Arithmetic Logic Unit (ALU)

F_{2:0} | Function
--- | ---
000  | A & B
001  | A | B
010  | A + B
011  | not used
100  | A & ~B
101  | A | ~B
110  | A - B
111  | SLT
## ALU Design

<table>
<thead>
<tr>
<th>$F_{2:0}$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>$A &amp; B$</td>
</tr>
<tr>
<td>001</td>
<td>$A \mid B$</td>
</tr>
<tr>
<td>010</td>
<td>$A + B$</td>
</tr>
<tr>
<td>011</td>
<td>not used</td>
</tr>
<tr>
<td>100</td>
<td>$A &amp; \neg B$</td>
</tr>
<tr>
<td>101</td>
<td>$A \mid \neg B$</td>
</tr>
<tr>
<td>110</td>
<td>$A - B$</td>
</tr>
<tr>
<td>111</td>
<td>SLT</td>
</tr>
</tbody>
</table>
Set Less Than (SLT) Example

- Configure a 32-bit ALU for the set if less than (SLT) operation. Suppose $A = 25$ and $B = 32$.
  - $A$ is less than $B$, so we expect $Y$ to be the 32-bit representation of 1 ($0x00000001$).
  - For SLT, $F_{2:0} = 111$.
  - $F_2 = 1$ configures the adder unit as a subtracter. So $25 - 32 = -7$.
  - The two's complement representation of -7 has a 1 in the most significant bit, so $S_{31} = 1$.
  - With $F_{1:0} = 11$, the final multiplexer selects $Y = S_{31}$ (zero extended) = $0x00000001$. 
Shifters

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

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

- **Rotator**: rotates bits in a circle, such that bits shifted off one end are shifted into the other end
  - Ex: 11001 ROR 2 =
  - Ex: 11001 ROL 2 =
Shifter Design
Shifters as Multipliers and Dividers

- A left shift by $N$ bits multiplies a number by $2^N$
  - Ex: $00001 << 2 = 00100$ ($1 \times 2^2 = 4$)
  - Ex: $11101 << 2 = 10100$ ($-3 \times 2^2 = -12$)

- The arithmetic right shift by $N$ divides a number by $2^N$
  - Ex: $01000 >>> 2 = 00010$ ($8 \div 2^2 = 2$)
  - Ex: $10000 >>> 2 = 11100$ ($-16 \div 2^2 = -4$)