Appendix F: wAx 2.1 Addendum

wAx 2.1 includes the following changes:

- Go (the G tool) improves stack integrity for post-BRK continuation.
- The Register Editor can now be used to change the PC, to aid the aforementioned post-BRK continuation.
- The Register Editor now provides syntax to keep any register as it was, without providing a value.
- The Register display now includes visual representation of processor flags, in addition to the processor flag byte value.
- Variable substitution for addresses now allows hexadecimal strings.
- Two "illegal" instructions' mnemonics and byte sizes have been changed.
- A buffer overwrite bug, which could prevent partial filename search from working in emulated environments (e.g., VICE) has been remediated.

I hope you enjoy these updates. Read on for additional information.

Be Excellent To Each Other,

Jason
October 2023

Go

```
.G
```

When the G tool is invoked with no address, execution continues at the address specified in the Register Editor. The return address to BASIC is not put on the stack with this syntax; rather the stack is left as it was after the last BRK point.

Example

```
.A 1800 LDA #$55
.A 1802 PHA
.A 1803 LDA #$44
.A 1805 BRK
.A 1806 NOP
.A 1807 PLA
.A 1808 BRK
.G 1800
```

This will push $55 onto the stack and change A to $44, at which point, execution will break, showing A to be $44 and PC to be $1807 (two bytes after the BRK).

If you then continue execution with G alone, the $55 is pulled from the stack into A.

Register Editor

Registers may now be set with

```
.;ac [xr] [yr] [pr] [sp] [pc]
```

where:

ac is the Accumulator as a valid hexadecimal byte, or --

xr is the X Register as a valid hexadecimal byte, or --

yr is the Y Register as a valid hexadecimal byte, or --

pr is the Processor Status Register as a valid hexadecimal byte, or --

sp is the Stack Pointer as a valid hexadecimal byte, or --, but has no effect on the Stack Pointer

pc is the Program Counter as a valid 16-bit hexadecimal address

For any register, you may enter -- instead of a hex byte. Doing this will leave the register's value unchanged.

You may change the Program Counter before issuing G alone (*see Go, p. 2*).

Note: The Stack Pointer is not changeable because there is no corresponding pre-SYS byte available for it. Arguments in the *sp* position are ignored.

Processor Status Display

The Register display now includes on/off annunciators for five processor flags:

- The Negative flag (N)
- The Overflow flag (V)
- The Decimal flag (D)
- The Zero flag (Z)
- The Carry flag (C)

When an annunciator is in reverse text, the corresponding flag was set at the end of execution. Otherwise, it was clear.

Note: wAx's post-execution code clears the Decimal flag because the BASIC environment cannot function if the Decimal flag is set. If the Decimal flag's annunciator is "on" in the Register display, it will be set at the next invocation of the G tool.

BASIC Variable Substitution

BASIC string variables may now be used in tools where only numeric variables were previously permitted, provided they are valid hexadecimal addresses, and provided they are the correct length for the context.

Examples

```
S$ = "F27A"
E$ = "F28A"
.D `S$` `E$`

A$ = "033C"
.* `A$`
```

Note: This type of substitution *can not* be used within the A tool (Assembly or Memory Editors).

"Illegal" Instruction Changes

Two of the illegal instructions have changed mnemonics and lengths. These are:

- **$34** DOP zero page,x is now SKB (*skip byte)*
- **$3C** TOP absolute,x is now SKW (*skip word)*

Although the 6502 treats these as two- and three-byte instructions, respectively, wAx will assemble and disassemble (via the E tool) them as immediate mode (one-byte) instructions.

This allows SKB and SKW to be used in selectors:

```
.A * @P LDA #"%"
.A *    SKW
.A * @C LDA #","
.A *    SKW
.A * @A LDA #"&"
.A *    JSR $FFD2
```