

Computer Programming

Lecture 5

이윤진
서울대학교
2007.1.4.

Slide Credits

- 엄현상 교수님
 - 서울대학교 컴퓨터공학부
 - Computer Programming, 2007 봄학기

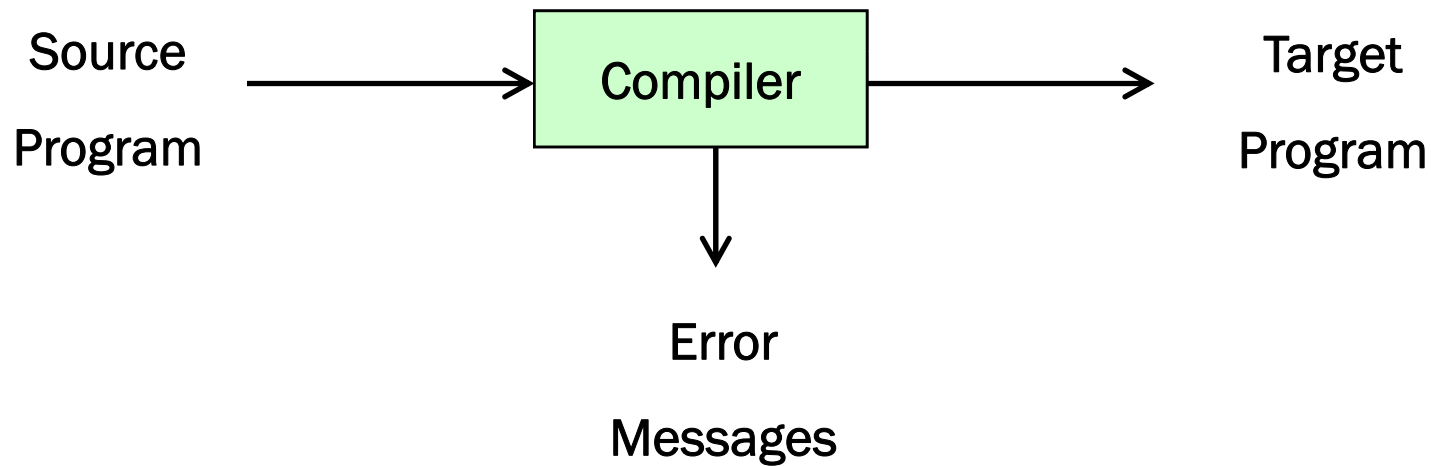
Compiler and Linker (1)

순서

- Compiler and Linker
 - Compiler 기본
 - Compiler Context
 - Compiler Operation
 - Statement Translation Example
 - Assembling and Link-Editing
- Q&A

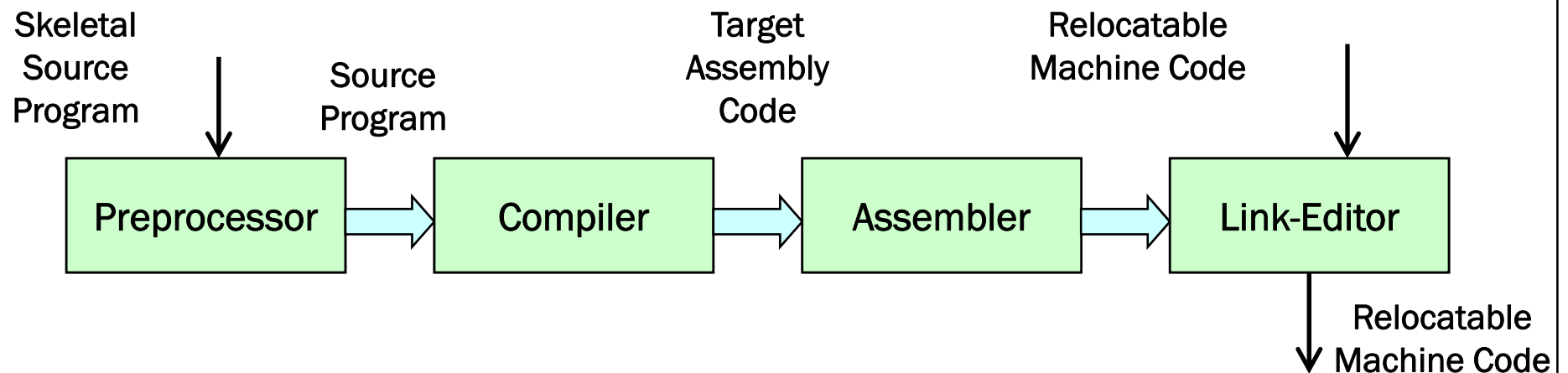
Compiler 기본

- Compiler
 - Program That Reads a Program Written in One Language and Translates It into an Equivalent Program in another Language



Compiler Context

- Before Compilation
 - Preprocessing
 - Include files
 - Expand macros
- After Compilation
 - Assembling
 - Link-Editing

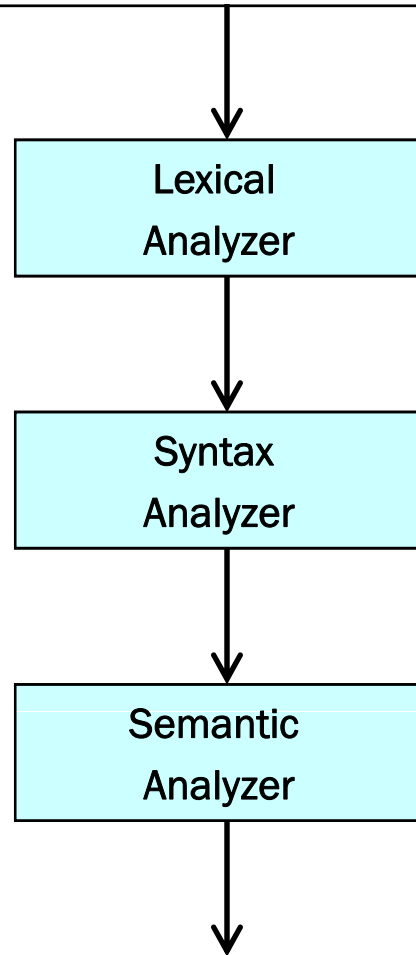


Compiler Operation

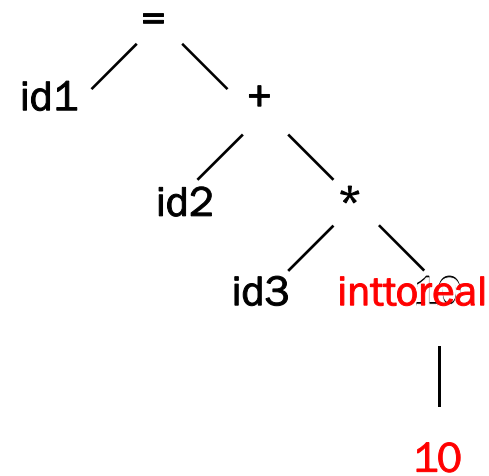
- Main Activities: 6 Phases
 - Lexical Analyzer
 - Syntax Analyzer
 - Semantic Analyzer
 - Intermediate Code Generator
 - Code Optimizer
 - Code Generator
- Other Activities
 - Symbol-Table Manager
 - Error Handler

Statement Translation Example

result = base + rate * 10



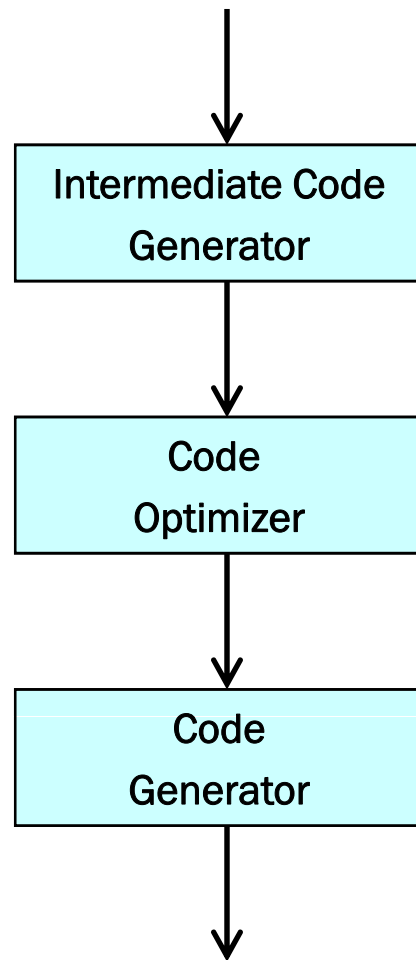
id1 = id2 + id3 * 10



Symbol Table

result	...
base	...
rate	...

Statement Translation Example (계속)



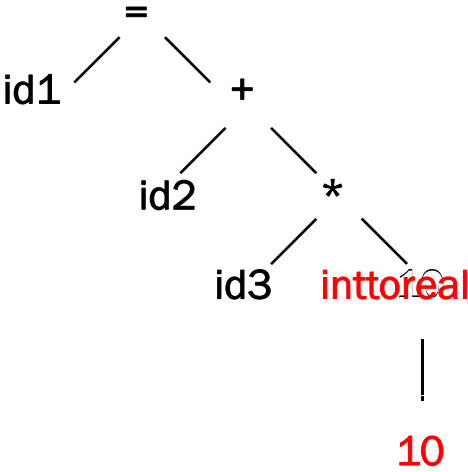
```
t1 = inttoreal(10)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

```
t1 = id3 * 10.0
id1 = id2 + t1
```

```
MOVF id3, R2
MULF #10.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```

Symbol Table

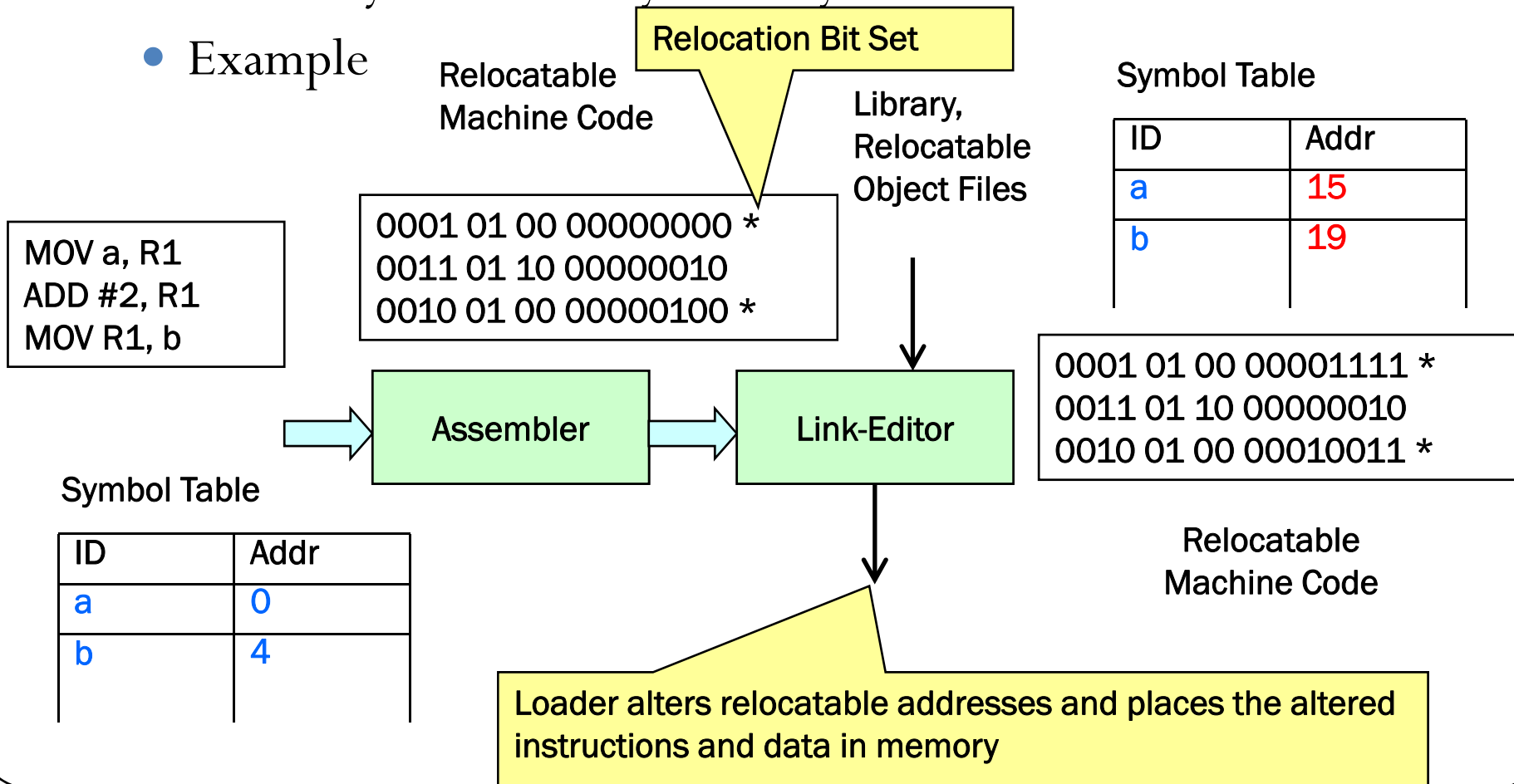
result	...
base	...
rate	...



Assembling and Link-Editing

- Relocatable Machine Code
 - Possibly Loaded at any Memory Location

- Example

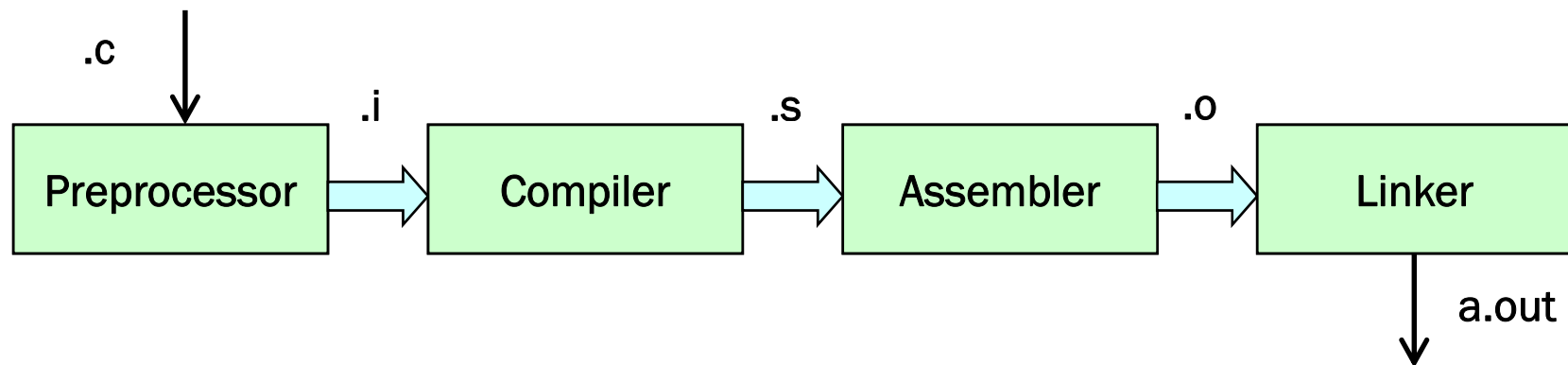


순서

- C Compiler and Linker
 - gcc 기본
 - gcc Basic Options
 - gcc Other Options
 - Assembler and Linker
 - Static and Shared Libraries
- Q&A

gcc 기본

- gcc: FSF's compiler
 - Compile the Source Code, Producing the Assembly Language Code
 - Assemble the Resulting Code
 - Invoke a Linker (Link-Editor), Producing an Executable
- Compilation Stages



gcc Basic Options

- -v (verbose)
 - Print Compiler's Version Number
 - Print How Each Pass Is Executed

```
martini:~$ gcc -v test.c
```

```
Reading specs from /usr/lib/gcc-lib/i386-linux/3.0.4/specs
```

```
... (Omitted)
```

```
gcc version 3.0.4
```

```
...
```

```
/usr/lib/gcc-lib/i386-linux/3.0.4/cc1 ... test.c ... -o /tmp/cccvZhCk.s
```

```
GNU CPP version 3.0.4 (cpplib) (i386 Linux/ELF)
```

```
GNU C version 3.0.4 (i386-linux)
```

```
    compiled by GNU C version 3.0.4.
```

```
...
```

```
as ... -o /tmp/ccp7YrWq.o /tmp/cccvZhCk.s
```

```
GNU assembler version 2.12.90.0.1 (i386-linux) using BFD version 2.12.90.0.1 20020307
```

```
Debian/GNU Linux
```

```
/usr/lib/gcc-lib/i386-linux/3.0.4/collect2 ... /tmp/ccp7YrWq.o ...
```

specify various things such as the linker name

Executable and Linking Format: binary format

gcc Basic Options (계속)

- -o
 - Name the Executable
- -c
 - Compile but Do Not Link

```
martini:~$ gcc -c backup1.c
```

```
martini:~$ gcc -c backup2.c
```

```
martini:~$ gcc -o backup backup1.o backup2.o
```

produce backup1.o

produce backup2.o

combine all the object files to produce the executable

gcc Basic Options (계속)

- -D
 - Set the Value of a Symbol
- -I (Capital i)
 - Include Files in a Non-Standard Directory

```
#define INFO_FILE "infofile"
```

```
martini:~$ gcc -c -DINFO_FILE=\ "infofile\" backup1.c  
martini:~$ gcc -c -DUSE_ODIR backup2.c  
martini:~$ gcc -c -I../include backup3.c
```

indicate where to find the header files

```
#define USE_ODIR  
-----  
#ifdef USE_ODIR  
...  
#else  
...  
#endif
```

gcc Basic Options (계속)

- -l (Small L)
 - Search for **libname.a** in the Directories `/lib` and `/usr/lib`, and Link the Program to the Library
- -L
 - Look in a Particular Directory for Libraries and Link the Program to the Libraries

`/usr/lib/libm.so or /usr/lib/libm.a`

```
martini:~$ gcc -o solve main.o -lm
martini:~$ gcc -o solve main.o -L../lib -lusermath -lm
martini:~$ gcc -L../lib -luserlib solve.c
```

look for libraries first
in `../lib`

can't resolve any function
references needed for linking to
the user (static) library

gcc Other Options

- -w (None), -W, -Wall
 - Produce Warning Messages
- -O (= -O1), -O0 (None), -O1, -O2
 - Determine the Optimization Level
- -p, -pg, -g
 - Profiling and Debugging Options
- -Wa, *option-list*
 - Pass the *option-list* to the Assembler
- -Wl, *option-list*
 - Pass the *option-list* to the Linker
- -static
 - Link only to Static Library
- -shared (default)
 - Use Shared Libraries, if Available, Rather than Static Ones

Assembler and Linker

- Assembler
 - Take a Program Written in an Assembly Language and Produce an Object Module
 - *as list-of-options list-of-source-files*
 - -ah, -al, -as
- Linker (Link-Editor)
 - Combine Several Object Modules and Libraries into a Single Executable
 - *ld list-of-options list-of-source-files*
 - -s, -x, -n

display the symbol table of *a.out* numerically

```
martini:~$ nm -n
```

Static vs Shared (Dynamic) Libraries

- Static Libraries
 - Collections of Object Files That Are Linked into the Program during the Linking Phase of Compilation, and Are Not Relevant during Runtime

```
martini:~$ ar rc libusermath.a usermath.o
```

- Shared Libraries
 - Collections of Object Files That Are Linked into the Program at Runtime

```
martini:~$ gcc -shared -o libusermath.so usermath.o
```