Objective:

To provide deeper understanding of Pipelining Architecture using CPU- OS Simulator.

Pipeline Processor

Consider the following program:

```
program pipeline1

        x=10
        y=20

        z=30
        z=z+1
        y=y-1

        z=z+x
end
```

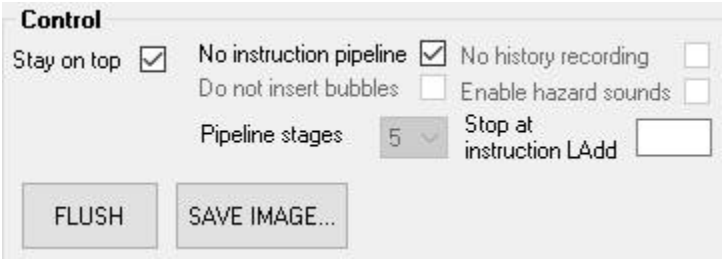
Compile the code and load it in CPU-OS simulator. Perform the following:

Execute the above program using non-pipelined processor and pipelined processor and answer the following questions.

Note: Every time flush the pipeline before running the code

A) Non-pipelined Processor:

To enable non-pipelined processor, check **"**No instruction pipeline **"** check box in control panel.



a) How many stages are there in non-pipelined processor? List them

There are totally 5 stages. They are listed below:

Fetch

Decode

Read

Operands

Execute

Write

Result

b) Fill in the following after executing of above program using non-pipelined processor

	Clocks	Instruction Count	CPI	Speed up Factor
Non Pipelined processor	105	19	5.53	0.9

c) What are the contents of General purpose registers after the execution of the program?
Contents of General purpose registers:

R01 = 10

R02 = 19

R03 = 41

R04 = 41

R05 = 41

Rest all general purpose registers value are zero.

B) Pipelined processor:

To use, enable pipelined processor, uncheck "No instruction pipeline " check box in control panel.

a) Fill in the following table with respect to pipelined processor execution of the above program:

Pipelined processor Conditions	Clocks	Instruction Count	CPI	Speed up Factor	Data hazard (Yes/No)	Contents of registers used by the program
Check "Do not insert bubbles" check box	33	19	1.74	2.87	No	R01 = 0 R02 = 21 R03 = 20 R04 = 20 R05 = 20
Uncheck "Do not insert bubbles"	38	19	2	2.5	Yes	R01 = 10 R02 = 19 R03 = 41 R04 = 41 R05 = 41

b) Is there a way to improve the CPI and Speed up factor? If so give the solution. Solution:

To be able to improve CPI and the speed-up factor you need to enable the pipeline. Making use of optimising compiler and clever CPU architecture will also improve CPI & Speed factor. The simulator's built-in compiler offers some help in reducing the CPI, for example loop unrolling, constant folding, identifying code dependencies, etc. In pipelined architecture, the goal is to reduce/prevent hazards and the above optimisations help do that. In addition, the pipeline has a means of forwarding operands, i.e. prior to previous instructions updating registers in a later stage, that contributes to reducing the CPI. There is also the technique of jump predictions.