1 Administrivia

Announcements

Assignment

From Last Time

Outline

1. Arithmetic and logical instructions.

2. Instruction operands.

3. Instruction formats.

Coming Up

Conditional execution.

2 Arithmetic and Logical Instructions

• add, sub, addi, addu, subu.
• and, or, andi, ori, sll, srl.

2.1 Instruction Semantics

add a, b, c # This, BTW, is a comment.
sub a, a, b
addi a, a, 100
and a, b, c
andi a, a, 32000

De-compile each of the following:

add a, b, c
add a, a, d
add a, a, e

De-compile further into a single HLL statement.

Compile each of the following:

a = b + c;
d = a - e;
f = (g + h) - (i + j);

Operands are registers.

3 Instruction Operands

Properties of registers:

1. Number of registers. 32 for MIPS, including the hardwired register. Two ways of naming: numbers, convention “nicknames”.

2. Number of bits/register. 32. Word size.
   Implications: size of address space, datapath width.
3.1 Using MIPS Registers

Recall:

\[ f = (g + h) - (i + j); \]

Assume \( f \) through \( j \) are in \$1 \) through \$5\), respectively. Compile the statement.

3.2 Memory Addressing

1. HLL have complex data structures such as arrays and structs. How are they handled?


3. Actual MIPS instructions: \texttt{lw, sw}.

   Base and offset addressing: \texttt{lw $s0, 8($s1)}

4. MIPS memory is byte addressable, so word addresses differ by 4:

   \begin{center}
   \begin{tabular}{|c|c|c|c|}
   \hline
   Word Address & Byte Address \\
   \hline
   0 & 0 & 1 & 2 & 3 \\
   4 & 4 & 5 & 6 & 7 \\
   8 & 8 & 9 & 10 & 11 \\
   12 & 12 & 13 & 14 & 15 \\
   \hline
   \end{tabular}
   \end{center}

   Compile the following:

   \[ g = h + A[8]; \]

   Where \( g \) is in \$s1\), \( h \) is in \$s2\), and the base address of \( A \), an array of 100 words, is in \$s3\).

   Base, offset addressing.

   Compile each of the following:
A[j] = h + A[i];

Base, offset addressing, using constant offsets, is similarly useful for accessing members of structures.

4 Instruction Formats

4.1 MIPS R-Format

Example instruction: add $s2, $s0, $s1

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>21</td>
<td>16</td>
<td>11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Op</td>
<td>Rs</td>
<td>Rt</td>
<td>Rd</td>
<td>Shamt</td>
<td>Func</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Fields:

2. Rs: First source operand.
4. Rd: Destination operand.
5. Shamt: Shift amount — ignore for now.

In assembly: Op/Func Rd, Rs, Rt

Notes:
1. Example encodings:

<table>
<thead>
<tr>
<th>Assembly</th>
<th>Op</th>
<th>Rs</th>
<th>Rt</th>
<th>Rd</th>
<th>Shamt</th>
<th>Func</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $1, $2, $3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>sub $4, $5, $6</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>34</td>
</tr>
</tbody>
</table>

4.2 MIPS I-Format

Example instruction: `lw $s0 8($s1)`

<table>
<thead>
<tr>
<th>Op</th>
<th>Rs</th>
<th>Rt</th>
<th>Immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>16</td>
</tr>
</tbody>
</table>

Fields:

2. Rs: Source register.
4. Address: 16-bit signed immediate value.

**Offset range?**

In assembly: `Op/Func Rt, address(Rs)`

Notes:

1. This format also used for immediate operands: `addi $1, $2, 123`.
2. Example encodings:

<table>
<thead>
<tr>
<th>Assembly</th>
<th>Op</th>
<th>Rs</th>
<th>Rt</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw $1, 1000($2)</td>
<td>35</td>
<td>2</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>sw $3, -12($4)</td>
<td>43</td>
<td>4</td>
<td>3</td>
<td>-12</td>
</tr>
</tbody>
</table>