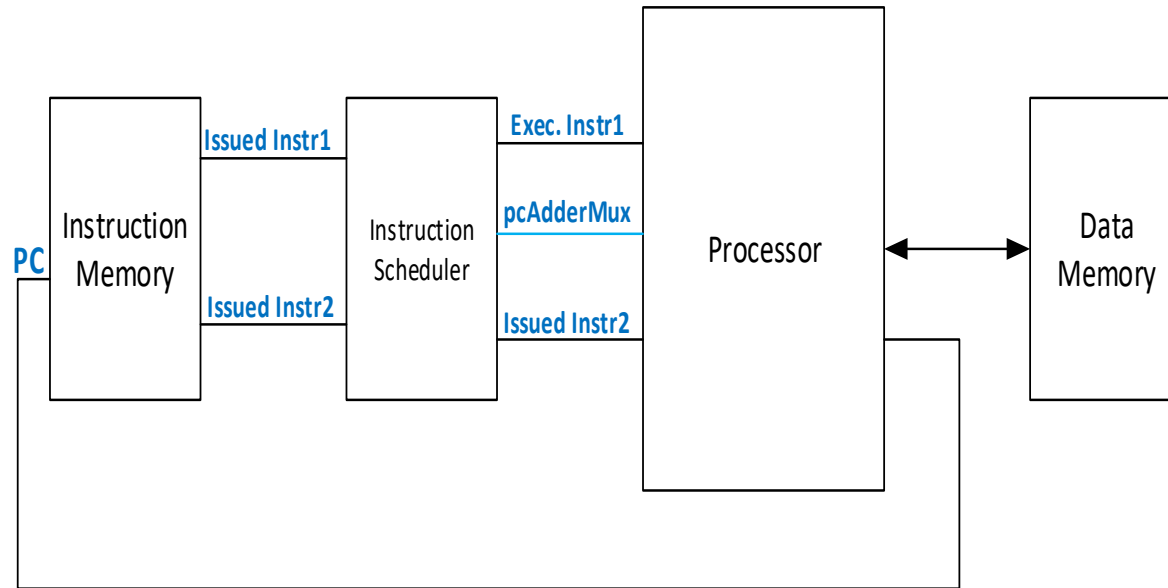


DESIGN METHODOLOGY AND STRATEGIES

X'TICS

- Duplicate functional units to allow multiple execution
- Processing is done **in-order**
- Dependences between a potential issue pair, that may result into a hazard are resolved by allowing only the first instruction in program order to execute.
- Hazards between succeeding and preceding stages are resolved using pipeline hazard resolution techniques

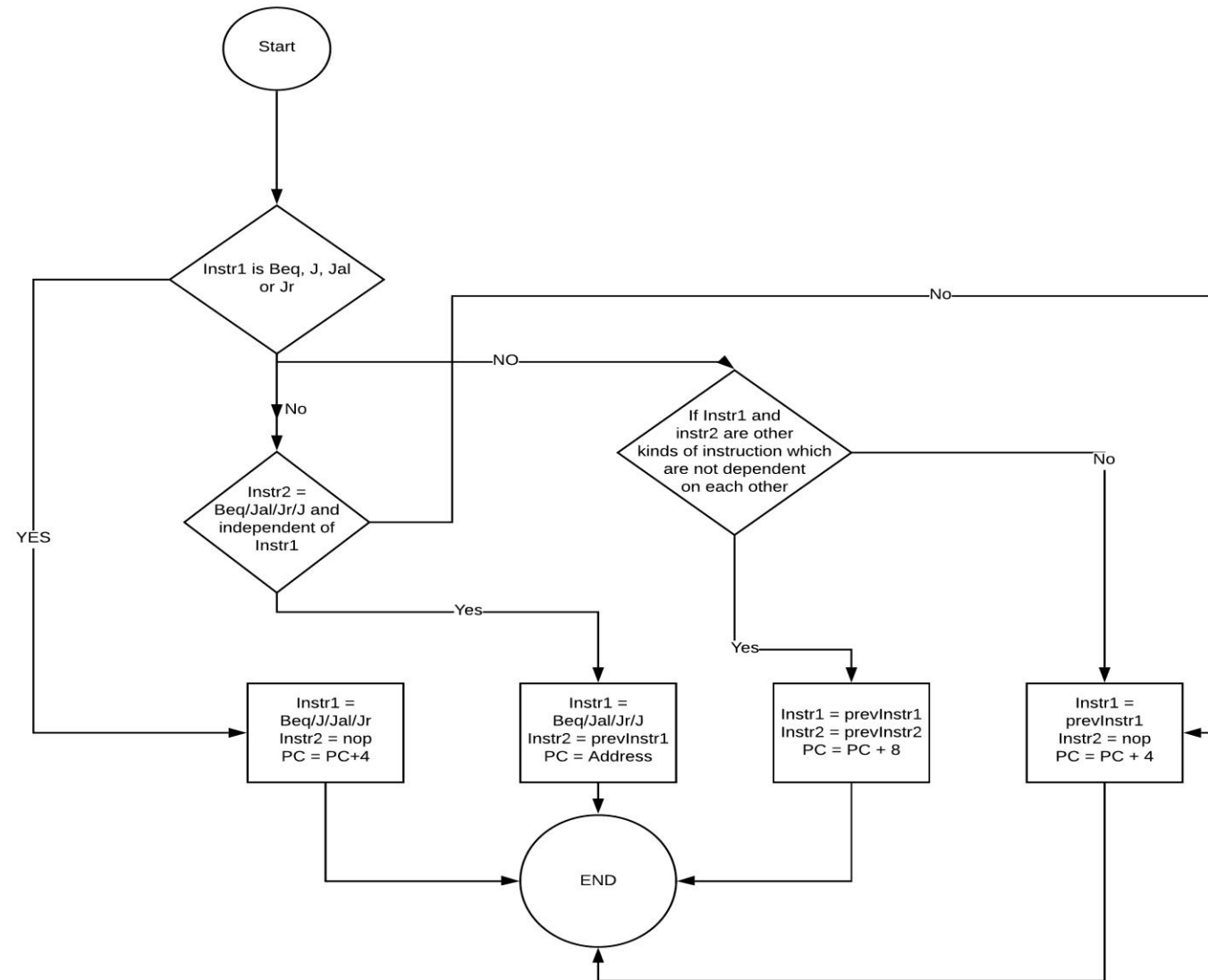
TOP LEVEL MODULE



- Instruction scheduler is between instruction memory and scalar processor to ensure no 2 instructions whose simultaneous execution may result into a hazard are executed together.
- Instruction scheduler generates a select signal that allows the pc be increased by 4 or 8

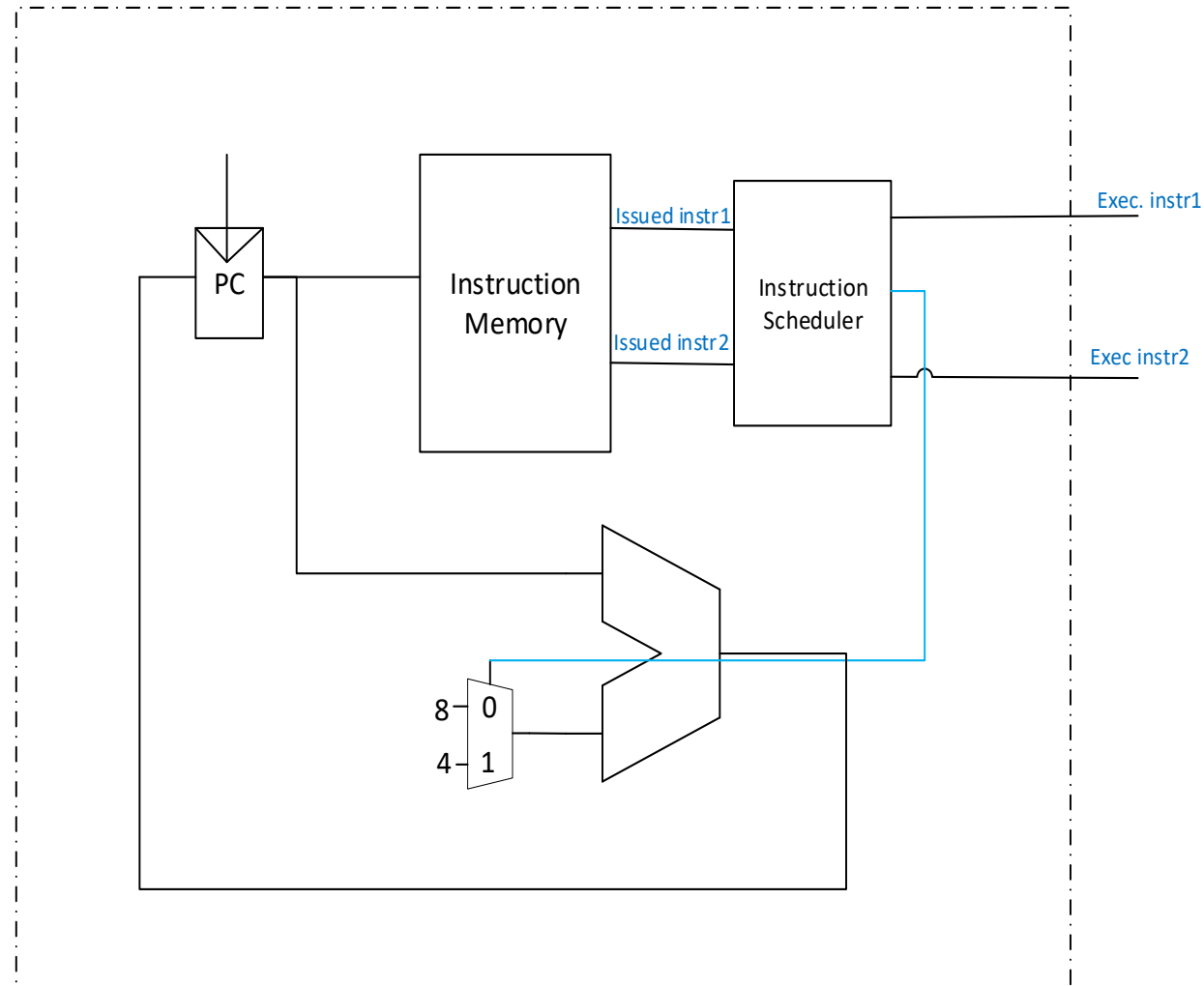
Instruction Scheduler (In depth)

- Determines if an execution pair is valid for execution
- Determination is done based on the existence of a data or control hazard
- Data and output dependences can be determined by examining the pair and the values in the fields of their architecture.
- The scheduler finally generates a mux select signal to select between the values 4 and 8 used to determine PC's next value



SIMPLIFIED FLOWCHART SHOWING OPERATION OF INSTRUCTION SCHEDULER

Testing the Instruction Scheduler

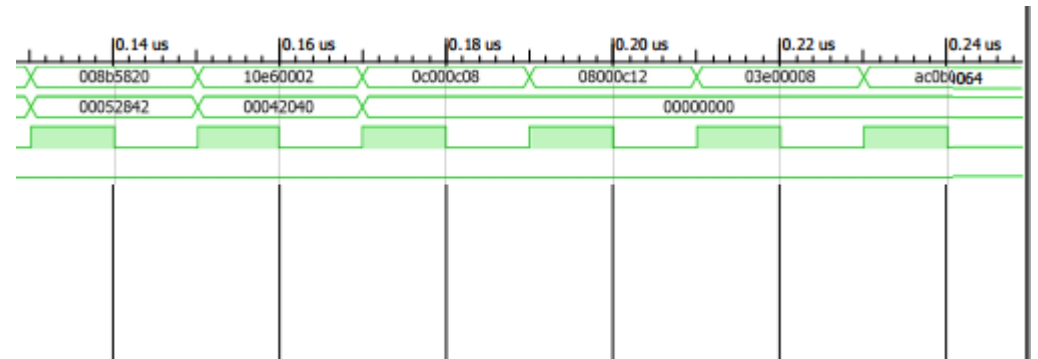
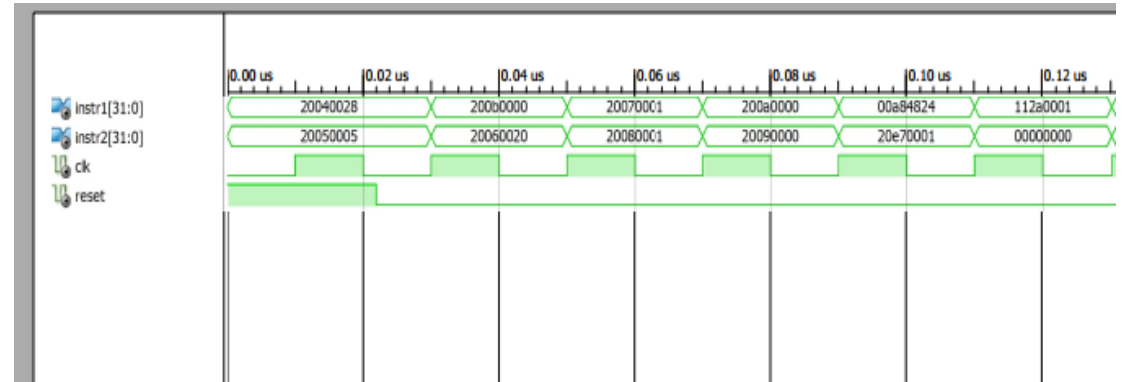


SIMULATION

```

addi $a0, $zero, 40           #20040028
addi $a1, $zero, 5            #20050005
addi $t3, $zero, 0            #200b0000
addi $a2, $zero, 32           #20060020
addi $a3, $zero, 1            #20070001
addi $t0, $zero, 1            #20080001
addi $t2, $zero, 0            #200a0000
addi $t1, $zero, 0            #20090000
CheckLastBit: and $t1, $a1, $t0 #00a84824
addi $a3, $a3, 1              #20e70001
beq $t1, $t2, shift           #112a0001
add $t3, $a0, $t3             #008b5820
shift: srl $a1, $a1, 1         #00052842
sll $a0, $a0, 1               #00042040
beq $a3, $a2, beforeEnd       #10e60002
jal CheckLastBit              #0c000c08
j end                          #08000c12
beforeEnd: jr $ra              #03e00008
end: sw $t3, 100($0)          #ac0b0064

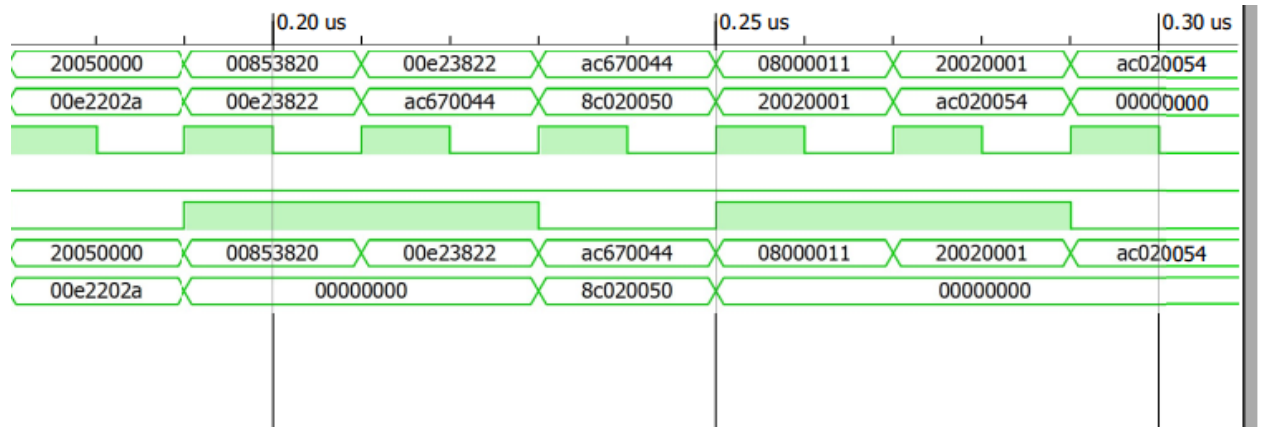
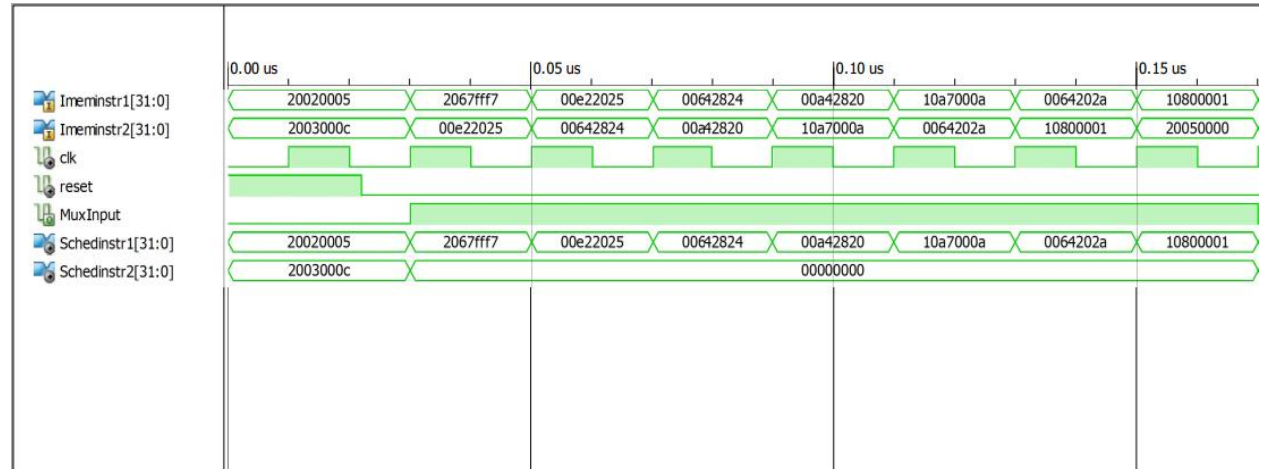
```



SIMULATION

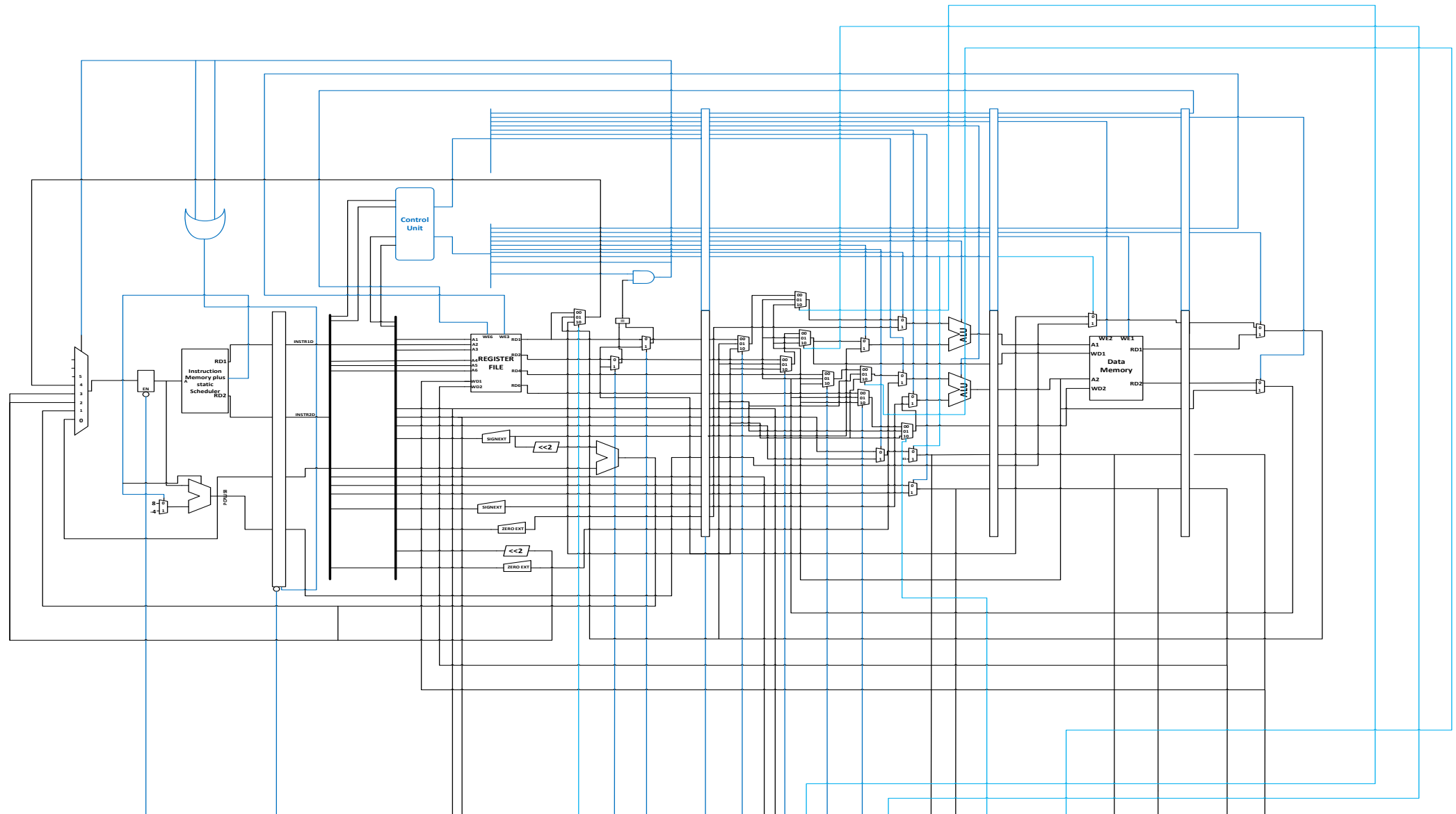
```

main: addi $2, $0, 5           #20020005
      addi $3, $0, 12          #2003000c
      addi $7, $3, -9          #2067fff7
      or $4, $7, $2            #00e22025
      and $5, $3, $4           #00642824
      add $5, $5, $4           #00a42820
      beq $5, $7, end          #10a7000a
      slt $4, $3, $4           #0064202a
      beq $4, $0, around       #10800001
      addi $5, $0, 0           #20050000
      around: slt $4, $7, $2   #00e2202a
      add $7, $4, $5           #00853820
      sub $7, $7, $2           #00e23822
      sw $7, 68($3)            #ac670044
      lw $2, 80($0)            #8c020050
      j end                    #08000011
      addi $2, $0, 1           #20020001
end: sw $2, 84($0)             #ac020054
  
```



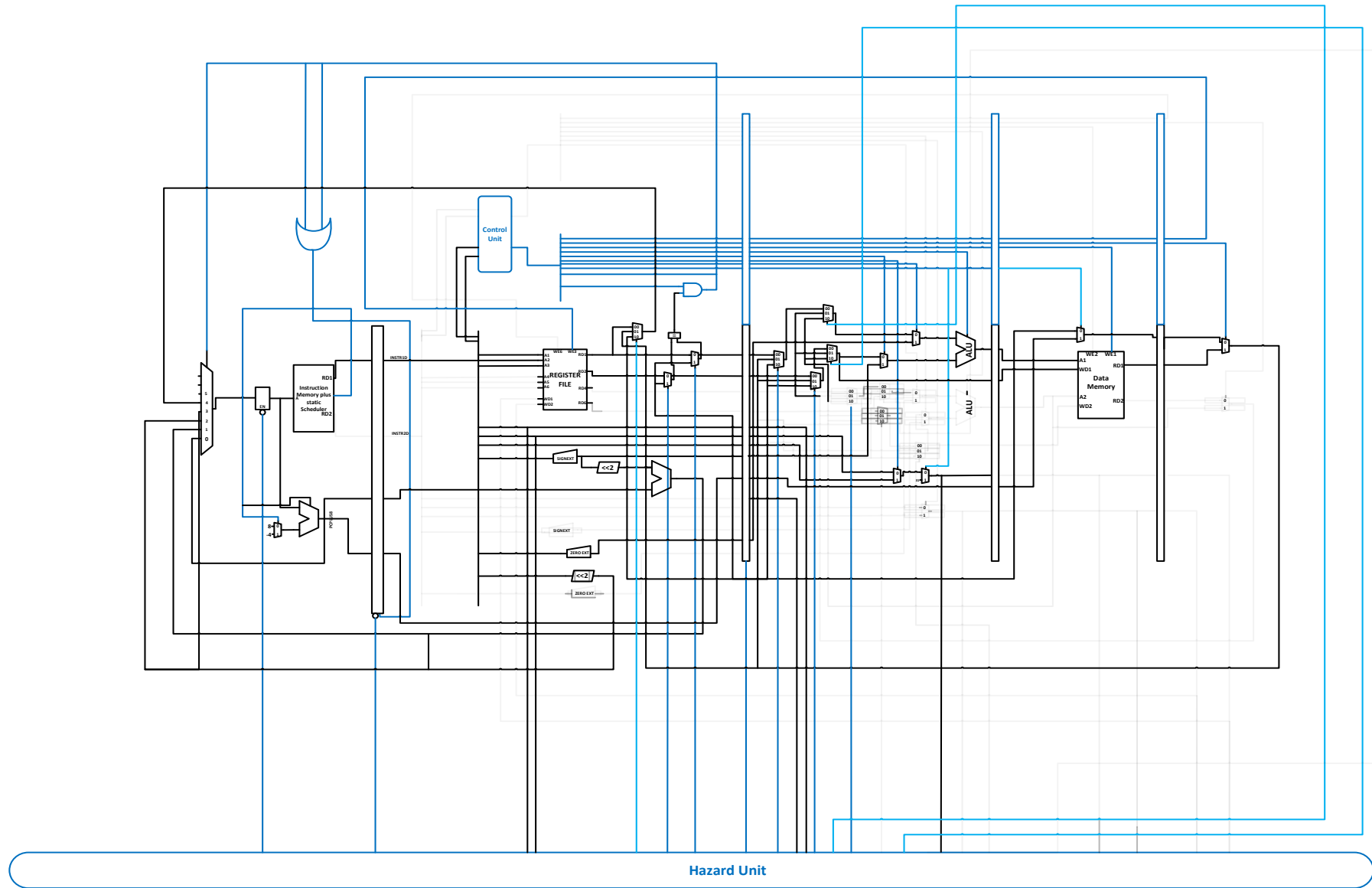
DATAPATH X'TICS

- Has enough functional units to support the execution of 2 instructions at a time;
IPC = 2
- Datapath is designed to accommodate conditional and unconditional Branches or Jumps on the first path only
- Accommodation is made possible by the operation of the instruction scheduler.
- Inter stage and inter path forwarding
- Inter stage data and control hazards are resolved by s

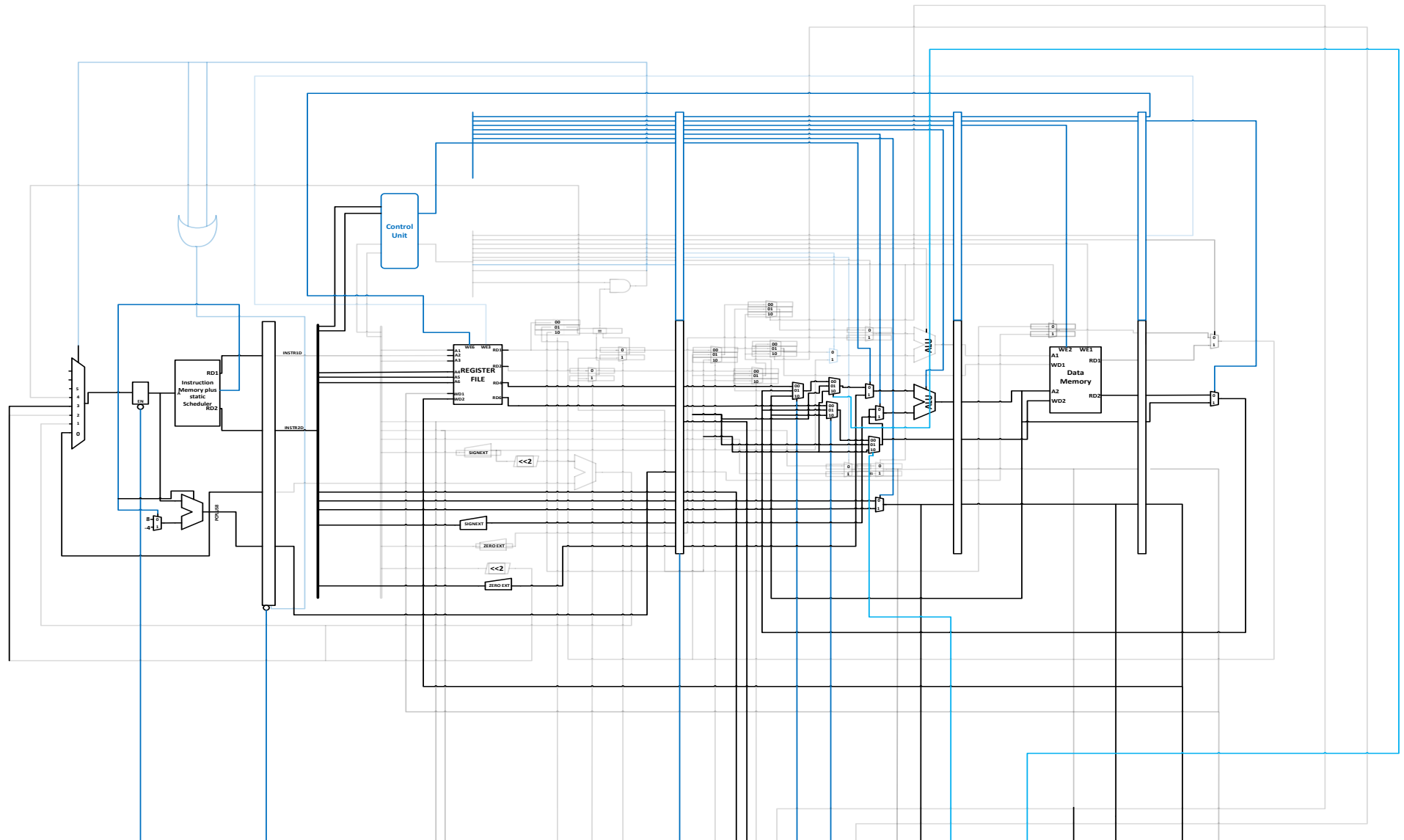


Hazard Unit

FULL PATH



PATH 1



Hazard Unit

PUTTING EVERYTHING TOGETHER AND SIMULATION

NEXT STEPS :

Design of a Dynamically Scheduled OOO processor based on the Tomasulo's algorithm with speculation

DYNAMIC SCHEDULING & TOMASULO'S ALGORITHM

