How does a microcomputer make a decision?

By David G. Larsen, Peter R. Rony, and Jonathan A. Titus

Mr. Larsen, Department of Chemistry, and Dr. Rony; Department of Chemical Engineering, are with the Virginia Polytechnic Institute and State University. Mr. Titus is *President, Tychon, Inc.*

ONE of the most important programming characteristics of any digital computer, including a microcomputer, is the ability to make a decision. For a typical microcomputer, we can define a *decision* as the process of determining further action based upon the logic state of a *flag. Aflag* is a single flip-flop that can be either set or cleared in response to operations occurring within the microcomputer system. A change of state of the flag is usually an indication either that a particular operation has been completed or that a certain condition exists as a result of a microcomputer operation. Flags can be located either internal or external to the microprocessor chip; the ones that we shall discuss in this column are the internal flags, which are set or cleared in response to specific types of microprocessor instructions such as arithmetic and logical instructions.

The flags that are located within the microprocessor chip are typically associated with the *arithmetic-logic unit (AL U],* a region within the chip where all arithmetic and logical operations are performed. In the 8080 microprocessor chip, for example, five flags indicate the following conditions:

Zero flag: If the result of an arithmetic or logical operation is zero, the zero flag is set to logic I; if nonzero, the zero flag is reset to logic O.

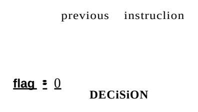
Sign flag: If the result of an arithmetic or logical operation is negative, the sign flag is set to logic I; if positive, the sign flag is reset to logic O.

Parity flag: If the result of an arithmetic or logical operation has even parity, the parity flag is set to logic I; if odd parity, the parity flag is reset to logic O.

Carry flag: If the result of an arithmetic or rotate operation has a carry out of the most significant bit of the 8-bit result, the carry flag is set to logic I; if not, the carry flag is reset to logic O. The carry flag is reset to logic 0 after all logic operations.

Auxiliary carry flag: If the result of an arithmetic operation has a carry out of bit 3 into bit 4 of the 8-bit result, the auxiliary carry flag is set to logic I; if not, the auxiliary carry flag is reset to logic O. The auxiliary carry flag is reset to logic O after most logical operations.

The following discussion is restricted to the zero flag. Shown below is the traditional flowchart *decision symbol* applied to an 8080 microprocessor decision



The next instruction that is executed depends upon the logic state of the flag that is associated with this specific decision. For example, consider the JNZ instruction, where JNZ means "jump if not zero":

Instruction code	Mnemonic
302	JNZ
<B2>	
<B3>	

Description

If the zero flag is at logic 0, jump to the 16-bit memory address given in bytes <B2> and <B3> of this three-byte instruction; if the zero flag is at logic I, ignore this instruction and proceed to the following instruction.

The statement, jump if not zero, refers to the 8-bit result of a preceding instruction, not the logic state of the zero flag. When this result is zero, the zero flag is set at logic I and program control passes to the next instruction, as shown in *Figure 1.*

The JNZ instruction is widely used in the creation of programmed

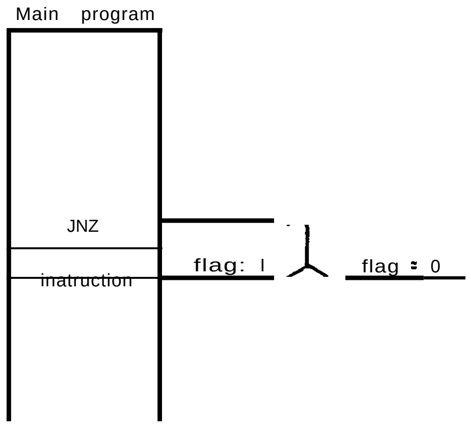


Figure 1 *The JNZ instruction. If the zero flag is at logic 1, the instruction is ignored and program control passes to the following instruction.*

Table 1

Microcomputer program demonstrating a simple time delay loop

LO memory address	Instruction byte	Mnemonic	Clock cycles	Description
000	006	MVI B		Move following timing byte into register B
001	*			Timing byte for register B
002	323	OUT 2	10	Generate device select pulse that sets the SN7474 flip-flop
003	002			Device code for set input to SN7474 flip-flop
004	005	DCR B		Decrement contents of register B by 1
005	302	JNZ	10	If zero flag is at logic 0, jump to the memory address given by the following two address bytes; otherwise, ignore this instruction
006	004			LO memory address byte
007	000			HI memory address byte
010	323	OUT 3	10	Generate device select pulse that clears the SN7474 flip-flop
all	003			Device code for clear input to SN7474 flip-flop
012	166	HLT		Halt the microcomputer

time delay loops, an example of which is provided in *Table I.* In this program, both the address and instruction bytes are in octal code; it is assumed that the HI memory address byte is 000. The program first moves an 8-bit timing byte into register B; this byte, indicated by an asterisk * has any value between 000 and 377. The value of the byte will determine the duration of the time delay.

At LO memory address 002, a device select pulse is generated to set the SN7474 flip-flop shown in *Figure* 2. The contents of register B are then decreased by I. The JNZ instruction immediately tests the logic state of the zero flag; if the contents of register B are not zero, the flag is at logic 0 and a jump occurs back to LO memory address 004. The DCR Band JNZ instructions are executed repeatedly until the contents of register B becomes zero, at which time the zero flag becomes logic I. The JNZ instruction tests the flag for the last time and shifts program control to the OUT 3 instruction at LO memory address 010. This output instruction generates a device select pulse that clears the SN7474 flip-flop. Once this has been done, the microcomputer comes to a halt.

The program shown in Table I generates a single output pulse the duration of which can take any value between 0.0125 and 1.925 msec in steps of 0.0075 msec and is

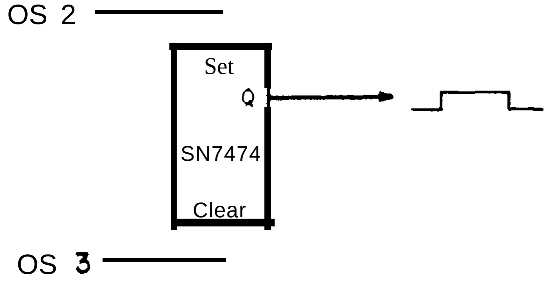


Figure 2 *SN7474 flip-flop used as a monostable multivibrator.*

determined by the timing byte at location 001, at the Q output of the SN7474 flip-flop. Some typical pulse widths are summarized in *Table* 2 for an 8080-based microcomputer that operates at a clock rate of 2 MHz. The calculations associated with the conversion of clock cycles to pulse widths have been discussed.' The number of clock cycles is a measure of the actual time that it takes the microcomputer to execute a single instruction or group of instructions. For a 2-MHz microcomputer, a single clock cycle has a duration of 500 nsec. The program in Table I and associated SN7474 flip-flop provide an example of what we mean by "the substitution of hardware by software," or, a simple program and a single flip-flop replace a much more complicated hardwired programmable monostable circuit.

Reference

1. *Bugbook III, Microcomputer Interfacing Experiments Using the Mark 80® Microcomputer, an 8080 System* (E & L Instruments, Inc., Derby, Conn., 1975).

Table 2

Examples of output pulse widths generated by the program in Table 1

Timing byte at LO **memory** address	Number of clock cycles	Pulse width, msec
000	3850	1.925
001	25	0.0125
002	40	0.02
003	55	0.0275
004	70	0.035
005	85	0.0425
010	130	0.065
020	250	0.125
050	610	0.305
100	970	0.485
200	1930	0.965
300	2890	1.445
350	3490	1.745
377	3835	1.9175