Instance Variables and Structs

Struct Allocation

- Static structs are allocated by the compiler
- Dynamic structs are allocated at runtime
- Variables that are an instance of a class or struct
- Created dynamically
- Many instances of the same variable can co-exist
- Java: objects are instances of non-static variables of a class
- C: structs are named variable groups, instance is also called a struct
- Accessing an instance variable
  - Requires a reference to a particular object (pointer to a struct)
  - Then variable name chooses a variable in that object (struct)

Struct Access

- Static and dynamic differ by an extra memory access
- Dynamic structs have dynamic address that must be read from memory
- In both cases the offset to variable from base of struct is static
- The revised load/store base plus offset instructions
  - Dynamic base address in a register plus a static offset (displacement)

The Revised Load-Store ISA

- Machine format for base + offset
  - Note that the offset will in our case always be a multiple of 4
  - Also note that we only have a single hex digit in instruction to store it
  - And so, we will store offset/4 in the instruction

The Revised ISA

<table>
<thead>
<tr>
<th>Name</th>
<th>Semantics</th>
<th>Assembly</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load immediate</td>
<td>r[d] = v</td>
<td>ld 5v, rd</td>
<td>0000000</td>
</tr>
<tr>
<td>Load base-offset</td>
<td>r[d] = m[rs]+r[p*4]</td>
<td>ld (rs), rd</td>
<td>10d</td>
</tr>
<tr>
<td>Load indexed</td>
<td>r[d] = m[rs+r[p*4]]</td>
<td>ld (rs), rd</td>
<td>20d</td>
</tr>
<tr>
<td>Store base-offset</td>
<td>m[r[d]+r[p*4]] = r[s]</td>
<td>st rs, rd</td>
<td>30d</td>
</tr>
<tr>
<td>Store indexed</td>
<td>m[r[d]+r[p*4]] = r[s]</td>
<td>st rs, (rs), rd</td>
<td>40d</td>
</tr>
</tbody>
</table>