확장된 가상스택기계의 어셈블리 언어 설명서

(Extended Abstract Stack Machine Assembly Language Manual)

버전 2.0

2019. 4.

충북대학교 소프트웨어학과

이 재 성

jasonlee@cbnu.ac.kr

* 이 설명서는 소프트웨어학과 “컴파일러” 강의의 보조 교재입니다. 개선사항이나 문의 사항은 저자에게 메일로 연락바랍니다. 가상스택기계(Abstract Stack Machine)는 “compilers principles, techniques, and tools”, A. V. Aho, R. Sethi, J. D. Ullman. 책의 2장에서 소개되어 있습니다. 본 설명서는 이를 확장한 어셈블리어와 이를 구현한 어셈블러와 시뮬레이터 사용법을 설명한 것입니다.

목차

1. 명령어 일람표

명령어 형식		설명 C언어 코드로 대응 의미 해석 사용예
PUSH num	설명: C해석: 사용예:	num값을 스택에 푸쉬한다. (num은 숫자값) push(num); push 5
POP	설명: C해석: 사용예:	스택의 톱에 있는 값을 팝하여 돌려준다. pop(); pop
COPY	설명: C해석: 사용예:	스택 상위에 있는 값을 복사하여 스택에 푸쉬한다. v=pop(); push(v); push(v); copy
RVALUE dataloc	설명: C해석: 사용예:	dataloc의 위치에 저장되어 있는 실제 값을 스택에 푸쉬한다. (dataloc은 데이터 위치를 나타내는 레이블 명) push(data[dataloc]); rvalue sum (sum이라는 변수명의 rvalue값을 푸쉬)
LVALUE dataloc	설명: C해석: 사용예:	dataloc값(lvalue) 자체를 스택에 푸쉬한다. (dataloc은 데이터 위치를 나타내는 레이블 명) push(dataloc); lvalue sum (sum이라는 변수명의 lvalue값을 푸쉬)
:=	설명: C해석: 사용예:	톱에 있는 값을 (톱-1)에 있는 자료 메모리 영역에 복사하고, 스택에서 두 개를 모두 팝해 버린다. v=pop();dataloc=pop();data[dataloc]=v; :=
+	설명: C해석: 사용예:	스택의 톱과 (톱-1)의 값을 팝하여 더한 후, 결과를 스택에 푸쉬한다. b=pop(); a=pop(); push(a+ b); +
-	설명: C해석:	스택의 톱과 (톱-1)의 값을 팝하여 뺀 후, 결과를 스택에 푸쉬한다. (스택의 톱-1에 있는 값을 스택의 톱에 있는 값으로 뺌) b=pop(); a=pop(); push(a-b);

	사용예:	-
*	설명: C해석: 사용예:	스택의 톱과 (톱-1)의 값을 팝하여 곱한 후, 결과를 스택에 푸쉬한다. b=pop(); a=pop(); push(a*b); *
/	설명: C해석: 사용예:	스택의 톱과 (톱-1)의 값을 팝하여 나눈 후, 결과를 스택에 푸쉬한다. (스택의 톱-1에 있는 값을 스택의 톱에 있는 값으로 나눔) b=pop(); a=pop(); push(a/b); /
GOTO memloc	설명: C해석: 사용예:	프로그램의 다음 수행 순서(PC에 저장된 값)를 memloc 위치로 변경한다. (memloc은 명령어 위치를 나타내는 레이블 명) pc=memloc; goto loop (loop라는 레이블로 점프)
GOFALSE memloc	설명: C해석: 사용예:	스택의 톱값을 팝한 후, 그 값이 0이면, 프로그램의 다음 수행 순서(PC에 저장된 값)를 memloc위치로 변경한다. (memloc은 명령어 위치를 나타내는 레이블 명) if(pop()==0) pc=memloc gofalse loop (스택의 톱값이 0일 경우 loop라는 레이블로 점프)
GOTRUE memloc	설명: C해석: 사용예:	스택의 톱값을 팝한 후, 그 값이 0이 아니면, 프로그램의 다음 수행 순서(PC에 저장된 값)를 memloc위치로 변경한다. (memloc은 명령어 위치를 나타내는 레이블 명) if(pop()!=0) pc=memloc gotrue loop (스택의 톱값이 0이 아닐 경우 loop라는 레이블로 점프)
GOPLUS memloc	설명: C해석: 사용예:	스택의 톱값을 팝한 후, 그 값이 양(+)이면, 프로그램의 다음 수행 순서(PC에 저장된 값)를 memloc위치로 변경한다. (memloc은 명령어 위치를 나타내는 레이블 명) if(pop()>0) pc=memloc goplus loop (스택의 톱값이 양의 수일 경우 loop라는 레이블로 점프)
GOMINUS memloc	설명:	스택의 톱값을 팝한 후, 그 값이 음(-)이면, 프로그램의 다음 수행 순서(PC에 저장된 값)를 memloc위치로 변경

		한다. (memloc은 명령어 위치를 나타내는 레이블 명)
	C해석:	if(pop()<0) pc=memloc
	사용예:	gominus loop
		(스택의 톱값이 양의 수일 경우 loop라는 레이블로 점프)
HALT	설명:	프로그램의 수행을 종료한다.
	C해석:	return;
	사용예:	HALT
INNUM	설명:	숫자를 입력으로 받아, 그 값을 스택에 푸쉬한다.
	C해석:	scanf("%d", &x); push(x);
	사용예:	INNUM
INCH	설명:	문자를 입력으로 받아, 그 값을 스택에 푸쉬한다.
	C해석:	x=getchar(); push(x);
	사용예:	INCH
OUTNUM	설명:	스택의 톱값을 팝한 후, 숫자값으로 출력한다.
	C해석:	printf("%d", pop());
	사용예:	OUTNUM
OUTCH	설명:	스택의 톱값을 팝한 후, 문자값(한글자)으로 출력한다.
	C해석:	x=pop(); putchar(x);
	사용예:	OUTCH
DW dataloc	설명:	dataloc의 자료 위치를 나타내며, word크기 만큼 위치를 확보한다.
	C해석:	대응되는 C코드처리는 없음
	사용예:	DW sum
		(sum 이라는 변수명을 위해 메모리 확보)
LABEL memloc	설명:	현재의 명령어 수행위치를 memloc이라는 이름의 레이블에 정해준다.
	C해석:	대응되는 C코드처리는 없음
	사용예:	LABEL loop
		(현재의 위치를 loop라는 레이블로 지정)
END	설명:	프로그램의 끝을 나타냄
	C해석:	대응되는 C코드처리는 없음
	사용예:	END
$	설명:	컴멘트(주석) 문장임을 나타냄, 이 기호가 나온 이후, 그 줄 끝까지 컴멘트로 처리됨 (대응되는 C코드처리는 없음)
	사용예:	$ --이것은 주석문임
		OUTNUM $ 결과 출력 (컴멘트)

2. ASM 어셈블리어의 수행

사용자가 작성한(혹은 컴파일러에 의해 생성된) ASM 어셈블리어 프로그램은 우선 ASM 어셈블러에 의해 기계어로 번역되고, 그 기계어를 ASM 시뮬레이터가 시뮬레이션하여 결과를 생성한다. 이를 간단하게 도식화한 것이 그림 1이다.

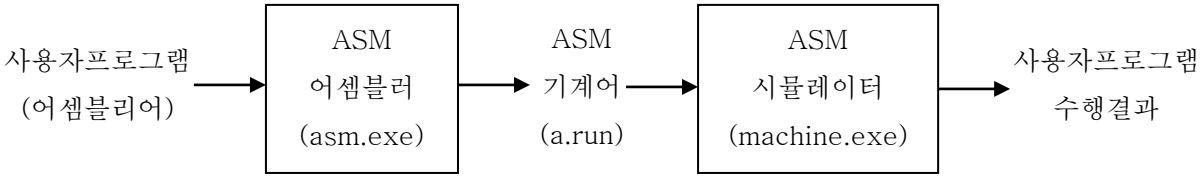

그림 1. ASM 어셈블리어의 수행 단계

어셈블러와 시뮬레이터는 모두 MS-DOS 명령어 프롬프트상에서 수행되며 그 사용예는 그림 2와 같다. 우선 프롬프트상에서 "asm sample.asm"을 입력하여 sample.asm 프로그램을 어셈블러가 번역하도록 한다. (sample.asm은 두수를 입력으로 받아 변수 A, B에 저장하고, 그중에서 큰 것을 MAX에 저장한 후, 그 결과를 출력하는 프로그램으로 4장에 소개되어 있다.) 번역 결과인 기계어는 "a.run" 파일에 저장된다. 또, "machine a.run"을 수행시켜 시뮬레이터(machine.exe)가 "a.run"파일을 수행하도록 한다. a.run의 수행과정에서 10과 20을 A, B의 변수에 각각 입력받고, 그 중 큰 값을 MAX값을 출력한 것을 그림2에서 볼 수있다. a.run의 수행이 종료될 때는 데이터 영역에 할당된 모든 변수 값을 출력하도록 하여 프로그램의 수행 결과를 볼 수 있도록 했다.

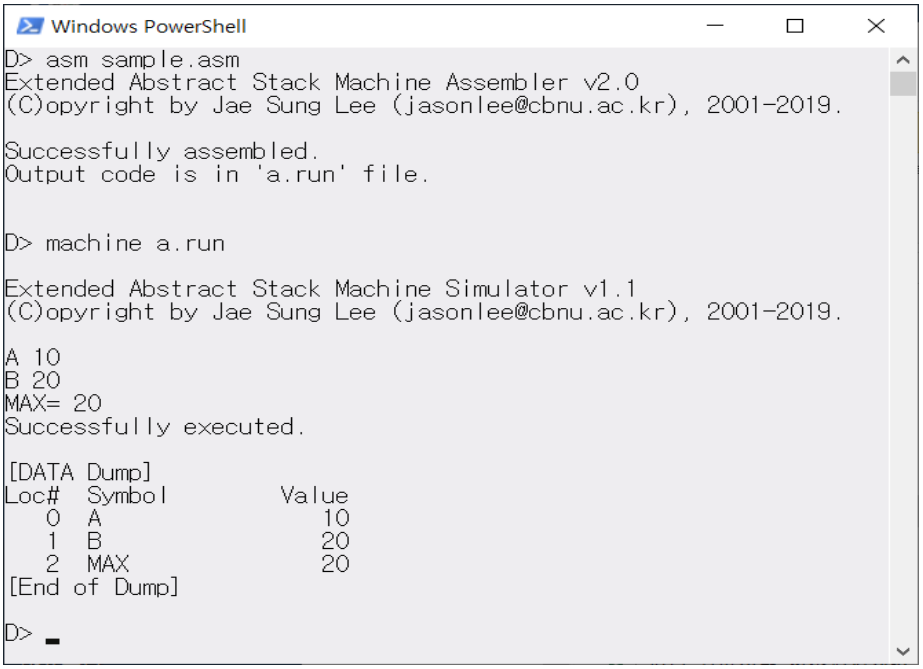

그림 2. 어셈블러 asm과 시뮬레이터 machine의 수행 예

3. ASM 기계어 형식

ASM의 수행파일은 일반 텍스트 파일 형식이며, 각 명령어의 값을 십진수로 인쇄한 것이다. 각 명령어는 단순성을 위해 명령어의 연산자(operator) 코드 필드와 피연산자(operand) 필드로 구성된다. 피연산자가 없는 명령어는 피연산자 필드에 –1의 값이 채워진다. a.run파일의 첫번째 숫자는 명령어의 개수, 두번째 숫자는 데이터 정의(DW)의 개수를 나타내고 그 뒤부터는 각각의 명령어들을 십진수로 표시한 숫자들이다.

명령어	기계어 코드(십육진수)	기계어 코드(십진수)
PUSH	0x0010	16
POP	0x0011	17
COPY	0x0012	18
RVALUE	0x0013	19
LVALUE	0x0014	20
ASSGN	0x0015	21
PLUS	0x0020	32
MINUS	0x0021	33
MUL	0x0022	34
DIV	0x0023	35
GOTO	0x0030	48
GOFALSE	0x0031	49
GOTRUE	0x0032	50
GOPLUS	0x0033	51
GOMINUS	0x0034	52
OUTCH	0x0040	64
OUTNUM	0x0041	65
INCH	0x0042	66
INNUM	0x0043	67
HALT	0x0050	80

4. 샘플 프로그램

A와 B에 두 숫자를 입력으로 받아 그 중 큰 값을 MAX에 저장하고 그 값을 출력하는 프로그램.

* 기계어 첫 줄의 38, 3은 기계어 명령어가 아니고, 명령어의 숫자가 38개, 데이터 저장 위치가 3개임을 나타내는 파일 정보임.

어셈블리어 프로그램	번역된 기계어 결과(a.run파일 내용)
	38 3
PUSH 65	16 65
OUTCH	64 -1
PUSH 32	16 32
OUTCH	64 -1
LVALUE A	20 0
INNUM	67 -1
:=	21 -1
PUSH 66	16 66
OUTCH	64 -1
PUSH 32	16 32
OUTCH	64 -1
LVALUE B	20 1
INNUM	67 -1
:=	21 -1
RVALUE A	19 0
RVALUE B	19 1
-	33 -1
GOMINUS OUT	52 22
LVALUE MAX	20 2
RVALUE A	19 0
:=	21 -1
GOTO OUT1	48 25
LABEL OUT	
LVALUE MAX	20 2
RVALUE B	19 1

:=	21 -1
LABEL OUT1	
PUSH 77	16 77
OUTCH	64 -1
PUSH 65	16 65
OUTCH	64 -1
PUSH 88	16 88
OUTCH	64 -1
PUSH 61	16 61
OUTCH	64 -1
PUSH 32	16 32
OUTCH	64 -1
RVALUE MAX	19 2
OUTNUM	65 -1
HALT	80 -1
DW A	
DW B	
DW MAX	
END	

* 어셈블리어와 기계어를 대응시키기 위해 임의로 빈줄을 삽입하였음

주: 버전 0.91부터는 한글 변수명이 사용가능함