Tutorial 3: Number Systems

Problem 1: Data Representation

Consider the number A = 0xF2. What is the value of this number when represented in; 8-bit unsigned number, 8-bit sign-magnitude, 8-bit one’s complement, and 8-bit two’s complement?

Unsigned: \(1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 = 128 + 64 + 32 + 16 + 2 = 242\)

Signed-magnitude: \((-1)^1 \times (1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2) = -(64 + 32 + 16 + 2) = -114\)

One’s complement: \((1 \times 2^7 - 1) + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 = 128 + 64 + 32 + 16 + 2 = -13\)

Two’s complement: \(-1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 = -128 + 64 + 32 + 16 + 2 = -14\)

Problem 2: Data Types

Consider the C code in Figure 1. Answer the following questions.

What are the outputs of the printf statements?

What are the outputs of the printf statements if (\(a = 0xFFFFFFE5\)) and (\(b = 0xFFFFFFE5\))?

```c
#include <stdio.h>

int main (void)
{
    int a = 0xE5;
    char b = 0xE5;

    printf("integer = \"%d\"\n\n", a);
    printf("char = \"%d\"\n\n", b);

    return 0;
}
```

Figure 1: Program on Data Type Conversion

The number (\(a = 0xE5\)), being declared as an int (32-bit number), will be internally represented as (a = 0x000000E5) which is (a = 14 \times 16^1 + 5 = 229). The number (\(b = 0xE5\)) being declared as a char (8-bit number), will be internally represented as (a = 0xE5 = 0b11100101 = -1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 = -27). So, the output of the printf statements are.

integer = "229"
char = "-27"

Figure 2: The Outputs of printf Statements

The number (\(a = 0xFFFFFFE5\)), being declared as an int (32-bit number), is simply a representation of 8-bit number (\(0xE5 = -27\)) in 32 bits. The number (\(b = 0xFFFFFFE5\)) being declared as a char (8-bit number), has more bits than can accommodate. So all the bits after bit 7 will be discarded, and the number will be internally represented as (a = 0xE5 = -27). So, the output of the printf statements are.
Problem 3: More Data Type Conversion

Consider the C code in Figure 4. Answer the following questions.

Which of the print statements will be printed out?

```c
#include <stdio.h>

int main (void)
{
    unsigned int a = 0xFFFFFFFB;
    int b = 0xFFFFFFFB;
    char c = 0xFD;

    if (a < (unsigned) c)
        {printf("a < c \n\n");}
    if (b < (int) c)
        {printf("b < c \n\n");}

    return 0;
}
```

The statement `(if (a < (unsigned) c))` treats both `a` and `c` as positive numbers (`a = 0xFFFFFFFB = 4294967291`) and (`c = 0xFD = 253`), and therefore the condition `(a < c)` is not true, and the first print statement does not execute.

The statement `(if (b < (int) c))` treats both `b` and `c` as negative numbers (`a = 0xFFFFFFFB = -5`) and (`c = 0xFD = -3`), and therefore the condition `(b < c)` is true and the second print statement executes.

Problem 4: Binary Prefixes

How much is $2^{27}$ Byes?

We need to access a memory organisation as large as 2.4 Mi Bytes. How many address lines are required for this purpose?

$2^{27}$ Byes = $2^7 \times 2^{20}$ Byes = 128 mebi Byes = 126 Mi Byes.

$2.4 \text{ Mi} \Rightarrow \left\lfloor \log_2 (2.4 \text{ Mi}) \right\rfloor = \left\lfloor \log_2 (2.4) \right\rfloor + \log_2 (\text{Mi}) = 2 + \log_2 (2^{20}) = 2 + 20 = 22 \text{ lines.}$