(b) [10 points] When the given code segment is executed on Machine II, dependencies between instructions are resolved in hardware. Explain **when** data is forwarded and **which instructions** are stalled and **when** they are stalled.

In every iteration, data are forwarded for `sw` and for `bne`. The instruction `sw` is dependent on `lw`, so it is stalled one cycle in every iteration

(c) [5 points] Calculate the *machine code size* of the code segments executed on Machine I (part (a)) and Machine II (part (b)).

Machine I: Machine I - 20 bytes (because of the additional `nop`)

Machine II: Machine II - 16 bytes

(d) [7 points] Calculate the number of cycles it takes to execute the code segment on Machine I and Machine II.

Machine I:

The compiler reorders instructions and places one `nop`. This is the execution timeline of the first iteration:

1	2	3	4	5	6	7	8	9
F	D	E	M	W				
	F	D	E	M	W			
		N	N	N	N	N		
			F	D	E	M	W	
				F	D	E	M	W

9 cycles for one iteration. As there are 5 instructions in each iteration and 25 iterations, the total number of cycles is 129 cycles.

Machine II:

The machine stalls `sw` one cycle in the decode stage. This is the execution timeline of the first iteration:

1	2	3	4	5	6	7	8	9
F	D	E	M	W				
	F	D	D	E	M	W		
		F	F	D	E	M	W	
				F	D	E	M	W

9 cycles for one iteration. As there are 4 instructions in each iteration and 25 iterations, and one stall cycle in each iteration, the total number of cycles is 129 cycles.

(e) [3 points] Which machine is faster for this code segment? Explain.

For this code segment, both machines take the same number of cycles. We cannot say which one is faster, since we do not know the clock frequency.

7 Out-of-order Execution [45 points]

In this problem, we will give you the state of the Register Alias Table (RAT) and Reservation Stations (RS) for an out-of-order execution engine that employs Tomasulo's algorithm, as we discussed in lectures. Your first task is to determine the original sequence of **four instructions** in program order.

The out-of-order machine in this problem behaves as follows:

- The frontend of the machine has a one-cycle fetch stage and a one-cycle decode stage. The machine can fetch one instruction per cycle, and can decode one instruction per cycle.
- The machine executes *only* register-type instructions, e.g., $OP\ R_{dest} \leftarrow R_{src1},\ R_{src2}.$, where OP can be ADD or MUL.
- The machine dispatches one instruction per cycle into the reservation stations, in program order. Dispatch occurs during the decode stage.
- An instruction always allocates the first reservation station that is available (in top-to-bottom order) at the required functional unit.
- When an instruction in a reservation station finishes executing, the reservation station is cleared.
- The adder and multiplier are **not** pipelined. An ADD operation takes 2 cycles. A multiply operation takes 3 cycles.
- The result of an addition and multiplication is broadcast to the reservation station entries and the RAT in the writeback stage. A dependent instruction can begin execution in the next cycle after the writeback if it has all of its operands available in the reservation station entry. There is *only* one broadcast bus, and thus multiple instructions *cannot* broadcast in the same cycle.
- When multiple instructions are ready to execute at a functional unit at the same cycle, the oldest ready instruction is chosen to be executed first.

Initially, the machine is empty. Four instructions then are fetched, decoded, and dispatched into reservation stations. Pictured below is the state of the machine when the final instruction has been dispatched into a reservation station:

RAT

Reg	V	Tag	Value
R0	–	–	–
R1	0	A	5
R2	1	–	8
R3	0	E	–
R4	0	B	–
R5	–	–	–

ID	V	Tag	Value	V	Tag	Value
A	0	D	–	1	–	8
B	0	A	–	0	A	–
C	–	–	–	–	–	–

+

ID	V	Tag	Value	V	Tag	Value
D	1	–	5	1	–	5
E	0	A	–	0	B	–
F	–	–	–	–	–	–

×

(a) [15 points] Give the four instructions that have been dispatched into the machine, in program order. The source registers for the first instruction can be specified in either order. Give instructions in the following format: "opcode destination ⇐ source1, source2."

MUL	R1	⇐	R1	,	R1
ADD	R1	⇐	R1	,	R2
ADD	R4	⇐	R1	,	R1
MUL	R3	⇐	R1	,	R4

(b) [15 points] Now assume that the machine flushes all instructions out of the pipeline and restarts fetch from the first instruction in the sequence above. Show the full pipeline timing diagram below for the sequence of four instructions that you determined above, from the fetch of the first instruction to the writeback of the last instruction. Assume that the machine stops fetching instructions after the fourth instruction.

As we saw in lectures, use "F" for fetch, "D" for decode, "En" to signify the nth cycle of execution for an instruction, and "W" to signify writeback. Fill in each instruction as well. You may or may not need all columns shown.

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
MUL R1 ← R1, R1	F	D	E1	E2	E3	W										
ADD R1 ← R1, R2		F	D				E1	E2	W							
ADD R4 ← R1, R1			F	D						E1	E2	W				
MUL R3 ← R1, R4				F	D								E1	E2	E3	W