Function mult (c, a, b, r)

```
while c > 0 do
  var i = 0
  var r1 = 0
  while i < a do
    r1 = r1 + b
    i = i + 1
  end
  c = c - 1
end
```

Calling conventions:
```
ret
```

Virtual register x64:
```
vr1
```

Stack call x64:
```
%ebp
```

```
```

Code:
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Lecture topics:
I Register allocation
II Intro to optimizations
III Constant folding

I Register allocation
(see first page)

II Intro to optimizations

Big picture

High-level IR \xrightarrow{\text{Portable optimizer}}

Code generator

Low-level IR \xrightarrow{\text{Machine dependent optimizer}}

Printer

Target code

Typical optimizations

<table>
<thead>
<tr>
<th>Name</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant folding</td>
<td>h := 60×60</td>
<td>h := 3600</td>
</tr>
<tr>
<td>Copy propagation</td>
<td>t := h</td>
<td>t := h</td>
</tr>
<tr>
<td></td>
<td>d := 24×t</td>
<td>d := 24×h</td>
</tr>
<tr>
<td>Dead code elimination</td>
<td>t := h</td>
<td>d := 24×h</td>
</tr>
<tr>
<td></td>
<td>d := 24×h</td>
<td>return d</td>
</tr>
<tr>
<td>Common subexpression elimination</td>
<td>h := 60×60</td>
<td>h := 60×60</td>
</tr>
<tr>
<td></td>
<td>d := 24×h</td>
<td>d := 24×h</td>
</tr>
<tr>
<td>Algebraic simplification</td>
<td>while i &lt; n do (…)</td>
<td>while i &lt; n do (…)</td>
</tr>
<tr>
<td>Strength reduction</td>
<td>x := y + y</td>
<td>x := y + y</td>
</tr>
<tr>
<td></td>
<td>z := y / 4</td>
<td>z := y / 4</td>
</tr>
<tr>
<td>Loop invariant code motion</td>
<td>while i &lt; n-1 do (…)</td>
<td>while i &lt; n-1 do (…)</td>
</tr>
</tbody>
</table>

III Constant folding

Function fadd(int):int

```java
int fadd(int n, int r, int t)
    return r + t;
```

After constant folding:

```java
int fadd(int n, int r, int t)
    return r + t;
```

Reminders:
- hw 8 due 11/28
- pr 3 moved to 12/5 (IR code gen)
- pr 4 canceled (>64 code gen)