6502 Registers

The NMOS 65xx processors are not ruined with too many registers. In addition to that, the registers are mostly 8-bit. Here is a brief description of each register:

```
PC   Program Counter

     This register points the address from which the next
     instruction byte (opcode or parameter) will be fetched.
     Unlike other registers, this one is 16 bits in length. The
     low and high 8-bit halves of the register are called PCL
     and PCH, respectively.

     The Program Counter may be read by pushing its value on
     the stack. This can be done either by jumping to a
     subroutine or by causing an interrupt.

S    Stack pointer

     The NMOS 65xx processors have 256 bytes of stack memory,
     ranging from $0100 to $01FF. The S register is a 8-bit
     offset to the stack page. In other words, whenever
     anything is being pushed on the stack, it will be stored
     to the address $0100+S.

     The Stack pointer can be read and written by transfering
     its value to or from the index register X (see below) with
     the TSX and TXS instructions.

P    Processor status

     This 8-bit register stores the state of the processor. The
     bits in this register are called flags. Most of the flags
     have something to do with arithmetic operations.

     The P register can be read by pushing it on the stack
     (with PHP or by causing an interrupt). If you only need to
     read one flag, you can use the branch instructions.
     Setting the flags is possible by pulling the P register
     from stack or by using the flag set or clear instructions.

     Following is a list of the flags, starting from the 8th
     bit of the P register (bit 7, value $80):

     N   Negative flag

         This flag will be set after any arithmetic operations
         (when any of the registers A, X or Y is being loaded
         with a value). Generally, the N flag will be copied
```

```
            from the topmost bit of the register being loaded.

            Note that TXS (Transfer X to S) is not an arithmetic
            operation. Also note that the BIT instruction affects
            the Negative flag just like arithmetic operations.
            Finally, the Negative flag behaves differently in
            Decimal operations (see description below).

        V   oVerflow flag

            Like the Negative flag, this flag is intended to be
            used with 8-bit signed integer numbers. The flag will
            be affected by addition and subtraction, the
            instructions PLP, CLV and BIT, and the hardware signal
            -SO. Note that there is no SEV instruction, even though
            the MOS engineers loved to use East European abbreviations,
            like DDR (Deutsche Demokratische Republik vs. Data
            Direction Register). (The Russian abbreviation for their
            former trade association COMECON is SEV.) The -SO
            (Set Overflow) signal is available on some processors,
            at least the 6502, to set the V flag. This enables
            response to an I/O activity in equal or less than
            three clock cycles when using a BVC instruction branching
            to itself ($50 $FE).

            The CLV instruction clears the V flag, and the PLP and
            BIT instructions copy the flag value from the bit 6 of
            the topmost stack entry or from memory.

            After a binary addition or subtraction, the V flag
            will be set on a sign overflow, cleared otherwise.
            What is a sign overflow?  For instance, if you are
            trying to add 123 and 45 together, the result (168)
            does not fit in a 8-bit signed integer (upper limit
            127 and lower limit -128). Similarly, adding -123 to
            -45 causes the overflow, just like subtracting -45
            from 123 or 123 from -45 would do.

            Like the N flag, the V flag will not be set as
            expected in the Decimal mode. Later in this document
            is a precise operation description.

            A common misbelief is that the V flag could only be
            set by arithmetic operations, not cleared.

        1   unused flag

            To the current knowledge, this flag is always 1.

        B   Break flag
```

This flag is used to distinguish software (BRK) interrupts from hardware interrupts (IRQ or NMI). The B flag is always set except when the P register is being pushed on stack when jumping to an interrupt routine to process only a hardware interrupt.

The official NMOS 65xx documentation claims that the BRK instruction could only cause a jump to the IRQ vector ($FFFE). However, if an NMI interrupt occurs while executing a BRK instruction, the processor will jump to the NMI vector ($FFFA), and the P register will be pushed on the stack with the B flag set.

D Decimal mode flag

This flag is used to select the (Binary Coded) Decimal mode for addition and subtraction. In most applications, the flag is zero.

The Decimal mode has many oddities, and it operates differently on CMOS processors. See the description of the ADC, SBC and ARR instructions below.

I Interrupt disable flag

This flag can be used to prevent the processor from jumping to the IRQ handler vector ($FFFE) whenever the hardware line -IRQ is active. The flag will be automatically set after taking an interrupt, so that the processor would not keep jumping to the interrupt routine if the -IRQ signal remains low for several clock cycles.

Z Zero flag

The Zero flag will be affected in the same cases than the Negative flag. Generally, it will be set if an arithmetic register is being loaded with the value zero, and cleared otherwise. The flag will behave differently in Decimal operations.

C Carry flag

This flag is used in additions, subtractions, comparisons and bit rotations. In additions and subtractions, it acts as a 9th bit and lets you to chain operations to calculate with bigger than 8-bit numbers. When subtracting, the Carry flag is the negative of Borrow: if an overflow occurs, the flag will be clear, otherwise set. Comparisons are a special case of subtraction: they assume Carry flag

```
            set and Decimal flag clear, and do not store the
            result of the subtraction anywhere.

            There are four kinds of bit rotations. All of them
            store the bit that is being rotated off to the Carry
            flag. The left shifting instructions are ROL and ASL.
            ROL copies the initial Carry flag to the lowmost bit
            of the byte; ASL always clears it. Similarly, the ROR
            and LSR instructions shift to the right.

    A       Accumulator

            The accumulator is the main register for arithmetic and
            logic operations. Unlike the index registers X and Y, it
            has a direct connection to the Arithmetic and Logic Unit
            (ALU). This is why many operations are only available for
            the accumulator, not the index registers.

    X       Index register X

            This is the main register for addressing data with
            indices. It has a special addressing mode, indexed
            indirect, which lets you to have a vector table on the
            zero page.

    Y       Index register Y

            The Y register has the least operations available. On the
            other hand, only it has the indirect indexed addressing
            mode that enables access to any memory place without
            having to use self-modifying code.
```