Lecture 17:
ALU and Processor Design

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
Announcements

• Midterm 2 grades will be made available today

• One last component:
  Take home final (2% extra credit)
  – Released tomorrow, Thur at 10am, due Sat 3pm
  – Submissions via TED

• Academic Integrity
Arithmetic -- The heart of instruction execution
Designing an Arithmetic Logic Unit

- **ALU Control Lines (ALUop)**
  - 000: And
  - 001: Or
  - 010: Add
  - 110: Subtract
  - 111: Set-on-less-than
A One Bit ALU

- This 1-bit ALU will perform AND, OR, and ADD
A 32-bit ALU

1-bit ALU

32-bit ALU

Operation

CarryIn

Result

CarryOut

a

b

a0

b0

a1

b1

a2

b2

a31

b31

CarryIn

ALU0

CarryOut

Result0

CarryIn

ALU1

CarryOut

Result1

CarryIn

ALU2

CarryOut

Result2

CarryIn

ALU31

CarryOut

Result31
Subtract – We’d like to implement a means of doing A-B (subtract) but with only minor changes to our hardware. How?

1. Provide an option to use bitwise NOT A
2. Provide an option to use bitwise NOT B
3. Provide an option to use bitwise A XOR B
4. Provide an option to use 0 instead of the first CarryIn
5. Provide an option to use 1 instead of the first CarryIn

<table>
<thead>
<tr>
<th>Selection</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 alone</td>
</tr>
<tr>
<td>B</td>
<td>Both 1 and 2</td>
</tr>
<tr>
<td>C</td>
<td>Both 3 and 4</td>
</tr>
<tr>
<td>D</td>
<td>Both 2 and 5</td>
</tr>
<tr>
<td>E</td>
<td>None of the above</td>
</tr>
</tbody>
</table>
Full 32-bit ALU

what signals accomplish ADD?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>CIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

sign bit (adder output from bit 31)
Full 32-bit ALU

what signals accomplish OR?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>CIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

sign bit (adder output from bit 31)
Full 32-bit ALU

Little more intense – can you get this?

what signals accomplish SUB?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>CIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

sign bit (adder output from bit 31)
Shifters

- **Logical shifter:** shifts value to left or right and fills empty spaces with 0’s
  - Ex: 11001 >> 2 = 00110
  - Ex: 11001 << 2 = 00100

- **Arithmetic shifter:** same as logical shifter, but on right shift, fills empty spaces with the old most significant bit
  - Ex: 11001 >>> 2 = 11110
  - Ex: 11001 <<< 2 = 00100

- **Rotator:** rotates bits in a circle, such that bits shifted off one end are shifted into the other end
  - Ex: 11001 ROR 2 = 01110
  - Ex: 11001 ROL 2 = 00111
General Shifter Design

A_{3:0} \rightarrow 4 \rightarrow Y_{3:0} \rightarrow A_{3:0}

shamt_{1:0}

2

A_{3:0} \rightarrow \rightarrow \rightarrow Y_{3:0}

S_{1:0}

00 01 10 11

Y_{3}

S_{1:0}

00 01 10 11

Y_{2}

S_{1:0}

00 01 10 11

Y_{1}

S_{1:0}

00 01 10 11

Y_{0}
R-type Instruction: reg-reg ALU ops (e.g. add, and)

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>RS</th>
<th>RT</th>
<th>RD</th>
<th>shamt</th>
<th>FUNCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>25-21</td>
<td>20-16</td>
<td>15-11</td>
<td>10-6</td>
<td>5-0</td>
</tr>
</tbody>
</table>

**ADD $S1, $S2, $S3**

**ADD RD, RS, RT**

Source Register 1 (attached to “Read Register 1” input)

Source Register 2 (attached to “Read Register 2” input)

Shift amount (for sll, srl etc.)

Destination Register (attached to “Write Register” input)
Step 1 (R-type): Fetch instruction and advance PC
Step 2 (R-type): Read two registers and set control signals
Step 3 (R-type): Perform the ALU operation
Step 4 (R-type): Write result to register
Have a great summer and good luck for future courses!