Crosswize: ZX Spectrum. Self modifying Z80 code and the stack – a powerful combo!

The other day, I came across source code for Crosswize, a ZX Spectrum game I wrote back in the 80's. Crosswize was written in Z80 assembly (of course), and featured a smooth-scrolling background, accomplished through a goodly amount of self-modifying code, along with the discovery that the fastest way to read and write to memory on the Z80 is to use the stack (if you can free up enough register pairs that is). The reason the stack is so fast is because the "push" instruction is encoded in only a single byte which specifies the 16 bit register pair to push (the source), and which naturally decrements the stack pointer before writing to memory when executed. Similarly, loading register pairs (using "pop") from the stack is also fast, and that approach is also used here.

In the following code snippet (which is responsible for drawing 8 pixel rows of the smooth scrolling background), by the time the code gets to the label "push_00", the register pairs AF, BC, DE, IY and IY have been loaded with pre-shifted bitmap data, and the register pair HL cleared. In a previous setup routine, the block of .db 0 starting @ label push_00 has been pre-populated with (using self-modifying code) a series of "push" instructions. The stack pointer (sp) is set to the byte just after the right edge of a screen pixel row. When the Z80 CPU executes the code following push_00, 16 bit values (corresponding to 16 pixels on the ZX Spectrum) are written to the screen from right to left (a push instruction first decrements the stack pointer and then writes the 16 bit value to the new location).

I had a struggle to find enough registers (as may be evident from the code), so ended up using push ix and push iy opcodes (which have an extra 0xdd or 0xfd byte prefix), so the "jp 0" at the label skip_00 is self-modified to jump into the correct spot in the buffer to allow for all combinations of 1 and 2 byte opcodes.

This code is rife with self-modifying code, a technique that is mostly obsolete today, but which was a handy tool for many Z80 coders back in the day.

```
ld hl, 0 ; SMC - source data
ld c, 2</code>

section_00:
nop ; SMC
ld b, 8

scan_00:
ld sp, hl
ld hl, 0Ah
add hl, sp
exx
pop af
pop bc
pop de
pop iy
pop ix
exx
ex de, hl
ld sp, hl
ex de, hl
exx
ld hl, 0 ; clear pixels

skip_00:
jp 0 ; SMC

push_00: ; push buffer SMC
.db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
.db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

exx
inc d
djnz scan_00

ld de, 40BFh
dec c
jp nz, section_00
```

Share this: