MULTICYCLE_CPU

DATAPATH

Our cpu works in 5 clocks : 1) instruction fetch

2)instruction decode

3)execute

4)memory

5)write back

We have a unified memory, a register file , alu, and control 6 multiplexers , sign_extend and shift and transfer registers : 1)pc 2)BA 3) Instruction reg 4) D0 (that keeps amount of pc for jump 5)D1 : read_data1 from register file 6) D2 : read_reg2, 7) D3 : for shifted_signextended that might go to ALU unit ;

Continue next slide …

8) SR : which is next tu ALU and one of it’s outputs(it initializes inside of ALU unit)

9) low: which is ALU out put (I optimized it) and also keeps LSB output of mutiply

10) Hi : which keeps the MSB output of multiply

11) A0 : is memory data register (for write data in register in SW)

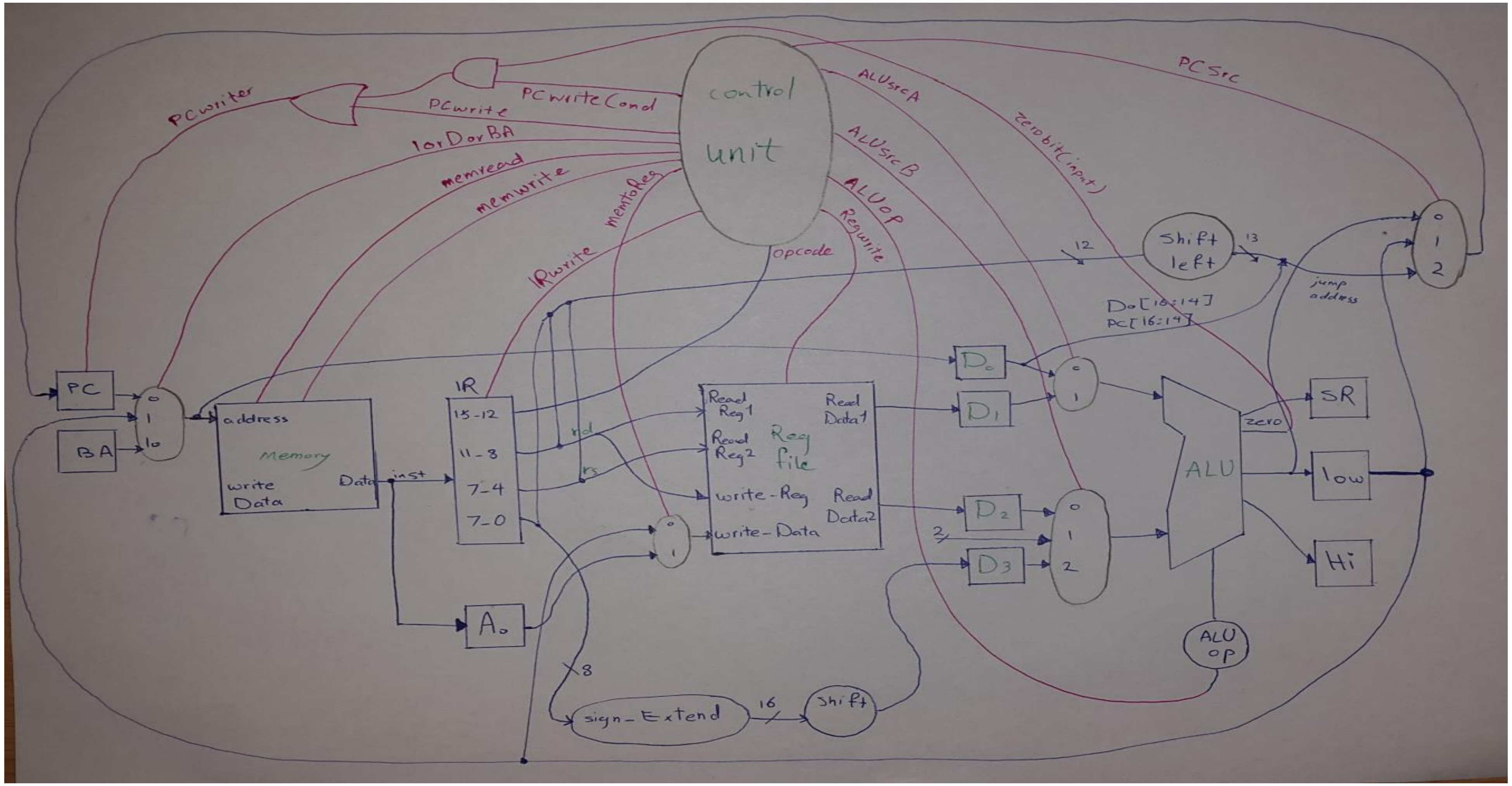
NOTE : I asked if I could omit A1, A2 and A3 registers because I don’t need them and you said it was ok!

NOTE : for multiply we wrote another entity that calculates the result and we port mapped it in ALU entity . (it converts those signed values to integers , multiplies them and then converts them back into std_logic_vector then divide them to two 16 bit outputs which goes to high and low registers then.

NOTE: we assume that register ZERO is the first register of register_file and it is always zero;

CONTROL UNIT

Our control unit is shown in this figure and the control unit is a state machine with 9 states (I put them in next slides);



Pc has an enable signal which is initialized in control unit that says to write pc or not

NOTE = our memory is initialized with 2*12-1 16 zero bits if you want to read an actual data you should initialize it

In first clock pc<= pc+2 and address goes to memory to be read from

Then in main cpu architecture data goes to instruction register

And IR decodes the instruction

NOTE that in instruction fetch D0 ‘s input is initialized (it’s PC)

NOTE that we assume jump address is IR[11 downto 0] & D0[15 downto 13] which goes to mux 4 to 2… (its in our data path picture)

NOTE : signal controls in each clock orders the cycle what to do

For example in EX signals determines that what are inputs and outputs

NOTE: inside of each part of our code we just initialized important inputs and outputs with actual signals we left the unimportant (don't care) signals open or connected them to ground signal

NOTE: we used multiplexers inside of IF / ID / EX , etc;

NOTE: in register file write reg and read_Reg1(rd) are the same (as you mentioned in your data types);

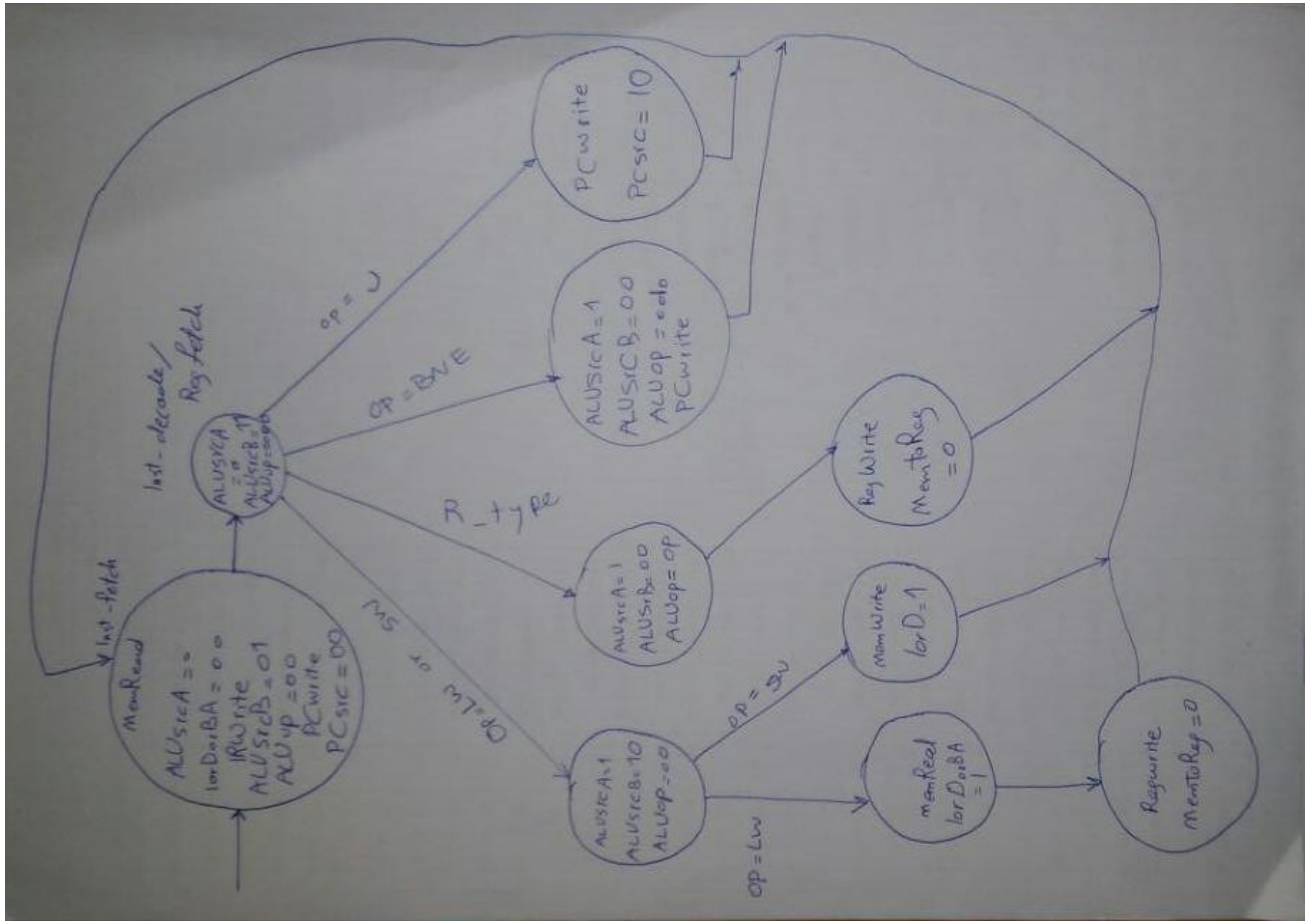
NOTE : ALU_OP signal is the actual op_code (add, sub , etc..) which is needed in ALU at that clock

NOTE: HI reg is zero when we have sum , subtract, and , or … (just not zero for multiply)

NOTE: both rs register and zero signal contain the zero signal . We used zero signal to simplify our work

NOTE: every time we use ALU unit we give it 4 inputs[15 downto 0] inside of ALU it decides which two to work with(with 2 multiplexers)

CONTROL STATE


Inst-fetch
MemRead
ALUSrcA = 0
IorD = 00
IRWrite
ALUSrcB = 01
ALUOp = 00
PCwrite
PCsrc = 00
Inst-decode/ Reg fetch
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
op = J
PCwrite
PCsrc = 10
op = BNE
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCwrite
R_type
ALUSrcA = 1
ALUSrcB = 00
ALUop = OP
RegWrite
MemToReg = 0
op = LW or SW
ALUSrcA = 1
ALUSrcB = 10
ALUop = 00
op = SW
MemWrite
IorD = 1
op = LW
memRead
IorD = 1
Regwrite
MemtoReg = 0