

# Computer Architecture-I

## Assignment 3 Solution

1. The baseline performance in cycles per loop iteration can be arrived at as shown below. We do not start any instruction unless and until the previous instruction is complete.

Instruction	CLK Execution	CLK Completion
LD F2,0 (Rx)	1	4
MULTD F2,F0,F2	5	9
DIVD F8,F2,F0	10	20
LD F4,0 (Ry)	21	24
ADDD F4,F0,F4	25	27
ADDD F10,F8,F2	28	30
SD F4,0 (Ry)	31	32
ADDI Rx,Rx,#8	33	33
ADDI Ry,Ry,#8	34	34
SUB R20,R4,Rx	35	35
BNZ R20,LOOP	36	<b>37</b>

As shown, it takes 37 cycles to complete an iteration.

2. In this case, we will stall only when there is true data dependence between successive instructions.

Instruction	CLK Execution
LD F2,0 (Rx)	1
<Stall>	2
<Stall>	3
<Stall>	4
MULTD F2,F0,F2	5
<Stall>	6
<Stall>	7
<Stall>	8
<Stall>	9
DIVD F8,F2,F0	10
LD F4,0 (Ry)	11
<Stall>	12
<Stall>	13

<Stall>	14
ADDD F4, F0, F4	15
<Stall>	16
<Stall>	17
<Stall>	18
<Stall>	19
<Stall>	20
ADDD F10, F8, F2	21
SD F4, 0 (Ry)	22
ADDI Rx, Rx, #8	23
ADDI Ry, Ry, #8	24
SUB R20, R4, Rx	25
BNZ R20, LOOP	26
<Stall>	<b>27</b>

The loop takes 27 cycles per iteration.

3. There are two execution pipes available. The design also assumes forwarding to make results available immediately. The scheduling of the code is as shown below:

CLK Cycle	Execution Pipeline 1	Execution Pipeline 2
1	LD F2, 0 (Rx)	NOP
2	<Stall>	NOP
3	<Stall>	NOP
4	<Stall>	NOP
5	MULTD F2, F0, F2	NOP
6	<Stall>	NOP
7	<Stall>	NOP
8	<Stall>	NOP
9	<Stall>	NOP
10	DIVD F8, F2, F0	LD F4, 0 (Ry)
11	<Stall>	<Stall>
12	<Stall>	<Stall>
13	<Stall>	<Stall>
14	<Stall>	ADDD F4, F0, F4
15	<Stall>	<Stall>
16	<Stall>	<Stall>
17	<Stall>	NOP
18	<Stall>	NOP
19	<Stall>	NOP

20	<Stall>	NOP
21	ADDD F10, F8, F2	SD F4, 0 (Ry)
22	ADDI Rx, Rx, #8	ADDI Ry, Ry, #8
23	SUB R20, R4, Rx	BNZ R20, LOOP
<b>24</b>	<Stall>	NOP

Thus, the iteration takes 24 cycles to complete in the 2 issue design.

4. When we have a multiple issue design as shown above, it can be seen that there is an out-of-order execution of the program. This can lead to the following hazards:

a. WAW: Consider two instructions, N and N+1 writing to the same destination register. If they are issued together, and if the latency of instruction N is more than N+1, we have a WAW hazard. Thus, it is essential for such hazards to be identified at the issue stage and stall if necessary.

b. Imprecise Exceptions: In the above code, consider the following pair of instructions.

```

DIVD F8, F2, F0
LD F4, 0 (Ry)

```

As seen in the table above, these instructions are issued together as they do not exhibit any data-dependence. It can also be seen that the DIV instruction takes 10 cycles whereas the LD instructions takes 2 cycles to complete. In such a scenario, if the DIV instruction were to cause an exception at CLK20, we have an imprecise exception as the register F4 has already been modified by the next instruction in program order. This can be solved by maintaining a 'history file' used to roll back to the value of F4 before the execution of LD instruction.

5. Now, we assume that the hazards mentioned above have been taken care of. We attempt to reorder the instructions to get a better performance in cycles per loop iteration. The code is shown below:

CLK Cycle	Execution Pipeline 1	Execution Pipeline 2
1	LD F2, 0 (Rx)	LD F4, 0 (Ry)
2	<Stall>	<Stall>
3	<Stall>	<Stall>
4	<Stall>	<Stall>
5	MULTD F2, F0, F2	ADDD F4, F0, F4

6	<Stall>	<Stall>
7	<Stall>	<Stall>
8	<Stall>	SD F4,0 (Ry)
9	<Stall>	<Stall>
10	DIVD F8,F2,F0	ADDI Rx,Rx,#8
11	<Stall>	ADDI Ry,Ry,#8
12	<Stall>	NOP
13	<Stall>	NOP
14	<Stall>	NOP
15	<Stall>	NOP
16	<Stall>	NOP
17	<Stall>	NOP
18	<Stall>	NOP
19	<Stall>	SUB R20,R4,Rx
20	<Stall>	NOP
21	ADDD F10,F8,F2	BNZ R20,LOOP
<b>22</b>	<Stall>	<Stall>

The reordered code takes *22 cycles* to complete one iteration.