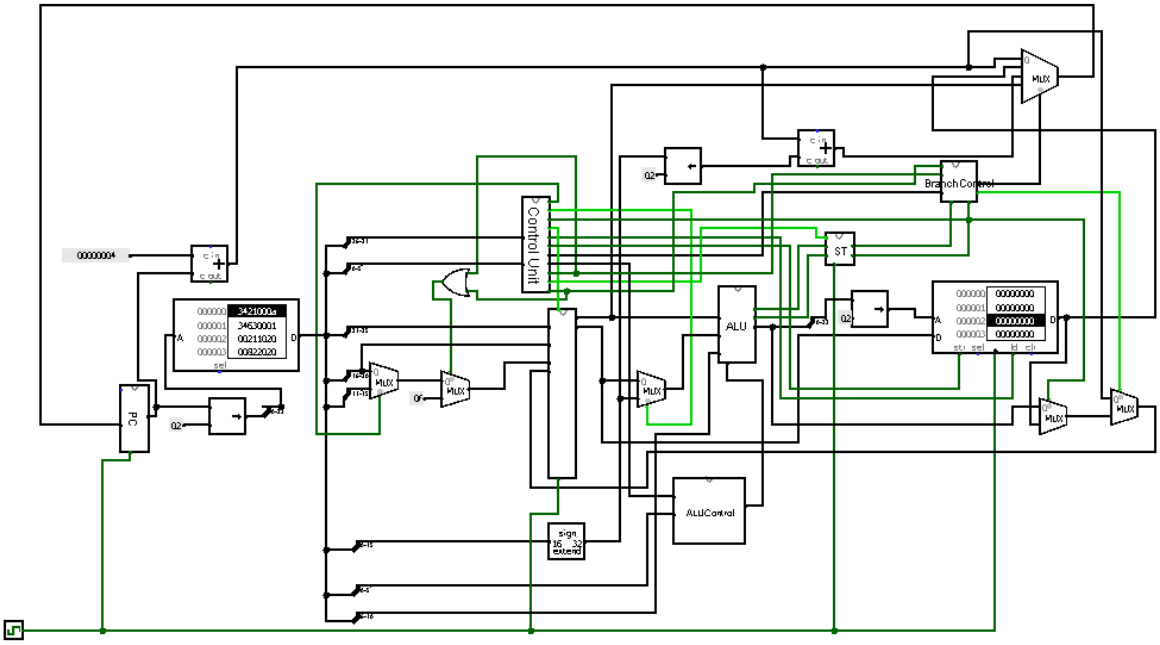
CSE 3038 PROJECT2 REPORT

ALI BERAT CETIN-150117822, RIDVAN GULCU 150117508

ABSTRACT

In this project, we are supposed to make additional implementations of the MIPS instruction set. We implemented the default single cycle datapath design on logism for further implementations. We wanted to see real design and test them before implementing on Verilog. It made so easy before Verilog implementation.



Instructions implemented: **Bgez, Jmadd, Balrn, Ori, Jpc, Srl.**

Bgez If R[rs] >= 0, branch to PC-relative address.

We used negative-flag of ALU to check magnitude of rs register. If the register rs holds value less than zero, negative flag is on. So branch decision is wrong. If negative flag is off(zero) branch decision is true PC is set to Label. Control unit generates an AluOp for Alu Control. Alu Control passes the AluOp to the Alu directly. Since this in an I-type instruction, there is no function code.

The Alu input is 011 for Bgez instruction. Alu doesn't make any operation to check Rs>=0 just uses value of the rs register. In default design, RS register is the first operand of Alu input. Alu result=rs in this case. So negative value holds " >=0" information for us. As we learned, BranchControlUnit is implemented. Because there are bunch of branch and jump instructions that need logic around negative and zero flags. There is a single Mux that decides the next PC address. Since the label address of the Bgez instruction same as Beq instruction because of they are I-type, the target address comes from the

same part of the instruction. BranchControlUnit just decides whether the Bgez condition is true. The label address if sign extended of 15-0 bits.

Ori Put the logical OR of register $rs and the zeroextended immediate into register $rt.

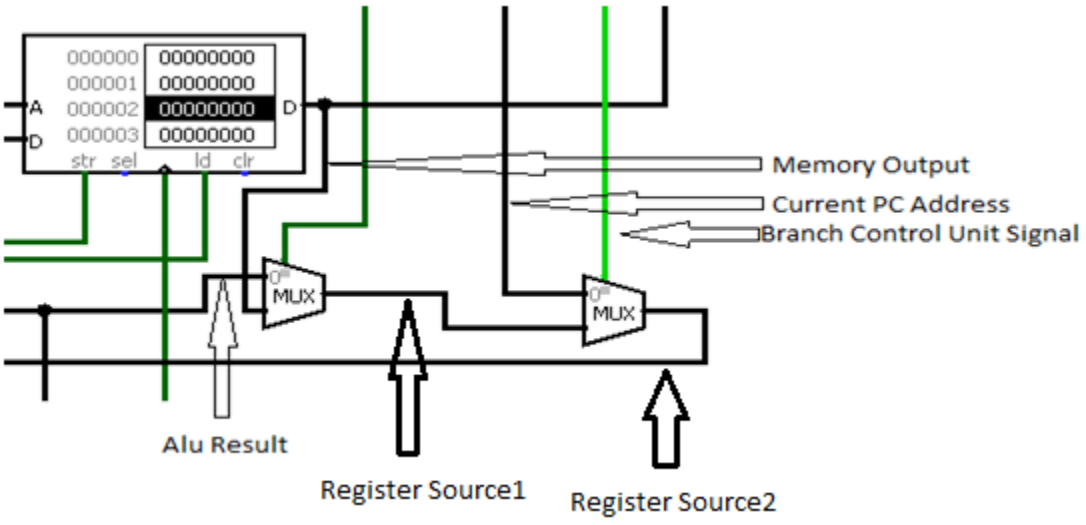
The difference between Or and Ori is opcode, immediate part and destination part. Alu is already capable to make Or operation. So we just need the immediate value to the second operand of the Alu. AluSrc makes it. So control unit generates AluSrc signal as 1 to choose immediate value from instruction. First operand is rs. As mentioned in first part, the destination register of the I-type is different r-type. RegDst is 0 to select rt(20-16bits) of the instruction. There is no other design need for Ori instruction.

Balrn If Status [N] = 1, branches to address found in register $rs link address is stored in $rd (which defaults to 31)

Because balrn instruction checks Negative Flag, we need to decide when to update these flags. Addition to default design we added a StatusRegWrite signal. For the instructions that use flags StatusRegWrite signal should be zero. The root of this problem is the single cycle datapath. We implemented a BranchControlUnit for Bgez instruction. 1-bit zero and negative flag registers are in BranchControlUnit(BCU). In default design, the address decision Mux has three input. Jump, Branch and PC+4 target address. Additionally on Balrn instruction, there is an input from Register File to be PC set to Rs register. BCU generates mux signal to choose PC-Target-Address as value of Rs register. In our provided default design, the sources of the destination register were memory output and alu result. In this instruction we need to add PC+4 to be source of destination register. So there are three different source for the destination register now. PC+4, Memory and Alu. We put a second mux with a selector bit comes from BCU.

Jpc jumps to PC-relative address (formed as beq and bne do) link address is stored in $rt (which defaults to 31)

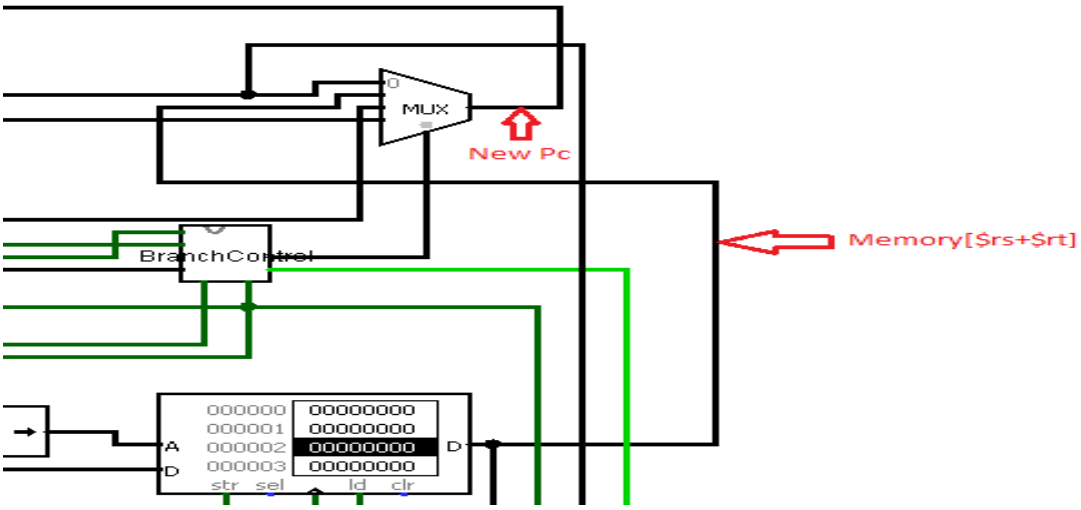
Jpc is unconditional jump and link instruction. BranchControlUnit generates a signal for Mux that selects which source will be set as next PC address. With the jpc instruction PC address choices are PC+4, Jump and Branch. As we did in Balrn instruction jpc also stores the current address to $31(default).



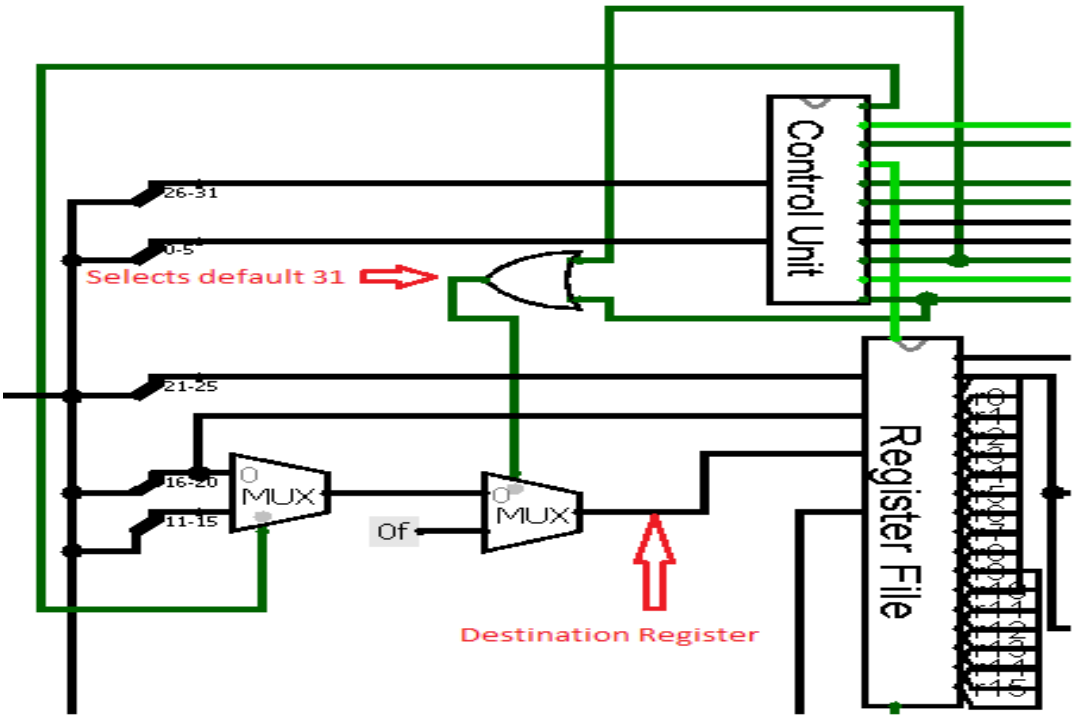
Jmadd jumps to address found in memory [$rs+$rt]

Jmadd is like Lw. Because it sums up two value and read from data memory. Difference is Jmadd doesn't have immediate value. To make sum operation, alu control unit generates corresponding alu operation signal. Because jmadd is an R-type instruction, no update needed in Alu. It uses default sum operation. But the Alu result refers to a memory address. So MemRead signal should be generated from control unit according to the function code of jmadd instruction. The memory output is connected to

the PC-Addr-Mux. Because PC = Memory[$rs+$rt]. Link address should be stored in $31. The implementation of storing PC to the $31 is same as jpc instruction.



The mux above has 4 inputs. These are PC+4,JumpLabel,BranchLabel,AddressFromMemory. Branch Control Unit knows the current instruction and generates relevant signal for PC-Addr-Mux.

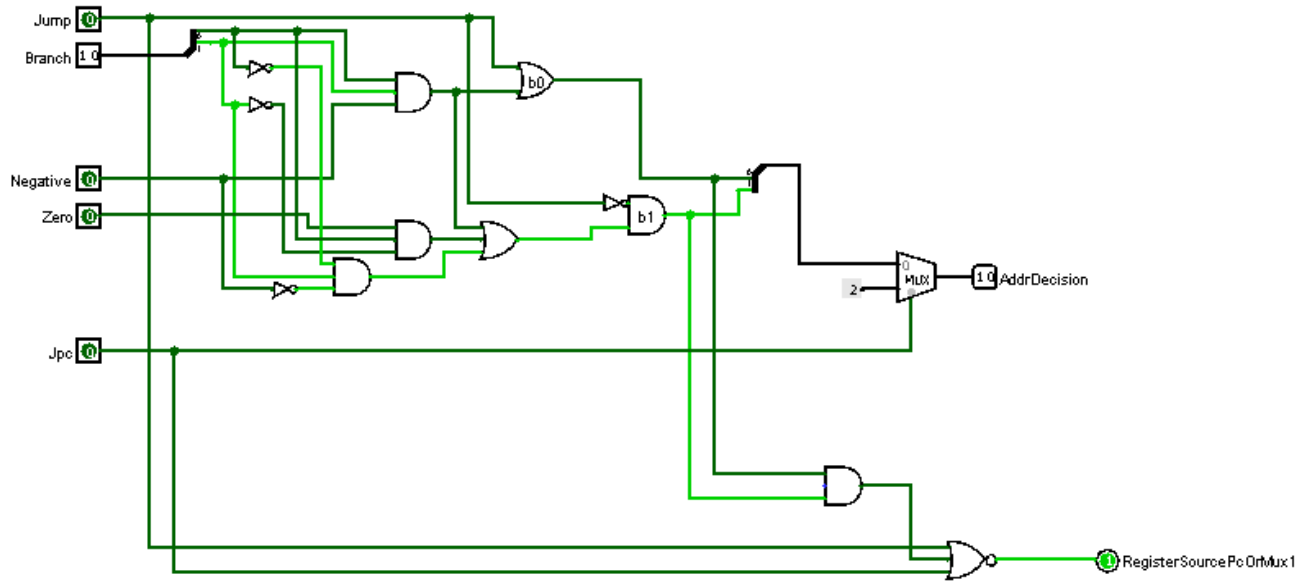


The inputs of the or gate in the figure above are Jpc and Jmadd instruction. So the destination register remains a default $31. In logism we used $15 but in Verilog it is $31.

Srl shift register $rt to right by shift amount (shamt) and store the result in register $rd.

Srl is an R-type instruction so according to the function code(2) alu control unit generates an aluop signal to pass the shifted value of the register to the result. Shift operation is made on rt register. So the second alu input is shifted. Because rt is the second register output of the register file. Shift amount is given in instruction so alu takes that value to know how many bits to be shifted to the left. It is logical right so sign is not preserved.

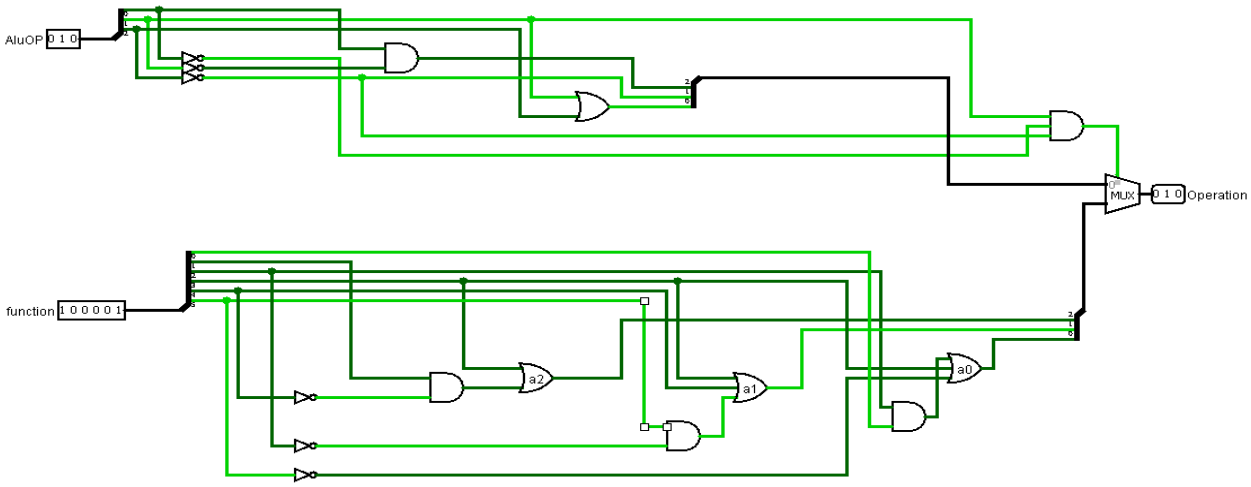
BRANCH CONTROL UNIT



Branch control unit operates the logic for branch instructions and decides which address should be the new PC. RegisterSourcePcOrMux1 signal is generated to select register source as current PC(linking). Two bit branch input represents the branches. In our implementation two branch instruction is present. Beq(default), Bgez and Balrn.

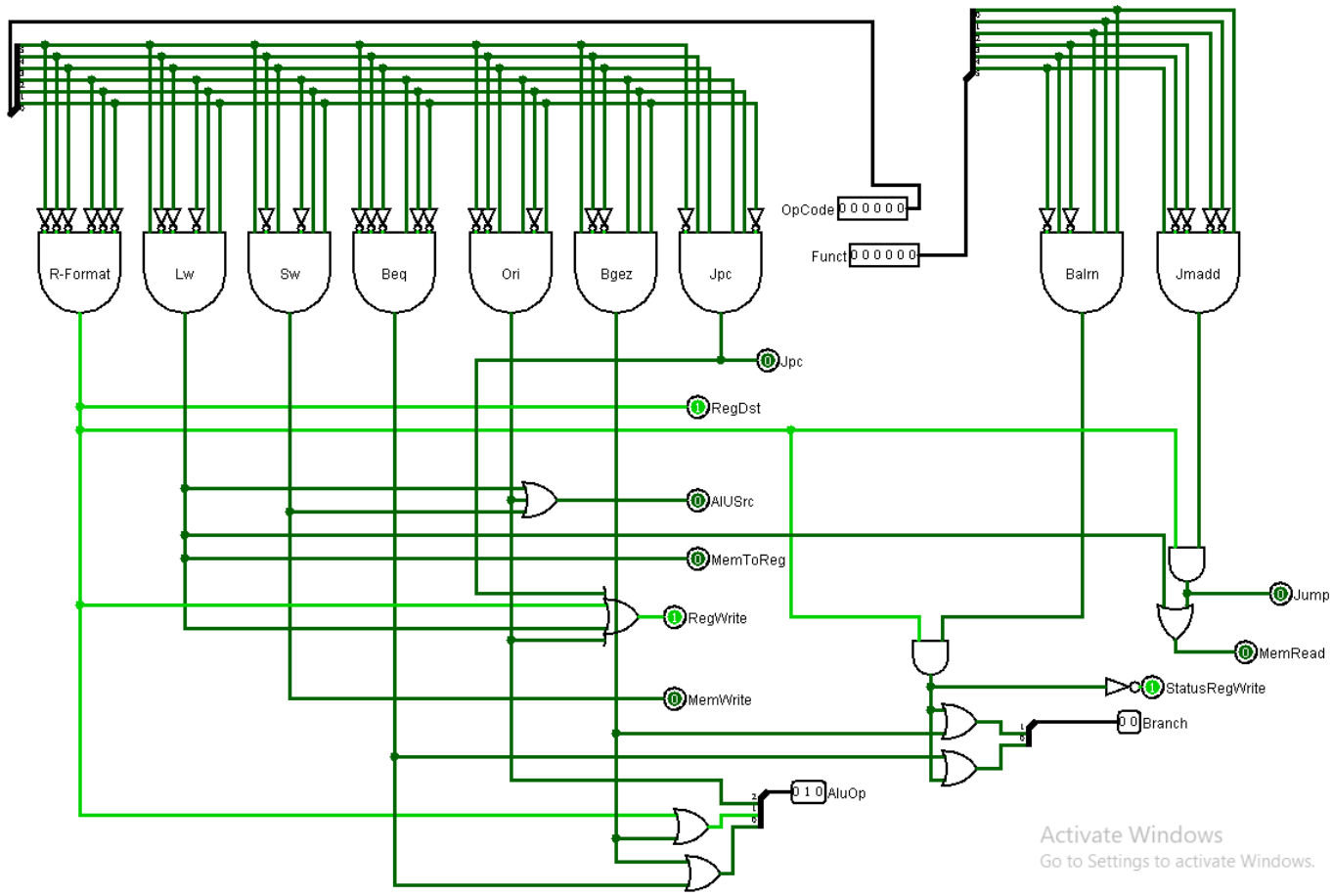
ALU CONTROL UNIT, CONTROL UNIT,BRANCH CONTROL UNIT & TRUTH TABLES

Alu Control & Truth Table



	INPUT									OUTPUT
	Op2	Op1	Op0	F5	F4	F3	F2	F1	F0	Alu
ori	1	0	0	x	x	x	x	x	x	001
lw&sw	0	0	0	x	x	x	x	x	x	010
bgez	0	1	1	x	x	x	x	x	x	011
	0	0	1	x	x	x	x	x	x	110
add fnc=32	0	1	0	**1**	**0**	**0**	**0**	**0**	**0**	010
slt fnc=42	0	1	0	**x**	**x**	**1**	**x**	**x**	**x**	111
sub fnc=34	0	1	0	**1**	**0**	**0**	**0**	**1**	**0**	110
or fnc=37	0	1	0	**1**	**0**	**0**	**1**	**0**	**1**	001
and fnc=36	0	1	0	**1**	**0**	**0**	**1**	**0**	**0**	000
Shift fnc = 2	0	1	0	**0**	**0**	**0**	**0**	**1**	**0**	101
blrn fnc=23	0	1	0	**0**	**1**	**0**	**1**	**1**	**1**	011
jmadd fnc=33	0	1	0	1	0	0	0	0	1	010

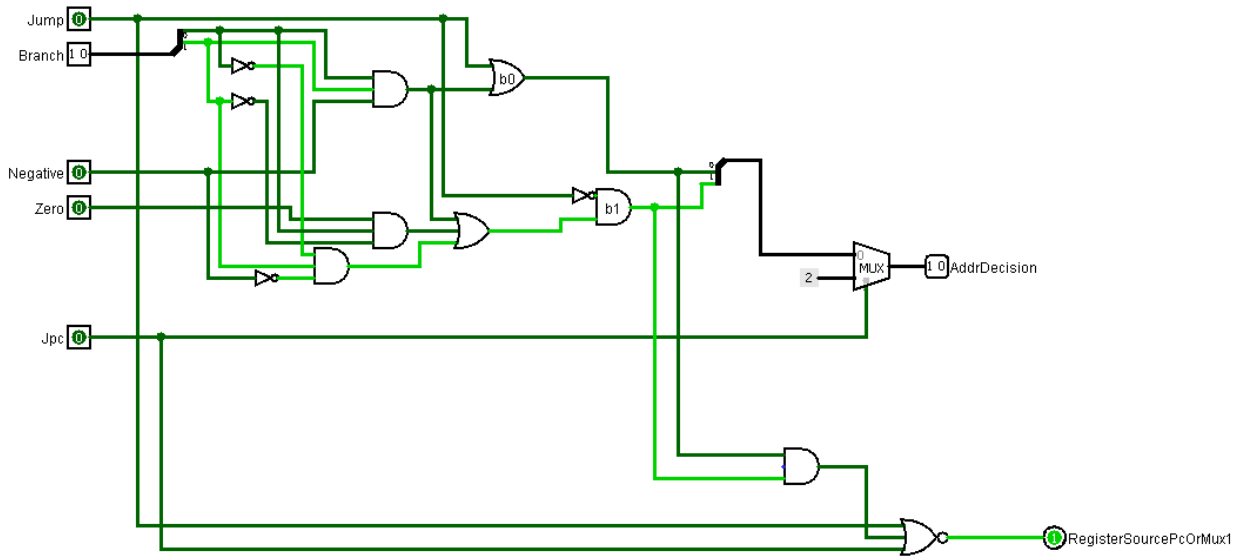
Control Unit



Control Unit's AluOp Signal Truth Table

INPUT						OUTPUT		
r-type	**beq**	**bgez**	**lw**	**sw**	**Ori**	**op2**	**Op1**	**Op0**
x	x	x	x	x	1	1	0	0
x	x	x	x	1	x	0	0	0
x	x	x	1	x	x	0	0	0
x	x	1	x	x	x	0	1	1
x	1	x	x	x	x	0	0	1
1	x	x	x	x	x	0	1	0

Branch Control Unit & Truth Table

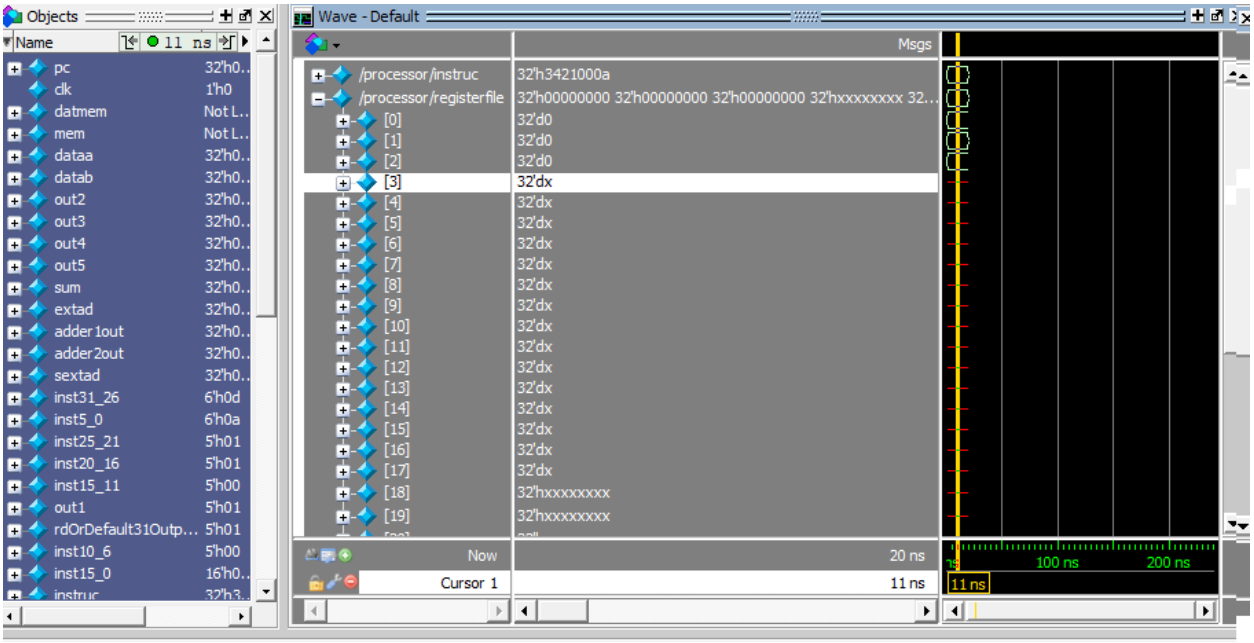


INPUT					OUTPUT	
b1	b0	j	n	z	o1	O0
x	x	1	x	x	0	1
1	0	0	0	x	1	0
0	1	0	x	1	1	0
1	1	0	1	x	1	1

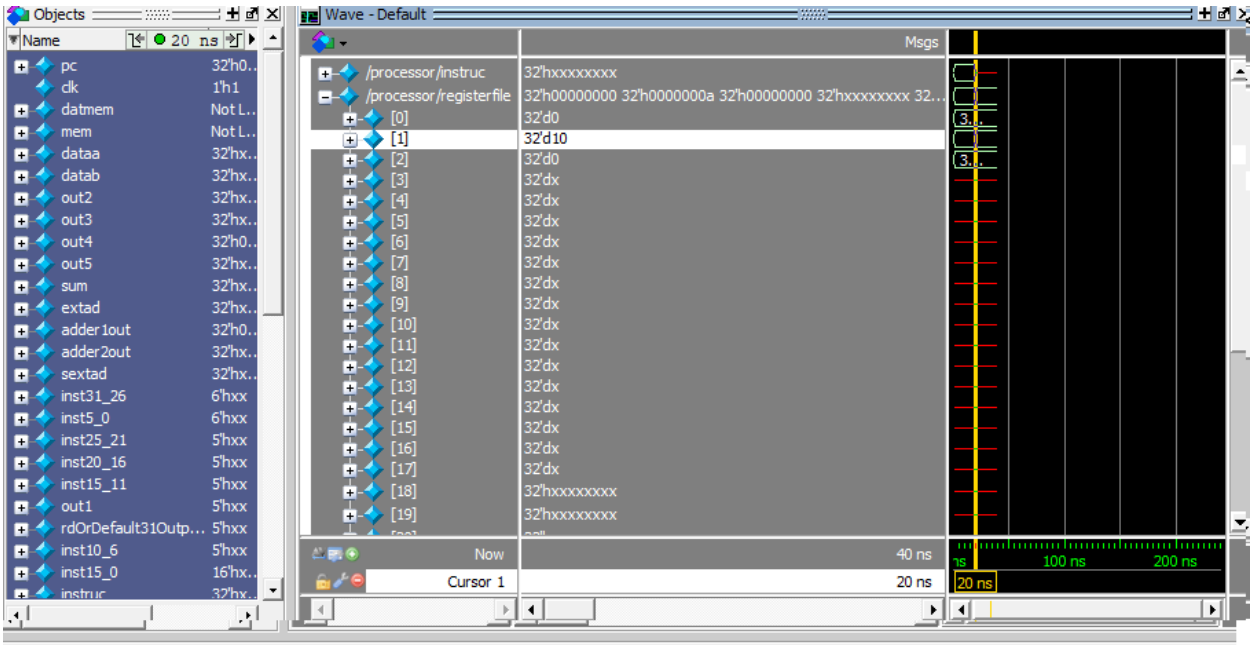
EXAMPLE RUNS

ORI

r1 initialized as 0

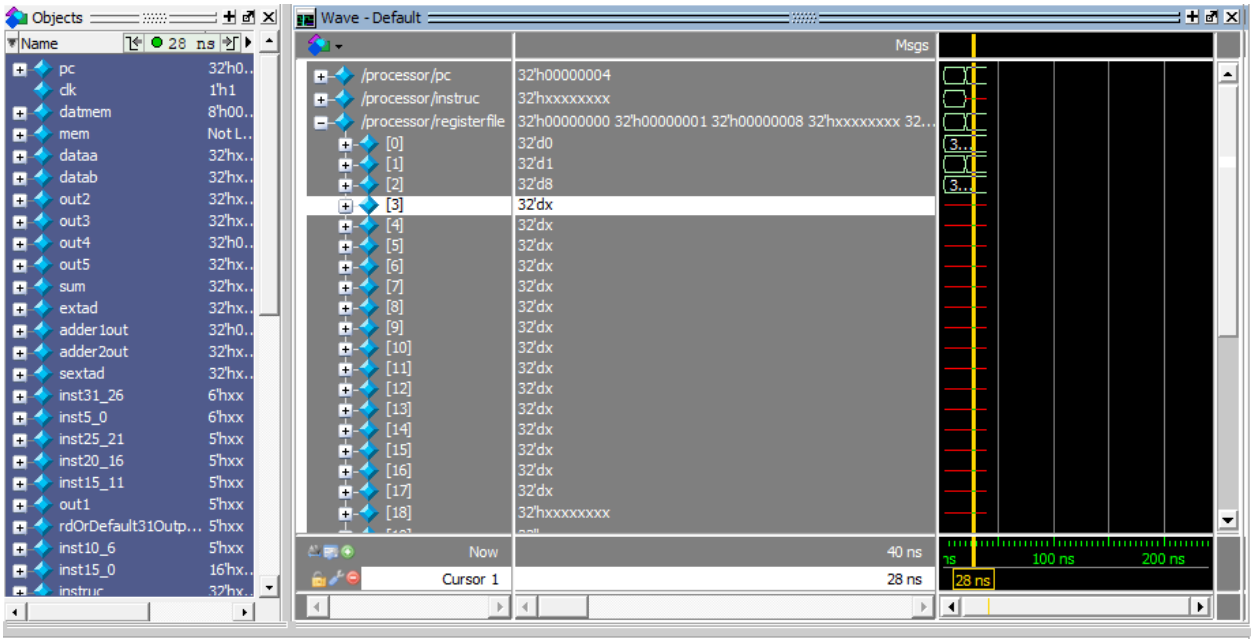


Executed instruction: ori r1,r1,10

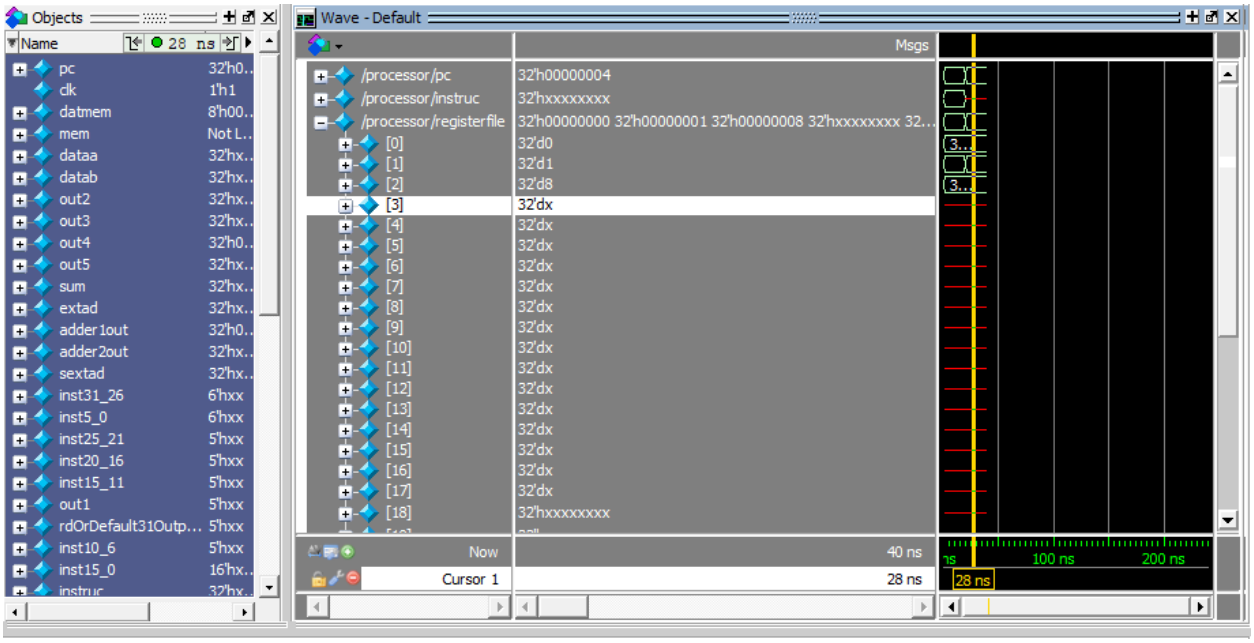


SRL

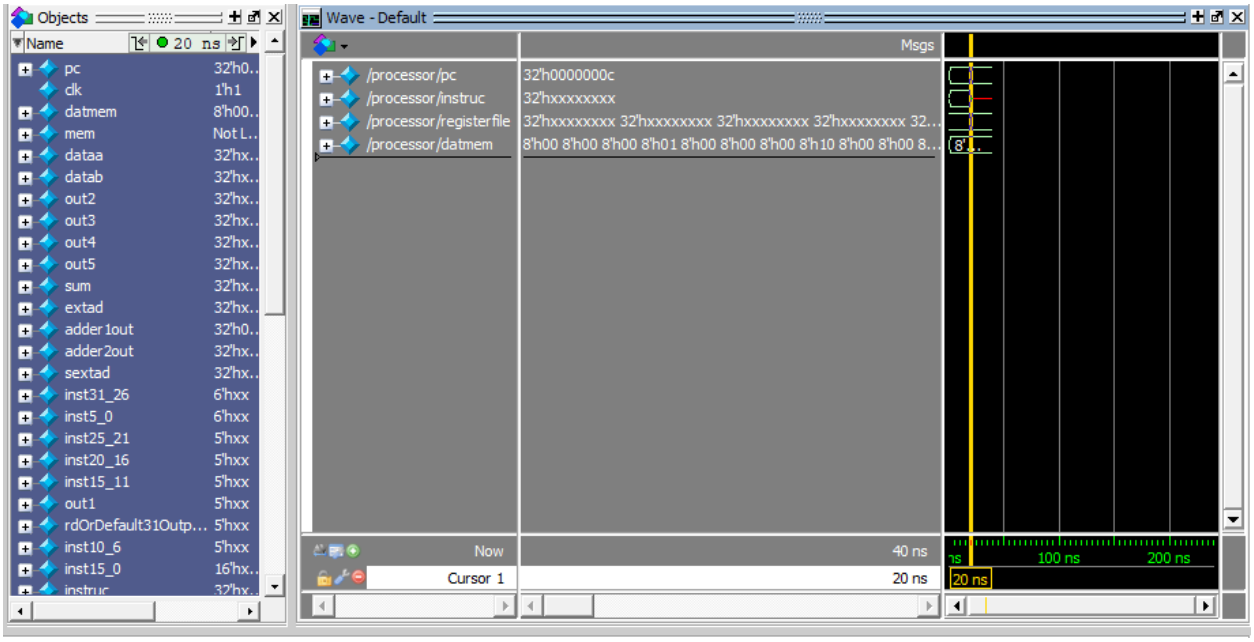
R2 initialized 8



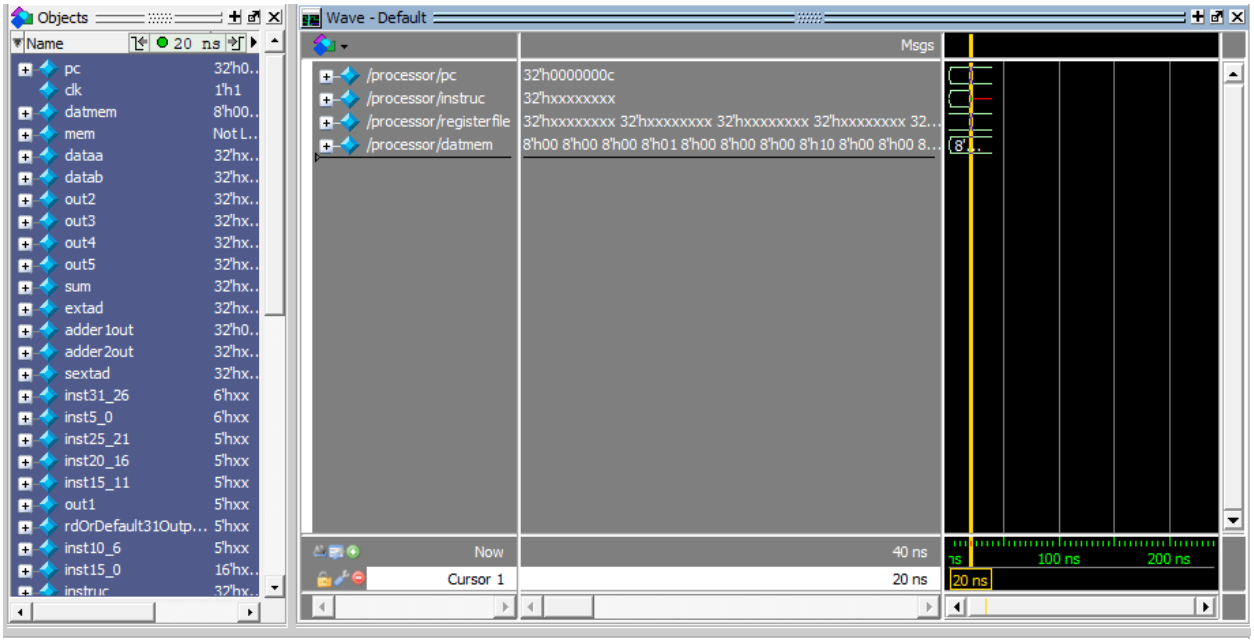
Srl r1,r2,3. Shift 8 3 times, it is 1.



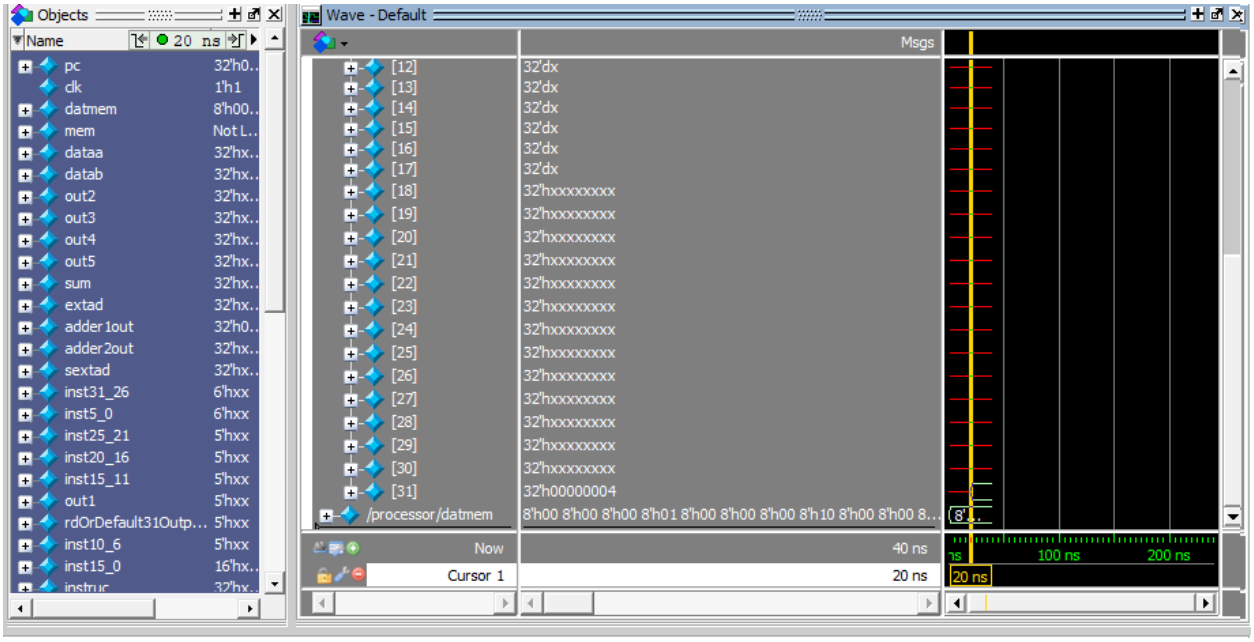
JPC



Jpc 2 = PC+4+8



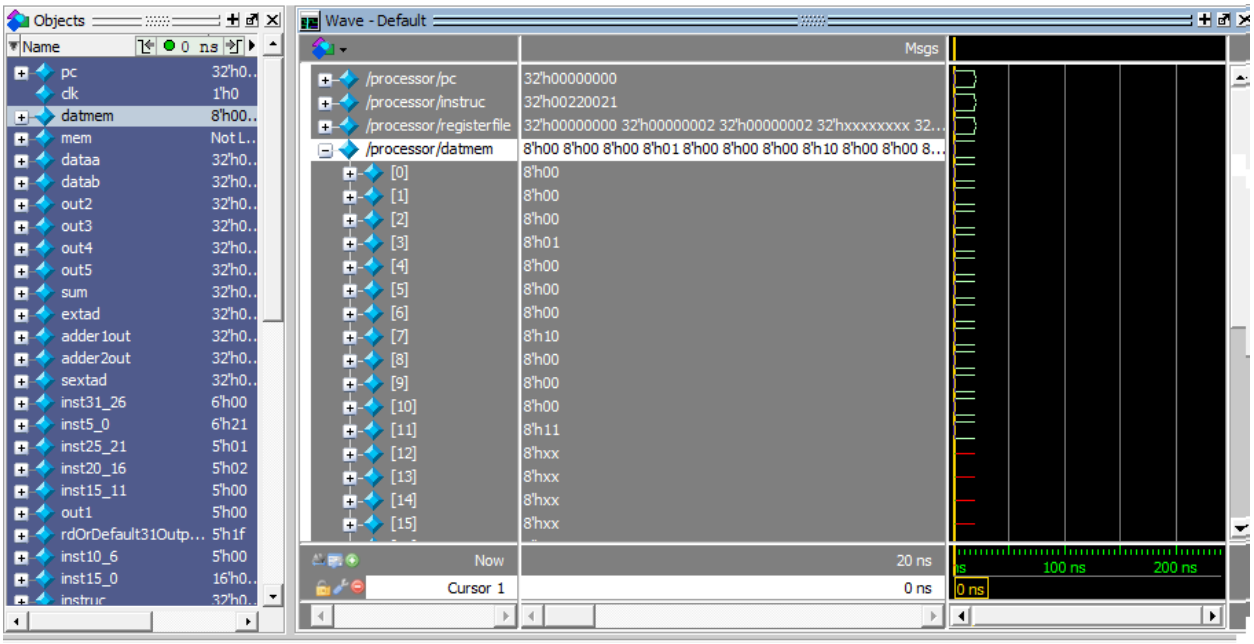
Jpc updates $31 with PC+4(linking)

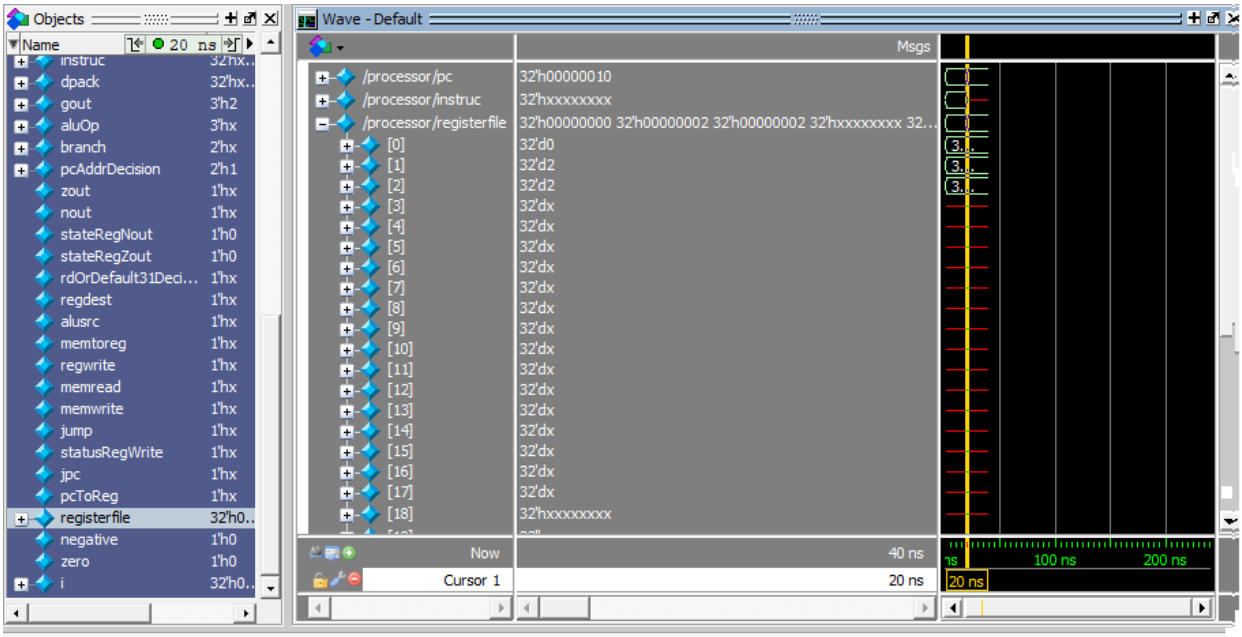


Jmadd

For jmadd, data memory and register file initialized.

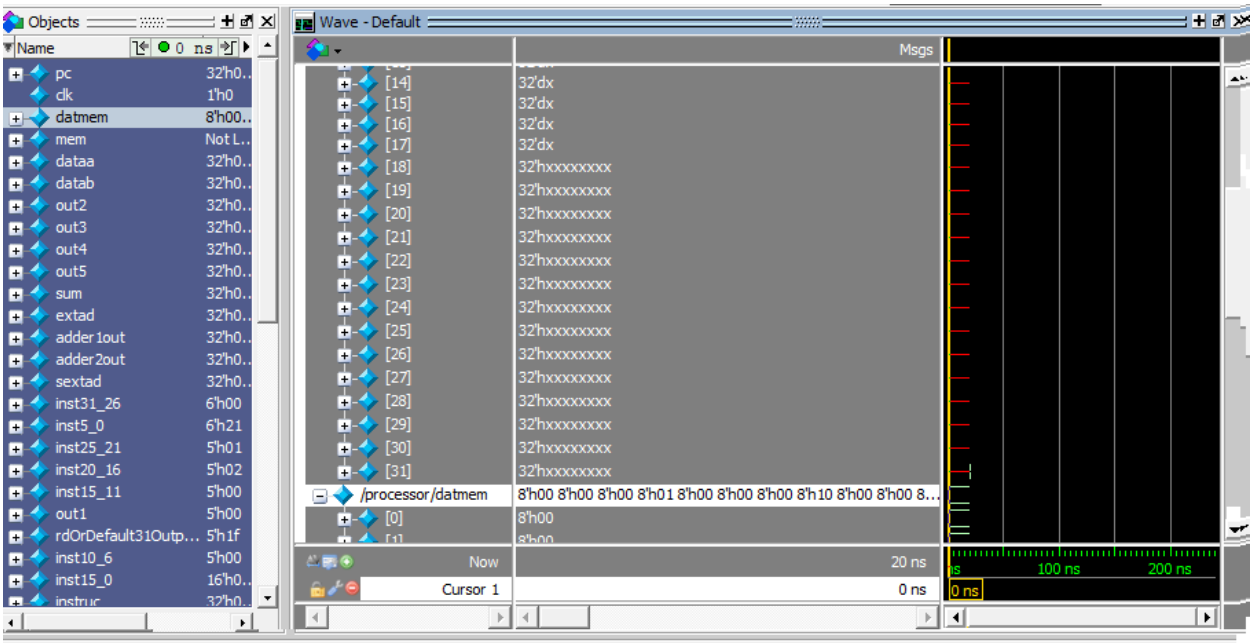
Jmadd $r1,$r2



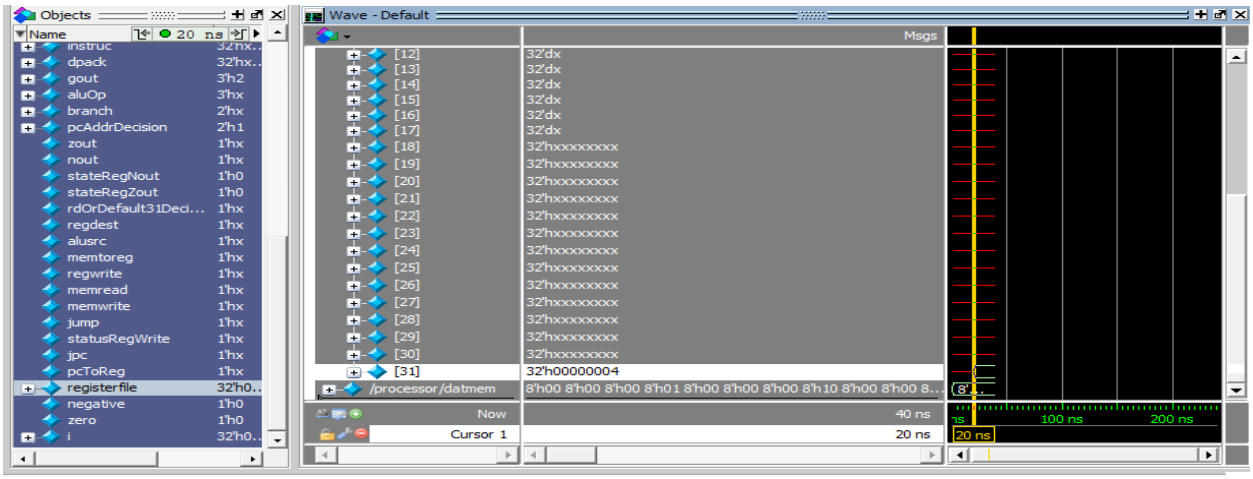


Jmadd $r1,$r2. $r1+$r1 is 4. New pc address will be Memory[4]. Memory has [00 00 00 10] in 4th byte address. So pc is updated as d'2. Jmadd also updated $31 as PC+4

Before Jmadd executed, $31 is empty.



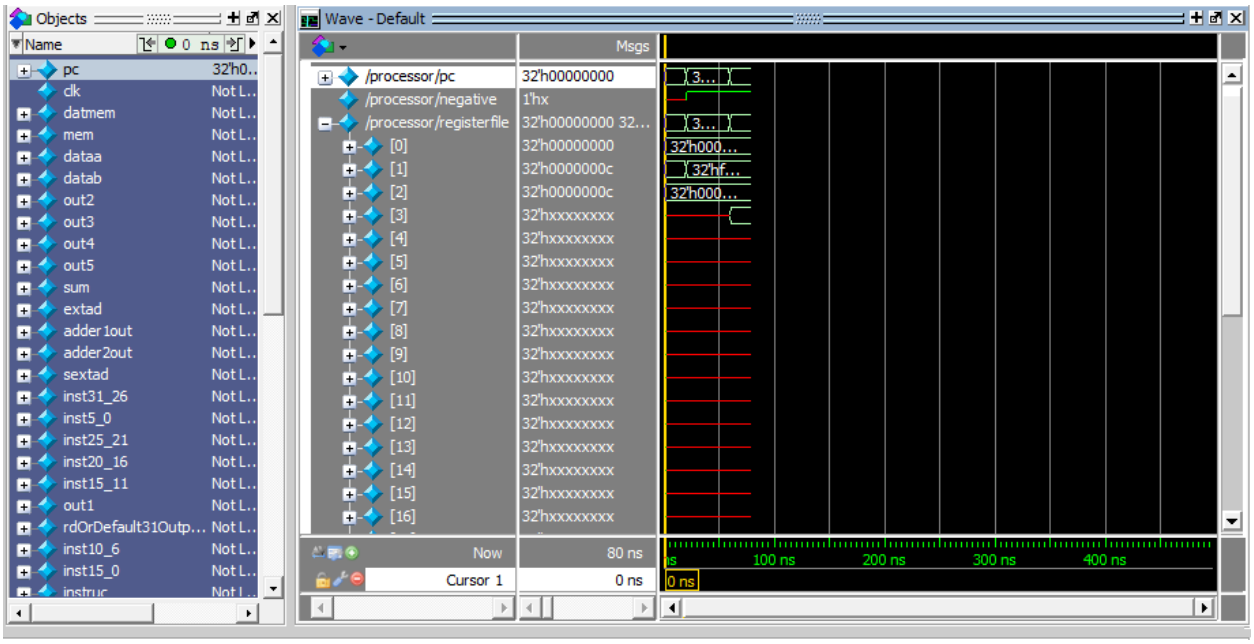
After Jmadd executed $31 is PC+4



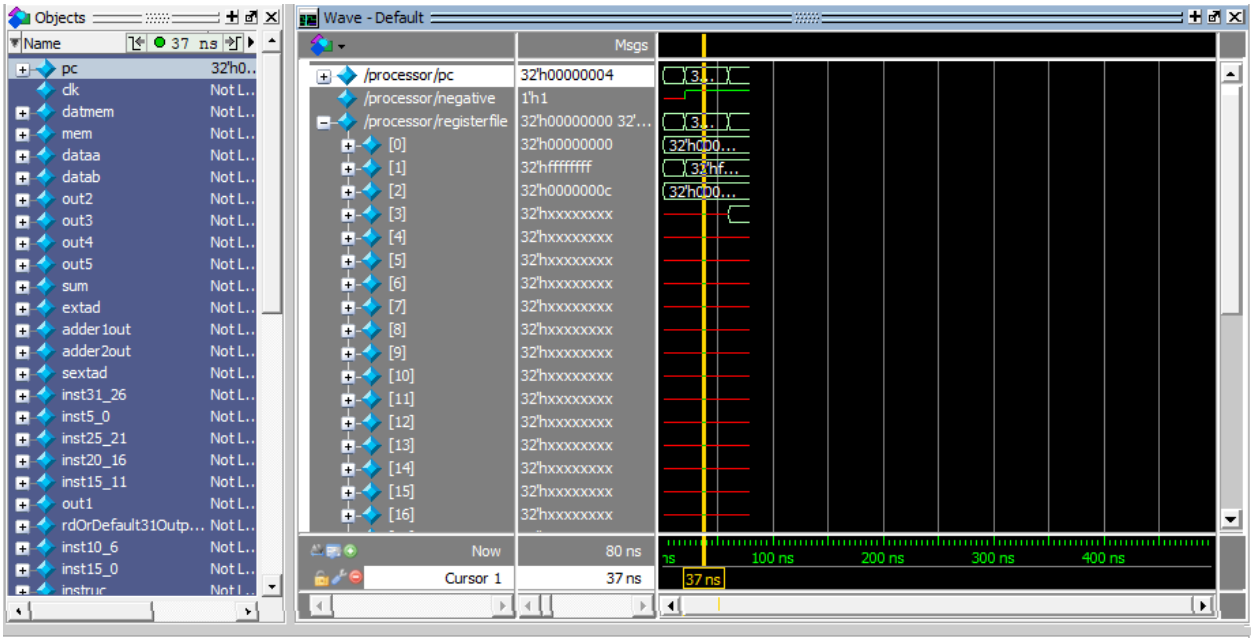
Balrn

Before executing balrn we need to set negative flag. So ori r1,r1,-1 makes negative flag 1.

Before execting ori r1,r1,-1

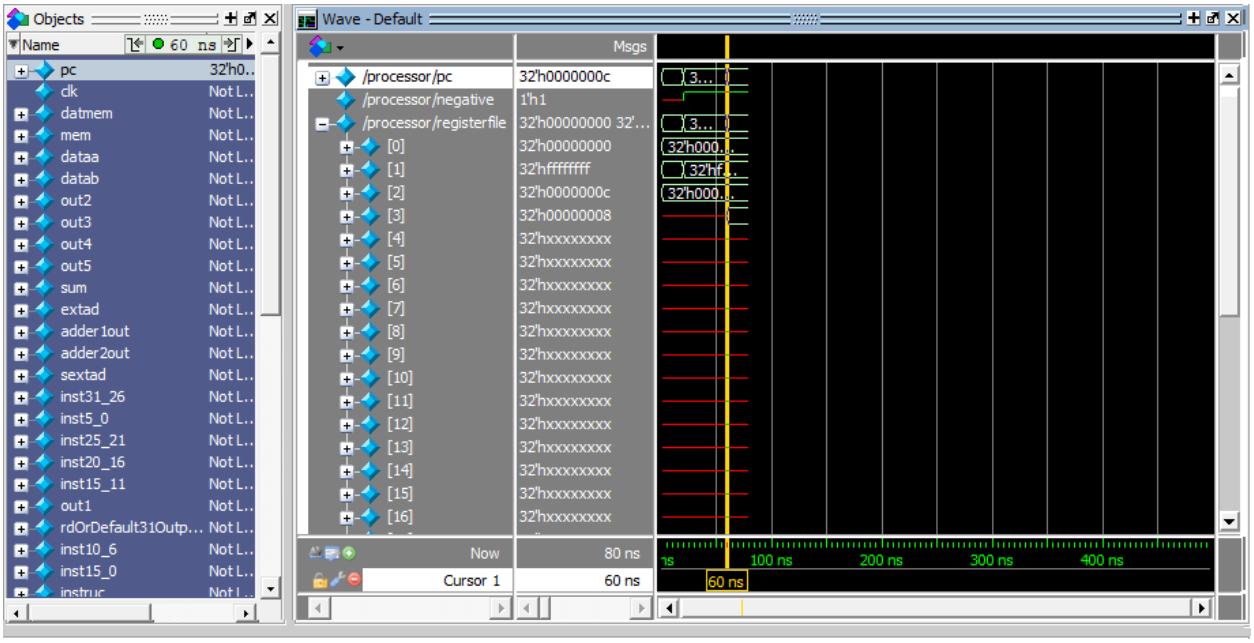


After executing ori r1,r1,-1



Negative flag is 1 and r1 is -1. Balrn branch decision should be true now.

Barln $r2,$r3

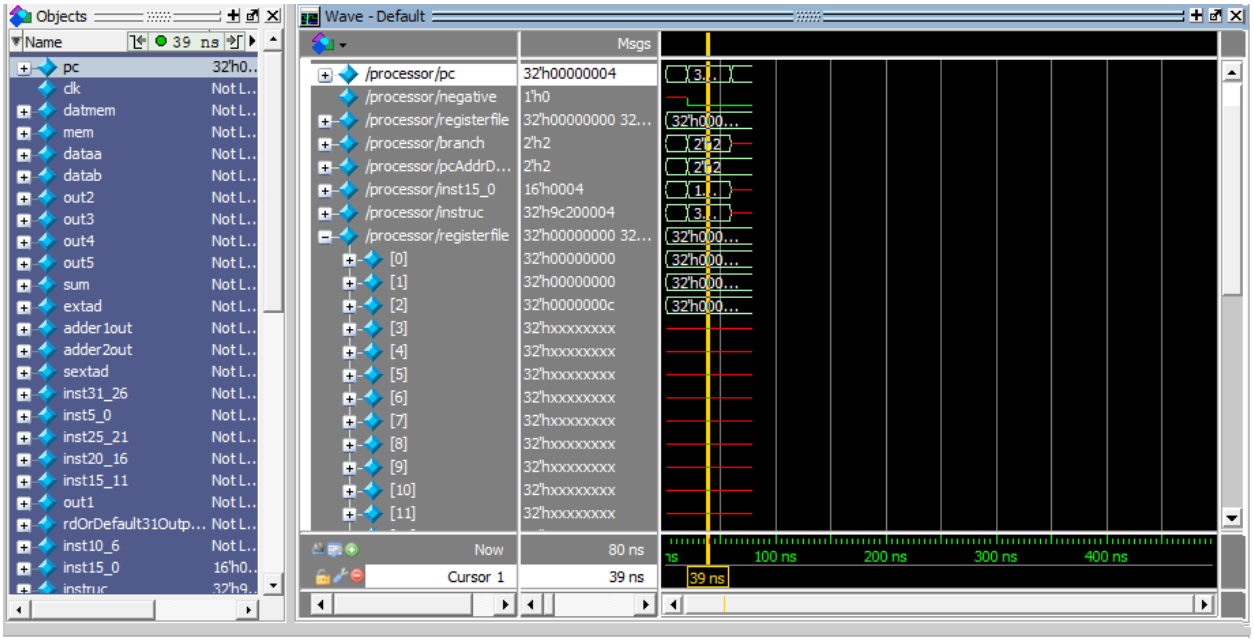


Pc was d’4 and jumped to value of r2(d’12) and balrn also store PC+4 to r3 which is 4+4=8

Bgez

Bgez $r1,4

Before Bgez executed



PC was 4 and bgez has label 4 which is 16 in byte address. So new PC is PC+4+16 so it is 24.