3.3.2 CLD--Clear Decimal Mode

This instruction sets the decimal mode flag to a 0. This causes all subsequent ADC and SBC instructions to operate as simple binary operations.

CLD affects no registers in the microprocessor and no flags other than the decimal mode flag which is set to a 0.

3.4 BREAK COMMAND (B)

The break command flag is set only by the microprocessor and is used to determine during an interrupt service sequence whether or not the interrupt was caused by BRK command or by a real interrupt. A more detailed discussion of BRK is in the interrupt section. This bit should be considered to have meaning only during an analysis of a normal interrupt sequence. There are no instructions which can set or which reset this bit.

3.5 EXPANSION BIT

The next bit in the flag register is an unused bit. It is most likely that this bit will appear to be on when one is analyzing the bit pattern in the processor status register; however, no guarantee as to its state is made as this bit will be used in expanded versions of the microprocessor.

3.6 OVERFLOW (V)

As discussed in the section on arithmetic operations, if one is to look at the binary arithmetic operations as signed binary operations, there needs to be some indication of the fact the result of the arithmetic operation has a greater value than could be contained in the 7 bits of the result. This bit is the overflow bit and during ADC and SBC instructions represents a status of an overflow into the sign position. The user who is not using signed arithmetic can totally ignore this flag during his programming; however, this flag has the same meaning as the carry to the user who is using signed binary numbers. It indicates that a sign correction routine must be used if this bit is on after an add or subtract using signed numbers.

In addition to its use to monitor the validity of the sign bit in ADC and SBC instructions, the overflow flag in the MCS650X products is dramatically changed from PDP11 and the MC6800. In those systems the overflow flag was very carefully controlled so as to allow certain signed branches for analysis of signed numbers. These branches have been deleted from the MCS6500 series because of confusion and difficulty often associated with using them, and so therefore, the overflow flag is applicable only to the operation of ADC and SBC, and then only when using signed numbers.

However, in order to maximize the effectiveness of this testable flag the BIT instruction which may be used to sample interface devices, allows the overflow flag to reflect the condition of bit 6 in the sampled field. During a BIT instruction the overflow flag is set equal to the content of the bit 6 on the data tested with BIT instruction. When used in this mode, the overflow has nothing to do with signed arithmetic but is just another sense bit for the microprocessor. Instructions which affect the V flag are ADC, BIT, CLV, PLP, RTI and SBC. On certain versions of the microprocessor the V bit will also be available for stimulus from the outside world.

3.6.1 CLV--Clear Overflow Flag

This instruction clears the overflow flag to a 0. This command is used in conjunction with the set overflow pin which can change the state of the overflow flag with an external signal.

CLV affects no registers in the microprocessor and no flags other than the overflow flag which is set to a 0.

3.6.2 Determination of Overflow

To briefly recap the concept of overflow detection, one must understand that the machine signals an overflow based on the data entered to the operation and the final result. Since, with signed arithmetic, the range of numbers that be represented is +127 to -128, the overflow flag will never set when numbers of opposite sign are added, since their result will never exceed that range. The machine deals with this by recognizing that for any 2 positive numbers, the "bit 7" of each is a "0" and that for any arithmetic operation yielding a result less than or equal to +127, the resultant "bit 7" must be a "0." If it is a 1, the overflow flag is set.

Similarly, when two negative numbers are added, the "bit 7" of each is a "1" and for any result yielding a value less than or equal to -128, the resultant "bit" must be a "1." If it is a 0, the overflow flag is set.

Therefore, the machine recognizes by knowledge of the "bit 7" of each of the numbers to be added what the resultant "bit 7" must be in a non-overflow situation. If these conditions are not met, the overflow flag goes set.

3.7 NEGATIVE FLAG (N)

As already discussed, one of the uses of the microprocessor is to perform arithmetic operations on signed numbers. To allow the user to readily sample the status of the sign bit (bit 7), the N flag is set equal to bit 7 of the resulting value in all data movement and data arithmetic. This means, for instance, after a signed add one can determine the sign of the result by sampling the N flag directly rather than finding a way to isolate bit 7. Although signs were the primary purpose for which the N flag was intended, its usefulness far exceeds that of strictly a sign bit. Because of every operation including simple moves and add operations the N bit is equal to the status of bit 7 as a result of the operation; its primary use becomes that of an easily testable bit. Almost all single-bit instructions, all interrupts and all I/O status flags use bit 7 as a sense bit. This allows the user to perform some type of memory access operation such as Load A followed by immediate conditional branch based on the status of bit 7 as reflected in the N flag. Like the Z bit, this flag is not settable or controllable by the programmer and represents the status of the last data movement operation. Instructions which affect the negative flag are ADC, AND, ASL, BIT, CMP, CPY, CPX, DEC, DEX, DEY, EOR, INC, INX, INY, LDA, LDX, LDY, LSR, ORA, PLA, PLP, ROL, BIT, SBC, TAX, TAY, TSX, TXA and TYA.

3.8 FLAG SUMMARY

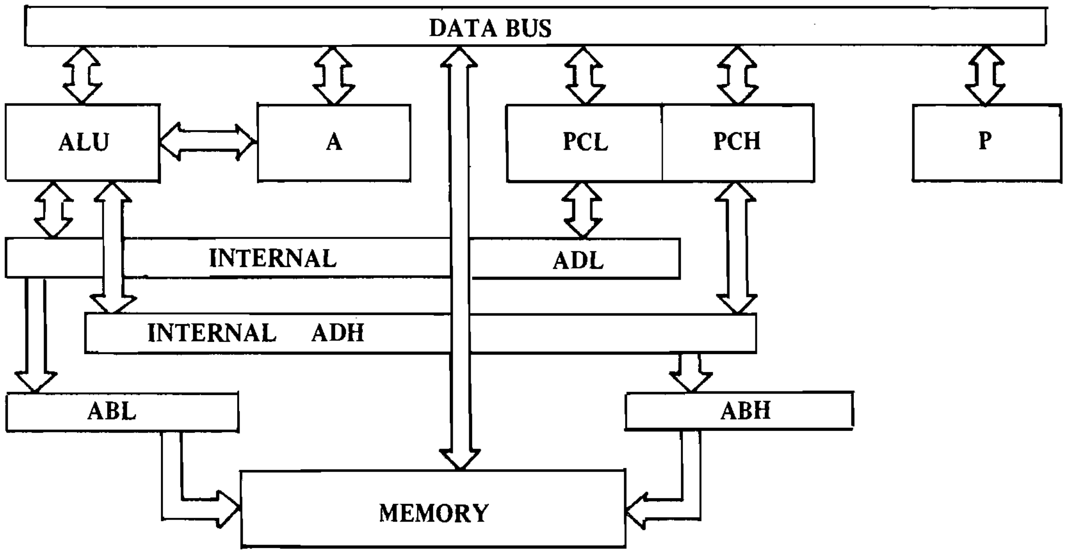
To summarize, the microprocessor treats a series of flags or status bits as a single register called the "P" or "Program Status" register. Some of these flags are controllable only by the programmer (such as the D flag); others are controllable by both the user program and microprocessor (such as the interrupt disable flag). Some of them are set and reset by almost every processor operation, such as the N and Z flags. Each of these flags has its own meaning to the programmer at a particular point in time. When combined with the concept of conditional branches, they represent a powerful test and jump capability not normally found in a machine of this magnitude. Other than perhaps the carry flag which is used as part of the arithmetic instructions, the flags by themselves have relatively little meaning unless one has the ability to test them. For this purpose there is a series of conditional branch instructions designed into the machine.

CHAPTER 4

TEST, BRANCH AND JUMP INSTRUCTIONS

4.0 CONCEPTS OF PROGRAM SEQUENCE

In all the discussions up until now, there has been little discussion about how the microprocessor understands the instructions used to perform various arithmetic and accumulator manipulations. However, it is appropriate that the concept of a program and how the microprocessor determines each instruction be developed. More registers are required in the machine as shown in the figure below.



Partial Block Diagram of MCS650X Including Program Counter and Internal Address Bus

FIGURE 4.1

Although two 8 bit registers have been added, they are the only registers in the machine that act as though they are one 16 bit register. They implement a concept known as program count or program sequence and subsequently their value will be referred to as PC or program count. In certain operations it may be convenient to talk about how one affects the program count low (PCL) which will be the lower 8 bit register or the program count high (PCH) which will be the higher 8 bit register. The reason for this register being 16 bits in length is that if it had only 8 bits it would only be able to reference 256 locations. Since it is through the address bus that one accesses memory, the program counter which defines the addressable location, should be as wide a word as possible.

The accessing of a memory location is called "addressing". It is the selection of a particular eight-bit data word (byte) out of the 65,536 possibilities for memory data locations. This selection is transmitted to the memory through the 16 address lines (ADH, ADL) of the microprocessor.

For a more detailed discussion of how an individual memory byte is selected by the address lines, the reader is referred to Chapter 1 of the Hardware Manual.

If the program counter was only 1 byte and if the bit pattern which allows the microprocessor to choose which instruction it wants to act on next, such as "LDA" as opposed to an "AND", was contained in one byte of data we could only have 256 program steps. Although the machine of this length might make an interesting toy, it would have no real practical value. Therefore, almost all of the competitive 8 bit microprocessors have chosen to go to a double length program counter. Even though some of the microprocessors of the MCS650X family do not have all of the output address lines necessary to allow the user to address 65K bytes of program (due to package pinout constraints), in all cases the program counter is capable of addressing a full 65K by virtue of it's 16 bit length.

4.0.1 Use of Program Counter to Fetch an Instruction

The microprocessor contains an internal timing and state control counter. This counter, along with a decode matrix, governs the operation of the microprocessor on each clock cycle. When the state of the microprocessor indicates that a new instruction is needed, the program counter (program address pointer) is used to choose (address) the next memory location and the value which the memory sends back is decoded in order to determine what operation the MCS650X is going to perform next.

To use the program counter to perform this operation correctly, it must always be addressing the operation the user wants to perform next. This operation may be an instruction or may be data on which the instruction will operate.

In the MCS650X family, the program counter is set with the value of the address of an instruction. The microprocessor then puts the value of the program counter onto the address bus, transferring the 8 bits of data at that memory address into the instruction decode. The program counter then automatically increments by one and the microprocessor fetches further data for address operation necessary to complete the instruction. In the simple example below,

Example 4.1: Accessing Instructions with the P Counter Value

P Counter*	Location Contents
0100**	LDA
0101	ADC
0102	STA

*Program Counter
**Hexadecimal Notation

one can see how the program counter is used to access the instruction sequence load A, add with carry, and store the result. In this example, the program counter would start out containing 0100. The microprocessor would read location 0100 by using the program counter to access memory and would then interpret and implement the LDA instruction as previously described. The program counter will automatically increment by one on each instruction fetch, stepping to 0101. After performing the LDA, the microprocessor would fetch the next instruction addressing memory with the program counter. This would pick up the ADC instruction, the add would then be performed, the program counter which has been incremented to 0102 would be used to address the next instruction, STA. The P counter incrementing once with each instruction is an oversimplified view of what actually transpires within the microprocessor.

The MCS650X processors usually require more than one byte to correctly interpret an instruction. The first byte of an instruction is called the OP CODE and is coded to contain the basic operation such as LDA (load accumulator with memory) and also the data necessary to allow the microprocessor to interpret the address of the data on which the operation will occur. In most cases, this address will appear in memory right after the OP CODE byte. This allows the microprocessor to use the program counter to access the address as well as the OP CODE.

The following example shows how the program counter picks up the instruction and the address of data located at address 5155.

Example 4.2: Accessing Data Address With P Counter Value

P Counter	Location Contents
0100	LDA
0101	55
0102	51
0103	Next Instruction

The OP CODE appears in Location Address 0100. The code for the 55 would appear next in Location Address 0101 and the 51 would appear in Location Address 0102, and the OP CODE for the next instruction appears in Location Address 0103. In this example, we see that the program counter is used not only to pick up the operation code, LDA, but is also used to pick up the address of the memory location from which the LDA is going to obtain its data. In this case, the program counter automatically is incremented three times to pick up the full instruction with the microprocessor interpreting each of the individual fetches as the appropriate data. In other words, the first