Register Names

We will learn other register names later.

$\text{t}$ registers: why "temporary"?

(C instruction) \[ g = g + h + i \]

(MIPS instructions)

\[
\begin{align*}
\text{add} & \quad $t0, \quad $s1, \quad $s2 \\
\text{add} & \quad $s1, \quad $t0, \quad $s3
\end{align*}
\]

Conditional Branch (last lecture) e.g. "branch equals" (\texttt{beq})

(C instruction)

\[
\begin{align*}
\text{if} \ (a \neq b) & \quad // \text{if } a == b \\
\quad f = g + h & \quad // \text{then don't do it}
\end{align*}
\]

(MIPS instructions)

\[
\begin{align*}
\text{beq} & \quad $s1, \quad $s2, \quad \text{Exit1} \\
\text{add} & \quad $s3, \quad $s4, \quad $s5
\end{align*}
\]

(Exit 1:)

Other Conditional Branches?

\[
\begin{align*}
\text{if} \ (a \leq b) & \quad \text{\texttt{bgt}} \quad \ldots \\
\text{if} \ (a < b) & \quad \text{\texttt{bge}} \quad \ldots \\
\text{if} \ (a \geq b) & \quad \text{\texttt{blt}} \quad \ldots \\
\text{if} \ (a > b) & \quad \text{\texttt{ble}} \quad \ldots
\end{align*}
\]

These instructions don't exist in MIPS.

Why not? What to do?

Instead use "set less than" \texttt{slt}

\[
\begin{align*}
\text{\texttt{slt}} & \quad \text{\texttt{$t0, \quad $s1, \quad $s2}} \\
\text{Assign} \quad \text{\texttt{$t0 = 1}} & \quad \text{\texttt{, if } $s1 < $s2} \\
& \quad \text{\texttt{0, if } $s1 \geq $s2}
\end{align*}
\]

\[
\begin{align*}
\text{\texttt{bgt}} & \quad \text{\texttt{$s1, \quad $s2, \quad \text{Exit1}}} \\
\text{\texttt{bge}} & \quad \text{\texttt{$s1, \quad $s2, \quad \text{Exit1}}} \\
\text{\texttt{blt}} & \quad \text{\texttt{$s1, \quad $s2, \quad \text{Exit1}}} \\
\text{\texttt{ble}} & \quad \text{\texttt{$s1, \quad $s2, \quad \text{Exit1}}}
\end{align*}
\]
How to express inequalities using only "<" and "not"?

\[
\begin{align*}
a \geq b & \iff \overline{a < b} \\
a > b & \iff \overline{b < a} \\
a \leq b & \iff \overline{b < a}
\end{align*}
\]

Exercise

<table>
<thead>
<tr>
<th>Desired</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbf{bge} \ 50, 51, \ Exit</td>
<td>\textbf{ble} \ 50, 51, \ Exit</td>
</tr>
<tr>
<td>\textbf{bgt} \ 50, 51, \ Exit</td>
<td>\textbf{ble} \ 50, 51, \ Exit</td>
</tr>
</tbody>
</table>

Why was MIPS designed this way?

The benefits of having fewer instructions available outweighs the cost of having more instructions in each program.

(recall: RISC - reduced instruction set computer)

Signed versus Unsigned instructions

\[
\begin{align*}
\text{add} & \ 50, 51, 52 \\
\text{addu} & \ 50, 51, 52 \\
\end{align*}
\]

Q: What is the difference here?
A: The "overflow" conditions (recall Exercises 2 Q 12)

\[
\begin{align*}
\text{e.g. } & \ 01...... \quad \text{signed -> overflow} \\
& + \ 01...... \quad \text{unsigned -> no overflow}
\end{align*}
\]

To avoid headaches, MIPS assemblers allow you to use pseudoinstructions such as \texttt{blt}, \texttt{ble}.

Assemblers (e.g. MARS) convert these to real MIPS instructions.
Exercise:
MIPS doesn't have these two. Why not?

Signed       Unsigned
R
add
sub
addu
subu
slt
sliu
(signed and unsigned)
I
addi
addiu
subi
sli
sliu
(signed and unsigned)

Example
Let $s0 = 01$
Then $slt \ s0, \ s0, \ s1 \Rightarrow s0 = 0$
$sliu \ s0, \ s0, \ s1 \Rightarrow s0 = 1$

Sign Extension
addi $s16, \ s17, \ 3$

Data path. (coming soon...)

Manipulating Bits
How to put some 32 bit pattern e.g. 0x37b1 fa93 into a register?

Announcements
- A1 due Wednesday night
- A2 will be posted Wednesday (MIPS programming)
- no office hours today for me

Shifting Bits
# shift left logical
$sll \ s0, \ s1, \ 7$
# shift right logical
$srl \ s0, \ s0, \ 8$

(You might have thought this was "I" format, but in fact it is "R" format. Why?)