What the … is a Lattice?

• Represents values: for one item, or vector of all (often boolean to powerset)
• Has a defined top and bot
• According to ASU:
  - Top is least info:
    \[ \top \land X = X \]
  - Bot is end:
    \[ \bot \land X = \bot \]
  - Init in[b] with top, out[b] with \( F_b(\top) \).

Constructing Gen & Kill

<table>
<thead>
<tr>
<th>Stmt s</th>
<th>gen[s]</th>
<th>kill[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t \leftarrow x \text{ op } y )</td>
<td>( { x \text{ op } y } - \text{kill}[s] )</td>
<td>( { \text{exprs containing } t } )</td>
</tr>
<tr>
<td>( t \leftarrow M[a] )</td>
<td>( { M[a] } - \text{kill}[s] )</td>
<td>( { \text{exprs containing } t } )</td>
</tr>
<tr>
<td>( M[a] \leftarrow b )</td>
<td>( { } )</td>
<td>( { \text{for all } x, M[x] } )</td>
</tr>
<tr>
<td>( f(a, \ldots) )</td>
<td>( { } )</td>
<td>( { \text{for all } x, M[x] } )</td>
</tr>
<tr>
<td>( t \leftarrow f(a,\ldots) )</td>
<td>( { } )</td>
<td>( { \text{exprs containing } t } )</td>
</tr>
</tbody>
</table>
Constructing Gen & Kill

<table>
<thead>
<tr>
<th>Stmt s</th>
<th>gen[s]</th>
<th>kill[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>t &lt;- x op y</td>
<td>{x op y}-kill[s]</td>
<td>{exprs containing t}</td>
</tr>
<tr>
<td>t &lt;- M[a]</td>
<td>{M[a]}-kill[s]</td>
<td>{exprs containing t}</td>
</tr>
<tr>
<td>M[a] &lt;- b</td>
<td>{M[a]}</td>
<td>{for all x, M[x]}</td>
</tr>
<tr>
<td>f(a, ...)</td>
<td>{}</td>
<td>{for all x, M[x]}</td>
</tr>
<tr>
<td>t &lt;- f(a,..)</td>
<td>{}</td>
<td>{exprs containing t for all x, M[x]}</td>
</tr>
</tbody>
</table>

But...what if b gets overwritten?

But, how is a store an available expression?
Big Picture

- `.c`
- SUIF
  - front end
  - opts
  - many passes
  - massage
  - Schedule SUIF
- Build Pegasus
- Optimize Pegasus
  - Assignment 1
- Emit C6X Assembly code

Schedule Pegasus

- users must come after producers
- only independent ops can be scheduled in the same instruction
Schedule Pegasus

- Ops must execute in a specific slot - on a specific function unit
- Resource conflicts might require ops to be delayed

Do register allocation

- If two spans overlap, they can't use the same register...
- End result: label each wire/edge with a register

Do register allocation

- If two spans overlap, they can't use the same register...
- End result: label each wire/edge with a register

More Pegasus!

- etas/mus
- Token edges
- Multiplexors
- Crt: current point
- Other weird stuff
Pegasus: Etas (gateways)

Pegasus: Etas (gateways)

Pegasus etas and mus

Pegasus etas and mus
Pegasus etas and mus

\[ \eta = \text{eta} = \text{gate} \]

\[ \mu = \text{mu} = \text{merge} \]
Pegasus etas and mus

\[ F \leq a \leq c \leq b \leq T \]

\[ \eta = \text{gate} \]

\[ \mu = \text{merge} \]
Pegasus token edges

- No data; they just enforce ordering between operations
- They are also wires, and go through etas and mus...
- Warning: inputs not shown in correct order

Pegasus multiplexors/predication

```plaintext
if (i)
  x++;  
else
  x--;  
  x += 100;
```

If we go ahead and execute both paths, can we get rid of some overhead?
Pegasus multiplexors/predication

if (i)
  x++;
else
  x--;
x += 100;

hyperblock

Assignment 1

Dataflow: Edges

• Our framework supports differentiated information on the outgoing edges.

• Task 1A: add data information resulting from analysis of the conditionals to our existing conditional constant propagation pass

Aggressive DCE

• Task 1B: implement ADCE: assume everything dead unless proven live
Task 1B Alternatives

- Extend CCP+1A to range analysis: a value might not be constant, but is discovered to be bounded by a range
  - This is a somewhat open-ended problem, so talk to us before you go down this route

- Perform induction variable based strength reduction in Pegasus

Admin

- TJC will have office hours tomorrow 10am-noon.
- Until I figure out 4-up landscape pdf handouts, you can look at the lectures from 15745-s05...