Review Problem 21

- Immediate vals for some instructions are sign-extended, while others are not. Build a 16bit to 64bit sign-extend unit that can handle both.
Branch RTL

Branch Instruction: B BrAddr26

\[ J_{adr} = \text{MEM}[\text{PC} + 1] \]

\[ \text{PC} = \text{PC} + 4 \times \text{SEC(BrAddr26)} \]
Datapath + Branch

\[ PC = PC + 4 \times SE (Br \ Add \ 26) \]
Conditional Branch RTL

Conditional Branch Instruction: CBZ Rd, CondAddr19

Instr = Mem [PC]
Cond = (Reg[Rd] == 0)
if (Cond) PC = PC + SEC(CondAddr19) < 2; /* SEC(Addr) */
else PC = PC + 4;

<table>
<thead>
<tr>
<th>Opcode</th>
<th>CondAddr19</th>
<th>Rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Datapath + Conditional Branch

Diagram showing the flow of data through a datapath with conditional branch logic, including instructions, memory, and registers.
Control

Identify control points for pieces of datapath
  Instruction Fetch Unit
  ALU
  Memories
  Datapath muxes
  Etc.

Use RTL for determine per-instruction control assignments
Complete Datapath
## Control Signals

<table>
<thead>
<tr>
<th>Opcode[31:26]</th>
<th>100010</th>
<th>110000</th>
<th>111110</th>
<th>111110</th>
<th>000101</th>
<th>101101</th>
</tr>
</thead>
<tbody>
<tr>
<td>11000</td>
<td>ADD</td>
<td>SUB</td>
<td>LDUR</td>
<td>STUR</td>
<td>B</td>
<td>CBZ</td>
</tr>
<tr>
<td>Reg2Loc</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>ALUSrc</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>MemToReg</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RegWrite</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MemWrite</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BrTaken</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>UncndBr</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ALUop</td>
<td>Add</td>
<td>Sub</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
<td>X</td>
</tr>
</tbody>
</table>

Uncnd Br = Instr[31]
ADD Control

Instruction = Mem[PC];
Reg[Rd] = Reg[Rn] + Reg[Rm];
PC = PC + 4;
SUB Control

Instruction = Mem[PC];
Reg[Rd] = Reg[Rn] - Reg[Rm];
PC = PC + 4;
LDUR Control

Instruction = Mem[PC];
Addr = Reg[Rn] + SignExtend(DAddr9);
Reg[Rd] = Mem[Addr];
PC = PC + 4;
STUR Control

Instruction = Mem[PC];
Addr = Reg[Rn] + SignExtend(DAddr9);
Mem[Addr] = Reg[Rd];
PC = PC + 4;
B Control

Instruction = Mem[PC];
PC = PC + SignExtend(BrAddr26) << 2;
CBZ Control

Instruction = Mem[PC];
Cond = (Reg[Rd] == 0);
if (Cond)
  PC = PC + SE(CondAddr19)<<2;
else
  PC = PC + 4;

---

[Diagram of CBZ Control flowchart]

---

94