Pipelining Example

The machine language from our textbook has a machine cycle consisting of 3 stages—Fetch, Decode, and Execute. Each stage takes one clock cycle (or step) to execute. This means that in order to execute 5 instructions, it takes 15 clock cycles:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F	D	E	F	D	E	F	D	E	F	D	E	F	D	E

Pipelining takes advantage of the fact that the each stage of a machine cycle can be implemented in separate hardware. This allows the stages of the machine cycle to be performed on different instructions at the same time. Thus, while one instruction is being fetched, another is being decoded, and yet another is being executed.
Continuing the example from above, if we execute the same 5 instructions using pipelining, it only takes 7 steps:

1	2	3	4	5	6	7
F	D	E				
	F	D	E			
		F	D	E		
			F	D	E	
				F	D	E

Notice that pipelining is about 15/7≈2.143 times faster than normal execution when executing 5 instructions. As the number of instructions increases, we approach a limit of how much faster pipelining is than normal execution. We call this limit the *speedup*. What is the speedup for our machine cycle with three stages? I'll give you a hint: it is *not* 2. Do calculations like the ones above for normal and pipelined execution for 10 instructions, then 100. Then you should be able to see a pattern that leads to formulas for n instructions. Based on this, you can determine the correct speedup.

The idea of pipelining can be extended to machine cycles with any number of stages (most real-word CPUs have between 2 and 7 stages, with some as high as 31). For a CPU with k stages, it is not too difficult to develop formulas for:

- $N_k(n)$ = number of steps to execute n instructions normally.
- $P_k(n)$ = number of steps to execute n instructions using pipelining.
- $F_k(n)$ = how much faster pipelining is than normal execution of n instructions.
- S_k = the speedup of pipelining over normal execution.

Once you have $N_k(n)$ and $P_k(n)$ you can easily compute $F_k(n)$. Finally, S_k can be determined by taking the limit as n approaches infinity of $F_k(n)$. Don't worry if you don't understand limits—instead think about what value S_k is getting closer to as n get larger and larger. The details are left to the reader. Try to compute the formulas for k=3 and k=7, for instance.