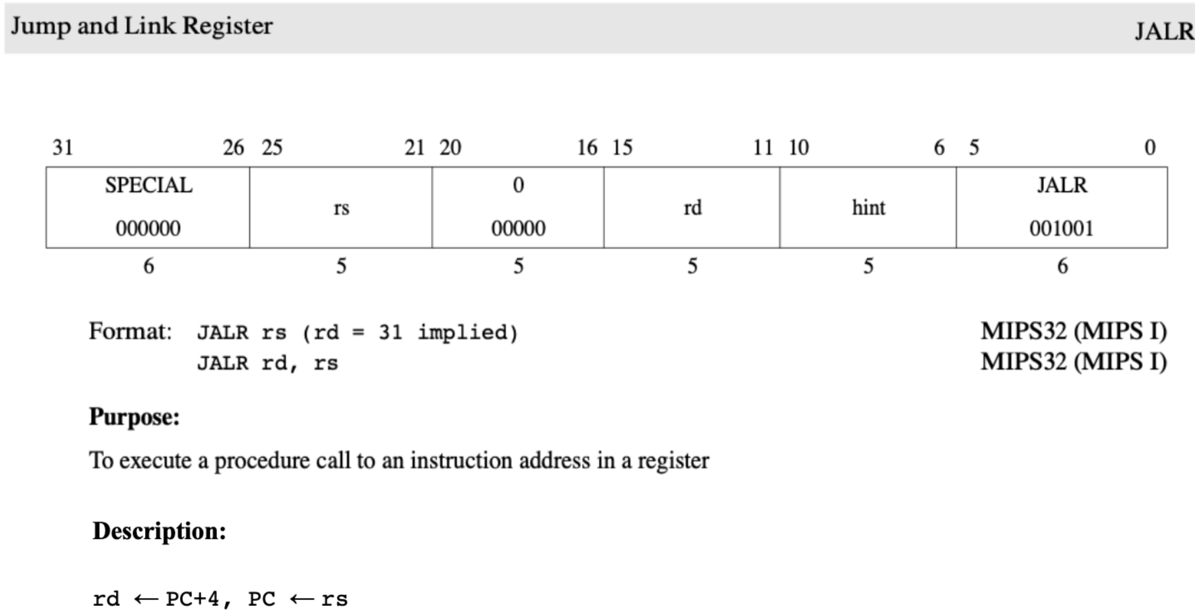
COMP30080 Assignment 5

Assignment 5

by Ruben van Breda

19200704

Modifying the MIPS Single Clock Circuit to implement a new instruction



Jump and Link Register JALR

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	0 00000	rd	hint	JALR 001001
6	5	5	5	5	6

Format: `JALR rs (rd = 31 implied)` MIPS32 (MIPS I)
`JALR rd, rs` MIPS32 (MIPS I)

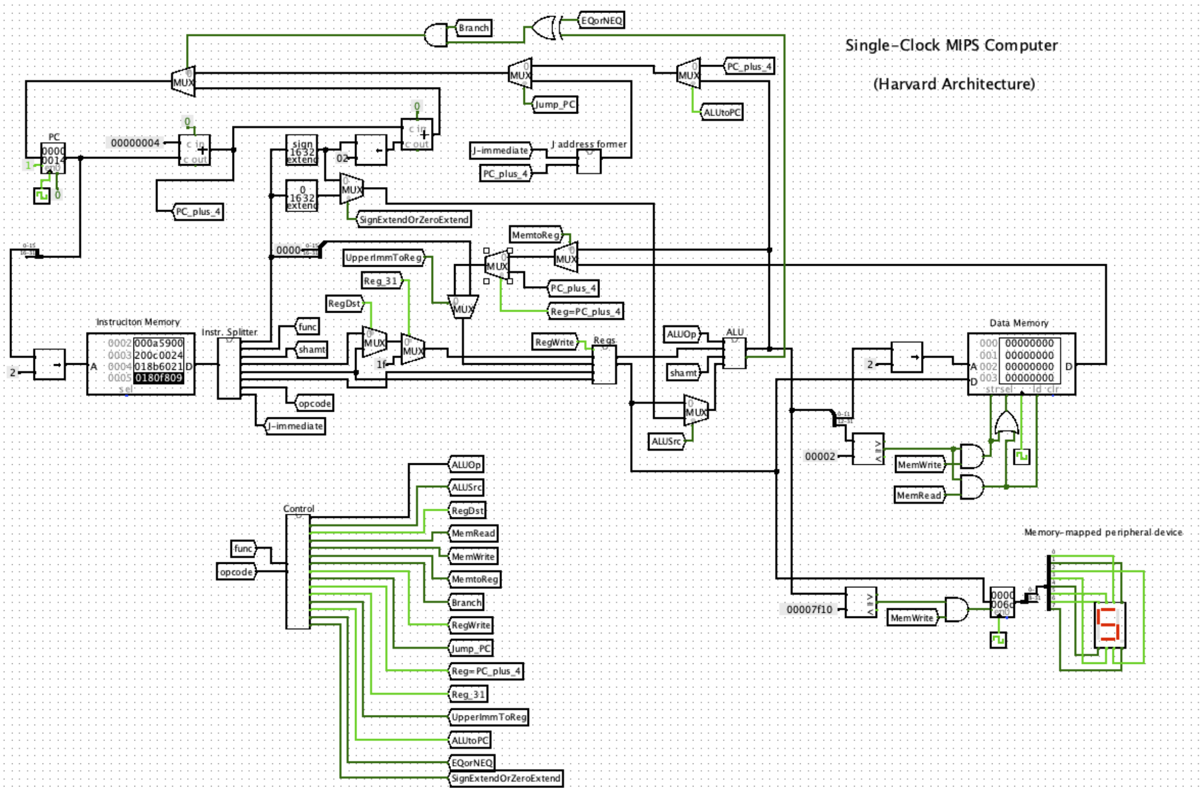
Purpose:

To execute a procedure call to an instruction address in a register

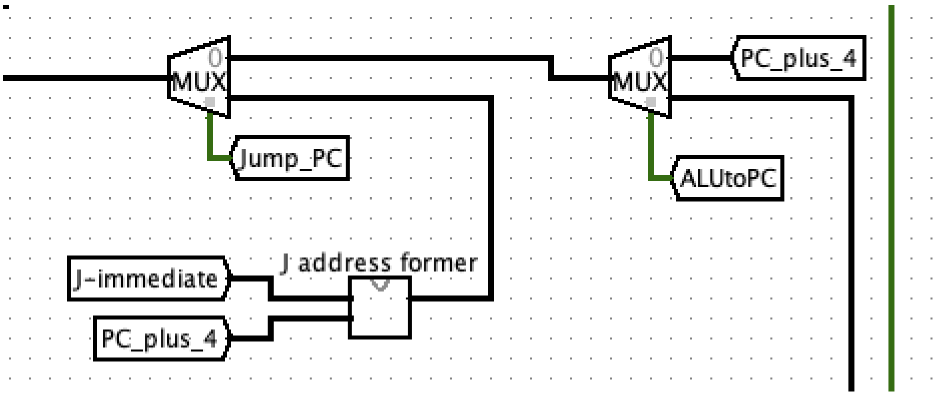
Description:

`rd ← PC+4, PC ← rs`

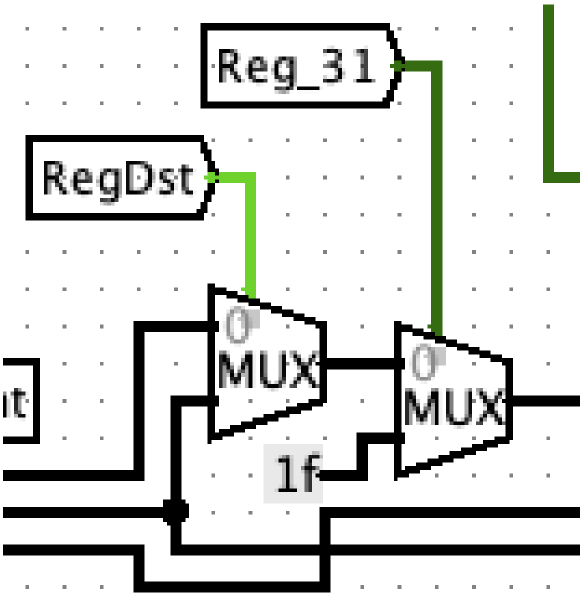
Main



Circuit to calculate new value of program counter

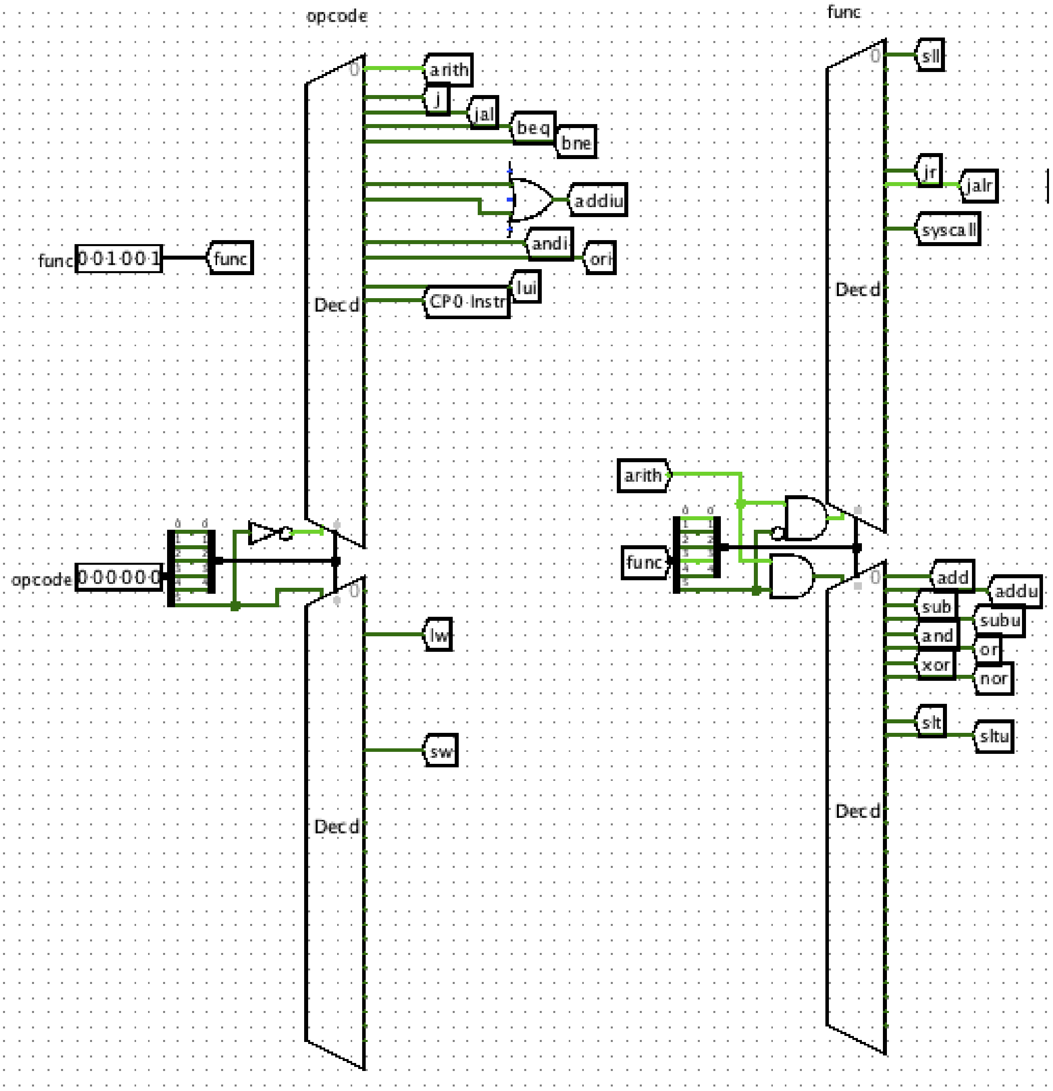


Expanded our address circuit so we can supply the `$ra` (*address 31*)



Control Unit

Decoding the Opcode and Function

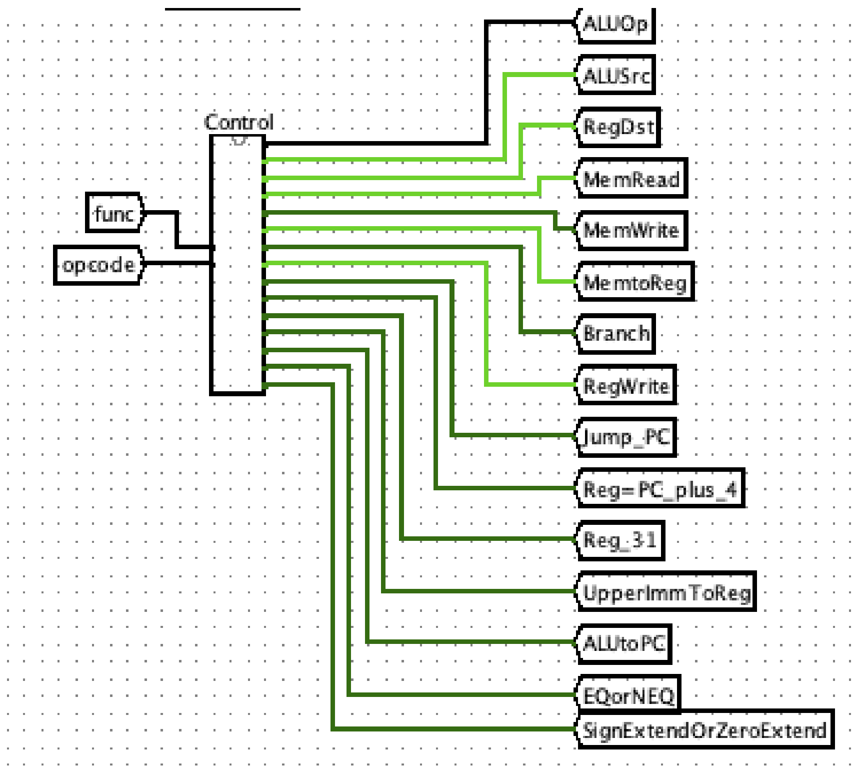


We need to add the opcode 00000 and the function 001001, adding a tunnel or connection for a $jalr$ label. Testing the circuit with text

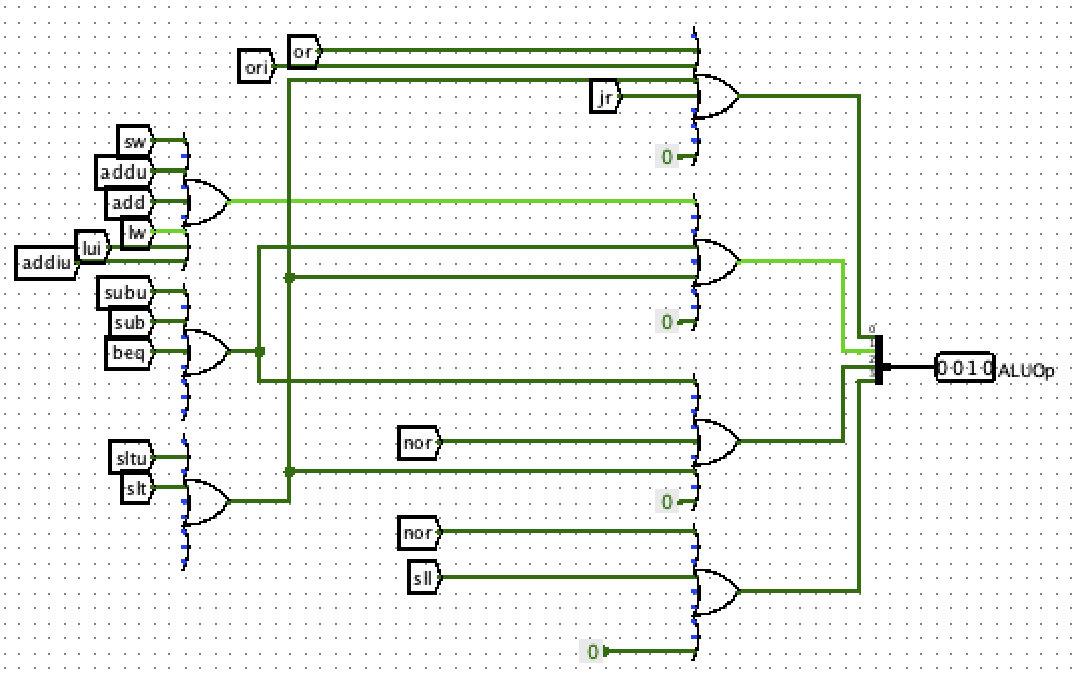
$$10001101011011000010000000000000$$

we can see that decoding correctly signals $jalr$

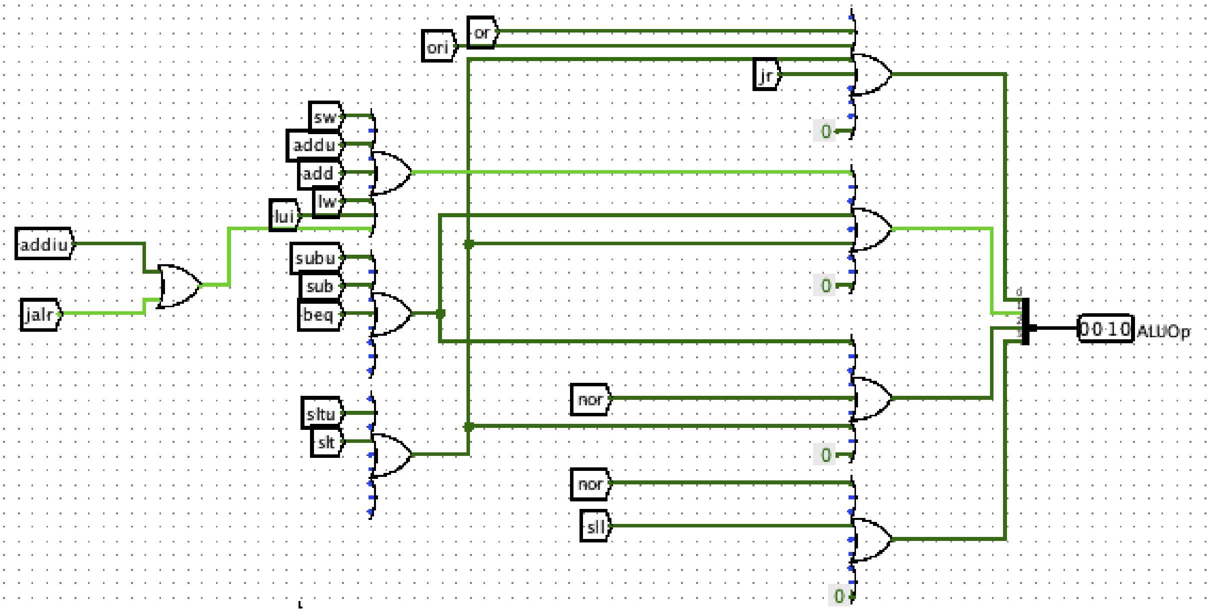
When implementing the JALR or any instruction we need to consider what values these control signals should have



Setting the ALUOp



adding



So that OPCODE is not 0000

For a JALR MIPS instruction we need to consider

Does it need memory access?

does it branch?

does it need to JUMP?

We need to have need to write $ra into Rd

⇒ RegDst = 1

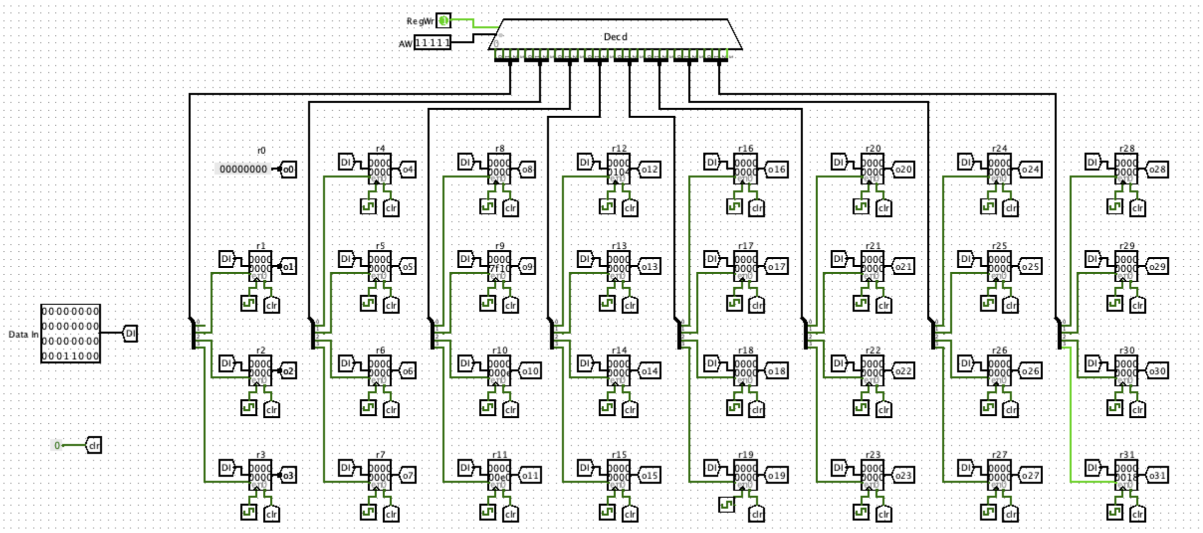
Also the $ra is implied to be register 31 so will need to set

⇒ Reg_31 = 1

We need to write the contents to register 31

⇒ RegWrite = 1

Doing so we can test and see that the register does indeed get written too.



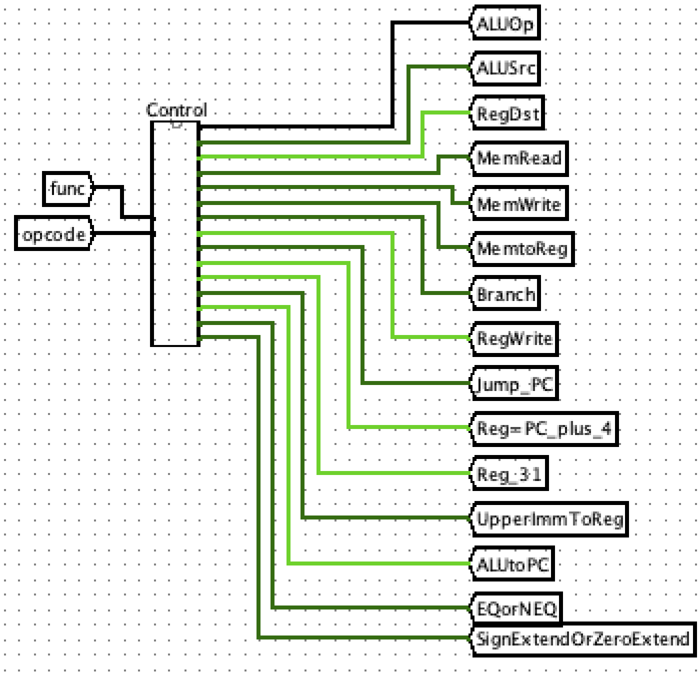
We also need to increase the PC + 4

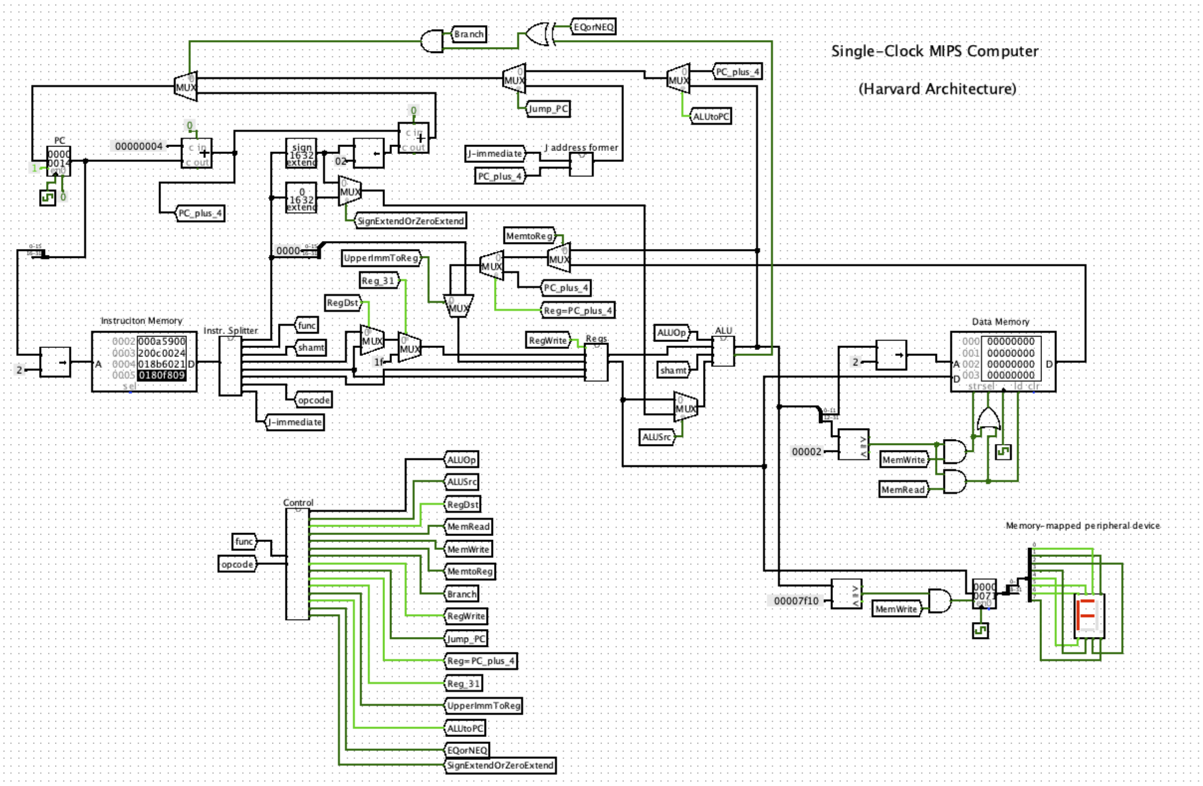
⇒ PC_plus_4 = 1

Lastly wee need to ensure the MUX outputs the ALU content so

⇒ ALUtoPC = 1

Control Signals for JALR




Single-Clock MIPS Computer
(Harvard Architecture)
Branch
EQorNEQ
MUX
PC_plus_4
Jump_PC
ALUtoPC
PC
00000004
c in
c out
sign
16 32
extend
02
J-immediate
J address former
PC_plus_4
SignExtendOrZeroExtend
MemtoReg
UpperImmToReg
0000
Reg_31
RegDst
Reg=PC_plus_4
func
shamt
RegWrite
Regs
ALUOp
ALU
ALUSrc
Instruciton Memory
Instr. Splitter
0002 000a5900
0003 200c0024
0004 018b6021
0005 0180f809
sel
opcode
J-immediate
1f
2
Data Memory
000 00000000
001 00000000
002 00000000
003 00000000
str sel ld clr
00002
MemWrite
MemRead
Memory-mapped peripheral device
00007f10
0000
0073
Control
ALUOp
ALUSrc
RegDst
MemRead
MemWrite
MemtoReg
Branch
RegWrite
Jump_PC
Reg=PC_plus_4
Reg_31
UpperImmToReg
ALUtoPC
EQorNEQ
SignExtendOrZeroExtend