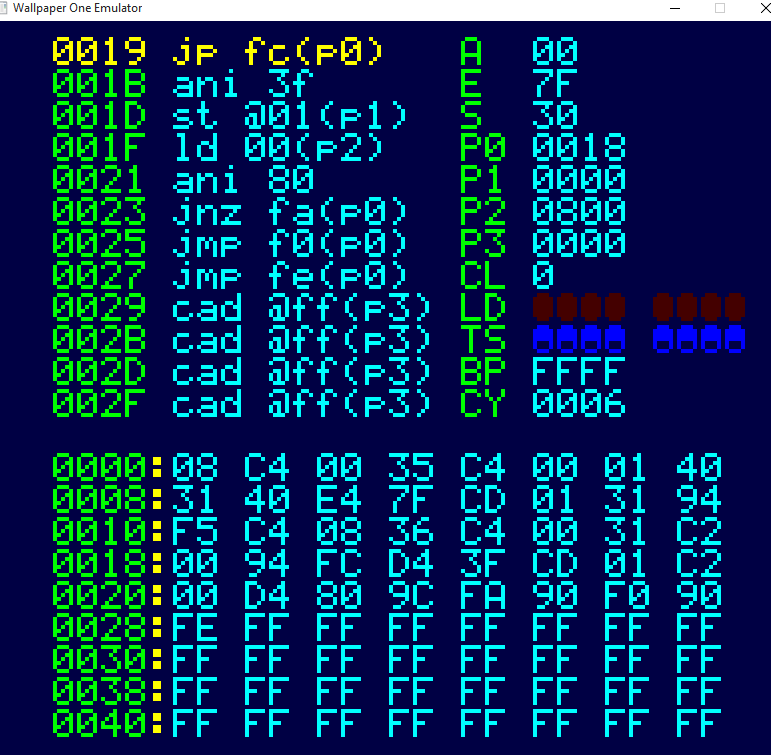
**Commit : Processor directory, Miscellany ST7920 Driver**

****Emulation is coming along quite rapidly.

The main reason for this is I already have a framework which does a fair chunk of the work, and I can cannibalise code from emulation of a different machine.

Almost all the new code is the actual CPU instructions themselves. I’ve done this via a definition/generation approach, which uses a Python script to generate the often very repetitive code, which makes errors less likely and easier to find, because they would be repeated.

The other change is an SC/MP oddity, in the fetch cycle. Most processors fetch an instruction and execute it. On the SC/MP it’s the other way around, you increment the PC and then fetch the instruction. So the first instruction is actually at location $0001. This meant a little bit of tweaking but nothing terrible.

It seems to work fine. The screenshot shows code (left) processor (right) and memory (bottom). LD and TS are the output LEDs and input toggle switches, BP breakpoint and CY cycle counts. The rest are the standard SC/MP registers

It is also designed with the physical hardware port in mind. I won’t do a physical port of the basic machine because it’s more difficult to build than the advanced one and you can’t do anything with it. The advanced one is a fairly simple build ; I am using an Arduino Mega2560 (because it has lots of static RAM, 8k in total), an ST7920 128x64 GLCD, a piezo buzzer for the sound and a PS/2 adaptor for the keyboard, which itself has a USB to PS/2 adaptor plugged in, which has a keyboard plugged into it.

I can also cannibalise a GLCD driver I wrote. Things like u8glib and so on are very good libraries, but don’t actually do what I want, which is to blat on the screen as quickly as possible, so I originally took a fairly advanced ST7920 driver and cut it back to the raw basics, just plotting a single word on the screen – an ST7920 draws on its screen in horizontal lines of 16 bits. With this the GLCD won’t slow the system down.

The rest of it is standard Arduino libraries – the PS/2 keyboard driver and tone() function.