I found this document somewhere on a gopher archive in an Apple2's programming directory and I think this could be very useful information for example for emulator writers :)

2-Nov-1994 Ivo van Poorten <ipoorten=www@cs.vu.nl>

With all the books on 6502 programming, and all the years that the 6502 has been around and popular, you'd think that any quirks would have been well documented by now. But nooooo! So... for those of you involved in assembler or machine-language programming, here are some 6502 booby-traps -- those marked with * are supposedly fixed in the CMOS parts such as 65C02 and 65C102 (but not the C-64's 6510).

- Return address pushed on the stack by JSR is one less than actual next instruction. RTS increments PC after popping. RTI doesn't.
- The status bits pushed on the stack by PHP have the breakpoint bit set.
- The D (decimal mode) flag is not defined after RESET.
- The D (decimal mode) flag is not cleared by interrupts.
- The ADC and SBC instructions don't set N,V, and Z status bits if decimal mode is on. C status is set correctly.
- An indirect JMP (xxFF) will fail because the MSB will be fetched from address xx00 instead of page xx+1.
- If an interrupt occurs on a BRK instruction, the breakpoint is ignored.
- The ROR instruction didn't exist in the very earliest (pre-'77) chips.
- Undefined op-codes do strange things, some lock up the CPU.

Unlike most microprocessors, the 6502 does not make memory accesses on an "as needed" basis. It always does a fetch or store on every single clock cycle. There are a few cases, though, where there isn't anything to be fetched or stored, and a "garbage" fetch or store occurs. This is mainly of importance with the memory-mapped I/O devices:

- When adding a carry to the MSB of an address, a fetch occurs at a garbage address. The CMOS chips refetch the last byte of the instruction.
- When doing a fetch-modify-store instruction (INC, DEC, ASL, LSR, ROL, ROR) garbage is stored into the location during the "modify" cycle... followed by the "real" store cycle which stores the correct data. The CMOS chips do a second fetch instead of a garbage store.

These aren't really "bugs", but there are some instructions that just seem like they ought to be there, but aren't:

- INC A and DEC A.
- BRA relative. Unconditional branch.
- BIT #immediate. The CMOS chips also have BIT absolute,X and BIT zeropage,X.
- STX absolute,Y and STY absolute,X. How come LDX and LDY can use these addressing modes, but the matching store instructions can't?
- SEV. Set overflow bit. Probably not really useful, but you can set and clear all of the other status bits, and there *is* a CLV.
- Personally, I'd also like to have "Clear A" and "Test A" instructions. These would be twice as fast and half as big as LDA #0 and CMP #0. Ironically, the CMOS chips have an STZ (Store Zero) instruction which can clear a memory location, but still can't clear the Accumulator.
- I'd also like to have BSR relative. A subroutine call that's relocatable.
- Fixed in the CMOS versions (65C02, 65C102, etc.)