Register Allocation, iii

Bringing in functions & using spilling & coalescing
Function Calls

;; f(x) = let y = g(x)
;; in h(y+x) + y*5

(:f
  (x <- eax) ;; save our argument
  (call :g)  ;; call g with our argument
  (y <- eax) ;; save g's result in y
  (eax += x) ;; compute h's arg
  (call :h)  ;; call h
  (y5 <- y)  ;; compute y*5 in y5, i
  (y5 *= 5)   ;; compute y*5 in y5, ii
  (eax += y5) ;; add h's res to y*5
  (return)) ;; and we're done.
Gen & Kill

**gen**

1: :f

2: (x <- eax) (eax)

3: (call :g) (eax ecx edx)

4: (y <- eax) (eax)

5: (eax += x) (eax x)

6: (call :h) (eax ecx edx)

7: (y5 <- y) (y)

8: (y5 *= 5) (y5)

9: (eax += y5) (eax y5)

10: (return) (eax edi esi)

**kill**

1: ()

2: ()

3: (x)

4: (eax ecx edx eax)

5: (y)

6: (eax)

7: (ebx ecx edx eax)

8: (y)

9: (y5)

10: (y5)

11: (eax)

12: ()
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>()</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>()</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>()</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>()</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>()</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>()</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>()</td>
</tr>
<tr>
<td>8</td>
<td>(y5 * 5)</td>
<td>()</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>()</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>()</td>
</tr>
</tbody>
</table>
## Liveness

<table>
<thead>
<tr>
<th></th>
<th><strong>in</strong></th>
<th><strong>out</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>()</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax x)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
### Liveness

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( f )</td>
<td>()</td>
</tr>
<tr>
<td>2: ( x \leftarrow \text{eax} )</td>
<td>(eax)</td>
</tr>
<tr>
<td>3: ( \text{call} :\text{g} )</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>4: ( y \leftarrow \text{eax} )</td>
<td>(eax)</td>
</tr>
<tr>
<td>5: ( \text{eax} += x )</td>
<td>(eax x)</td>
</tr>
<tr>
<td>6: ( \text{call} :\text{h} )</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>7: ( y_5 \leftarrow y )</td>
<td>(y)</td>
</tr>
<tr>
<td>8: ( y_5 *= 5 )</td>
<td>(y5)</td>
</tr>
<tr>
<td>9: ( \text{eax} += y_5 )</td>
<td>(eax y5)</td>
</tr>
<tr>
<td>10: ( \text{return} )</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
## Liveness

<table>
<thead>
<tr>
<th></th>
<th><strong>in</strong></th>
<th><strong>out</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edx y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
### Liveness

<table>
<thead>
<tr>
<th></th>
<th><strong>in</strong></th>
<th><strong>out</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edx y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 * 5)</td>
<td>(eax y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
Liveness

```
in
1:  :f  (eax ecx edx)
2:  (x <- eax)  (eax ecx edx)
3:  (call :g)  (eax ecx edx x)
4:  (y <- eax)  (eax ecx edx x)
5:  (eax += x)  (eax ecx edx x y)
6:  (call :h)  (eax ecx edx y)
7:  (y5 <- y)  (eax y)
8:  (y5 *= 5)  (eax edi esi y5)
9:  (eax += y5)  (eax edi esi y5)
10:  (return)  (eax edi esi)

out
(eax ecx edx)
(eax ecx edx)
(eax ecx edx x)
(eax ecx edx x)
(eax ecx edx x y)
(eax ecx edx y)
(eax y)
(eax y5)
(eax edi esi y5)
(eax edi esi)
()``
Liveness

\[
\begin{align*}
1: & \quad \texttt{f} \quad (\text{eax ecx edx}) \\
2: & \quad (\texttt{x <- eax}) \quad (\text{eax ecx edx}) \\
3: & \quad (\texttt{call :g}) \quad (\text{eax ecx edx x}) \\
4: & \quad (\texttt{y <- eax}) \quad (\text{eax ecx edx x}) \\
5: & \quad (\texttt{eax += x}) \quad (\text{eax ecx edx x y}) \\
6: & \quad (\texttt{call :h}) \quad (\text{eax ecx edx y}) \\
7: & \quad (\texttt{y5 <- y}) \quad (\text{eax y}) \\
8: & \quad (\texttt{y5 *= 5}) \quad (\text{eax edi esi y5}) \\
9: & \quad (\texttt{eax += y5}) \quad (\text{eax edi esi y5}) \\
10: & \quad (\texttt{return}) \quad (\text{eax edi esi})
\end{align*}
\]
Liveness

```
in
1:  :f          (eax ecx edx)
2:  (x <- eax)  (eax ecx edx)
3:  (call :g)   (eax ecx edx x)
4:  (y <- eax)  (eax ecx edx x)
5:  (eax += x)  (eax ecx edx x y)
6:  (call :h)   (eax ecx edx y)
7:  (y5 <- y)   (eax edi esi y)
8:  (y5 *= 5)   (eax edi esi y5)
9:  (eax += y5) (eax edi esi y5)
10: (return)    (eax edi esi)
```

```
out
1:  (eax ecx edx)
2:  (eax ecx edx x)
3:  (eax ecx edx x)
4:  (eax ecx edx x y)
5:  (eax ecx edx y)
6:  (eax y)
7:  (eax edi esi y5)
8:  (eax edi esi y5)
9:  (eax edi esi)
10: ()
```
### Liveness

<table>
<thead>
<tr>
<th></th>
<th><strong>in</strong></th>
<th><strong>out</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edx x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edx y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>


## Liveness

<table>
<thead>
<tr>
<th></th>
<th><strong>in</strong></th>
<th><strong>out</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edx x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(eax edi esi)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>()</td>
</tr>
</tbody>
</table>
## Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edx x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>


Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f (eax ecx edx)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax) (eax ecx edx)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g) (eax ecx edx x)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax) (eax ecx edx x)</td>
<td>(eax ecx edx x y)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x) (eax ecx edi edx esi x y)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h) (eax ecx edi edx esi y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y) (eax edi esi y)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5) (eax edi esi y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5) (eax edi esi y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return) (eax edi esi)</td>
<td>()</td>
</tr>
</tbody>
</table>
Liveness

\begin{align*}
1: & \quad :f & (eax\ ecx\ edx) \\
2: & \quad (x \leftarrow ex) & (eax\ ecx\ edx) \\
3: & \quad (call: g) & (eax\ ecx\ edx\ x) \\
4: & \quad (y \leftarrow eax) & (eax\ ecx\ edx\ x) \\
5: & \quad (eax + x) & (eax\ ecx\ edi\ edx\ esi\ x\ y) \\
6: & \quad (call: h) & (eax\ ecx\ edi\ edx\ esi\ y) \\
7: & \quad (y5 \leftarrow y) & (eax\ edi\ esi\ y) \\
8: & \quad (y5 \ast 5) & (eax\ edi\ esi\ y5) \\
9: & \quad (eax + y5) & (eax\ edi\ esi\ y5) \\
10: & \quad (return) & (eax\ edi\ esi) \\
\end{align*}

\begin{align*}
& \quad (eax\ ecx\ edx) \\
& \quad (eax\ ecx\ edx\ x) \\
& \quad (eax\ ecx\ edx\ x) \\
& \quad (eax\ ecx\ edi\ edx\ esi\ x\ y) \\
& \quad (eax\ ecx\ edi\ edx\ esi\ y) \\
& \quad (eax\ edi\ esi\ y) \\
& \quad (eax\ edi\ esi\ y5) \\
& \quad (eax\ edi\ esi\ y5) \\
& \quad (eax\ edi\ esi) \\
& \quad ()
\end{align*}
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f (eax ecx edx)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax) (eax ecx edx)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g) (eax ecx edx x)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax) (eax ecx edi edx esi x)</td>
<td>(eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x) (eax ecx edi edx esi x y)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h) (eax ecx edi edx esi y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y) (eax edi esi y)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5) (eax edi esi y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5) (eax edi esi y5)</td>
<td>(eax edi esi)</td>
</tr>
<tr>
<td>10</td>
<td>(return) (eax edi esi)</td>
<td>()</td>
</tr>
</tbody>
</table>
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edx x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
Liveness

```
in
1:  :f  (eax ecx edx)
2:  (x <- eax)  (eax ecx edx)
3:  (call :g)  (eax ecx edi edx esi x)
4:  (y <- eax)  (eax ecx edi edx esi x)
5:  (eax += x)  (eax ecx edi edx esi x y)
6:  (call :h)  (eax ecx edi edx esi y)
7:  (y5 <- y)  (eax edi esi y)
8:  (y5 *= 5)  (eax edi esi y5)
9:  (eax += y5)  (eax edi esi y5)
10: (return)  (eax edi esi)

out
(eax ecx edx)
(eax ecx edx x)
(eax ecx edi edx esi x)
(eax ecx edi edx esi x y)
(eax ecx edi edx esi y)
(eax edi esi y)
(eax edi esi y5)
(eax edi esi y5)
(eax edi esi)
()
```
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>

20
## Liveness

**in**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Out Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$f$</td>
<td>(eax ecx edx)</td>
</tr>
<tr>
<td>2:</td>
<td>$(x \leftarrow \text{eax})$</td>
<td>(eax ecx edi edx esi)</td>
</tr>
<tr>
<td>3:</td>
<td>(call :g)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>4:</td>
<td>$(y \leftarrow \text{eax})$</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>5:</td>
<td>(eax $+= x$)</td>
<td>(eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>6:</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7:</td>
<td>$(y5 \leftarrow y)$</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8:</td>
<td>$(y5 \ast= 5)$</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9:</td>
<td>(eax $+= y5$)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10:</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>f (eax ecx edx)</td>
<td>(eax ecx edi edx esi)</td>
</tr>
<tr>
<td>2:</td>
<td>(x &lt;- eax) (eax ecx edi edx esi)</td>
<td>(eax ecx edi edx esi x) (eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>3:</td>
<td>(call :g) (eax ecx edi edx esi x)</td>
<td>(eax ecx edi edx esi x) (eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>4:</td>
<td>(y &lt;- eax) (eax ecx edi edx esi x)</td>
<td>(eax ecx edi edx esi x) (eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>5:</td>
<td>(eax += x) (eax ecx edi edx esi x y)</td>
<td>(eax ecx edi edx esi x y) (eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>6:</td>
<td>(call :h) (eax ecx edi edx esi y)</td>
<td>(eax ecx edi edx esi y) (eax edi esi y)</td>
</tr>
<tr>
<td>7:</td>
<td>(y5 &lt;- y) (eax edi esi y)</td>
<td>(eax edi esi y) (eax edi esi y5)</td>
</tr>
<tr>
<td>8:</td>
<td>(y5 *= 5) (eax edi esi y5)</td>
<td>(eax edi esi y5) (eax edi esi y5)</td>
</tr>
<tr>
<td>9:</td>
<td>(eax += y5) (eax edi esi y5)</td>
<td>(eax edi esi y5) (eax edi esi)</td>
</tr>
<tr>
<td>10:</td>
<td>(return) (eax edi esi)</td>
<td>()</td>
</tr>
</tbody>
</table>
Liveness

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:f</td>
<td>(eax ecx edi edx esi)</td>
</tr>
<tr>
<td>2</td>
<td>(x &lt;- eax)</td>
<td>(eax ecx edi edx esi)</td>
</tr>
<tr>
<td>3</td>
<td>(call :g)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>4</td>
<td>(y &lt;- eax)</td>
<td>(eax ecx edi edx esi x)</td>
</tr>
<tr>
<td>5</td>
<td>(eax += x)</td>
<td>(eax ecx edi edx esi x y)</td>
</tr>
<tr>
<td>6</td>
<td>(call :h)</td>
<td>(eax ecx edi edx esi y)</td>
</tr>
<tr>
<td>7</td>
<td>(y5 &lt;- y)</td>
<td>(eax edi esi y)</td>
</tr>
<tr>
<td>8</td>
<td>(y5 *= 5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>9</td>
<td>(eax += y5)</td>
<td>(eax edi esi y5)</td>
</tr>
<tr>
<td>10</td>
<td>(return)</td>
<td>(eax edi esi)</td>
</tr>
</tbody>
</table>
Liveness

:f
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(return)
Spilling y

Before:

: f
(x <- eax)
(call : g)
(y <- eax)
(eax += x)
(call : h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(return)

After:

: f
(x <- eax)
(call : g)
((mem ebp -4) <- eax)
(eax += x)
(call : h)
(y5 <- (mem ebp -4))
(y5 *= 5)
(eax += y5)
(return)
Spilling y

:f
(x <- eax)
(call :g)
((mem ebp -4) <- eax)
(eax += x)
(call :h)
(y5 <- (mem ebp -4))
(y5 *= 5)
(eax += y5)
(return)
Spilling x

Before:
:f
(x <- eax)
(call :g)
((mem ebp -4) <- eax)
(eax += x)
(call :h)
(y5 <- (mem ebp -4))
(y5 *= 5)
(eax += y5)
(return)

After:
:f
(((mem ebp -8) <- eax)
(call :g)
(((mem ebp -4) <- eax)
(sx0 <- (mem ebp -8))
(eax += sx0)
(call :h)
((y5 <- (mem ebp -4))
(y5 *= 5)
(eax += y5)
(return)

Note that this time we introduce a sx0, but compare its live range to x’s
Spilling x

:f
((mem ebp -8) <- eax)
(call :g)
((mem ebp -4) <- eax)
(sx0 <- (mem ebp -8))
(eax += sx0)
(call :h)
(y5 <- (mem ebp -4))
(y5 *= 5)
(eax += y5)
(return)
Live ranges

:f
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(return)
Live ranges

We spilled two variables that have relatively short live ranges, but look at those long live ranges with no uses of the variables that the callee save registers, i.e. edi and esi, have. We’d rather spill them.
Spilling callee saves

Unfortunately, it makes no sense to spill real registers. Instead, a trick: we just make up new variables to hold their values. Semantics of the program does not change, but:

• Now the real registers now have short live ranges, and

• New temporaries are spillable
Adding new variables g-ans

**Before:**

```
:f
  (x <- eax)
  (call :g)
  (y <- eax)
  (eax += x)
  (call :h)
  (y5 <- y)
  (y5 *= 5)
  (eax += y5)
  (return)
```

**After:**

```
:f
  (z1 <- edi)
  (z2 <- esi)
  (x <- eax)
  (call :g)
  (y <- eax)
  (eax += x)
  (call :h)
  (y5 <- y)
  (y5 *= 5)
  (eax += y5)
  (edi <- z1)
  (esi <- z2)
  (return)
```

Init new variables at beginning of fun, restore them before returning or making a tail call.
Live ranges with new variables

:f
(z1 <- edi)
(z2 <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(edi <- z1)
(esi <- z2)
(return)
Spilling z1 & z2

Before:
: f
(z1 <- edi)
(z2 <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(edi <- z1)
(esi <- z2)
(return)

After:
: f
((mem ebp -4) <- edi)
((mem ebp -8) <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(edi <- (mem ebp -4))
(esi <- (mem ebp -8))
(return)
How to choose spills

• Pick variables with long live ranges and few uses to spill (callee saves have this profile)

• In this case, we can spill $z_1$ and $z_2$ to save us from spilling variables $x$ and $y$ that are frequently accessed
Finishing up register allocation

When you’ve spilled enough to successfully allocate the program, count the number of spills and adjust esp at the beginning of the function. (Recall calling convention.)
Calling convention reminder (see lecture03.txt)

\[(\text{call s}) \Rightarrow \text{pushl }\$<\text{new-label}>\]
\[
\text{pushl }\%\text{ebp} \\
\text{movl }\%\text{esp}, \%\text{ebp} \\
\text{jmp } <s> \\
<\text{new-label}>: \\
\]

\[(\text{return}) \Rightarrow \text{movl }\%\text{ebp}, \%\text{esp} \\
\text{popl }\%\text{ebp} \\
\text{ret } // \text{ pop } \& \text{ goto} \]

\[(\text{tail-call s}) \Rightarrow \text{movl }\%\text{ebp}, \%\text{esp} \\
\text{jmp } <s> \\
\text{(cleaned up version of call followed by return)}\]
Registers: allocated

(:f
  (esp -= 8)
  ((mem ebp -4) <- esi)
  ((mem ebp -8) <- edi)
  (esi <- eax)
  (call :g)
  (edi <- eax)
  (eax += esi)
  (call :h)
  (ebx <- edi)
  (ebx *= 5)
  (eax += ebx)
  (esi <- (mem ebp -4))
  (edi <- (mem ebp -8))
  (return))
Coalescing

If we see a \((x \leftarrow y)\) instruction, we might be able to just change all of the \(x\)'s into \(y\)'s

That’s called coalescing \(x\) and \(y\)

Let's use coalescing to remove \(y\) and \(y5\) from our example program
Coalescing example

:f
((mem ebp -4) <- edi)
((mem ebp -8) <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y5 <- y)
(y5 *= 5)
(eax += y5)
(edi <- (mem ebp -4))
(esi <- (mem ebp -8))
(return)

:f
((mem ebp -4) <- edi)
((mem ebp -8) <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y <- y)
(y *= 5)
(eax += y)
(edi <- (mem ebp -4))
(esi <- (mem ebp -8))
(return)
Live ranges before coalescing

:f
((mem ebp -4) <- edi)
((mem ebp -8) <- esi)
(x <- eax)
call :g
(y <- eax)
(eax += x)
call :h
(y5 <- y)
(y5 *= 5)
(eax += y5)
(edi <- (mem ebp -4))
(esi <- (mem ebp -8))
(return)
Live ranges after coalescing

:f
((mem ebp -4) <- edi)
((mem ebp -8) <- esi)
(x <- eax)
(call :g)
(y <- eax)
(eax += x)
(call :h)
(y <- y)
(y *= 5)
(eax += y)
(edi <- (mem ebp -4))
(esi <- (mem ebp -8))
(return)
Let's try to register allocate the coalesced graph
It worked that time, but this doesn’t always work
Coalescing Problem

(r:a <- 1)
(r:z <- r:a)
(r:1 <- 1)
(r:2 <- 2)
(r:3 <- 3)
(r:a += 1)
(r:4 <- 4)
(r:5 <- 5)
(r:6 <- 6)
(r:1 += 1)
(r:2 += 1)
(r:3 += 1)
(r:z <- 1)
(r:4 += 1)
(r:5 += 1)
(r:6 += 1)
(r:z += 1)
Now, we coalesce $r:a$ and $r:z$
Coalescing Problem

```
(r:a <- 1)
(r:z <- r:a)
(r:1 <- 1)
(r:2 <- 2)
(r:3 <- 3)
(r:a += 1)
(r:4 <- 4)
(r:5 <- 5)
(r:6 <- 6)
(r:1 += 1)
(r:2 += 1)
(r:3 += 1)
(r:z <- 1)
(r:4 += 1)
(r:5 += 1)
(r:6 += 1)
(r:z += 1)
```

r:1  r:2  r:3  r:4  r:5  r:6  r:a  r:z eax ebx ecx edi edx esi
Coalescing Problem

(r:z <- 1)
(r:z <- r:z)
(r:1 <- 1)
(r:2 <- 2)
(r:3 <- 3)
(r:z += 1)
(r:4 <- 4)
(r:5 <- 5)
(r:6 <- 6)
(r:1 += 1)
(r:2 += 1)
(r:3 += 1)
(r:z <- 1)
(r:4 += 1)
(r:5 += 1)
(r:6 += 1)
(r:z += 1)
Coalescing during graph coloring

Extend the interference graph with a new kind of edge, called a move edge.

A move edge connects two nodes if there is a $(x \leftarrow y)$ instruction in the program.

Combine two move-edge connected nodes into a single node (at any step in the coloring algorithm), if:

• They don’t interfere
• The resulting node has fewer than 6 neighbors

This ensures the graph is colorable (if it was before)
Roadmap: putting it all together
Programming

There are a number of different modules to put together

• A liveness library: gen & kill functions on instructions; the in and out loop; going from liveness to interference

• A graph library: creating graphs, creating nodes, creating edges, removing nodes (and their edges), iterating over edges and nodes

• An interference library: build a graph from the liveness information
Programming

There are a number of different modules to put together, co td

• A coloring library: color a graph using the coloring algorithm

• A spilling library: given a variable and a stack position, rewrite the program to move the variable in and out of the stack right as it is used

• The final translation: When you have a valid coloring, rewrite the variables to use registers and insert the esp adjustment, turning the L2 program into an L1 one.
Unit testing

The most underrated part of developing good software is testing it well.

• Build simple (unit) tests for the api for each module, as you design the API

• As you write the code, write a test for each different case in the code together with the case itself

• Whenever you find a bug, always add a test case before fixing the bug; make sure the test case fails so you know you wrote it properly

• Testing only the overall L2 → L1 translation is a sure way to fail.