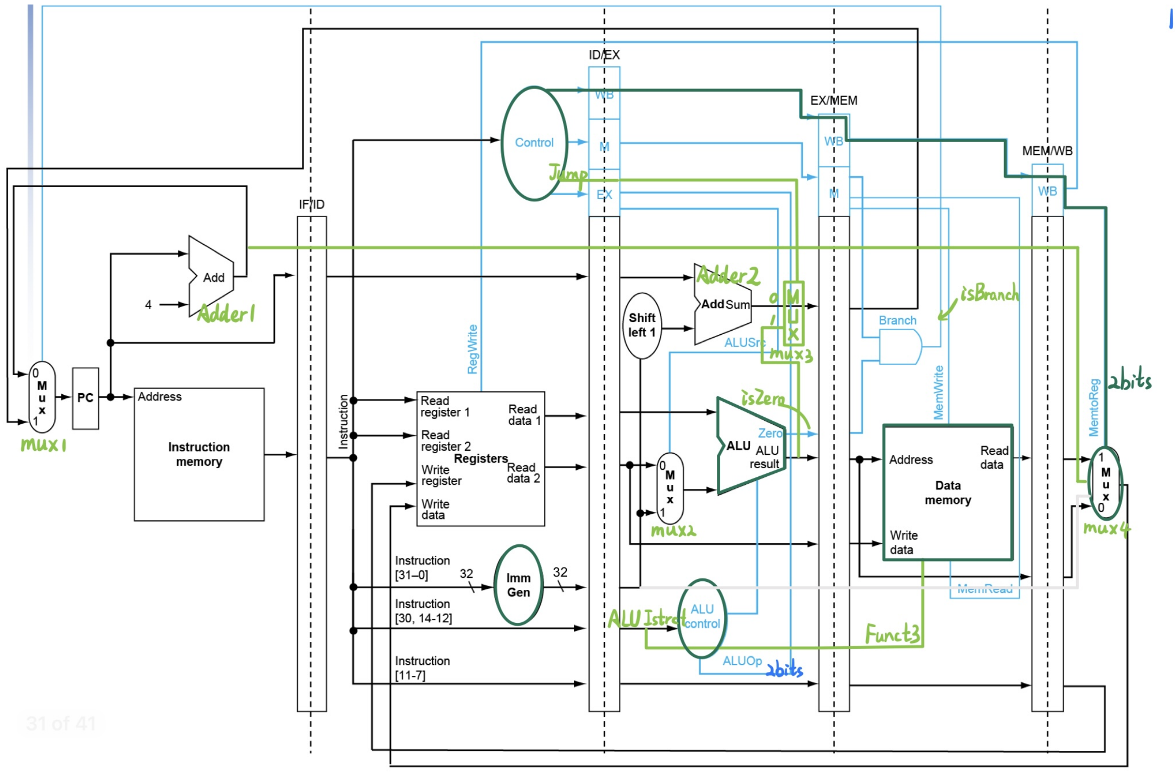
VE370 Lab 4 Report

Group 4 Wenqi Cao, Jingye Lin, Lan Wang

Modeling and Implementation

We modified the graph of pipelined processor in the lecture slide and changed the design of ALU control and ALU to support the instructions listed in the manuel. The new design is shown below:



The new design of the coding in ALU control is shown in the following table:

instruction type	ALU op (input)	`{instr[30],instr[14:12]}` (input)	ALU sel (output)
R-type (`add`)	10	4'b0000	4'b0010
R-type (others)	10	`{instr[30],instr[14:12]}`	`{instr[30],instr[14:12]}`
I-type (`addi`)	11	4'bX000	4'b0010
I-type (others; exclude load)	11	`{instr[30],instr[14:12]}`	`{1'b0,instr[14:12]}`
B-type (`bge`)	01	4'bX101	4'b1110
B-type (others)	01	`{instr[30],instr[14:12]}`	`{1'b1,instr[14:12]}`
load and jump	00	`{instr[30],instr[14:12]}`	4'b0010

The new design of ALU sel and the corresponding ALU operation is shown in the following table:

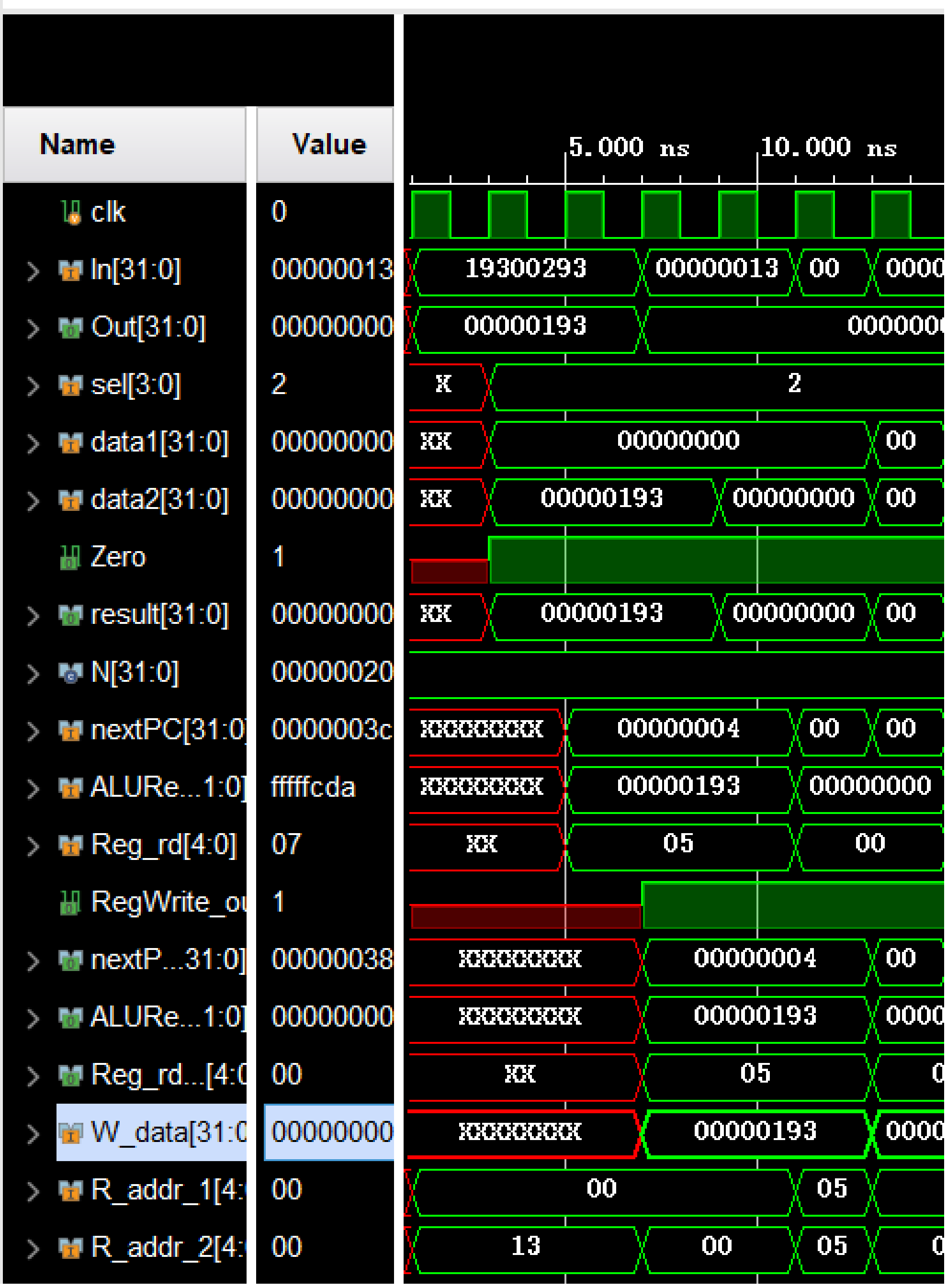
Operation	or	and	sll	srl	sra
ALU sel	4'b0110	4'b0111	4'b0001	4'b0101	4'b1101
ALU output	data1|data2	data1&data2	data1<<data2	data1>>data2 (use 0)	data1>>data2 (use sign bit)
zero	1'b0	1'b0	1'b0	1'b0	1'b0

Operation	add	sub	sub_neq	sub_blt	sub_bge
ALU sel	4'b0010	4'b1000	4'b1001	4'b1100	4'b1110
ALU output	data1+data2	data1-data2	data1-data2	data1-data2	data1-data2
zero	1'b1	`(output==0)?1'b0:1'b1`	`(output==0)?1'b0:1'b1`	`(output<0)?(~sel[1]):(sel[1])`	`(output<0)?(~sel[1]):(sel[1])`

Simulation Result

`addi t0 x0 0x193`

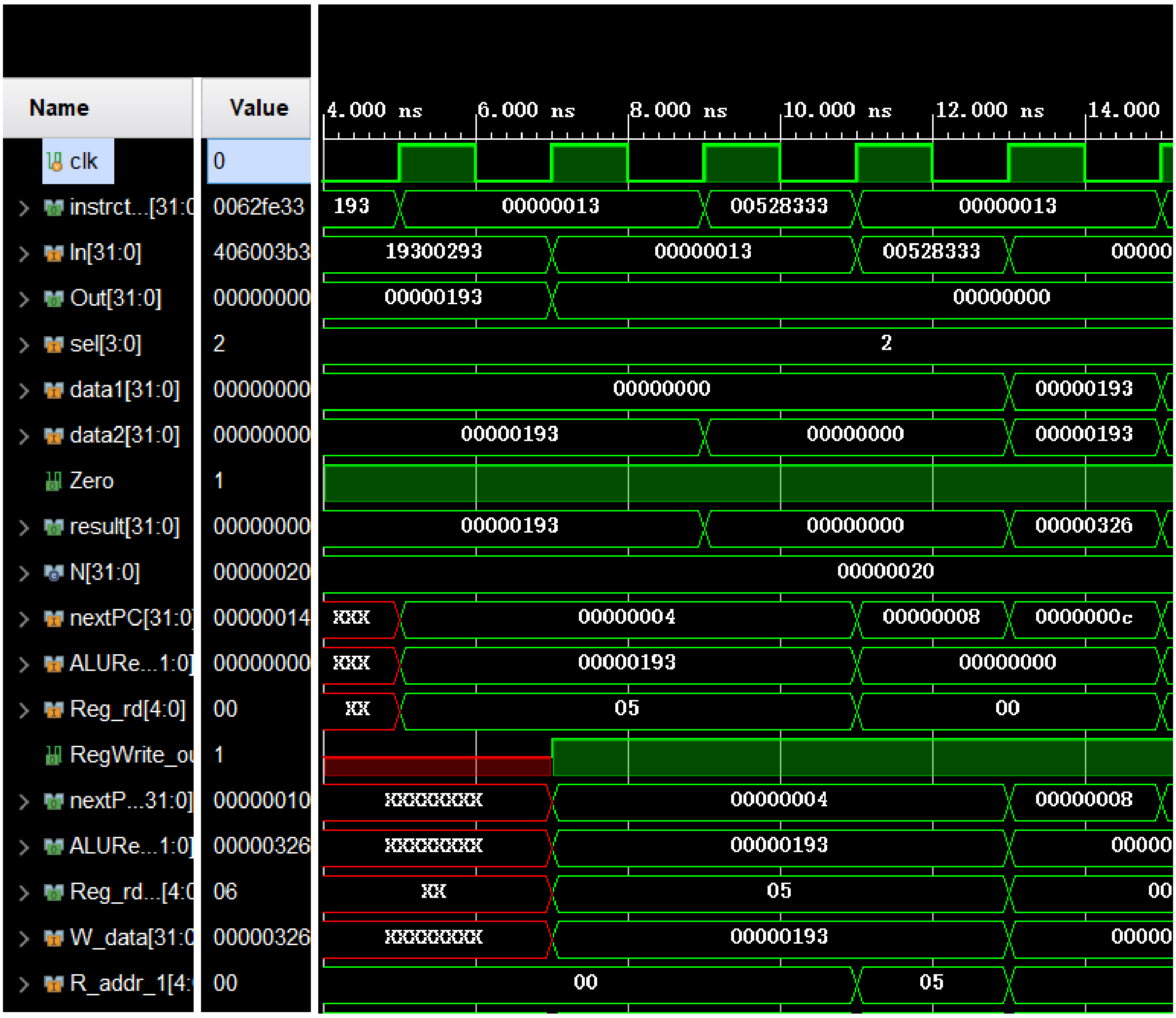
get `t0=0x193` and write back to register file



Name	Value
clk	0
In[31:0]	00000013
Out[31:0]	00000000
sel[3:0]	2
data1[31:0]	00000000
data2[31:0]	00000000
Zero	1
result[31:0]	00000000
N[31:0]	00000020
nextPC[31:0]	0000003c
ALURe...1:0]	fffffcda
Reg_rd[4:0]	07
RegWrite_ou	1
nextP...31:0]	00000038
ALURe...1:0]	00000000
Reg_rd...[4:0	00
W_data[31:0	00000000
R_addr_1[4:	00
R_addr_2[4:	00

`add t1 t0 t0`

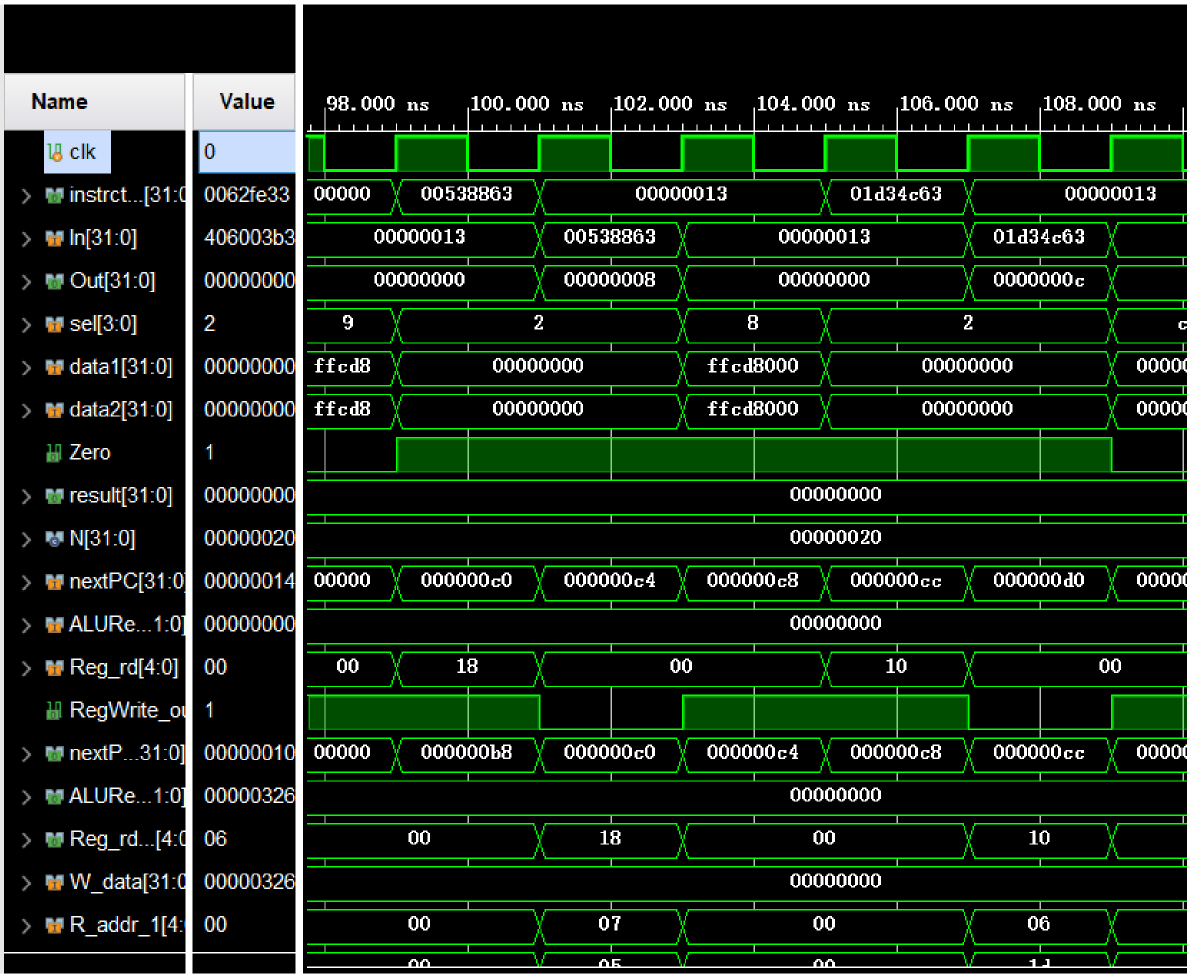
get `t1=t0+t0` and write back to register file



`beq t2 t0 right_branch_3`

`t2=t0` so this instruction should branch.

jump from instruction `0x00538865` to `0x01d34c63`

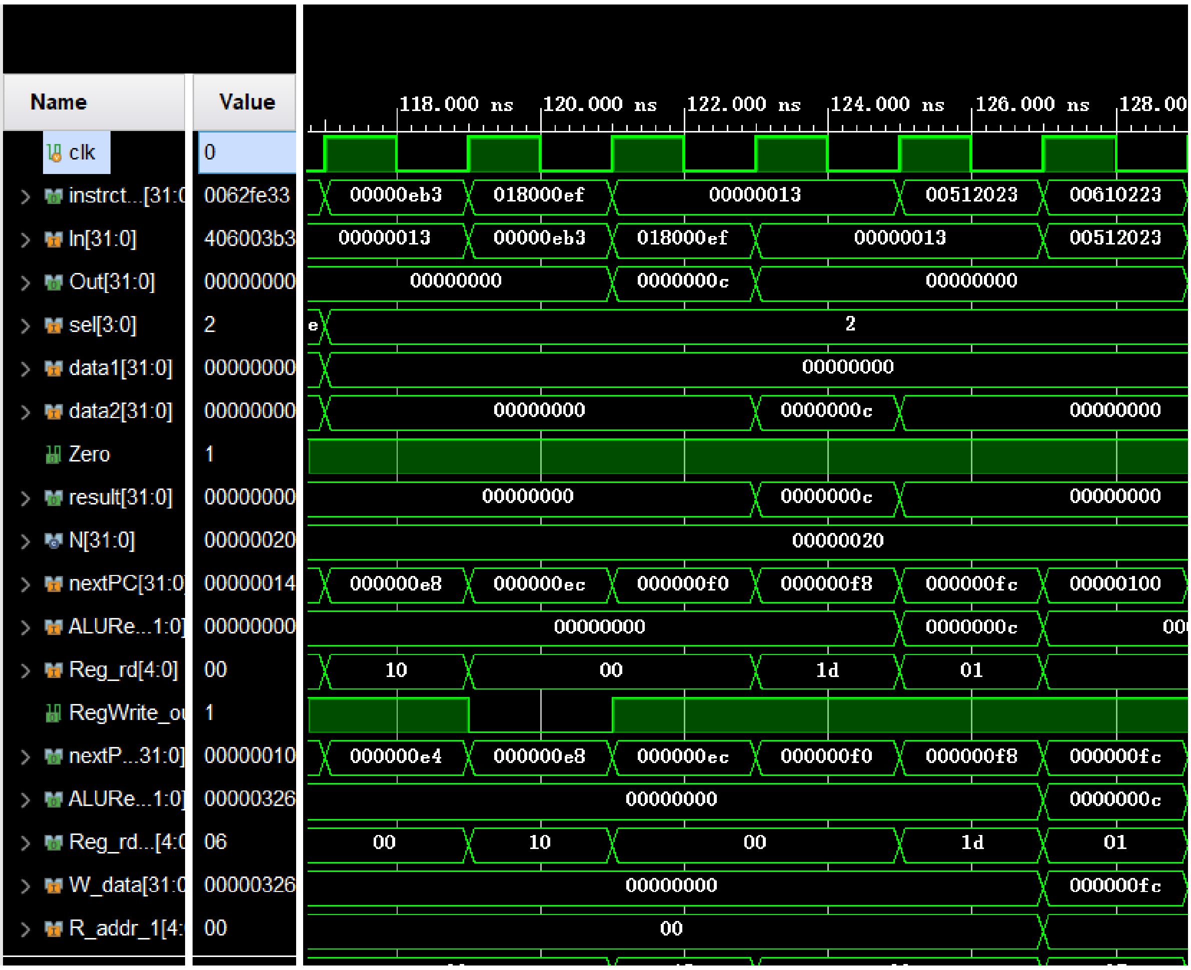


Name	Value
clk	0
instrct...[31:0	0062fe33
In[31:0]	406003b3
Out[31:0]	00000000
sel[3:0]	2
data1[31:0]	00000000
data2[31:0]	00000000
Zero	1
result[31:0]	00000000
N[31:0]	00000020
nextPC[31:0]	00000014
ALURe...1:0]	00000000
Reg_rd[4:0]	00
RegWrite_ou	1
nextP...31:0]	00000010
ALURe...1:0]	00000326
Reg_rd...[4:0	06
W_data[31:0	00000326
R_addr_1[4:	00

98.000 ns
100.000 ns
102.000 ns
104.000 ns
106.000 ns
108.000 ns
00000 00538863 00000013 01d34c63 00000013
00000013 00538863 00000013 01d34c63
00000000 00000008 00000000 0000000c
9 2 8 2 c
ffcd8 00000000 ffcd8000 00000000 0000
ffcd8 00000000 ffcd8000 00000000 0000
00000000
00000020
00000 000000c0 000000c4 000000c8 000000cc 000000d0 0000
00000000
00 18 00 10 00
00000 000000b8 000000c0 000000c4 000000c8 000000cc 0000
00000000
00 18 00 10
00000000
00 07 00 06
00 05 00 1d

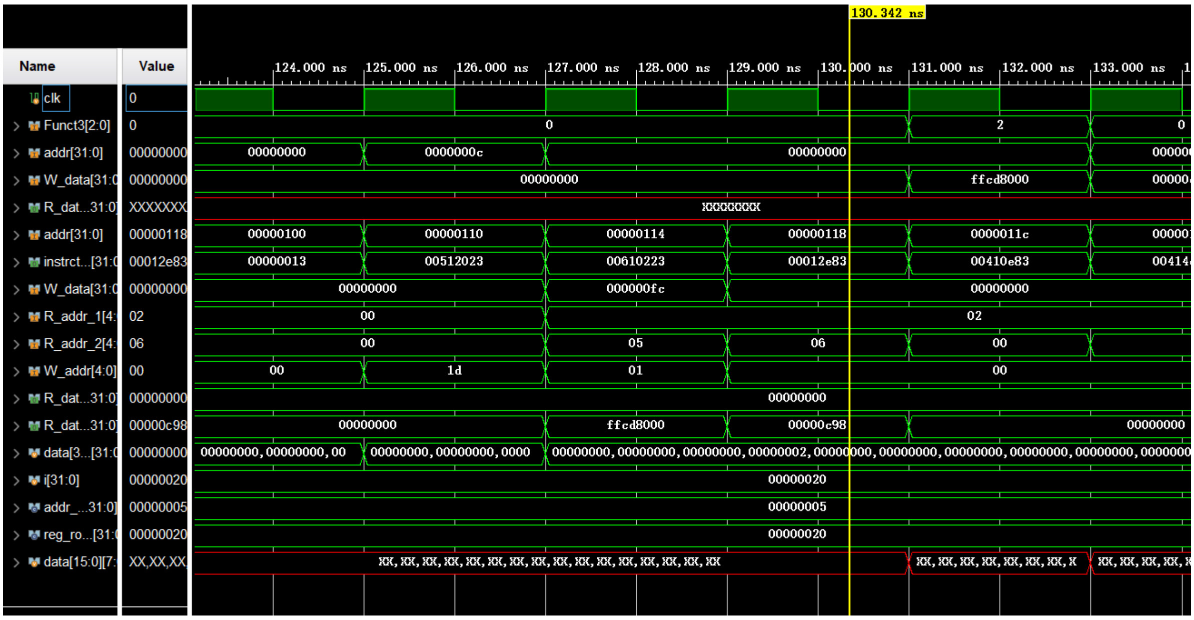
```
jal x1 memory_test # ra=0x000000bc
```

jump from instruction `0x018000ef` to `0x00512023`



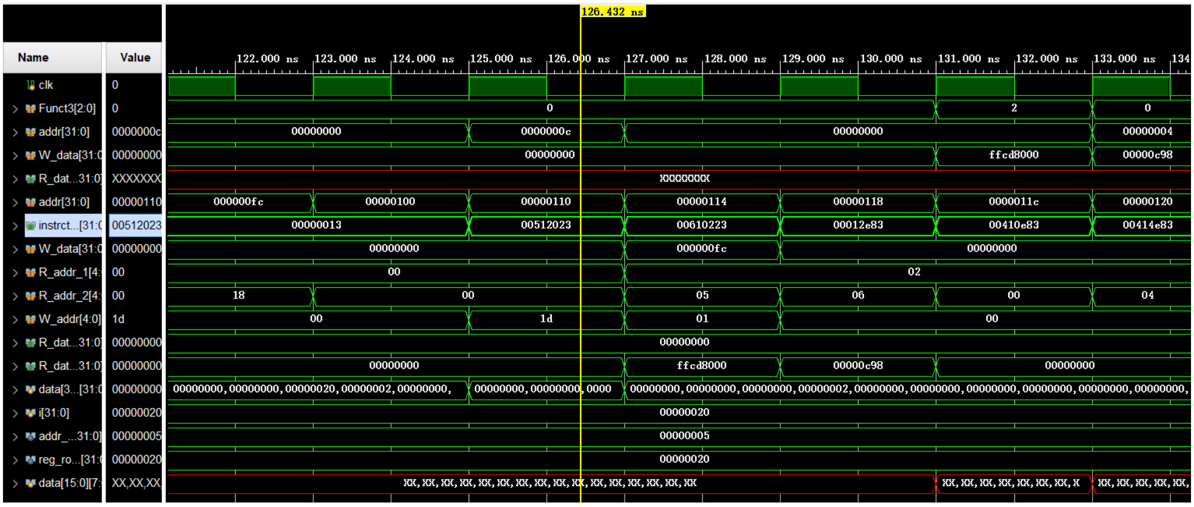
`lw t4 0(sp)`

load content of `x29` to `x2`



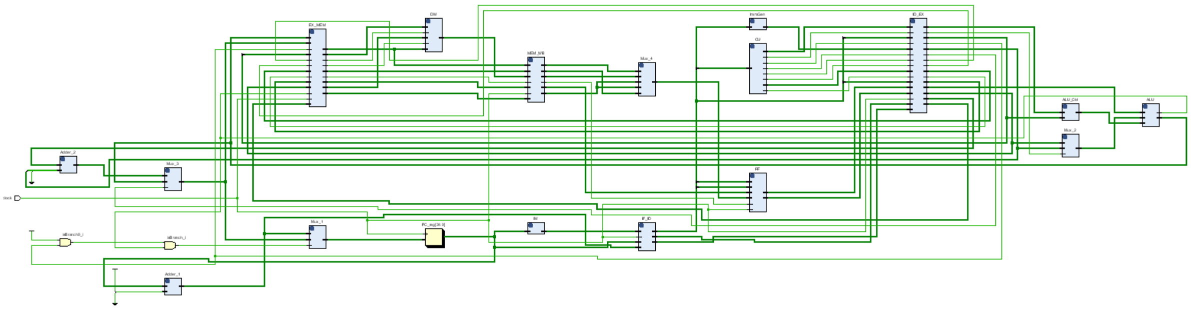
`sw t0 0(sp)`

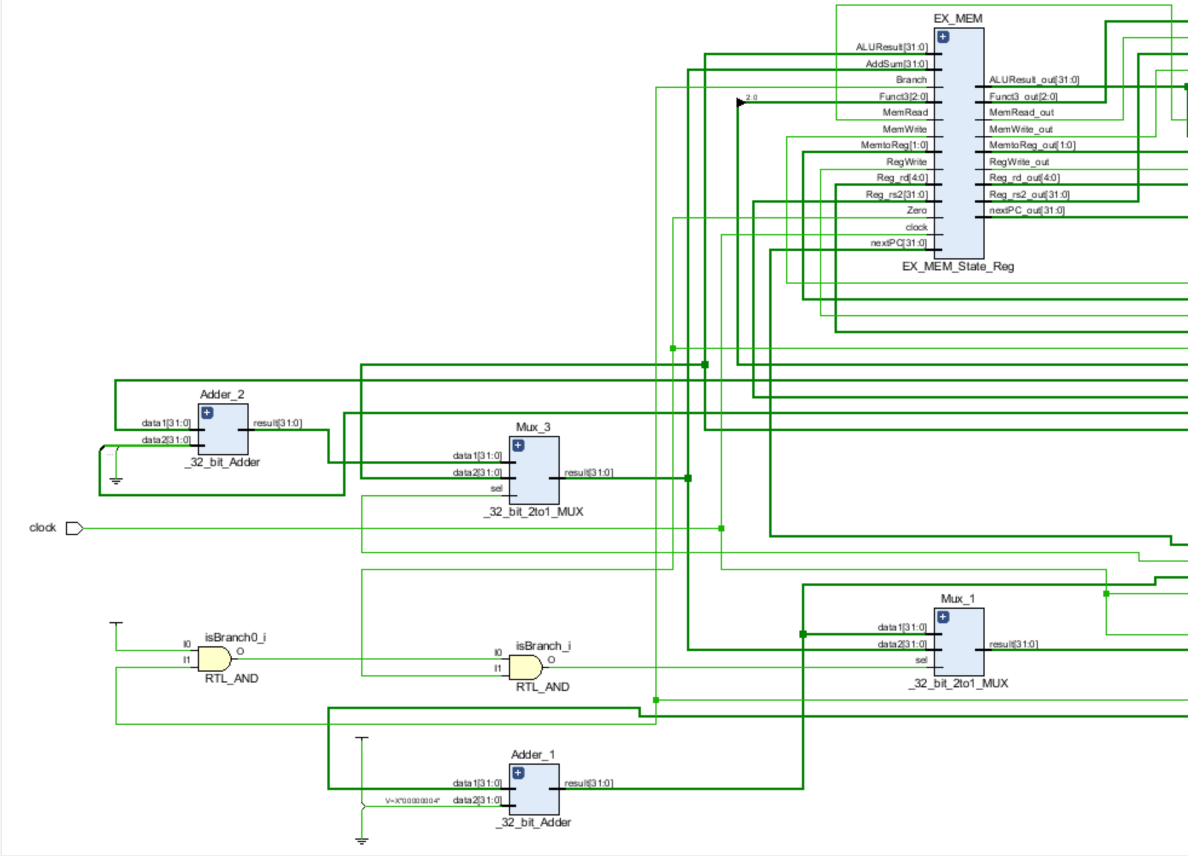
store content of `x2` to `x5`

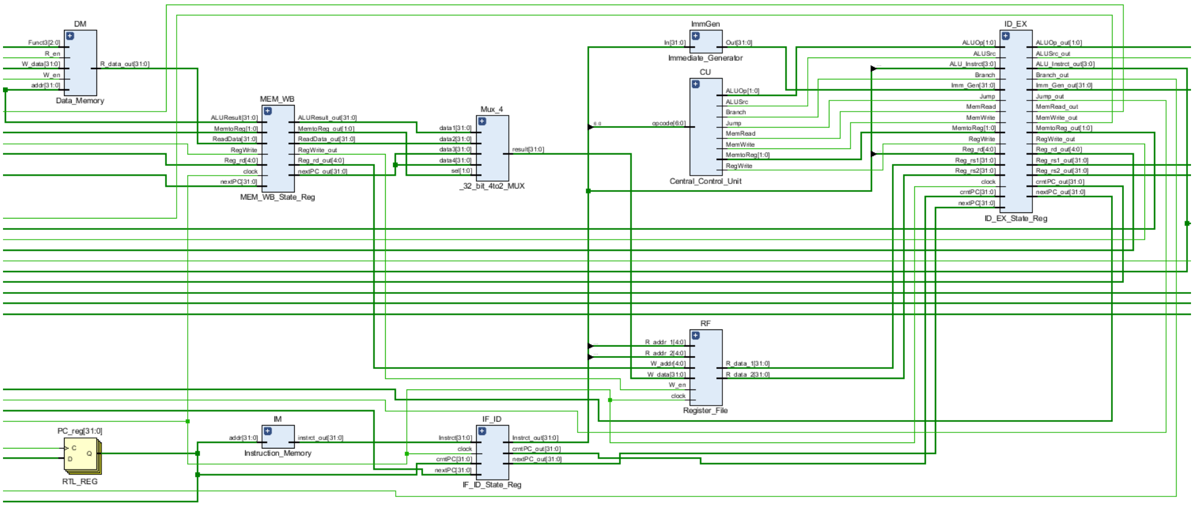


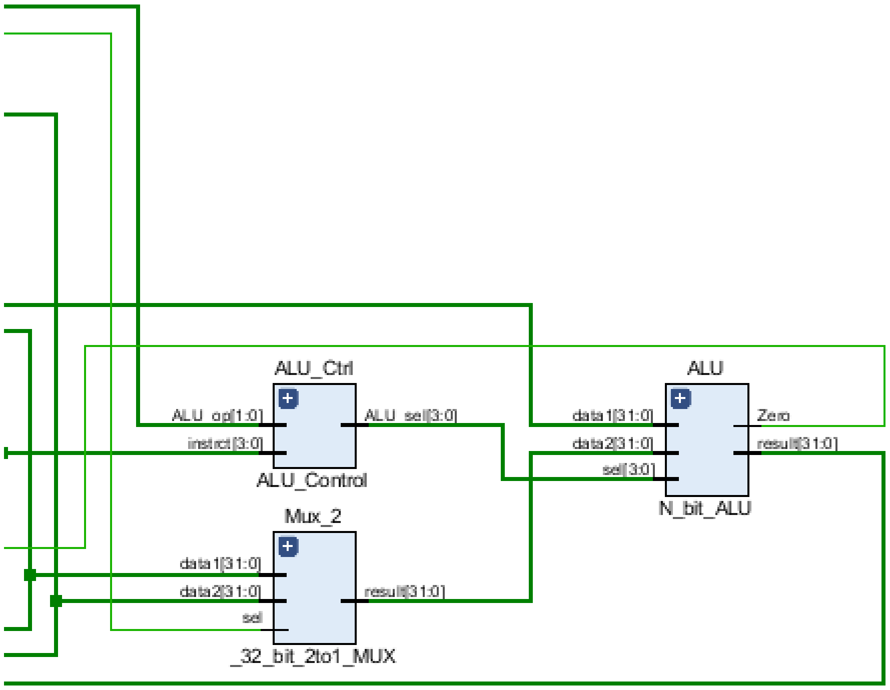
Name	Value
clk	0
Funct3[2:0]	0
addr[31:0]	0000000c
W_data[31:(	00000000
R_dat...31:0]	XXXXXXX
addr[31:0]	00000110
instrct...[31:(	00512023
W_data[31:(	00000000
R_addr_1[4:	00
R_addr_2[4:	00
W_addr[4:0]	1d
R_dat...31:0]	00000000
R_dat...31:0]	00000000
data[3...[31:(	00000000
i[31:0]	00000020
addr_...31:0]	00000005
reg_ro...[31:(	00000020
data[15:0][7:	XX,XX,XX,

RTL Design






DM
Funct3[2:0]
R_en
W_data[31:0]
W_en
addr[31:0]
R_data_out[31:0]
Data_Memory
MEM_WB
ALUResult[31:0]
MemtoReg[1:0]
ReadData[31:0]
RegWrite
Reg_rd[4:0]
clock
nextPC[31:0]
ALUResult_out[31:0]
MemtoReg_out[1:0]
ReadData_out[31:0]
RegWrite_out
Reg_rd_out[4:0]
nextPC_out[31:0]
MEM_WB_State_Reg
Mux_4
data1[31:0]
data2[31:0]
data3[31:0]
data4[31:0]
sel[1:0]
result[31:0]
_32_bit_4to2_MUX
6:0
ImmGen
In[31:0]
Out[31:0]
Immediate_Generator
CU
opcode[6:0]
ALUOp[1:0]
ALUSrc
Branch
Jump
MemRead
MemWrite
MemtoReg[1:0]
RegWrite
Central_Control_Unit
ID_EX
ALUOp[1:0]
ALUSrc
ALU_Instrct[3:0]
Branch
Imm_Gen[31:0]
Jump
MemRead
MemWrite
MemtoReg[1:0]
RegWrite
Reg_rd[4:0]
Reg_rs1[31:0]
Reg_rs2[31:0]
clock
crntPC[31:0]
nextPC[31:0]
ALUOp_out[1:0]
ALUSrc_out
ALU_Instrct_out[3:0]
Branch_out
Imm_Gen_out[31:0]
Jump_out
MemRead_out
MemWrite_out
MemtoReg_out[1:0]
RegWrite_out
Reg_rd_out[4:0]
Reg_rs1_out[31:0]
Reg_rs2_out[31:0]
crntPC_out[31:0]
nextPC_out[31:0]
ID_EX_State_Reg
RF
R_addr_1[4:0]
R_addr_2[4:0]
W_addr[4:0]
W_data[31:0]
W_en
clock
R_data_1[31:0]
R_data_2[31:0]
Register_File
PC_reg[31:0]
C
D
Q
RTL_REG
IM
addr[31:0]
instrct_out[31:0]
Instruction_Memory
IF_ID
Instrct[31:0]
clock
crntPC[31:0]
nextPC[31:0]
Instrct_out[31:0]
crntPC_out[31:0]
nextPC_out[31:0]
IF_ID_State_Reg


ALU_Ctrl
ALU_op[1:0]
instrct[3:0]
ALU_sel[3:0]
ALU_Control
ALU
data1[31:0]
data2[31:0]
sel[3:0]
Zero
result[31:0]
N_bit_ALU
Mux_2
data1[31:0]
data2[31:0]
sel
result[31:0]
_32_bit_2to1_MUX