Normal forms resembling machine code

Computational machine instructions resemble C assignments, except they have a very limited form:

- At most one operator appears on the right-hand side.
- If there is a binary operator, the variable assigned to is the same as the right-hand argument.
- If an access to memory is involved, typically there is no operator.

Some examples:

\[
\begin{align*}
& y = x + y; & y = 17; \quad \text{// load 'immediate'} \\
& y = x - y; & y = \text{Array2\_map\_row\_major}; \\
& y = m[x+12]; \quad \text{// memory access: load} & y = (\text{double}) x; \\
& m[rsp-4] = x; \quad \text{// memory access: store} \\
\end{align*}
\]

Translation into this form is simple:

- For a complex expression like \(a \times (b + c)\), simplify by first storing \((b + c)\) in a variable.
- For a “three-address” expression like \(z = x + y\);, translate to two instructions:

\[
\begin{align*}
& z = y; \\
& z = x + z;
\end{align*}
\]

Translation problem

Floating-point parameters are passed in registers \(\%\text{xmm0} - \%\text{xmm7}\), and a floating-point result is returned in register \(\%\text{xmm0}\). Translate this procedure into normal form:

\[
\begin{align*}
\text{float luminance(float red, float green, float blue) \{}
& \quad \text{return } 0.299 \ast \text{red} + 0.587 \ast \text{green} + 0.114 \ast \text{blue;}
\}\]

(bonus problem on the back)
Bonus translation problem

Integer parameters are passed in registers %rdi, %rsi, %rdx, %rcx, %r8, and %r9. Translate this procedure into normal form:

```c
/* squared difference of scaled integers; denominators may differ */
double sqdiff(int n1, int d1, int n2, int d2) {
    double diff = (double)n1/(double)d1 - (double)n2/(double)d2;
    return diff * diff;
}
```