name:	Vince Rothenberg
course:	csci 10
assignment:	homework 5
prepared:	Tue, Feb 25, 2020 // 10:47 am

1. Provide a basic definition of logical operations.

Logical operations are performed bit by bit on the input operands (which may be values in registers or constants) and the result placed in a destination register. In computing, logical operations are necessary because they can be used to model the way that information flows through electrical circuits, such as the circuits inside a CPU. These types of operations are called boolean operations.

2. Describe the basic rule(s) of the AND instruction. What is the AND instruction useful for?

AND will perform a logical AND between the two operands, placing the result in the destination register; this is useful for masking the bits you wish to work on. Operand 1 is a register, operand 2 can be a register, shifted register, or an immediate value (which may be shifted).

Syntax:
AND<suffix> <dest>, <op 1>, <op 2>

Function:
dest = op_1 AND op_2

AND is useful for masking bitfields in order to extract only the part that is necessary. For example, if the lower two bits contain flags, ANDing with 3 will keep those flag bits and discard any other data.

3. Describe the basic rule(s) of the ORR instruction. What is the ORR instruction useful for?

ORR will perform a logical, bitwise, OR between the two operands, placing the result in the destination register; this is useful for setting certain bits. Operand 1 is a register, operand 2 can be a register, shifted register, or an immediate value (which may be shifted. If the S bit is set (ORRS), the N and Z flags are set according to the result, the C flag is set according to the shift (if used), and the V flag remains unaltered.

Syntax:
ORR<suffix> <dest>, <op 1>, <op 2>

Function:
dest = op_1 OR op_2

ORR is useful for setting bits in a bitfield. For each bit, ORRing with 1 will set the bit, and ORRing with 0 will leave it as it was.

4. Describe the basic rule(s) of the EOR instruction. What is the EOR instruction useful for?

EOR will perform a logical, bitwise, Exclusive OR between the two operands, placing the result in the destination register; this is useful for inverting certain bits. Operand 1 is a register, operand 2 can be a register, shifted register, or an immediate value (which may be shifted. If the S bit is set (ANDS), the N and Z flags are set according to the result, the C flag is set according to the shift (if used), and the V flag remains unaltered. Also known as XOR in other languages.

Syntax:
EOR<suffix> <dest>, <op 1>, <op 2>

Function:
dest = op_1 EOR op_2

EOR is useful for inverting bits in a bitfield. For each bit, EORing with 1 will invert the bit, and EORing with 0 will leave it as it was.

5. Describe the basic rule(s) of the BIC instruction. What is the BIC instruction useful for?

BIC will perform a bitwise AND of one register with the complement of a second. This sounds complicated, so think of it as a simple way to clear selected bits in a register - BIC with 1 clears that bit, and with 0 leaves it unchanged.

Operand 1 is a register, operand 2 can be a register, shifted register, or an immediate value (which may be shifted).

If the S bit is set (BICS), the N and Z flags are set according to the result, the C flag is updated according to the shifter, while the V flag is not altered.

Syntax:
BIC<suffix> <dest>, <op 1>, <op 2>

Function:
dest = op_1 AND (NOT op_2)

BIC is useful for clearing selected bits in a register.

6. At its most basic, what does a shift operation do?

A shift operation moves the bits in a register to the left or right and filling the vacant holes with zeros or ones. For instance, shifting bits left by one bit is equivalent to multiplying the original number by 2. Shifting bits right is equivalent to division. Thus shift operations can simulate certain forms of integer multiplication and division.

7. What are the two logical shift instructions on ARM?

Logical Shift Left and Shift Right.

The Logical Shift Left operation:

Rx, LSL Rn

LSL takes the contents of the register Rx and shifts to a more significant position by the amount specified by n or in the register Rn. The shift amount can be from zero to thirty one. The least significant bits introduced are zeroes. The high bits shifted off to the left are discarded, except for the notional bit thirty three (i.e., the last bit to be shifted off) which becomes the value of the carry flag on exit from the barrel shifter.

The Logical Shift Right operation:
Rx, LSR Rn

LSR does the notional opposite of shifting left. Everything is shifted to the right, to a less significant position. It is the same as register = value >>> shift. The shift amount can be from one to thirty two, and the carry-out is the value of the last bit shifted out.

8. How many bits left would you need to shift a value to multiply it by 2^3?

Logical Shift Left by 3 bits would be the same as multiplying by 2, 3 times.

9. Briefly describe the arithmetic shift instructions.

Arithmetic Shift Right is similar to LSR, with the exception that the high bits are filled with the value of bit 31 of the register being shifted (Rx), in order to preserve the sign in 2's complement maths. It is otherwise the same as register = value >> shift.

10. Describe the ROR instruction.

Rotate Right is similar to a Logical Shift Right, except the bits which would normally be shifted off the right are replaced on the left, thus the bits 'rotate'. If it was possible to specify the amount to be shifted as 32, the result would be that all the bits would be rotated by 32 places, right back to their original positions.

It is not possible to ROR #0, as that encoding is used to signify RRX. The carry-out is the value of the last bit shifted.