Name: Prashant Serai lastname.#: serai.1
Assignment 3 - Computer Architecture CSE 6421

Q1.

Question sheet filled in with answers attached.

Q2.

a.
The timing diagram is as follows:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I1	IF	ID	EX	MEM	MEM	MEM	WB											
I2		IF	ID	Stall	Stall	Stall	EX	MEM	MEM	MEM	WB							
I3			IF	Stall	Stall	Stall	ID	Stall	Stall	Stall	EX	MEM	WB					
I4							IF	Stall	Stall	Stall	ID	EX	MEM	WB				
I5											IF	ID	EX	MEM	MEM	MEM	WB	
I6												IF	ID	EX	EX	EX	MEM	WB
I1-2													Stall	Stall	Stall	Stall	IF	ID

Assumptions: Execution is strictly in-order. Also, the pipeline stalls when a branch is encountered, until it is resolved. [This results into the next instruction being fetched only after Execution of I6 has completed.]

b.
As the figure in (a) shows, the Instruction Fetch stage for the next instruction is held back by 4 cycles, so 4 cycles are lost per loop iteration to branch overhead.

c.
If all predictions are correct and BTB is not used.

- If the prediction is Taken, we will still need to stall until the address has been computed. So, we will lose 2 cycles in Branch Overhead (with the assumptions below).
- If the prediction is Not Taken, we can simply fetch the next instruction, and we will not lose any cycles in Branch Overhead.

Assumptions:
-The Branch prediction is available by the end of the same cycle as when the Branch Instruction is fetched, and is available for use in the next cycle.
-If branch is taken the "branch-to" address is computed within the first or only cycle of Execution Time, and is available to use in the next cycle.

d.
With a correct prediction and a BTB, no cycles should be lost in Branch Overhead.

Q3.

a.
We may want to rename the following:
- The destination “Rx” in I5, and correspondingly as required the sources labels “Rx” in the following instruction/s.
- The destination “Ry” in I6, and correspondingly as required the sources labels “Ry” in the following instruction/s.
- The source “F4” in in I7, and correspondingly the destn. “F4” in the preceding instruction I3.
- The destination “F2” in ‘Loop’, in the later iterations, so that the next value can be fetched from memory before the previous one has been fully utilized, and correspondingly as required the source labels “F2” in the following instructions. [Note: this is not needed for the first iteration, but may be utilized to good effect as shown in the diagram in (b)]

b. The scheduling is shown up till the cycle in which the last instruction of the first iteration completes.

Cycle	ALU 0	ALU 1	LD / ST
N+ 0	I5	I6	Loop
1	I8	<stall>	I2
2	<stall>	I9	-
3	<stall>	-	-
4	<stall>	<stall>	-
5	I0	<stall>	-
6	-	I3	<stall>
7	-	-	<stall>
8	-	I5-2	I7
9	-	I6-2	-
10	-	I8-2	Loop-2
11	-	I9-2	I2-2
12	-	<stall>	-
13	-	<stall>	-
14	-	<stall>	-
15	-	I0-2	-
16	-	-	…
17	-	-	…
18	I1	-	…
19	-	-	…
20	-	-	…
21	-	-	…
22	-	-	…
23	-	-	…
24	I4	-	…

25	-	-	…

25 clock cycles are taken, if we consider count the time taken from the start of the first iteration to the point when the last remaining instruction of the first iteration i.e. I4 is scheduled.

Assumptions:
-The branch is "executed" in the ALU pipeline, and can be scheduled as soon as its operand (in this case - R20) becomes available.
-Assuming the branch is taken, the instructions for about 2 iterations of the loop are also made available in the RS for when needed, and that those instructions may be scheduled.
-The size of RS, is not a limiting factor.

c.
The "available upto" column indicates the last instruction up to which all the instructions are available in the RS, at the beginning of the particular cycle.

Cycle	ALU 0	ALU 1	LD / ST	Available upto
N+ 0	<stall>	<stall>	Loop	I0
1	<stall>	<stall>	I2	I3
2	I5	<stall>	-	I5
3	I6	<stall>	-	I7
4	I8	<stall>	-	I9
5	I0	I9	-	I0-2
6	-	-	<stall>	I3-2
7	-	I3	<stall>	I5-2
8	-	-	<stall>	I7-2
9	-	I5-2	I7	I9-2
10	-	I6-2	-	…
11	-	I8-2	Loop-2	…
12	-	I9-2	I2-2	…
13	-	-	-	…
14	-	<stall>	-	…
15	-	<stall>	-	…
16	-	I0-2	-	…
17	-	-	…	…
18	I1	-	…	…
19	-	-	…	…
20	-	-	…	…
21	-	-	…	…
22	-	-	…	…
23	-	-	…	…
24	I4	-	…	…
25	-	-	…	…

Again the number of cycles taken is 25, by the same metric as in the previous question.

Assumptions:
-The branch is "executed" in the ALU pipeline, and can be scheduled as soon as its operand (in this case - R20) becomes available.
-Assuming the branch is taken, the instructions for up to about 2 iterations of the loop are also made available in the RS for when needed, and that those instructions may be scheduled.
-The size of RS, is not a limiting factor.

d.
An example of a code sequence where the Common Data Bus (CDB) creates a bottleneck might be:

1. MULTD F4, F0, F4
2. ...<some independent instruction>…
3. ...<some independent instruction>…
4. ...<some independent instruction>…
5. ADDD F10, F8, F2
6. SUBD F6, F10, F4

Now, given the latencies in the other problems, MULTD and ADD would finish in the same cycle, and SUBD would be waiting for the result of both. Now, even though both MULTD and ADDD would have completed execution, they wouldn't be able to send their results simultaneously if we use a relatively simplistic CDB that the question seems to refer to. This would create a bottleneck preventing the SUBD from executing immediately as soon as MULTD and ADDD have finished execution.