CO502-2021 : Advanced Computer Architecture

Part 4 (Hazard Control)

Group 4

Data Path

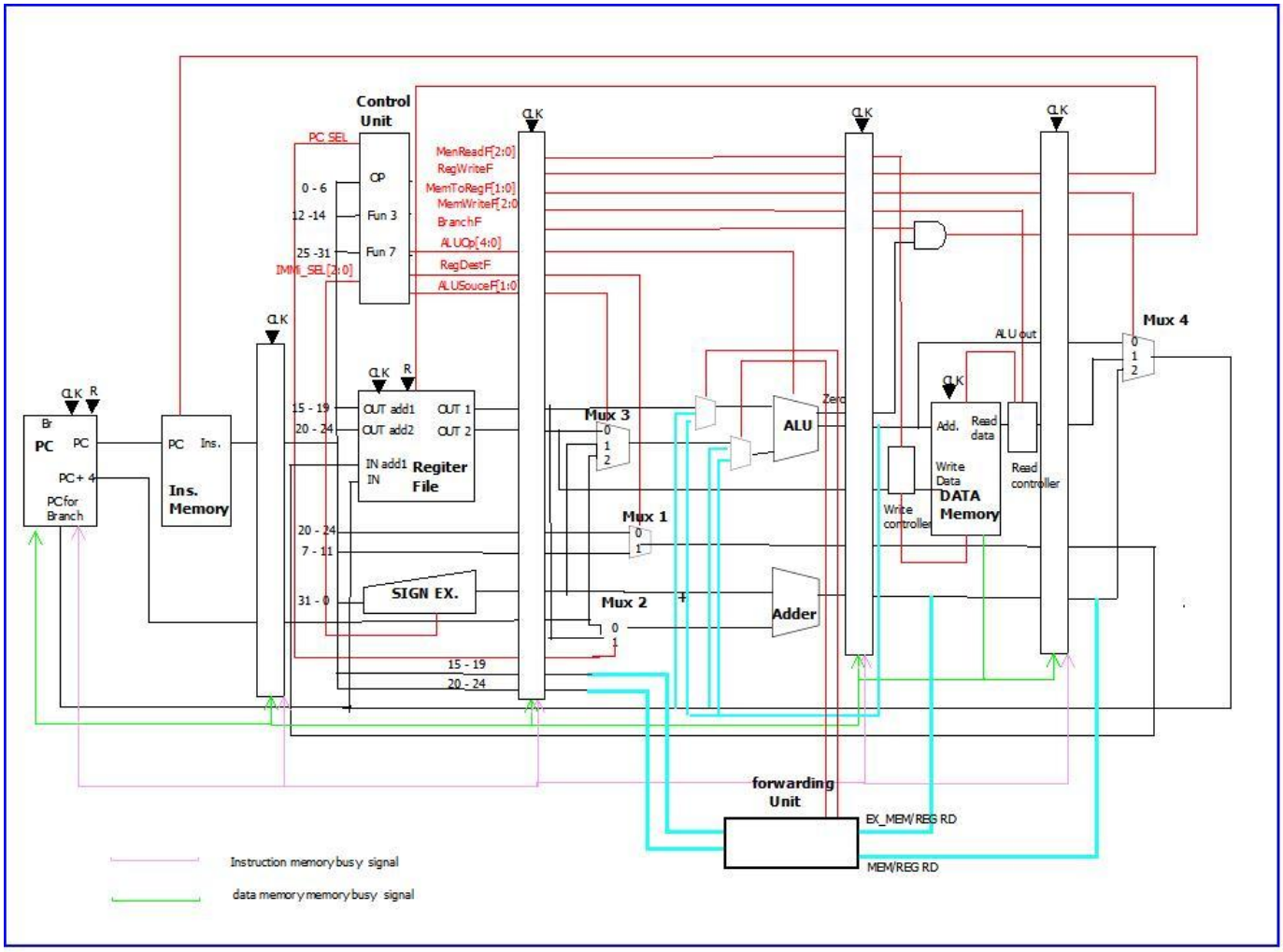

FIGURE 1 - Data path

Hardware Units

1. ALU - Arithmetic Logic Unit

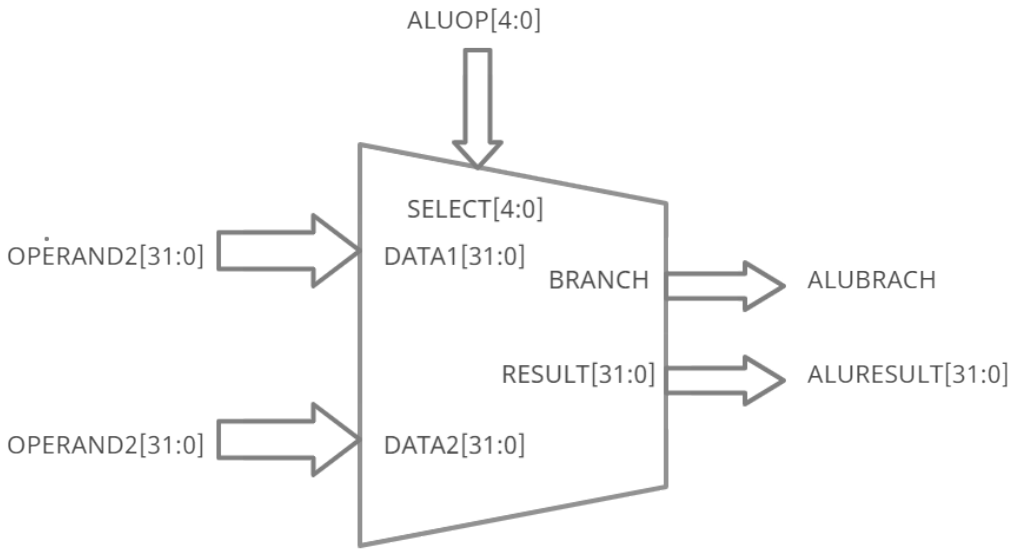

FIGURE 2 - ALU design

Inputs :

DATA1[31:0] - 32 bit operand 1
DATA2[31:0] - 32 bit operand 2
SELECT[4:0] - 5 bit ALUOp

Output :

RESULT[31:0] - 32 bit ALU Result
BRANCH - 1 bit Branch Result (which used for branch instructions)

- In the alu the main function is getting 2 DATA inputs and do the operation according to the SELECT(which contain the alu opcodes) and give RESULT and BRANCH as output.

- 5 bits are assigned for ALUOp and 32 options for SELECT. And used 17 options for giving various RESULT outputs. And used another 6 for giving BRANCH output. If there is no RESULT or BRANCH output needed for an ALUOp it returns 0 as default.

The ALUOps assigned as Table 1 for operation,

ALUOP[4:0]	RESULT[31:0]	BRANCH
00000	RES_ADD	0
00001	RES_SLL	0
00010	RES_SLT	0
00011	RES_SLTU	0
00100	RES_XOR	0
00101	RES_SRL	0
00110	RES_OR	0
00111	RES_AND	0
01000	0	BR_BEQ
01001	0	BR_BNE
01010	0	0
01011	0	0
01100	0	RES_BLT
01101	0	RES_BGE
01110	0	RES_BLTU
01111	0	RES_BGEU
10000	RES_SUB	0
10001	RES_FWD	1
10010	0	0
10011	0	0
10100	0	0
10101	0	0
10110	0	0
10111	0	0
11000	RES_MUL	0
11001	RES_MULH	0
11010	RES_MULHSU	0
11011	RES_MULHU	0
11100	RES_DIV	0
11101	RES_REM	0
11110	RES_FWD	0
11111	RES_REMU	0

Table 1 : ALU Op codes

- ALL the operation takes DATA2 which comes through the multiplexer therefore as an initial plan set 2 time unit delay.

- RESULT gives in 32 bit binary format therefore values which contain more than 32 bit send RESULT by negotiating extra bits. Therefore, It can't do a calculation where the result value exceeds the 32 bit and can't give the exact correct answers.

- The ALU module is tested using a test bench and all the functions work properly in the ALU model test.

2. Register File

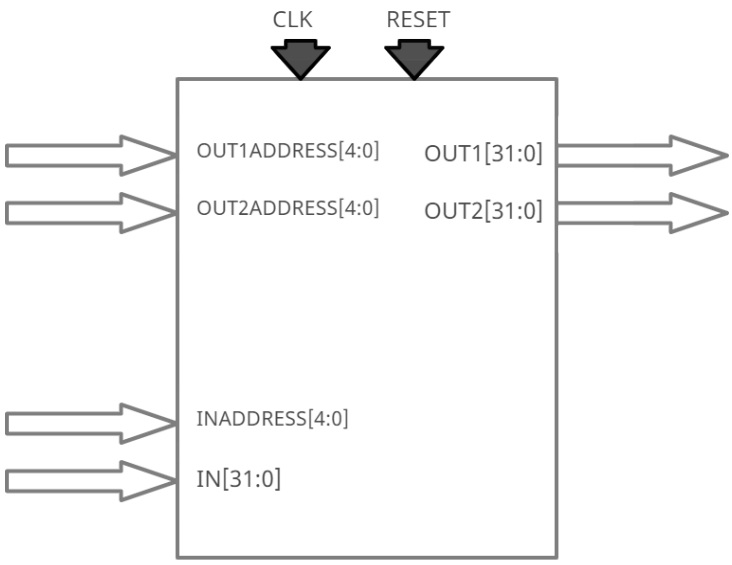

FIGURE 3 : Register file design

Inputs:

OUT1ADDRESS [4:0] - address of OUT1 (data which is read from the register file)
OUT2ADDRESS [4:0] - address of OUT2 (data which read from the register file)
INADDRESS[4:0] - address of IN (Data which is written on the register file)
IN [31:0] - The which need to write a register file
WRITEANABLE - the signal to enable the writing data on the register. This is 1 when I need to write on register
CLK - clock
RESET- result signal

Outputs:

OUT1 [31:0] - OUT1ADDRESS Data value 1 read from the register according to the OUT1ADDRESS

OUT1 [31:0] - OUT1ADDRESS Data value 1 read from the register according to the OUT1ADDRESS

The register file can keep 32 data values of 32 bits (4 Bytes).
After 2 time units Writing will happen to the Register.

3. Control Unit

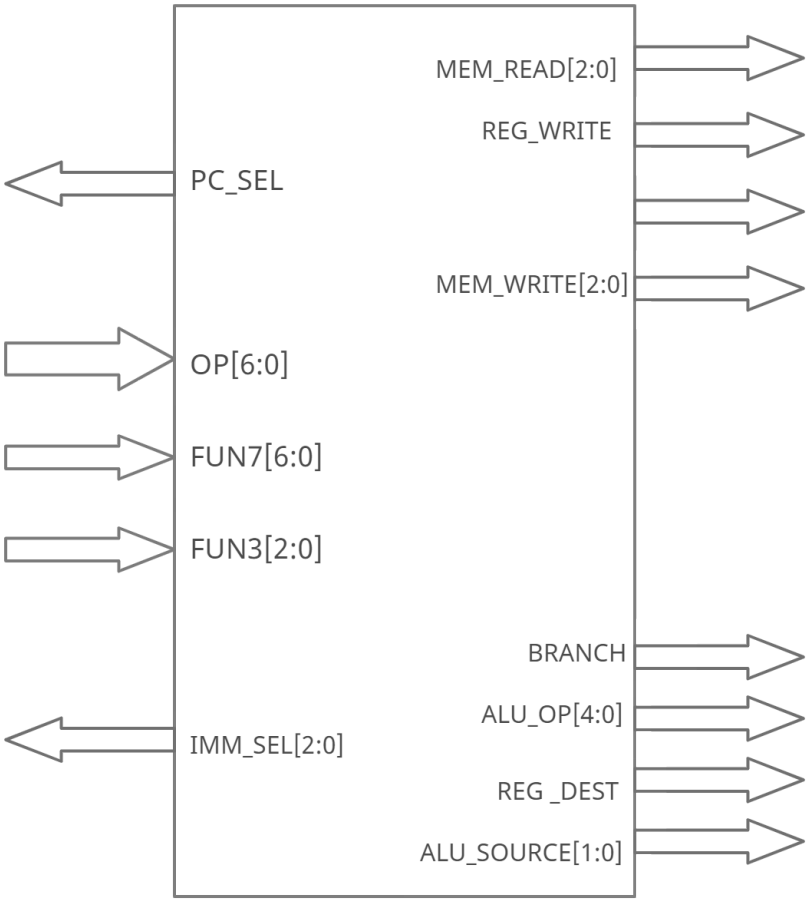

FIGURE 4 : Control unit design

Inputs :

OPCODE[6:0] - opcode of the instruction

FUN7[6:0] - function 7 of the instruction

FUN3[2:0] -function 3 of the instruction

Outputs :

Memory Read Flag

Register write Flag

Memory Write flag

Memory to register flag

Branch flag

Alu opcode

Register destination flag

Alu source flag

Immediate value selector

PC selector

There are 10 control signals which are controlled by this unit.

Instruction	Type	MEM read	MEM Write	Memory to reg	REG write	ALU source	Branch	Register destination	IMM_SEL	PC_SEL
LB	I-Type	1	0	1	1	1	0	1	0	x
LH	I-Type	2	0	1	1	1	0	1	0	x
LW	I-Type	3	0	1	1	1	0	1	0	x
LBU	I-Type	4	0	1	1	1	0	1	0	x
LHU	I-Type	5	0	1	1	1	0	1	0	x
ADDI	I-Type	0	0	0	1	1	0	1	0	x
SLTI	I-Type	0	0	0	1	1	0	1	0	x
SLTIU	I-Type	0	0	0	1	1	0	1	0	x
XORI	I-Type	0	0	0	1	1	0	1	0	x
ORI	I-Type	0	0	0	1	1	0	1	0	x
ANDI	I-Type	0	0	0	1	1	0	1	0	x
SLLI	I-Type	0	0	0	1	1	0	1	1	x
SRLI	I-Type	0	0	0	1	1	0	1	1	x
SRAI	I-Type	0	0	0	1	1	0	1	1	x
SB	S-Type	0	1	x	0	1	0	x	2	x

SH	S-Type	0	2	x	0	1	0	x	2	x
SW	S-Type	0	3	x	0	1	0	x	2	x
SBU	S-Type	0	4	x	0	1	0	x	2	x
SHU	S-Type	0	5	x	0	1	0	x	2	x
ADD	R-Type	0	0	0	1	0	0	1	x	x
SUB	R-Type	0	0	0	1	0	0	1	x	x
SLL	R-Type	0	0	0	1	0	0	1	x	x
SLT	R-Type	0	0	0	1	0	0	1	x	x
SLTU	R-Type	0	0	0	1	0	0	1	x	x
XOR	R-Type	0	0	0	1	0	0	1	x	x
SRL	R-Type	0	0	0	1	0	0	1	x	x
SRA	R-Type	0	0	0	1	0	0	1	x	x
OR	R-Type	0	0	0	1	0	0	1	x	x
AND	R-Type	0	0	0	1	0	0	1	x	x
MUL	R-Type	0	0	0	1	0	0	1	x	x
MULH	R-Type	0	0	0	1	0	0	1	x	x
MULHSU	R-Type	0	0	0	1	0	0	1	x	x
MULHU	R-Type	0	0	0	1	0	0	1	x	x
DIV	R-Type	0	0	0	1	0	0	1	x	x
REM	R-Type	0	0	0	1	0	0	1	x	x
REMU	R-Type	0	0	0	1	0	0	1	x	x
LUI	U-type	0	0	0	1	1	0	1	3	x
AUIPC	U-type	0	0	2	1	x	0	1	3	x
JALR	I-Type	0	0	0	1	2	1	1	3	0
JAL	J-Type	0	0	0	1	2	1	1	3	1
BEQ	B-Type	0	0	2	0	0	1	x	2	0
BNE	B-Type	0	0	2	0	0	1	x	2	0
BLT	B-Type	0	0	2	0	0	1	x	2	0
BGE	B-Type	0	0	2	0	0	1	x	2	0
BLTU	B-Type	0	0	2	0	0	1	x	2	0
BGEU	B-Type	0	0	2	0	0	1	x	2	0

Table 2 : Instruction with control signals

https://docs.google.com/spreadsheets/d/1OVitoZ5JZzUCN-YxDwMBUPv7XC754ZeRF_KaZLkNERE/edit?usp=sharing

Follow this link for control signals with all ALU op codes.(Sheet 3)

- To process S type and J type instructions 2 new control signals are added(immediate value selector and PC selector).Since there are few types of immediate value range selections,a new Immediate value selector control signal is there and which will be processed in the Sign extender unit.
- Also there connection is made between ALU and the PC value,because of JAL and JALR instructions.Because they write PC value to the source register.
- Also there is another wire to connect the PC adder and ALU output to process branch instructions and U-type instructions.
- There are no time delays for the control unit signal generation.
- And also signals are generated asynchronously.

4. Programme Counter

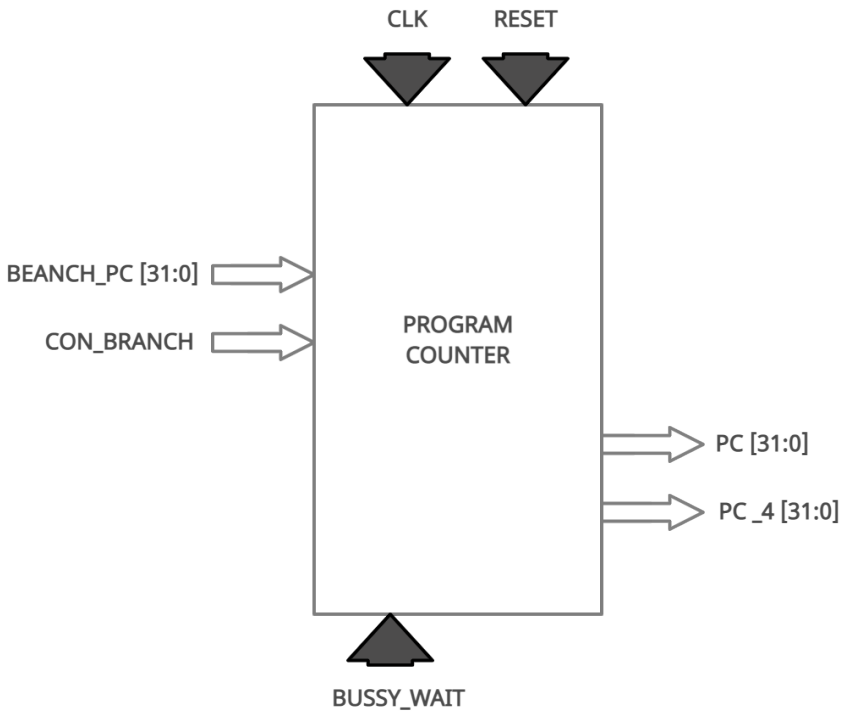

FIGURE 5 : PC unit design

Inputs:

BRANCH_PC -	control signal for branch instructions
Branch_pc -	Programme counter value when a branch is happening
CLK -	clock
RESET-	result signal

Outputs:

PC -	Programme counter value
PC -	programme counter value + 4

- signals are generated synchronously.
- If the BUSY WAIT and BRANCH control signals are zero,the positive edge at the clock signal PC value will be updated(Increased by 4) .
- If the BRACH control signal is 1,Branch will happen according to the BRANCH PC value.
- The PC_4 output value is used to predict the new PC value for branch instructions and u-type instructions.

Instruction memory hierarchy

- Instruction cache module is directly connected to the cpu while Instruction memory is connected to the cache.
- Cache module has 8 data arrays with 4 instructions wide.
- When There is a miss instruction cache, communicate with the instruction memory and read 4 instructions wide data.It will take about 10 clock cycles and in the meantime processor is stolen using the busy wait signals.Intermediate Busy signal is used to communicate with cache and the memory.

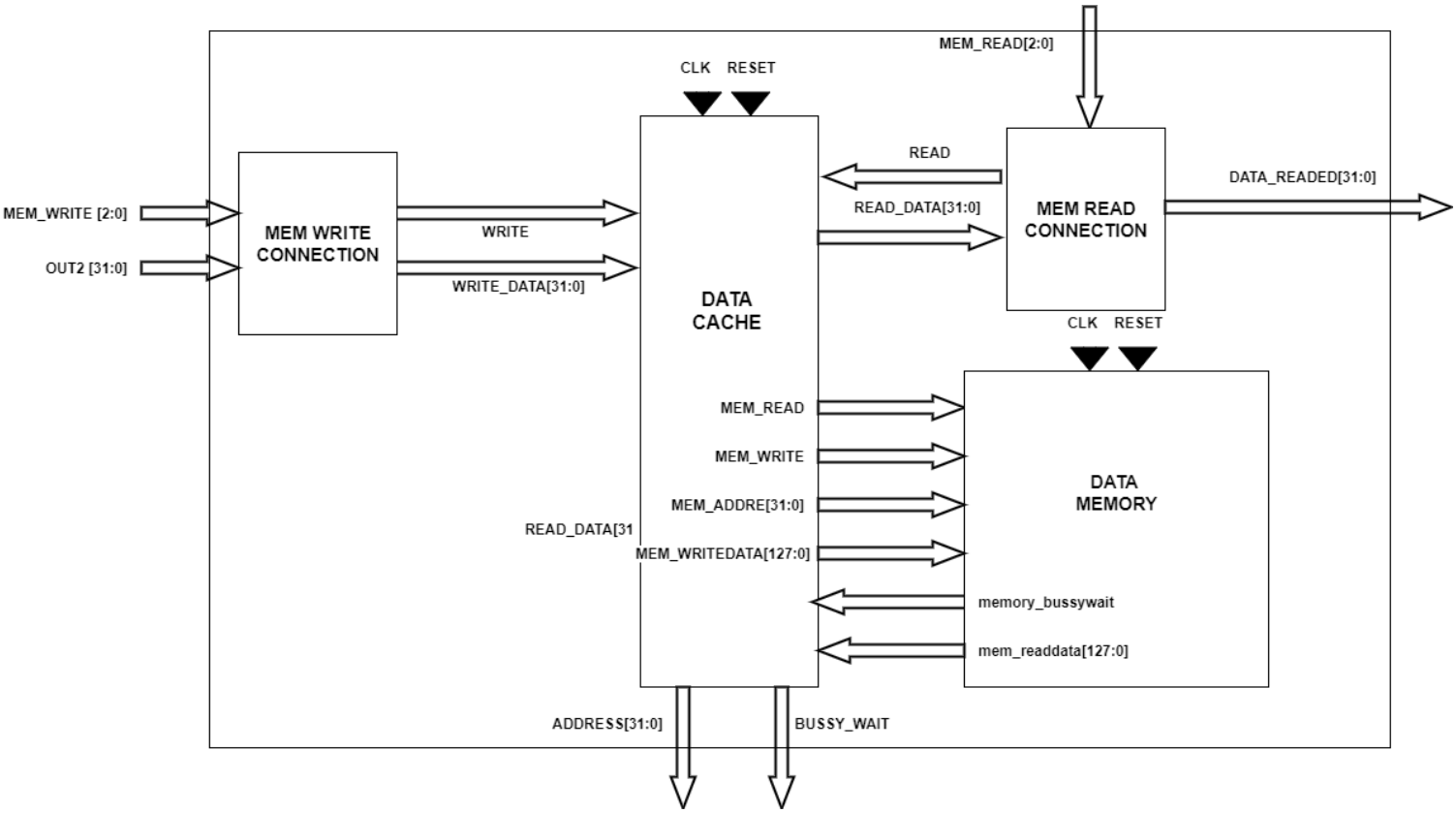

FIGURE 6 : Data Memory hierarchy

Data memory hierarchy

- Data cache module is directly connected to the cpu while Data memory is connected to the cache.
- Cache module has 8 data arrays 4 words wide.
- Write back policy is used in data memory hierarchy.
- Same as the instruction memory ,It will take about 10 clock cycles when there is a miss and in the meantime processor is stolen using the busy wait signals.Intermediate Busy signal is used to communicate with cache and the memory.

Read And Write Controller

- As figure 2 shows there are Two modules between the data cache and other modules.
- Those modules are there to handle special Memory read and write Instruction like LH,LB,SB,SH.

- According to the Opcode ,3 bit long control signals are sent to those modules and These modules will filter Byte or a half word from the data reading or Writing.
- And also those modules generate the write or read control signal for the data cache according to the 3 bit control signal coming from the control unit.

Pipeline Registers

There are four pipeline registers to connect the 5 stages of the RISC-V processor. The for pipeline registers are,

- Pipeline 1 (IF-ID Pipeline)
- Pipeline 2 (ID-EX Pipeline)
- Pipeline 3 (EX-MEM pipeline)
- Pipeline 4 (MEM-WB pipeline)

All the pipeline sends data to next stages with time units delay according to the need.

1. **Pipeline1 (IF_ID_Pipeline) Register**

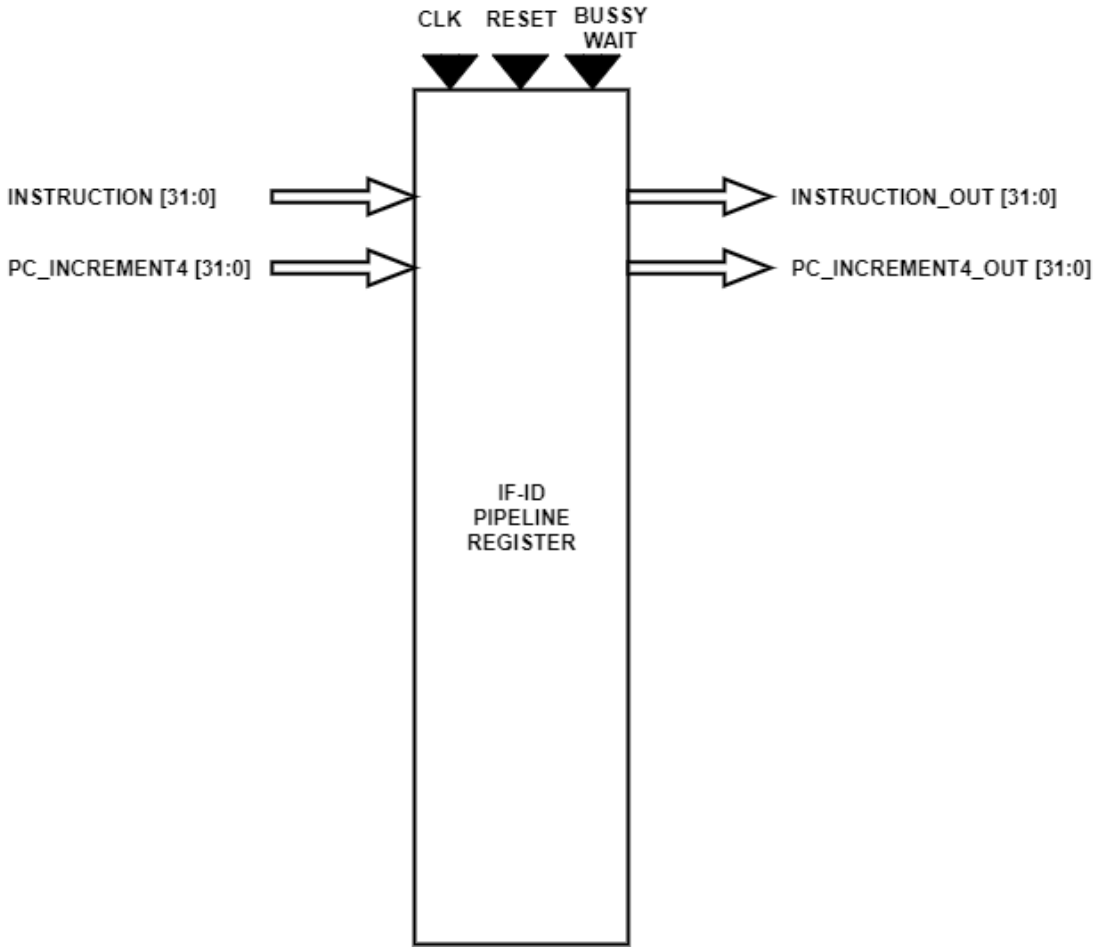

FIGURE 7: IF-ID Pipeline Register

Inputs: CLK
RESET
BUSSY_WAIT
INSTRUCTION [31:0]
PC_INCREMENT4 [31:0]

Outputs:
INSTRUCTION_OUT [31:0]
PC_INCREMENT4_OUT [31:0]

Time Delays:
After positive clock edge 1 time unit delay added to set INSTRUCTION_OUT

- This pipeline register is located in between Instruction Fetching and Instruction decoding stages.

- The BUSSY_WAIT signal is to decide whether the data need to be set for the next stage or wait. Because when Instruction memory or Data memory bussy need to hold the data on the pipeline to freeze the system.
- When the RESET signal is 1 set INSTRUCTION_OUT to 0 and PC_INCREMENT4_OUT to -4 to reset the processor.

2. Pipeline2 (ID_EX_Pipeline) Register

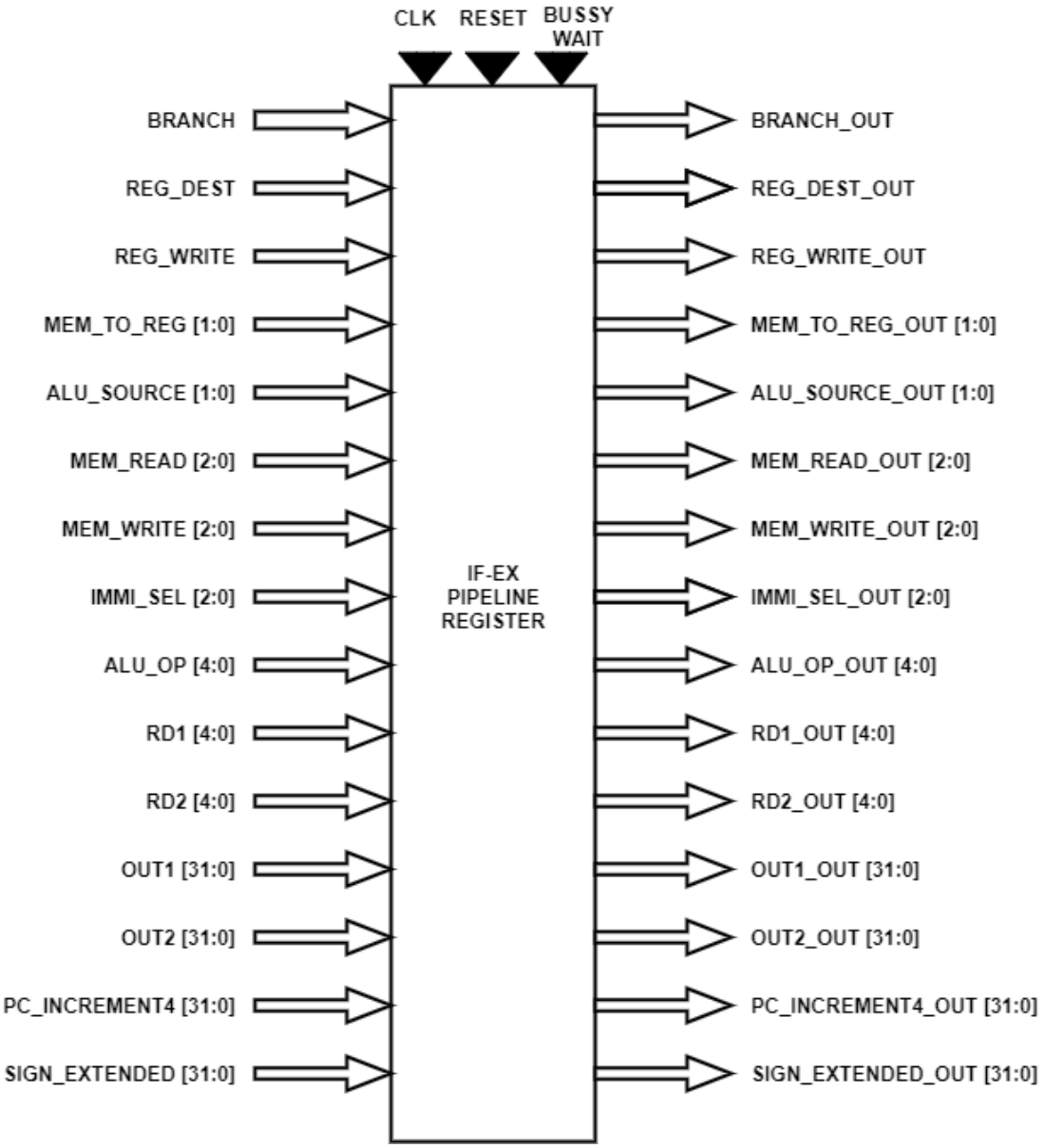

FIGURE 8: ID-EX Pipeline Register

Inputs: CLK
RESET
BUSSY_WAIT
BRANCH

REG_DEST
REG_WRITE
PC_SEL
MEM_TO_REG [1:0]
ALU_SOURCE [1:0]
MEM_READ [2:0]
MEM_WRITE [2:0]
IMMI_SEL [2:0]
ALU_OP [4:0]
RD1 [4:0] - Register Destination 1
RD2 [4:0] - Register Destination 2
OUT1 [31:0] - Data 1 from register file
OUT2 [31:0] - Data 2 from register file
PC_INCREMENT4 [31:0]
SIGN_EXTENDED [31:0]

Outputs:

BRANCH_OUT
REG_DEST_OUT
REG_WRITE_OUT
PC_SEL_OUT
MEM_TO_REG_OUT [1:0]
ALU_SOURCE_OUT [1:0]
MEM_READ_OUT [2:0]
MEM_WRITE_OUT [2:0]
IMMI_SEL_OUT [2:0]
ALU_OP_OUT [4:0]
RD1_OUT [4:0] - Register Destination 1
RD2_OUT [4:0] - Register Destination 2
OUT1_OUT [31:0] - Data 1 from register file
OUT2_OUT [31:0] - Data 2 from register file
PC_INCREMENT4_OUT [31:0]
SIGN_EXTENDED_OUT [31:0]

Time Delay:

Add 3 time units delays after the positive clock edge for setting input values to ***MEM_TO_REG*** & ***REG_WRITE*** output registers output registers. Because that data flows directly through the next two pipelines (EX-MEM pipeline & MEM-WB pipeline). To prevent overwriting data in two pipelines before sending data to the next stage added this delay.

Add 2 time units delays after the positive clock edge for setting input values to ***BRANCH_OUT***, ***MEM_READ_OUT*** & ***MEM_WRITE*** output registers output registers. Because that data goes directly to the next pipeline(EX-MEM Pipeline) without going through any modules. Therefore add additional time unit delay to prevent overwriting data before sending to the next stage in the EX-MEM pipeline.

Add 1 time unit delays for setting all other output registers values.

- This pipeline register is located in between the Instruction decoding process and ALU.
- The BUSSY_WAIT signal is to decide whether the data need to be set for the next stage or wait. Because when Instruction memory or Data memory bussy need to hold the data on the pipeline to freeze the system.
- When the RESET signal is 1 set PC_INCREMENT4_OUT to -4 and all other output register values to 0 to reset the processor.

3. Pipeline3 (EX_MEM_Pipeline) Register

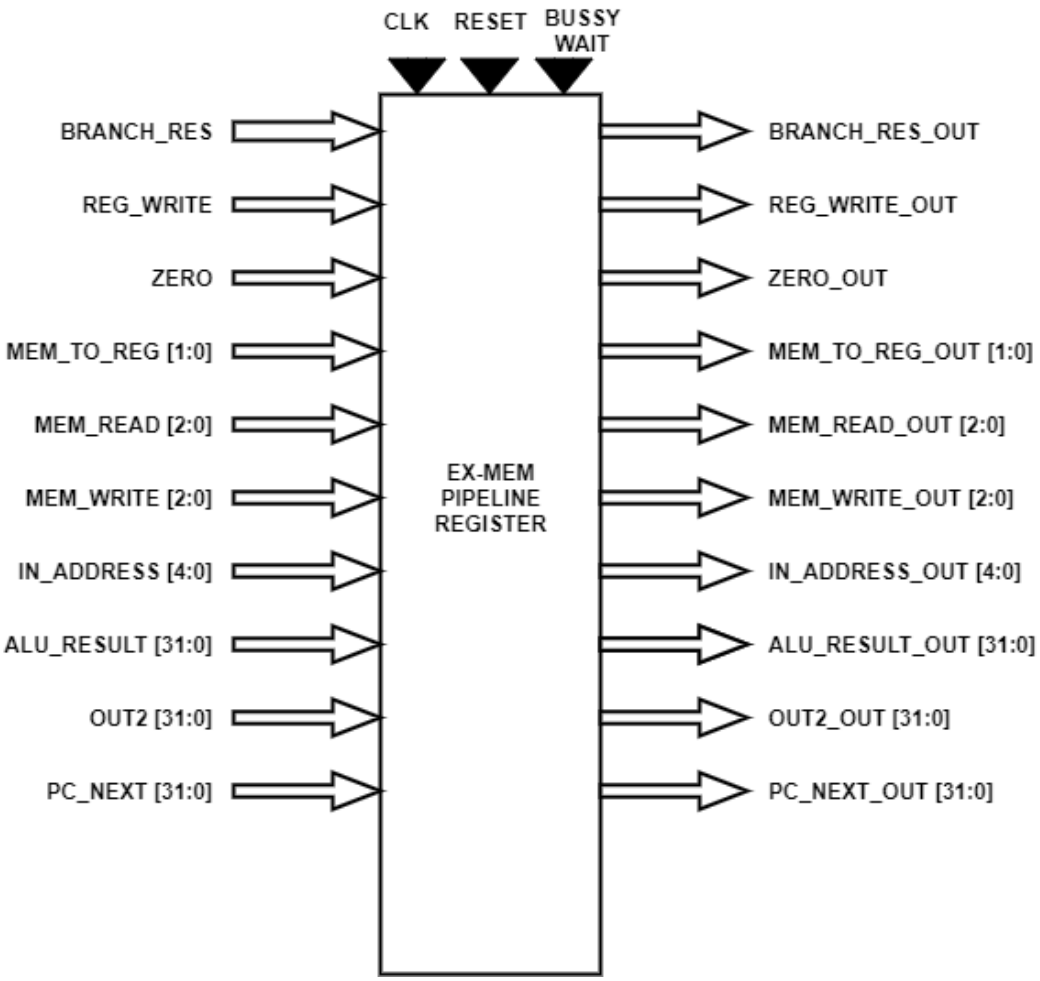

FIGURE 9: EX-MEM Pipeline Register

Inputs: CLK
RESET
BUSSY_WAIT
BRANCH_RES - Alu result for the branch
REG_DEST
REG_WRITE
MEM_TO_REG [1:0]
MEM_READ [2:0]
MEM_WRITE [2:0]
IN_ADDRESS [4:0] - address where the data need to write on register file
ALU_RESULT [31:0]
OUT2 [31:0] -
PC_NEXT [31:0]

Outputs:
BRANCH_RES_OUT - Alu result for the branch

REG_DEST_OUT
REG_WRITE_OUT
MEM_TO_REG_OUT [1:0]
MEM_READ_OUT [2:0]
MEM_WRITE_OUT [2:0]
IN_ADDRESS_OUT [4:0] - address where the data need to write on register file
ALU_RESULT_OUT [31:0]
OUT2_OUT [31:0]

Time Delays:

Add 2 time units delays after the positive clock edge for setting input values to ***REG_WRITE_OUT***, ***MEM_TO_REG_OUT, PC_NEXT_OUT*** & ***IN_ADDRESS_OUT*** output registers output registers. Because that data goes directly to the next pipeline(MEM_WB Pipeline) without going through any modules. Therefore add additional time unit delay to prevent overwriting data before sending to the next stage in the MEM-WB pipeline.

Add 1 time unit delays for setting all other output registers values.

- This pipeline register is located in between the ALU and Memory.
- The BUSSY_WAIT signal is to decide whether the data need to be set for the next stage or wait. Because when Instruction memory or Data memory bussy need to hold the data on the pipeline to freeze the system.
- When the RESET signal is 1 set all the output register values to 0 to reset the processor.

4. **Pipeline4 (MEM-WB_Pipeline) Register**

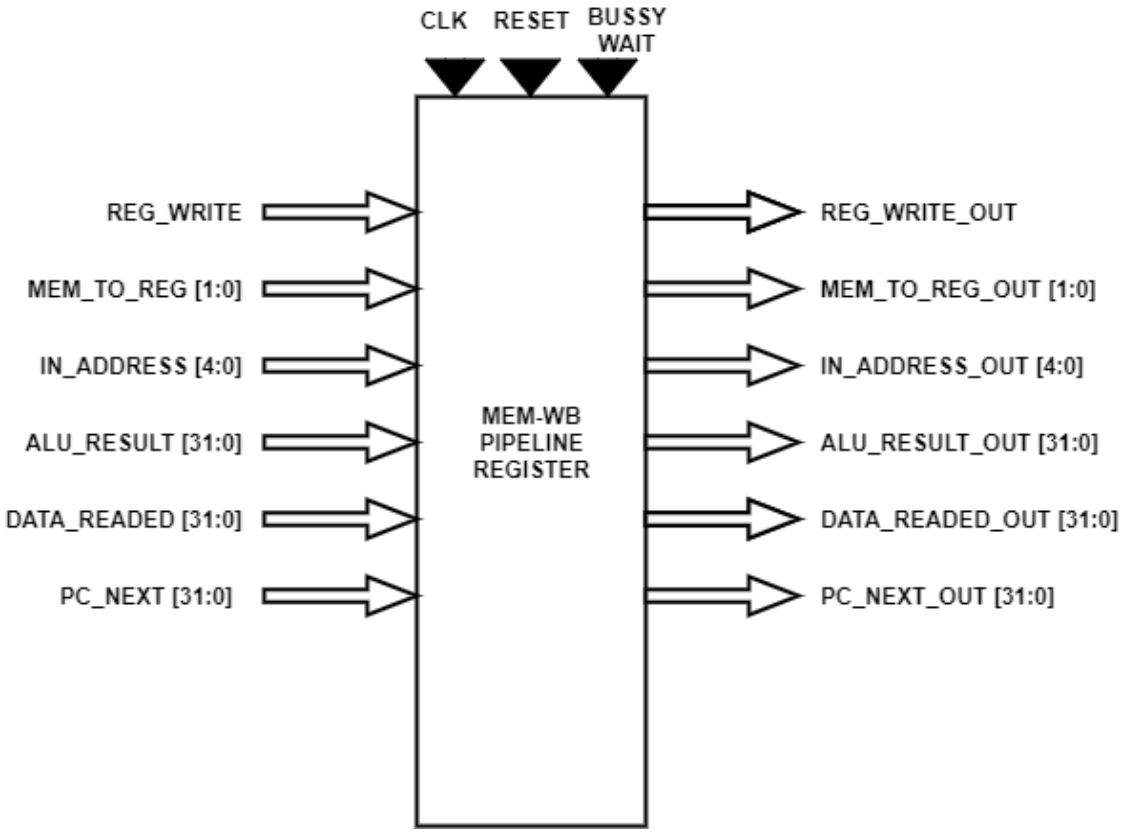

FIGURE 10: EX-MEM Pipeline Register

Inputs: CLK
RESET
BUSSY_WAIT
REG_WRITE
MEM_TO_REG [1:0]
IN_ADDRESS [4:0] - address where the data need to write on register file
ALU_RESULT [31:0]
DATA_READED [31:0]
PC_NEXT [31:0]

Outputs:
REG_WRITE_OUT
MEM_TO_REG_OUT [1:0]
IN_ADDRESS_OUT [4:0] - address where the data need to write on register file
ALU_RESULT_OUT [31:0]
PC_NEXT_OUT [31:0]
DATA_READED [31:0]

Time Delays:

Add 1 time unit delay for setting all output registers values.

- This pipeline register is located in between the MEMORY stage and WRITEBACK stages.
- The BUSSY_WAIT signal is to decide whether the data need to be set for the next stage or wait. Because when Instruction memory or Data memory bussy need to hold the data on the pipeline to freeze the system.
- When the RESET signal is 1 set all the output register values to 0 to reset the processor.

Forwarding Unit

The forwarding unit was implemented to control the data hazard which occured when reading register values. To control this forward values from MEMORY and WRITEBACK stages if there is any data hazard

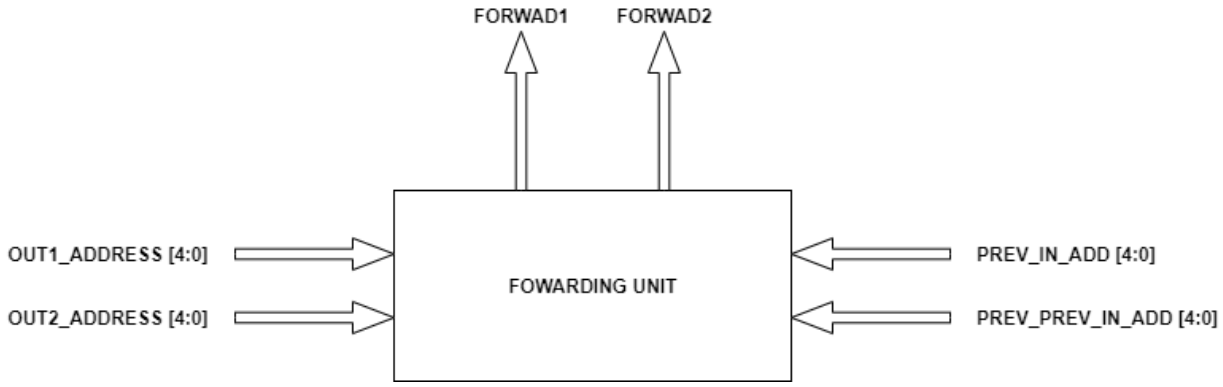

FIGURE 11 - Diagram of Forwarding Unit

INPUTS: OUT1_ADDRESS [4:0]- Read register address 1 of current instruction on EX stage
OUT2_ADDRESS [4:0]- Read register address 2 of current instruction on EX stage
PREV_IN_ADD[4:0] - Write register address of instruction in Memory stage
PREV_PREV_IN_ADD[4:0]-Write register address of instruction in WB stage

OUTPUTS: FORWAD1[1:0] - Select signal for ALU mux1
FORWAD2[1:0] - Select signal for ALU mux2

In this unit compare the OUT1_ADDRESS & OUT2_ADDRESS with PREV_IN_ADD & PREV_PREV_IN_ADD separately and generate the FORWARD signals. Thos signal generation done according to this condition below,

Condition	FORWAD1
OUT1_ADDRESS not equal both	0
OUT1_ADDRESS equal to PREV_IN_ADD	1
OU1_ADDRESS not equal PREV_IN_ADD and equal to PRE_PREV_IN_ADD	2

Condition	FORWAD2
OUT2_ADDRESS not equal both	0
OUT2_ADDRESS equal to PREV_IN_ADD	1
OU2_ADDRESS not equal PREV_IN_ADD and equal to PRE_PREV_IN_ADD	2

This is how the forwarding unit handles data hazar by bypassing the values.

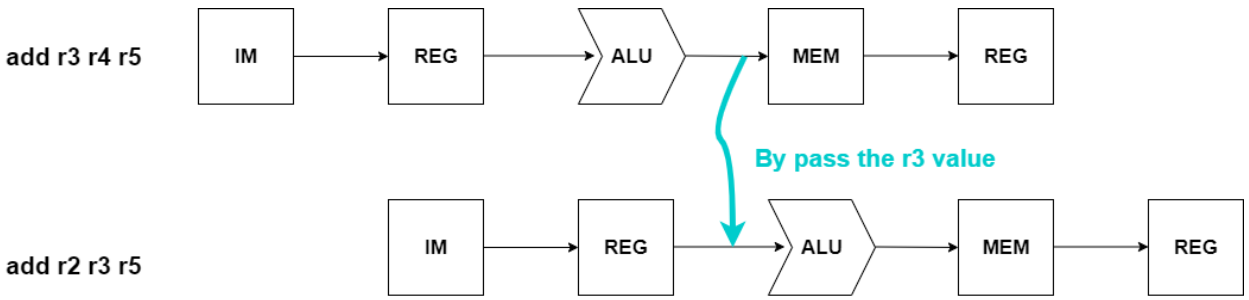

FIGURE 12 : Bypassing data from previous instruction

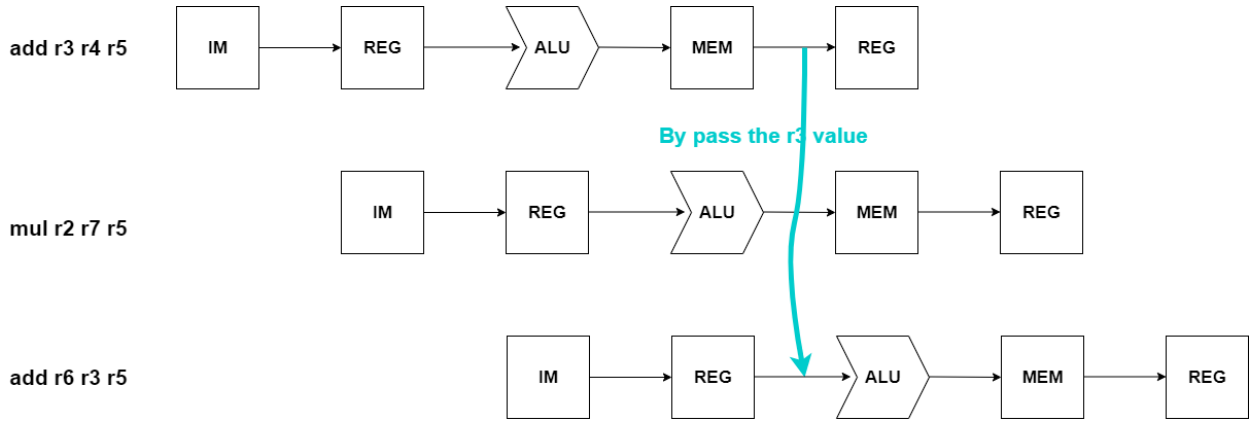

FIGURE 13 : Bypassing data from previous instruction

ALU Mux1

This module is implemented to choose the relevant DATA1 according to the FOWARD1 value.

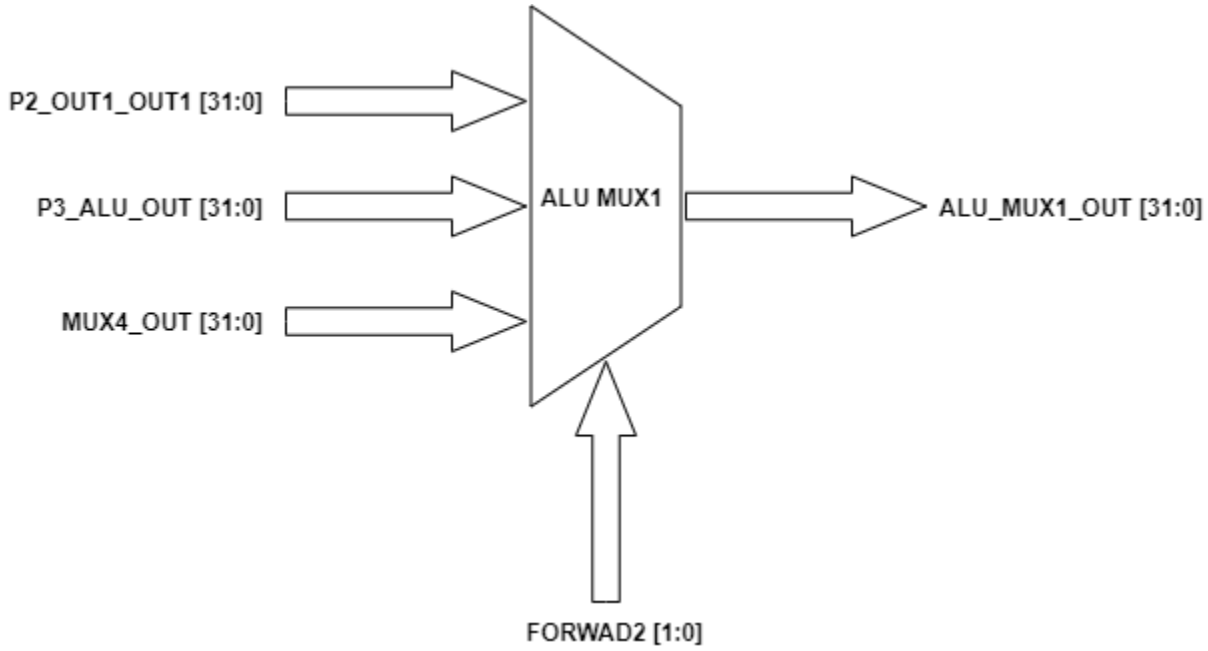

FIGURE 14 : ALU MUX1 Diagram

INPUTS :

P2_OUT1_OUT - Data read from the register

P3_ALU_OUT - ALU result value from the previous instruction

MUX4_OUT - Register Write data from 2nd Previous instruction.

FORWAD1

OUTPUT:

ALU_MUX1_OUT - Data value 1 to the ALU

Addea one time unite delay in the mosule

ALU Mux2

This module is implemented to choose the relevant DATA2 according to the FOWARD1 value.

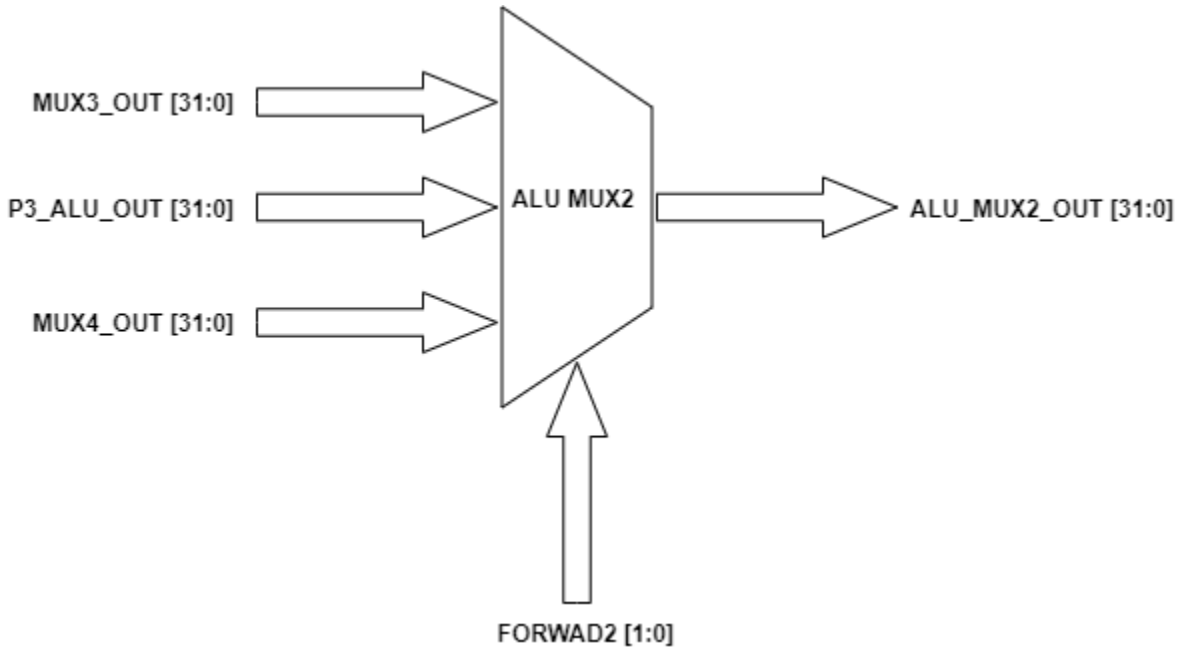

FIGURE 15 : ALU MUX1 Diagram

INPUTS :

MUX3_OUT - Data 2 value from register or immediate value

P3_ALU_OUT - ALU result value from the previous instruction

MUX4_OUT - Register Write data from 2nd Previous instruction.

FORWAD1

OUTPUT:

ALU_MUX2_OUT - Data value 2 to the ALU

Addea one time unite delay in the mosule

Timing

Stage	Process	Timing
Stage1: Instruction Fetch	1.PC update	3

	2.Instruction tag comparison	1
	3.Output the the instructions	1
Stage 2: Instruction Decode	1.Output pipeline data	1
	2.Generating control signals	1
	3.Data writes to register	3
	4.Data Reads from register	5
	5.Sign extender	1
Stage3 : Instruction Execut	1.Output pipeline data	1
	2.Mux 3	1
	3.Mux 2	1
	4.Mux 1	1
	5.ALU Mux1	1
	6.ALU Mux1	1
	7.ALU	2
	8.Adder	2
Stage 4:Memory Access	1.Output pipeline data	1
	2.And gate	1
	3.Write controller	1
	4.read Controller	1
	5.Data Memory Write	2
	6.Data Memory Read	5
Stage5: Write Back	1.Output pipeline data	1
	2.Mux 4	1

	3.Write back	3

Table 3: Timing delays

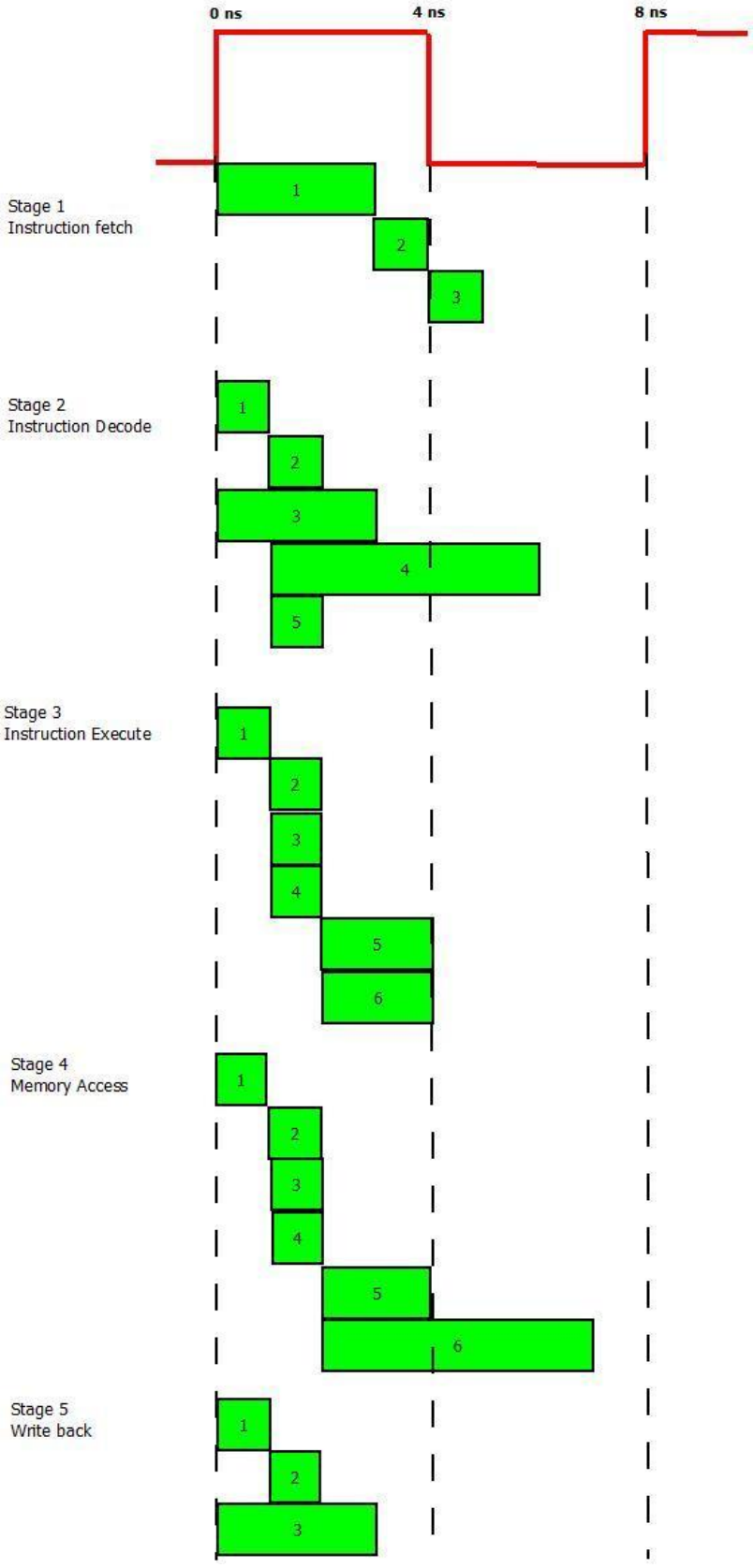

FIGURE 16: Timing diagram

- Pipeline data forwarding,register file writing and PC updating happen synchronously to the clock while Other modules do not.
- PC updating happens after 3ns delay after receiving the Branch signal which takes 2ns.
- Since the register write back happen in synchronously,for data writing back there is 3ns delay.Because of writing data will be available at the register file only after 2ns(pipeline delay + MUX 4 delay)
- Pipeline data forwarding happen at the rising clock edge after 1ns of delay.But except for the pipeline signals which are not used in the current stage.As a example in 3rd stage (Instruction Execute stage) AluOp code,RegDef and AluSource signals are forwarding after 1ns and Other siglas like BranchF takes 2ns to forward.Those 2ns delays are there Because of the Other wise Those type of signals can be forwarded to next stage also at the same clock cycle.
- When register reading ,it waits 5 ns from address reserving because the reading should be done after the register writing completes.
- Same as the register file reading also in the memory reading waits 5ns,because it gives time to update the memory.

Changes made to other Unit after phase 3

1.Pipeline Register 2

Pipeline Register 2 change by passing OUT1_ADDRESS & OUT2_ADDRESS to the pipeline register. Because I need those two in the forwarding unit.

2. Assembler

Remove the stall made to handle the data hazard because data hazard handling in the forwarding unit.

3. ALU

Additional 1 time unit delay was added for all instructions because ALU Mux modules were added before the ALU and those ALU Mux modules had 1 time unit delay.

4.Delay

New module Delay implemented to give addition 1 time unit delay when Prev_IN_ADD pass through the pipeline 3 to 4