**Recompiling CP/M 2.2 from the sources**

Preparation required for z80pack release 1.8:   
Before you do anything else, you need to get your drive i: harddisk that came with **z80pack** out of the way, the 4 disk drives weren't enough for a setup:

cd ~/z80pack-x.y/cpmsim/disks

mv drivei.cpm library/hd-tools.dsk

This is not required with **z80pack** releases later than 1.8, the disk setup in the distribution is done appropriate already.

Download and Installation of the disk images:   
[Download](ftp://www.unix4fun.org/z80pack/cpm2src.tgz) the archive with the disk images and unpack the archive under ~/z80pack-x.y/cpmsim with **tar xzvf cpm2src.tgz**. Type **cpm2src** to mount the disks and start the emulation, the script checks if the drives c:, d: and i: are empty, before formating and mounting disks.

Drive a: contains a CP/M 2.2 system, the ISIS interface to CP/M 2.2 and the necessary tools. Also the ISIS assembler **asm80** is on drive a.   
Drive b: contains the ISIS PL/M-80 compiler, linker and tools needed to compile and link the CP/M programs written in PL/M-80.   
Drive c: is the scratch disk used by the build scripts.   
Drive d: is the destination disk, we are going to build a bootable CP/M 2.2 from the sources on this drive.   
Drive i: includes the CP/M 2.2 sources.

The ISIS environment included on the both tool disks is the minimal required set for building CP/M 2.2, due to drive size limitations. More complete ISIS tool disks are available as separate download.

Compiling the transient programs:   
The new OS on drive d: is build with a few batch commands:

submit cmd1

submit cmd2 <- this will break ed, don't execute, needs to be fixed

submit cmd3

submit cmd4

submit cmd5

Compiling the kernel:   
For the kernel (MOVCPM) there are two choices:

submit kernel-m build for Intel MDS800, as it was distributed by DRI

submit kernel-s build for Z80 simulator, you probably want this

Creating a bootable disk:   
Use the following commands to relocate the kernel and write it onto system tracks of drive d:

d:movcpm 64 \*

save 34 d:cpm64.sys

d:sysgen d:cpm64.sys

type d for destination drive, followed by two CR

**NOTE:**

The new build **movpm** program might not work on your current system, DRI used a lot of serialization code in cpmove.asm to verify the kernel in movcpm.com with the current running one. The program will run into a HALT opcode which will abort the emulation, in case the versions don't match. To overcome this problem run **movcpm** under control of **ddt** and patch out the jumps to the error handler:

ddt d:movcpm.com

DDT VERS 2.2

NEXT PC

2700 0100

-s234

0234 C2 0

0235 5A 0

0236 02 0

0237 02 .

-s2cb

02CB C2 0

02CC 5A 0

02CD 02 0

02CE 23 .

-i64 \*

-g

CONSTRUCTING 64k CP/M vers 2.2

READY FOR "SYSGEN" OR

"SAVE 34 CPM64.COM"

To try out the new disk leave cpmsim, move the disk image from drive d: to drive a: and start cpmsim, if everything went ok CP/M 2.2 will boot.

Patched CP/M 2.2 source disk:   
There is a second harddisk image available for [download](ftp://www.unix4fun.org/z80pack/cpm2src-pat.tgz) with some of the official DRI patches applied to the sources. Because this archive includes the harddisk image and shell script to mount disks only, the original source archive must be installed first.

|  |  |  |
| --- | --- | --- |
| **Patches applied so far** | | |
| Note/Patch | Files affected | Description |
| CPM22APN.02 | OS3BDOS.ASM | make RUBOUT identical to BACKSPACE |
| CPM22PAT.03 | SUBMIT.PLM | Create $$$.SUB always on A:, so that a submit job can run from any drive |
| CPM22APN.12 | SUBMIT.PLM | Do not convert input to upper case. Try i:subtest.sub for example |

In the sources the original code is commented out and the modification is commented with the patch number in this form: \*\*\*CPM22APN.02\*\*\*.

Building a CP/M 2.2 kernel for other systems:   
This can easily be done by using kernel-s.sub as a start. Modify the script to use boot and BIOS code for the destination system, instead of using boot.z80 and bios.asm for the Z80 emulation. The boot code needs to be originated at the address, where the boot ROM loads it into memory, **movcpm** doesn't relocate the boot code. A custom BIOS needs to be originated at address 1600H for the relocation.

Questions and Answers:

|  |  |
| --- | --- |
| Q: | Why are there relocation errors when building ddt, xsub and movcpm? |
| A: | The **genmod** utility normaly is used to build page relocatable binaries. The program is abused a bit, by building executables including a non relocatable module, the loader, and a relocatable module. The relocation errors are warnings about the non relocatable parts of the programs and can be ignored. |