1. How would you negate a number in sign magnitude notation?

   (a) Flip all the bits
   (b) Flip the magnitude bits
   (c) Flip the sign bit
   (d) Flip the sign bit and add 1 to the magnitude

   Answer: (c)

2. How would you negate a number in 1's complement representation?

   (a) Flip all bits
   (b) Flip the most significant bit
   (c) Flip all bits and add 1 to the result
   (d) Flip the most significant bit and add 1 to the result

   Answer: (a)

3. How would you negate a number in 2's complement representation?

   (a) Flip all bits
   (b) Flip the most significant bit
   (c) Flip all bits and add 1 to the result
   (d) Flip the most significant bit and add 1 to the result

   Answer: (c)

4. Which of the following are advantages of the 2's complement notation over the other two representations? Select all that apply.

   (a) Only in the 2's complement notation, there is a unique representation for zero
   (b) Only in the 2's complement notation, there is a unique representation for all powers of 2
   (c) The addition algorithm is the same for signed and unsigned values
   (d) Addition and subtraction algorithms are nearly the same

   Answer: (a), (c), (d)

5. Which of the three number representations does MIPS use?

   (a) Sign magnitude notation

   (b) 1's complement notation

   (c) 2's complement notation

   Answer: (c)

6. Why does MIPS use the 2's complement notation?

   Answer: In 2's complement, negative numbers are obtained by flipping all the bits of a positive number and adding 1. This representation simplifies arithmetic operations and has a unique representation for zero, which is advantageous for hardware implementation.

7. What is the difference between `lb` and `lbu`?

   (a) There is no difference, `lbu` is a pseudo-instruction used in place of `lb` sometimes

   (b) `lb` performs sign extension, to get the right 32-bit representation

   (c) `lbu` performs sign extension, to get the right 32-bit representation

   (d) `lb` loads onto the lower order 8 bits, `lbu` onto the higher order 8 bits

   Answer: (b)

8. Do we need a separate `lwu` instruction?

   (a) Yes

   (b) No

   Answer: (b)

9. Which of the following instructions have an unsigned version in MIPS? Select all that apply.

   (a) `sll`

   (b) `and`

   (c) `slt`

   (d) `slti`

(e) `beq`

Answer: (c), (d)

10. For a pair of 32-bit registers we add, is the resultant 32-bits of `add` and `addu` the same always?

    (a) Yes
    (b) No

    Answer: (a)

11. If the result 32-bit patterns of `add` and `addu` are the same, why do we have a separate `addu` instruction?

    (a) `addu` is actually a pseudo-instruction for `add`
    (b) `addu` produces overflow under different conditions compared to `add`

    Answer: (b)

12. Register `$s0` is used to index into an array of 100 bytes. We want to set `$t0` if `$s0` has a valid index value, in a single `sltiu` instruction. Specify this instruction. (Use all small case, a space between the instruction and the operands, a comma and no space between each adjacent operand).

    Answer:

```
sltiu $t0, $s0, 100
```

13. How come we do not need six bits to encode the register number, given that we have a total of 64 registers in MIPS?

    (a) The earlier information was wrong, we actually need six bits
    (b) We use a bit from the co-processor for this purpose
    (c) The set of FP instructions are separate from the set of integer instructions
    (d) Whether FP regs are used or int regs are used is decided at the time of program loading

    Answer: (c)

14. What is a safe way of implementing an almost equivalent version of x==y when they both are floating point numbers?

(a) There is no safe way, one has to run the program and hope for the best

(b) Run the program on different operating systems

(c) Run the program on different machines

(d) Check if `|x - y|` is below a threshold

(e) Check if `x != y` and negate the comparison result

Answer: (d)