Classwork 2 – Logical operations

Basics

From the point of view of the programmer, the logical operations almost coincide with arithmetic operations. The fundamental difference is that logical operations do not create execution (other than shift operations) and you do not need to worry about negative numbers. Logical operations are called bitwise because they act on individual bits. Basic or fundamental logical operations are:

```
NOT     Invert bits
AND     Logical and
OR      Logical or
```

Derivative logical operations that can be expressed in terms of the fundamental operations are (not an exhaustive list):

```
XOR     Exclusive or
NAND    NOT AND (inverted and)
NOR     NOT OR (inverted or)
```

Shift operations can sometimes be grouped with logical operations and sometimes not. This is because these operations are not fundamental boolean operations, but operations on bits. The shift operation moves all bits in a word one or more positions left or right. Typical shift operations are:

```
LSL     logical shift left, left-most bit is shifted into carry bit
LSR     logical shift right, right-most bit is shifted into carry bit
ROL     rotate left, shift bits left, left-most bit into right-most position
ROR     rotate right, shift bits right, right-most bit into left-most position
```

The command system of some microprocessors includes other logical operations, which are optional and can be synthesized using other operations.

```
Bit Set     bit i of a word to 1.
Bit Clear   bit i of a word to 0.
Bit Toggle  complement bit i of a word
```

ARM logical operations

Few microprocessors implement all of the above logical operations. Some microprocessors implement special logic operations. Logical operations on ARM are:

```
MVN   MVN r0, r1               ; r1 gets inverted into r0
AND   AND r0, r1, r2           ; r1 and r2 into r0
ORR   OR  r0, r1, r2           ; r1 or r2 into r0
EOR   XOR r0, r1, r2           ; r1 xor r2 into r0
BIC   BIC r0, r1, r2           ; r1 and not r2 into r0
LSL   MOV r0, r1, LSL r2       ; r1 is shifted left by the number of places in r2
LSR   MOV r0, r1, LSR r2       ; r1 is shifted right by the number of places in r2
```

Two unusual commands are `MVN` (move with negation) and `BIC` (reset bit). A movement negation instruction acts like an instruction movement (`MOV`), except that the bits are inverted. Note that the bits in the source register remain unchanged. The `BIC` command clears the bits of the first operands when the bits of the destination operand are set. This operation is equivalent to AND between the first and negative second operand. For instance, an 8-bit `BIC r0, r1, r2` (with r1 = 00001111 and r2 = 11001010) will result in r0 = 11000000.