Computer Programming
Compiler and Linker
10th Lecture

엄현상 (Eom, Hyeonsang)
컴퓨터공학부
서울대학교

©Copyrights 2010 Eom, Hyeonsang All Rights Reserved
순서

- Shell Programming 보충
  - Variable Naming

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

- Q&A
Shell Programming 보충

Bash Variable Naming
- letter -> A | B | ... | Z | a | b | ... | z
- digit -> 0 | 1 | ... | 9
- u_s -> _
- var_name -> (letter | u_s)(letter | digit | u_s)*

```
a=1
echo $aa
```

To show the value 1, this ‘a’ should be ~, !, @, #, %, ^, *, -, +, =, {, [, }, ], :, \, ,, ,?, /, or [space]

Result
```
martini:~$ var_naming.sh
```
# Shell Programming 보충 (계속)

**Character Echoing**

- **Shown as Expected If There Is Space before and after the Character (Refer to Handout 7)**
- **Possibly Shown Verbatim If There Is an Adjacent Character**

**Some Exceptions**

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>echo ~;</td>
</tr>
<tr>
<td></td>
<td>echo ~/</td>
</tr>
<tr>
<td></td>
<td>echo ~+</td>
</tr>
<tr>
<td></td>
<td>echo ~:</td>
</tr>
<tr>
<td></td>
<td>echo ~\</td>
</tr>
<tr>
<td>#</td>
<td>echo /*.., #~@</td>
</tr>
<tr>
<td>*</td>
<td>echo *~</td>
</tr>
</tbody>
</table>
Compiler 기분

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

Lexical Analyzer

Syntax Analyzer

Semantic Analyzer

Symbol Table

<table>
<thead>
<tr>
<th></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td></td>
</tr>
<tr>
<td>base</td>
<td></td>
</tr>
<tr>
<td>rate</td>
<td></td>
</tr>
</tbody>
</table>

id1 = id2 + id3 * 10

```
inttoreal = id1 + id2 * id3
```
Statement Translation Example (계속)

Intermediate Code Generator

Code Optimizer

Code Generator

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

result
base
rate

Symbol Table

= +
id1

id2

id3

inttoreal

10

MOVF id3, R2
MULF #10.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
Assembling and Link-Editing

Relocatable Machine Code
- Possibly Loaded at any Memory Location

Example

MOV a, R1
ADD #2, R1
MOV R1, b

Assembler

Library, Relocatable Object Files

Symbol Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>15</td>
</tr>
<tr>
<td>b</td>
<td>19</td>
</tr>
</tbody>
</table>

Relocatable Machine Code

0001 01 00 00000000 *
0011 01 10 00000010
0010 01 00 00000100 *
0001 01 00 00001111 *
0011 01 10 00000010
0010 01 00 00010011 *
0001 01 00 00001111 *
0011 01 10 00000010
0010 01 00 00010011 *

Relocation Bit Set
Loader alters relocatable addresses and places the altered instructions and data in memory

Relocatable Machine Code

0001 01 00 00000000 *
0011 01 10 00000010
0010 01 00 00000100 *
0001 01 00 00001111 *
0011 01 10 00000010
0010 01 00 00010011 *

Link-Editor