

Lab04. CodeGeneration

조교 : 박성환

starjara@pusan.ac.kr



실습 목표

- pl0 언어를 활용해 중간언어(IR) 생성.
- 생성된 코드를 출력시켜 확인해 본다.
- 가상 머신을 통해 생성된 코드를 실행시켜 보고 결과를 확인한다.

코드 목록

- lit
 - 값 저장
- opr
 - 연산자를 가지는 코드
- lod
 - 메모리로 부터 읽어들이м
- sto
 - 메모리에 저장
- cal
 - 호출 명령어
- ret
 - 리턴
- ict
 - 스택의 탑을 증가시킴
- jmp
 - 무조건 분기
- jpc
 - 조건부 분기

연산자 목록

- neg, add, mul, sub, div
 - 기본적인 사칙연산자 및 음수를 위한 연산자 지원
- odd, eq, ls, gr, neq, lseq, greq
 - 홀수 판별 및 비교 연산자 지원
- wrt, wrl
 - 화면 출력을 위한 연산자 지원
 - wrt의 경우 일반 출력
 - wrl의 경우 개행 출력

함수 별 역할

- genCodeV
 - Value를 활용해 코드 생성
 - instruction과 value로 이루어짐
- genCodeT
 - Table index활용해 코드 생성
 - instruction과 상대 주소를 가짐
- genCodeO
 - Operator를 활용해 코드 생성
 - instruction과 해당하는 operator 가짐
- genCodeR
 - Return 코드 생성
 - insturction과 level및 address가짐

함수 별 역할

- backpatch
 - 백패치 수행
- nextCode
 - 다음 코드 생성 준비
- listCode
 - 생성된 코드 출력
- execute
 - 생성된 코드 실행

코드 및 연산자 정의

hw_codegen.h

```
1 typedef enum codes{
2     lit, opr, lod, sto, ict, ret, jmp
3 }OpCode;
4
5 typedef enum ops{
6     wrt, wrl
7 }Operator;
```

hw_codegen.c

```
9 typedef struct inst{
10     OpCode opCode;
11     union {
12         RelAddr addr;
13         int value;
14         Operator optr;
15     }u;
16 }Inst;
17
18 static Inst code[MAXCODE];
19 static int cIndex = -1;
```

코드 생성 함수 구현

- hw_codegen.c

```
24 int nextCode()
25 {
26     return cIndex + 1;
27 }
28
29 int genCodeV(OpCode op, int v)
30 {
31     checkMax();
32     code[cIndex].opCode = op;
33     code[cIndex].u.value = v;
34     return cIndex;
35 }
36
37 int genCodeT(OpCode op, int ti)
38 {
39     checkMax();
40     code[cIndex].opCode = op;
41     code[cIndex].u.addr = relAddr(ti);
42     return cIndex;
43 }
44
45 int genCodeO(Operator p)
46 {
47     checkMax();
48     code[cIndex].opCode = opr;
49     code[cIndex].u.optr = p;
50     return cIndex;
51 }
52
```

```
53 int genCodeR()
54 {
55     if(code[cIndex].opCode == ret)
56         return cIndex;
57     checkMax();
58     code[cIndex].opCode = ret;
59     code[cIndex].u.addr.level = bLevel();
60     code[cIndex].u.addr.addr = fPars();
61     return cIndex;
62 }
63
64 void checkMax()
65 {
66     if (++cIndex < MAXCODE)
67         return;
68     yyerror("too many code");
69 }
70
71 void backPatch(int i)
72 {
73     code[i].u.value = cIndex + 1;
74 }
```


yacc 파일 수정

- hw.y
- 코드 블록의 시작 및 종료

```
26 block      : { $<val>$ = genCodeV(jmp, 0); }  
27           declList      { backPatch($<val>1); genCodeV(ict, frameL()); }  
28           statement     { genCodeR(); blockEnd(); }  
29
```

- 대입 및 출력 연산자를 위한 코드 생성

```
67 statement  : /* empty */  
68           | IDENT COLSEQ expression  
69               { genCodeT(sto, searchT($1, varId));}  
70           | BEGINN statement stateList END  
71           | IF condition THEN { }  
72               statement      { }  
73           | WHILE { }  
74               condition DO { }  
75               statement      { }  
76           | RETURN expression { }  
77           | WRITE expression {genCodeO(wrt); }  
78           | WRITELN          { genCodeO(wrl); }  
79
```

yacc 파일 수정

- hw.y
- symbol을 탐색해 값을 읽어오기 위한 코드 생성

```
110
111 factor      : IDENT { int j, k; j = searchT($1, varId); k = kindT(j);
112                  switch(k){
113                  case varId: case parId:
114                      break;
115                  case constId:
116                      genCodeV(lit, val(j));
117                      break;
118                  }
119              }
120      | NUMBER    { }
121
122      | IDENT '(' expList ')' { }
123      | '(' expression ')'
124      ;
125
```

실습 결과

tmp.pl0

```
1 const n=5;
2 var x;
3 begin
4   x:=n;
5   write x;
6   writeln;
7 end.
```

수행 화면

```
code
0: jmp,1
1: ict,3
2: lit,5
3: sto,0,2
4: opr,wrt
5: opr,wrl
6: ret,0,0
start execution
5
```


과제 결과

- ex2.pl0 파일 수행한 결과
- 이외 for, if, while 등의 문법과
여러 연산자를 처리할 수 있어야 함

```
code
0: jmp,43
1: jmp,2
2: ict,5
3: lod,1,-2
4: sto,1,2
5: lod,1,-1
6: sto,1,3
7: lit,0
8: sto,1,4
9: lod,1,3
10: lit,0
11: opr,gr
12: jpc,29
13: lod,1,3
14: opr,odd
15: jpc,20
16: lod,1,4
17: lod,1,2
18: opr,add
19: sto,1,4
20: lit,2
21: lod,1,2
22: opr,mul
23: sto,1,2
24: lod,1,3
25: lit,2
26: opr,div
27: sto,1,3
28: jmp,9
29: lod,1,4
30: ret,1,2
31: jmp,32
32: ict,5
33: lod,1,-2
34: sto,1,2
35: lod,1,-1
36: sto,1,3
37: lod,1,2
38: lod,1,3
39: opr,add
40: sto,1,4
41: lod,1,4
42: ret,1,2
43: ict,4
44: lit,7
45: sto,0,2
46: lit,85
47: sto,0,3
48: lod,0,2
49: opr,wrt
50: lod,0,3
51: opr,wrt
52: opr,wrl
53: opr,wrl
54: ret,0,0
start execution
7 85
```

제출 안내

- 제출 마감 : 12월 05일 23시 59분
- 배포된 파일을 작성하여 Lab04_<학번>.tar.gz 형태로 압축하여 제출할 것
 - 제출 전 make clean 실행하여 소스 코드만 제출 할 것
- Makefile 활용하여 간단하게 compile 가능
 - make <- 실행 파일 생성
 - make clean <- 컴파일 과정에서 생성된 파일 삭제



Q&A