**컴파일러 프로젝트 보고서 4 Code Generation**

신소재공학부 2012006276 진현수

1) 컴파일 방법 및 환경

* Linux – Fedora 19
* gcc 컴파일 , yacc(bison)

2) Code generation구현 과정 및 주요 소스코드 설명

* **Stack 기반의 메모리 환경을 가진다.**
* Function Call이 일어나면 local variable, parameters, return address, function address가 차례로 쌓인다.
* 즉, function pointer를 fp라고 할 때
  + fp-1 : return address
  + fp-2 : 1st parameter
  + fp-3 : 2nd parameter
  + ...
  + fp-n : last parameter
  + fp-n-1 : 1st local variable
  + fp-n-2 : 2nd local variable
  + ...
* **main.c 수정**

#define NO\_CODE FALSE

int EchoSource = TRUE;

int TraceScan = FALSE;

int TraceParse = FALSE;

int TraceAnalyze = FALSE;

int TraceCode = TRUE;

* **analyze.c , symtab.c ,symtab.h 수정**

location : scope내의 변수의 위치

function pointer = fp

fp-2-loc에 변수 저장

* **cgen.c 수정**

syntaxTree를 순회하며 코드를 생성. 각 tree의 타입별로 genStmt, genExp, genDecl, genParam 이라는 함수를 각각 생성.

1. **genStmt**

* If, While : tiny의 코드를 그대로 사용
* Compound : Scope를 추가해 줌
* Return : return address(fp-1)로 expression 값을 내보냄

1. **getExp**

* OP, Const : tiny의 코드를 그대로 사용
* Id : Scope를 고려하여 변수의 주소 값을 계산
* Assign : lhs(lvalue)는 주소를 가져오고, rhs(rvalue)는 값을 가져오도록 구현
* Call : 함수콜이 일어날때 Call Stack에 파라메터, return value가 쌓이는 순서에 주의하여 구현

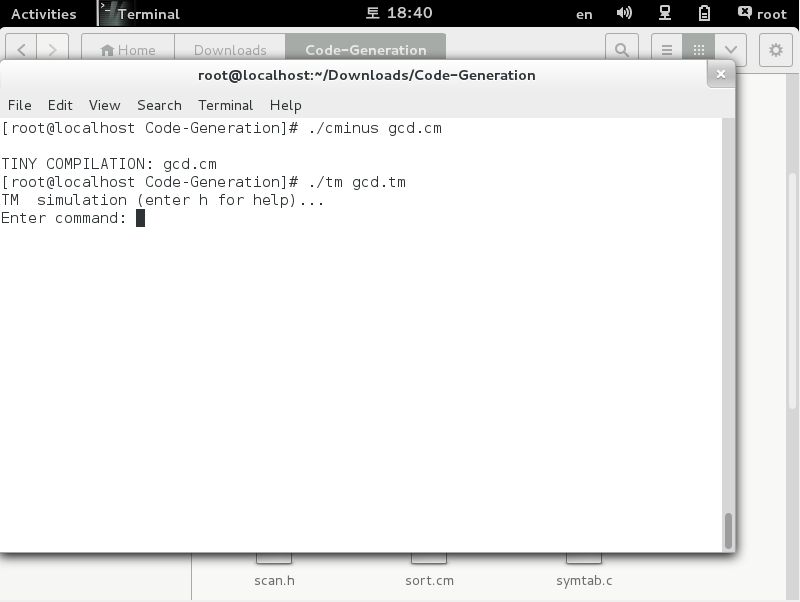
1. **getDecl & getParam**

* Func, Var, ArrVar : 이전단계에서 타입과 스코프는 검사하였으므로 할당될 메모리 크기만 계산한다. c-minus에서는 모든 변수타입이 1의 크기를 가지도록 설정하였다. Array의 경우 Array의 크기만큼 할당

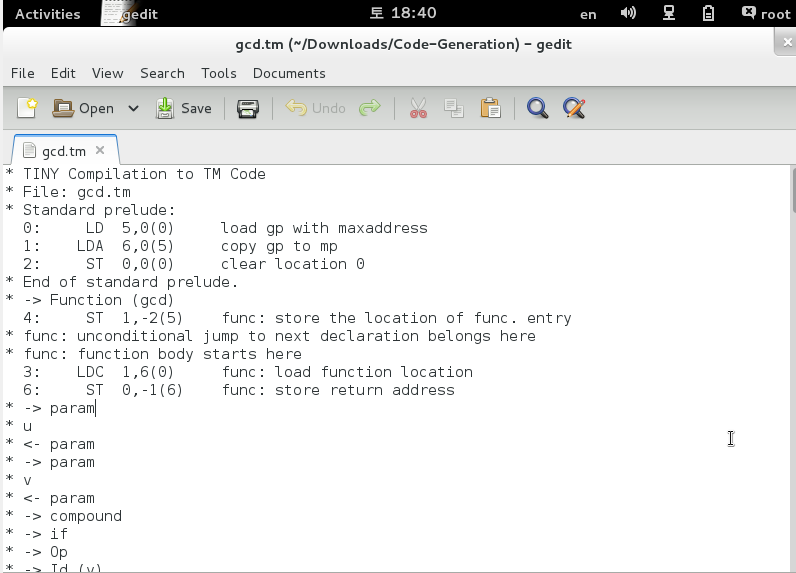
1. **getMainCall**

* syntaxTree를 돌아 code를 생성한 후, main를 호출하는 코드를 생성

3) 예시 및 결과 화면



cminus로 gcd.cm파일을 gcd.tm으로 코드를 tiny machine에 맞는 코드를 생성했음을 확인 할 수 있다.



위와 같은 gcd.tm파일이 형성되었고 정상 작동함을 확인할 수 있다.