MAR
30

ANOTHER COUPLE OF Z80 SECRETS

Oh hii, so did you know about this one too? Here's a good little trick I just used in the Sine Flood routine:

```
    MainLoop:
                call UpdateAnim

                ld (PrintLoopRestSP+1),sp
                ld sp,TableBuffer

                ld b,Depth
    PrintLoop:
                pop hl        ; <---- What's this doing here?

                push bc
                call PrintRaster
                pop bc
                djnz PrintLoop

    PrintLoopRestSP:
                ld sp,0

                jp MainLoop

                ds 20
    TableBuffer:
                ds Depth*2
```

Okay I simplified the loop to have CALLs in it where I might just plough on with the action code. But anyway. On first glance that extra POP HL looks wrong. But wait! If we have a look at UpdateAnim, we see it develops a table of data that serves up 2 bytes once per raster (=scanline). So the extra POP HL actually retrieves needed data from the TableBuffer - in actual fact it's a table of source addresses for the graphic. Good eh?

And on top of that, because it is rebuilt each time we loop through MainLoop, we can also keep using the same stack for 'usual' stack stuff. We won't corrupt the Table, I just needed to have that DS 20 before itso that the previous code isn't accidentally corrupted by too many PUSHes.

Very neat. So that's one way of using the stack to serve data efficiently.

Let's have another use while we're at it. This is a way to efficiently control program execution. Let's say I needed to do a whole list of little things but didn't want to have a control program to do the assembler version of an IF… statement (or SELECT CASE, or whatever your favourite flavour of high-level language prefers). Here's how we can use the stack to jump straight to the right routines in a row… in fact, they join each other up:

```
StackedRoutines:
                ld (RestoreStack+1),s
                ld sp,ListOfRoutineAddrs
                ret
    RestoreStack:
                ld sp,0
                …

    ListofRoutineAddrs:
                dw FirstRout, SecondRout, ThirdRout, RestoreStack
```

This all looks very odd. Let's look in more detail:

First the actual stack is stored before a temporary one is assigned. Then what happens? We hit a RET. The stack POPs off the first address of its routines and jumps to it. This routine ends in a RET so it POPs off the next Word and jumps into the second routine. And so on, until finally passing control back to the main program. Very efficient, and with a couple of great side effects:

* If you don't use the stack in the routines, you can keep that fake stack and run it again and again.

* If you have many common paths through some routines, you could keep them all in a list and just LDI the instructions into the fake stack area and run them all.

I like this system because it reminds me of some kind of scripting language, or compilation. As with using the stack to efficiently move sprite data around, it splits the compile and execute sides up, which means you can optimise the execution phase of the cycle. Also, you could extend the stack to having routine arguments in it as well, so that a common routine could POP these back before doing something.

This is a really powerful idea and one that's worth developing further. What else can I use it for??

Incidentally, the Sine Flood routine was looking passable at 50hz but was *much* nicer when I took it down to 25hz but pixel accuracy. This is shaping up. Might have to leak an animated GIF if I get the time.

Posted 7 years ago   3 notes   0 Comments

z80, 8-bit, SAM Coupé,

ABOUT
COOKING CIRCLE

Occasional programming adventures on SAM Coupé, mostly

HOME
FLAPPY BIRD
TETRIS
XOR
2048 GAME
ATIC ATAC
BITJOCKEY
CELESTE
FIRST GHOST
GALAXIANS
OUTRUN
PYSGOTA YN OL
R-TYPE
SPIKER
SPLITTING IMAGES
SID CHIP TESTS
THRUST
ASK ME ANYTHING
ARCHIVE
RSS FEED

samelite liked this

drryanboyd liked this

stewardsebas reblogged this from cookingcircle

cookingcircle posted this