HOMEWORK 1

§1.3 (Data Representation) – Binary representation; bits, LSB, MSB, binary addition. Storage sizes. Hexadecimal representation. Two's complement representation. Conversion of integers between representations. Maximum and minimum representable values. ASCII and Unicode representation of character strings.

§2.1 (General Concepts in Microcomputer Design) – CPU: registers, clock, control unit, ALU, memory storage unit. Data, I/O, control, and address buses. Clock cycles. Instruction execution cycle: instruction pointer/program counter; fetch-decode-execute (FDX) cycle, including fetching operands and storing output operands. Procedure for reading from memory; cache memory. Processes; tasks; multitasking; scheduler; task switching; preemptive vs. non-preemptive multitasking.

Supplemental – Executables (.exe files) vs. DLLs vs. static-link libraries; PE file format: text, data, idata sections; loaders; relationship between the text and data sections of a PE file and the code and data segments of a process

Directions: *This assignment is due by **2:00 p.m. on Friday, September 5**. Submit your answers as a PDF in Canvas; see the syllabus for detailed instructions and a description of the late policy. Paper submissions will **not** be accepted.*

Scoring (45 total points): *#1–5: 3 points each • #6–10: 4 points each • #11–12: 5 points each*

1. Consider the 8-bit binary number 00110001.

   (a) What are the values of bits 2 through 4?

   (b) What is the value of the MSB?

   (c) What is the hexadecimal representation of this number?

2. Add the following unsigned binary numbers, showing carries if applicable:

   ```
     00000101
   + 11100111
   ```

3. What is the decimal representation of each of the following 8-bit *signed* (two's complement) binary integers? *Show your work.*

(a)	(b)	(c)
10000100	01000000	11111111

4. What is the 8-bit *signed* (two's-complement) *binary* representation of each of the following decimal integers? Show your work.

(a)	(b)	(c)
–42	42	–128

5. If a processor is clocked at 3.2 GHz, what is the length of one clock cycle, in nanoseconds? *Show your work.*

6. What is the decimal representation of each of the following 8-bit *signed* (two's complement) hexadecimal integers? *Show your work.*

(a)	(b)
9Ch	31h

7. What is the 8-bit *signed* (two's-complement) *hexadecimal* representation of each of the following decimal integers? Show your work.

(a)	(b)
–47	47

8. Suppose a computer's memory contains the following four bytes: 4Eh 6Fh 21h 00h.

   (a) Assume these bytes represent an ASCII string. What string do they represent?

   (b) Would the UTF-8 encoding of this string be the same as, or different from, the ASCII encoding? Explain.

9. What is the range of values that can be represented by

   (a) a 73-bit unsigned integer?

   (b) a 73-bit signed integer?

10. Can the number –10 be represented as a 4-bit signed integer? If so, show its 4-bit signed (two's complement) representation. If not, explain why; describe how many bits are required, and show its representation using that many bits.

11. Although it has virtually no practical applications, it is possible to represent numbers in base –2; these are called *negabinary* numbers. Negabinary numbers use the digits 0 and 1, but they are converted to decimal by multiplying the digits by powers of –2. For example,

$$\begin{aligned} 00111001_{-2} &= 1 \times (-2)^5 + 1 \times (-2)^4 + 1 \times (-2)^3 + 1 \times (-2)^0 \\ &= -32 + 16 + -8 + 1 \\ &= -23. \end{aligned}$$

   What is the range of values that can be represented by a 16-bit unsigned negabinary integer? Explain your answer.

   (Hint: Try solving the problem for 4-bit unsigned negabinary integers first.)

12. Consider the fetch-decode-execute cycle. Suppose a computer's instruction set is designed such that every instruction will either *fetch operands,* or *store output operands,* but never both. Joe Bob decides that having separate *fetch operands* and *store output operand* steps is wasteful, since both of them access memory and it will never be necessary to perform both for the same instruction. So, he wants to get rid of the *store output operand* step; instead, he'll replace the *fetch operands* step with a single *fetch-operands-or-store-output-operand* step. This means the FDX cycle will look like this:

```
while (true) {
        Fetch next instruction
        Advance the instruction pointer
        Decode the instruction
        If memory operands involved, fetch operands or store output operand (Joe's step)
        Execute the instruction
}
```

   Is there a problem with his design, or will this work? Explain.