HIGHLY PROCESSED CODE

Our machine code course continues with the first in a series of articles that takes a close look at the Assembly language of the 6809 processor, which is used in the Dragon and Tandy Color home computers. We begin with an explanation of the role of the registers in the functioning of the processor.

A microprocessor can be regarded as having three main components: the registers, which are parts of memory inside the processor; an ALU (arithmetic and logic unit) where certain simple mathematical operations can be performed on the data stored in the memory; and a control unit that makes everything happen in the right sequence at the right time.

At its lowest level of operation, the microprocessor responds to voltage signals applied at some of its external connections to change its internal state (the contents of its registers) or to send and receive other signals. By representing the presence of a voltage as a one and its absence by a zero, we can think of these signals — travelling back and forth between the processor and memory, or within the processor itself — as numbers in binary notation. In this way, we can program the processor by applying a sequence of numbers as instructions for it to act on. The very lowest level of programming, therefore, involves thinking in terms of binary (or hexadecimal) numbers. It requires a knowledge of the effect of each number, or instruction code, on the processor.

An eight-bit processor, like the 6809, can send and receive binary numbers with eight bits, which gives a range of decimal numbers between 0 and 255. Many of the numbers are used to refer to addresses of memory locations, which on most eight-bit processors are given as 16-bit numbers (allowing for a range of memory locations between 0 and 65535). Of course, when dealing with these numbers, the processor is capable of transferring them only eight bits at a time.

6809 REGISTERS

The registers in a processor can take many forms, depending on their particular functions. Some are reserved for the internal use of the processor and cannot be accessed by the programmer. There are four main 6809 registers that the machine code programmer will use a lot.

The most commonly used register is the *accumulator*. This is where most of the data being used is stored and manipulated. For example, the usual function performed by an Assembly language ADD instruction is to add the contents of a specified memory location to the contents already stored in the accumulator. Thus, a new value will 'accumulate' in this register.

The *index register* is used to modify addresses so that we can step through tables and lists of data easily. When an instruction refers to a memory location using *indexed addressing* then the contents of this register are added on to the address given in order to specify the *effective address* of the data required. To step through a table of data, we have only to refer to the *base address* (of the first item of the table) and keep incrementing the index register. As the values stored in this register are normally addresses, index registers are generally 16 bits long, rather than eight.

The *stack pointer* is the register that indicates the location of the top of the *stack*, which is a convenient way of storing data and retrieving it quickly. The stack is used when it is necessary to save the internal contents of the processor (for example, when a subroutine is called) so that they can be restored later. The contents of some, or all, of the registers can be 'pushed onto' the stack and then 'pulled off' later when control returns to the main program. The stack pointer simply tells the processor the location of the last item put into the stack and where it can save or find the next item. Because they also refer to addresses in memory, stack pointers are generally 16 bits long.

The fourth register is very important, although its function is automatic most of the time. This is the *program counter*, which should always contain the address where the next instruction is stored. The processor goes to the location specified by the register, fetches the contents, interprets their meaning and acts on that instruction. Normally, the program counter will automatically be incremented as instructions are carried out, so that the instructions are fetched in sequence. Altering the contents of the program counter (by storing a new value there, or adding to or subtracting from the old value) will change the course of the program. In other words, this acts like a GOTO instruction, although at this level it would be referred to as a Jump (JMP) if a completely new address was provided, or a branch (BRA) if the current address was altered.

There is a fifth type of register, although it does not operate in the same way as the others. This is the *condition code* register, which is best thought of as a collection of individual bits, each representing some feature of the state of the processor. For example, one of these bits is used to signify to the processor whether the number