1. 컴파일러 개요

충북대학교

이 재 성

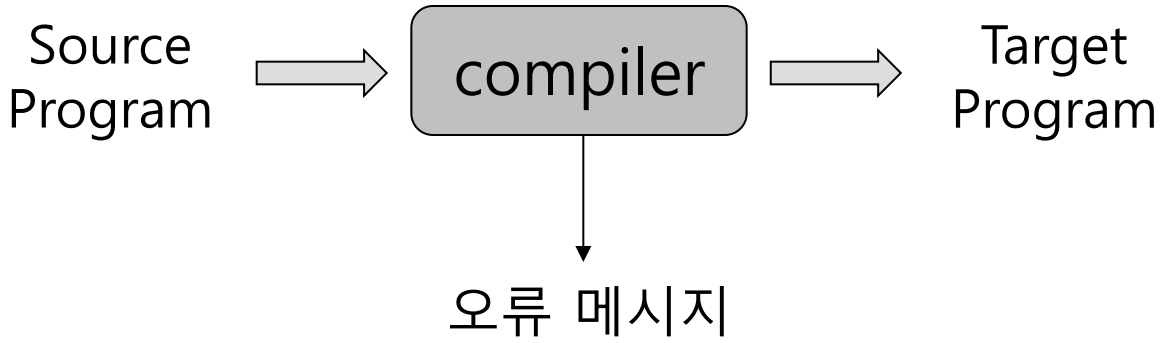
LaKE LAB

학습내용

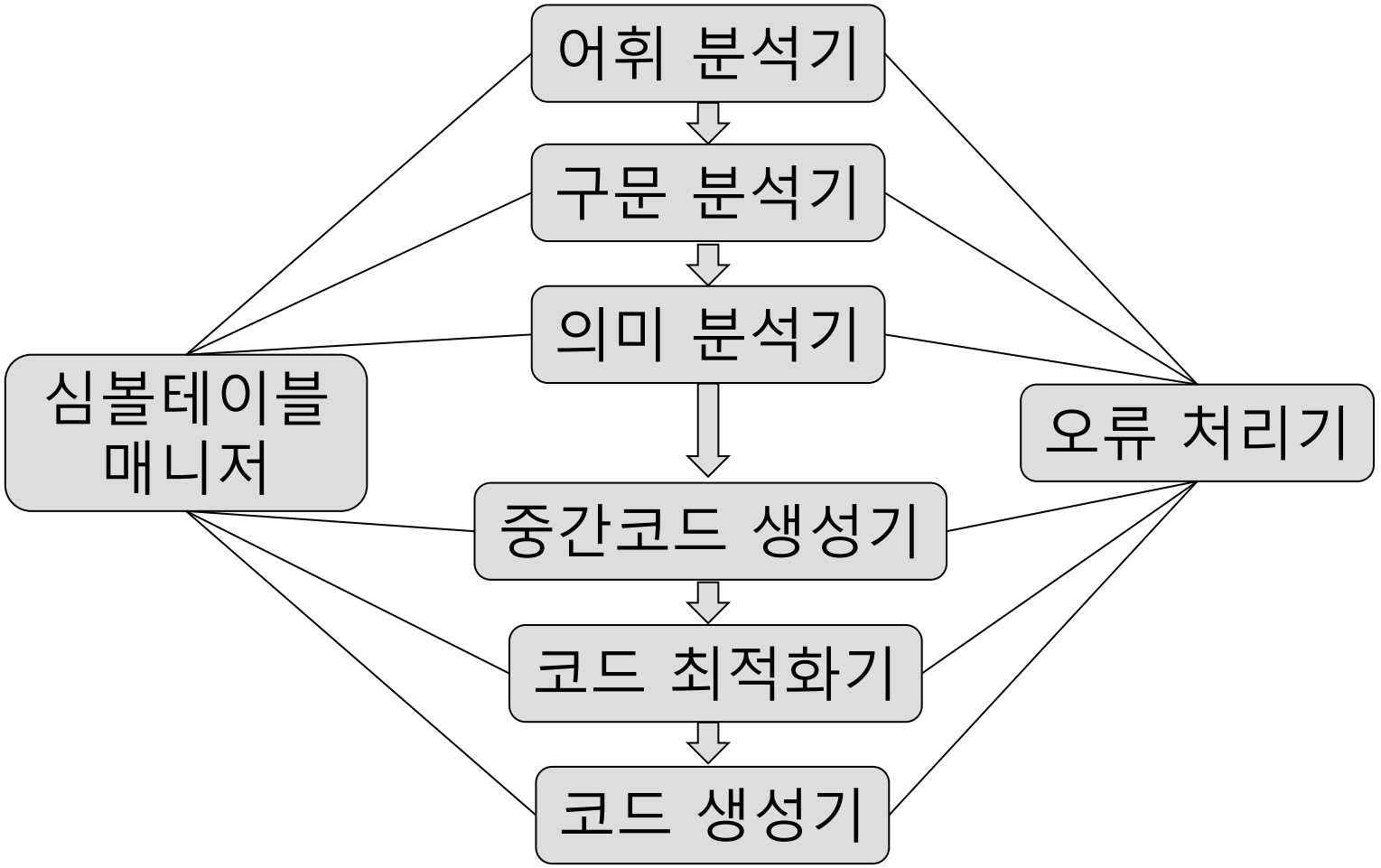
- 컴파일러 구축 단계에 대한 개요
- 컴파일 기법을 활용한 프로그램들의 예

컴파일러란?

- 정의
  - 어떤 언어(source language)로 쓰여진 프로그램을 입력으로 받아들여 대등한 다른 언어(target language)의 프로그램으로 바꿔주는 프로그램
- 예
  - C 컴파일러, Fortran 컴파일러, 어셈블러 등



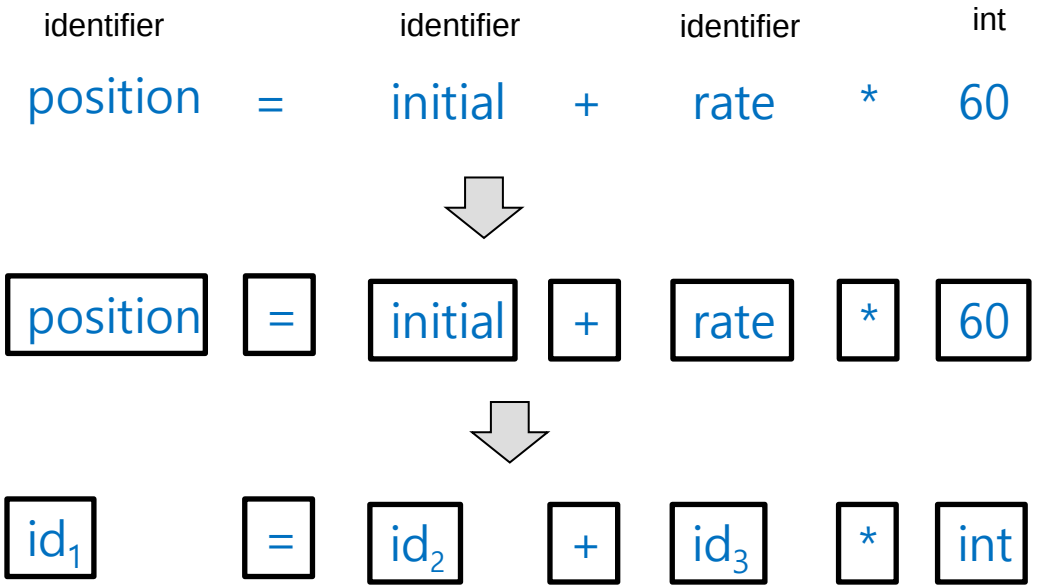
컴파일 단계



어휘 분석

어휘 분석(Lexical analysis)

- (선형 분석, 스캐닝)
- 문장을 토큰 단위로 분리
- 토큰 타입 분류



심볼테이블

idx	name	token
1	position	id
2	initial	id
3	rate	id

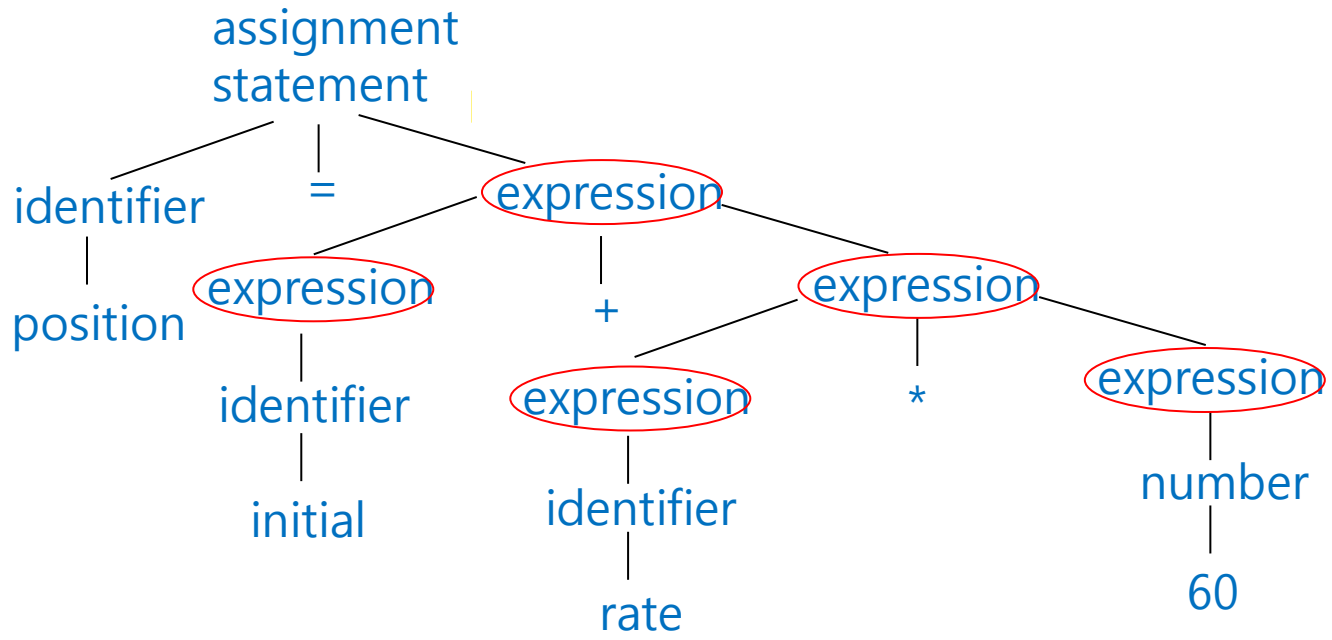
프로시저

4 60 int

구문 분석

구문분석(Syntax analysis)

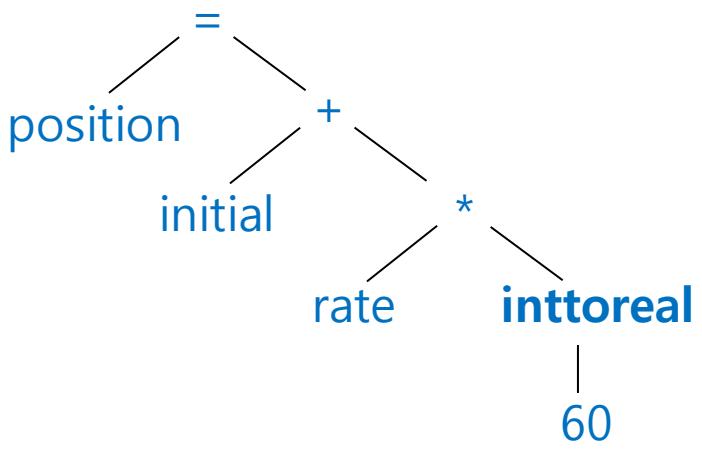
- (계층적 분석, 파싱)
- 프로그램 토큰들을 문법적 형태인 파스 트리 형태로 표현
- 반복적 규칙에 의해 표시



의미 분석

의미 분석(Semantic analysis)

- 소스프로그램의 의미 오류 검사
- 타입 조사 및 정보 정리
- 예: 배열의 첨자에 실수 사용 금지
- 예: 실수 변수 연산에서 정수형 상수를 실수형으로 형변환



중간코드 생성

중간코드

- 추상화된 기계에 대한 프로그램
- 어셈블리어 형태

기계어 형식(피연산자 주소)

0: add	s[0] = s[0]+s[1]
1: add a	s[0] = s[0]+a
2: add a, b	b = a+b
3: add a, b, c	c = a+b

중간코드의 특징

- 중간코드는 생성이 편리해야 함
- 목적프로그램으로의 변환이 쉬워야 함
- 세 개의 주소를 갖는 중간 형태
  - 각 기계어로의 변환 용이
- 계산된 결과를 담고 있을 임시 장소의 이름을 생성

목적 코드 생성

코드 최적화

- 중간 코드를 향상시켜 더 빠른 실행코드 생성
- 컴파일시 최적화 시간을 많이 소요- 최적화 컴파일러

코드 생성

- 재배치 가능한 기계코드 혹은 어셈블리어 코드로 생성
- 변수들을 레지스터에 할당

심볼테이블 관리 및 오류 처리

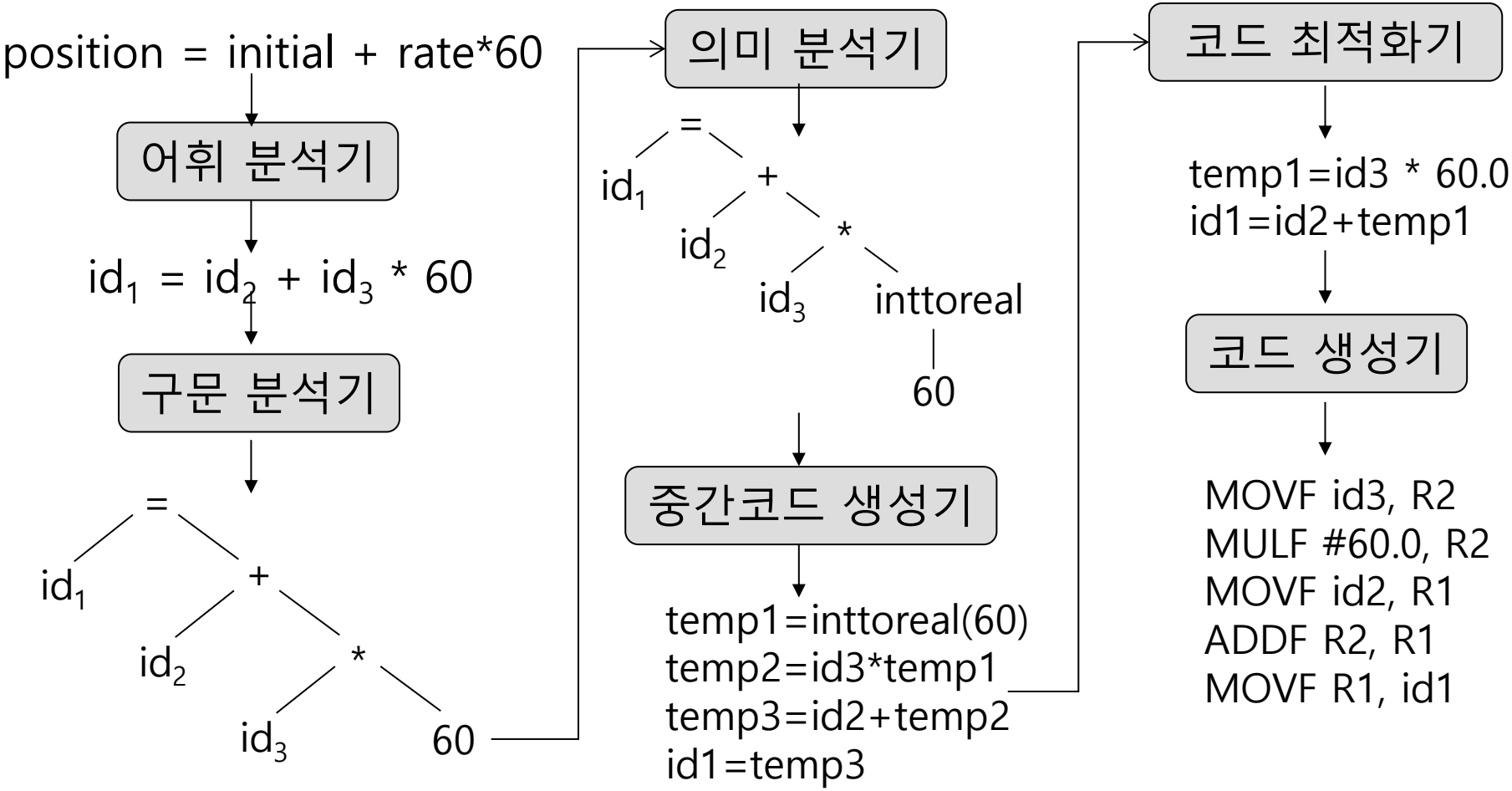
심볼 테이블 운용

- 각 식별자에 대한 레코드 저장
- 식별자의 저장장소, 형(type), 영향 범위 기록
- 프로시저 이름의 경우, 매개변수 정보 포함
- 각 분석 단계별로 얻어진 정보가 다음 단계에서 사용

오류 발견과 내용 출력

- 오류 회복 기능 필요
- 각 단계별 오류 형태의 내용 출력
  - 어휘분석 단계: 언어형식과 다른 토큰 구분
  - 구문분석: 언어의 구조적 형식에 위배되는 오류
  - 의미분석: 연산의 의미가 부적절한 것
    (예: 배열+프로시저)

문장 번역 예


position = initial + rate*60
어휘 분석기
id1 = id2 + id3 * 60
구문 분석기
=
id1
+
id2
*
id3
60
의미 분석기
=
id1
+
id2
*
id3
inttoreal
60
중간코드 생성기
temp1=inttoreal(60)
temp2=id3*temp1
temp3=id2+temp2
id1=temp3
코드 최적화기
temp1=id3 * 60.0
id1=id2+temp1
코드 생성기
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1

컴파일 기법을 이용한 프로그램들 - 1

구조 편집기

- 프로그램언어의 구조에 맞게 입력되었나 검증
- 키워드 자동 발생

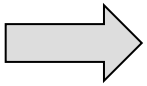
point->|

x
y

pretty 프린터

- 프로그램의 블럭 포함 관계을 분석하여 들여 쓰기를 하고, 괄호의 짝을 쉽게 볼 수 있도록 출력 (VC++의 ALT+F8)

```
while(a>10) {
if (b<1)
c = a;
else{
c =b;
}
}
```



```
while(a>10) {
        if (b<1)
                c = a;
        else{
                c =b;
        }
}
```

컴파일 기법을 이용한 프로그램들 - 2

정적 검사기

- 수행하기 전에 프로그램을 읽어서 분석
- 수행 경로에서 빠진 부분 검사
- 미정의 문자 발견
- 타입오류

인터프리터

- 소스프로그램 분석 후 바로 수행(목적 프로그램을 생성하지 않음)
- 데이터 값에 따라 다르게 수행되는 경우 사용(예: APL)

컴파일 기법을 이용한 프로그램들 - 3

텍스트 포멧터

- 각 문장 혹은 문단의 출력 형식을 정의하고 처리
- Tex, LaTex

.ce 컴파일러

인터넷 브라우저

- HTML 형식을 인식하여 화면에 표시하고 필요한 루틴도 수행

<body><h1> 컴파일러
</h1></body>

컴파일 기법을 이용한 프로그램들 - 4

- XML 파서
  - XML 형식의 문서를 필요한 요소 별로 구분할 수 있도록 기능 제공

- 질의 인터프리터
  - 데이터베이스의 질의어(query)를 해석하여 DBMS에서 처리할 수 있도록 기본 명령어들로 변환시켜줌.

```
select EmplId
from ASSIGN, JOB
where ASSIGN.JobId = JOB.JobId
```

컴파일러와 관련된 프로그램 - 1

- 전처리기
  - 상수 정의
  - 파일 포함 처리 등
- 어셈블러
  - 컴파일 결과가 어셈블리어로 나올 경우, 기계어로 변환

컴파일러와 관련된 프로그램 - 2

로더/링커

- 링커: 오브젝트 코드들을 연결하여 실행 파일 형태로 변환
- 로더: 실행 파일을 운영체제에 넘겨 수행 가능토록 처리

라이브러리 루틴

- 공통으로 많이 쓰는 부프로그램들의 모임
- 프로그램 작성시 가져다 쓸 수 있음

참고 문헌

[1] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, “Compilers – Principles, Techniques, and Tools,” Bell Telephone Laboratories, Incorporated, 1986.

[2] 오세만, “컴파일러 입문”, 정익사, 2004.