Consider a CPU that implements two parallel fetch-execute pipelines for superscalar processing. Show the performance improvement over scalar pipeline processing and no-pipeline processing, assuming an instruction cycle similar to figure 4.1 in the commentary, i.e.:

* a one clock cycle fetch
* a one clock cycle decode
* a four clock cycle execute

for:

a. a 10-instruction sequence

b. a 100-instruction sequence

**Answer**

a. For a ten-instruction sequence:

* **no pipelining** would require:

10 instructions x 6 clock cycles/instruction = 60 clock cycles

* **a scalar pipeline** would require:

6 clock cycles for the first instruction, plus one additional clock cycle for each of the other nine instructions:

6 clock cycles + (9 instructions x 1 clock cycle/instruction) = 15 clock cycles

* **a superscalar pipeline** with two parallel units would require:

6 cycles for the first set of two instructions, plus one additional clock cycle for each of the other four instruction sets:

6 clock cycles + (4 instructions x 1 clock cycle/instruction) = 10 clock cycles

b. For a 100-instruction sequence:

* **no pipelining** would require:

100 instructions x 6 clock cycles/instruction = 600 clock cycles

* **a scalar pipeline** would require:

6 clock cycles for the first instruction, plus one additional clock cycle for each of the other 99 instructions:

6 clock cycles + (99 instructions x 1 clock cycle/instruction) = 105 clock cycles

* **a superscalar pipeline** with two parallel units would require:

6 cycles for the first set of two instructions, plus one additional clock cycle for each of the other 49 instruction sets:

6 clock cycles + (49 instructions x 1 clock cycle/instruction) = 55 clock cycles