CSE 3038 COMPUTER ORGANIZATION

Programming Project 2 Report

04.06.2022

Prepared By

İSMAİL ÖKSÜZ - 150119516
BARIŞ HAZAR - 150118019
ULAŞ DENİZ IŞIK - 150118887
EMRE SAĞIROĞLU - 150119766

TABLE OF CONTENTS

1.Introduction:

Tables for each signals that were used, extra Verilog files that were added, instructions that we used (represented in this section either), new components and single-cycle datapath are represented in this document.

Before we begin, the instructions that were used to complete this project are balmn, brz, jmadd, bn, sllv and jspal. In short:

Instr	Type	Code	Syntax	Meaning
balmn	I-type	opcode=24	balmn $rt, imm16($rs)	if Status [N]=1, branches to address found in memory, link address is stored in $rt
brz	R-type	funct=20	brz $rs	if Status [Z]=1, branches to address found in register $rs
jmadd	R-type	funct=32	jmadd $rs,$rt	jumps to address found in memory [$rs+$rt], link address is stored in $31
bn	J-type	opcode=25	bn Target	if Status [N]=1, branches to pseude-direct address (formed as j does)
sllv	R-type	func=4	sllv $rd, $rt, $rs	shift register $rt to left by the value in register $rs, and store the result in register $rd
jspal	I-type	opcode=19	jspal	jumps to address found in memory where the memory address is written in register 29 ($sp) and link address is stored in memory (DataMemory[Register[29]])

2.Tables:

2.1.Table 1:

Table 1 shows the values of all instructions for each control signals.

	RegDst	Jump	Branch	MemRead	MemToReg
(5) balmn	0	0	0	1	X
(7) brz	X	0	0	0	0
(11) jmadd	X	0	0	1	X
(15) bn	X	0	0	0	X
(24) sllv	1	0	0	0	0
(28) jspal	X	0	0	1	0

	AluOP	MemWrite	AluSRC	RegWrite
(5) balmn	00	0	1	1
(7) brz	X	0	X	0
(11) jmadd	00	0	0	1
(15) bn	X	0	X	0
(24) sllv	10	0	0	1
(28) jspal	X	1	X	0

2.2.Table 2:

Table 2 shows the values of all instructions for each signals that were added by us. Name of the signals and the values are represented as:

	MUX3 Signal	31 Signal	linkAdr Signal	29 Signal	PC+4 Signal	AddrMem Signal
(5) balmn	001	0	1	0	0	0
(7) brz	010	0	0	0	0	0
(11) jmadd	011	1	1	0	0	0
(15) bn	100	0	0	0	0	0
(24) sllv	000	0	0	0	0	0
(28) jspal	101	0	0	1	1	1

3.Additional Verilog Files:

To complete the design accurately, some extra Verilog files were required. Consequently, three extra Verilog files were added.

3.1.'mult8_to_1_32.v'

The first extra Verilog file that is created is 'mult8_to_1_32.v'. That file represents a 8 to 1 multiplexer.

```
1   module mult8_to_1_32(out, i0,i1,i2,i3,i4,i5,i6,i7,s0);
2   output [31:0] out;
3   input [31:0]i0,i1,i2,i3,i4,i5,i6,i7;
4   input [2:0]s0;
5   assign out = (s0 == 3'b000) ? i0 :
6                (s0 == 3'b001) ? i1 :
7                (s0 == 3'b010) ? i2 :
8                (s0 == 3'b011) ? i3 :
9                (s0 == 3'b100) ? i4 :
10               (s0 == 3'b101) ? i5 :
11               (s0 == 3'b110) ? i6 : i7;
12  endmodule
```

3.2.'new_component.v'

The other extra Verilog file that is created is 'new_component.v'. That file represents a new component which processes ALU operation according to the ALU control line values. This file created for using the values of n and z in the previous cycle.

```
1   module new_component(aluResult,z,n);
2   input [31:0] aluResult;
3   output z,n;
4   assign z = ~(|aluResult);
5   assign n = aluResult[31];
6   endmodule
```

3.3.’flipflop.v'

The last extra Verilog file that is created is 'flipflop.v'. That file stands for flip flop circuit. In this file, N stands for negative results and Z stands for zero results.

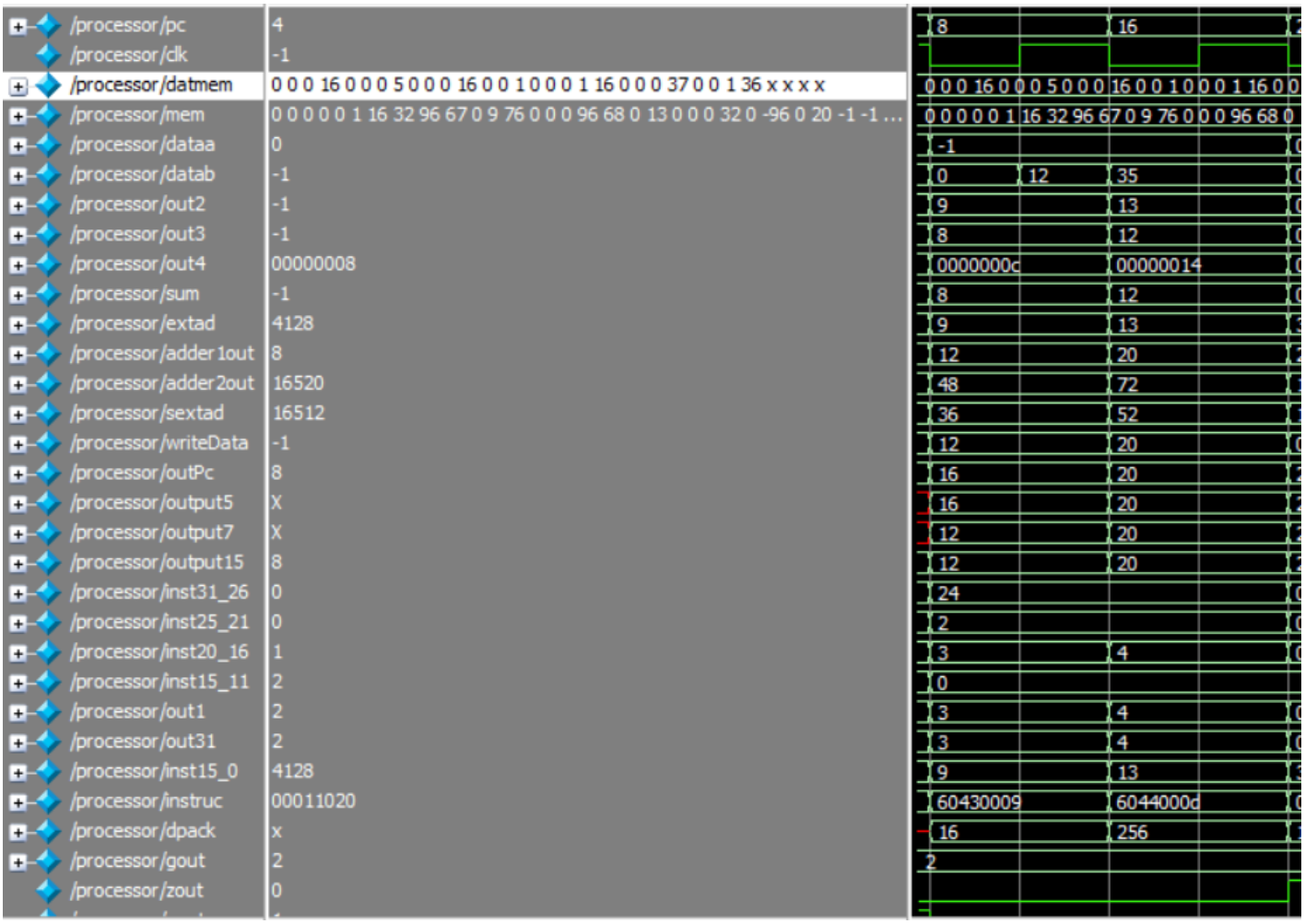
```
1   module flipflop (input d,
2                    input clk,
3                    output reg q);
4
5       always @ (negedge clk)
6           q <= d;
7   endmodule
8
```

4.Instructions:

4.1.balmn

opcode	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

Balmn is an I-type instruction



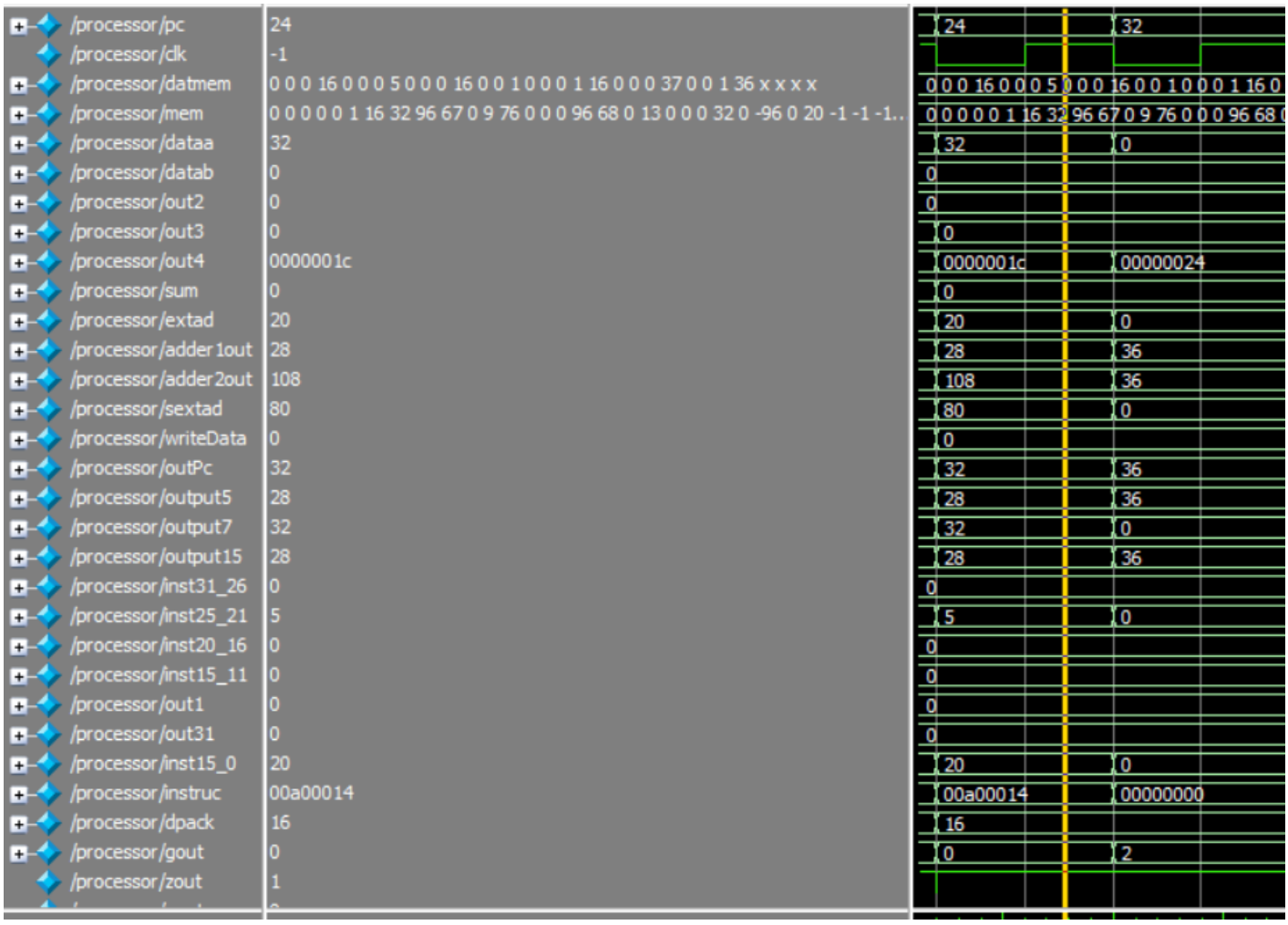
Instruction: 011000 00010 00011 0000000000001001 [if n is 1, branch to memory[$r2 + 9] (r3 holds pc+4)] 0x60430009

$r2 holds -1. And in memory[8], the value is 0x20. So pc branches to 32

4.2.brz

opcode	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Brz is a R-type instruction



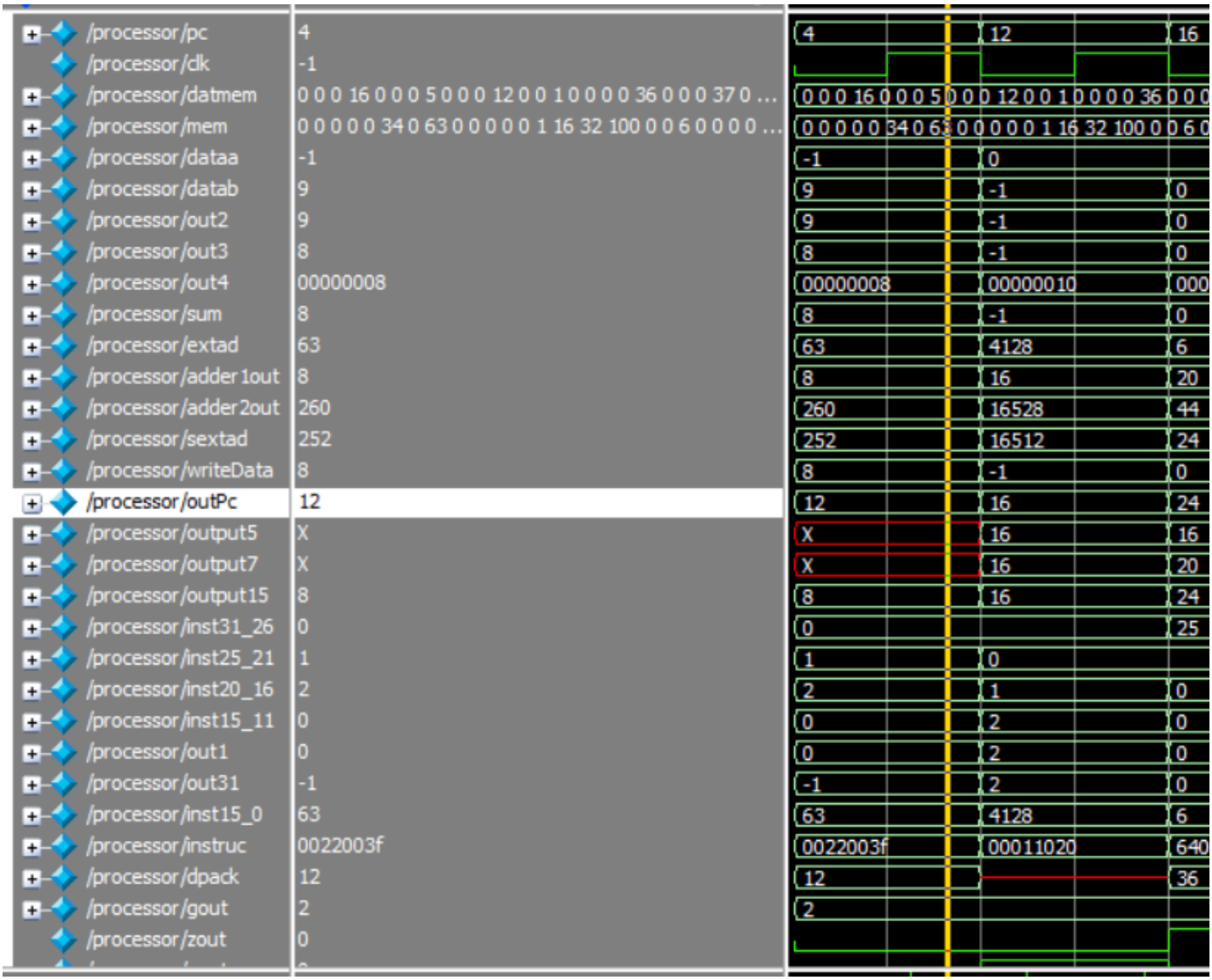
Instruction: 000000 00101 00000 00000 00000 010100 (if z flag is 1 branch to address in $r5) 0x00A00014

Register $r5 holds the value 32, so pc becomes 32 instead of 28

4.3.jmadd

opcode	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Jmadd is a R-type instruction



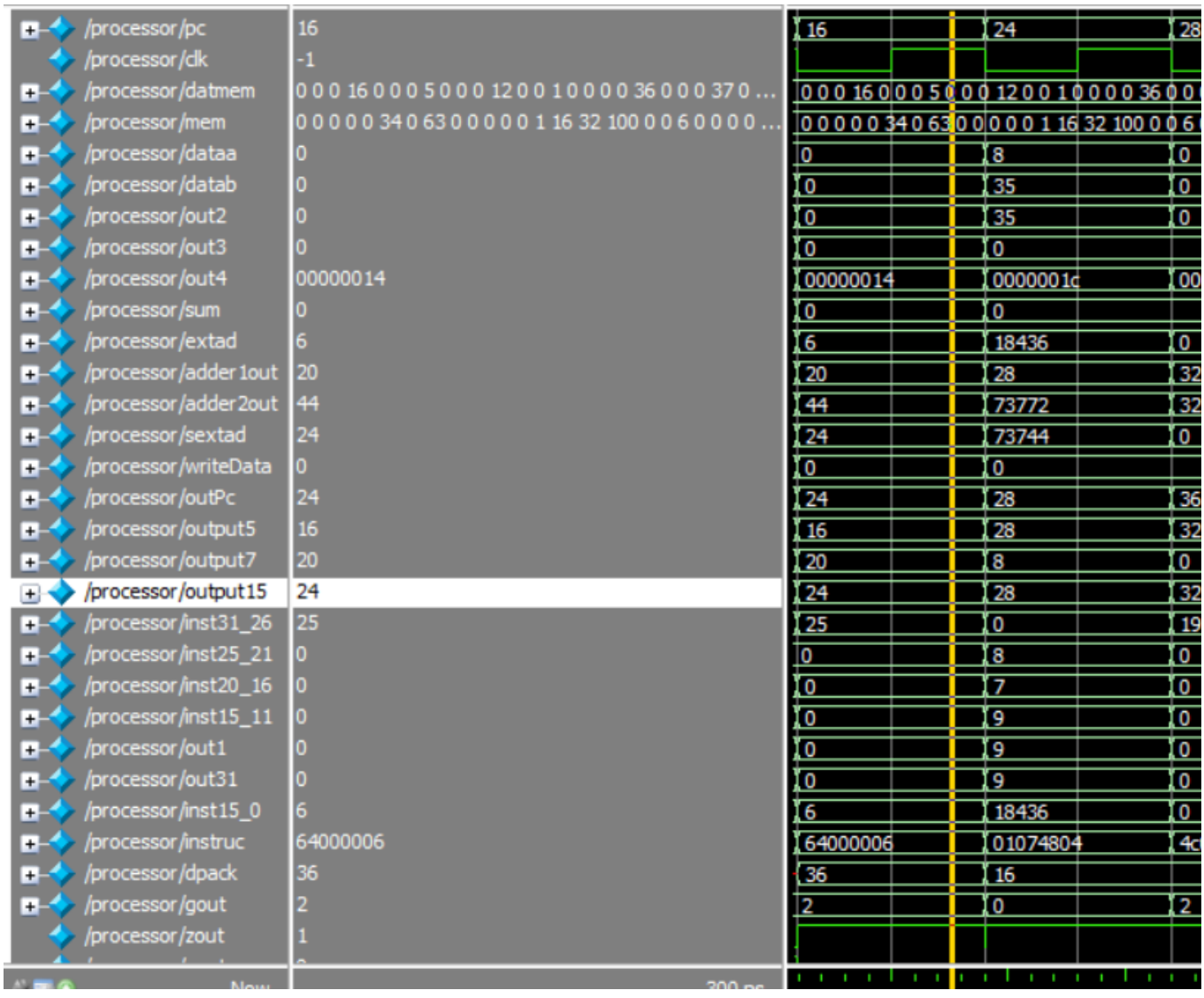
Instruction: 000000 00001 00010 00000 00000 111111

0x0022003F (Jump to address in memory [$r1+$r2] and in $31 store the link address))

4.4.bn

opcode	address
6 bits	26 bits

Bn is a J-type instruction



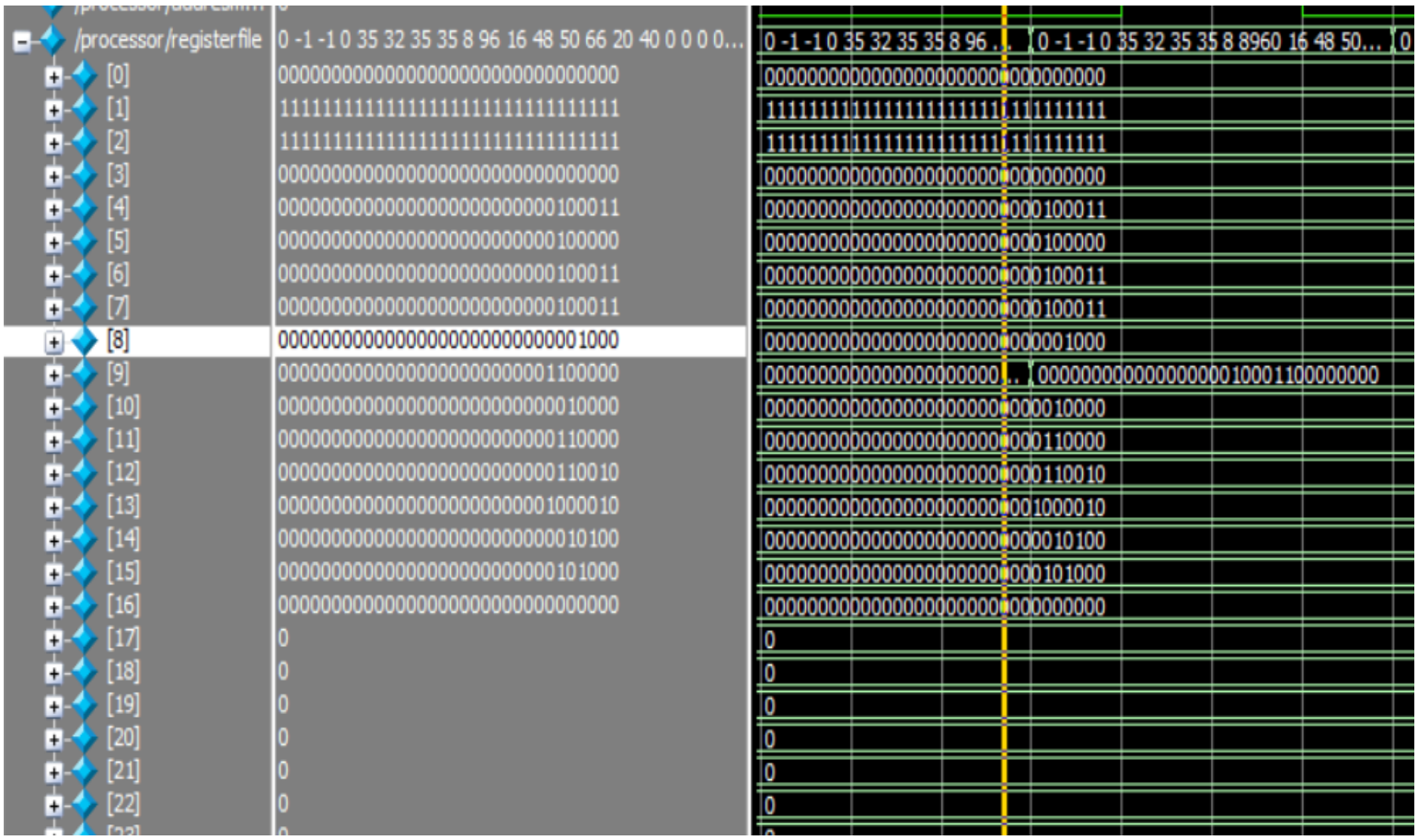
Instruction: 011001 00000000000000000000000110

0x64000006 (If n flag is 1 branch to pseudo direct address with the value 6))

4.5.sllv

opcode	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Sllv is a R-type instruction



Instruction: 000000 01000 00111 01001 00000 000100

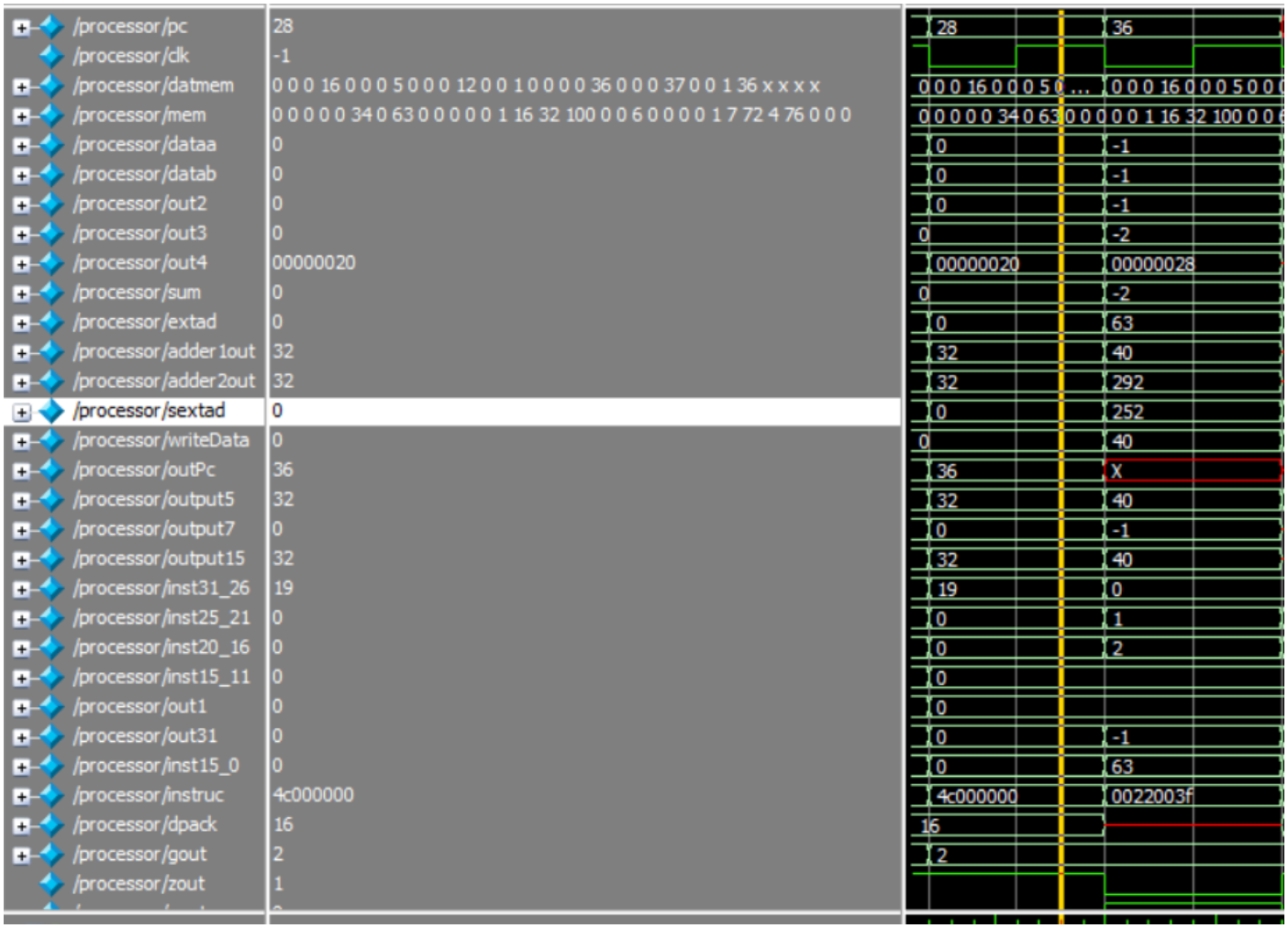
[shift $r7 to left by $r8 and store result in $r9]

0x01074804

4.6.jspal

opcode	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

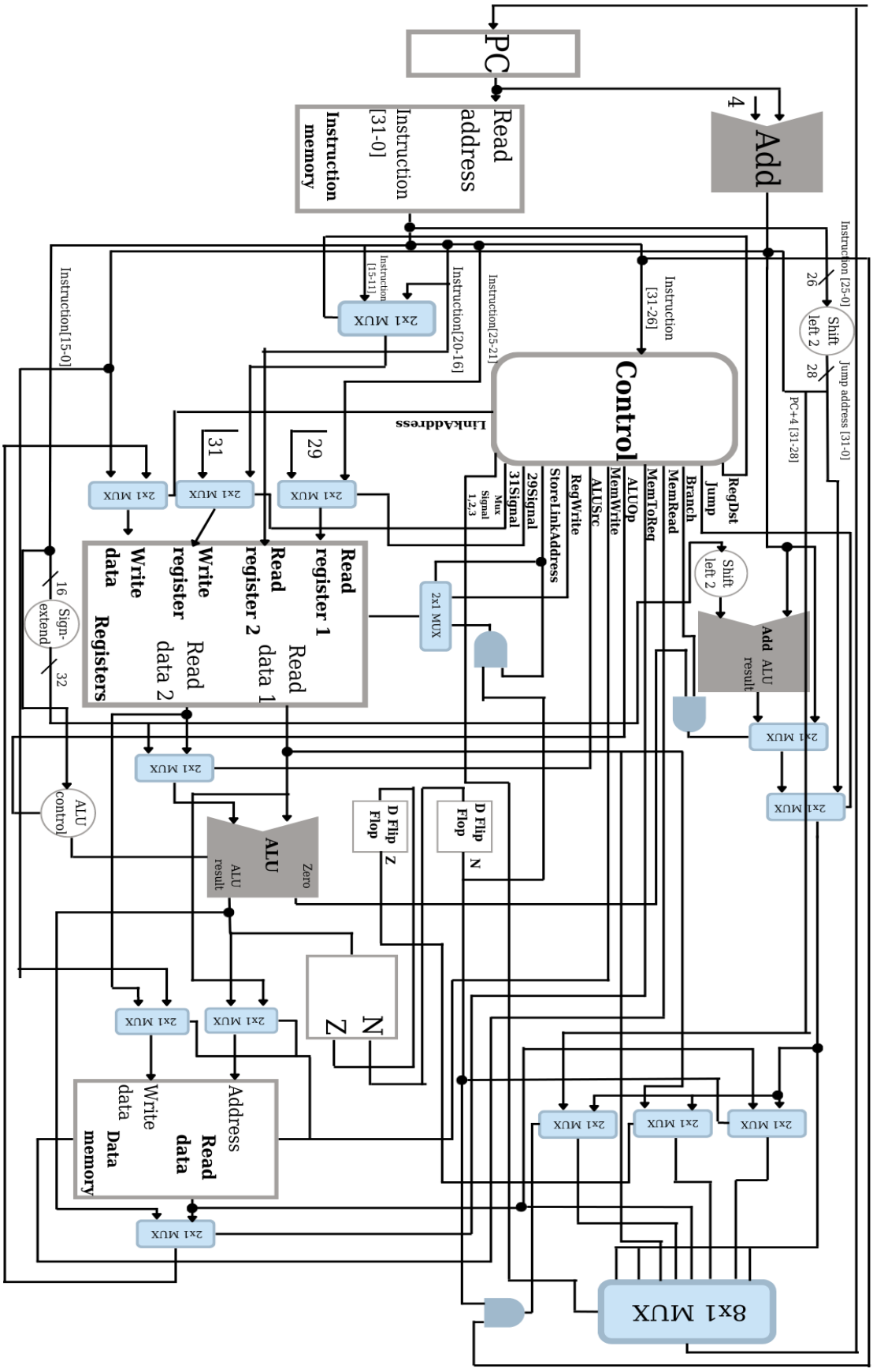
Jspal is an I-type instruction



$r 29 holds the value 16, and in data memory[16], it has the value 36. Thus, our new pc becomes 36 and link address (32) is stored in data memory[16].

Instruction: 010011 00000 00000 00000 00000 000000 0x4C000000 [branch to the address in mem[$r29] and link address to that place in memory]

5.Single Cycle Datapath:



6.Conclusion & Summary:

In Conclusion; Every requirement that is necessary to complete this assignment, including; tables for each signal, additional Verilog files, instructions that were used for this assignment and a single-cycle datapath has been outlined in this document.

7.References:

-Template for Single Cycle Datapath
https://i.stack.imgur.com/5d5XB.png

-Commented MIPS files from Lokman ALTIN

-https://www.chipverify.com/verilog/