4.7.1

Sign extend will take the last 16 bits and sign extend it into 32 bits, so in this question, it is

0000 0000 0000 0000 0000 0000 0001 0100

Jump and shift will take the last 26 bits and shift it left 2 bits, in this question, it is

0001 1000 1000 0000 0000 0101 0000

4.7.2

From the first 6 bits we know that this is an I-type store word operation. So the ALU OP is 00, function field is don't cares, and since we want the add operation, ALU control input is 0010.

4.7.3

Store word simply cause PC to become PC+4. The path is

PC → PC+4 (ALU) → (brach controlled) MUX choose 0 → (jump controlled) MUX choose 0 → PC

4.7.4

Write register mux: since we are not writing to the register, the control signal of this mux is don't care. So the output can be either of the two inputs: bits[ 20-16] or bits [15-11] , namely, 2 or 0.

ALUSRC Mux: since we want the sign extended input, control signal will be 1, and the output will be sign extended bits[16-0], which is 0000 0000 0000 0000 0000 0000 0001 0100 = 20

MemtoReg Mux: since we are not writing from memory to register, the control signal will be a don't care. If the signal is 0, the output will be a direct addition of sign extended immediate and read data 1. Since bits [25-21] is 00011, which is a 3. Read data 1 will be value in r3, which is a -3. So the output will be 20-3=17. If the signal is 1, we will read address 17 in data memory. This value can be 0 if all values in data memory is 0 initially and read data happens before memory write.

Branch Mux: the control signal will be 0 since we do not branch, and the output will be PC+4.

Jump Mux: the control signal will be 0 since we do not jump, and the output will be PC+4.

4.7.5

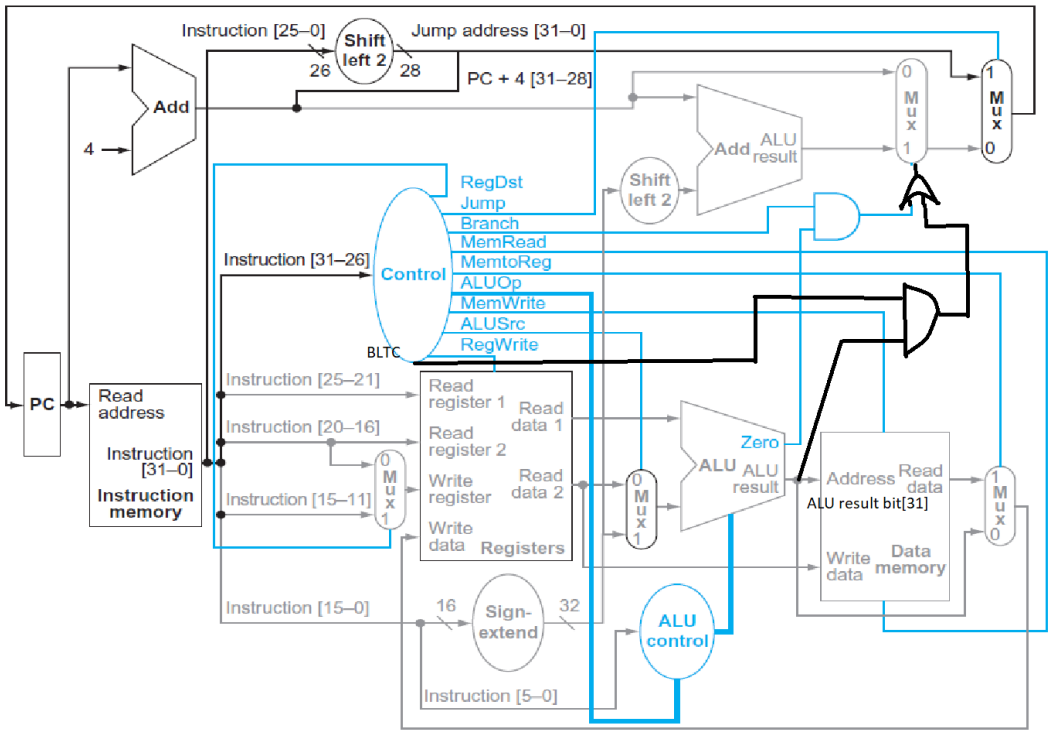
The ALU gets input from r[rs] and sign extended immediate, which is -3 and 20.
The PC+4 add unit gets input from PC and 4.
The add unit for branch operation will take inputs from PC+4 and SES(immediate). Sign extended and shifted 16 bits immediate, in this case, will be 20 *4 =80.

4.7.6

Read register one will be bits[ 25-21] =3
Read register two will be bits [20-16]=2
Write register will be either bits [20-16] or bits [15-11], namely, 2 or 0.
Write data will be 17 or value stored in data memory at address 17.

RegWrite is 0 since we are not writing to the register.

BLT:




if (R[rs] < R[rt])

PC = PC + 4 + SE(I)

else

PC = PC + 4

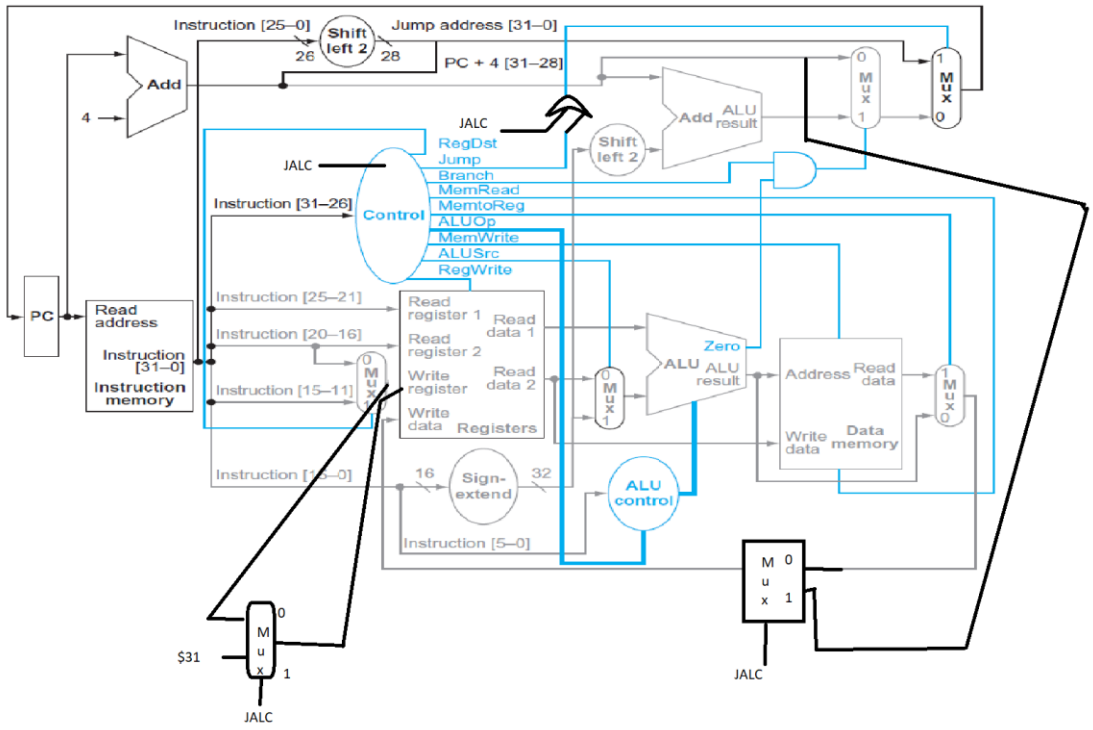
We take the most significant bit from ALU result, namely, bit 31, and use it as a control signal to AND it with BLTC (the BLT control signal). Then we OR this signal with the original control signal of the branch Mux. Note that the ELSE part is already built in.

The control signals:

| | |
|---|---|
| REGDST | X |
| ALUSRC | 0 |
| ALUOP | 01 |
| MEMTOREG | X |
| REGWRITE | 0 |
| MEMWRITE | 0 |
| MEMREAD | 0 |
| BRANCH | 0 |
| JUMP | 0 |
| BLTC | 1 |

BLTC is 0 for all other operations.

JAL:




R[$r31] = PC+4

PC = [31..28](PC+4) | [27..0] (I<<2)

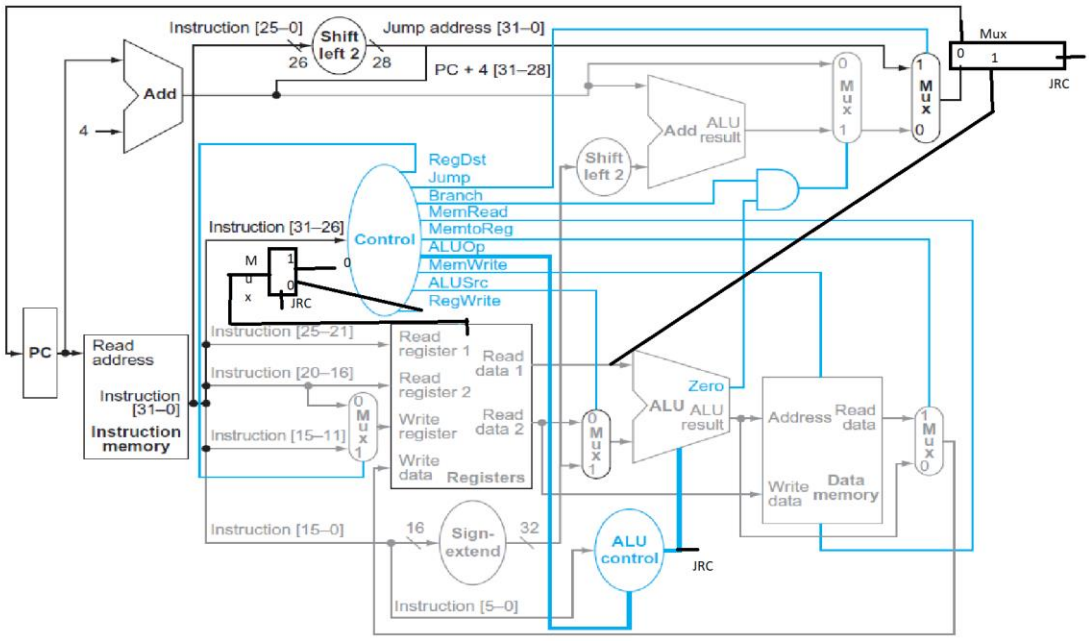
We mux PC+4 and the original write data, and use JALC as control signal. We take the original output from the write register mux and mux it with $31, and use JALC as control signal. The result of this Mux goes into write register. We OR JALC and JUMP signal, and wire the result into the original JUMP Mux.

The control signals:

| | |
|---|---|
| REGDST | X |
| ALUSRC | X |
| ALUOP | XX |
| MEMTOREG | X |
| REGWRITE | 1 |
| MEMWRITE | 0 |
| MEMREAD | 0 |
| BRANCH | X |
| JUMP | 0 |
| JALC | 1 |

JALC is 0 for all other operations.

JR: R-type




PC = R[rs]

We will make the ALU control generate a JRC signal (can be done by manipulating the bits of ALUOp and Funct field). We take read data 1 and mux it with the original input to PC, and use JRC as control signal. We take the original RegWrite and mux it with 0, use JRC as control signal, and wire the output of the mux to RegWrite.

ALU controller is modified as below:

| Instruction opcode | ALUOp | Instruction operation | Funct field | Desired ALU action | ALU control input | JRC (produced by some logic of ALUOp and Funct field) |
|---|---|---|---|---|---|---|
| LW | 00 | load word | XXXXXX | add | 0010 | 0 |
| SW | 00 | store word | XXXXXX | add | 0010 | 0 |
| Branch equal | 01 | branch equal | XXXXXX | subtract | 0110 | 0 |
| R-type | 10 | add | 100000 | add | 0010 | 0 |
| R-type | 10 | subtract | 100010 | subtract | 0110 | 0 |
| R-type | 10 | AND | 100100 | AND | 0000 | 0 |
| R-type | 10 | OR | 100101 | OR | 0001 | 0 |
| R-type | 10 | set on less than | 101010 | set on less than | 0111 | 0 |
| R-TYPE | 10 | JR | JR( 6 bit specific to JR) | X (don't care) | X(don't care) | 1 |