Addressing and Basic Machine Language

CSE 361S

Addressing Modes
- Register addressing – operand is in register

\[
\text{addl } %eax, %ebx \quad %ebx \leftarrow %ebx + %eax
\]

- Last character of op code denotes data size:
  b byte (8 bits) w word (16 bits)
  l double word (32 bits) q quad word (64 bits)

- Immediate addressing – operand is explicitly present in code

\[
\text{subl } 10, %edx \quad %edx \leftarrow %edx - 10
\]

- Constant value uses C notation
  - Default base is 10
  - Hex uses 0x10 notation
  - Negative constants are allowed, e.g., $-12$

- Direct/Absolute addressing – memory address of operand is explicit in code

\[
\text{addl } 0x42, %ecx \quad %ecx \leftarrow %ecx + M[0x42]
\]

\[
\text{movl } %edx, \text{var1} \quad M[\text{var1}] \leftarrow %edx
\]

- Note: use of named symbol var1 is most common
  - Assembler translates into actual address
  - Or linker if specified in another file

Practice Problems

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0xff</td>
<td>%eax</td>
<td>0x100</td>
</tr>
<tr>
<td>0x104</td>
<td>0xab</td>
<td>%ecx</td>
<td>0x1</td>
</tr>
<tr>
<td>0x108</td>
<td>0x13</td>
<td>%edx</td>
<td>0x3</td>
</tr>
</tbody>
</table>

Operand Value Addr. mode

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
<th>Addr. mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>%eax</td>
<td>0x104</td>
<td></td>
</tr>
<tr>
<td>$0x108</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Indirect addressing — memory address is stored in register

  \[
  \text{movl} (\%ebx), \%eax \quad \%eax \leftarrow M[\%ebx]
  \]

- It is often called register indirect
- Note similarity with base-displacement, with displacement of zero

  \[
  \text{movl} 0x0(\%ebx), \%eax \quad \%eax \leftarrow M[0+\%ebx]
  \]

- Indexed addressing — memory address is formed by adding two registers (one called “base”, the other called “index”)

  \[
  \text{addl} (\%ebx,\%esi),\%ecx \quad \%ecx \leftarrow \%ecx + M[\%ebx+\%esi]
  \]

- Scaled indexed addressing — index register can be scaled by 1, 2, 4, or 8

  \[
  \text{movl} \$0x7f, (\%ebx,\%edi,2) \quad M[\%ebx + 2 \times \%edi] \leftarrow 0x7f
  \]

- Full form — literal (direct), base register, index register, scale are all allowed

  \[
  \text{movl} 0x50(\%ebx,\%esi,4), \%eax \quad \%eax \leftarrow M[0x50 + \%ebx + 4 \times \%esi]
  \]

- Form commonly output by disassemblers:

  \[
  \text{addl} \text{name}(1), \%ecx
  \]

---

**Practice Problems**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0xff</td>
<td>%eax</td>
<td>0x100</td>
</tr>
<tr>
<td>0x104</td>
<td>0xab</td>
<td>%ecx</td>
<td>0x1</td>
</tr>
<tr>
<td>0x108</td>
<td>0x13</td>
<td>%edx</td>
<td>0x3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operand</th>
<th>Value</th>
<th>Addr. mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>(%eax)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4(%eax)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1(%eax,%edx)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operand Sizes**

- `<op>l` 32-bit operand
- `<op>w` 16-bit operand
- `<op>b` 8-bit operand
General Form

label: opcode operands   comment

• Label is optional
• Comments use /* comment */ notation
  – Many other assemblers use ; comment
  – Or some other notation, e.g., # comment
• Pseudo-operations are commands to assembler
  .text means instructions follow