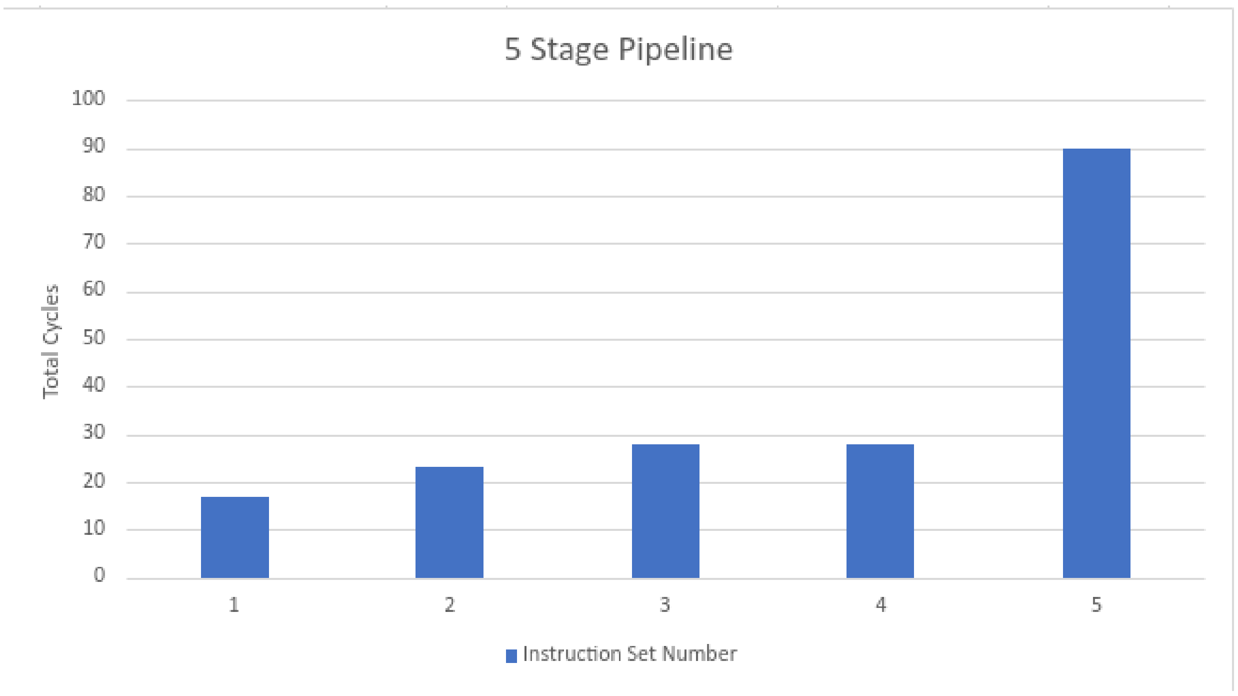
COL216 ASSIGNMENT-2

ACHYUT SALUNKHE(2021CS10121) :- 50%

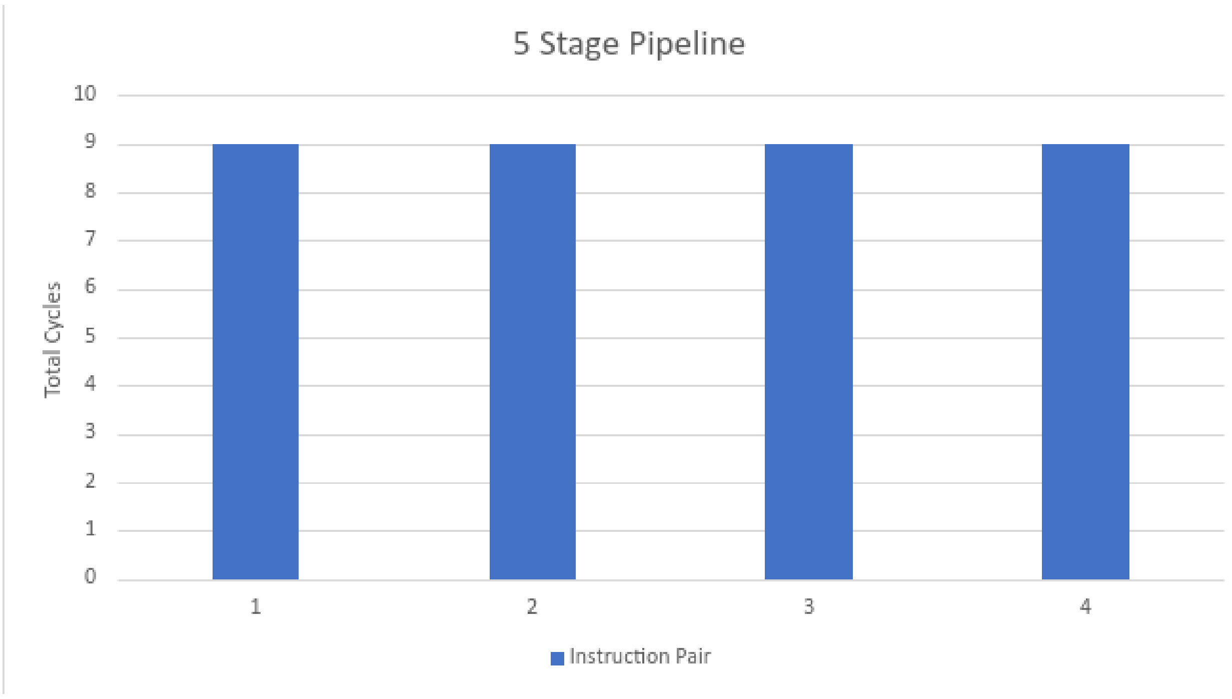
RISHIT SINGLA(2021CS10547) :- 50%

Clock cycles for 5 stage Pipeline without Bypassing:

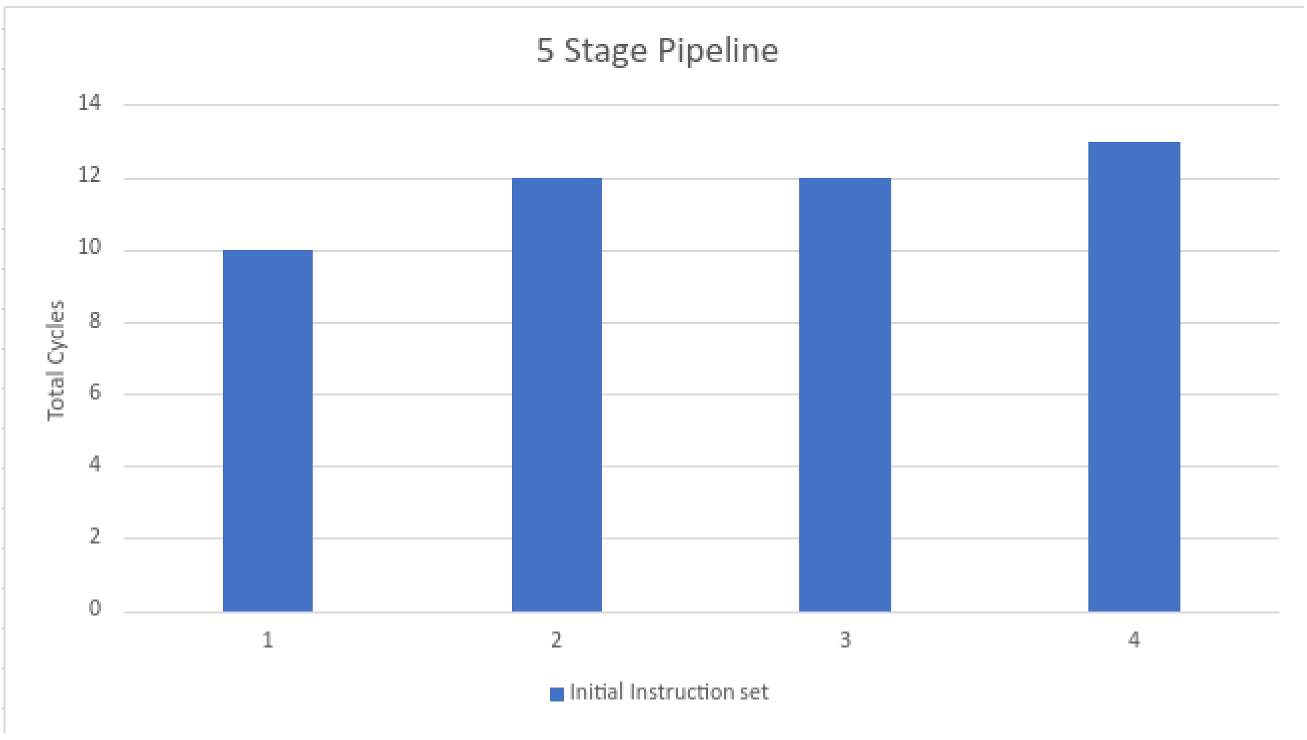
Instruction Set Number	Total cycles
1	17
2	23
3	28
4	28
sample	90



Instruction Pair Number	Total cycles
1	9
2	9
3	9
4	9

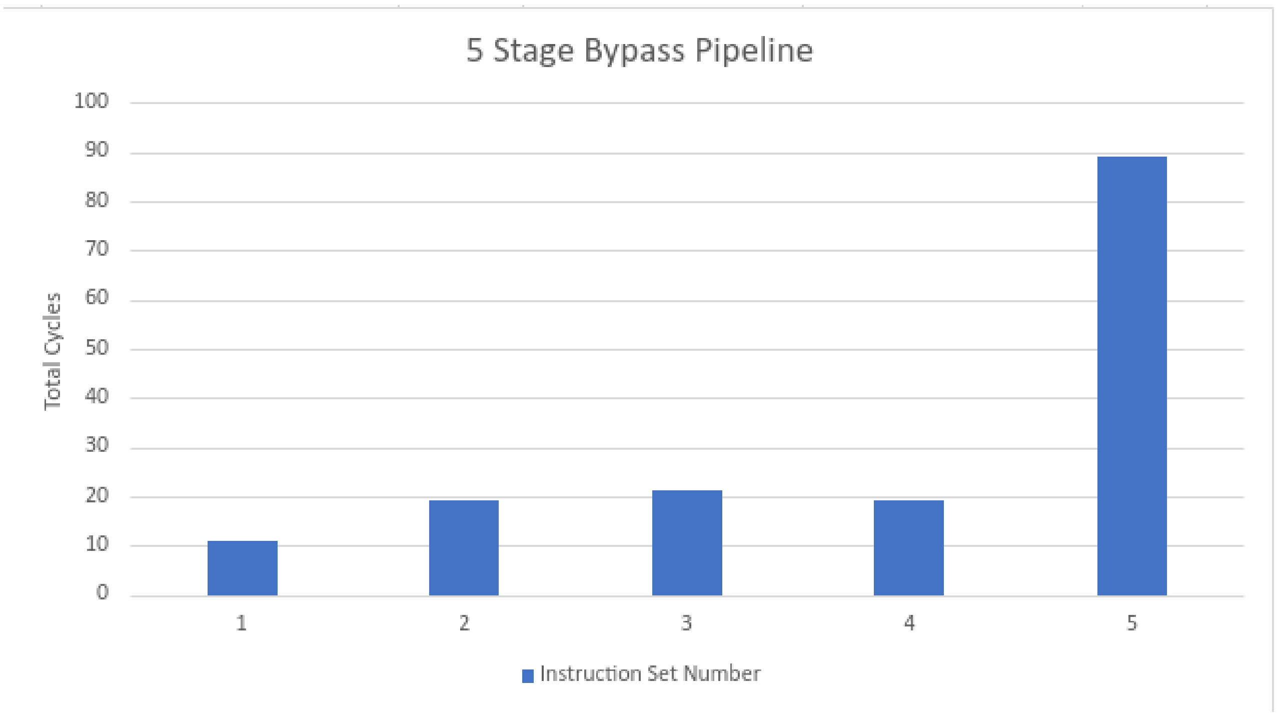


Initial Instruction Set	Total cycles
1	10
2	12
3	12
4	13

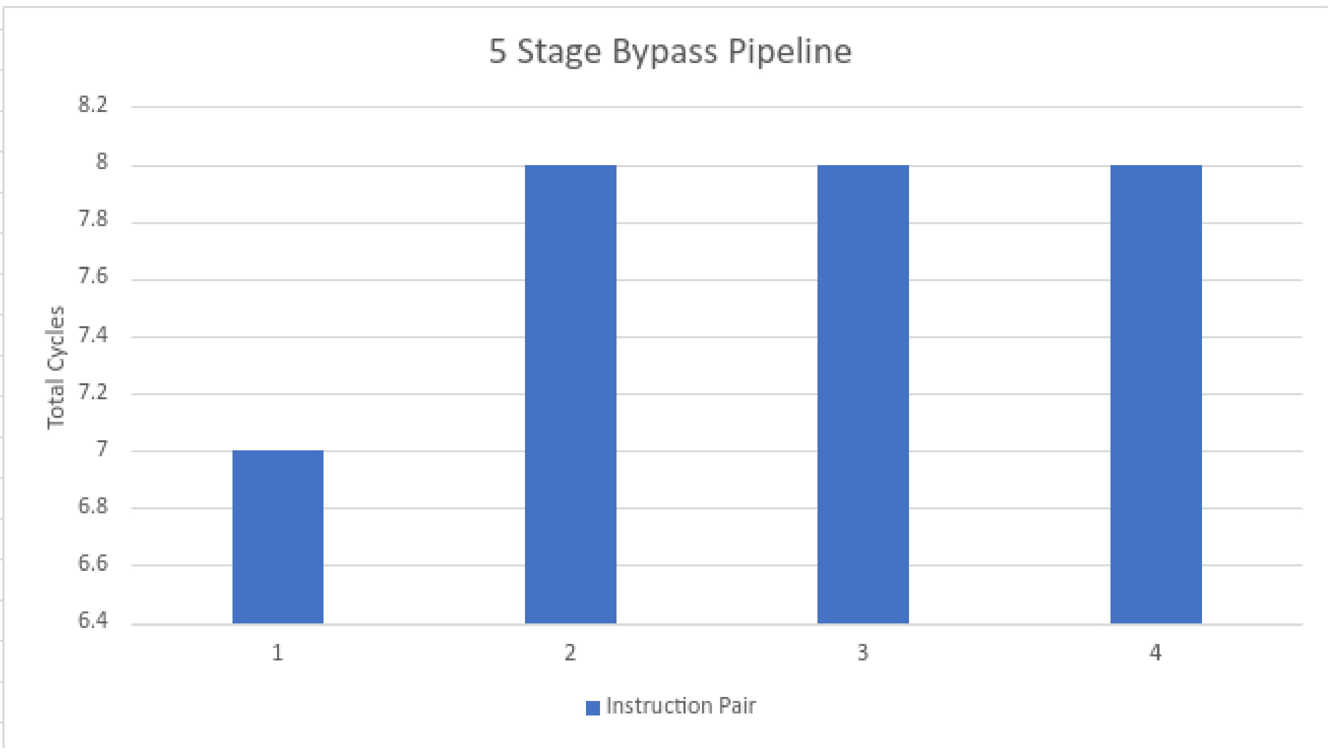


Clock cycles for 5-stage Pipeline with Bypassing:

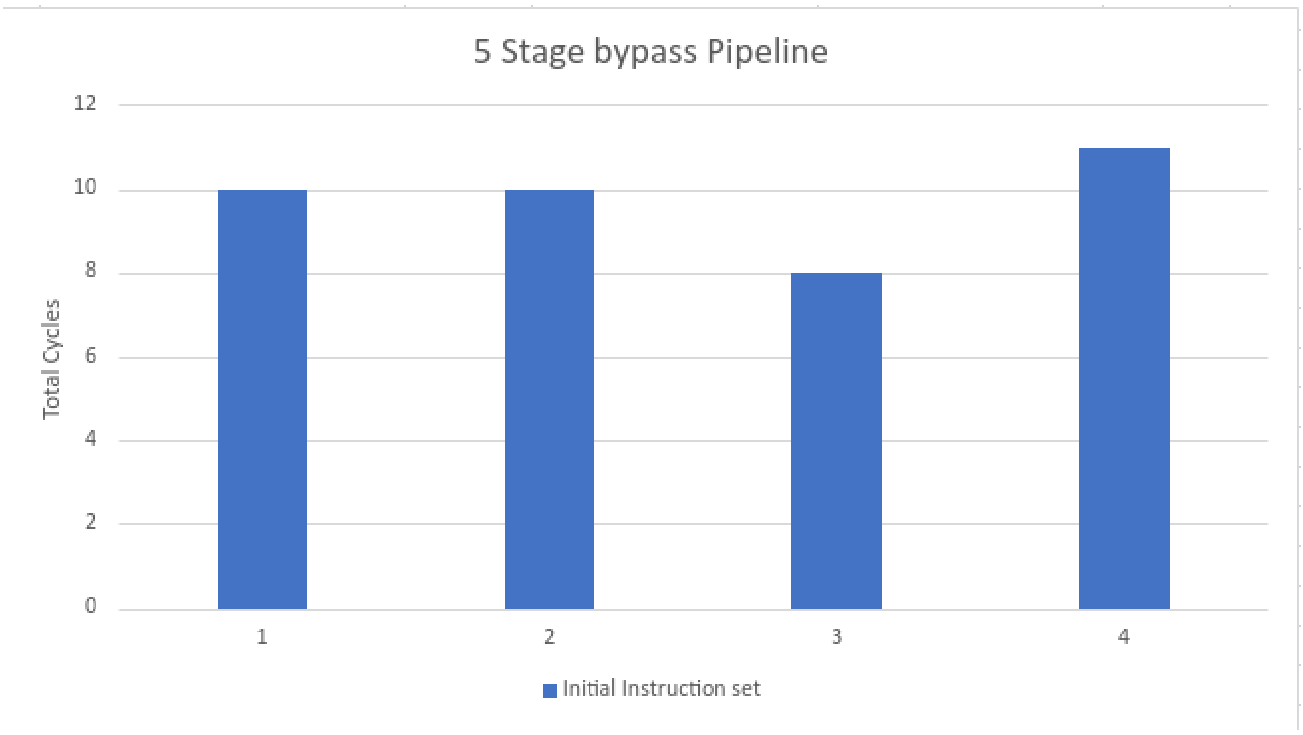
Instruction Set Number	Total cycles
1	11
2	19
3	21
4	19
sample	89



Instruction Pair Number	Total cycles
1	7
2	8
3	8
4	8

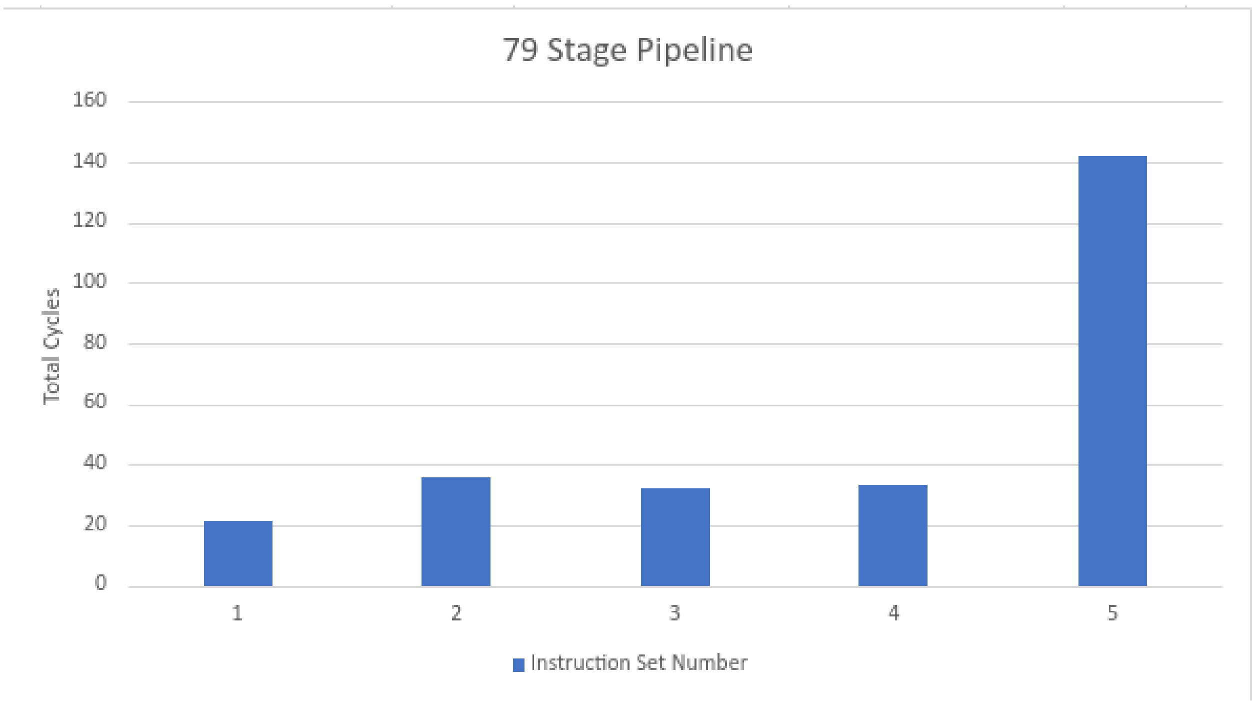


Initial Instruction Set	Total cycles
1	10
2	10
3	8
4	11

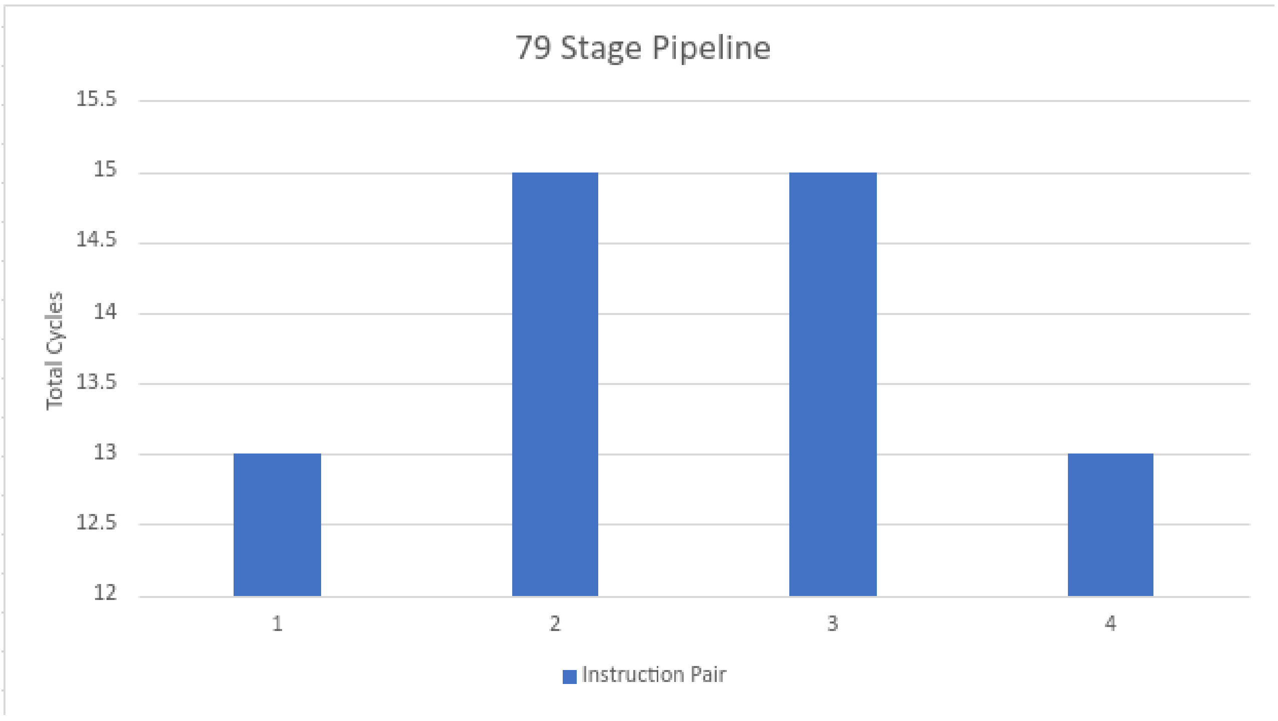


Clock cycles for 79-stage Pipeline without Bypassing:

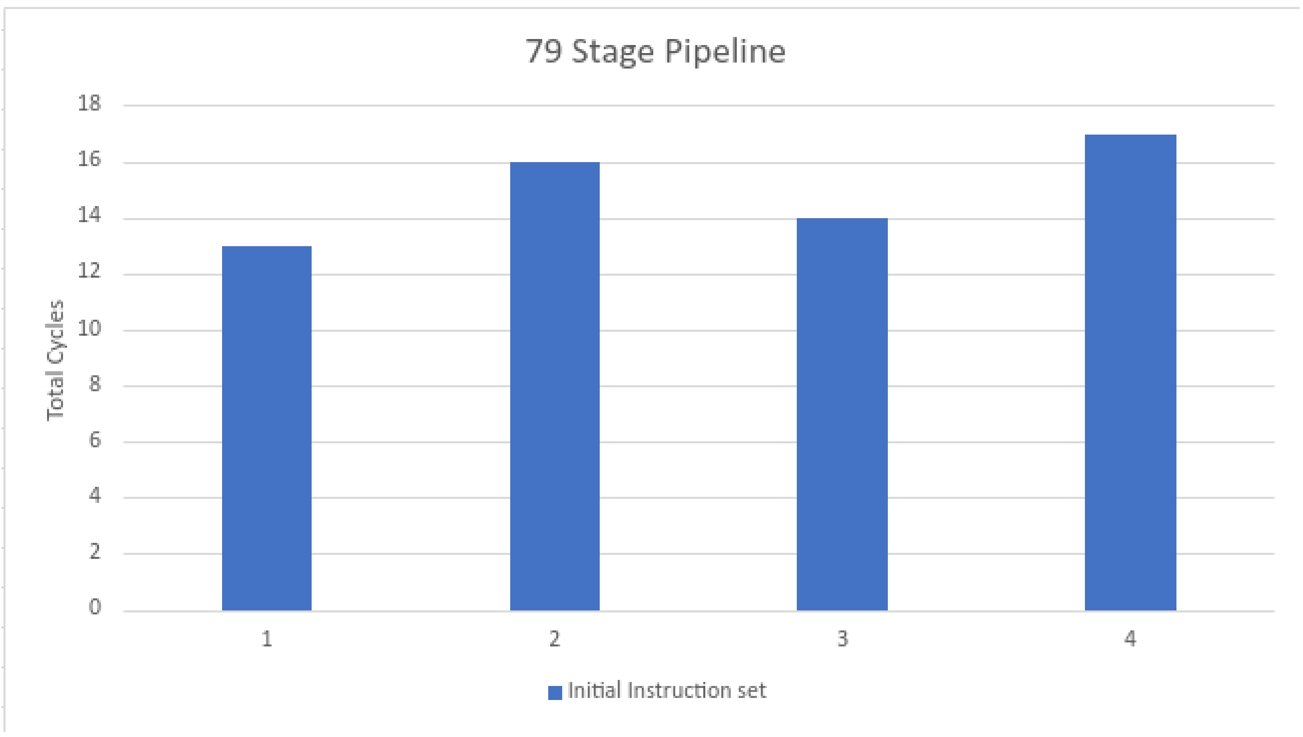
Instruction Set Number	Total cycles
1	21
2	36
3	32
4	33
sample	142



Instruction Pair Number	Total cycles
1	13
2	15
3	15
4	13

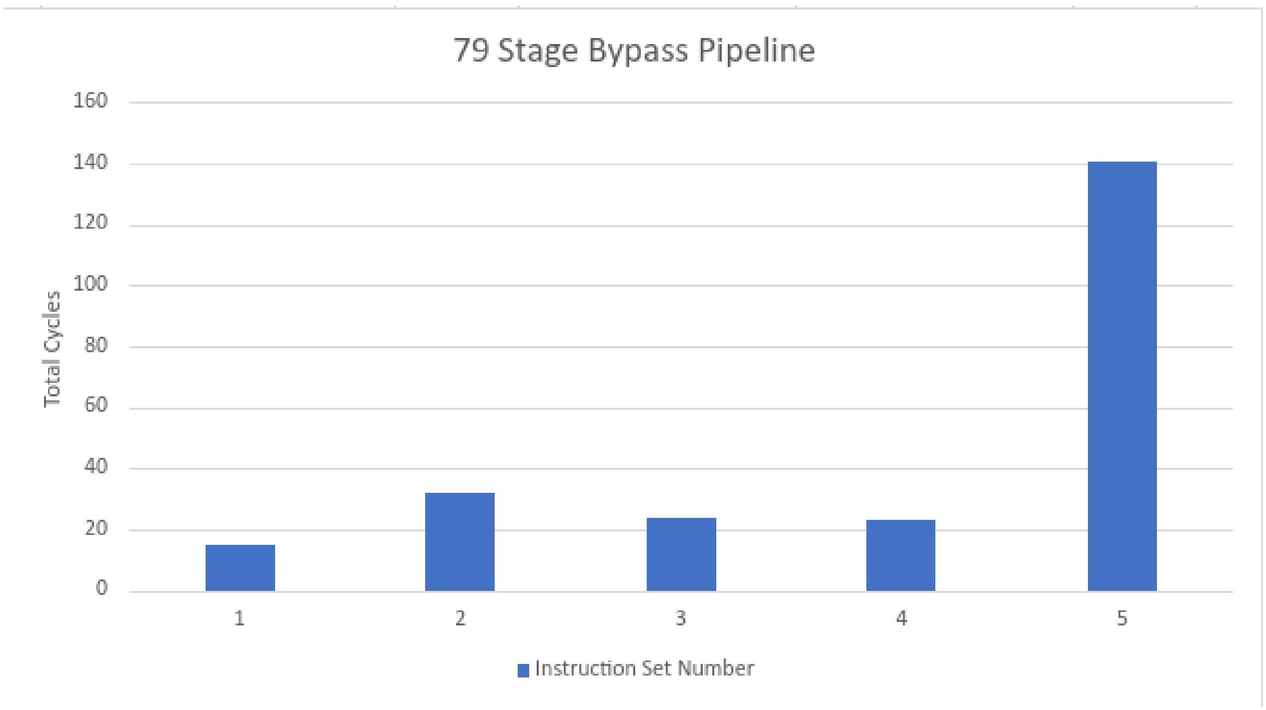


Initial Instruction Set	Total cycles
1	13
2	16
3	14
4	17

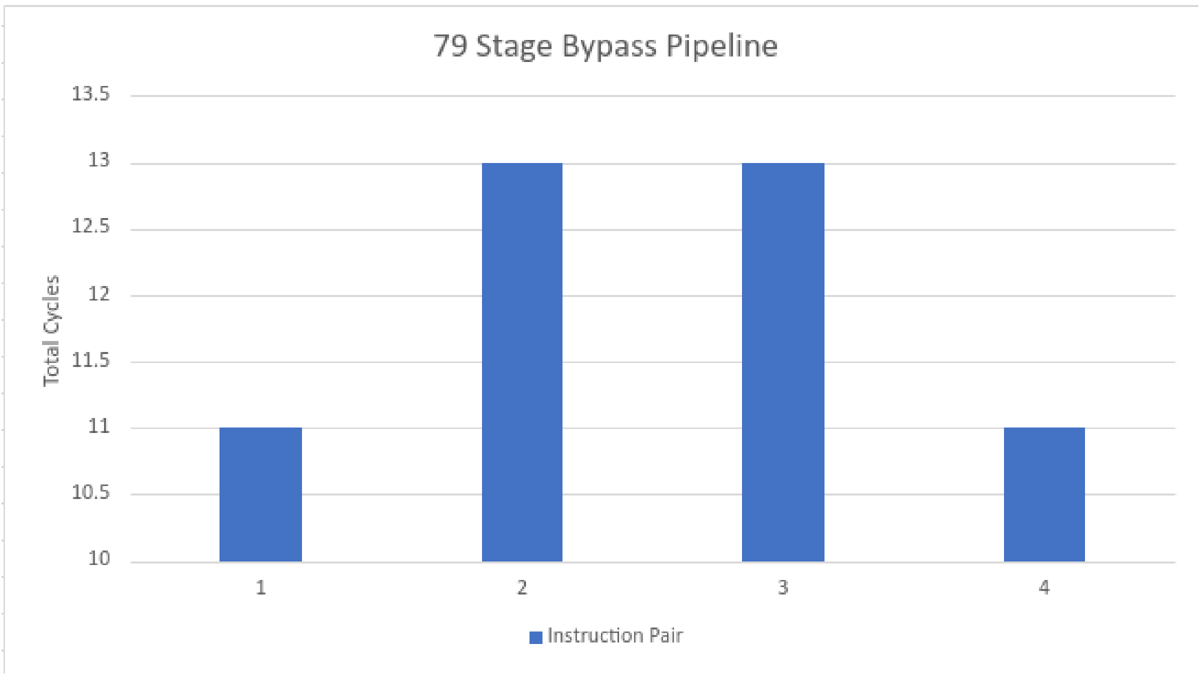


Clock cycles for 79-stage Pipeline with Bypassing:

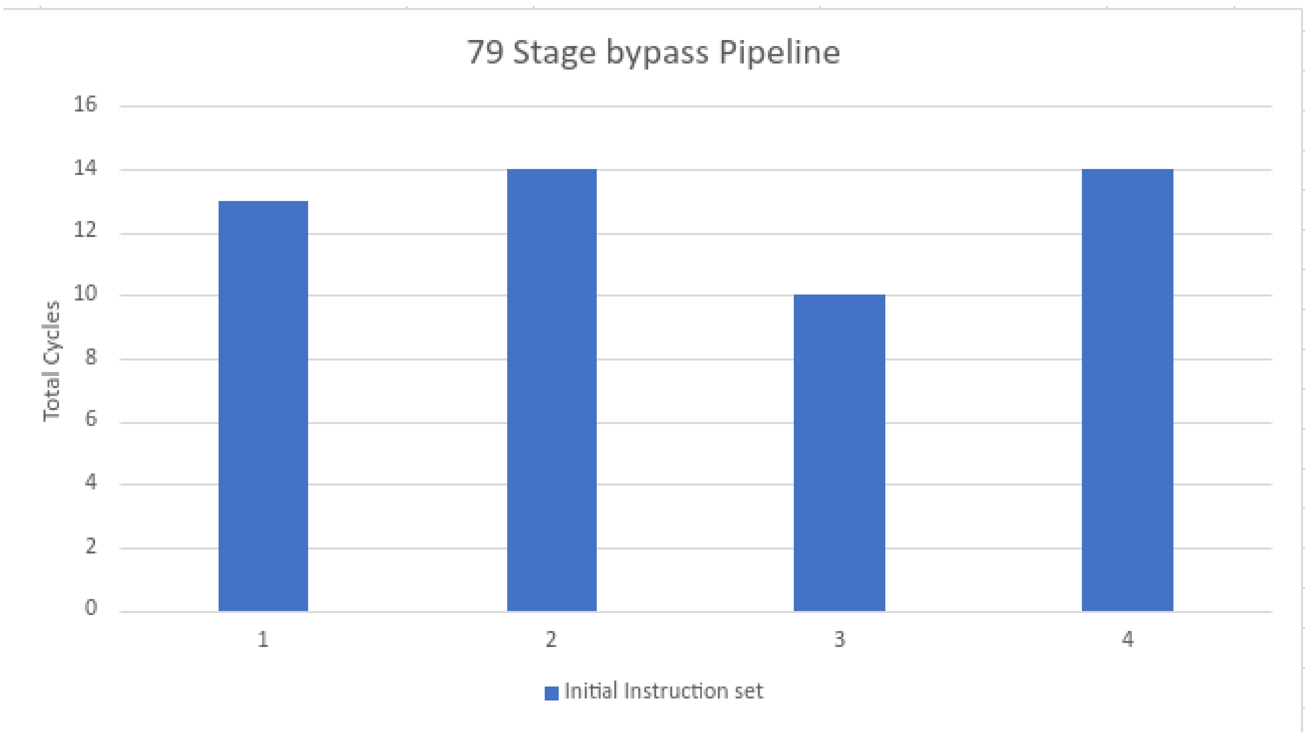
Instruction Set Number	Total cycles
1	15
2	32
3	24
4	23
sample	141



Instruction Pair Number	Total cycles
1	11
2	13
3	13
4	11



Initial Instruction Set	Total cycles
1	13
2	14
3	10
4	14



Observation:

On bypassing, it is clear from the graphs and table data that the number of clock cycles needed to complete an instruction set reduces (or, in some cases, it may remain the same). This is due to the reduced number of NoOps used for stalling. Hence it fastens the program. Depending on the set of instructions, this reduction can be negligible as well as considerable.

The number of clock cycles taken by 79 stage pipeline is more than that of the 5 stages. The number of cycles has increased in every case. But in 7 stages pipeline, the time of one clock cycle will be less than the time of one clock cycle in 5 stage pipeline. So, the total time taken by 79 stage pipeline might be less than that of the 5 stages. However, this may not be the case. It all depends upon the ratio of time of one clock cycle of the different staged pipelines.

Our program uses an object-oriented design for the implementation. Two structures are created for pipeline 5 and 79 stages, respectively. Each latch register is also created as a nested struct. A nested struct is also created for control. For 5-stage pipeline implementation, the first half and the second half of the clock cycle are done separately. This was impossible in 79 stages, as it takes a complete clock cycle to perform any task. The same code for inserting NoOps is used in all four cases due to object-oriented design. For both pipelines, the code with bypassing and without bypassing is the same with different function arguments. This was again possible due to object-oriented design. Hence, our program uses object-oriented design and reuses as much code as possible.

Branch Prediction Strategy:

Strategy	Starting Value	Accuracy
There will be a saturating counter for every instruction, and they will be stored in a table of size (2^14) with 14 LSB bits of instruction counter as their hash function.	00	79.0146
	01	83.9416
	10	87.9562
	11	86.6788
A global BHR will store whether or not the last two branches were taken. Accordingly, there will be saturating counters for different values of BHR.	00	71.5328
	01	72.2628
	10	72.6277
	11	72.8102
There will be BHRs, which store what path the last two branches have taken. After getting the BHR, we concatenate it with 14 LSBs of the instruction counter, and we will get the index for the saturating counter in the combination vector.	00	75.9124
	01	81.2044
	10	87.4088
	11	85.5839