1 Data Representation and Binary Operations

1. What does the byte 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 integer that can be represented using a 32-bit two’s complement representation? The smallest?

6. Compute the following expressions, assuming that all values are 4-bit two’s complement:
   (a) $0001 + 0011$
   (b) $0100 - 0010$
   (c) $1111 + 0001$

7. Compute the following:
   (a) $15 >> 3$
   (b) $0 | 99999999$
   (c) $0xff & 32$
2 MIPS Programming

All procedures in this section should use callee-save.

1. Write the MIPS code that is equivalent to the following C code snippet:

   ```c
       A[c] = B[d];
   }
   ```

   You may assume that $A, B$ are arrays whose addresses are stored in $\$1$ and $\$2$, and $c, d$ are variables whose addresses are stored in $\$3$ and $\$4$ respectively.

2. Write a MIPS procedure called `caps` which interprets the value in $\$1$ as the address of a null-terminated array of ASCII characters. The procedure should compute the number of upper case letters and store it in $\$3$.

3. Write a MIPS procedure called `exec` which interprets the value in $\$1$ as the address of a MIPS program and executes that program.

4. Write a MIPS procedure called `map` which interprets the value in $\$1$ as the address of an array preceded by its length (for example, if the memory at $\$1$ contains 5, then the following 5 words in memory are the contents of the array) and the value in $\$2$ as the address of a single-argument procedure. `map` should transform each array value $a$ into $p(a)$, where $p$ is the procedure in $\$2$.

3 Assembling MIPS

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

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

2. Why does the MIPS assembler need to do two passes through its input?

3. (a) To prevent competitors from being able to steal her code, Alice obfuscated her solution to the following program by releasing it stored as hexadecimal values:
Show that Alice’s method is ineffective by converting the above code to a more human-readable MIPS program. Use labels whenever possible and avoid .word except where it was likely used in Alice’s original program.

(b) What does Alice’s program do?

4. What information can we not always get back when disassembling a program?

5. Bob writes a MIPS program that needs a very long loop (roughly 90,000 instructions). What problem might Bob run into when writing his loop in the usual way? Suggest two ways that he could resolve it.

4 Regular Languages

1. Show that the language \( L = \{a, b, c, d, e\} \) is regular.

2. Give both a regular expression and a DFA for each of the following languages:

   (a) The language over \( \Sigma = \{e, g, o, s\} \) which contains words similar to “goose” or “geese” but with any number (minimum 1) of “o”s and “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 over \( \Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \) of positive decimal numbers divisible by 4.

3. Give a DFA for the language over \( \Sigma = \{a, b, c\} \) of words without consecutive a or b.

4. Give an NFA for each of the following languages over \( \Sigma = \{0, 1, 2, 3\} \):

   (a) The language of strings ending in 00 or 11.

   (b) The language of strings beginning in 233 or 332.

   (c) The language of strings beginning in 123 and ending in 321.
5. Give an $\epsilon$-NFA for the language $(AB) \cup C$, where $A$, $B$, and $C$ are the three languages from the previous question.

6. Convert the following NFA into a DFA using the subset construction method.

7. Convert the following $\epsilon$-NFA into an NFA using the $\epsilon$ closure method.

5 Scanning

(a) Give an example of a scanning DFA and input string such that the simplified maximal munch algorithm will give an error despite the input string having a possible tokenization.
(b) Consider the following scanning DFA:

![DFA Diagram]

Give tokenizations for the following strings. In any case where you encounter an error, put ERROR and stop tokenizing.

i. a=b!=c
ii. a=0=c
iii. a=b=0
iv. a=!b
v. var a == b
vi. var a==b