Logical Operations

`&` compares every bit

- 0 and 0 = 0
- 0 and 1 = 0
- 1 and 0 = 0
- 1 and 1 = 1

0	**0**	**0**	**1**	**0**	**0**	**0**
0	0	0	1	0	1	1
-	-	-	-	-	-	-

`||` operates in the following way

- 0 or 0 = 0
- 0 or 1 = 1
- 1 or 0 = 1
- 1 or 1 = 1

`Xor` works in the following way

- 0 xor 0 = 0
- 0 xor 1 = 1
- 1 xor 0 = 1
- 1 xor 1 = 0

Xor being the negated version of or

Bitwise Logical Instructions

How to implement NOT using NOR?

- Using `$zero` as one of the input operands
- It is included in some implementations of MIPS as a pseudo instruction

```
ADD $target, $source1, $source2
```

The key letter `I` may be added in order to add an intermediate.
The key letter `U` may be used in order to make it unsigned

Memory Operands

- memory contains both data and instructions
- Memory can be viewed as a large array of bytes
- The address of a variable or instruction is its offset from the beginning of memory

`g = h + A[5]`

- lets say g and h are associated with the registers `$s1` and `$s2` respectively. Let's also say that the base address of A is associated with register `$s3`

```
lw $t0, 20($s3) #load the element at a 20 byte offset from $s3
add $s1, $s2, $t0
```