Translation of Bitwise Operations

Take an example of this C code. This code is inside some function.

```
int a = 4;
int b = 8;
int c = a | b;
int d = b & c;
```

Location of local variables of the stack (local variables are explained here (memorymanagement.html)).

```
a => -4(%ebp)
b => -8(%ebp)
c => -12(%ebp)
d => -16(%ebp)
```

The use of registers as temporary memory is described here (arithmeticop.html#tempVaribaleUsage)
Generated assembly code:

```
movl    $4, -4(%ebp)
movl    $8, -8(%ebp)
movl    -8(%ebp), %eax
movl    -4(%ebp), %edx
orl     %edx, %eax
movl    %eax, -12(%ebp)
movl    -12(%ebp), %eax
movl    -8(%ebp), %edx
andl    %edx, %eax
movl    %eax, -16(%ebp)
```

Comments on generated assembly code:

```
# a = 4
movl    $4, -4(%ebp)

# b = 8
movl    $8, -8(%ebp)

# tmp = b
movl    -8(%ebp), %eax

# tmp2 = a
movl    -4(%ebp), %edx

# tmp = tmp | tmp2
orl %edx, %eax

# c = tmp
movl    %eax, -12(%ebp)

# tmp = c
movl    -12(%ebp), %eax

# tmp2 = b
movl    -8(%ebp), %edx

# tmp = tmp & tmp2
andl    %edx, %eax

# d = tmp
movl    %eax, -16(%ebp)
```

Content of (local)variables are moved to registers then assembly instructions for bitwise operators (in the example above, andl and olr)are used then the value again stored back into the memory.

‹ Translation of Arithmetic Operations (/cin/arithmeticop.html)
up (/cin/cin.html)
Translation of Branch Statement › (/cin/branchstmt.html)