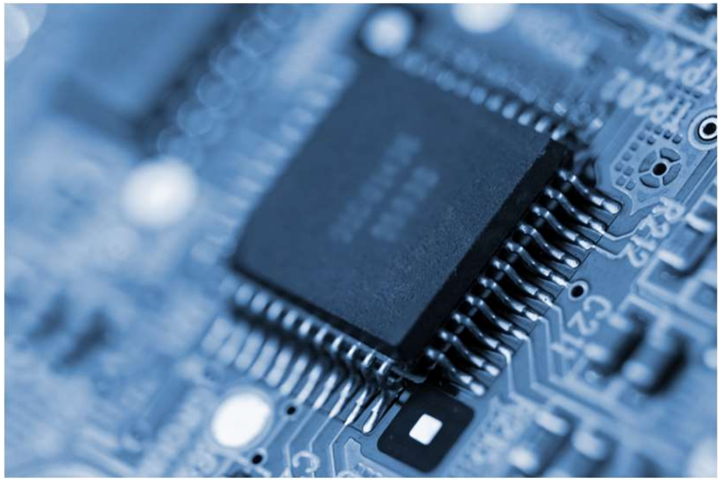
UNIVERSITY OF LEEDS

Computer Processors

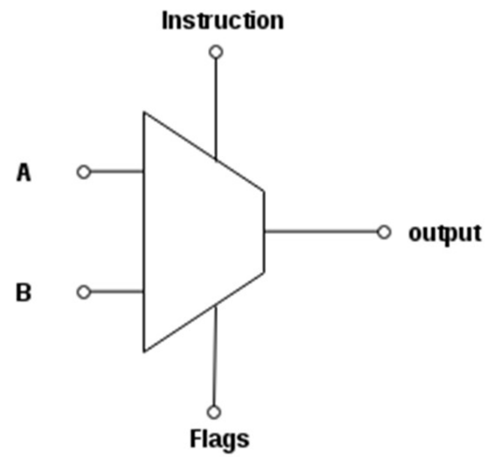
Arithmetic Logic Unit



Lecture 9

Arithmetic Logic Unit

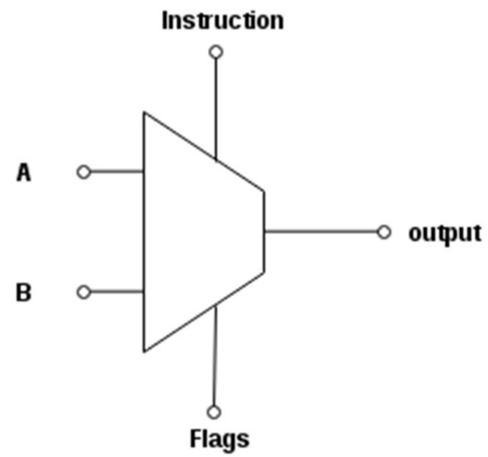
- The Arithmetic Logic Unit (ALU) is the centerpiece of any modern day computer
- The ALU implements a set of logic commands
- The ALU allows for the selection of which command should be output



Arithmetic Logic Unit

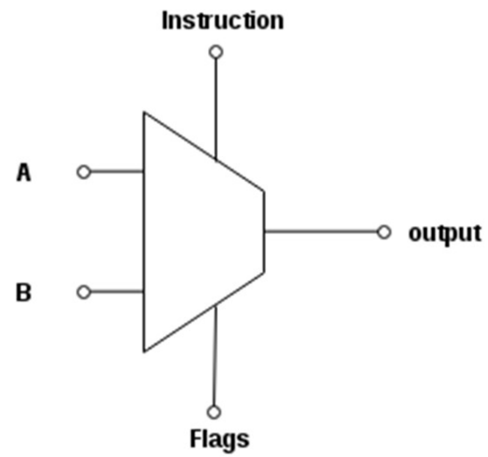
UNIVERSITY OF LEEDS

- ALU's implement a fixed set of functions
- Each function can take at most two multi-bit arguments
- The ALU outputs the result of the boolean function and a set of flags



Arithmetic Logic Unit

- The instruction is a set of control bits which select which function the ALU should output
  - And
  - Adder
  - Pre/post processing on inputs/outputs



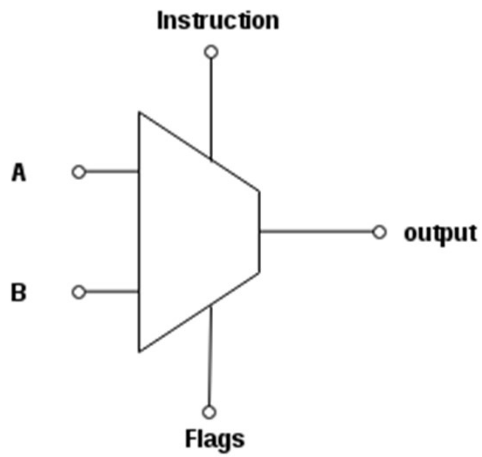
Arithmetic Logic Unit

Our specific ALU accepts

- Two 16-bit wide arguments
- A 6-bit wide instruction (control bits)

And outputs

- A 16-bit wide output
- A bit to indicate if the output is 0
- A bit to indicate if the output is less that 0



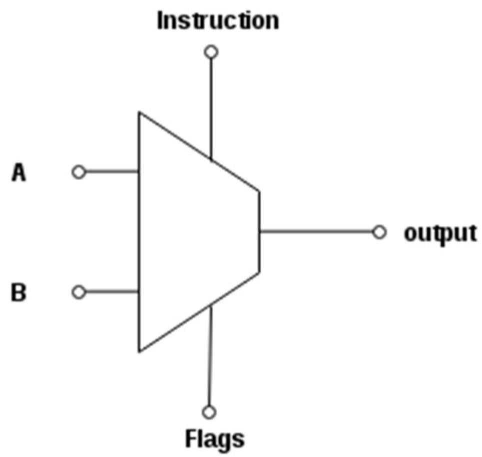
Arithmetic Logic Unit

UNIVERSITY OF LEEDS

- With a 6-bit wide instruction gives the ALU

  a possible 2^6 (=64) instructions

- We are only interested in 18 of them



Control bits

UNIVERSITY OF LEEDS

- The ALU has the choice between two basic functions logical **And** and **Addition**
- All operations are multi-bit operations
- All of the required components were covered in **Lecture 4 - Common Logic Configurations**

Bit	Description
za	Zero input **A**
na	Negate input **A**
zb	Zero input **B**
nb	Negate input **B**
f	Select between & and +
no	Negate output

Control bits

UNIVERSITY OF LEEDS

- Each of the 18 configurations of the control bits does something of interest
- Let us consider the control bits set to 001110
  - The ALU is instructed to compute a-1
  - Input **A** is neither zero'ed nor negated
  - Input **B** is zeroed and then negated (2's complement -1_{10})
  - f-bit set to **Addition**

za	na	zb	nb	f	no	output
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	a
1	1	0	0	0	0	b
0	0	1	1	0	1	$\neg a$
1	1	0	0	0	1	$\neg b$
0	0	1	1	1	1	$-a$
1	1	0	0	1	1	$-b$
0	1	1	1	1	1	$a+1$
1	1	0	1	1	1	$b+1$
0	0	1	1	1	0	$a-1$
1	1	0	0	1	0	$b-1$
0	0	0	0	1	0	$a+b$
0	1	0	0	1	1	$a-b$
0	0	0	1	1	1	$b-a$
0	0	0	0	0	0	$a\&b$
0	1	0	1	0	1	$a\|b$

	za	na	zb	nb	f	no	output
	0	0	1	1	1	0	a-1
a							
b							

	za	na	zb	nb	f	no	output
	0	0	1	1	1	1	-a
a							
b							

<table>
<tr><td></td><td>za</td><td>na</td><td>zb</td><td>nb</td><td>f</td><td>no</td><td>output</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>a-1</td></tr>
<tr><td>a</td><td>a</td><td>a</td><td>a</td><td>a</td><td rowspan="2">a-1</td><td rowspan="2">a-1</td><td rowspan="2">a-1</td></tr>
<tr><td>b</td><td>b</td><td>b</td><td>0</td><td>-1
(11111111)</td></tr>
</table>

<table>
<tr><td></td><td>za</td><td>na</td><td>zb</td><td>nb</td><td>f</td><td>no</td><td>output</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>-a</td></tr>
<tr><td>a</td><td>a</td><td>a</td><td>a</td><td>a</td><td rowspan="2">a-1</td><td rowspan="2">!(a-1)</td><td rowspan="2">-a</td></tr>
<tr><td>b</td><td>b</td><td>b</td><td>0</td><td>-1</td></tr>
</table>

Control bits

UNIVERSITY OF LEEDS

You should verify that the rest do what is claimed in the table

What do the other 46 control bit configurations do?

za	na	zb	nb	f	no	output
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	a
1	1	0	0	0	0	b
0	0	1	1	0	1	$\neg a$
1	1	0	0	0	1	$\neg b$
0	0	1	1	1	1	$-a$
1	1	0	0	1	1	$-b$
0	1	1	1	1	1	$a+1$
1	1	0	1	1	1	$b+1$
0	0	1	1	1	0	$a-1$
1	1	0	0	1	0	$b-1$
0	0	0	0	1	0	$a+b$
0	1	0	0	1	1	$a-b$
0	0	0	1	1	1	$b-a$
0	0	0	0	0	0	$a\&b$
0	1	0	1	0	1	$a\|b$

Implementation

UNIVERSITY OF LEEDS

Notice that the ALU is composed of

- A logic circuit that can negate or zero an input (times 2)
- Logic circuits for **Addition** and **And.**
- A logic circuit to select between them
- A logic circuit to negate the output
- A logic circuit to handle flags

Only the last one has not been implemented already

ALU

UNIVERSITY OF LEEDS

- The ALU we implement is quite basic
- No multiplication/division
- No floating point numbers
- No exponentiation (required for some encryptions)
- Modern ALU’s support many more operations

ALU most similar to a microprocessor or 1970’s ALU’s

Summary

UNIVERSITY OF LEEDS

- Introduced the ALU
- Discussed the implementation
- Compared it to current architectures

Interested in logic design? [Digital Logic Design]