Tutorial 5: "Pipeline"

Q1: 3 staged Pipeline, delay 10, 20 and 40 ns. Assume no delay extra and no hazards. Assuming one instruction fetched every cycle, total execution time for 100 instructions?

Solution:

$(K + (n-1)) * t_{pipeline}$ $k = 3$, $n=100$ and $t_p = \max (10, 20, 40)$

Q2: 4 stage pipeline (5, 6, 11, 8 ns), 1 ns for register in between. Speed UP with and without pipeline.

Solution:

Non pipelined - for an instruction : 30 ns (5+6+11+8)

Pipelined – 12 ns (assuming after 1st instruction)

Speed Up= 2.5

Q3: 5 stage pipeline, F, D, FO, E, WO with 5 ns, 7 ns, 10 ns, 8 ns and 6 ns delays respectively. After each stage, pipeline register with delay 1 ns. We have a program:

I1, I2, I3, I4, I5…….I9

Instruction I3 is the only branch instruction, and its branch target is I7. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is?

Solution:

13 cycles i.e. 13*(11) ns = 143 ns

Q4: 5 stage pipeline, register read is performed in EX stage, EX stage takes 1 cycle for ADD and 2 for MUL. Ignore pipeline register latencies. Consider below seq of instruction -

ADD, MUL, ADD, MUL, ADD

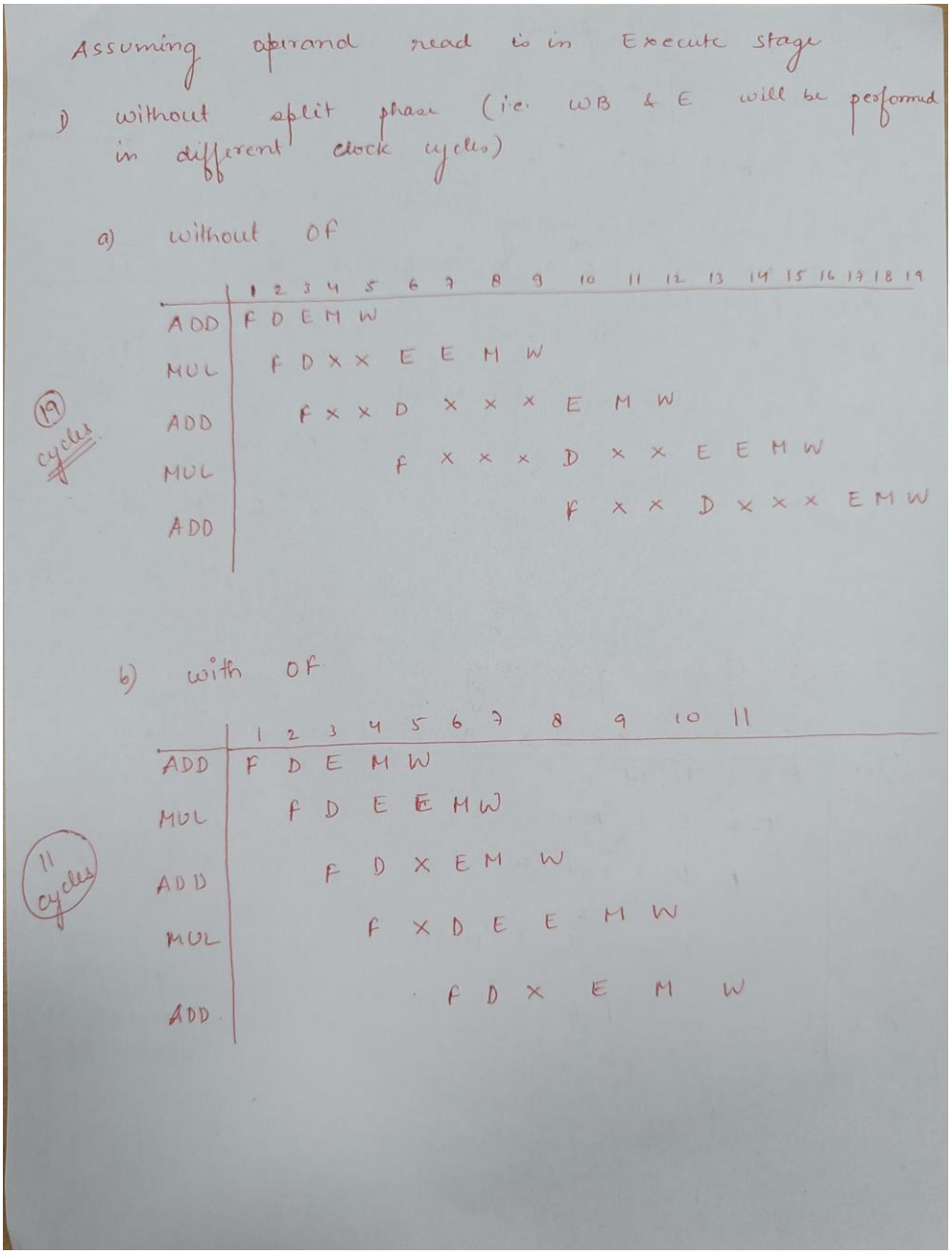
From the second instruction, all are data dependent on previous one, what is speed up with operand forwarding over without. [Draw Phase Diagram for both]

Solution:

Assuming operand read is in Execute stage

1) without split phase (i.e. WB & E will be performed in different clock cycles)

a) without OF



b) with OF

Speed Up - 19/11 = 1.7272

Assuming No split phase

1) Assuming operand read

2) with split phase (W & E can be done in same clock)

⇒ without operand forwarding

operands read here ↓

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADD	F	D	E	M	W										
MUL		F	D	x	E	E	M	W							
ADD			F	x	D	x	x	E	M	W					
MUL				F	x	x	D	x	E	E	M	W			
ADD							F	x	D	x	x	E	M	W	

15 cycles.

⇒ with operand forwarding

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADD	F	D	E	M	W										
MUL		F	D	E	E	M	W								
ADD			F	D	x	E	M	W							
MUL				F	x	D	E	E	M	W					
ADD						F	D	x	E	M	W				

(11) cycles

We see no effect of split phase same as without OF.

Speed Up - 15/11 = 1.36

NOTE:

1. **Some of your colleagues have assumed Operand Fetch is done in Decode stage, that too is correct.**
2. **Do mention whatever you assume in whatever stage.**
3. **Make sure you also mention whether you are using SPLIT phase or not, if not mentioned in question.**

Q5: Pipelined System with 5-stage, IF, ID, FO, E and WB, each take 1 clock for any instruction except E stage. E stage takes 1 clock cycle for ADD and SUB instructions,3 clock cycles for MUL instruction, and 6 clock cycles for DIV instruction. Tell number of clock cycles system will take with and without operand forwarding?

Note: FO, is preferred to be performed just before E stage.

For below instructions:

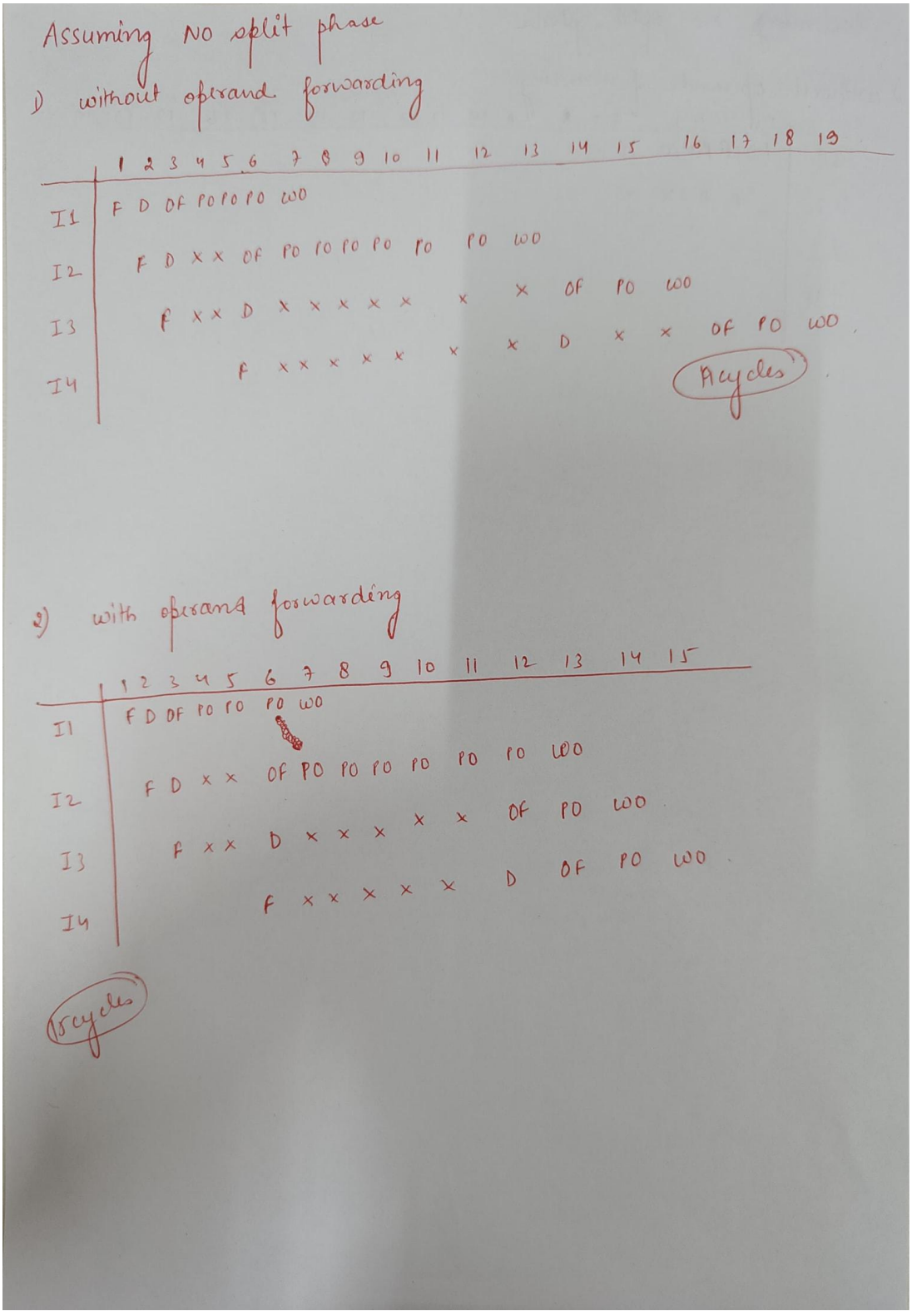
Instruction	Meaning of instruction
I0 :MUL R2 ,R0 ,R1	R2 ¬ R0 *R1
I1 :DIV R5 ,R3 ,R4	R5 ¬ R3/R4
I2 :ADD R2 ,R5 ,R2	R2 ¬ R5+R2
I3 :SUB R5 ,R2 ,R6	R5 ¬ R2-R6

Solution:

FO (fetch operand) = OF (operand fetch) and PO (perform operation) = E (execute)

with FO: 15

Without FO: 19



Note: Some of you have assumed Split phase is present, for that you may give a try, I haven't solved it, you may take reference of above question.

During tutorial, we concluded if split phase is in place, answers would be:

without OF: 17

with OF: I don't see any change i.e. 15 cycles only.

Q6: X1: Processor operating at 2GHz, 5 stage pipeline, CPI = 1, no pipeline hazards. We have a program P1, 30% instructions are branch, control hazard incur 2 cycle stalls for every branch. X2: processor operating at 2GHz but it has a Branch Prediction Unit, eliminates stalls for correct prediction. For wrong predictions no saving and no additional stalls. No structural and data hazards. BPU accuracy 80 %, speedup by X2 over X1

Solution:
CPI X1 for P1: 0.7*1 + 0.3*3 = 1.6
CPI X2 for P1: 0.7*1 + 0.3 (0.8*1 + 0.2 * 3) = 0.7 + 0.42 = 1.12
Speed up = 1.6/1.12 = 1.42857

Other way:

CPI X1 for P1: 1 (for each instruction) + 0.3*2 (penalty) = 1.6
CPI X2 for P1: 1 + 0.3 (0.8*0 + 0.2 * 2) = 1 + 0.12 = 1.12
Speed up = 1.6/1.12 = 1.42857

Q7: Non-Pipelined Processor, 5 clock cycle to complete an instruction with 2.5 GHz processor freq.

Now we are making a 5-stage pipeline, 2 GHz (due to pipeline overheads) processor operating freq.

The program we wish to run has 30 % memory, 60% ALU and the rest of the branch. 5% memory instructions cause stalls of 50 clock cycles each due to some reason and 50 % branch instructions cause stalls of 2. No stall for ALU, speed up??

Solution:
Non-Pipeline: 5*(1/2.5) ns = 2 ns for one instruction.
Pipeline: 1+ 0.3*0.05*50 + 0.1*0.5*2 = 0.925
Speed Up: 2/0.925 = 2.16

Also, we can do it as:
Let we have n instructions,
Non-Pipeline: 2n ns
Pipeline:
(0.6*1 + 0.3 (0.05 (1+50) + 0.95 (1)) + 0.1 (0.5*1 + 0.5*(1+2))) cycles * (0.5 ns) time
0.6 + 0.3 (3.5) + 0.2

0.925
Speed Up: 2/0.925 = 2.162