Problem 1.3:
-------------

There are two main translation steps required: The COMPILER converts the program in the high-level language to assembly language, and the ASSEMBLER then converts the assembly language program to machine language.

Problem 1.6:
------------

The program has 1.0E6 instructions divided into four classes: 10% class A, 20% class B, 50% class C, and 20% class D. P1 has a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3. P2 has a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2.

Part a)

It's possible to calculate the CPI for each processor without calculating the runtime:

For P1, it's (10% x 1 + 20% x 2 + 50% x 3 + 20% x 3) --> 2.6 cycles/inst
For P2, it's (10% x 2 + 20% x 2 + 50% x 2 + 20% x 2) --> 2.0 cycles/inst

Part b)

The number of clock cycles required is the CPI from above times the number of instructions. There 1x10^6 instructions, so the number of clock cycles for P1 is 2.6E6 and for P2 it's 2.0E6.

Faster?

Neither part a) nor part b) ask us to determine runtimes, though the initial problem description does ask which implementation is faster. P1's runtime is:

\[ \frac{2.6 \times 10^6 \text{ cycles}}{2.5 \times 10^9 \text{ cycles/sec}} = 1.04 \times 10^{-3} \text{ seconds} \]

For P2 it's 2.0E6 / 3.0E9 = 6.66 x 10^-4 seconds. P2 wins.

Problem 1.7:
------------

Compiler A produces 1.0E9 instructions, and the program runs in 1.1 seconds. Compiler B produces 1.2E9 instructions, and runs in 1.5 seconds. Instead of giving us the clock frequency, they've specified the clock time (or clock *period*) as 1E-9 seconds. (The clock rate is therefore 1/1E-9 or 1.0 GHz.)

Part a)

Since the clock period is the inverse of the clock rate, our performance equation can be rewritten as:

\[ \text{runtime} = \text{CPI} \times \# \text{ instructions} \times \text{clock period} \]

We're after CPI in this case, so we can rewrite that again as:

\[ \frac{\text{runtime}}{\# \text{ instructions} \times \text{clock period}} = \text{CPI} \]
Plugging in the values for Compiler A:

\[
1.1 \text{ seconds} \\
\text{CPI} = \frac{1.1 \text{ cycles/instruction}}{1.0 \times 10^9 \text{ instructions} \times 1.0 \times 10^{-9} \text{ seconds/cycle}} = 1.1 \text{ cycles/instruction}
\]

For Compiler B we get 1.25 cycles/instruction.

Part b)

We don't know what the runtime is, but we know it's the same on both systems. We can rearrange our performance equation again to get:

\[
\frac{\text{runtime}}{\text{clock period}} = \frac{1}{\text{CPI} \times \#\text{instructions}}
\]

The problem wants us to determine the ratio of A's clock cycle time (that is, clock period) to B's clock cycle. When we take that ratio, the runtimes will cancel out:

\[
\frac{\text{runtime}}{\text{clock period A}} = \frac{\text{CPI}_A \times \#\text{inst}_A}{\text{CPI}_B \times \#\text{inst}_B}
\]

\[
\frac{\text{clock period B}}{\text{runtime}} = \frac{\text{CPI}_A \times \#\text{inst}_A}{\text{CPI}_B \times \#\text{inst}_B}
\]

Plugging in values, we get a ratio of 1.37. That is, A's period is 1.37 times as long as B's.

Part c)

There are multiple ways to set this up: We could find the ratios of just the number of cycles required for each scenario, or the ratio of the runtimes. Since they give us the runtimes for the first two programs, let's just find the runtime for this new scenario and then find ratios. The output from the new compiler has a runtime of:

\[
6.0 \times 10^8 \text{ inst.} \times 1.1 \text{ cycles/inst.} \times 1 \times 10^{-9} \text{ sec/cycle} = 0.66 \text{ seconds}
\]

Speedup over A is therefore \(1.1/0.66 = 1.67\)

Speedup over B is \(1.5/0.66 = 2.27\)