LIDE의 어셈블러는 문법 오류와 의미 오류를 검사하여 오류 메시지를 출력한다. 오류가 없으면 1대 1로 매칭되는 기계어로 번역하는 역할을 한다. 어셈블러의 구현은 2 패스로 이루어지도록 하였으며, C++를 이용하여 객체 단위로 작성하였다. CReadLine, CLc3Pass1 및 CLc3Pass2의 세 가지 객체로 구분 지을 수 있으며 객체간의 관계는 그림 3과 같다.

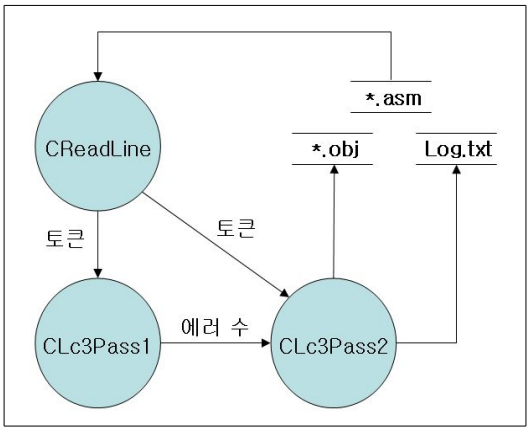

그림 3. 2 패스 어셈블러의 데이터 흐름도

4.2 CReadLine 객체

객체 CReadLine은 소스 프로그램의 스캐너 역할을 가진다. 사용자가 작성한 소스 파일에서 의미를 가지는 가장 작은 단위의 단어로 토큰을 추출해 낸다. 추출된 각 토큰을 구분 짓는 구분자로는 띄어쓰기와 쉼표, 탭 키 및 그리고 개행 문자를 사용한다. 세미콜론으로 시작되는 주석부분은 따로 읽어오지 않는다. 구분된 토큰은 레이블, 명령어, 지시어 혹은 피연산자가 되며, 레이블의 이름은 예약어인 표 3의 명령어나 지시어는 사용할 수 없도록 하였다.

표 3. 명령어와 지시어

명 령 어		지시어	
ADD	STI	.ORIG	PUTSP
AND	LD	.END	HALT
LDR	BR	.FILL	R0
STR	JMP	.BLKW	R1
LDI	JSR	.STRIN	R2
LEA	JSRR	GZ	R3
NOT	TRAP	GETC	R4
ST	RET	OUT	R5
	RTI	PUTS	R6
		IN	R7

지시어에는 R0부터 R7 까지의 레지스터들과 LC-3에서 사용하는 .ORIG .END .FILL .BLKW .STRINGZ와 같은 예약어를 포함한다. 또한 이 객체에서는 어셈블리 코드의 명확성을 두기위해 레이블을 제외한 두 줄 이상의 코드는 서로 다른 명

령어 단위로 판단한다.

4.3 문법 오류 검사

객체 CLc3Pass1은 대부분의 오류를 검사하고 레이블의 주소를 검사하여 심볼 테이블을 만든다. 문법 오류는 표 1의 문법을 따르며 LABEL의 위치에는 현재와 해당 LABEL의 차이 값이 들어가도 무관하다. CReadLine 객체를 이용해 소스 파일에서 토큰을 구분한 후 이 토큰을 바탕으로 오류를 검사하고 오류가 발생하면 발생 횟수를 결과로 출력한다. 표 1과 같이 명령어마다 필요로 하는 피연산자와 사용 방법이 다르고, 명령어 이외의 지시어를 이용하여 코딩이 가능하기 때문에 이 객체에서 이들을 검사한다.

LC-3의 소스 코드에는 시작주소를 표시하는 지시어는 '.ORIG 시작주소'의 형태를 갖는다. CReadLine을 통해 이 지시어를 찾고 그 시점부터 해당 명령어를 반복적으로 처리한다. 처음에 이 부분이 나타나지 않으면 오류 메시지를 출력해 준다. 시작과 마찬가지로 소스의 마지막에는 마지막을 알리는 '.END'로 구성된다. 이 지시어도 '.ORIG' 와 같게 처리하고 이후에 오는 명령어들은 무시된다.

4.4 오류 메시지와 2진 목적파일

객체 CLc3Pass2 부분에서는 문법에 대한 오류 검사와 의미 오류에 대한 심볼 테이블을 검사하여 레이블에 대한 에러를 확인한다. 이 경우 직접적으로 문법오류를 검사하는 것이 아니고 오류에 대한 메시지를 CLc3Pass1에서 전달 받는다. 그리고 이 정보를 통합하여 Log.txt 파일을 생성한다. 사용자가 작성한 소스 파일의 코드에 아무런 오류가 없다면 해당 소스코드에 대한 기계어 코드가 *.obj 형태의 2진 목적파일을 생성한다.

Log.txt 파일은 오류에 상관없이 모두 작성하게 되지만 그 내용은 오류가 없는 경우, 오류가 있는 경우에 따라 달라진다. 첫째로 오류가 없을 경우, 객체 CLc3Pass1으로 부터 얻은 오류의 수(이 경우 0)를 먼저 Log.txt 파일에 출력하고 해당 코드의 시작 주소를 출력해 준다. 이후에는 레이블의 이름과 레이블의 주소를 연속하게 출력해 준다. 이는 시뮬레이터에서 적절히 처리 가능하도록 정보를 넣어준 것이다.

둘째로 오류가 있는 경우 오류의 수를 먼저 출력해 주고 해당 코드의 시작 주소를 출력해 준다. 이후에는 발생한 오류에 대한 메시지들을 한 줄 단위로 출력해 준다. 여기서 에러의 수와 메시지의 수는 동일하며 에디터에서 어셈블러를 실행했을 때 이 오류 메시지들을 화면에 보여준다.

5. 시뮬레이터

어셈블러에 의해 생성된 목적 파일 (*.obj)은 그림 4와 같은 형식의 기계어 코드로 작성된다. 하나의 명령어는 하나의 코드값을 갖게 되는데 12-15 비트 영역에 명령어를 식별할 수 있는 코드가 입력된다. 이를 이용하여 시뮬레이터에서는 어셈블러를 통해 작성된 목적 파일의 2진 코드들을 읽어서 그 코드가 어떠한 명령어인지를 판단하고 해당 명령어에 맞도록 정보를 가져와 명령들을 수행한다.

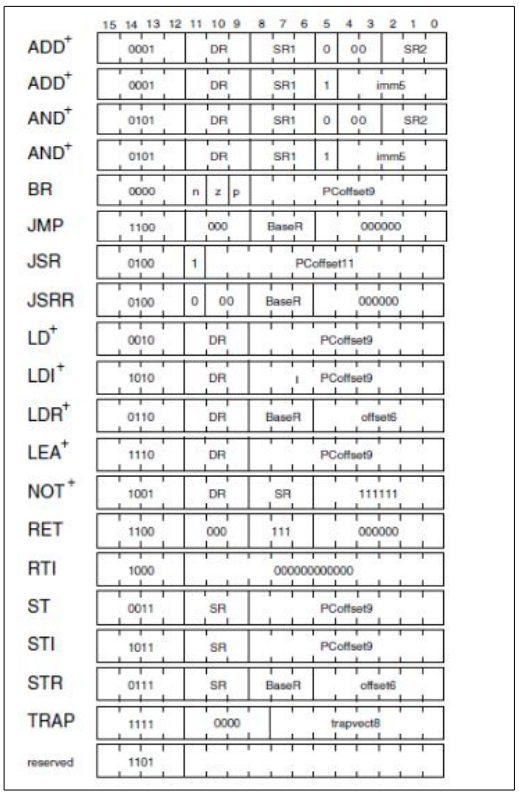

그림 4. 각 명령어에 해당하는 기계어 코드

시뮬레이터에서는 해당 데이터들을 알 수 있도록, 동작 뿐 아니라 특수 레지스터인 PC, CC, IR, 레지스터들의 값을 따로 저장하고 반환해 주는 함수를 작성하였다. PC (Program Counter)는 현재의 실행 주소를 가리키며 메모리에서 이 주소에 해당하는 부분을 읽어와 실행하게 된다. CC (Condition Code)는 n, z, p 세 가지로 구분된다. n은 negative, z는 zero, p는 positive를 의미한다. 이들은 그림 5에서 명령어 다음의 +표시를 가지는 명령어들의 동작에 의해 결정되는데 동작한 결과가 음수이면 n, 0이면 z, 양수이면 p로 저장된다. IR (Instruction Register)은 주소에 해당하는 하나의

명령어를 의미한다. 레지스터는 R0부터 R7까지 8개로 구성되며 프로세서 내부에 존재한다.

시뮬레이터는 MFC로 작성하여 사용자들이 쉽게 값들을 확인 할 수 있도록 작성하였다. 그림 5에서와 같이 크게 4 부분으로 나누어 볼 수 있다. 1 부분에서는 주소와 명령어 그리고 실제 어셈블리 언어를 확인 할 수 있다. 2 부분에서는 실행과 관련된 부분으로 1개의 명령어 수행 혹은 모든 명령어 수행을 할 수 있고, 작성된 코드를 리셋하는 버튼과 특정 위치에서 실행을 멈추게 하는 브레이크 포인트 기능도 가지고 있다. 3 부분은 실행에 대한 결과 창으로 아스키코드에 의해 출력된다. 4 부분에서는 실행에 대한 데이터의 현황을 확인 할 수 있다.

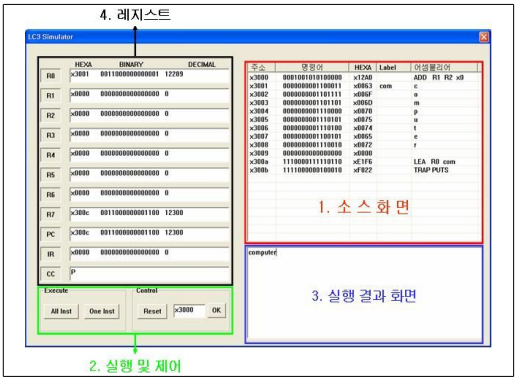

그림 5 시뮬레이터의 실행 화면

6. 결론

어셈블리 언어는 프로세서를 이해하는데 있어서 핵심적인 부분이며, C 언어로는 표현이 불가능한 내용이나 최적화가 필요한 부분을 프로그램으로 작성하는데도 필수적이다. 본 논문에서 기술한 LIDE는 교육용으로 널리 사용되는 LC-3 프로세서의 어셈블리 언어를 사용하여 기계어 프로그램의 단계적 동작결과를 알아보고, 마이크로프로세서의 동작원리를 이해 할 수 있다. 어셈블리 언어 및 마이크로프로세서를 접하기 시작한 사람들이나 초급 프로그래머들이 좀 더 쉽게 이해, 이용할 수 있다는 이점을 가지고 있다. 현재 마이크로프로세서 과목에서 본 논문에서 소개된 LIDE를 실제 활용하고 있는데 어셈블리 언어를 처음 접하게 되는 학생들로써 마이크로프로세서를 쉽게 이해할 수 있게 돕고, 어셈블리 언어를 이용한 프로그래밍 연습에도 유용하게 활용되고 있다.

참 고 문 헌

[1] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems from bits & gates to C& beyond*, McGraw-Hill, August. 2003.

[2] 정일홍, *MFC로 구현한 윈도우 프로그래밍*, 생능, 파주. 8. 2006.

[3] 권호열, *알기쉬운 소프트웨어설계*, 강원대학교 출판부, 춘천, 2. 2006.

[4] 코드 프로젝트, http://www.codeproject.com/

[5] 위키피디아, http://www.wikipedia.org/wiki/LC-3

[6] A. Brownfield and C. Norris, “LC3uArch: A Graphical Simulator of the LC-3 Microarchitecture”, *ACM SIGCSE Bulletin*, vol. 41, issue 1, March, 2009, pp. 413-417.

[7] K. Buchheit, *Guide to Using the Unix version of the LC-3 Simulator*, Jan. 2001.