Any problem labelled *Exercise* will not be solved during the review session. You should look at these problems yourself and make sure you understand how to do them.

## 1 Data Representations and Binary Operations

1. What does 10001101 represent?

2. Give the 8-bit two’s complement binary representation of the following decimal numbers:
   - (a) $127_{10}$
   - (b) $0_{10}$
   - (c) $-30_{10}$

3. Give the decimal representations of the following two’s complement numbers:
   - (a) $10000001_2$
   - (b) $01111111_2$

4. Give the hexadecimal representation of $1000010100001111_2$

5. What is the largest positive integer that can be represented using a 32-bit two’s complement representation?

6. Compute the following expressions, assuming that all values are 4-bit two’s complement:
   - (a) $0100 - 0010$
   - (b) $1111 + 0001$

7. Compute the following:
   - (a) $7 \ll 3$
   - (b) $0 | 999999$
   - (c) $0xff \& 32$
2 MIPS Programming

1. Write a MIPS procedure called *StringMap* that interprets the value in $1 as an address of a string (an array of characters that ends with -1), and the value in $2 as the address of a procedure that interprets the value in $1 as a character and returns a character in $3 (we call it the character transformation function). *StringMap* should apply the character transformation function to character of the string and place the address of the transformed string in $3. You may assume the character transformation function will save every used registers and restore them after it finishes. However, you may not assume the caller of *StringMap* store/restore the registers.

Hint: [https://en.wikipedia.org/wiki/Map_(higher-order_function)](https://en.wikipedia.org/wiki/Map_(higher-order_function))

*StringMap* is a variant of map.

2. Write a MIPS program that interprets the value in $1 as the address of an array of MIPS instructions and executes the program stored in the array. You may assume each element contains exactly one instruction, and the MIPS program stored in the array is valid (ending with "jr $31") and takes no arguments.

3. Exercise. Write a MIPS procedure called *gcd* which interprets the values in $1 and $2 as unsigned decimal integers and computes their greatest common divisor using recursion, placing the result in $3.

Hint: [https://en.wikipedia.org/wiki/Greatest_common_divisor#Using_Euclid.s_algorithm](https://en.wikipedia.org/wiki/Greatest_common_divisor#Using_Euclid.s_algorithm)

3 MIPS assembler

1. Give the symbol table for following MIPS assembly language program:

```
a: b: c: add $0, $0, $0
label1: sub $1, $1, $1
label3: labelx: beq $1, $0, label1
         jr $31
end:
```

2. What is the purpose of the first and second pass in an assembler? Why does assemblers require two passes?

3. Ed tries to write his CS350 assignment in MIPS that requires to loop through 90,000 lines of code. Describe a potential problem that ED could encounter if he uses branching instructions. How could Ed solve this problem?
4. (a) A student tries to obfuscate a MIPS program by storing it in hexadecimal format (as was done in Assignment 1):

```
.word 0x00001814
.word 0x00000001
.word 0x00002014
.word 0x00000001
.word 0x00620018
.word 0x00001812
.word 0x00240822
.word 0x1420fffc
.word 0x03e00008
```

Convert the above code to a more human-readable MIPS program. Your program must use labels where possible.

(b) What does the program do?

4 Regular Languages

Note that for simplicity, leading zeroes are allowed.

1. Prove that the language \( L\) over \( \{A, B, C, D\} \) is regular.

2. Give a DFA for each of the following languages:

   (a) The language \( L \) over \( \Sigma = \{g, o, e, s\} \) which contains words similar to *goose* or *geese*: with 1 or more o's, e's respectively between the g and the s. For example, *goose*, *gese* and *geeeeeese* are words in the language, but *gse* and *goese* are not.

   (b) The language \( L \) over \( \Sigma = \Gamma \cup \{+\} \), where \( \Gamma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \), such that:

   \[
   L = \{u + v : u, v \in \Gamma^*, u \equiv 0 \pmod{2} \text{ and } v \equiv 1 \pmod{2}\}
   \]

   In words, language \( L \) contains all addition expression with exactly two operands whose first operand is an even decimal number and the second operand is an odd decimal number.

   (c) *Exercise.* The language \( L \) over \( \Sigma = \{0, 1\} \) which contains strings with a binary value that is congruent to 3 modulo 8. (That is, the binary value of string \( x \) divided by 8 gives a remainder of 3). Hint: Think about the 3 least-significant (rightmost) bits.

3. Give a regular expression for each of the following languages:

   (a) The language \( L \) over \( \Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \) of positive decimal numbers divisible by 4.
(b) the language \( L \) over \( \Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, \text{.word, x, .} \} \) of valid MIPS \texttt{.word} directives with positive decimal and all hexadecimal arguments. (Use \_ to represent a space)

4. You are given a regular language \( L \) and a NFA \( M(L) = \{\Sigma, Q, q_0, A, \delta\} \) for the language. In pseudocode, write a function that takes a string \( s \in \Sigma^* \) as an argument. The function returns 1 if the string is in language \( L \), return 0 otherwise.

5. Give an NFA for each of the following languages:
   (a) The language \( A \) over \( \Sigma = \{0, 1\} \) of strings ends with either "100" or "1".
   (b) The language \( B \) over \( \Sigma = \{0, 1\} \) that is the complement of \( A \). (For any language \( A \subseteq \Sigma^* \), the complement of \( A \) is \( \Sigma^* \setminus A \))

6. For the 2 NFA from the previous question, give an \( \epsilon \)-NFA for the language \( B^* \cup A^* \), by connecting the 2 NFA using \( \epsilon \)-transitions. Describe the language accepted by this automaton.

7. For the following NFA, convert it into a DFA using subset construction.
5 Context-Free Languages

1. Show that all regular languages are context-free.

2. Give a CFG for each of the language below. For each CFG, find a string that is in the language and show its leftmost derivation using the CFG.

   (a) The language over $\Sigma = \{a, b\}$ of words which start with an $a$ and end with a $b$.

   (b) The language over $\Sigma = \{0, 1\}$ of palindromes whose length is even.

   (c) The language over $\Sigma = \{(,\})$ of matched parentheses, where each opening parenthesis is followed by a unique closing parenthesis.

   (d) The language over $\Sigma = \{a, b\}$ of words that have an equal number of $a$’s and $b$’s.