

# ICS Exercise 1

September 26, 2023

## 1 Number Conversion

Given  $Y = X + 0x38$ ,  $Z = (Y \mid X) - 56$ ,  $N = (M \gg 2) \& X$ . Fill in the table below.

Variable	Binary	Octal	Decimal	Hexadecimal
X	1 1101 0111	727	471	0x1d7
Y	10 0000 1111	1017	527	0x20f
Z	11 1010 0111	1647	935	0x3a7
M	1 1111 0111	767	503	0x1f7
N	101 0101	125	85	0x55

## 2 Print the decimal numbers

In computing and electronic systems, binary-coded decimal (BCD) is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits. Packed BCD 8421 (also known as Simple Binary-Coded Decimal, SBCD) uses four bits to represent the range 0 to 9 while the weight of each bit is '8 4 2 1'. For example, the decimal number 2019 is turned into 0x2019 in packed BCD 8421 encoding. The following table represents decimal digits from 0 to 9 in packed BCD 8421 and ASCII.

Digit	Packed BCD 8421	ASCII	Digit	Packed BCD 8421	ASCII
0	0b0000	0x30	5	0b0101	0x35
1	0b0001	0x31	6	0b0110	0x36
2	0b0010	0x32	7	0b0111	0x37
3	0b0011	0x33	8	0b1000	0x38
4	0b0100	0x34	9	0b1001	0x39

You are asked to write a function `bcd2str` in C to turn a 4-byte packed BCD 8421 number into a string `num_str[9]` which represents the same number and ends up with a `'\0'`. For example, the packed BCD 8421 number 0x2019 is supposed to be turned into `num_str[9] = {'0', '0', '0', '0', '2', '0',`

'1', '9', '\0'}. The pointer `num_str` has been pointed to the pre-allocated `num_str[9]` before calling this function. And the `input` stores the 4-byte packed BCD 8421 number which you should transform.

```

1 void bcd2str(unsigned int input,
2             char *num_str)
3 {
4     for (int i = 0; i < 8; i++) {
5         num_str[i] = 0x30 | \
6             (input >> ((7 - i) * 4) & 0xF);
7     }
8     num_str[8] = '\0';
9 }

```

## 3 Binary Operations

### 3.1

Consider the following C program

```

1 int a = 0x3f;
2 unsigned short ua = a;
3 int b = ua >> 1;
4 short c = (b | 0);
5 unsigned int d = (~(unsigned int)a) ^ 0xc4;
6 int e = 0x95 & 0x6f;

```

Assume the program will run on an **8-bit** machine. A 'short' integer is encoded in **4 bits**, while a normal 'int' is encoded in **8 bits**. Please fill in the blanks below.

Expression	Binary Representation
a	0011 1111
ua	1111
b	0000 0111
c	0001
d	0000 0100
e	0000 0101
(d   (!0)) ^ e	0000 0000
(e + 0x30) + (0x13 >> 2)	0011 1001
(ua >> c) + (b << d)	0111 0111

### 3.2

Given two 32-bit numbers, **N** and **M**, and two bit positions, **i** and **j** (**i** < **j**). Write a method to set all bits between **i** and **j** in **N** equal to bits between **i** and **j** in **M**.

```
1 int mask = (((unsigned)(~0)) >> (31-j+i)) << i;
2 int result = (N & (~mask)) | (M & mask);
```

### 3.3

Write a function that add two numbers **A** and **B** using bit operations (+ not allowed).

```
1 while (B != 0) {
2     int a = A ^ B;
3     int b = (A & B) << 1;
4     A = a;
5     B = b;
6 }
7 return A;
```