Last of “how computers work”

Lecture 18
Machine code and more

Admin

• Exam scores
  – I still don’t have scores from all of the GSIs.
  – It looks like the median and average are around 70, but I really can’t tell yet.
  – I expect to have them all today, so I should be able to post the numbers and graphs later today.
  – Answers will be posted over the weekend.

Until break

• Our plans are as follows:
  – Monday and Wednesday will be an introduction to Matlab.
    • Matlab reading will be posted on the website.
  – Friday will cover some additional C++ language constructs and perhaps finish up Matlab (as needed)
  – The Monday we get back the next project will be discussed.

Last of “How a Computer works”

• We’ve covered some basics, but I felt a lot of you were lost.
  – So today I’m going to give a very detailed example of the execution of a computer.
  – The goal is to get you all to have some appreciation for what a computer is doing
  – The example is very simplistic, but at the same time does provide an accurate view of what a computer does.
Disclaimer

- This is an example of a very simplified computer. A real computer might have 100s of different instructions and around a billion bytes of memory. This one has 4 instructions and 64 bytes of memory.
  - Still, while simplified it does illustrate the basics of computer operation.

- Each computer instruction takes up 16 bits (2 bytes) of memory. The instructions are encoded as follows:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>00</td>
<td>memA</td>
<td>memB</td>
</tr>
<tr>
<td>addi</td>
<td>01</td>
<td>memA</td>
<td>immediate</td>
</tr>
<tr>
<td>beq</td>
<td>10</td>
<td>memA</td>
<td>target</td>
</tr>
<tr>
<td>print</td>
<td>11</td>
<td>memA</td>
<td>unused</td>
</tr>
</tbody>
</table>

beq: if(Mem[memA]==0) PC=target
print: print Mem[memA] and halt.

So 0000 0000 0000 0000 says to add the byte at memory location 0 to itself and store the result in memory location zero.

Memory

- Memory is just a big array
  - Well not-so-big in our case.
- Each address contains a single byte (8 bits)
- The “memory address” is an index into the array.
  - Note that with 64 bytes of memory we need \( \log_2(64) \) or 6 bits to address the memory.
  - 6 bits can represent 0 to 63.
PC

- The program counter, or “PC” points to the next instruction to be fetched.
  - It is just a special 6-bit memory location, separate from “memory”
- Recall that an instruction takes up 16 bit, so an instruction will use 2 memory addresses, PC and PC+1.
- When an instruction finished executing, it sets the PC = PC+2.
  - Branches will sometimes be an exception to this!

OK, let’s look at an example:

- The algorithm
  
  \[
  \text{sum} = \text{sum} + x \\
  y = y - 1 \\
  \text{if}(y = 0) \text{done}
  \]

\[
\begin{array}{ll}
\text{start:} & \text{add sum, } x \\
          & \text{addi } y, -1 \\
          & \text{beq } y, \text{done} \\
          & \text{beq } z, \text{start} \\
\text{done:} & \text{print sum} \\
\end{array}
\]

\[
\begin{array}{ll}
\text{sum:} & 0 \\
\text{y:} & 3 \\
\text{x:} & 4 \\
\text{z:} & 0
\end{array}
\]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>00</td>
<td>memA</td>
<td>memB</td>
</tr>
<tr>
<td>addi</td>
<td>01</td>
<td>memA</td>
<td>immediate</td>
</tr>
<tr>
<td>beq</td>
<td>10</td>
<td>memA</td>
<td>target</td>
</tr>
<tr>
<td>print</td>
<td>11</td>
<td>memA</td>
<td>unused</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00001010</td>
</tr>
<tr>
<td>1</td>
<td>00001100</td>
</tr>
<tr>
<td>2</td>
<td>01001011</td>
</tr>
<tr>
<td>3</td>
<td>11111111</td>
</tr>
<tr>
<td>4</td>
<td>10001011</td>
</tr>
<tr>
<td>5</td>
<td>00001000</td>
</tr>
<tr>
<td>6</td>
<td>10001101</td>
</tr>
<tr>
<td>7</td>
<td>00000000</td>
</tr>
<tr>
<td>8</td>
<td>11001010</td>
</tr>
<tr>
<td>9</td>
<td>00000000</td>
</tr>
<tr>
<td>10</td>
<td>00000000</td>
</tr>
<tr>
<td>11</td>
<td>00000011</td>
</tr>
<tr>
<td>12</td>
<td>00000100</td>
</tr>
<tr>
<td>13</td>
<td>00000000</td>
</tr>
</tbody>
</table>