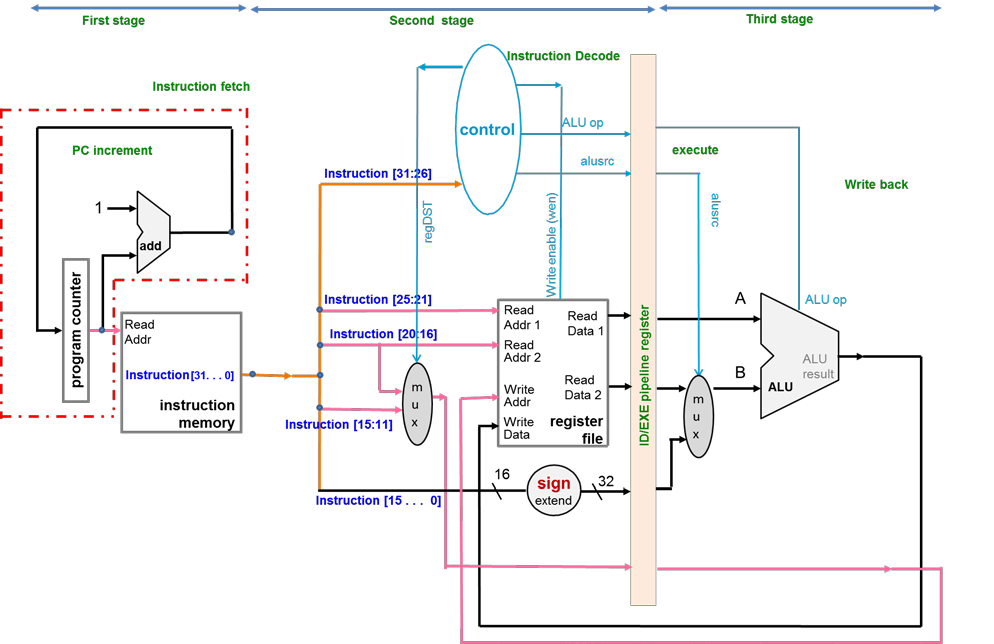
**PART A (20 Marks)**

1. You will use the data path designed in lab8 i.e., Hardware Lab 2.
2. You will write Verilog code for a program counter and insert appropriate pipeline register for IF/ID stage.
3. The instruction memory is already given to you with a set of instructions.
4. Using (1), (2) and (3), you will create a simple three-stage pipelined 32-bit CPU for execution of register (R) and I type instructions only. The Register File has thirty-two 32-bit registers (as in lab 8).
5. You will simulate the three-stage pipelined implementation for R-type and I-type instructions and show screen shots of simulation waveform to show that your CPU is working as intended.
6. You will report the area of your 32-bit CPU.
7. The skeleton testbench code of three-stage pipelined implementation for R and I-type instructions is given to you.

**PART B (20 Marks)**

1. You will implement a four-stage pipelined 32-bit CPU architecture by making changes to the design in Part A. You will have to insert the EXE/WB stage pipeline register.
2. You will simulate the four-stage pipelined 32-bit CPU architecture for R and I-type instructions and show screen shots of simulation waveform to show that your CPU is working as intended using the same set of instructions as those in Part A. You may have to add some NOPs. If you add NOPs, write down the complete set of instructions after adding NOPs.
3. The skeleton testbench code of four-stage pipelined implementation for R and I-type instructions is given to you.
4. You will report the area of the four-stage pipelined 32-bit CPU and compare the same with the three-stage pipelined architecture done in Part A.

# THREE-STAGE PIPELINED IMPLEMENTATION OF CPU( FOR REGISTER TYPE AND ADDI INSTRUCTIONS ONLY)



**IF/ID Pipeline Register**

**Fig. 1 Three-stage pipelined CPU (R and ADDI-type instructions). Note that PC increment is not considered as part of the first stage.**

The Verilog file ‘pipelined\_3stage.v’ (provided) is the top module of the three-stage pipelined CPU. This top module should instantiate other modules in your design in Part A.

Note that as there are no load and store instructions implemented in this datapath, we initialize the registers inside ‘register.v’(please note regdata[8] is initialized as 5 and regdata[3] is initialized as 2 in ‘regfile.v’).

1. Before doing step the synthesis, right click “Synthesis” and select “Synthesis Settings”. Then set -max\_dsp to 0. View the RTL schematic (recall steps from HW Lab 1). It should closely resemble Fig. 1 after you have modified the code properly.

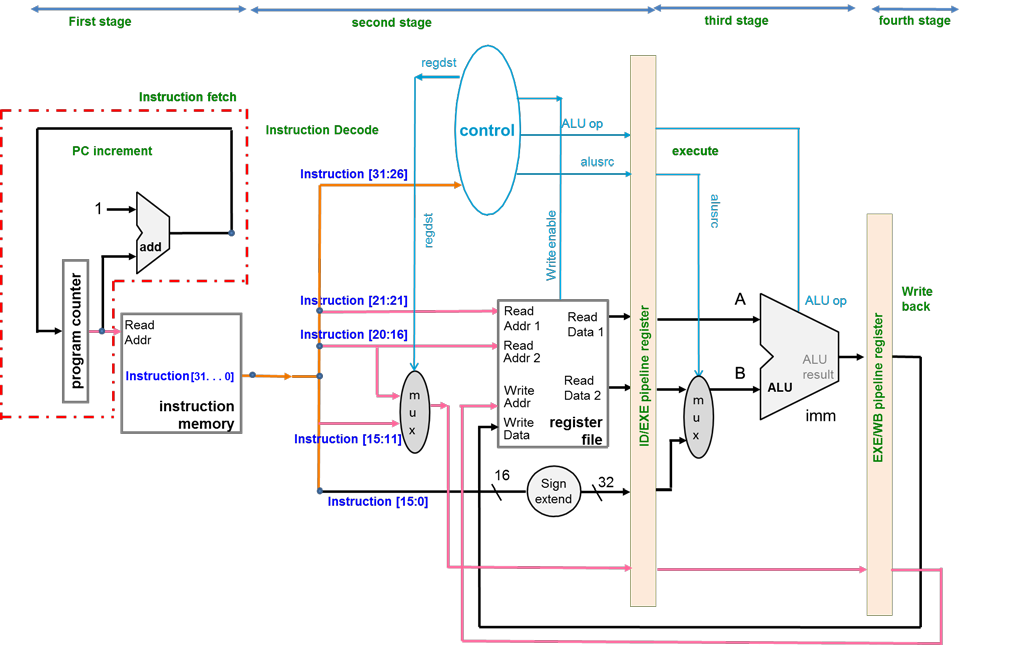
1. Test the three-stage pipelined CPU implementation using the test bench given ‘testbench\_3\_stage\_pipeline.v’. Note the operation and analyze the same. The ‘memory.v’ uses file operation to get the instruction and it reads “imem\_txt0.txt” as shown in Fig.2 as per the address given by PC. Verify the operation. You can add more inputs to fully check the functionality.

1. Note the cycle-by-cycle changes in the testbench for each instruction as well as the changes for the destination register content once you execute one instruction.

|  |  |  |
| --- | --- | --- |
| 32 bit PC address (in hex) | 32 bit instruction from IMEM | meaning |
| 00000000 | 00000000 | NOP |
| 00000001 | 05031000 | SUB $2,$8,$3 |
| 00000002 | 00430800 | ADD $1,$2,$3 |
| 00000003 | 0901F000 | AND $30, $8,$1 |
| 00000004 | 1502F800 | MUL $31,$8,$2 |
| 00000005 | 03fA5000 | ADD $10, $31,$26 |
| 00000006 | 18E40001 | ADDI $4,$7,1 |
| 00000007 | 00000000 | NOP |
| 00000008 | 00000000 | NOP |

**Fig. 2: R and ADDI-type instructions in the IMEM text file**

# FOUR STAGE PIPELINED IMPLEMENTATION OF R & I- TYPE 32 BIT CPU



**IF/ID Pipeline Register**

**Fig. 4 Four-stage Pipelined CPU implementation diagram (R and ADDI type instructions)**

Repeat steps 1, 2 and 3 as earlier for 3-stage processor for your understanding.