Instance Variables

- Variables that are an instance of a class or struct
  - created dynamically
  - many instances of the same variable can co-exist

Java vs C
- 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)

Structs in C

- A struct is a
  - collection of variables of arbitrary type, allocated and accessed together

Declaration
- similar to declaring a Java class without methods
- name is “struct” plus name provided by programer
  - static
  - dynamic

Access
- static
- dynamic
**Struct Allocation**

- Static structs are allocated by the compiler
  
  ```
  struct D {
    int e;
    int f;
  };
  ```

- Dynamic structs are allocated at runtime
  
  - The variable that stores the struct pointer may be static or dynamic
  - The struct itself is allocated when the program calls `malloc`

  ```
  struct D* d1;
  ```

**Static Memory Layout**

- 0x1000: value of `d0.e`
- 0x1004: value of `d0.f`

**Struct Access**

- Static and dynamic differ by an extra memory access
  
  - Dynamic structs have a dynamic address that must be read from memory
  - In both cases, the offset to variable from base of struct is static

  ```
  d0.e = d0.f;
  d1->e = d1->f;
  ```

**The revised load/store base plus offset instructions**

- Dynamic base address in a register plus a static offset (displacement)

  ```
  ld $0x1000, r0    # r0 = address of d0
  ld 4(r0), r1     # r0 = d0.f
  st r1, (r0)      # d0.e = d0.f
  ```

  ```
  ld $0x1000, r0    # r0 = address of d1
  ld 4(r0), r1     # r0 = d1.e
  st r2, (r1)      # d1->e = d1->f
  ```
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] \leftarrow v )</td>
<td>ld $v$, rd</td>
<td>0d--vvvvvvv</td>
</tr>
<tr>
<td>load base+offset</td>
<td>( r[d] \leftarrow m[r[s]+(o=p*4)] )</td>
<td>ld o(r$s$), rd</td>
<td>1psd</td>
</tr>
<tr>
<td>load indexed</td>
<td>( r[d] \leftarrow m[r[s]+4*r[i]] )</td>
<td>ld (r$s$,r$i$,4), rd</td>
<td>2sid</td>
</tr>
<tr>
<td>store base+offset</td>
<td>( m[r[d]+(o=p*4)] \leftarrow r[s] )</td>
<td>st r$s$, o(rd)</td>
<td>3spd</td>
</tr>
<tr>
<td>store indexed</td>
<td>( m[r[d]+4*r[i]] \leftarrow r[s] )</td>
<td>st r$s$, (rd,r$i$,4)</td>
<td>4sdi</td>
</tr>
</tbody>
</table>