In the src directory is an emulator for 8080 processor that I wrote in c++. All the emulator source files begin qkz80

The main2.cc is an attempt to use the emulator to create an emulator for the CP/M 2.2 operating system. You need to write a much better version of main2.cc.

In the cpmdocs directory are specifications summaries of the CP/M 2.2 os. Original manuals, original source code (in PL/M) and commented disassemblies are available on the web. Note however that sophisticated programs like mbasic infer a lot more from the documentation than is obvious, see mbasic below

Character input input and output go to unix character input and output. For disk IO it should translate open/close/read/write/directory scan/delete to unix operations. That is to say when a program running in the emulator dose a BDOS call to open a file we need to open the corresponding unix file. At the BIOS level there are calls for block level disk access that do not need to be emulated. Those are fore the real CP/M os to use to implement BDOS.

In /home/wohl/z80/tnylpo/ is a program that does the same thing I am asking you to do. However, tnylpo takes a much simpler view of the file emulation. In CP/M a filename has an 8 byte name, and 3 byte extension (padded out with spaces, when past to open a file). All upper case. If a program asks tnylpo to open FOO.BAS (which would be passed as name 'FOO ' extension 'BAS' and the unix file exists it would open it. But unix files tend to have lower case file names. Also they tend to be located in other directories. We recently made an implementation of the CP/M era basic interpreter mbasic. In testing with the real mbasic it could be run as 'tnylpo mbasic.com FOO.BAS' but the test basic programs are organized in directory trees. It would be better if a unix file passed on the command line (or a config file?) could specify long unix paths to map into a 8.3 filename. Feel free to examine tnylpo for how to do things you don't know how to do. However, once you know what the emulated program is asking, ignore the higher level filename mapping in tnylpo, as we need to do a much better job.

In the mbasic_src are the source files for mbasic.com. They may be of help to see what the the progam is doing. In CP/M a program loads into the TPA (transient program area) starting at 100H and running up the the start of the CP/M os. Trap and access (read, write, jump) out of this area to see what needs to be emulated. Not all is obvious from the above CP/M docs. For example, as mbasic starts in init.mac just after label init: it does:

```
 lhld  cpmwrm+1    ;get start of bios vector table

 lxi b,0+4    ;csts
 dad b    ;add four
 mov e,m    ;pick up csts address
 inx h
 mov d,m
 xchg      ;get csts address
 shld  const3##+1    ;third control-c check
```

In the cpmdocs it says you do a JMP 0 to quit the program and return to the OS. JMP is a one byte instruction at location 0. So locations 1 and 2 are the destination to jump to. That is all as documented. What is not clear from the docs and which mbasic uses, is that jump is into the BIOS to reload the CCP and return to the OS. The above code loads cpmwrm (which is 0 for cpm warm start) and does some math on it to get the address if a routine in the bios jump table to check for available character input that mbasic calls to check for ^C while running. So for things mbasic does like open / close a file the docs are fine. But trap any access like jump into bios and figure out what it is trying to do.

Not all access outside the TPA is mysterious, in low memory there a default FCB (file control block), default DMA (file transfer buffer), and the command line passed in to the program as defined in the low memory doc.

When starting the .com program pass in the documented register. I believe some reasonable stack pointer is set.

When a program does disk IO (open a file etc) and there is an IO error (on the floppy disk) there are a copy of error handling addresses in the BDOS or BIOS that are called. Normally that would ask the user if they want to retry. Some programs store these address and replace them with a different handler.