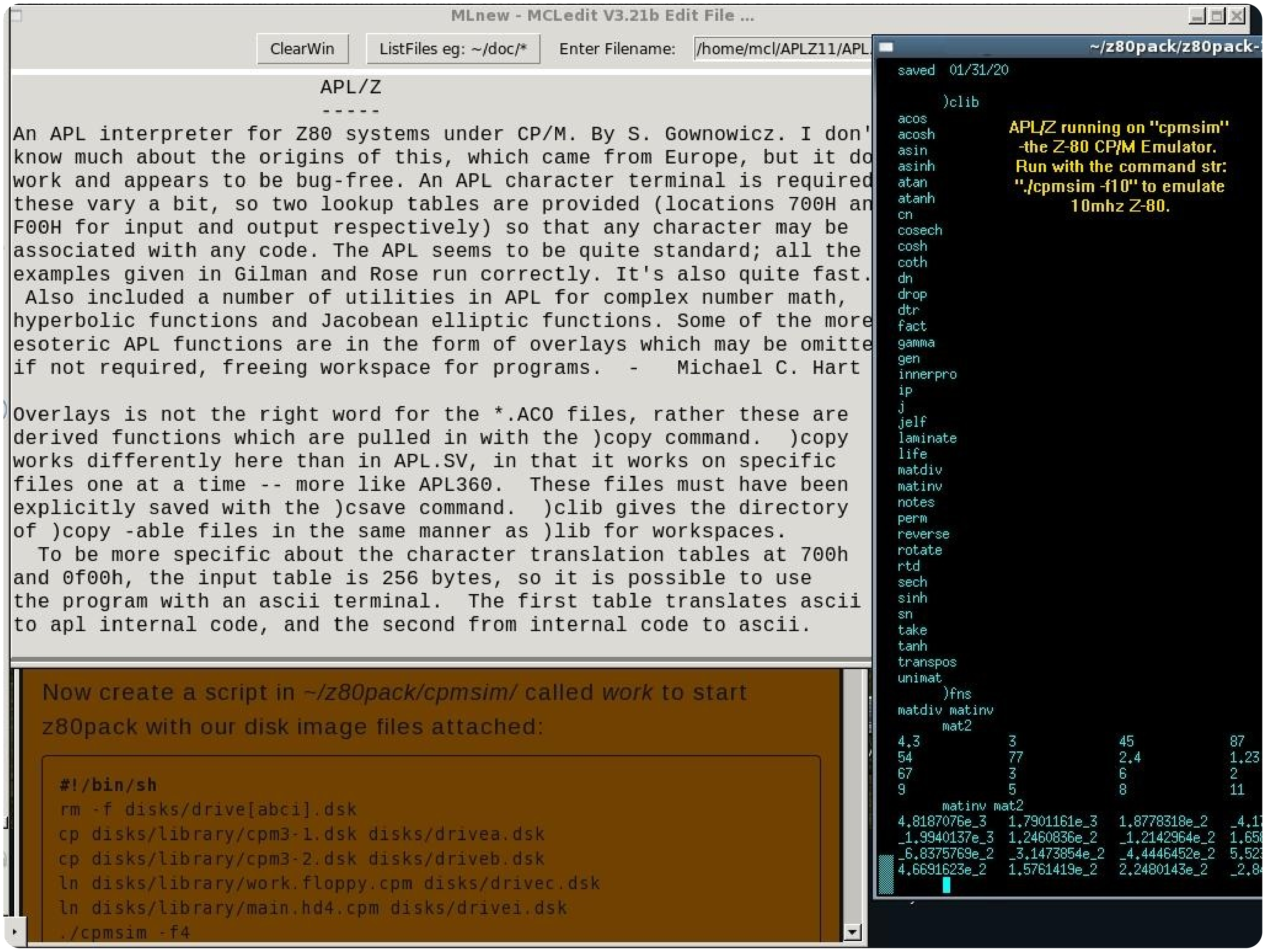
GEMESYS Canada

Research & Consulting Services in Advanced Technology & Artificial Intelligence



```
                               APL/Z
                               -----
An APL interpreter for Z80 systems under CP/M. By S. Gownowicz. I don'
know much about the origins of this, which came from Europe, but it do
work and appears to be bug-free. An APL character terminal is required
these vary a bit, so two lookup tables are provided (locations 700H an
F00H for input and output respectively) so that any character may be
associated with any code. The APL seems to be quite standard; all the
examples given in Gilman and Rose run correctly. It's also quite fast.
 Also included a number of utilities in APL for complex number math,
hyperbolic functions and Jacobean elliptic functions. Some of the more
esoteric APL functions are in the form of overlays which may be omitte
if not required, freeing workspace for programs.  -   Michael C. Hart

Overlays is not the right word for the *.ACO files, rather these are
derived functions which are pulled in with the )copy command.  )copy
works differently here than in APL.SV, in that it works on specific
files one at a time -- more like APL360.  These files must have been
explicitly saved with the )csave command.  )clib gives the directory
of )copy -able files in the same manner as )lib for workspaces.
  To be more specific about the character translation tables at 700h
and 0f00h, the input table is 256 bytes, so it is possible to use
the program with an ascii terminal.  The first table translates ascii
to apl internal code, and the second from internal code to ascii.
```

Now create a script in *~/z80pack/cpmsim/* called *work* to start z80pack with our disk image files attached:

```
#!/bin/sh
rm -f disks/drive[abci].dsk
cp disks/library/cpm3-1.dsk disks/drivea.dsk
cp disks/library/cpm3-2.dsk disks/driveb.dsk
ln disks/library/work.floppy.cpm disks/drivec.dsk
ln disks/library/main.hd4.cpm disks/drivei.dsk
./cpmsim -f4
```

```
saved  01/31/20

      )clib
acos
acosh
asin
asinh
atan
atanh
cn
cosech
cosh
coth
dn
drop
dtr
fact
gamma
gen
innerpro
ip
j
jelf
laminate
life
matdiv
matinv
notes
perm
reverse
rotate
rtd
sech
sinh
sn
take
tanh
transpos
unimat
      )fns
matdiv matinv
      mat2
4.3            3              45             87
54             77             2.4            1.23
67             3              6              2
9              5              8              11
      matinv mat2
4.8187076e_3   1.7901161e_3   1.8778318e_2   _4.1
_1.9940137e_3  1.2460836e_2   _1.2142964e_2  1.65
_6.8375769e_2  _3.1473854e_2  _4.4446452e_2  5.52
4.6691623e_2   1.5761419e_2   2.2480143e_2   _2.8
```

APL/Z running on "cpmsim" -the Z-80 CP/M Emulator. Run with the command str: "./cpmsim -f10" to emulate 10mhz Z-80.

APL on a CP/M Z-80: This APL/Z program is running on the Z80Pack/Z80Pack-1.36/cpmsim emulator. This actually is pretty exciting for us. I found something called "APL/Z-11" - a zip-file collection of what amounts to a complete APL interpreter for the Z-80. Except it runs under full CP/M, of course, and so far, I only have the binary image of the interpreter. (The "APL.COM" CP/M file).

To test it, I had to build a complete CP/M machine - complete with virtual disks, and a running Z-80 box emulator and all of CP/M - on one of my Linux machines. This was an experimental hack, and I honestly did not expect it to all work. One has to download and build from the source, the Z80Pack utilities, and then also get the CPMTools utilities, that let you copy your files from the Linux directory, to the virtual disk-drive that your CP/M emulator/simulator looks at.

Then, you can load and run the APL, and need to figure out the translate-tables the original author had used. But to my astonishment, it all worked - rather well, actually. The instructions were on a site called "Techtinkering.com", and I followed these to the letter, to get all the CP/M emulation code working, and then used the CPMtools program ("cpmcp") to copy all the APL stuff onto a virtual CP/M disk, called "I:".

You can list the files on the CP/M disk, with "cpmls" - from your Linux command-line session. Once you start the CP/M simulator - with the disks defined so they can be seen - you can see all the APL files, and you can start APL and load "matinv" and "matdiv", and invert a matrix. It's like black-magic, really. You can even save and load your workspace.

The cyan/black-background right-side panel shows the Z-80 CP/M APL session, with results of inversion of a simple 4x4 matrix.

*The various math functions, shown with ")clib" command, are APL/Z *.ACO files. Each of these files is one function, which can be loaded into the small CP/M Z-80 workspace (which is about 30K), and can be used as needed, and then erased to save room and to prevent the symbol-table (number of variables) limit from being reached.*

*The *.ACO files of individual functions are full math programs in their own right. And they are not overlays - you ")COPY and ")ERASE" them explicitly, as needed.*

But this is a full, professional-quality APL, and seem to be free of bugs. You can set the index-origin using the old APL-360 command ")ORIGIN n", where n=0 or n=1. Note that to use the "matinv" and "matdiv" functions, you need index origin set to 1.

The command "y matdiv x J.^ 0 1 2" works, where y and x are vectors defining a scatter-plot on the x-y grid, which you want to pass a second--order least-squares curve through. J is the APL jot character, and "^" is the exponentiation character, normally a "". The "*" char has been remapped for "multiply" and the "%" has been remapped for "divide". The high-minus is remapped as an underscore. The APL arrow (assignment) operator is remapped as the backwards quote "`". But it all works.*

Eg:
```
y ` 100 300 320 600 1300 1400 2000 2200 2300 2300
x ` 80 100 140 150 200 240 300 360 390 420

y matdiv x J.^ 0 1 2
```

(The above program returns the co-efficients:...)

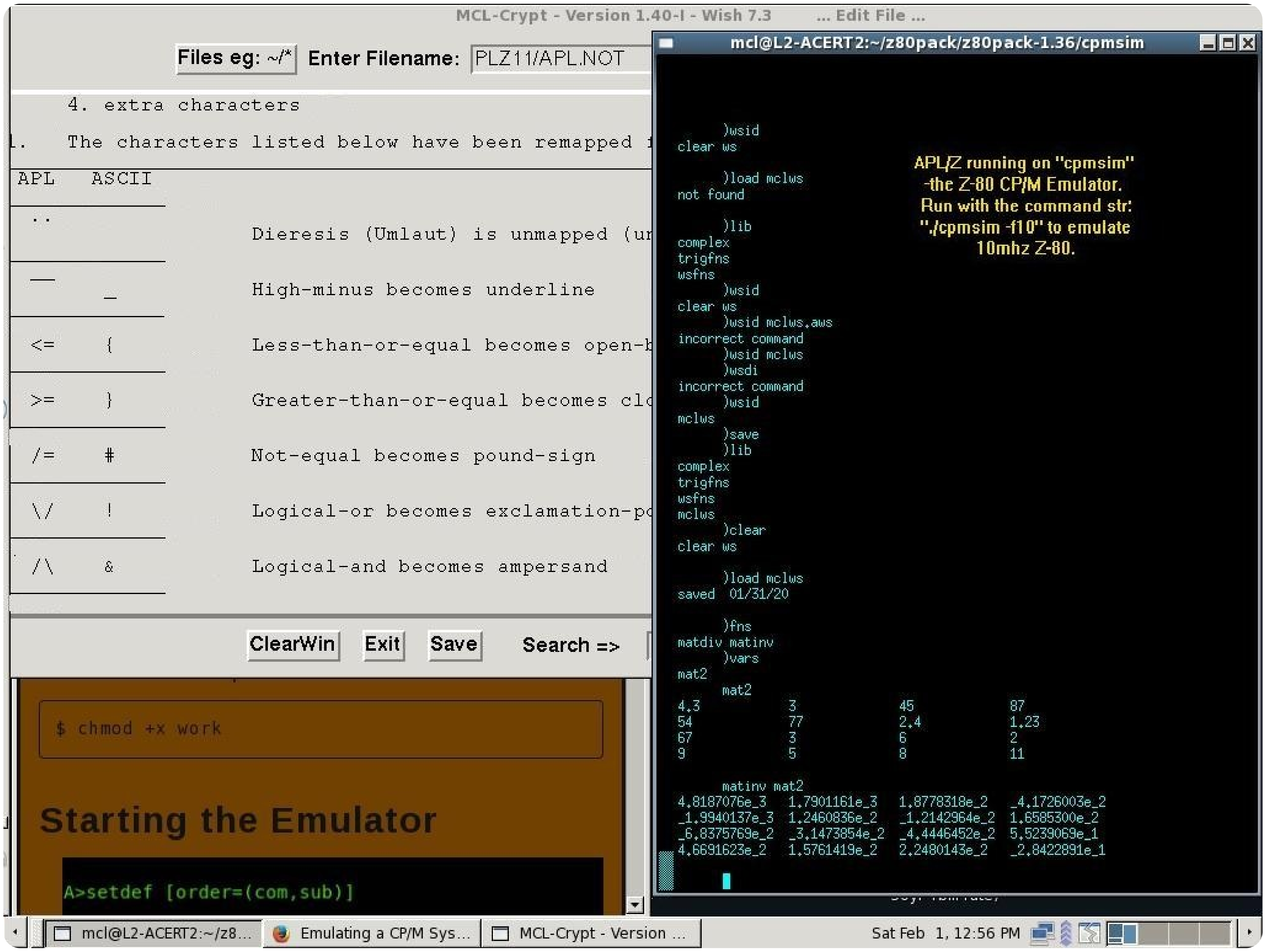
```
-9.9687297e2 1.3215355e1 -1.2303062e2
```

You can load the app "sAPL" onto your Android device, and create vectors x and y, and use function "domino2" (the dyadic - ie left and right parameter function) for matrix divide, and get the same answer. Try it. Note, you will have to create the "domino2" function by editing the "domino" function, since it is monadic. Send me a note, if you want detailed instructions and are learning APL. It's easy. Only reason I have function "domino" in the sAPL workspace, is because it is difficult (or impossible!) to generate the quad+divide-symbol in Android. I will update the sAPL "continue" workspace, to have this example, with x, y and domino2, in next release of sAPL.
(Note, you can run sAPL in side a Windows CMD session, or in any DOSbox on Linux, MacOSx or Windows)

On sAPL, the program:

y domino2 x <jot>. 0 1 2 (the <jot> is made by pressing <ALT>J on the Android device, using "Hackers Keyboard".)*

sAPL program returns: -996.8729766 13.2153557 -0.01230306278



MCL-Crypt - Version 1.40-I - Wish 7.3 ... Edit File ...
Files eg: ~/*
Enter Filename: PLZ11/APL.NOT
ClearWin
Exit
Save
Search =>
mcl@L2-ACERT2:~/z80pack/z80pack-1.36/cpmsim
mcl@L2-ACERT2:~/z8...
Emulating a CP/M Sys...
MCL-Crypt - Version ...
Sat Feb 1, 12:56 PM

```
    4. extra characters
1.   The characters listed below have been remapped f
```

APL	ASCII	
..		Dieresis (Umlaut) is unmapped (un
‾‾	_	High-minus becomes underline
<=	{	Less-than-or-equal becomes open-b
>=	}	Greater-than-or-equal becomes clo
/=	#	Not-equal becomes pound-sign
\/	!	Logical-or becomes exclamation-po
/\	&	Logical-and becomes ampersand

```
$ chmod +x work
```

Starting the Emulator

```
A>setdef [order=(com,sub)]
```

```
      )wsid
clear ws

      )load mclws
not found

      )lib
complex
trigfns
wsfns
      )wsid
clear ws
      )wsid mclws.aws
incorrect command
      )wsid mclws
      )wsdi
incorrect command
      )wsid
mclws
      )save
      )lib
complex
trigfns
wsfns
mclws
      )clear
clear ws

      )load mclws
saved  01/31/20

      )fns
matdiv matinv
      )vars
mat2
      mat2
4.3            3              45             87
54             77             2.4            1.23
67             3              6              2
9              5              8              11

      matinv mat2
4.8187076e_3   1.7901161e_3   1.8778318e_2   _4.1726003e_2
_1.9940137e_3  1.2460836e_2   _1.2142964e_2  1.6585300e_2
_6.8375769e_2  _3.1473854e_2  _4.4446452e_2  5.5239069e_1
4.6691623e_2   1.5761419e_2   2.2480143e_2   _2.8422891e_1
```

APL/Z running on "cpmsim"
-the Z-80 CP/M Emulator.
Run with the command str:
"./cpmsim -f10" to emulate
10mhz Z-80.

This is intial page, showing some APL-ASCII character translation information, the example of the cpmsim emulator starting, and a full APL/Z session running, with simple 4x4 matrix "mat2" being inverted, using the APL/Z "MATINV" function, which would correspond to the traditonal "Domino" character for matrix-inverse, in APL.