6 Pipelining (Reverse Engineering) [45 points]

Algorithm 1 contains a piece of assembly code. Table 2 presents the execution timeline of this code.

```
1      MOVI R1, X          # R1 <- X
2      MOVI R2, Y          # R2 <- Y
3  L1:
4      ADD  R1, R1, R2     # R1 <- R1 + R2
5      MUL  R4, R2, R3     # R4 <- R2 x R3
6      SUBI R3, R1, 100    # R3 <- R1 - 100, set condition flags
7      JNZ  L1             # Jump to L1 if zero flag is set
8      MUL  R1, R1, R2     # R1 <- R1 x R2
9      MUL  R2, R3, R4     # R2 <- R3 x R4
10     ADD  R5, R6, R7     # R5 <- R6 + R7
```

Algorithm 1: Assembly Program

Dyn. Instr. Number	Instructions	Cycles													
		1	2	3	4	5	6	7	8	9	10	11	12	13	...
1	MOV R1, X	F	D	E1	E2	E3	M	W							
2	MOV R2, Y		F	D	E1	E2	E3	M	W						
3	ADD R1, R1, R2			F	D	-	-	E1	E2	E3	M	W			
4	MUL R4, R2, R3				F	-	-	D	E1	E2	E3	M	W		
5	SUBI R3, R1, 100							F	D	-	E1	E2	E3	M	...
6	JNZ L1								F	-	D	-	-	E1	...
7	...														

Table 2: Execution timeline (F:Fetch, D:Decode, E:Execute, M:Memory, W:WriteBack)

Use this information to reverse engineer the architecture of this microprocessor to answer the following questions. Answer the questions as precisely as possible with the provided information. If the provided information is not sufficient to answer a question, answer "Unknown" and explain your reasoning clearly.

(a) [10 points] List the necessary data forwardings between pipeline stages to exhibit this behavior.

The result of E3 stage is forwarded to E1 stage (e.g., R1's value at clock cycle 10 and R2's value at clock cycle 7).
The result of E3 stage is forwarded to the condition registers (e.g., SUBI and JNZ at clock cycle 13).
There is no other information for any other data forwarding. Therefore, other data forwardings are unknown.

(b) [5 points] Does this machine use hardware-interlocking or software-interlocking? Explain.

Hardware-interlocking. It detects data dependencies and stalls the pipeline accordingly without needing any software-induced NOPs.

(c) [15 points] Consider another machine that uses the opposite of your choice in the previous question. (e.g., if your answer is software-interlocking for the previous question, consider another machine using hardware-interlocking, or vice-versa). How would the execution timeline shown in Table 2 change? What would be different? Fill the following table and explain your reasoning below. (*Notice that the table below consists of two parts: the first seven cycles at the top, and the next seven cycles at the bottom.*)

We inject NOP instructions in between existing instructions to delay instructions with data dependencies.

Dyn. Instr. Number	Instructions	Cycles						
		1	2	3	4	5	6	7
1	MOV R1, X	F	D	E1	E2	E3	M	W
2	MOV R2, Y		F	D	E1	E2	E3	M
3	NOP			F	D	E1	E2	E3
4	NOP				F	D	E1	E2
5	ADD R1, R1, R2					F	D	E1
6	MUL R4, R2, R3						F	D
7	NOP							F
8	SUB R3, R1, 100							
9	NOP							
10	NOP							
11	JNZ L1							

		8	9	10	11	12	13	14
3	NOP	M	W					
4	NOP	E3	M	W				
5	ADD R1, R1, R2	E2	E3	M	W			
6	MUL R4, R2, R3	E1	E2	E3	M	W		
7	NOP	D	E1	E2	E3	M	W	
8	SUB R3, R1, 100	F	D	E1	E2	E3	M	...
9	NOP		F	D	E1	E2	E3	...
10	NOP			F	D	E1	E2	...
11	JNZ L1				F	D	E1	...

For the rest of this question, assume the following:

- X = Y = 1 in Algorithm 1.
- Branch conditions are resolved at the stage E1.
- Branch predictor is static and predicts "always taken".
- The machine uses hardware-interlocking.

At a given clock cycle T,

- the value stored in R1 is 98.
- the processor fetches the dynamic instruction N which is `ADD R1, R1, R2`

(d) [10 points] Calculate the value of T. Show your work.

T = 682.

Explanation.
Steady state throughput of an iteration is 4 instructions in 7 cycles. The first iteration takes 10 cycles as shown below.

If R1 = 98, this iteration is executed for 97 times so far.

Since in cycle T the first instruction of the loop is being fetched, no cycles of the 98th iteration have executed so far.

Then, $T = 10 + 96 \times 7 + 0 = 682$

(e) [5 points] Calculate the value of N. Show your work.

N = 390.

Explanation.
Loop iterates for 97 times before reaching to clock cycle T.

There are two instructions before the loop starts.

Then, $N = 2 + 97 \times 4 = 390$, assuming that the instruction indices start from 0.