Solutions to Tutorial 1

1.1 Number representation

(1) Solutions:

binary number	unsigned	2's-complement
(a) 0111 1111_2	127	127
(b) 1111 1111_2	255	-1
(c) 0000 0000_2	0	0
(d) 1000 0000_2	128	-128
(e) 1111 1110_2	254	-2

(2) Solutions: unsigned magnitude range **0 to 255**
2's complement range **-128 to 127**

(3) See lecture notes (Chap 12, page 15) on the range of different C data types.

(4) Suggested solution: (a) **`signed char`** .
(b) **`unsigned short int`**.
(c) **`unsigned long long int`**.
(d) **`_Bool`** from the stdbool.h header.

1.2 Hexadecimal number representation

(1) (a) and (d)
(2) (c) and (e)
(3) (a) **`-1`** (b) **`15`**
(4) Yes, this illustrates the behavior of sign extension.

1.3 Data representation in memory

(1) **`0x0002` (start address).** The format is **Little Endian.**

(2) "**`*Login:`**". Any subset of this is also correct.

(3) (a) `r.i = 0x67` (1-byte value)
(b) **`r.j = 0x696E3A00`** (4-byte, MSByte at lowest address since big endian)
(c) **`r.a[0] = 0x61`** (1-byte, first element of 3 element array)
(d) **`r.a[2] = 0x63`** (1-byte, last element of 3 element array)

Solutions to Tutorial 1

1.4 Data and Address Busses

(1) 16 Mbyte

(2) Two bytes

(3)
```
struct rec {
  long int j;  %starts at even address by
  unsigned char i;  %swapping j and i
  char a[3];
};
struct rec r;
```

1.5 VIP instructions, its execution and representation in memory

(1) **12 bits.**

(2) The machine code is **`0x02C001`**.

(3) **`MOV R1,#0xFFF`** moves the hex value 0xFFF into the register R1.
`MOV R2,#0x001` moves 0x001 into register is R2.
`ADD R1,R2` adds the content of registers R2 and R1, then place the result R1.

(4) After executing the first instruction, the content of the PC points to the next instruction, which starts at address **`0x002`**. The new value of the PC is **`0x002`** since the current content of the PC tells us which instruction will be executed next.

(5) Executing **`MOV R1,#0xFFF`** will set the Negative (N) flag because the value that was moved into register R1 is negative. The N flag is set whenever the MSB is 1.

(6) A total of **5 memory cycles** to execute all 3 instructions. **`MOV R1,#0xFFF`** and **`MOV R2,#0x001`** each takes 2 cycles each since there are 2 instruction words to be fetch from memory for each instruction. **`ADD R1,R2`** takes 1 cycle because this instruction has only 1 word. The execution of the add instruction does not incur any memory cycle due to pipelining.

(7) Not necessarily. Programs with `for` and `while` loops can be written with only a few instructions but can take longer to execute.

(8) The **`Z`** and **`C`** flag is set.

(9) The instruction encoding for mnemonic **`ADD R1,R2`** is **`0x412`**.