Instructions: Data Transfer

load and store architecture

MIPS: can access memory only to transfer data to or from registers
CISC: may allow, for example, add to memory location, store in register

load: copy the data from memory into a register.
store: copy the data from a register to memory.

lw  $rt, offset($rs)  Load word from memory location to register
sw  $rt, offset($rs)  Store word from register to memory location

Offset is a 16-bit 2C value (immediate); all are l-type
semantics of lw

    Addr <-- R[s] + (IR_{15})^{16}:IR_{15-0}
    R[t] <-- M_4[ Addr ]

compute address

- add the contents of register s (base) to the sign-extended offset
- offset is immediate
- non-aligned address: hardware exception

get data

- copy 4 bytes located at memory address starting at Addr to register t.
- CPU fetches the four bytes based on the endianness of the machine

sw similar to lw but the 4 byte quantity is copied from the register to memory.

    M_4[ Addr ] <-- R[t]  stored in the endianness of the machine
Instructions: Data Transfer

Byte operations

\[
\begin{align*}
\text{lb} & \quad \text{Load sign-extended byte from memory location to register} \\
\text{lbu} & \quad \text{Load zero-extended (unsigned) byte from memory location to register} \\
\text{sb} & \quad \text{Store the least significant byte of a register to memory location}
\end{align*}
\]

What about \text{sbu}?

For \text{lb}, the address is computed the same way as \text{lw}, but the address does not have to be word aligned.

\[
\begin{align*}
\text{Addr} & \leftarrow R[s] + (IR_{15})^{16::IR_{15-0}} \\
R[t] & \leftarrow (M_1[\text{Addr }])^{24::M_1[\text{Addr }]} \\
\end{align*}
\]

Since the value is interpreted as 2C, the fetched byte is sign-extended to 32 bits.

\text{lbu} is just like \text{lb} except the byte is zero-extended in the register.

\text{sb} is similar to \text{sw}:

\[
M_1[\text{Addr }] \leftarrow R[t]_{7-0}
\]

The least significant byte of register t is copied to the address in memory.

Do we really need to have separate instructions to load, store bytes?
Instructions: Data Transfer

machine code

\texttt{lw \ \$rt, \ offset($rs)}

\begin{center}
\begin{tabular}{ccccccc}
100101 & 01000 & 01001 & 00000 & 00000 & 100000 \\
\hline
b_{31-26} & b_{25-21} & b_{20-16} & b_{15-0} \\
opcode & $rs$ & $rt$ & immediate \\
\end{tabular}
\end{center}

opcodes: 100 xxx (load)
101 xxx (store)

$rs$: base address
$rt$: target register
immediate: offset
Instructions: Data Transfer

Halfword operations (short int)

- short int is usually 16 bits
- lh $rt, offset($rs)  Load halfword from memory location to register
  Data is sign-extended in register
- lhu $rt, offset($rs) Data is zero-extended in register
- sh $rt, offset($rs)  Store halfword from register to memory location

Loading constant
- lui $rt, immed

Semantics:

$\text{R}[t] = IR_{15-0} 0^{16}$
- load the lower halfword of \text{immed} into upper halfword of \text{$rt$}
- lower bits of \text{$rt$} are set to 0
- \text{$rs$} is ignored
# Data transfer: summary

<table>
<thead>
<tr>
<th>Load</th>
<th>R-type</th>
<th>l-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td></td>
<td>lw</td>
</tr>
<tr>
<td>Halfword</td>
<td></td>
<td>lh</td>
</tr>
<tr>
<td>Halfword unsigned</td>
<td></td>
<td>lhu</td>
</tr>
<tr>
<td>Byte</td>
<td></td>
<td>lb</td>
</tr>
<tr>
<td>Byte unsigned</td>
<td></td>
<td>lbu</td>
</tr>
<tr>
<td>Constant</td>
<td></td>
<td>lui</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Store</th>
<th>R-type</th>
<th>l-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td></td>
<td>sw</td>
</tr>
<tr>
<td>Halfword</td>
<td></td>
<td>sh</td>
</tr>
<tr>
<td>Byte</td>
<td></td>
<td>sb</td>
</tr>
</tbody>
</table>
Instruction Types

- Arithmetic
- Logical
- Data Transfer
- Compare/Branch
- Jump