Any problem labelled \textit{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.

\textbf{Contents}

1 Bits and Bytes 1

2 MIPS Assembly Language Programming 2

3 MIPS Assembler 3

4 Regular Languages 4

5 Scanning 5

1 Bits and Bytes

1. Give a two’s complement binary representation (and corresponding hexadecimal representation) of the following decimal numbers:

   (a) 127
   (b) 241
   (c) −30

2. Give the decimal and binary representations of the following two’s complement numbers:

   (a) 0x81
   (b) 0x7F

3. Give the (unsigned) decimal and binary representations of 0xCC.

4. What is the range (in decimal) of a 32-bit two’s complement number?
2. MIPS Assembly Language Programming

1. Write a MIPS procedure called `StringMap` that interprets the value in $1 as the address of a string (an array of characters that ends with -1), and the value in $2 as the address of a procedure which 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 restore any registers it modifies. However, you may not assume the caller of `StringMap` stores/restores any registers.

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.%27s_algorithm](https://en.wikipedia.org/wiki/Greatest_common_divisor#Using_Euclid.%27s_algorithm)

4. Exercise. According to the MIPS assembly language reference sheet, if you load a word from address 0xffff0004, it will place the next byte from standard input in the destination register. Write a program which takes a number in $1 and reads in that many elements from standard input into an array, stores the address of the array in $1 and the number of elements in $2, and then jumps to a symbol called "start". This is essentially mips.array except that the array elements are provided as characters rather than as integers.

5. Exercise. Given the following hexdump of an assembled MIPS assembly language program, determine the original program without using .word.

```
00000000: 0000 1814 ....
00000004: 0000 0001 ....
00000008: 0000 2014 .. .
0000000c: 0000 0001 ....
00000010: 0062 0018 .b..
00000014: 0000 1812 ....
00000018: 0024 0822 .$..
0000001c: 1420 fffe . .
00000020: 03e0 0008 ....
```

Your program must use labels where possible. What does the program do?
3 MIPS Assembler

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

```
mult: mult $1, $2
beq $0, $0, branchLocation
frog:
gorf: .word gorf

branchLocation: ; branch here
; now return

jr $31
end: ; end of the program
```

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. What information can we not get back when we disassemble a program?
4 Regular Languages

Note that for simplicity, you may chose to allow leading zeroes.

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

2. Give a DFA for each of the following languages:
   
   (a) The language \( L \) over \( \Sigma = \Gamma \cup \{+\} \), where \( \Gamma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \), such that:
   
   \( L = \{u + v : u, v \in \Gamma^*, u \equiv 0 \pmod{2} \text{ and } v \equiv 1 \pmod{2}\} \)
   
   (b) 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) Exercise. 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 .word directives with positive decimal and all hexadecimal arguments. (Use _ to represent a space)

4. You are given a regular language \( L \) and a DFA \( M = \{\Sigma, Q, q_0, A, \delta\} \) for the language. In pseudocode, write a function that takes a string \( s \in \Sigma^* \) as an argument and returns 1 if the string is in language \( L \), 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 \( \Sigma^* \setminus A \))
   
   (c) The language over \( \Sigma = \{c,a,r,t\} \) where words in the language must end with “car” or have ”cat” as a substring.
   
   (d) The language over \( \Sigma = \{c,a,r,t\} \) where words must end with ”tar” and have “rat” as a substring.
   
   (e) Exercise. The language over \( \Sigma = \{a,b,c,d\} \) where words must either include at least one a and no bs, or must include at least one b followed by at least one c.

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

7. Convert the following NFA into a DFA.

![NFA Diagram]

\( \text{start} \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \)
5 Scanning

1. Using the above DFA, produce a tokenization of the following strings using the Simplified Maximal Munch algorithm. If the scanning produces an error, output all the tokens recognized up until the error and then the string ERROR.

(a) 0xc0x1-23--70xff
(b) 0xbeef--0xkcd