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 meaning*

- Revise R1

<table>
<thead>
<tr>
<th>Syntax Rule</th>
<th>Semantic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: &lt;n&gt;j -&gt; w</td>
<td>M(&lt;n&gt;j) = w</td>
</tr>
</tbody>
</table>

- 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  ->  <n>j   M(<e>i) = M(<n>j)
  code(<e>i) = nothing
  - No code is generated!

- Revise R3
  Syntax Rule  Semantic Rules
  R3: <s>k  ->  <n>j = <e>i;  code(<s>k) =  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 derivation to control process
  - Associated semantic rules are applied at each step
- Use rules to generate code for $x = y$;

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
</table>
| S1         | R3: $s_1 \rightarrow n_2 = e_3$; $\text{code}(s_1) = \text{code}(e_3)$ | $\text{COPY } AX, M(e_3)$  
              |              | $\text{COPY } M(n_2), AX$ |

**MEANING:**  
$\text{code}(s_1) = \text{code}(e_3)$  
$\text{COPY } AX, M(e_3)$  
$\text{COPY } M(n_2), AX$

$n_2 = e_3$;

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: $n_2 \rightarrow x$</td>
<td>$M(n_2) = x$</td>
<td></td>
</tr>
</tbody>
</table>

**MEANING:**  
$\text{code}(s_1) = \text{code}(e_3)$  
$\text{COPY } AX, M(e_3)$  
$\text{COPY } X, AX$
# Generating Code

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = e3;$</td>
<td>R2: $e3 \rightarrow n4$</td>
<td>M($e3$) = M($n4$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>code($e3$) = \textit{nothing}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEANING: code(s1) = \textit{nothing}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY AX, M($n4$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY X, AX</td>
</tr>
<tr>
<td>$x = n4;$</td>
<td>R1: $n4 \rightarrow Y$</td>
<td>M($n4$) = Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEANING: code(s1) =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY AX, Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY X, AX</td>
</tr>
<tr>
<td>$x = Y;$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
More Rules

• Revise R4

Syntax Rule          Semantic Rule
R4: \(<e>i \rightarrow (e<j + e>k)\)  \(M(<e>i) = \text{createname}\)

\[
\begin{align*}
\text{code}(<e>i) &= \text{code}(<e>j) \\
\text{code}(<e>k) &= \text{code}(<e>k) \\
\text{COPY AX, } M(<e>j) & \text{ ADD AX, } M(<e>k) \\
\text{COPY } M(<e>i), \text{ AX }
\end{align*}
\]

- 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 \(<e>i\)
More Rules

- Revise R5

Syntax Rule | Semantic Rule
---|---
R5: `<e>i` -> `( `<e>j`*`<e>k` )`  | `M(<e>i) = createname`
`code(<e>i)= code(<e>j)`
`code(<e>k)`
`COPY AX, M(<e>j)`
`MUL AX, M(<e>k)`
`COPY M(<e>i), AX`

- 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 `*` 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>
</table>
| s1         | R3: s1 -> n2=e3; code(s1) = code(e3) | \[
\begin{align*}
&\text{code(s1)} = \text{code(e3)} \\
&\text{COPY } AX, M(e3) \\
&\text{COPY } M(n2), AX
\end{align*}
\] |

MEANING:
\[
\begin{align*}
&\text{code(s1)} = \text{code(e3)} \\
&\text{COPY } AX, M(e3) \\
&\text{COPY } M(n2), AX
\end{align*}
\]

<table>
<thead>
<tr>
<th>n2 = e3;</th>
<th>R1: n2 -&gt; z</th>
<th>M(n2) = z</th>
</tr>
</thead>
</table>
| MEANING:   | code(s1) = code(e3) | \[
\begin{align*}
&\text{code(s1)} = \text{code(e3)} \\
&\text{COPY } AX, M(e3) \\
&\text{COPY } Z, AX
\end{align*}
\] |
**Code for** \( z = (x + y); \)

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z = e3; )</td>
<td>R4: ( e3 \rightarrow (e4+e5) )</td>
<td>M(e3) = CN1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>code(e3) = code(e4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>code(e5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY AX, M(e4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD AX, M(e5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY M(e3), AX</td>
</tr>
</tbody>
</table>

**MEANING:** code(s1) = code(e4) code(e5)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COPY AX, M(e4)</td>
</tr>
<tr>
<td></td>
<td>ADD AX, M(e5)</td>
</tr>
<tr>
<td></td>
<td>COPY CN1, AX</td>
</tr>
<tr>
<td></td>
<td>COPY AX, CN1</td>
</tr>
<tr>
<td></td>
<td>COPY Z, AX</td>
</tr>
</tbody>
</table>
**Code for** \[ Z = (X + Y); \]

\[ Z = (e4 + e5); \quad R2: e4 \rightarrow n6 \quad M(e4) = M(n6) \]

*MEANING:*  
\[
\begin{align*}
\text{code(e4)} &= \quad \text{nothing} \\
\text{COPY AX, M(n6)} \\
\text{ADD AX, M(e5)} \\
\text{COPY CN1, AX} \\
\text{COPY AX, CN1} \\
\text{COPY Z, AX}
\end{align*}
\]

\[ Z = (n6 + e5); \quad R1: n6 \rightarrow X \quad M(n6) = X \]

*MEANING:*  
\[
\begin{align*}
\text{code(s1)} &= \quad \text{code(e5)} \\
\text{code(e5)} \\
\text{COPY AX, X} \\
\text{ADD AX, M(e5)} \\
\text{COPY CN1, AX} \\
\text{COPY AX, CN1} \\
\text{COPY Z, AX}
\end{align*}
\]
Code for \( Z = (X + Y) \); \quad .4

\[ Z = (X + e^5); \quad R2: e^5 \rightarrow n7 \quad M(e^5) = M(n7) \]

\[ \text{code}(e^5) = \quad \text{nothing} \]

**MEANING:**

\[
\begin{align*}
\text{code}(s1) &= \quad \text{nothing} \\
&= \quad \text{COPY AX, X} \\
&= \quad \text{ADD AX, M(n7)} \\
&= \quad \text{COPY CN1, AX} \\
&= \quad \text{COPY AX, CN1} \\
&= \quad \text{COPY Z, AX}
\end{align*}
\]

\[ Z = (X + n7); \quad R1: n7 \rightarrow Y \quad M(n7) = Y \]

**MEANING:**

\[
\begin{align*}
\text{code}(s1) &= \quad \text{COPY AX, X} \\
&= \quad \text{ADD AX, Y} \\
&= \quad \text{COPY CN1, AX} \\
&= \quad \text{COPY AX, CN1} \\
&= \quad \text{COPY Z, AX}
\end{align*}
\]

\[ Z = (X + Y); \]
Towards a Real Program

- More complicated statement:
  \[ U1 = (X + (Y \times Z)) ; \]
  - Done on pages 277-279 in text
  - (Note that book uses \(<i>j\) where we used \(<n>j\))

- Rules for Looping Sequence of statements

- Rules 6 and 7: \textit{A sequence of statements}

<table>
<thead>
<tr>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>R6: (&lt;q&gt;i \rightarrow &lt;s&gt;j) \text{ code}(&lt;q&gt;i) = \text{ code}(&lt;s&gt;j) \text{ code}(&lt;q&gt;k))</td>
<td></td>
</tr>
<tr>
<td>R7: (&lt;q&gt;i \rightarrow &lt;s&gt;j) \text{ code}(&lt;q&gt;i) = \text{ code}(&lt;s&gt;j))</td>
<td></td>
</tr>
</tbody>
</table>
  - 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
  
  R8: <c>i -> {
    
  <q>j
    
  code(<c>i) = code(<q>j)

  }

- **Rule 9: While Statement**
  
  Syntax Rule
  
  R9: <s>i ->

  while (<n>j < <e>k)  
    
  <c>h

  code(<s>i) =

  M(<s>i) code(<e>k)

  COPY AX, M(<n>j)

  CMP AX, M(<e>k)

  JNB M'(s<s>i)

  code(<c>h)

  JMP M(<s>i)

  M'(s<s>i) NO-OP
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)