Today's topics

Language Translation
Generating Code

Upcoming
Inheritance
(back to Chapter 5)

Reading
Great Ideas, Chapters 9

Adding Semantics

♦ Need to add semantic components to our rules
☐ For every syntax rule, we will add a semantic rule
☐ This will show the assembler code generated
☐ The code, as interpreted by the machine will provide the meaning

♦ Revise R1
Syntax Rule Semantic Rule
R1: \(<n>_j \rightarrow w\) \(M(<n>_j) = w\)
☐ \(M\) ... Meaning of ... Name ... Memory location
☐ In other words, use same identifier/name in both Java and Assembler

Adding Semantics

♦ Revise R2
Syntax Rule Semantic Rule
R2: \(<e>_i \rightarrow <n>_j\) \(M(<e>_i) = M(<n>_j)\)
\(\text{code}(<e>_i) = \text{nothing}\)
☐ No code is generated!

♦ Revise R3
Syntax Rule Semantic Rules
R3: \(<s>_k \rightarrow <n>_j = <e>_i\); \text{code}(<s>_k) = \text{code}(<e>_i)\)
COPY AX, M(<e>_i)
COPY M(<n>_j), AX
☐ Says code for statement is code to calculate expression \(<e>_i\) and cod to copy it into memory associated with \(<n>_j\)

Generating Code

♦ Now have enough to demonstrate simplest case
☐ Use syntactic production to control process
☐ Associated semantic rules are applied at each step

♦ Use rules to generate code for \(X = Y\);
Derivation Syntax Rule Semantic Rules
S1 R3: s1 -> n2=e3; \(\text{code}(s1) = \text{code}(e3)\)
COPY AX, M(e3)
COPY M(n2), AX

\begin{tabular}{|c|c|}
\hline
MEANING: & \text{code}(s1) = \text{code}(e3) \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
MEANING: & \text{code}(s1) = \text{code}(e3) \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
MEANING: & \text{code}(s1) = \text{code}(e3) \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
MEANING: & \text{code}(s1) = \text{code}(e3) \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
MEANING: & \text{code}(s1) = \text{code}(e3) \\
\hline
\end{tabular}
Generating Code

### Derivation

<table>
<thead>
<tr>
<th>X = e3;</th>
<th>R2: e3 -&gt; n4</th>
<th>M(e3) = M(n4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>code(e3) = nothing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MEANING:**
- code(s1) = nothing
- \( \text{COPY AX, } M(n4) \)
- \( \text{COPY X, } AX \)

### More Rules

#### Revise R4

<table>
<thead>
<tr>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>R4: (&lt;e&gt;_i \rightarrow (&lt;e&gt;_j + &lt;e&gt;_k))</td>
<td>M(&lt;e&gt;_i) = createname</td>
</tr>
<tr>
<td>code(&lt;e&gt;_i) = code(&lt;e&gt;_j) + code(&lt;e&gt;_k)</td>
<td></td>
</tr>
<tr>
<td>( \text{COPY AX, } M(&lt;e&gt;_j) )</td>
<td></td>
</tr>
<tr>
<td>( \text{ADD AX, } M(&lt;e&gt;_k) )</td>
<td></td>
</tr>
<tr>
<td>( \text{COPY } M(&lt;e&gt;_i), AX )</td>
<td></td>
</tr>
</tbody>
</table>

- Says code for \(<e>_i\) is code to calculate expression \(<e>_j\) followed by code to calculate expression \(<e>_k\) and code to add them together and store that sum into memory associated with \(<e>_i\).

#### Revise R5

<table>
<thead>
<tr>
<th>Syntax Rule</th>
<th>Semantic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5: (&lt;e&gt;_i \rightarrow (&lt;e&gt;_j \times &lt;e&gt;_k))</td>
<td>M(&lt;e&gt;_i) = createname</td>
</tr>
<tr>
<td>code(&lt;e&gt;_i) = code(&lt;e&gt;_j) \times code(&lt;e&gt;_k)</td>
<td></td>
</tr>
<tr>
<td>( \text{COPY AX, } M(&lt;e&gt;_j) )</td>
<td></td>
</tr>
<tr>
<td>( \text{MUL AX, } M(&lt;e&gt;_k) )</td>
<td></td>
</tr>
<tr>
<td>( \text{COPY } M(&lt;e&gt;_i), AX )</td>
<td></td>
</tr>
</tbody>
</table>

- Says code for \(<e>_i\) is code to calculate expression \(<e>_j\) followed by code to calculate expression \(<e>_k\) and code to multiply them together and store that sum into memory associated with \(<e>_i\).
- Basically, rules R4 and R5 are identical except that the + and ADD in one are replaced by the \( \times \) and MUL in the other.

### Code for \( Z = (X + Y) \);

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1 R3: s1 -&gt; n2 = e3; code(s1) = code(e3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{COPY AX, } M(e3) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{COPY M(n2), AX } )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MEANING:**
- code(s1) = code(e3)
- \( \text{COPY AX, } M(e3) \)
- \( \text{COPY Z, AX } \)
Towards a Real Program

- More complicated statement:
  \( U_1 = (X + (Y \times Z)) \);
  \( U_1 \) done on pages 277-279 in text
  \( U_1 \) (Note that book uses \(<i>j\) where we used \(<n>j\)

- Rules for Looping Sequence of statements
- Rules 6 and 7: A sequence of statements

Syntax Rule Semantic Rules

R6: \(<q>_i\) -> \(<s>_j\) \( \text{code}(<q>_i) = \text{code}(<s>_j) \)

R7: \(<q>_k\) -> \(<s>_j\) \( \text{code}(<q>_i) = \text{code}(<s>_j) \)

- Says code for a sequence of statements is the code for the first statement followed by the code for the next statement, etc.
- Notice the recursive nature of these statements.
More Complicated Statements

- **Rule 8: Compound Statement**
  
  Syntax Rule: `<c>i` → `{ `<q>j` code(`c>i`) = code(`q>j`) }

- **Rule 9: While Statement**
  
  Syntax Rule: `<s>i` → `createname` while (<n>j) < (<e>k) `createname`
  
  Semantic Rules:
  ```
  M(<s>i) = createname
  M'(<s>i) = createname
  M(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  M'(<s>i) = code(<s>i)
  ```

Final Thoughts

- **Clean Up Translation**
  - Some code generated can be removed
  - Modern compilers spend a lot of effort optimizing

- **Important:** Everything done by *simple substitution*

- **Everything “adds up”**
  - `code( { <s>1;<s>2;<s>3 } )` is
    ```
    code(<s>1)
    code(<s>2)
    code(<s>3)
    ```