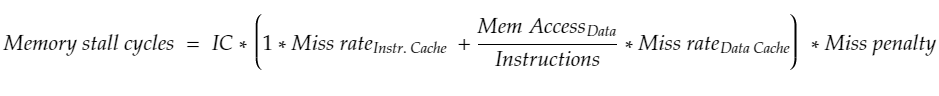
**QUESTION 2**

First, I’ll calculate the speed of the processor that has misses:



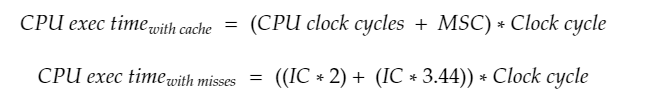
We know the miss rate of the instruction cache is 2%. For the data cache, we know that 36% of the instructions are load/stores, which means that 36% of instructions access the data cache, and this data cache has a miss rate of 4%.

So, our equation above will have a slightly different form:









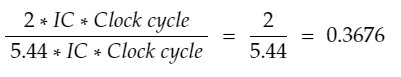


Now to get the CPU execution time if it has no misses at all:





Now as a ratio of each other:



This means that a CPU with a perfect cache will run at 0.3676 the time it would take for the CPU with misses to execute, the CPU with perfect cache is much faster.

**QUESTION 3**

**Part 1**

Access time to cache to check if it is present is 2.5ns. Since its not present, now we must access main memory. This is a 50 ns penalty (52.5ns running total) to get the first 4 bytes from memory. A line in cache is 64 bytes, so this means that we need to get 60/4 more words, which take 5 ns each. So this is 15 \* 5 = 75 ns, which gives us a running total so far of 127.5 ns. We are asked to assume that the cache waits until the line is fetched from memory and re-executes for a hit, so we assume this re-execution takes the initial access time of 2.5 ns. This gives us a final total of **130 ns**, which is our miss penalty.

**Part 2**

Average memory access time = Hit time + Miss rate \* Miss penalty

We know that the miss penalty for a line size of 64 bytes was 130 ns.

To find the miss penalty for 128 bytes, we use the same formula from Part 1, but swap out the 60/4 part with 124/4. So that’s 31 \* 5 = 155 ns. Adding 52.5 ns and 2.5 ns, we get a total miss penalty of 210 ns for 128 bytes.

Avg Mem Access Time (64 byte line) = 2.5ns + 0.05 \* 130ns = 9 ns

Avg Mem Access Time (128 byte line) = 2.5ns + 0.03 \* 210ns = 8.8 ns

So yes, increasing the line size to 128 reduces the average memory access time.

**QUESTION 4**

1. It takes 1 clock cycle to send an address, and 4 to read the value in the address and transfer it to cache. Therefore 5 clock cycles are the miss-penalty here, as we have to do both.
2. This case is the same as the miss penalty of Part 1 multiplied by the number of words, as the lack of a burst-mode means that we have a send an address each time. Therefore, miss penalty is 20 clock cycles.
3. We get the result from Part 1 and add 3 for the 3 subsequent words read from memory, as we don’t need to send the address again and the subsequent words don’t have the 4 cycle penalty for them, so 8 clock cycles total.