Chapter 4
High and Low Level Programming

Students who are already familiar with number systems, Gray and ASCII codes and the BASIC programming language may wish to omit sections 4.1 to 4.5. Section 4.6 covers a short summary of 6502 assembly language which may likewise be omitted if the reader has some prior experience of programming in this language.

4.1 NUMBER SYSTEMS

In the manipulation of data within a computer there are no margins for error and a simple two state numbering system is adopted. This is termed the binary system and it is based on an ON/OFF, HIGH/LOW, LOGIC ‘1’/LOGIC ‘0’ principle. The physical measure of the states is represented as voltage levels. Ideally for the semi-conductor integrated circuits in a microcomputer system, 5 V denotes a logic ‘1’ while 0 V denotes a logic ‘0’. In practice a tolerance band is adopted with say 2.4 V to 5 V representing ‘1’ and 0 V to 0.8 V representing ‘0’.

The micro-electronic devices in the system handle the transfer of information in 1s and 0s termed bits which is derived from binary digit. A group of eight bits is termed a byte. A number of microcomputers are referred to as 8-bit machines since they handle the transfer of data in 8-bit codes. 16-bit and 32-bit machines are also available.

Computers generally operate with three numbering systems — decimal, binary and hexadecimal. In order to communicate with external devices it becomes necessary to be able to translate between each number system. In a binary system, the only possible numbers are 0 and 1 and the base is chosen as 2.
Consider 8-bits:

<table>
<thead>
<tr>
<th>bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

most significant bit (MSB)       least significant bit (LSB)

Suppose we have the binary number:

```
1 0 0 1 1 0 0 1
```

The corresponding decimal number is:

```
128 + 0 + 0 + 16 + 8 + 0 + 0 + 1 = 153
```

The binary value for a decimal number of 37 using the above table, is:

```
0 0 1 0 0 0 1 0 i.e. (32 + 4 + 1)
```

It can be seen that the largest binary number represented by 8 bits is 11111111 = 255 in decimal. Or decimal values in the range 0 to \(2^8 - 1\) only can be accommodated as an 8-bit binary number, i.e. 0 to 255.

For decimal numbers larger than 255, two bytes can be used giving numbers in the range 0 to \(2^{16} - 1\) = 0 to 65535.

The decimal value corresponding to a 16-bit value can easily be obtained using the above 8-bit conversion technique as follows. Suppose we have the 16-bit binary number:

```
High byte          Low byte
0 0 0 1 1 0 1 0    1 0 1 1 0 1 0 0
```

This gives a decimal value of:

\[
((16 + 8 + 2) \times 256) + (128 + 32 + 16 + 4) \\
= 6656 + 180 \\
= 6836
\]

Note that there is a multiplying factor of 256 between corresponding bits of the high and the low byte, i.e. a binary value of 0000 0001 in the high byte corresponds to the decimal value of 256.

Conversion from decimal to 16-bit binary is performed by first dividing the decimal value by 256. For example, 50395 in decimal to binary:

```
50395/256 = 196 remainder 219
```

The 196 represents the decimal value associated with the high order byte and the 219 represents that associated with the low byte.

Thus the 16-bit binary representation is 11000100 11011011.