5 Stage Pipelined MIPS R3000 Microprocessor

Features Implemented:

All the 20 Instructions given in the assignment has been implemented successfully with Data Hazard Detection, Data Forwarding, Control Hazard Detection & Branch Delay Slot.

i. Data Hazard Detection:

For Implementing Hazard Detection the pipeline has been stalled when the following condition has been detected:
if (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt))) => **stall the pipeline**. This Hazard detection unit has been implemented in the Instruction DECODE stage of the pipeline and was found successful to prevent the situations where the result written in any instruction is used by the subsequent instruction as its operands. With the forwarding unit implemented in the design the stall cycle in the situation where the operand has been used immediately in the next instruction after the LOAD has been reduced from 2 to 1. Data Hazard Detection has been verified in the Implementation results below. Eg cases: LW $2, 20($1) followed by ADD $4, $2, $5 & SUB $2, $1, $3 followed by AND $12, $2, $5

ii. Data Forwarding:

For implementation of Data forwarding two 3 input Mux's have been added before the ALU in the EX stage and the inputs to the ALU have been forwarded from the MEM & WB stage whenever the register required by the instruction in the EX stage matched with the register/s in the MEM and/or WB stage. Forwarding the values from the MEM stage, WB stage & both MEM and WB stage simultaneously has been successfully verified in the Implementation results below. Eg case: LW $16, 8($0) followed by ADD $10, $1, $16 followed by ADD $17, $10, $16

iii. Control Hazard Detection:

Control Hazards are caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps). In the given design all the conditional branches has been considered as **not taken**. If the branches are detected to be taken the IF/ID FF is being flushed and the pipeline is loaded again with the new instructions from the target address. This Control Hazard Detection logic is being implemented in the EX stage of the pipeline as the ALU is being used for the branch outcome decision. All the Branch and Jump instructions have been verified in the Implementation results below.

iv. Branch Delay Slot:

Implementing the Branch Delay Slot requires that the instruction following the Branch and Jumps is executed regardless of the outcome of the Branch. For implementing the same the ID/EX FF is **not flushed** even if the branches are found to be taken.

IPC Calculation:

For IPC calculation loop consisting of 50 instructions containing all given 20 instructions has been coded and this loop is iterated 10 times, thus leading to 50 * 10 = 500 instruction execution. Number of cycles found to execute given 500 instructions is found to be 554. Hence IPC = 500/554 = 0.9025 is being recorded.

Design Parameters:

'area_umsq': 11041.438061
'delay_ns': data required time - slack = 0.7935 - (- 3.5629) = 4.3563
'dyn_power_mw': 0.0998423
'error_count': 0.00 out of all tests performed
'lkg_power_mw': 0.0085090
'target_ns': 2.0

Design Size & Composition:

Number of ports:	278
Number of nets:	3503
Number of cells:	3206
Number of combinational cells:	2827
Number of sequential cells:	379
Number of macros/black boxes:	0
Number of buf/inv:	391
Number of references:	48

Implementation of Results:

```
clk_cnt= 11 ICache: iCRdAdr= 4000 iCRdDat=8c010001   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0= 0 rfWrDat_p0=00000000
```

- Comments: Input PC = 4000, Instruction = LW, load $r1 = 00f0f0f0, Output Received = None

```
clk_cnt= 12 ICache: iCRdAdr= 4004 iCRdDat=8c020002   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 1 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC = 4004, Instruction = LW, load $r2 = 000f0f0f, Output Received = None

```
clk_cnt= 13 ICache: iCRdAdr=4008 iCRdDat=8c030004   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 2 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC = 4008, Instruction = LW, load $r3 = 0000000a, Output Received = None

```
clk_cnt= 14 ICache: iCRdAdr=4012 iCRdDat=8c1e0008   DCache:dCAdr=         1 dCRdDat=00f0f0f0 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 3 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC = 4012, Instruction = LW, load $r30 = 00000fa0, Output Received = None
  Starting Toggle Collection

```
clk_cnt= 15 ICache: iCRdAdr= 4016 iCRdDat=00415020   DCache:dCAdr=         2 dCRdDat=000f0f0f dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1=30 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 1 rfWrDat_p0=00f0f0f0
```

- Comments: Input PC = 4016, Instruction = ADD, $r1+$r2 => $r10, Output Received = for 1st LW instruction

```
clk_cnt= 16 ICache: iCRdAdr= 4020 iCRdDat=14220008   DCache:dCAdr=         4 dCRdDat=0000000a dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 2 rfRdDat_p0=000f0f0f rfRdAdr_p1= 1 rfRdDat_p1=00f0f0f0 rfWAdr_p0= 2 rfWrDat_p0=000f0f0f
```

- Comments: Input PC = 4020, Instruction = BNE, $r1 != $r2 => Branch to target, Output Received = for 2nd LW instruction

```
clk_cnt= 17 ICache: iCRdAdr= 4024 iCRdDat=0022482a   DCache:dCAdr=         8 dCRdDat=00000fa0 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1= 2 rfRdDat_p1=000f0f0f rfWAdr_p0= 3 rfWrDat_p0=0000000a
```

- Comments: Input PC = 4024 (1st Inst of Branch Delay Slot, will be executed), Instruction = SLT, $r1< $r2 => $r9 = 1, Output Received = for 3rd LW instruction

```
clk_cnt= 18 ICache: iCRdAdr= 4028 iCRdDat=00415020   DCache:dCAdr= 16777215 dCRdDat=00000000 dCWDat=00f0f0f0
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1= 2 rfRdDat_p1=000f0f0f rfWAdr_p0=30 rfWrDat_p0=00000fa0
```

- Comments: Input PC = 4028 (2nd Inst of Branch Delay Slot, won't get executed), Instruction = ADD, $r1+$r2 => $r10, Output Received = for 4th LW instruction

```
clk_cnt= 19 ICache: iCRdAdr= 4056 iCRdDat=342c0f0f   DCache:dCAdr=         0 dCRdDat=00000000 dCWDat=000f0f0f
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0=10 rfWrDat_p0=00ffffff
```

- Comments: Input PC = 4056 (Branch Taken), Instruction = ORI, $r1+0f0f => $r12, Output Received = for add Inst on pc 4016

```
clk_cnt= 20 ICache: iCRdAdr= 4060 iCRdDat=10220004   DCache:dCAdr=         0 dCRdDat=00000000 dCWDat=000f0f0f
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1=12 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=00000000
```

- Comments: Input PC = 4060, Instruction = BEQ, if $1 == $2 => Branch to target, Output Received = ALU outputs 0 for BNE on pc 4020

```
clk_cnt= 21 ICache: iCRdAdr= 4064 iCRdDat=00225083   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1= 2 rfRdDat_p1=000f0f0f rfWAdr_p0= 9 rfWrDat_p0=00000000
```

- Comments: Input PC = 4064(BEQ is not taken hence normal execution), Instruction = SRA, $2 >> shamt(2) => $10, Output Received = SLT (Branch Delay Slot on pc 4024)

```
clk_cnt= 22 ICache: iCRdAdr=4068 iCRdDat=8c100008   DCache:dCAdr= 15794175 dCRdDat=00000000 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1= 2 rfRdDat_p1=000f0f0f rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC = 4068, Instruction = LW, load $r16 = 00000fa0, Output Received = Single cycle stall for the BNE instruction

```
clk_cnt= 23 ICache: iCRdAdr= 4072 iCRdDat=00305020   DCache:dCAdr=         1 dCRdDat=00000000 dCWDat=000f0f0f
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1=16 rfRdDat_p1=xxxxxxxx rfWAdr_p0=12 rfWrDat_p0=00f0ffff
```

- Comments: Input PC = 4072, Instruction = ADD, $r1+$r16 => $r10 (Forwarding the value of $r16 from MEM stage), Output Received = for ORI Instruction at pc 4056

```
clk_cnt= 24 ICache: iCRdAdr= 4076 iCRdDat=01508820   DCache:dCAdr=    246723 dCRdDat=00000000 dCWDat=000f0f0f
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1=16 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=00000001
```

- Comments: Input PC = 4076, Instruction = ADD, $r10+$r16 => $r17 (Forwarding the value of $r16 from WB stage and $r10 from MEM stage), Output Received = for BEQ Instruction ALUResult is zero if both operands are equal, here branch not taken hence 1'b1

```
clk_cnt= 25 ICache: iCRdAdr= 4076 iCRdDat=01508820   DCache:dCAdr=         8 dCRdDat=00000fa0 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1=16 rfRdDat_p1=xxxxxxxx rfWAdr_p0=10 rfWrDat_p0=0003c3c3
```

- Comments: Input PC= 4076 (stall for LW Inst on pc 4068), Instruction = ADD, $r10+$r16 => $r17, Output Received = for SRA Inst on pc 4064

```
clk_cnt= 26 ICache: iCRdAdr=4080 iCRdDat=0800041a   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0=10 rfRdDat_p0=0003c3c3 rfRdAdr_p1=16 rfRdDat_p1=00000fa0 rfWAdr_p0=16 rfWrDat_p0=00000fa0
```

- Comments: Input PC= 4080, Instruction = Jump, Jump to target, Output Received = for LW Inst on pc 4068

```
clk_cnt= 27 ICache: iCRdAdr=4084 iCRdDat=ac1e0040   DCache:dCAdr= 15794320 dCRdDat=00000000 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0=10 rfWrDat_p0=ffffffff
```

- Comments: Input PC= 4084, (1st Inst of Branch Delay Slot, will be executed), Instruction = SW, Store Mem [64] to $r30, Output Received = stall for the LW instruction at pc 4068 due to the data dependency on $r10 used immediately by ADD on pc 4072

```
clk_cnt= 28 ICache: iCRdAdr= 4088 iCRdDat=00415020   DCache:dCAdr= 15798320 dCRdDat=00000000 dCWDat=00000fa0
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1=30 rfRdDat_p1=00000fa0 rfWAdr_p0=10 rfWrDat_p0=00f10090
```

- Comments: Input PC= 4088, (2nd Inst of Branch Delay Slot, won't get executed), Instruction = ADD, $r1+$r2 => $r10, Output Received = for ADD at pc 4072 (Forwarding the value of $r16 from MEM stage)

```
clk_cnt= 29 ICache: iCRdAdr= 4200 iCRdDat=8c190040   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0=17 rfWrDat_p0=00f11030
```

- Comments: Input PC= 4200 (Due to Jump Instruction), Instruction = LW, load $r25= 00000fa0, Output Received = for ADD at pc 4076 (Forwarding the value of $r16 from WB stage and $r10 from MEM stage)

```
clk_cnt= 30 ICache: iCRdAdr= 4204 iCRdDat=03200008   DCache:dCAdr=        64 dCRdDat=00000000 dCWDat=00000fa0
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1=25 rfRdDat_p1=xxxxxxxx rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC= 4204, Instruction = JR, jump to Mem [$25], Output Received = for SW on pc 4084

```
clk_cnt= 31 ICache: iCRdAdr= 4208 iCRdDat=382d0f0f   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0=25 rfRdDat_p0=xxxxxxxx rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0= 0 rfWrDat_p0=00000000
```

- Comments: Input PC= 4208, Instruction = XORI, $r1 ^ 0f0f => r13, Output Received = for the Flush due to jump Inst on pc 4080

```
clk_cnt= 32 ICache: iCRdAdr= 4208 iCRdDat=382d0f0f   DCache:dCAdr=        64 dCRdDat=00000fa0 dCWDat=xxxxxxxx
RegFile: rfRdAdr_p0=25 rfRdDat_p0=xxxxxxxx rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC= 4208(stall for LW Inst on pc 4200, 1st Inst of Branch Delay Slot, will be executed), Instruction = XORI, $r1 ^ 0f0f => r13, Output Received = Single cycle stall due to LW Inst at pc 4200

```
clk_cnt= 33 ICache: iCRdAdr= 4212 iCRdDat=01508820   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 1 rfRdDat_p0=00f0f0f0 rfRdAdr_p1=13 rfRdDat_p1=xxxxxxxx rfWAdr_p0=25 rfWrDat_p0=00000fa0
```

- Comments: Input PC= 4212 (2nd Inst of Branch Delay Slot, won't get executed), Instruction = ADD, $r10+$r16 => $r17, Output Received = for LW Inst at pc 4200

```
clk_cnt= 34 ICache: iCRdAdr= 4000 iCRdDat=8c010001   DCache:dCAdr=4294967295 dCRdDat=00000000 dCWDat=00000000
RegFile: rfRdAdr_p0= 0 rfRdDat_p0=00000000 rfRdAdr_p1= 0 rfRdDat_p1=00000000 rfWAdr_p0= 0 rfWrDat_p0=ffffffff
```

- Comments: Input PC= 4000 (JR Instruction), Instruction = LW, load $r1 = 00f0f0f0, Output Received = Default ALU for JR Inst as JR does not use ALU (ALUOut = 0xffff_ffff hardcoded in ALU if the ALUControl Signal is not received)