Bootloading Z80 using NMI, RESET and CLOCK.

Z80 system without ROM. Using NMI, RESET and CLOCK to bootload.

An NMI will push a value on the stack. The value depends on the instruction address at the time where NMI is given. At beginning is the memory contens unknown, and it is only possible to know the address immidiately after NMI is given (0066H). Because instructions is unknown, will it not be possible to know the address later.

The memory will need to be filled with a known value to control it.

To fill out memory with known values, is a lot of NMI's given until all memory is filled by pushing 0067H to stack.

The value may not actualy be 0067H because some instruction is more than one byte. Jump and restarts cause another address too. But in nearly all cases, will the instruction pushed on the stack not be an instruction that change the address (jump or restart). And when modifying the instruction at 0066H will it cause 0067H to be stored on stack. Now 0067H is stored. Number of cycles to fill memory:

Any instruction at 0066H:	5+3+3+23 cycles.
The 4-cycle instruction after modifying 0066H:	5+3+3+4 cycles.
Total cycles of fill:	ramsize * 49 cycles.

The fill is done by relasing the clock and running at 4MHz.

The value 0067H is a 4 cycle load and a 4 cycle nop instruction. Now will the known number of clocks to next instruction after 0066H be known. An NMI pushes the instruction address on stack, and set the instruction pointer to 0066H. The clock is now controlled to the address that should be pushed next.

Now is it possible to load the code by pushing values on the stack by an NMI at the address with the value that should be pushed into memory. Everytime NMI is given is instruction pointer set to 0066H by the processor.

It is only possible to use instructions, with instruction code less than the address where the code is stored. If the stack is at top of memory, then most instructions are accessable. If memory is small is less instructions accessable. To handle this, will it need to be loaded a bootloader. This bootloader use instructions from 0067H to 0100H+$ only, where $ is address of code to store, and the bootloader allow to be runned on a small system with 256 bytes only.

Before loading bootloader, is stack set to the destination address. The LD SP,n instruction may be placed at any address in memory. It need two pushes, and it is importent that first push not desturb the timing. The push, will need to store instructions that is four cycles, even they are intended for loading the stack. The stack load operation has more cycles, but it will not desturb timing, since it is last word to store. SP is now known after the memory has been executed. This is done by running the clock at 4MHz.

The LD SP,n is somewhere in memory, and the addresses used for pushing values by NMI, need to be known. A fill by making lot of NMI's will push 0067H from SP and down. A typical SP is 402H. (0404H + 0231H stored, where 0404H is two 4 cycles INC B instructions.). NMI is given until SP is down at 0066H. The 67H 00H instructions is stored on stack, and will cause the number of clocks to be known. Now is numbers between 0067H to 0400H able to be pushed on stack.

First a special bootloader is saved into memory. The bootloader only use instructions from 0067H to 0113H. It is possible to use on a small 256 byte system (8085 + 8155 only). The bootloader is stored at 0000H to 001BH and a 256 byte system use address 00H..1BH for instructions in range 0100H to 0100H+$. These instruction need to be stored before the address is used. Address is stored from top, and the instructions that use 0100H to 0113H is not in 0000H to 0013H address room.

The bootloader uses reset, and clock to store (or read) data. Operation depends on instruction at reset.

The bootloader is not critical to which value to store, and it is possible to change, modify, read etc. A15 is used to return with bit's from bootloader.

As far as I see may the methode be used for a wide range of CPU's. I have only testet it on Z80A, but more processors may be able to bootload using the methode.

The code, and the schematic is attached to this document.

Circuit
Oscilator
Pascal Source to control - use lpt port
Photo
Download all the files

Systems Without Memory | Running unknown look-up's | DRAM

This page is made by
E-Mail address: Jens.Madsen@post3.tele.dk
HomePage of Jens Madsen