가상스택기계 시뮬레이터 (StackSim)

설명서

(Abstract **Stack** Machine **Sim**ulator Manual)

버전 2.0

2023. 4.

충북대학교 소프트웨어학부

이 재 성

jasonlee@cbnu.ac.kr

* 이 설명서는 소프트웨어학부 “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 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, word 크기 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. StackSim 시뮬레이터의 수행

사용자가 작성한(혹은 컴파일러에 의해 생성된) ASM 어셈블리어 프로그램은 StackSim이 시뮬레이션하여 결과를 생성한다. 이를 간단하게 도식화한 것이 그림 1이다.

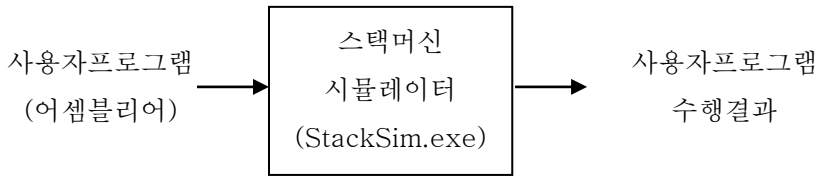

그림 1. 시뮬레이터의 수행

시뮬레이터는 MS-DOS 명령어 프롬프트상에서 수행되며 그 사용 예는 그림 2와 같다. 프롬프트상에서 “StackSim sample.asm”을 입력하여 sample.asm 프로그램을 시뮬레이터가 번역하고 수행하도록 한다. (sample.asm은 두수를 입력으로 받아 변수 A, B에 저장하고, 그 중에서 큰 것을 MAX에 저장한 후, 그 결과를 출력하는 프로그램으로 3장에 소개되어 있다.) 수행과정에서 10과 20을 A, B의 변수에 각각 입력받고, 그 중 큰 값을 MAX값을 출력한 것을 그림2에서 볼 수 있다. 수행이 종료될 때는 데이터 영역에 할당된 모든 변수 값을 출력하도록 하여 프로그램의 수행 결과를 볼 수 있도록 했다.

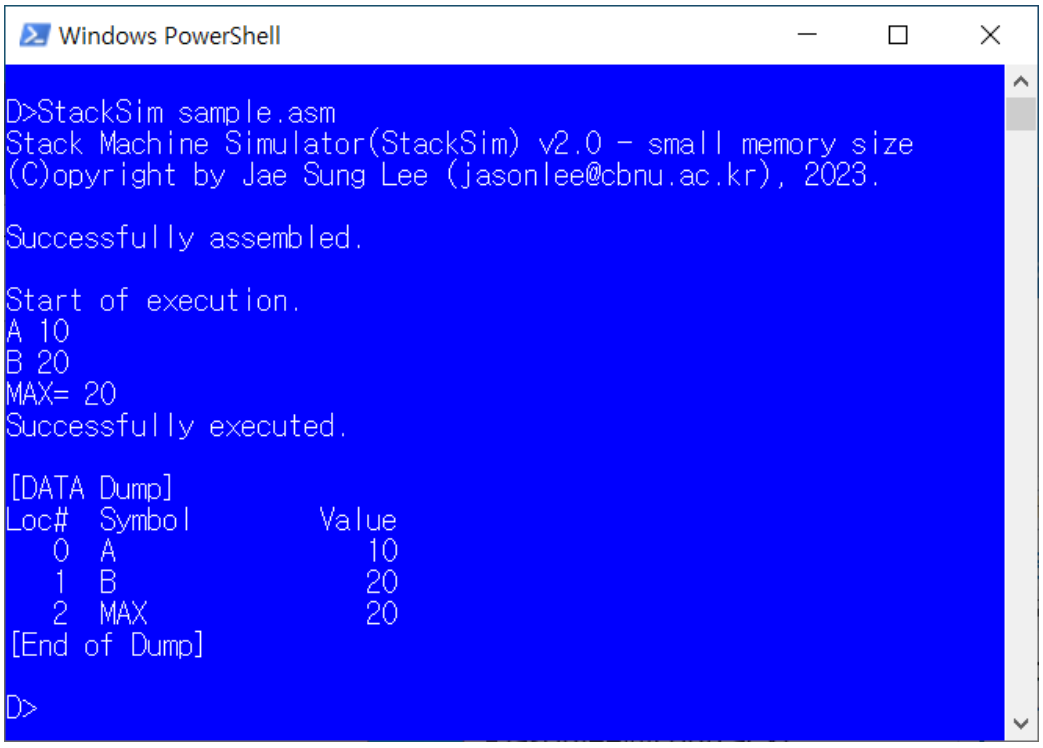

그림 2. 시뮬레이터 StackSim의 수행 예

StackSim v 2.0부터는 메모리 할당을 조절하여 더 큰 프로그램을 수행할 수 있다. 크기는 small, medium, large이며, 명령문의 마지막에 크기의 첫 글자인 s, m, l중의 하나를 넣어 설정한다. 예를 들어 sample.asm을 medium 메모리 크기로 수행하고자 하면 "StackSim sample.asm m" 의 명령어를 입력하면 된다. 아래 표 1은 각 메모리 모델에서의 최대 명령어 수, 심볼(변수명, 레이블명 등) 수, 총 할당 메모리 크기를 보여준다.

모델	최대 명령어 수	최대 심볼수	총 할당 메모리
small	1000	300	14K
medium	2000	600	28K
large	3000	900	42K

표 1. 모델별 최대 명령어 및 심볼수

3. 샘플 프로그램

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

어셈블리어 프로그램	설명
PUSH 65	A를 스택에 푸시
OUTCH	A를 화면에 출력
PUSH 32	빈칸을 스택에 푸시
OUTCH	빈칸을 화면에 출력
LVALUE A	
INNUM	
:=	입력을 받아 A에 배정(저장)
PUSH 66	
OUTCH	B를 화면에 출력
PUSH 32	
OUTCH	빈칸을 화면에 출력
LVALUE B	
INNUM	
:=	입력을 받아 B에 배정(저장)
RVALUE A	

RVALUE B	
-	A-B를 수행
GOMINUS OUT	스택 탑의 값이 음수이면 OUT으로 점프
LVALUE MAX	
RVALUE A	
:=	A값을 MAX에 저장
GOTO OUT1	OUT1으로 점프
LABEL OUT	
LVALUE MAX	
RVALUE B	
:=	B값을 MAX에 저장
LABEL OUT1	
PUSH 77	
OUTCH	화면에 M출력
PUSH 65	
OUTCH	화면에 A 출력
PUSH 88	
OUTCH	화면에 X 출력
PUSH 61	
OUTCH	화면에 = 출력
PUSH 32	
OUTCH	화면에 빈칸 출력
RVALUE MAX	
OUTNUM	화면에 MAX값을 숫자로 출력
HALT	수행 종료
DW A	A 데이터 영역 선언
DW B	B 데이터 영역 선언
DW MAX	MAX 데이터 영역 선언
END	프로그램 끝

주: 한글 변수명을 사용가능함