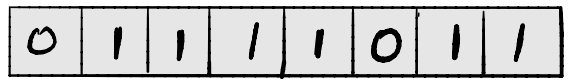
Homework 1 Solutions

1. Using 8-bit bytes, show how to represent 123. Clearly state the byte values using hexadecimal, and the number of bytes required for each context. Simply indicate the case if the code is not able to represent the information.
   (a) Unsigned integer
   (b) Two's complement
   (c) BCD
   (d) ASCII

123 →



(Binary)

a) Unsigned Integer

0111 1011 (hex)

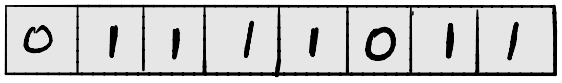
7 B $= (7B)_{16}$

No. of bytes required = 1

b) Since 123 is a positive integer, the two's complement representation will be the same as in (a).

Two's Complement

123 →



(Binary)

a) Unsigned Integer

0111 1011 (hex)

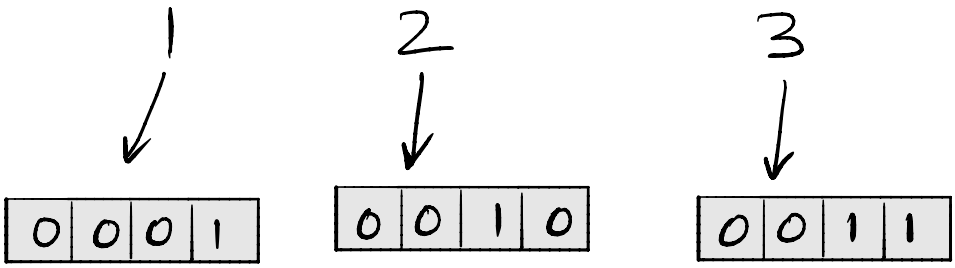
7 B $= (7B)_{16}$

Takes 1 Byte

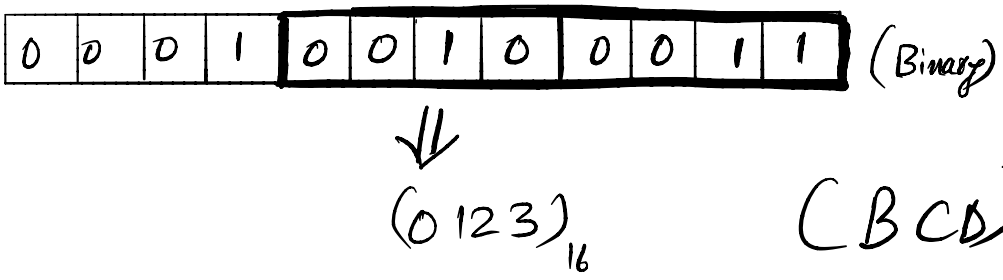
c)

BCD Each Decimal Digit is represented by a group of 4 binary digits. (bits)

1 2 3



Takes 2 Bytes



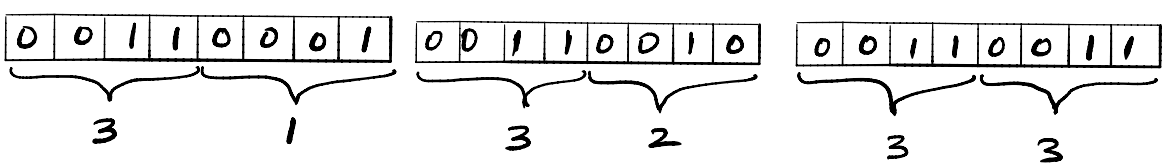
$(0123)_{16}$ (BCD)

d)

ASCII **Shortcut**: add 48 to each Digit.

1 2 3

49 50 51



Takes 3 Bytes.

$(313233)_{16}$ (hex)

2. Using 8-bit bytes, show how to represent -123. Clearly state the byte values using hexadecimal, and the number of bytes required for each context. Simply indicate the case if the code is not able to represent the information.
   (a) Unsigned integer
   (b) Two's complement
   (c) BCD
   (d) ASCII

a) -123 cannot be represented as an unsigned integer.

b) 123 → 1 1111011

1st complement 1 0000100

2's complement 1 0000100 + 1 = 1 0000101 # Take 1 byte

1 0000101

8 5 = $(85)_{16}$ (hex)

c) BCD

By machine implement packed BCD differently.

One of the examples is as follows:

-123

1011 0001 0010 0011 ⇒ $(B123)_{16}$ (BCD) # Takes 2 bytes.

(negative sign encoded as (1011))

d) ASCII

−123 → 51

2D 49 50

Add 48 for digits

Takes 4 bytes

(ASCII value for "−" is 2D) = $(2D\ 49\ 50\ 51)_{16}$

Q3)

3. Using 8-bit bytes, show how to represent 56,789. Clearly state the byte values using hexadecimal, and the number of bytes required for each context. Simply indicate the case if the code is not able to represent the information.
   (a) Unsigned integer
   (b) Two's complement
   (c) BCD

a) 56,789 →

Takes 2 bytes

1101 1101 1101 0101

D D D 5

↓

$(DDD5)_{16}$ ⇒ (hex)

b) Since it a positive number the 2's complement will be the same as above.

c) 5 6 7 8 9 → 1001

0101 0110 0111 1000

0000 0101 0110 0111 1000 1001

Takes 3 bytes

$(056789)_{16}$

(BCD)

Q4

4. Using the variable x, give definitions for the following:
   a. An integer
   b. A pointer to an integer
   c. An array of 10 integers
   d. An array of 10 pointers to integers

a) `int x;`

b) `int* x;`

c) `int x[10];`

d) `int *x[10];`

Q5)

5. What is the output of the following program:

```
#include <stdio.h>
int main ()
{
        int vals[5] = {4, 3, 2, 5, 1};
        int i;
        for (i=0; i<=5; i++)
        {
                printf("vals[%d]=%d\n", i, vals[i]);
        }
        return 0;
}
```

Output:

vals [0] = 4

vals [1] = 3

vals [2] = 2

vals [3] = 5

vals [4] = 1

Q6

6. What is the output to the following program:

```
#include <stdio.h>
void fun(int y)
{
        y = 30;
}

int main()
{
        int y = 20;
        fun(y);
        printf("%d", y);
        return 0;
}
```

Output: 20

Explanation: When we call the function fun(y), another copy of y is created & that copy is only their until the function is not finished. As soon as the control comes out of

the function, the new created (y=30) is deleted. This is why the output will be 20 & not 30. This is known as Call by value.

Q1)

7. What is the output to the following program:

```
#include <stdio.h>
void fun(int *y)
{
        *y = 30;
}

int main()
{
        int y = 20;
        fun(&y);
        printf("%d", y);
        return 0;
}
```

Output: 30

The output will be 30 in this case because now we are passing the address of the variable & not its value. So, all changes to the made to y will be at the same address & no new copies are created. This is known as Call by Reference.

Q8)

Described the purpose of the keyword "volatile".

Ans The "volatile" keyword is used to direct the compiler to not make any optimizations in the implementation of the variable. This is because the value of the variable may be changed at any point during the program (may be by one of the peripherals) Volatile variables are very common with memory-mapped I/O registers.

Q9)

Describe why it is not a good coding practice to dynamically allocate memory in embedded programming.

There are a number of reasons why dynamic memory allocation is discouraged.

1) Embedded systems can run for years & dynamically allocated memory causes wastage of memory due to fragmentation.

2) Dynamic allocation makes the debugging fairly difficult.

3) Dynamic allocation leads to memory leaks which are not desirable.

4) Dynamic memory is relatively slow.