Design and Development of Compiler  
for C- Language  
(설계 프로젝트 수행 결과)

과목명: [CSE4120] 기초 컴파일러 구성

담당교수: 서강대학교 컴퓨터공학과 정 성 원

개발자: 14조 20131547 김한길, 20131612 최대운

개발기간: 2019. 5. 28. ~ 2019. 6. 25.

각 단계별 결과 보 고 서

프로젝트 제목: Design and Development of Compiler for C- Language:

Phase 4: A code generator

제출일: 2019. 6. 25.  
개발자: 14조 20131547 김한길 20131612 최대운

1. 개발 목표  
   - C- 언어로 작성된 코드를 SPIM 머신에서 동작할 수 있는 어셈블리 코드로 만들어준다.
2. 개발 범위 및 내용
   1. 개발 범위  
      C- 언어로 작성된 코드를 어셈블리 코드 파일로 바꿔야 한다.

3번째 프로젝트에 code.c,h cgen.c,h 를 확장하고 변형하여 구현한다.

* 1. 개발 내용
* Input, Output 함수에 대한 구현을 한다.
* 문법에 따라 어셈블리 코드를 생성한다. 자세한 내용은 하단 제작 내용 참조

1. 추진 일정 및 개발 방법
   1. 추진 일정

|  |  |
| --- | --- |
| 일정 | TODO |
| 05. 28.~ 06. 24 | 프로젝트 명세서 파악 및 개발 |
| 06. 24. | 보고서 작성 |
| 06. 25. | 제출 |

* 1. 개발 방법  
     - Code.c/h, Cgen.c/h를 적절히 수정하여 개발한다.

1. 연구 결과

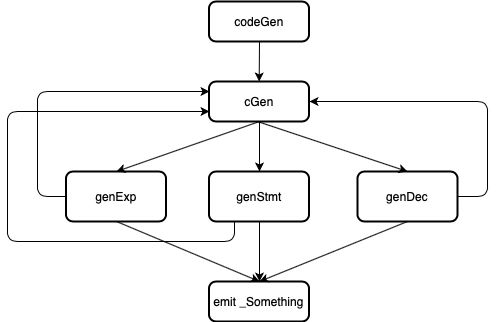
이전 프로젝트에서 abstract syntax tree에 Memory Location이나 타입 등 각 노드에 필요한 데이터들을 저장 시켰다. 이를 이용하여 cgen파일의 codeGen을 통해 어셈블리코드 생성을 시작한다.

* + 1. 분석 내용

codeGen함수는 cGen함수를 호출하고 cGen함수는 각 노트 타입(DeclK, ExpK, StmtK)에 따라 genDec, genExp, genStmt함수를 호출해서 코드를 생성한다.

genDec, genExp, genStmt 함수 내에서 child 노드의 유무에 따라 cGen함수를 child노드를 파라미터로 호출하면서 위 코드 생성을 반복한다.

Input, output 함수는 사용자가 정의하지 않았지만 사용된다.



<코드 생성 구성도>

* + 1. 제작 내용

Input, output 함수는 시스템콜처럼 사용자가 정의하지 않았지만 에러를 발생시키지 않고 정상적으로 사용된다. 이번 프로젝트에서는 input, output 함수 호출을 Grammar로 만들어서 처리했다.



<Input, Output 함수 호출 Grammar>

또한 Input, output에 해당하는 어셈블리 코드는 항상 고정적으로 아래와 같이 정의를 해두었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<Read Int, Write Int, Write String 함수의 어셈블리 코드>

모든 연산의 결과나 리턴 값은 $v0에 저장된다.

함수 호출에 사용되는 Parameter의 값은 $a0~$a3에 저장되어서 넘어간다.

Label Number는 \_getLabelNumber 함수를 통해 할당 받아 사용하며 중복된 라벨이 나오지 않게 한다.

Fp ->

Globlas

Args, 4개

Control Link

Return Address

Free space

sp ->

24bytes

실제로 만들게 되는 스택 구조는 위와 같다. 코드로 구현된 결과는 다음과 같다. Args 부분은 tree->child[0]을 cGen하면서 코드가 생성된다.



Gencode 부분을 declaration , expression , statement 세 부분으로 나눠서 분류 후 구성하였다.

먼저 declaration 부분을 먼저 작성하였는데 simple 와 array 부분은 이전 프로젝트에서 이미 구분을 해놓은 것을 nodekind 부분을 거쳐 입력받으면 sp부분을 그 크기만큼 아래로 내려주는 것으로 간단하게 구성하였다.

Parameter 부분은 args부분은 미리 저장해놓은 파라미터개수에 따라 fp를 기점으로 이미 function에서 파라미터 부분을 16을 크기로 잡고 공간을 남겨두었기때문에 그장소에 a레지스터들을 사용하여 값을 저장하는 것으로 구현하였다.

Function이 가장 문제였는데 펑션 같은 경우는 먼저 리턴어드레스와 컨트롤링크의 공간, 파라미터의 공간을 마련하기위하여 먼저 sp를 24크기만큼 내려놓고 sp를 기점으로 리턴어드레스와 컨트롤링크값을 저장한뒤에 sp에 4크기만큼 더한곳에 fp를 두는것으로 activation record를 구성하였다.

이후 parameter개수를 측정하고 cgen을 다시호출하여 파라미터에 대한 공간을 하나하나 저장하였고 다시 cgen을 호출하여 compound statement부분을 해결하였다 이후 여기서 특이한 점은 function을 구성하는부분에서 다시 function이 끝나는 경우까지 생각하여 과거의 스택포인터에 저장해뒀던 리턴어드레스와 컨트롤링크를 레지스터에 다시 돌려놓고 sp를 24만큼 다시올리고 리턴어드레스로 점프해서 나중 리턴부분에서는 간단하게 값만 저장하는 방식으로 구현하였다.

스테이트먼트는 5개 부분으로 나눠진다. If,while,compound,return,assing 다섯개로 나눠진다.

if문같은경우는 getlabelnumber 이라는 함수를 따로만들어 static int를 선언하고 label이름을 계속 만들어주는 방법으로 만들었고 if문같은경우는 if then 과 if then else 로 나눠질 수 있어 child[2] 가 존재하면 else가 존재하는 것으로 간주하여 두개의 label을 통해 pcode에서 배웠던것처럼 구현하였고 if then 구문같은경우는 그냥 밑에 label1을 만들어서 밑으로 바로 점프하는 방식으로 만들었다.

While문 같은 경우도 비슷한데 while문은 맨 위에 label1을 하나 만들어두고 반복문이 끝나는 곳에 다시 label2를 만들어 안의 compound statement를 수행하는 것으로 구성하였다.

Assign문 같은 경우는 left hand side에서는 주소를 right hand side에서는 값을 받아와야 하기 때문에 재귀호출을 사용하여 child[0]에 대해서는 주소를 v0레지스터에 저장하여 불러들여오고 child[1]에 대해서는 주소를 v1레지스터에 저장하는 것으로 구현하였다. Assign 뿐 아니라 대부분의 값을 리턴하는 경우는 v0에 값을 v1에는 주소를 넣는식으로 구성하였다.

Return 문 같은 경우는 원래는 function이 끝나면서 돌아가야 하지만functiondeclaration에서 return label을 만들어 두면서 함수 마지막으로 점프시키고 이미 declaration에서 sp와 fp를 돌리는 작업을 해두었기에 따로 처리는 하지않았어도 됐다.

Compound 문 같은 경우 안에서 선언된 local varaiable를 재귀함수를 통해 크기를 측정하고 그 측정된 크기만큼 공간을 확보하는 것으로 구성했다.

Exp 같은경우는 크게 operator ,funccall,const,Id 네가지로 크게구성된다.

Op문 같은 경우 왼쪽의 인자와 오른쪽의 인자의 값을 v0와 tq에 각각 저장하고 sp를 확보한 후에 operator에 맞는 연산을 switch문으로 구성한 뒤 결과 값을 v0에 다시 넣는 방법으로 구성하였다.

Const 문은 lif라는 명령어로 tree의 값을 넣어주는 방식으로 구성하였다.

Id문 같은 경우 global과 global이 아닐 때 그리고 배열일 때와 배열이 아닐 때 네가지를 고려해야했다. 우선 배열이 아닐 때는 간단하다. 값과 주소를 v0와 v1에 로드하는 방법으로 간단하게 구성되는데 global인 경우는 gp를 기점으로 lw와 la를 구성하고 global이 아닐 경우는 fp를 기점으로 lw 와 la 를 통해 구현하였다. 그리고 배열일 경우에는 우선 먼저 a[5]일 경우로 생각한다면 a의 위치와 값을 먼저 저장하였고 재귀호출로 다시 그안의 5의 위치와 값을 추적하였다. 이후 5의 값을 4를 곱해주면서 위치를 a에 더해주고 그 더해준 값과 위치를 v0와 v1에 저장하는 방식으로 구현하였고 위에서와 같이 global일 경우는 gp를 기점으로 global이 아닐경우는 fp를 기점으로 주소를 지정했다.

functioncall 같은 경우는 파라미터를 재귀호출을 통해서 v0에 값을 넣어준 후 그 넣어준 값을 다시 a0부터 시작해서 레지스터에 추가시켜준 후 점프를 통해 function으로 점프하는 것으로 구현하였다.

* + 1. 시험 내용

실행하는 방법은 다음과 같다.

* + - 1. Make
      2. ./project4\_14 [testFile].c
      3. spim -file [testFile].tm

제시된 입력 결과에 대한 결과는 아래와 같다.

1. 1번 예시

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 2번 예시

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 임의의 테스트 케이스

모니터, 화면, 검은색이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

Base와 지수를 입력받아 Power를 구해주는 함수와 두 수를 입력받아 공배수를 구하는 함수 gcd를 두가지 방법으로 구현한 코드이다. 실행 결과는 아래와 같다.



1. 임의의 테스트 케이스2
2. ㅇ
3. 기타
   1. 자유 기술

지난 프로젝트3에서 구현한 Symtab이 트리 구조로 되어 있어서 이것을 리니어하게 바꿨다. Symtab출력을 위해 그렇게 구현했었다. 바뀐점은 Phase1이후 Symtab이 가지는 정보들을 AST 노드에 저장하는 방식으로 바꿨다. 주소 값을 저장하고 꺼내오는 과정에서 MIPS instruction이 이유없이 정상작동 하지 않아서 애먹었다.

* 1. 팀원의 역할 분담

김한길(50%), 최대운(50%): symtab의 구조를 다시 잡아 Project3 발표에서 나왔던 문제점들을 보완하고 SPIM 매뉴얼과 MIPS 매뉴얼을 참고 해가면서 개발하고 보고서도 같이 작성했습니다.