1. In lab3.2, update Uop?
2. Where are the other functions?
3. Defalt of the MUX?
4. The MARMUX: 1. Is it realy the input of PC? 1. Why is there another PC-4 (lab3.1) 2. In lab3.2, no Lshift? …
5. Just use bus in lab3.2?

// // cal pc\_mux\_1\_value

// /// cal mar\_max\_value

// //// cal mar\_mux\_1\_value

// int mar\_mux\_1\_value = sext\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 12), 20);

// //// cal mar\_mux\_0\_value

// ///// addr1mux\_value

// int addr1\_control\_value = get\_ADDR1MUX(CURRENT\_LATCHES.MICROINSTRUCTION);

// int pc = CURRENT\_LATCHES.PC;

// int rs1 = rs1\_en(get\_RS1En(CURRENT\_LATCHES.MICROINSTRUCTION), CURRENT\_LATCHES.REGS[mask\_val(CURRENT\_LATCHES.IR, 19, 15)]);

// int B\_imm = sext\_unit(b\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 7, 7), mask\_val(CURRENT\_LATCHES.IR, 11, 8), mask\_val(CURRENT\_LATCHES.IR, 30, 25), mask\_val(CURRENT\_LATCHES.IR, 31, 31)), 12);

// int addr1mux\_value = addr1\_mux(addr1\_control\_value, 0, pc, rs1, B\_imm);

// ///// addr2mux\_value

// int addr2\_control\_value = get\_ADDR2MUX(CURRENT\_LATCHES.MICROINSTRUCTION);

// int I\_imm = sext\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 20), 12);

// int S\_imm = sext\_unit(s\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 11, 7), mask\_val(CURRENT\_LATCHES.IR, 31, 25)), 12);

// int J\_imm = sext\_unit(j\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 31), mask\_val(CURRENT\_LATCHES.IR, 30, 21), mask\_val(CURRENT\_LATCHES.IR, 20, 20), mask\_val(CURRENT\_LATCHES.IR, 19, 12)), 20);

// int addr2mux\_value = addr2\_mux(addr2\_control\_value, 0, I\_imm, S\_imm, J\_imm);

// //// mar\_mux\_0\_value

// int mar\_mux\_0\_value = addr1mux\_value + addr2mux\_value;

// /// cal marmux

// int mar\_mux\_value = mar\_mux(get\_MARMUX(CURRENT\_LATCHES.MICROINSTRUCTION), mar\_mux\_0\_value, mar\_mux\_1\_value);

// pc\_mux\_1\_value

// cal pc\_mux\_0\_value

int pc\_mux\_0\_value = CURRENT\_LATCHES.PC + 4;

// // cal pc\_mux\_1\_value

// /// cal mar\_max\_value

// //// cal mar\_mux\_1\_value

// int mar\_mux\_1\_value = sext\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 12), 20);

// //// cal mar\_mux\_0\_value

// ///// addr1mux\_value

// int addr1\_control\_value = get\_ADDR1MUX(CURRENT\_LATCHES.MICROINSTRUCTION);

// int pc = CURRENT\_LATCHES.PC;

// int rs1 = rs1\_en(get\_RS1En(CURRENT\_LATCHES.MICROINSTRUCTION), CURRENT\_LATCHES.REGS[mask\_val(CURRENT\_LATCHES.IR, 19, 15)]);

// int B\_imm = sext\_unit(b\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 7, 7), mask\_val(CURRENT\_LATCHES.IR, 11, 8), mask\_val(CURRENT\_LATCHES.IR, 30, 25), mask\_val(CURRENT\_LATCHES.IR, 31, 31)), 12);

// int addr1mux\_value = addr1\_mux(addr1\_control\_value, 0, pc, rs1, B\_imm);

// ///// addr2mux\_value

// int addr2\_control\_value = get\_ADDR2MUX(CURRENT\_LATCHES.MICROINSTRUCTION);

// int I\_imm = sext\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 20), 12);

// int S\_imm = sext\_unit(s\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 11, 7), mask\_val(CURRENT\_LATCHES.IR, 31, 25)), 12);

// int J\_imm = sext\_unit(j\_format\_imm\_gen\_unit(mask\_val(CURRENT\_LATCHES.IR, 31, 31), mask\_val(CURRENT\_LATCHES.IR, 30, 21), mask\_val(CURRENT\_LATCHES.IR, 20, 20), mask\_val(CURRENT\_LATCHES.IR, 19, 12)), 20);

// int addr2mux\_value = addr2\_mux(addr2\_control\_value, 0, I\_imm, S\_imm, J\_imm);

// //// mar\_mux\_0\_value

// int mar\_mux\_0\_value = addr1mux\_value + addr2mux\_value;

// /// cal marmux

// int mar\_mux\_value = mar\_mux(get\_MARMUX(CURRENT\_LATCHES.MICROINSTRUCTION), mar\_mux\_0\_value, mar\_mux\_1\_value);

// pc\_mux\_1\_value

int pc\_minus\_4 = CURRENT\_LATCHES.PC - 4;

// int pc\_mux\_1\_value = mar\_mux\_value + pc\_minus\_4;

int pc\_mux\_1\_value = BUS + pc\_minus\_4;

// calculate the final value

NEXT\_LATCHES.PC = pc\_mux(get\_PCMUX(CURRENT\_LATCHES.MICROINSTRUCTION), pc\_mux\_0\_value, pc\_mux\_1\_value);

Next we should consider the BUS value. Since the bus value should be some address offset, it is evident that it must come from the MARMUX. And since the relative address branching cannot come from the instructions that has 20 bits immediate value, when the pc update is performed, the MARMUX control value should always be 0. So theoretically we don’t need to implement the MARMUX control and can directly put the addr1mux\_value + addr2mux\_value on the bus. But in order to fully simulate the hardware, I still implemented the MARMUX multiplexer.

The LShift20 is chosen when MARMUX is 1, so I call it mar\_mux\_1\_value. Since it is the 31..12 bits of the instruction, it can be retrieved by mask\_val(CURRENT\_LATCHES.IR, 31, 12), with a sign extension.

The addr1mux\_value + addr2mux\_value is chosen when MARMUX is 0, so I call it mar\_mux\_0\_value.

Let’s first consider the multiplexer of addr1mux\_value: the control signal is ADDR1MUX, which can be gotten by the MACRO get\_ADDR1MUX. The choice when ADDR1MUX is 0 is constant 0, the choice when ADDR1MUX is 1 is current PC value. The choice when ADDR1MUX is 2 is current rs1, rs1 can be gotten from the function rs1\_en, which returns the value of rs1 when the output of rs1 is enabled and returns 0 when the output is disabled. The choice when ADDR1MUX is 3 is the B immediate value, which can be generated by then function b\_format\_imm\_gen\_unit, followed by a sign extension.Then parameters addr1\_control\_value, 0, pc, rs1, B\_imm are fed to function addr1\_mux to generate addr1mux\_value.

Then consider the multiplexer of addr1mux\_value: the control signal is ADDR2MUX, which can be gotten by the MACRO get\_ADDR2.MUX. The choice when ADDR1MUX is 0 is constant 0. The choice when ADDR1MUX is 1 is the I immediate value, which can be retrieved by mask\_val from the 31..20 bits from the IR, followed by a sign extension. The choice when ADDR1MUX is 2 is the S immediate value, which can be generated by then function s\_format\_imm\_gen\_unit, followed by a sign extension. The choice when ADDR1MUX is 3 is the J immediate value, which can be generated by then function j\_format\_imm\_gen\_unit, followed by a sign extension. Then parameters addr2\_control\_value, 0, I\_imm, S\_imm, J\_imm are fed to function addr2\_mux to generate addr2mux\_value.

Then the