PES, Section 2.4
Hexadecimal

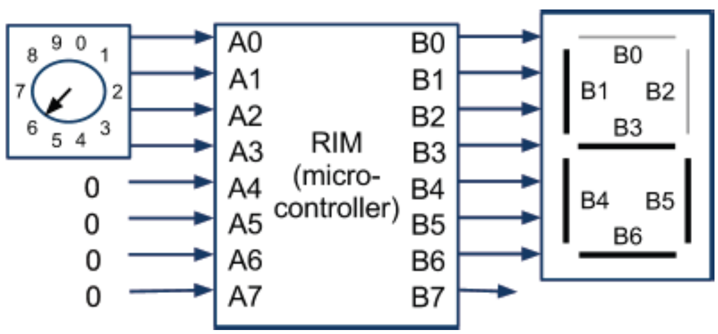
1. Rewrite the following single-line RIMS-compatible C statement to use hexadecimal.

   B = 247;
   B = 0xF7;

2. Write a single statement for RIMS that sets B0 to 1 if A3-A0 are all 1s, and A7-A4 are all zeroes. (Use a hex constant).

   B0 = (A == 0x0F);

3. Consider the following embedded system with a dial that can set A3..A0 to binary 0 to 9, and a 7-segment display.



On the next page is a (partial) RIMS C program that appropriately sets the display for the given dial position.

A new company has created a new dial that can represent all 16 4-bit combinations, from 0-F. Extend the partial RIMS C program to display the six hexadecimal characters larger than 9 (pay attention to upper/lowercase):

A
b
C
d
E
F

```
#include “RIMS.h”
void main()
{
    while (1) {
        switch( A ) {
            case 0 : B = 0x77; break;       // 0111 0111 (0)
            case 1 : B = 0x24; break;       // 0010 0100 (1)
            case 2 : B = 0x5d; break;       // 0101 1101 (2)
            //...
            case 9 : B = 0x6f; break;       // 0110 1111 (9)

            case 10 : B = 0x3f; break;    // 0011 1111 (A)
            case 11 : B = 0x7A; break;   // 0111 1010 (b)
            case 12 : B = 0x53; break;   // 0101 0011 (C)
            case 13 : B = 0x7C; break;   // 0111 1100 (d)
            case 14 : B = 0x5B; break;   // 0101 1011 (E)
            case 15 : B = 0x1B; break;   // 0001 1011 (F)

            // 1000 0000 (Activate B7 to indicate an error)
            default: B = 0x80; break;
        }
    }
}
```

PES, Section 2.5
Bitwise ops

1. What does the C language statement (0x03 || 0x01) evaluate to?

   True (0x01)

2. Give examples of two 8-bit hexadecimal values x and y such that (x | y) and (x || y) produce the same value.

   x = 0x01; y = 0x01

   (x | y) → (0x01 | 0x01) → 0x01
   (x || y) → (true || true) → true (0x01)

3. Give examples of two 8-bit hexadecimal values x and y such that (x & y) and (x && y) produce different values.

   x = 0x07; y = 0x03;

   (x & y) → (0x07 & 0x03) → (0000 0111 & 0000 0011) → 0000 0011 → 0x03
   (x && y) → (true & true) → true (0x01)

   Note: Multiple solutions are possible

4. Consider the bitwise xor operator ^, e.g. as used in the following C statement:

   z = x ^ y;

   Rewrite the C statement to use the other bitwise operations (&, |, ~), but not ^.

   z = (x & ~y) | (~x & y);

PES, Section 2.2
C Data Types

1. How many bits are in an unsigned long?

32

2. What is the range of values that an unsigned short can have?

An unsigned short is 16-bits long. Therefore the range of values is [0, 2^{16} – 1]

3. How do you represent a 1-bit value in C?

Use a char; 7 bits are unavoidably wasted.
You could also use an enumerated data type.

4. Suppose that you have an unsigned short that will represent the age of a person. What is an appropriate name for the variable?

usage;
(or us_age).
(pre-pend "us" so that you always know that the data type is <u>u</u>nsigned <u>s</u>hort)

5. Does use of the signed or unsigned 'int' data type affect portability? Explain.

Yes. C does not define the number of bits in the 'int' data type, so it varies from compiler-to-compiler and platform to platform.

6. Suppose that you have a variable foo that will represent a value in the range -1 to 255. What data type should you use and why?

signed short;
- **"signed" is necessary to represent negative values**
- **9 bits are required to represent 257 discrete values. The smallest C data type that can represent 9-bit values is "short" (16 bits).**

PES, Section 2.3
RIMS I/O

1. Write a short single-line RIMS-compatible C statement that sets the value of B to twice the value of A (ignore the possibility of overflow).

   B = 2*A;

2. The following single-line RIMS-compatible C statement adds 5 to the value of A and outputs the result on B (ignore overflow). Correct the program to adhere to the standards outlined in the section.

   const unsigned char C = 5;
   B = A + C;

3. What is the mistake in the following RIMS-compatible C program fragment (ignore overflow)?

```
const signed char C = 16;
const unsigned char D = 10;
unsigned char i;
for(i = 0; i < D; i++) {
        B = A + C;
        C = (A + C)/2;
}
```

   It is illegal to overwrite the value of a variable declared 'const'.