Today’s topics

**Language Translation**
   Generating Code

**Upcoming**
   Electric Circuits (Hardware)
      (not in text)

**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

<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  
R2: \(<e>_i -> <n>_j\)

Semantic Rules  
\(M(<e>_i) = M(<n>_j)\)
\(\text{code}(<e>_i) = \text{nothing}\)

▫ No code is generated!

❖ **Revise R3**

Syntax Rule  
R3: \(<s>_k -> <n>_j = <e>_i; \text{ code}(<s>_k) = \text{ code}(<e>_i)\)

Semantic Rules  
\(\text{COPY AX, } M(<e>_i)\)
\(\text{COPY M(<n>_j), AX}\)

▫ Says code for statement is code to calculate expression \(<e>_i\) and code to copy it into memory associated with \(<n>_j\)
Generating Code for $X = Y$;

- 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 $X = Y$;

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

**MEANING:** code(s1) = code(e3)

<table>
<thead>
<tr>
<th>n2 = e3;</th>
<th>R1: n2 -&gt; x</th>
<th>M(n2) = x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>COPY AX, M(e3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY M(n2), AX</td>
</tr>
</tbody>
</table>

**MEANING:** code(s1) = code(e3)

|          |             | COPY AX, M(e3) |
|          |             | COPY X, AX |
Generating Code for $x = y$;

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = e_3;$</td>
<td>R2: $e_3 \rightarrow n_4$</td>
<td>$M(e_3) = M(n_4)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{code}(e_3) = \text{nothing}$</td>
</tr>
</tbody>
</table>

**MEANING:**

$\text{code}(s_1) = \text{nothing}$

$\text{COPY AX, } M(n_4)$

$\text{COPY X, AX}$

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = n_4;$</td>
<td>R1: $n_4 \rightarrow y$</td>
<td>$M(n_4) = y$</td>
</tr>
</tbody>
</table>

**MEANING:**

$\text{code}(s_1) = \text{COPY AX, } y$

$\text{COPY X, AX}$

$x = y;$
More Rules

- **Revise R4**

  **Syntax Rule**
  
  R4: $<e>_i \rightarrow (<e>_j + <e>_k)$

  **Semantic Rules**

  $M(<e>_i) = createName$

  code($<e>_i$) = code($<e>_j$) + code($<e>_k$)

  COPY AX, M($<e>_j$)

  ADD 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 add them together and store that sum into memory associated with $<e>_i$.
More Rules

- **Revise R5**

  **Syntax Rule**
  \[ R5: \langle e \rangle_i \rightarrow (\langle e \rangle_j * \langle e \rangle_k) \]

  **Semantic Rules**
  \[ M(\langle e \rangle_i) = \text{createname} \]
  \[ \text{code}(\langle e \rangle_i) = \text{code}(\langle e \rangle_j) \]
  \[ \text{code}(\langle e \rangle_k) \]
  \[ \text{COPY AX, } M(\langle e \rangle_j) \]
  \[ \text{MUL AX, } M(\langle e \rangle_k) \]
  \[ \text{COPY } M(\langle e \rangle_i), AX \]

  - Says code for \( \langle e \rangle_i \) is code to calculate expression \( \langle e \rangle_j \) followed by code to calculate expression \( \langle e \rangle_k \) and code to multiply them together and store that sum into memory associated with \( \langle e \rangle_i \)
  
  - 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>
<tbody>
<tr>
<td>s1</td>
<td>R3: ( s1 \to n2=e3 ); ( \text{code}(s1) = \text{code}(e3) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \text{COPY AX, M(e3)} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \text{COPY M(n2), AX} )</td>
<td></td>
</tr>
</tbody>
</table>

**MEANING:** \( \text{code}(s1) = \text{code}(e3) \)

<table>
<thead>
<tr>
<th>n2 = e3;</th>
<th>R1: ( n2 \to Z ) ( M(n2) = Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{COPY AX, M(e3)} )</td>
</tr>
<tr>
<td></td>
<td>( \text{COPY M(n2), AX} )</td>
</tr>
</tbody>
</table>

**MEANING:** \( \text{code}(s1) = \text{code}(e3) \)

// COPY AX, M(e3) // COPY M(n2), AX //
**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) = \text{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>
<tr>
<td>MEANING:</td>
<td></td>
<td>code(s1) =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 CN1, AX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY AX, CN1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY Z, AX</td>
</tr>
</tbody>
</table>
Code for $Z = (X + Y)$ ; .3

$Z = (e4 + e5)$ ; R2: $e4 \rightarrow n6$  
$M(e4) = M(n6)$  
$\text{code}(e4) =$ \textit{nothing}

MEANING: \begin{align*}
\text{code}(s1) &= \textit{nothing} \\
\text{code}(e5) &= \\
\text{COPY AX, } M(n6) \\
\text{ADD AX, } M(e5) \\
\text{COPY CN1, AX} \\
\text{COPY AX, CN1} \\
\text{COPY } Z, \ AX
\end{align*}

$Z = (n6 + e5)$ ; R1: $n6 \rightarrow x$  
$M(n6) = x$

MEANING: \begin{align*}
\text{code}(s1) &= \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 \[ Z = (X + Y); \] .4

\[ Z = (X + \text{e5}); \quad \text{R2: e5} \rightarrow \text{n7} \quad \text{M(e5)} = \text{M(n7)} \]
\[ \text{code(e5)} = \text{nothing} \]

** MEANING: **

\[ \text{code(s1)} = \text{nothing} \]
\[ \text{COPY AX, X} \]
\[ \text{ADD AX, M(n7)} \]
\[ \text{COPY CN1, AX} \]
\[ \text{COPY AX, CN1} \]
\[ \text{COPY Z, AX} \]

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

** MEANING: **

\[ \text{code(s1)} = \]
\[ \text{COPY AX, X} \]
\[ \text{ADD AX, Y} \]
\[ \text{COPY CN1, AX} \]
\[ \text{COPY AX, CN1} \]
\[ \text{COPY Z, AX} \]

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

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

- Rules for Looping Sequence of statements

- Rules 6 and 7: A sequence of statements

Syntax Rule | Semantic Rules
--- | ---
R6: \( <q>_{i} \rightarrow <s>_{j} \) | code\( ( <q>_{i} ) = \) code\( ( <s>_{j} ) \)
\( <q>_{k} \) | code\( ( <q>_{k} ) \)

R7: \( <q>_{i} \rightarrow <s>_{j} \) | code\( ( <q>_{i} ) = \) 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**
  
  R8: \(<c>\_i \rightarrow \{
  
  \langle q\rangle_j
  
  \}
  
  **Semantic Rules**
  
  \(\text{code}(<c>_i) = \text{code}(<q>_j)\)

- **Rule 9: While Statement**
  
  **Syntax Rule**
  
  R9: \(<s>_i \rightarrow\)
  
  \(\text{while } (<n>_j < <e>_k)\)
  
  \(<c>_h\)

  **Semantic Rules**
  
  \(\text{M}(<s>_i) = \text{create name}\)
  
  \(\text{M}'(<s>_i) = \text{create name}\)
  
  \(\text{code}(<s>_i) =\)
  
  \(\text{M}(<s>_i) \text{ code}(<e>_k)\)
  
  \(\text{COPY AX, M}(<n>_j)\)
  
  \(\text{CMP AX, M}(<e>_k)\)
  
  \(\text{JNB M}'(<s>_i)\)
  
  \(\text{code}(<c>_h)\)
  
  \(\text{JMP M}(<s>_i)\)
  
  \(\text{M}'(<s>_i) \text{ NO-OP}\)
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)`