void vadd (int A[], int B[], int C[], int N)
{
    int i;
    for(i = 0; i < N; i++) // assert N > 0;
        A[i] = B[i] + C[i];
}

--- That COULD generate the following (hypothetical machine) assembler ---

L1:
    r11 = load 8(fp)   # load local "i"
    r12 = load -16(fp) # load parm "N"
    r9  = r12 - r11    # N - i
    beq L3              # branch on ==

L2:
    r13 = load 8(fp)   # load local "i"
    r14 = r13 * 4      # t1 = i * 4
    r15 = r14 + B      # t2 = t1 + &B
    r25 = load 0(r15)  # load B[i]

    r16 = load 8(fp)   # load local "i"
    r17 = r16 * 4      # t3 = i * 4
    r18 = r17 + C      # t4 = t3 + &C
    r26 = load 0(r18)  # load C[i]

    r19 = r25 + r26    # t5 = B[i] + C[i]
    r20 = load 8(fp)   # load local "i"
    r21 = r20 * 4      # t6 = i * 4
    r22 = r21 + A      # t7 = t6 + &A
    store r19, 0(r22)  # store A[i]

    r23 = load 8(fp)   # load local "i"
    r24 = r23 + 1      # i = i + 1
    store r24, 8(fp)   # store i
    br L1               # branch to loop’s top

L3:
    # Whatever comes next
***********  First Improvement --- Register Assignment ***********

# Assume i stored in r29, N in r30

| L1: | r9  = r30 - r29  # N - i  
|     | beq L3          # branch on == |
|     |----------------------------------------------------
| L2: | r14 = r29 * 4  # t1 = i * 4 
|     | r15 = r14 + B   # t2 = t1 + &B 
|     | r25 = load 0(r15) # load B[i] 
|     | r17 = r29 * 4  # t3 = i * 4 
|     | r18 = r17 + C   # t4 = t3 + &C 
|     | r26 = load 0(r18) # load C[i] 
|     | r19 = r25 + r26 # t5 = B[i] + C[i] 
|     | r21 = r29 * 4  # t6 = i * 4 
|     | r22 = r21 + A   # t7 = t6 + &A 
|     | store r19, 0(r22) # store A[i] 
|     | r29 = r29 + 1  # i = i + 1 
|     | br  L1          # branch to loop's top |
|     |----------------------------------------------------

| L3: | # Whatever comes next |
|     |----------------------------------------------------|
********  Next, note we compute $i \times 4$ several times  ********

L1:

```
| r9 = r30 - r29          # N - i |
| beq L3                  # branch on == |
```

L2:

```
| r14 = r29 * 4           # t1 = i * 4 |
| r15 = r14 + B           # t2 = t1 + &B |
| r25 = load 0(r15)       # load B[i] |
| r18 = r14 + C           # t4 = t1 + &C |
| r26 = load 0(r18)       # load C[i] |
| r19 = r25 + r26         # t5 = B[i] + C[i] |
| r22 = r14 + A           # t7 = t1 + &A |
| store r19, 0(r22)       # store A[i] |
| r29 = r29 + 1           # i = i + 1 |
| br L1                   # branch to loop's top |
```

L3:

```
# Whatever comes next
```
******** Next, note we don’t need compute \( i \times 4 \) in loop at all ********

L0:

\[
\begin{align*}
\text{r21} &= \text{A} & \# \text{r21} &= \&(\text{A}[0]) \\
\text{r22} &= \text{B} & \# \text{r22} &= \&(\text{B}[0]) \\
\text{r23} &= \text{C} & \# \text{r23} &= \&(\text{C}[0]) \\
\end{align*}
\]

L1:

\[
\begin{align*}
\text{r9} &= \text{r30} - \text{r29} & \# \text{N} - i \\
\text{beq} \ L3 & & \# \text{branch on ==}
\end{align*}
\]

L2:

\[
\begin{align*}
\text{r25} &= \text{load} \ 0(\text{r22}++) & \# \text{load B}[i], \text{auto-inc} \\
\text{r26} &= \text{load} \ 0(\text{r23}++) & \# \text{load C}[i], \text{auto-inc} \\
\text{r19} &= \text{r25} + \text{r26} & \# t5 = \text{B}[i] + \text{C}[i] \\
\text{store} \ \text{r19}, \ 0(\text{r21}++) & \# \text{store A}[i], \text{auto-inc} \\
\text{r29} &= \text{r29} + 1 & \# i = i + 1 \\
\text{br} \ \text{L1} & & \# \text{branch to loop’s top}
\end{align*}
\]

L3:

\[
\begin{align*}
\text{# Whatever comes next}
\end{align*}
\]
Finally, using compiler voodoo we get

<table>
<thead>
<tr>
<th>L0:</th>
</tr>
</thead>
<tbody>
<tr>
<td>r21 = A</td>
</tr>
<tr>
<td>r22 = B</td>
</tr>
<tr>
<td>r23 = C</td>
</tr>
<tr>
<td>r28 = r30 * 4</td>
</tr>
<tr>
<td>r29 = r28 + A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>r25 = load 0(r22++)</td>
</tr>
<tr>
<td>r26 = load 0(r23++)</td>
</tr>
<tr>
<td>r19 = r25 + r26</td>
</tr>
<tr>
<td>store r19, 0(r21++)</td>
</tr>
<tr>
<td>r9  = r29 - r21</td>
</tr>
<tr>
<td>bgt L2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L3:</th>
</tr>
</thead>
<tbody>
<tr>
<td># Whatever comes next</td>
</tr>
</tbody>
</table>