

EMULITH

A Diser / ETH Lilith emulator.

1) What is Emulith ?

Emulith is a register-level simulation, in C, of the ETH Lilith Modula-2 computer.

The ETH Lilith is a 16 bit workstation developed at the ETH by a team under lead of Prof. N. Wirth between ca. 1979 and 1982. Main reasons to fame were the fact that the hardware was developed to fit the Modula-2 language requirements and its high resolution display.

It is also one of the earliest personal computers, ie a computer dedicated to a single user. It is also one of the earliest machines to have a mouse as a standard input device.

The machines were later commercialized, with moderate success, under the Diser brand.

The machine in this picture is my Diser Lilith s.n. 148, which I restored and used as the basis for Emulith.



The Lilith has a 2901 bit-slice based architecture, uses a 128Kx16 main memory, a 704x928 monochrome high resolution display and a 10MB cartridge disk. The main memory can also be read in 64 bit chunks, as is done by the high resolution display and the Mcode instruction/data fetch unit.

A 40bit wide microcode implements, on the real machine, the Mcode interpreter that forms the virtual, stack-based, Modula-2 computer. Additional sections of microcode are used to read a boot file, boot the machine and start the operating system.

This operating system, Medos, is written entirely in Modula-2.

My Lilith emulator Emulith is a functional equivalent of the actual Lilith hardware, uses unchanged microcode and disk images of the real machine and gives a reasonable good idea of what the real Lilith looks like.

High resolution display, cartridge disk, mouse and keyboard are all part of the emulation. Not covered (yet ?) are Ethernet and printer interface, RS232 is partially covered.

Within these limitations the emulator runs all programs I have thrown at it.

2) Getting started

2.1 requirements

The following is needed to run Emulith :

- A reasonable fast PC (> 1 GHz) , running Windows, Linux or OS-X.
- Minimal screen resolution of 1024x768, but 1280x1024 is greatly preferred.
- A 3-button mouse is helpful.
- Some willingness to read documentation : the Lilith is a 30year old system and its usage is not comparable with current OS'ses. A Lilith manual is part of the documentation.

The performance of Emulith on a 1 GHz machine is comparable with the real hardware, which used a 7Mhz main clock. Any operation involving the disk is considerably faster on the emulator. I choose not to optimize things further, but to keep a rather simple and readable C implementation of the Lilith CPU.

2.2 download & install

- Go to my FTP site <ftp://jcdreesen.dyndns.org/emulith>

- Download the windows install package : **emulith_v13.msi**

Just run the normal Windows install . You will find a new program, "emulith", and one pointer each to the Lilith manual and the Emulith manual.

- or download the Linux package : **emulith_v13.tgz**

Just unpack the .tgz file and you should be ready to go. A makefile is provided if you prefer or need to recompile the package. Manuals are in the 'docu' subdirectory.

- or the OS-X package : download **emulith_v13.dmg**

Unpack the dmg and drag the Emulith application to /Applications. Documentation and diskimages are in the Resources directory of the application. The path is /Applications/Emulith.app/Contents/Resources

Important : since the disk images files are located inside the application resource fork, you cannot write to them. So for OS-X it will normally be necessary to copy the disk images to some other directory and use the System->ImageDir command to point to this new location. The same is valid for the virtual floppies.

The OS-X package should be universal (IE PPC and i386, but only the inlet variant has been tested, due to lack of appropriate hardware.)

- Two zip files containing more disk images are available.

These need to be unpacked and copied into Emulith's 'img' subdirectory. Images are independent of host system OS.

2.3 the package

After program installation you will find a directory with this contents :

Files

- emulith(.exe) : emulator executable
- emulith.ini : contains startup variables.
- ascii.def : defines which Medos files are considered to be text files.
- Makefile : makefile to assist in recompiling the project.

Subdirectories

- Src : emulator source code directory. Mainly C and a tiny bit of C++.
- include : .h include files for the source code
- img : disk images for the emulator. See Chapter 5 for a description.
Any additional images files should also be moved in this place.
- docu : Emulith and scan of the original Lilith manual.
- mcode : binary dumps of the firmware prompts, mcode source & assembler.
- floppy : virtual floppy disk.
- support : some supporting C-code.
- fltk : placeholder for the FLTK toolkit.
- vid : bitmaps to be used in creating movies.

A more thorough description of the package, including tips on how to recompile, is given in chapter 6.

The OS-X version has its files distributed between the MacOS and Resources directories inside the .app.

3) Emulator usage

3.1) booting Emulith

3.1.1) normal boot

The emulator is started with the following command : `./emulith` , or , for the GUI lovers, by clicking on the filename or symbol.

The program comes up with a big window on the left side and some emulator GUI elements on the right hand side. The left hand window is the actual Lilith screen.

Emulation is already running, but the Lilith has not booted yet.

To boot, press the spacebar. The emulated Lilith will load the PC.Bootfile from the disk image and start Lilith's operating system Medos.

Alternatively, if a 'startup.cmd' is present on the working directory, it will be used to provide keyboard input to the emulator.

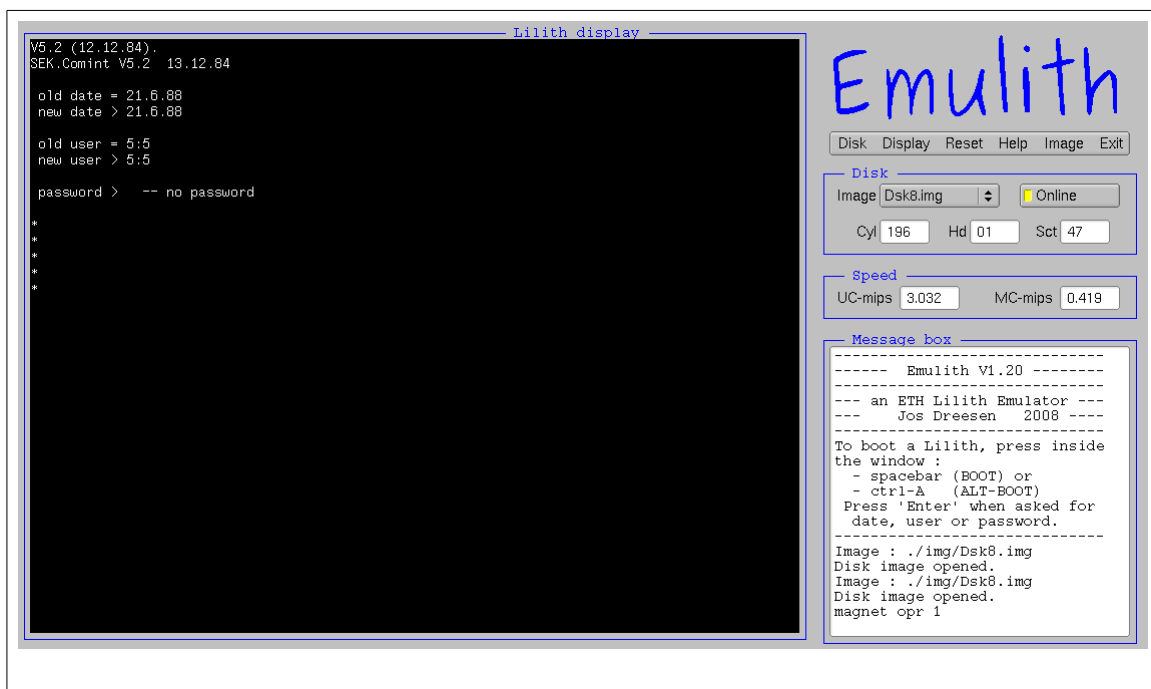
Medos comes up with a version prompt : `V5.2`

It now checks the disk, which will take a couple of seconds, and print a "." when ready.

Medos now prompts for "Date", "User" and "Password". All of these can be answered with the <Enter> key.

Medos then prints its prompt, a '*', and is ready for accepting commands.

The screen now looks as follows :



3.1.2) Alternative boot sequence .

If ctrl-A is pressed instead of the space bar the alternative boot sequence is started. Instead of the normal PC.Bootfile the alternative PC.Bootfile.Back will be loaded.

This makes it possible to try out alternative Medos versions, since one known working bootfile can always be kept accessible this way.

3.1.3) When booting fails...

First thing to check is whether the disk image is online. The emulated Lilith cannot boot without the disk image being online. Also a little patience is needed : checking the file system takes several seconds, just like on the real hardware.

Second thing to try is the alternative bootsequence : try System->Reset and type Ctrl-A.

If this also fails check with another disk image. You can always copy the PC.BootFile from a working diskimage to the non-working disk image with the “memfilexfer” program.

Note that not all disk images I provide have valid bootfiles : they are straight copies of the cartridges I have and therefore some might have erroneous bootfiles.

Also check if the working directory is OK : you will need the “img” directory with diskimage files, each exactly 9633792 bytes long, and the “mcode” directory with 5 2Kx8 prom images (Lilith microcode) and 3 256x4 prom images (Mcode jump table)

3.2) The Emulith GUI

Emulith has several GUI elements : the main menu, a disk image select box, a disk online switch and various indicators.

The main menu contains these elements:

Disk->WrProt	write-protects the virtual HB120 disk.
Disk->WrBack	Write back diskimage at program end or diskchange.
Disk->ShowCHS	show the virtual disk cylinder/head/sector information.
Display>Landscape	Run the emulation in landscape format. (first models)
Display->Portrait	Run the emulation in portrait format. (normal opr. mode)
Display->WhiteOnBlack	Use white text on black background. (Normal mode)
Display->BlackOnWhite	Use black text on a white background. (Reversed mode)
Display->ReduceGUI	Use a minimal GUI set. All elements are in the top menu bar.
Display->Borders	Draw ornamental boxes around Emulith's GUI elements
Display->MakeMovie	Save individual bitmaps to generate movie. See chapter 7.
System->Reset	Reset the Lilith. The user must press a key to reboot.
System->CmdFile	Stream keyboard input from file.
System->ImageDir	Select location of disk image files.
System->FloppyDir	Select location of virtual floppy drive.
System->ShowMsg	Show message box (for use in reduced GUI mode).
System->ClearMsg	Clear the message box contents.
System->HideMsg	Remove the message box.
Help	Display a help text. Select again to remove help text.
Image->Split_Img	Split a disk image into separate files. See chapter 3.4.3
Image->Make_Img	Combines files into a disk image. See chapter 3.4.4
Image->Fix EOLN	Change EOLN characters when handling images.
Image->CloseLog	Close log file generated with 'split' and 'make' command
Exit	Close the disk image and exits Emulith.

Further controls are the image select box : this allows the user to select between the different disk images in the 'img' directory. Take note that whenever a new disk is selected it will be offline.

The “diskonline” button now needs to be pressed, after which a 'return' or 'ctrl-A' will boot the Lilith.

All other GUI elements are indicators : there are the disk cylinder-head-sector information boxes which tells the currently selected disk sector. Display of this information can be controlled by 'Disk->ShowCHS' command.

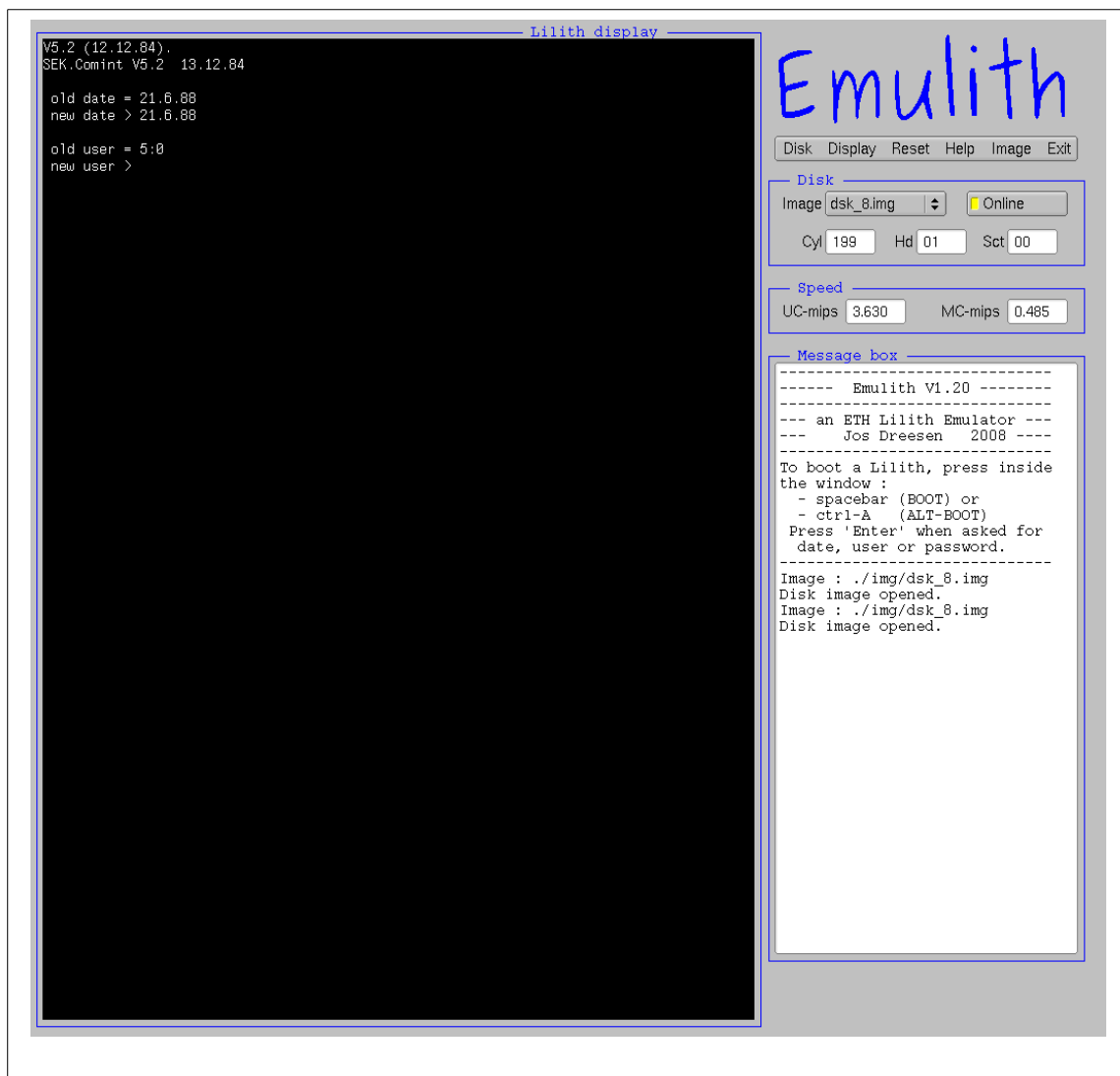
Furthermore a speed indicator is present : this tell the number of firmware operations per second, as well as the number of Mcode operations per second. Note that, as long as the Lilith has not booted, the Mcode-mips indicator is zero : this is correct since loading the bootcode is a firmware operation. Only after the boot file is loaded will the Mcode interpreter start and the Mcode mips will go up.

Last indicator is the message window, used by Emulith to relay status and error messages. This window can be cleared by the Display->ClearMsg menu command.

3.3) Emulith operation modes.

Emulith has several operational modes : first of all it is possible to switch between landscape and portrait display mode. The Lilith normally uses a portrait monitor, but many laptops will struggle with the high resolution required to properly display this. Therefore a landscape mode has been implemented, which uses the same resolution as the first Lilith prototypes. Most, but not all, Lilith/Medos software will poll which display type is in use and act accordingly.

A Lilith in portrait operating mode (with full GUI, black on white display):



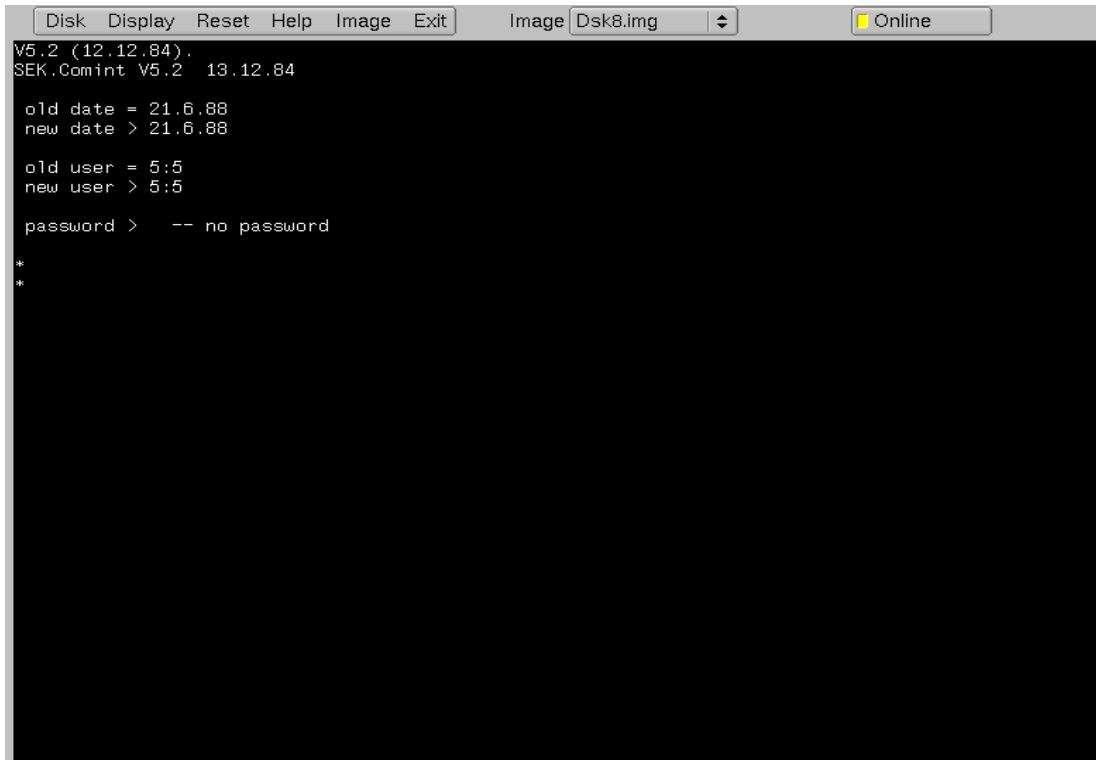
A Lilith operating in landscape mode (and full GUI, blackonwhite mode) :



Note : the above picture shows a Lilith with the virtual disk in the offline state. It is not possible to boot in this state, the user needs to set the disk online.

A second choice that Emulith gives is the choice between full GUI mode and reduced GUI mode. In reduced GUI mode just the menu, diskselect and diskonline buttons are present in the top area. This resembles closely the real operation. In reduced mode the Display->ShowMsg command can be used to display the message window.

A Lilith, freshly booted, and running in reduced GUI, landscape mode would look like :



```
Disk Display Reset Help Image Exit Image Dsk8.img Online
V5.2 (12.12.84).
SEK.Comint V5.2 13.12.84

old date = 21.6.88
new date > 21.6.88

old user = 5:5
new user > 5:5

password > -- no password

*
*
```

Changing between any of the above modes requires the virtual Lilith to reboot.

Another switch is the BlackonWhite / WhiteonBlack video mode select. Normal Lilith operation is WhiteonBlack, but the other option might be easier on the eye. No reboot is required when switching between both video modes.

In full GUI mode a control switch is given that enables or disable the drawing of ornamental boxes around the GUI elements. In reduced mode no boxes are drawn.

3.4 File transfers.

3.4.1) File transfers between disk images.

In normal operation the Medos “memfilexfer” program can be used to copy files from one disk image to another. This is a standard program and can be used just as on the real Lilith. Simply start the program, type the filename(s), set the disk off-line, swap diskimages, set back online, wait a few seconds and the file will be written on the newly selected image. Note that this program uses the display area as extra memory, so you can actually see the bits being transferred. This program requires a reboot after use, since Medos needs to reread the disk's directory structure.

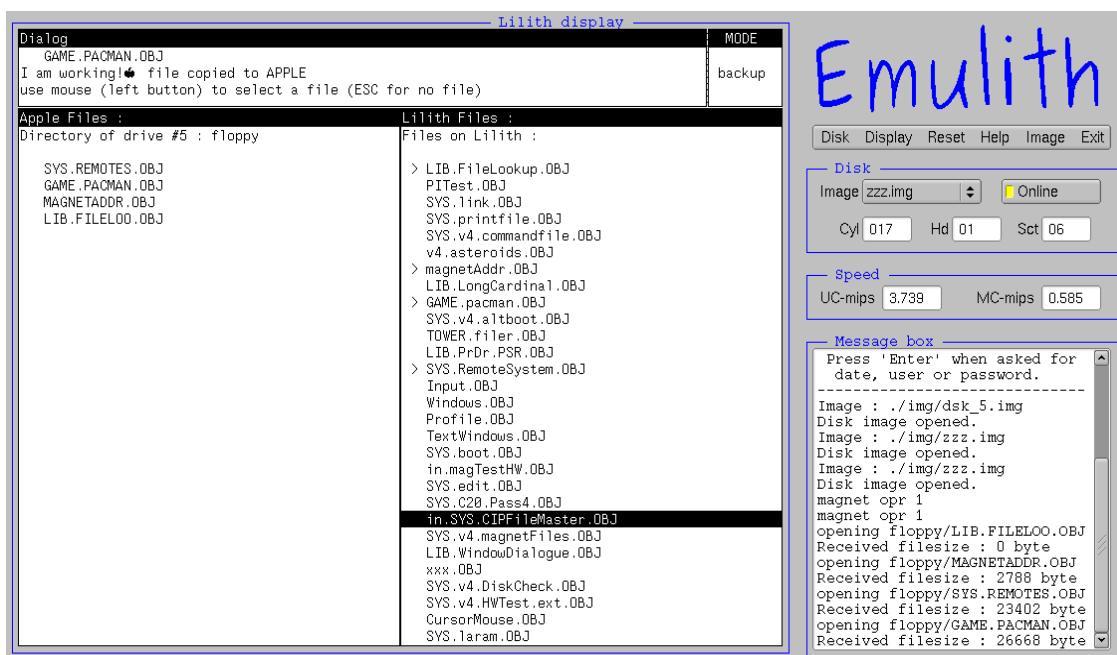
Make sure you have the WriteBack switched on if you want to retain the copied file(s) in the diskimage for subsequent use.

3.4.2) Filetransfer to hostsystem, using the virtual floppy drive.

The real hardware allows for a floppydrive, called FD6502, which uses a Apple-II compatible 5” format. This is connected to the Lilith via the 9600 bd. V24 (RS232) interface. The Lilith “applecopy” program can be used to transfer files between the Lilith and the FD6502. Beware, not all disk images have it.

Emulith has imaged the host subdirectory “floppy” to the FD6502. This means that files transferred to the floppy will be present in the “floppy” subdirectory . This directory can be changed with the System->FloppyDir switch. Note that files transferred in this way have a 4-byte header describing the file length. So although these files are now present in the host system they are nevertheless not directly usable inside this host system.

Emulation of this device is less than perfect though. Both the “filer” and “futil” programs, which are very usable filetransfer tools, do not work due to my limited knowledge of the FD6502 protocol. Volunteers to disassemble and interpret 4K of 6502 code can step forward....



3.4.3) Diskimage to files conversion.

If you would like to handle many files at once then Emulith's build-in disk image handling tool will be helpful.

Splitting a disk image is initiated by selecting the Image->Split_Img command. A popup will allow the selection (or creation) of a subdirectory. Into this directory all files in the currently selected disk image will be written. A log file of this operation will pop up in Emulith's main window, as well as in the "split_img.log" log file.

If the 'Image->FixLF' button is set, then end-of-line character replacement will be done for text files. Text files are defined as files having a filename with an extension that is listed in the 'ascii.def' file. These files will be marked with an 'A' in the logs.

EOLN replacement involves swapping the Lilith's EOLN character (0x1E) by the UNIX EOLN (0x10) character. Also the Windows version of Emulith will map to UNIX EOLN characters.

Files converted in this way will be readable by all UNIX editors and most Windows editors. (but not Notepad !)

When changing text files under Windows ensure that no LF-CR combinations are entered, as this will produce problems if these files are used again in the Lilith.

Before the split is done emulation will stop and the disk image will be put in the offline state. The user must set it back online.

A log file produced by the split function could look like this :

The screenshot displays the Emulith interface. On the left, a window titled "Lilith display" shows a file directory listing with columns for ID, TP, PR, NAME, SIZE, Created, and Modified. The listing includes various system files and directories, such as FS.FileDirectory, FS.NameDirectory, PC.BootFile, and SYS.v4.rename.OBJ.

On the right, the main Emulith window is visible. It features a title bar with the name "Emulith" in blue. Below the title bar is a menu bar with options: Disk, Display, Reset, Help, Image, and Exit. The "Disk" panel shows the current image as "dsk_5.img" and a status of "Offline". It also displays cylinder (Cyl 000), head (Hd 00), and sector (Sct 00) information. The "Speed" panel shows UC-mips at 4.583 and MC-mips at 0.000. The "Message box" contains the following text:

```

----- Emulith V1.20 -----
--- an ETH Lilith Emulator ---
--- Jos Dreesen 2008 ---
-----
To boot a Lilith, press inside
the window :
- spacebar (BOOT) or
- ctrl-A (ALT-BOOT)
Press 'Enter' when asked for
date, user or password.
-----
Image : ./img/dsk_5.img
Disk image opened.

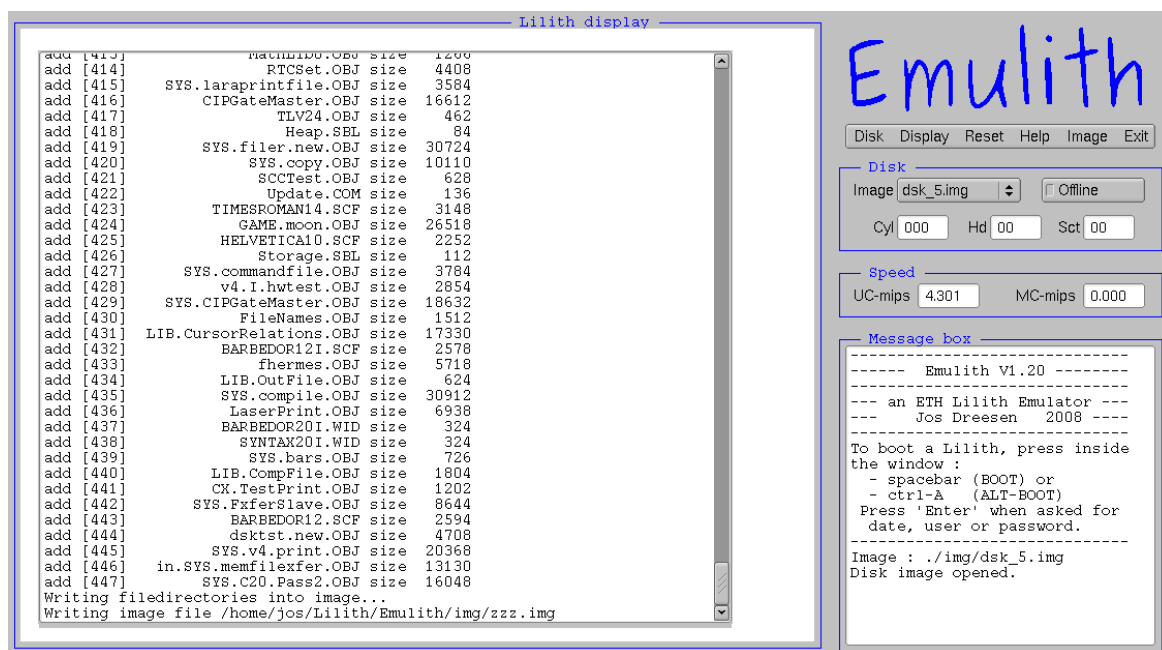
```

3.4.4) Generation of diskimages from file.

Emulith of course also allows for the reverse operation : the Image->Make_Img menu command can be used to trigger the building of a new disk image.

The user will firstly be prompted for the directory containing the files to be combined into an image. A second popup will ask for the target disk image name. Ensure that you create the disk image in the 'img' subdirectory, so that it can be selected for usage afterwards.

Once more a log file of the operation is presented. This looks as follows :



Once more EOLN conversion will be done if the appropriate menu-switch has been set.

Images created in this will function, but are of a lesser quality than the images provided with the emulator packages. Those are derived directly from actual Lilith disk cartridges, while the images created in the manner described here lack some details like protect status, correct date, file version number etc. However they will be good enough to be used in the system.

Generating a disk image will halt the simulator and the current disk image will be set into the offline state. The disk image select-box will be recreated such that any image freshly created in the "img" subdirectory can be immediately used.

Once more the operation is logged in a textfile, here it is "make_img.log".

3.5 Emulith usage notes.

3.5.1) disk images.

The real Lilith uses a Honeywell/Bull Cynthia cartridge disk drive. The emulator uses a single file, the disk image file, that contains a binary image of the cartridge data. This image includes the correct interleaving as used in the Lilith.

All disk images must reside in the `./img` directory and have an `.img` extension. Their default location is the `./img` subdirectory, but this can be controlled with the `System->` They should be exactly 9633792 bytes long, which is the sum of 48 sectors times 2 R/W-heads times 392 cylinders times 128 16-bit words per sector.

The emulator reads this images into a memory buffer. This buffer will only be rewritten to the hostsystem if the `Disk->WrBack` flag is set. If this flag is not set then all your updates will be lost when you finish ! You will not be prompted for this.

3.5.2) changing disks

To change the disk you need to stop the current disk and select the new one in the image select box. Then restart the disk(i.e. set it online) and press the space bar. Medos will reboot.

Changing disks must be done only when Medos, or one of the filer programs, is running. Changing disks while other programs are running will result in corrupted disks(-images), as is the case with the real Lilith. This is due to Medos not being aware that a disk has changed.

If you change disks while Medos is prompting you will see a small pictogram of a womans head wandering around the screen. According to prof. Wirth's docu, this is supposed to be "Lilith's ghost"....

3.5.3) Medos requirements.

Medos expects a disk to be writable, so Emulith's write protect button needs to be off.

3.5.4) Command files

There are several ways of working with commandfiles. Medos itself has the 'commandfile' programs which asks for, and executes, command files. This facility was used at the time for all bigger jobs, such as recompiling the compiler or the Medos system itself.

Emulith has 2 more possibilities : at startup the contents, if any, of the 'startup.cmd' file will used as keyboard input. Likewise the `System->Cmdfile` menu-switch can be used to redirect keyboard input to a host system file.

3.6) Other Lilith hardware

A great number of peripherals were in use at the time, some of these are listed below. Unless noted otherwise these are not implemented in Emulith.

3.6.1) MFM Disk

Later versions of the Lilith use a 15Mb MFM/ST-506 type disk as the main storage device. The controller was the WD-1001 . This combo is also emulated. But since the only microcode I have uses the 10MB cartridge disks as a main device, emulation of the MFM drive is rather moot. In effect no file system is present on this MFM disk, and it just has raw sector read/write capability.

In case I can locate later microcode sets I will update Emulith to cover later editions of the Lilith.

3.6.2) Ethernet

Lilith has a very early version of Ethernet, with a 3M bps speed and 256 addresses. This card is called "magnet", uses DMA access and was extensively used at the ETH.

3.6.3) V24 serial channel

The real Lilith has a 9600 baud serial channel. This is used for communication in general and for floppy IO in particular. Emulith only uses the V24 channel as a means to communicate to the FD6502 device.

3.6.4) FD6502 floppy

The real Lilith has a 5 ¼ floppy disk-drive. This floppy disk system is compatible with Apple-II disk drives. In fact the Lilith disk drive contains an Apple disk controller PCB...

The protocol of this device is only partially known, so emulation is also only partially.

3.6.5) MIDI interface

There has been a MIDI interface for the Lilith, however I lack any information on it.

3.6.6) SMD disk interface

A Lilith cluster would have had a central server-Lilith, which used a big-for-the-time 474MB Fujitsu M2351A SMD disk.

3.6.7) Laser printer

A Canon LPB-10 laser printer, one of the first, was used in combination with the Lilith. I lack information on the interface, but have the necessary microcode to interface to the LPB-10.

3.6.8) The DPU

The DPU, a 6802 based controller board, was part of early Liliths and was used to control the Lilith hardware. Access to the board was via a serial interface to an HP-2748 terminal. The terminal's cassette drive was instrumental in bringing up the first Liliths.

4) Lilith / Medos usage

4.1 introduction

Of course reading the actual Lilith manual is required to become proficient at using the emulator. But a few words on how to use Medos are given in this section.

Medos is, in itself, not graphical : it merely allows you to type in program names. This is done with auto-type ahead : if a partially typed name can only match one file, then the full name is given .

A particularity of Medos is that it does not have a hierarchical file system, so all files are in the one top directory. Since a cartridge can contain up to 768 files a directory list can become quite long.

In contrast to Medos many of the application programs are graphical. No fixed GUI guidelines are given, but in general the middle mouse button will give access to menus. Most often different windows on the screen have different menus associated with them. Also white borders between windows are possible candidates for menus.

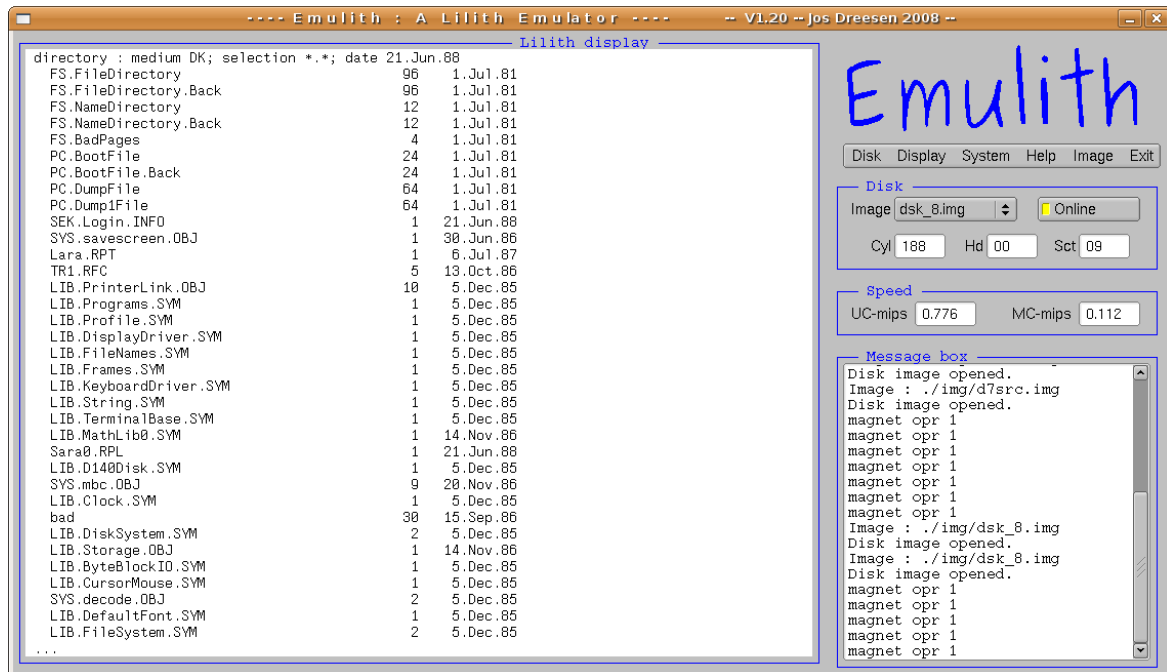
Some Medos commands to try out :

- 'directory' : gives out the directory listing.
- 'delete' : delete a file.
- 'copy' : copy a file.
- 'rename' : rename a file.
- 'list' : show the contents of a file.
- 'modula' : old 5-pass Modula-2 compiler.
- 'compile' : newer single-pass Modula-2 scompiler.
- 'inspect' or 'i' : debugger (graphical)
- 'edit' : editor (graphical)
- diskpatch : disk sector & directory editor. (graphical)
- 'commandfile' : run .COM command scripts.
- 'draw' : drawing tool
- 'Andra' : word processor.
- DEM.demo : run demo scripts. Try DEM.comdex.DEMO.
- ShowPicture : shows digitized .PICT pictures.

4.2 some programs

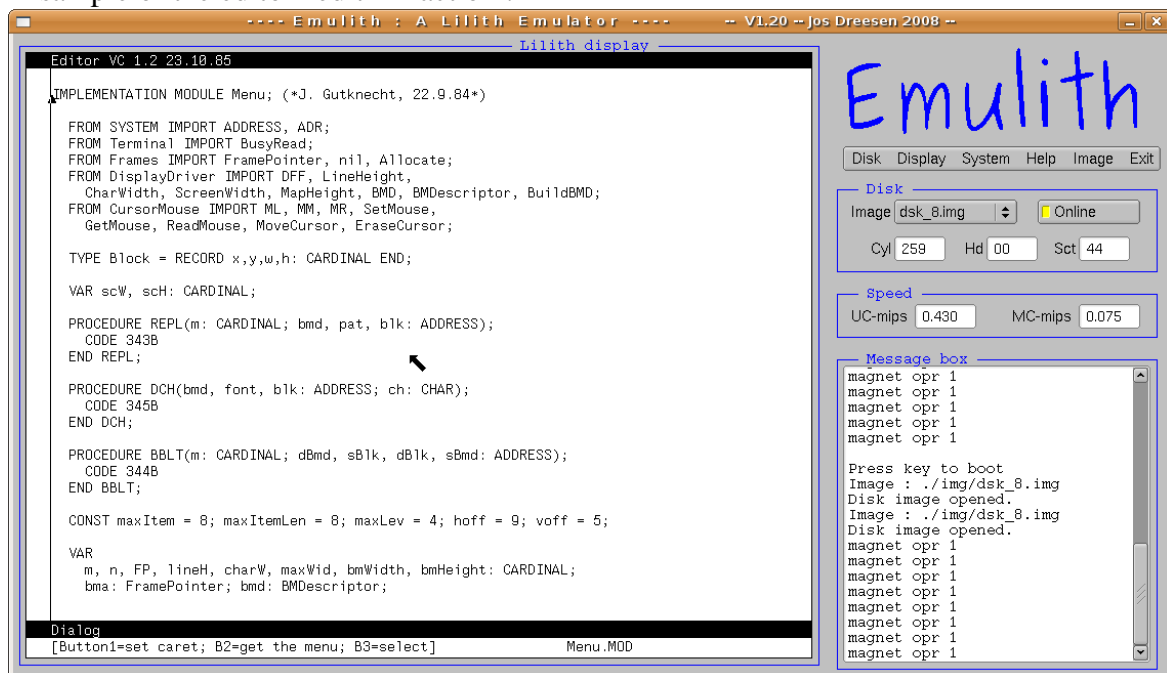
4.2.1) Directory

The picture below shows the output of a “directory” command.



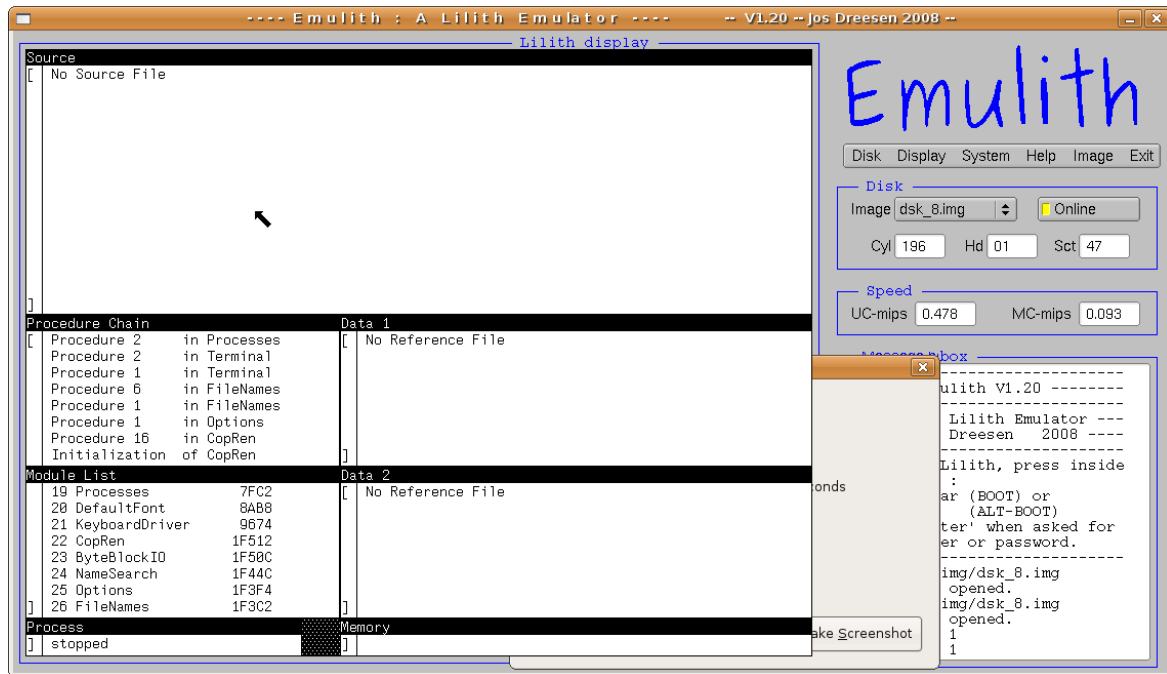
4.2.2) Edit

A sample of the editor “edit” in action :



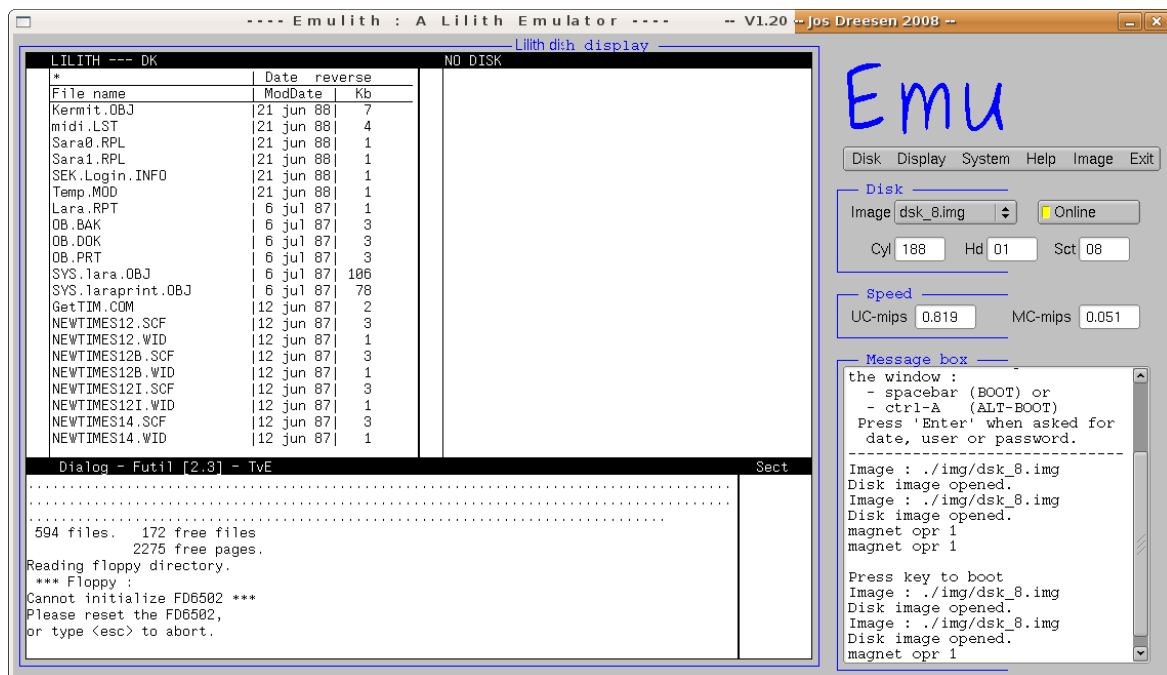
4.2.3) Inspect

“Inspect” is the Lilith debugger :



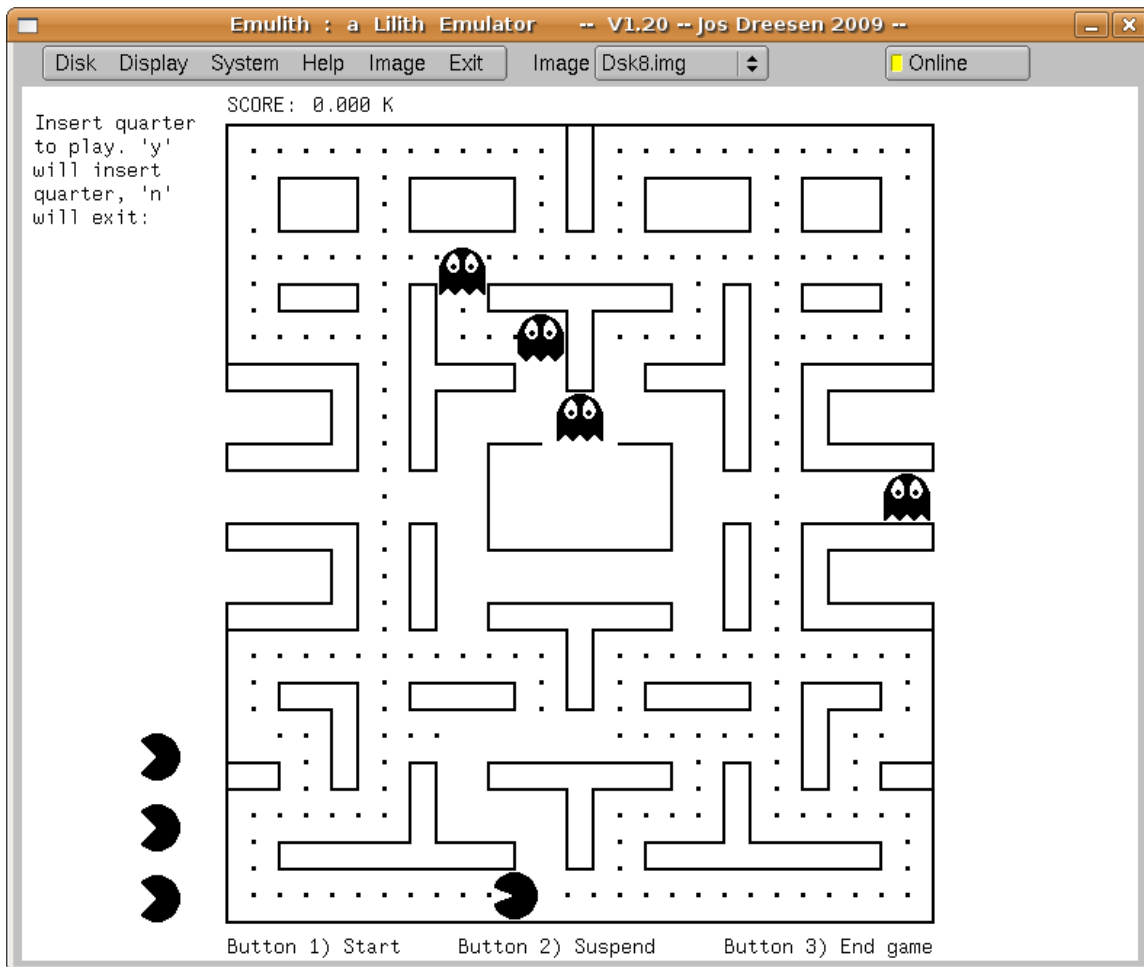
4.2.4) Futil

“futil” is a file utility for swapping files between Lilith's main disk and the floppy. (does not fully work in Emulith due to limited FD6502 emulation)



4.2.5) Pacman

Of course no eighties computer is complete without a Pacman implementation :



4.3 some more programs

A great deal of programs are present on the disk images. I will list just a few of them :

- Lara : a word processor
- Draw : drawing program
- DEM.demo : demo program, use with DEM.comdex.DEMO to obtain the demo that was shown on the Lilith's display at COMDEX.
- PUB.RandomDemo demonstration of Lilith's graphical capabilities.
- PUB.ShowPicture : shows some pictures.
- PUB.Sort : a very interesting graphical demonstration of various sorting algorithms.
- DiskPatch : low level disk editor and file repair tool.
- Xterm : terminal emulator
- GAME.Chess , GAME.Pool, and then some more...
- Kermit : the well known file transfer program.

5) Disk images description

Six disk images are provided :

5.1 Medos4 2 a/b.img

These 2 images contains a wealth of information : source code to Medos version 4.2, to the Modula compiler itself, and source code for the Lilith's bitslice firmware.

Also the firmware and the firmware compiler with its source code is provided.

The user guide is also present in Lilith-machine readable format.

A raw guideline as to what is available on this image:

- firmware source code and its assembler (MIA)
- User manual sources.
- modula mulit-pass compiler sources
- MEDOS sources
- Several demo programs.

Previous problems in disktransfer were overcome, this files are fully functional and you should be able to recompile Medos from scratch.

5.2 dsk_1.img

dsk_1.img is a bootable image of a working disk with Medos release 5.2. It has a general set of files and is typical of what would have been used during the Lilith's working life.

5.3 dsk_110.img

dsk_110.img is a bootable image with Medos 5.2. It contains both 5-pass standard M2 compiler and a functional single-pass Modula2 compiler, alas without source code.

Source code for a simple spreadsheet is present on this disk.

5.3 dsk_215.img

dsk_215.img is a bootable image with Medos 5.2. It contains a functional single-pass Modula2 compiler. It contains source code for several projects I have yet to look into.

5.3 dsk_229.img

dsk_229.img is a bootable image with Medos 5.2. It contains sourcecode to some sheet music editing software, as well as some MIDI software.

6) Program description

6.1 main source code

The program consists of the following source files :

<i>Name</i>	<i>description</i>
main.c	main simulation loop.
cpu.c	implementation of Lilith CPU.
io_proc.c	implementation of the Lilith's IO devices.
io_HB120.c	implementation of the Lilith's main disk.
io_ST419.c	implementation of the Lilith's MFM disk.
io_floppy.c	implementation of the Apple floppy interface
proms.c	reads the prom images into arrays for usage in cpu.c
img_cde.c	diskimage split and generate tool.
fltk_cde.c	FLTK based interface to host PC hardware.
lilith.h	main include file with Lilith machine description .
img_lil.h	Medos filesystem definitions.
fltk_lil.h	Some common definitions for the FLTK code.
help.h	help text definition.
bitrev.h	lookup table for byte-wide bit reversal.
mcode/150*.bin	binary images of the microcode and jump table proms. This are straight binary dumps of the devices in my Lilith.

Some notes on the source code :

From the second release I moved from Xlib to the FLTK package. This allows for easy portability between platforms. FLTK version 1.3 was used.

The relevant files are in the 'src' and 'include' directories. The 'mcode' directory contains also the source code for the bitslice firmware microcode and a simple UNIX-based assembler for it.

A second microcode version, release 14, is used to interface the emulator to a real ATA disk-drive.

6.2 Rebuilding Emulith

All versions of Emulith require version 1.3 of the FLTK toolkit, either installed locally in the fltk subdirectory or system-wide.

A common Makefile is provided will will enable compilation for all 3 hostplatforms.

6.2.1 Linux : this is the native environment where Emulith was developed. Here I used the normal set of GNU tools (gcc, g++, make) . The Makefile provided has targets for 32-bit linux systems, “make lin”, and 64-bit systems, “make lin64”.

6.2.2 Windows : the windows build has been realized within the MinGW / Msys environment. No source code changes were required. The Makefile target “make win” will build the Windows executable. The application was then packaged with the xyz tool and tested on XP and Vista.

Tool version were : MinGw 5.1.4, Msys1.0.10, gcc 3.4.5, fltk 1.3

6.2.3 Mac OS-X This was build using the GCC toolchain, not with Apple's Xcode. The makefile-target 'osx' will build the universal binary. Packing into the .app was done by hand.

6.2.4 Further targets : targets IDE and IDE_NT create Linux - versions of Emulith that use a real ATA drive hanging from the host parallel port. This is in preparation of moving my Lilith to an ATA-drive. It should also ease a potential FPGA-implementation.

6.3 supporting programs

The following files, in the support directory, are part of the supporting programs :

lifs.h	lilith file tool header file
lft_cde.c, lft_main.c	lilith file tool source codes
dmp.c	file dump tool
fix.c	various little fixups

More source code is provided in the support directory, but these are just some small programs I used in the development of Emulith and the download and treatment of diskimages. They have no current value anymore.

The lft tool in particular can be used to extract diskimages to the hostsystem, and create diskimages from directories with files on the host system.

Once a diskimage has been exported to the host, lft can do various operations on that directory.

It can also convert non-interleaved diskimages to interleaved images.

7) Generating movies

The current version has the possibility to generate movies. (Linux only, not tested on other platforms)

After enabling the Display->MakeMovie flag the system will dump up to 10000 bitmaps in the ./vid subdirectory. These will need to be postprocessed with the 'revbmp' program from the support subdirectory. This will generate valid bitmapfiles from the raw Emulith output.

Usage of the postprocessing program is ./revbmp vid/li*.bmp.

Subsequently the 'ffmpeg' utility can generate a mpg from the bitmaps.

Don't forget to remove the bitmaps before generating the next movie.

8) Limitations and errors

Several limitations exist in the current program:

- Ethernet hardware is not implemented.
- The “filer” and “futil” programs, which target the FD6502 device, do not fully work. This is due to the protocol being unknown. (in progress)
- Firmware for later Lilith versions is not available.
- Simulation speed is not that great. The OS-X version in particular suffers.
- Mouse emulation needs improvement. The mouse wraps around which sometimes gives the impression of locked mouse : just move the mouse in the opposite direction and it will reappear at the other side. This problem has proven very difficult to solve, due to Emulith not receiving mouse-events when the host mouse is moved outside the Emulith window.

9) ToDo

There is of course always something more to cater for. Examples are :

- Obtain later microcode (for MFM disk usage)
- Find sourcecode for the single-pass Modula-2 compiler.
- Look trough the 37 disks from the MFK.
- Generate software work packages.
- Recompile Modula-2 compiler from scratch.
- Recompile Medos from scratch.
- Retarget real Lilith to use an ATA-disk iso the Honeywell-Bull
- New microcode & Medos source to use an ATA disk nativly.
- Make a FPGA implementation.

10) Contact

Questions, remarks, bug reports or any other communication can be directed to `jos.dreesen` at `bluewin.ch`.

11) Thanks

Thanks are due to :

Ms. Beatrice Tobler , Museum of Kommunikation Berne : loan of all their diskcartridges, one of which was a Medos/Lilith master disk as distributed with the Lilith systems.

Prof. Juerg Gutknecht : permission to redistribute firmware & Lilith software.

Dr. Svend Knudsen : Permission to redistribute Medos source code.

Dr . Immo Noack : gift & loan of disk cartridges, loan of the ETH's Lilith hardware documentation.

I do not remember the name of the ex-ETH student from which I bought my (then non-functional) Lilith for a small amount, but thanks for not binning it.