Problem M3.8: Multithreading [?? Hours]

Problem M3.8.A

Since there is no penalty for conditional branches, instructions take one cycle to execute unless there is a dependency problem. The following table summarizes the execution time for each instruction. From the table, the loop takes **104 cycles** to execute.

Instruction	Start Cycle	End Cycle
`LW R3, 0(R1)`	1	100
`LW R4, 4(R1)`	2	101
`SEQ R3, R3, R2`	101	101
`BNEZ R3, End`	102	102
`ADD R1, R0, R4`	103	103
`BNEZ R1, Loop`	104	104

Problem M3.8.B

If we have N threads and the first load executes in cycle 1, `SEQ`, which depends on the load, executes in cycle $2 \cdot N + 1$. To fully utilize the processor, we need to hide the 100-cycle memory latency, $2 \cdot N + 1 \geq 101$. The minimum number of thread needed is **50**.

Problem M3.8.C

	Throughput	**Latency**
Better	√	
Same		
Worse		√

Problem M3.8.D

In steady state, each thread can execute 6 instructions (`SEQ`, `BNEZ`, `ADD`, `BNEZ`, `LW`, `LW`). Therefore, to hide 99 cycles between the second `LW` and `SEQ`, a processor needs $\lceil 99/6 \rceil + 1 = 18$ threads.

Problem M3.9: Multithreaded architectures [?? Hours]

Problem M3.9.A

4, since the largest latency for any instruction is 4.

Problem M3.9.B

2/12 = 0.17 flops/cycle, on average we complete a loop every 12 cycles

Problem M3.9.C

Yes, we can hide the latency of the floating point instructions by moving the add instructions in between floating point and store instructions – we'd only need 3 threads. Moving the third load up to follow the second load would further reduce thread requirement to only 2.

Problem M3.10: Multithreading [?? Hours]

Problem M3.10.A

Fixed Switching: 6 Thread(s)

If we have N threads and L.D. executes in cycle 1, FADD, which depends on the load executes in cycle 2N + 1. To fully utilize the processor, we need to hide 12-cycle memory latency, $2N + 1 \geq 13$. The minimum number of thread needed is 6.

Data-dependent Switching: 4 Thread(s)

In steady state, each thread can execute 4 instructions (FADD, BNE, LD, ADDI). Therefore, to hide 11 cycles between ADDI and FADD, a processor needs $\lceil 11/4 \rceil + 1 = 4$ threads.

Problem M3.10.B

Fixed Switching: 2 Thread(s)

Each FADD depends on the previous iteration's FADD. If we have N threads and the first FADD executes in cycle 1, the second FADD executes in cycle 4N + 1. To fully utilize the processor, we need to hide 5-cycle latency, $4N + 1 \geq 6$. The minimum number of thread needed is 2.

Data-dependent Switching: 2 Thread(s)

In steady state, each thread can execute 4 instructions (FADD, BNE, LD, ADDI). Therefore, to hide 2 cycles between ADDI and FADD, a processor needs $\lceil 2/4 \rceil + 1 = 2$ threads.

Problem M3.10.C

Consider a **Simultaneous Multithreading (SMT)** machine with limited hardware resources. **Circle** the following hardware constraints that can limit the total number of threads that the machine can support. For the item(s) that you circle, **briefly describe** the minimum requirement to support **N** threads.

(A) Number of Functional Unit	Since not all the treads are executed in each cycle, the number of functional unit is not a constraint that limits the total number of threads that the machine can support.
(B) Number of Physical Registers	We need at least [N × (number of architecture registers) + 1] physical registers.
(C) Data Cache Size	This is for performance reasons.
(D) Data Cache Associatively	This is for performance reasons.