**UNSTABLE UNDOCUMENTED OPCODES**

Posted July 2nd, 2016

http://forum.6502.org/viewtopic.php?f=4&t=3493&p=46165&hilit=Aptly+named+UFOs#p46165

I wanted to try to summarize where I'm at with these aptly named UFOs - particularly the "unstable" undocumented ones. I've learned more about these in the past couple of weeks, and I must say, my original hope of finding some clear and deterministic behaviour for these beasties has proven to be a little naive. There is just too much variability to contend with. The reasons are fascinating, but the net result is that at best, any one behaviour will work in some instances and fail in others. In fact, no 6502, commercial or otherwise, is immune here. So the good news is that we are in effect doing no worse by simply settling on one set of "common" behaviours and calling it a day. And this is exactly what I have chosen to do. I'll just mention, though, that the "Stable" opcodes not listed below all do behave as documented and are cycle accurate. The "unstables", on the other hand, I have decided to treat as follows:

1) (& H + 1) Group: AHX($$9F, $93), SHX($9E), SHY$(9C), TAS($9B) - these opcodes exhibit two instabilities - the (& H + 1) operation is "sometimes" dropped and the high-byte of the target address is corrupted on page-crossings in some cases. My own tests on a VIC20 seem to show that page-crossings that stay within the lower nibble of the high-byte appear to work correctly, but it's hard to be definitive about this. In the end, I have decided to honour the (& H + 1) construct in all cases and ignore any page-crossing anomalies.

2) MAGIC Constant Group: ALR($4B), ARR($6B), XAA/ANE($8B), LXA ($AB) - these are dependent on a so-called MAGIC constant which in fact varies across systems and in many cases even within specific systems at different times. Some claim that of these, only $8B and $AB are in fact unstable, but my own tests showed differences do exist with the others as well. There simply seems to be no right answer for the value of MAGIC. So, I have somewhat arbitrarily chosen to use specific values as a convenient and plausible solution knowing full well this will not work in all instances. With the chosen values, these opcodes reduce to the following functions:

ALR($4B) = A <- A & (CONST | #IMM), A <- LSR A, where (CONST = $00) becomes A <- A & #IMM, A <- LSR A
ARR($6B) = A <- A & (CONST | #IMM), ROR A, where (CONST = $00) becomes A <- A & #IMM, ROR A
XAA($8B) = A <- (CONST | A) & X & #IMM, where (CONST = $FF) becomes A <- X & #IMM
LXA($AB) = A,X <- (CONST | A) & #IMM, where (CONST = $FF) becomes A,X <- #IMM

Tests on my VIC20 showed that $8B seems to remain stable at $FF through thousands of iterations, but that certainly has not been what others have seen. In addition, with LXA($AB), the VIC20 showed $EE as a stable value, rather than the more common $FF which I have chosen to use.

Despite all this, I do feel I've landed in a good place: I can claim cycle-accurate and reliable operation for the 96 stable undocumented opcodes, and have a reasonable expectation regarding the 9 "unstables" that remain. Namely, that software that properly accounts for the variabilities in the unstables will run without a problem. This can be done by not relying on a given value for MAGIC and keeping page boundaries well away from "& H + 1" instructions, for example. Otherwise, all bets are off on this as well as other 6502 systems.

Time to move on to other challenges :)