# Due Date: April 20, 2021

## (Worth about 55% of the Project, + Bonus of about 20%)

**Goals:** 1. To design control unit, additional hardware for handling branch instruction, another memory unit for data memory and auxiliary multiplexers
2. Modify ALU to handle branch instructions (implementation of ovf is optional/bonus)
3. Combine all the phases of the project for **single cycle implementation** (Figure 1).
4. To **implement the piplined architecture** leveraging the single cycle processor design without the forwarding unit and hazard unit (Figure 2).

5. Bonus: Implement either **Stall or Forwarding (or both)** and demonstrate it for data-hazards (Figure 3). (Up to 20% worth of points if you successfully implement both, you'll not be penalized at all for not finishing the bonus tasks)

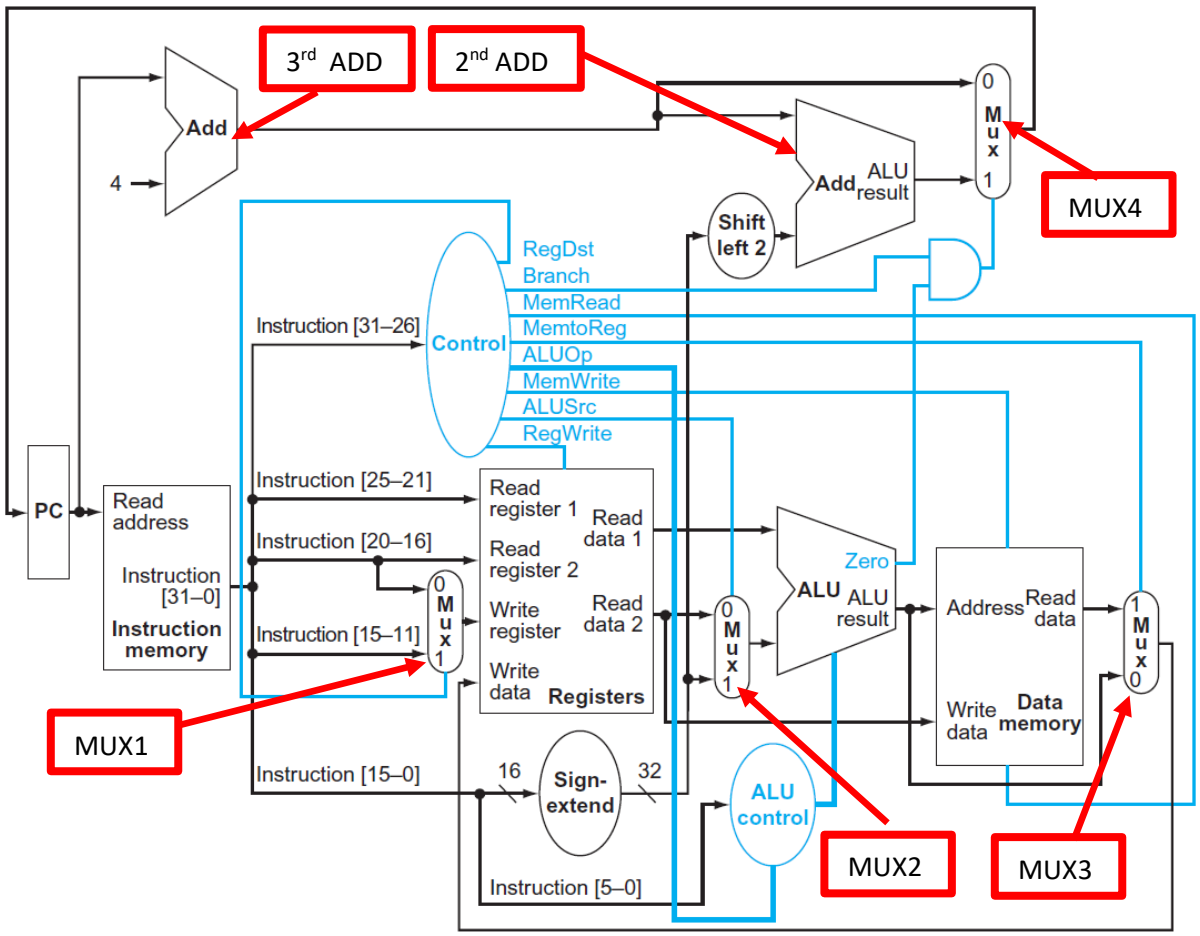


Figure 1. Single Cycle Processor (Required to get full credit)

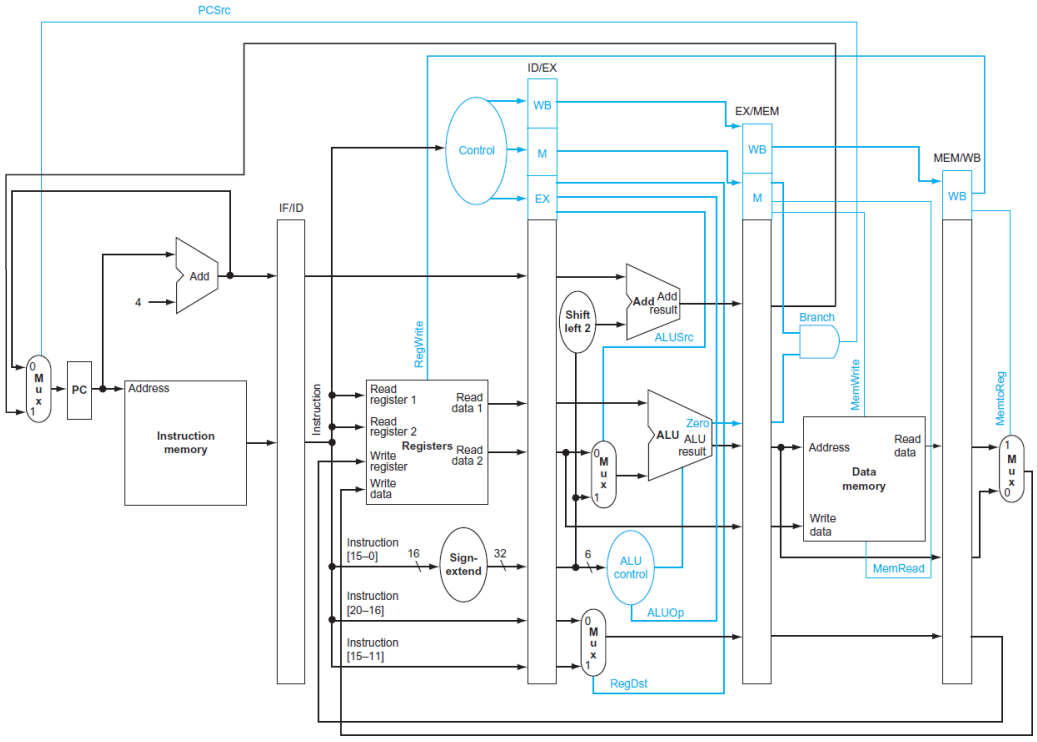


Figure 2. Simple Pipelined Architecture (Required to get full credit)

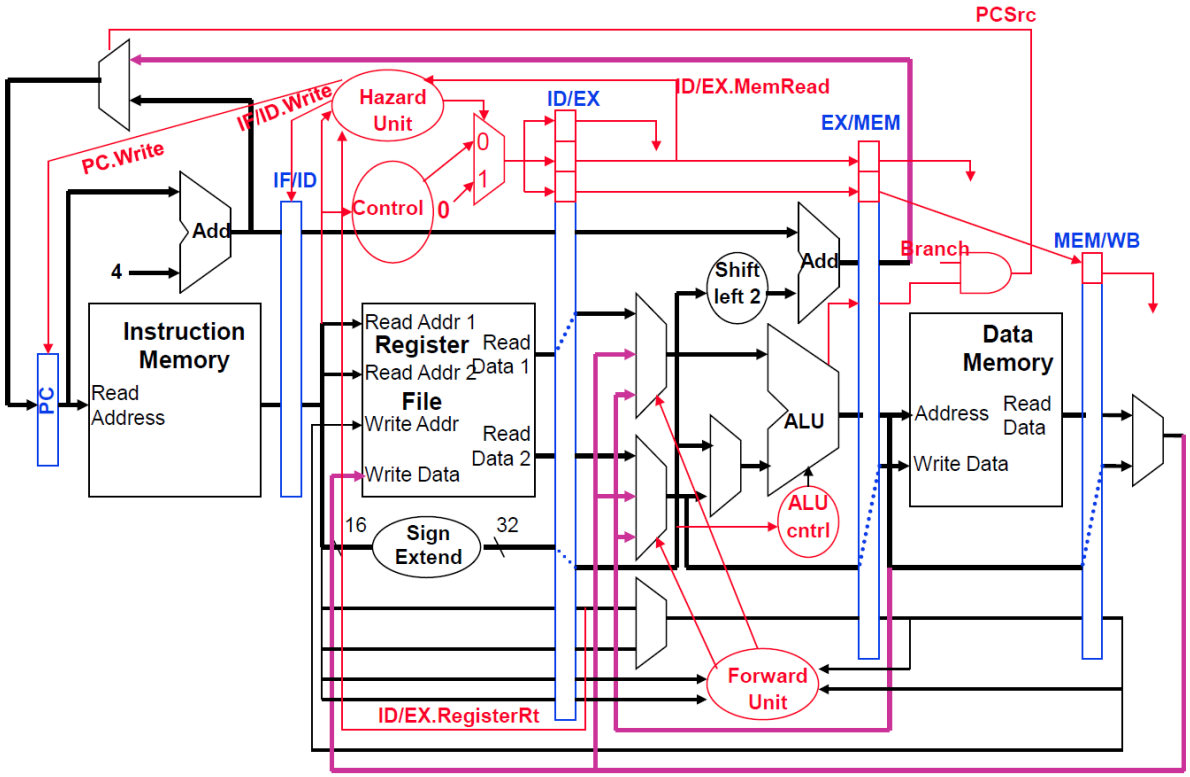


Figure 3. Pipeline with Forwarding and Hazard Units (Bonus)

# High level Description of each unit:

**You have already designed following units:**

Phase 1. PC, Instruction Memory, 1st Add units
Phase 2. Register, Sign Extend, Mux 2, ALU

In this phase you need to design and modify following units for Single Cycle Implementation:

1. Control unit, which has opcode as input and 8 outputs:
    a. Regdst → select signal for Mux1 to decide write address operand
    b. Branch → Indicates whether the instruction is a branch or not. This signal feeds to MUX4 select bit via AND gate.
    c. RegWrite → To write value into the register
    d. ALU Src → Select signal for MUX 2 (this MUX is already included in Phase 2 of the project), which decides the 2nd operand input to the ALU
    e. And f. **MemRead and MemWrite** → These signals are bolded because it needs to be connected with the wren of the memory mega function (Hint. You may not need to use both the signals).
    g. MemtoReg is the select signal to Mux3
    h. Branch is the input to AND gate
    i. ALUop decides what operation to perform in ALU, note that this is an input to ALU control unit.

2. Modify the ALU code given in phase 2 of the project to accommodate branch operation.
3. 2nd Add unit.
4. Three Multiplexers Mux 1, Mux 3 and Mux 4
5. ALU control Unit (you can make use of Figure 4.12 on Page 260 to implement it).
6. For Figure 2 you need to implement the additional registers and hold the control signals in these registers as well.

## *Constraints:*

1. Use another instantiation of the same memory megafunctions (IP) for data memory and modify as required.
2. Modify the general ALU code (pre-tested) that is provided in appendix of phase 2 of the project.
3. PC, Instruction Memory, Register, Control Unit, Data Memory all **should be clocked with the same clock source** for the single cycle implementation.
4. For testing purposes, follow the test bench requirements provided below.

### *Test Bench Requirements:*

Write testbench to execute following sets of MIPS instructions (for single cycle and pipelined implementation, implement these instructions with branch not taken).

i. beq $t1, $t2, Equal # if t1 equal t2 then go to Equal (iv instruction)
ii. add $t1, $t1, $t2
iii. sw $t1, 100($t2)
iv. or $t1, $t2, $t2

**For bonus points** Implement all the five instructions for the case of branch taken to show both stall and forward hazard units working properly.

i. beq $t1, $t2, Equal # if t1 equal t2 then go to Equal (iv instruction)
ii. add $t1, $t1, $t2
iii. or $t1, $t1, $t2
iv. lw $t4, 0($t1)
v. add $t2, $t4, $t2

## Deliverables:

### I. A pdf file containing following:

1. Modified block diagrams for two cases (three cases for bonus points), showing the clock signal, and explanation if any of the clock edges needs further explanation.
2. What are the objectives of this phase
3. Elaboration on VHDL implementation of each unit
4. Use the same device/board for synthesis that you used in ECE 4110 and show the snapshot of that device selection.
5. Flow summary, RTL view and Technology map view
6. Elaboration on test bench: e.g. how you confirmed the execution of these instructions, You can preload the register.
7. Elaboration on the waveform and snapshot of the waveform should also be included. Show both loading the register and writing the result into the registers using the waveforms. Clearly show all the control signal outputs. All the write operations should be done at the beginning of the cycle. Have two different set of waveforms for single cycle implementation and pipeline implementation. (For bonus points have a separate set of waveforms for all the five instructions).

   You should use as many waveforms as you feel necessary to elaborate nicely. Elaboration should state whether the waveform show the successful implementation, and if so why?

8. Timing Analyzer report on your setup and hold time and fmax report. Were you able to remove setup time slack??
9. State the improvements you noticed using pipelining in latency, throughput and Fmax.
10. State the hardware overhead related to pipeline only.
11. **For Bonus points:** In addition to above items, state the performance penalty associated with having the forwarding and hazard units.
12. **For Bonus points:** In addition to above items, state the hardware overhead associated with having the forwarding and hazard units.

II. **A zipped VHDL Implementation of the Project**: A folder containing all the files generated by Quartus II, (Ver. 11), including the all the .vhd, should be submitted to the dropbox via ilearn. Name the submitted folder as your lastname_S21_phase3.

**Presentation and Demonstration:**

In the last week of the classes we are going to have presentations and demonstrations. Each student is going to elaborate on the key achievements of their work with 5-minute presentation, where the student will be elaborate on set of the best results.

Each student is going to demonstrate a working project with the best simulation results on their computer – more details will be provided later on.