Course Name: Computer Architecture Lab	**Course Code:** CMPE-421L
Assignment Type: Lab	**Dated:** 4th December 2023
Semester: 7th	**Session:** 2020
Lab/Project/Assignment #: 10	**CLOs to be covered: CLO 3**
Lab Title: Pipelined Architecture	**Teacher Name:** Engr. Afeef Obaid

Lab Evaluation

CLO 3	Understanding and implementation of pipelined RISC-V architecture and handling hazards associated with it.					
Levels (Marks)	**Level1**	**Level2**	**Level3**	**Level4**	**Level5**	**Level6**
(10)						
					Total	**/10**

Rubrics for Current Lab Evaluation

Scale	**Marks**	**Level**	**Rubric**
Excellent	**9-10**	L1	Submitted all lab tasks, BONUS task, have good understanding.
Very Good	**7-8**	L2	Submitted the lab tasks but have good understanding
Good	**5-6**	L3	Submitted the lab tasks but have weak understanding.
Basic	**3-4**	L4	Submitted the lab tasks but have no understanding.
Barely Acceptable	**1-2**	L5	Submitted only one lab task.
Not Acceptable	**0**	L6	Did not attempt

Lab # 10

Lab Goals

By reading this manual, students will be able to:

- Understand the Pipelined Architecture
- Understand how to modify single cycle implementation to 3-stage pipelined architecture,

Equipment Required

- Computer system with ModelSim or Xcelium, installed on it.

Pipelined Architecture

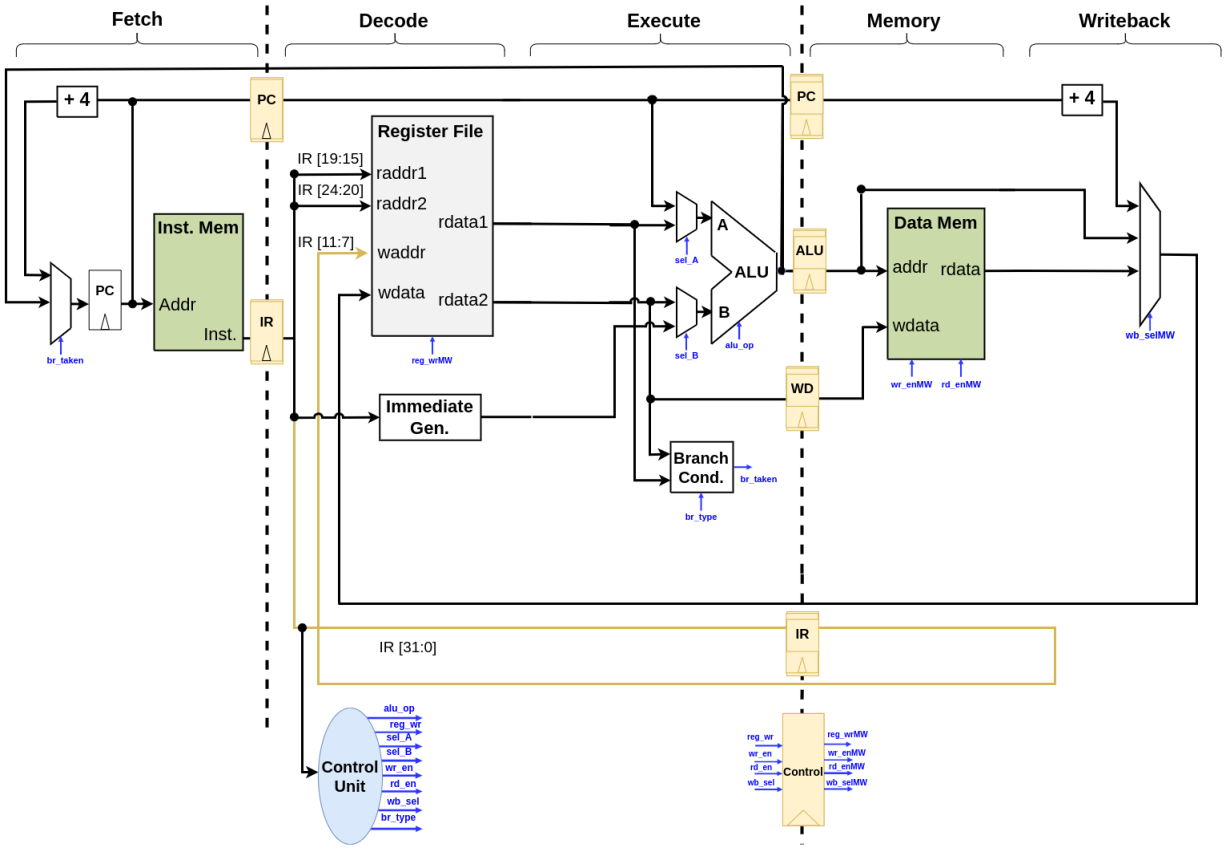
Pipelining does not reduce time-to-completion for an instruction, rather it increases the throughput of the processor. Multiple different operations (for different instructions) are performed simultaneously, using different hardware resources. Using pipelining, allows us to run the entire hardware at higher operating frequency, which effectively improves the system throughput.

From Single Cycle to Pipelined Architecture

We modify our single cycle implementation to 3-stage pipelined architecture. For that purpose we decompose the single cycle processor to the following three stages.

1) Fetch
2) Decode and Execute
3) Memory and Writeback

Specifically, the pipelined datapath is formed by splitting the single-cycle datapath into three stages, where each pair of consecutive stages is separated by pipeline registers. For instance, to introduce the pipeline stage between fetch phase and decode & execute phase, two registers (namely PC register and Instruction register) are required as can be seen from the following Figure



Pipelined processor microarchitecture.

Listing illustrates the configurable pipeline stage implementation at the top level.

```
// Fetch <-----> Decode pipeline/nopipeline
`ifdef IF2ID_PIPELINE_STAGE
  type_if2id_data_s                    if2id_data_pipe_ff;

  always_ff @(posedge clk) begin
      if (rst_n) begin
          if2id_data_pipe_ff <= '0;
      end else begin
          if2id_data_pipe_ff <= if2id_data;
      end
  end
`endif // IF2ID_PIPELINE_STAGE

// Instruction Decode module instantiation
decode decode_module (
    .rst_n               (rst_n),
    .clk                 (clk),

    // ID module interface signals
`ifdef IF2ID_PIPELINE_STAGE
    .if2id_data_i        (if2id_data_pipe_ff),
`else
    .if2id_data_i        (if2id_data),
`endif
    .id2if_fb_rdy_i      (id2if_rdy ),
    .id2exe_ctrl_o       (id2exe_ctrl),
    .id2exe_data_o       (id2exe_data),
    .wb2id_fb_i          (wb2id_fb)
);
```

Listing Configurable pipeline stage implementation for fetch and decode phase (Code segment from pipeline top module).

Tasks

- Implement a pipeline stage between execute and memory phases.
- Test a simple assembly program which has no data dependency between its instructions and verify your implementation.