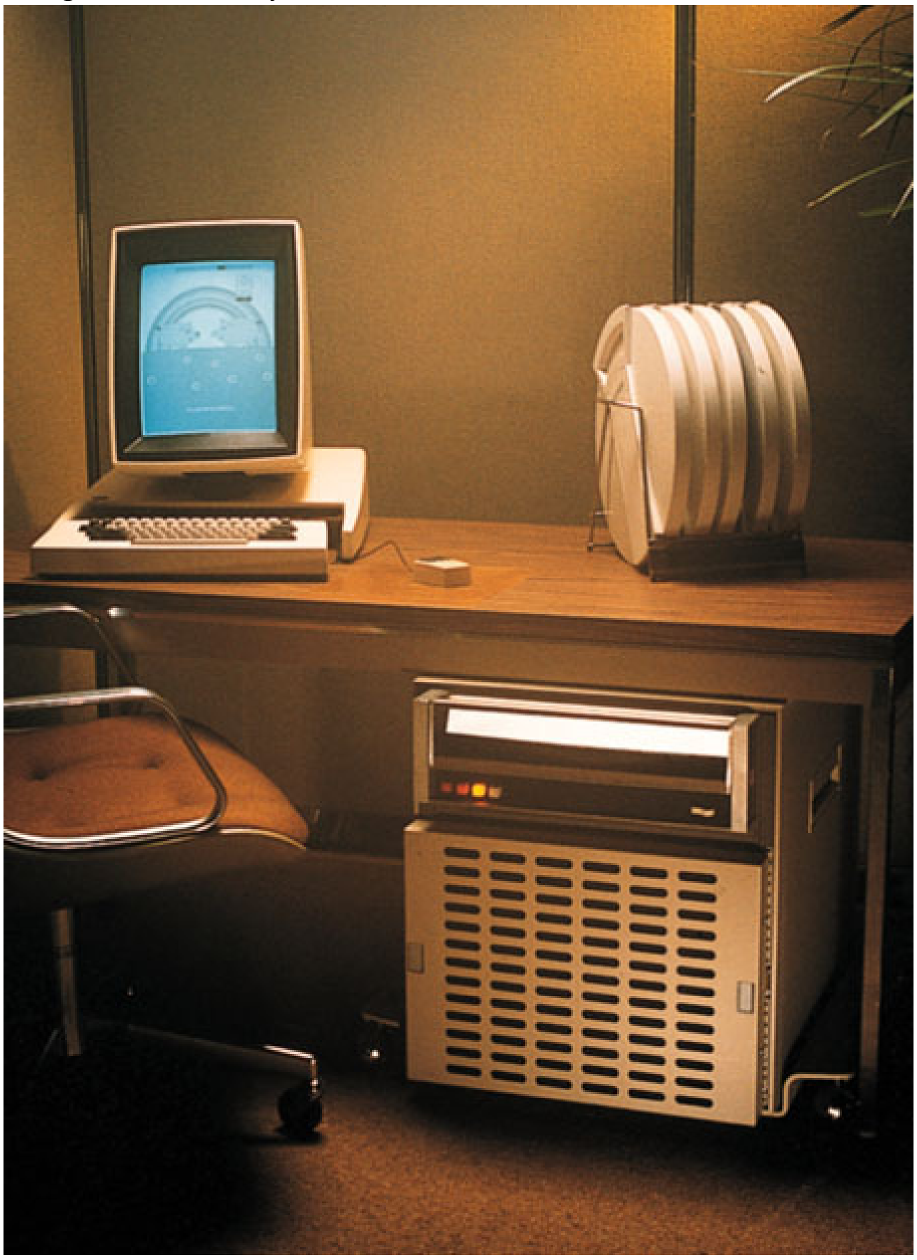
My Xerox Alto - WIP

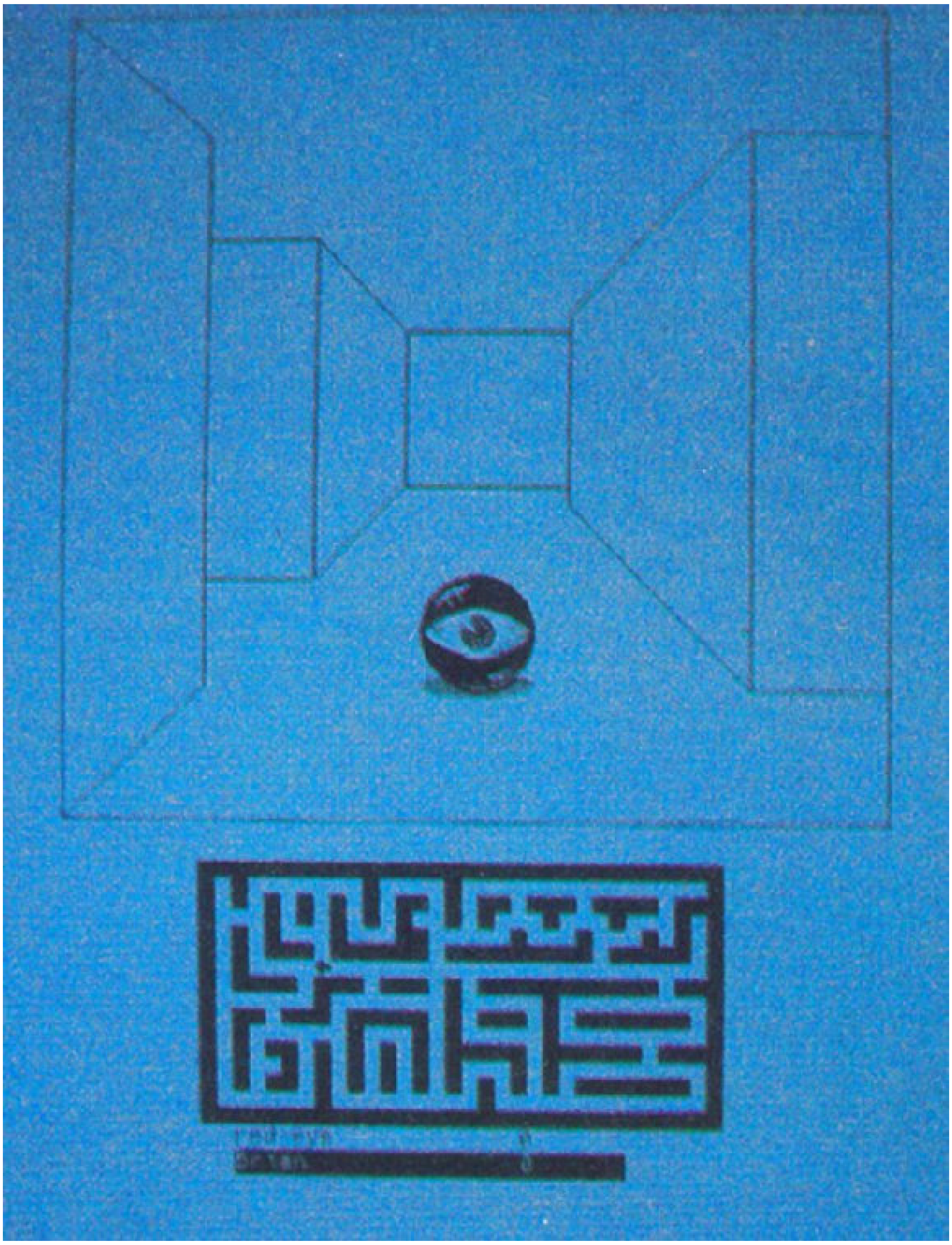
A picture of the Xerox Alto

Here's a picture of the Xerox Alto in all its glory. Look at the refrigerator-sized box under the desk, this is the rack containing all the PCBs. The Alto was built from TTL chips, and I think you could have saved the time it takes to knit a pair of legwarmers when you sat beneath it.



... and a screenshot

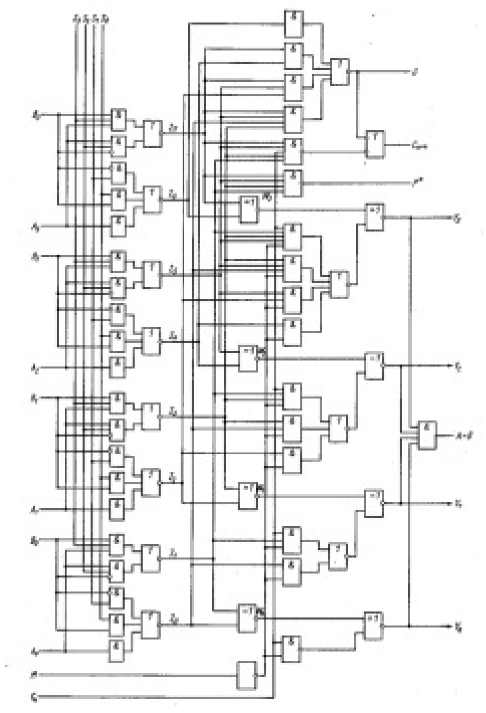
And here's a screenshot of an Alto running *maze*. If you search the net, there's a video of two people playing this game on two Altos connected over the the Ethernet. I want to be able to play this game over the internet with some other virtual Altos some day :-)



ALU schematics

June 3rd, 2007

The schematics of the inner structure of a 74181 4-bit ALU. The Alto uses 4 of this chips for its 16-bit ALU. The functions are selected through a PROM (described in alto.txt above). This is meant for those who want to understand even the level one stage below of what the Alto CPU contains to do act on binary numbers.



Plain text intro

June 4th, 2007
Some excerpts from the documentation are in alto.txt. I wrote this down after reading through some of the PDFs available at bitsavers.org, and I translated it to modern bit numbering and number base postfixes. This is fairly incomplete and may be even incorrect, so please take it with a grain of salt. It may just help to get a first overview of the machine code (the interpreted code of the Alto's own emulator), and the Alto microcode (the *real* machine code).

Disassembler

June 5th, 2007
Here's a microcode disassembler written in C. You also need to unpack the microcode and constant PROMs in the directory where you want to run the disassembler. The syntax is not too compatible with the actual Alto microassembler, it just helps you to get a view into the code. You can just as well take a look at the most recent output here.

Assembler (for the emulator mnemonics)

June 6th, 2007
I'm also writing an assembler to translate the Xerox Alto emulator mnemonics, i.e. the (user) level above the microcode, into emulator code words (16 bit). There are some files: the parser and the lexer, a list of emulator opcodes and of memory locations, and you may need the Makefile for the lex/yacc (or flex/bison) rules to be able to build the *altoasm* binary from it. There's a newer version included in *salto* (see below).

Here's some test input for the assembler, and you can also peek at the listing output generated by the latest *altoasm*. The code won't do much, it's really just meant to test the lexer and parser.

SALTO - my own simulator based on Eric's work and the docs

June 8th, 2007
I have *my own* code to simulate the Alto in the CVS. You can browse it via CVSWeb, or take a look at the Doxygen output of this project. And here's how to checkout your own copy:
My CVS server is a :pserver: at stop1984.com:/home/cvs. To get salto you will have to anonymously *login* to the CVS server once with:

```
cvs -d:pserver:anonymous@stop1984.com:/home/cvs login
```

The password is empty, so just hit return. Now you can checkout the alto stuff:

```
cvs -z3 -d:pserver:anonymous@stop1984.com:/home/cvs co
salto
```

Future updates are even easier. Go to your *~/salto* (or whereever) subdirectory and run:

```
cvs -z3 up -dAP
```

This screenshot is produced by *booting* into my helloworld.asm code.

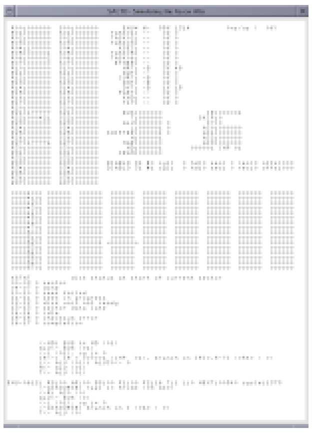
June 7th, 2007
After Al got the PROMs dumped for us, and after sorting out a lot of bugs, the video timing is now seemingly fine. It is now all based on the PROMs *a38* (FIFO control: MBEMPTY and STOPWAKE signals), *a63* (a horizontal phase state machine, generating HBLANK, HSYNC, SCANEND and HLCGATE) and *a66* (VBLANK and VSYNC for even and odd fields) of the display schematics.

The text is in memory as a bitmap once, and the DCBs are repeatedly addressing the same bitmap in with varying attributes (inverse, half pixel clock, indentation). The text is generated by the Alto's *CONVERT* instruction. The bigger font is a well-known public domain font from the X.Org archives (7x13.bdf), the smaller is 5x7 from the same source. I converted both to the Alto **.AL* font format using the tool convbdf, which I adapted to the *.AL format. This tool does not yet handle characters wider than 16 pixels.

```
Hello world! This is my first Xerox Alto program.
Hello world! This is my first Xerox Alto program.
Hello world! This is my first Xerox Alto program.
Hello world! This is my first Xerox Alto
[illegible]read the fine print when taking up a job!]
[illegible]
Hello world! This is my first Xerox Alto program.
```

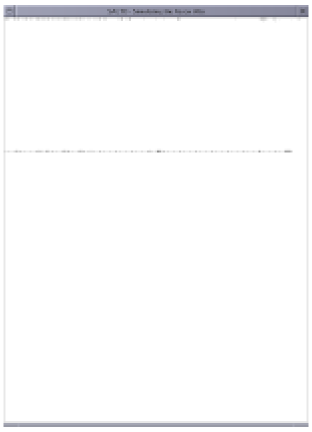
This is the *debugger* in action.

June 16th, 2007
Actually it's just displaying some internal states and a memory dump. The debug screen is included also in the non-debug build, where it is called at a rather low rate to update the register and memory status. In a debug build, it will also list all selected task's disassembled microcode, scrolling in the lower quarter of the display, and you can single-step through the code (see README for details).

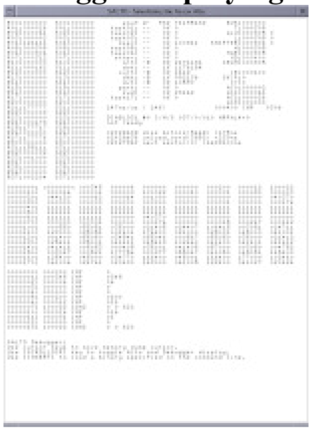


Booting from a disk image... trying.

June 20th, 2007
After quite some work I now got the emulation of the disk controller and Winchester drive far enough so that it loads some data. It's stuck with an error on every disk image I have, though. It either tries and tries to repeat the same command, or just stops trying. Must a single wrong bit somewhere ;-P The first screenshot is the display, the second the debug output of showing the boot sector in memory.



Debugger displaying the status and contents of page 0.



After fixing a lot of bugs...

June 28th, 2007

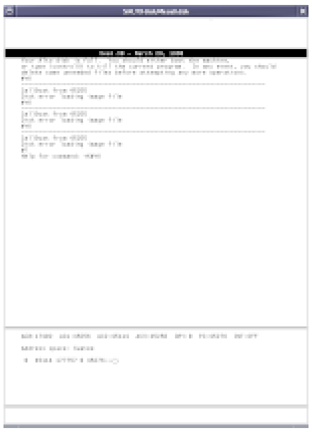
And finally we have SWAT... even after implementing the code that allows writing to the (in core) disk image, it seems that the data written to some sectors isn't readable afterwards. The timing is uber-critical; especially the DIABLO44 timing won't work, while with the DIABLO31 timing I get to these screens most of the times.

June 28th, 2007

Writing to the disk image not yet working (sync problems).

June 28th, 2007

Now I got writing to work, just so that SWAT can tell me the disk is full? How could I've known :-/



June 30th, 2007

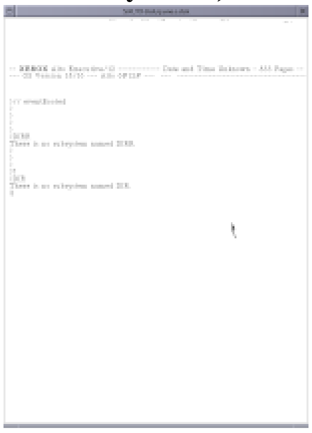
Refactored the ALU implementation with the 74181 docs in hands, and also did the F2 MAGIC and DNS functions according to the schematics - at least I think I did. The boot code is doing BITBLTs to the extended memory pages, then tries to select the other drive, and finally pops into SWAT. There's something messed up with the BITBLTs as you can see, and SWAT complains *BFSInit failed*?

The *final* bug squashed?

June 30th, 2007

Now I found one more bug: with MAGIC and DNS being in the *late* slots I

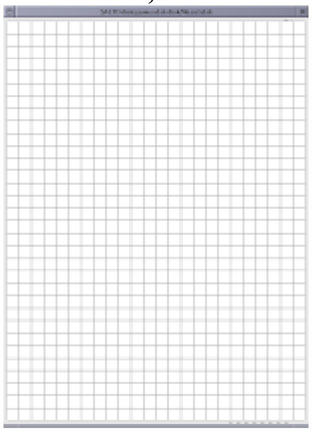
have to do late bus source after the F2 functions, or otherwise the shifter effects don't go into the BUS destinations. Now it's time to learn some basic Alto Executive commands, since DIR obviously doesn't work. Perhaps I should try LS :-)



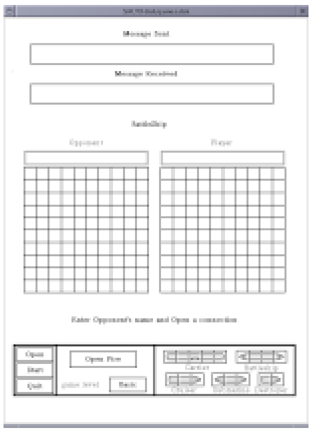
```
-- XEROX Alto Executive/[illegible] ---------- Date and Time Unknown -- [illegible] Pages --
--- OS Version [illegible] --- [illegible] ---

>[illegible]
>
>
>
>
>DIRR
There is no subsystem named DIRR.
>
>
>
>
>DIR
There is no subsystem named DIR.
>
```

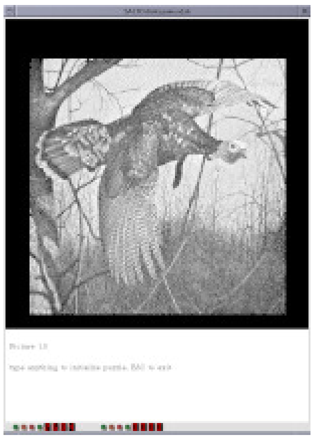
June 30th, 2007
Now I'm coming closer to the true sector layout. It was all guesswork, based on the constants from the microcode source. The first program that now loaded from disk was CRTTEST.RUN. Impressive? Well, perhaps not the screenshot, but the fact that it did indeed load and execute.



June 30th, 2007
That reminds me: Ethernet is TODO.



July 1st, 2007
Look, Ma! A turkey!1!1!!

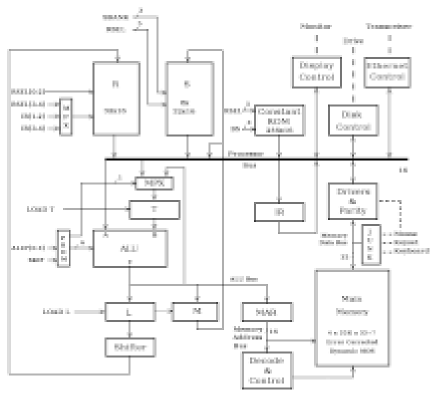


And now it looks a little puzzled.

CPU data paths

July 5th, 2007

This is an overview of the CPU data paths. The 16-bit data BUS is wired-AND, which means that all devices will drive one or more bits LOW. There are more buses, and much more registers on the various cards which aren't shown here. If you want to go into the details, visit the Bitsavers Xerox Alto archive.



Meanwhile: Alto fonts

July 7th, 2007

I spent some time on improving my Xerox Alto font decoder and viewer. It now creates PNG-files and hashes the font file (MD5) to get unique identifiers. I have a list of PNGs of all fonts I found, sorted by their MD5 hashes. Some may be broken, and some fonts may be handled wrong by my decoder - there are some fonts *hanging off* the right side of their bounding box (may be a different format, which is not left-aligned in the data words).

I am the Pinball Wizard

July 11th, 2007
At least I feel like I am. If you have a player that is capable of displaying *MNG* (read as *ming*, Multiple Image Network Graphics; see libpng.org/pub/mng for more.), you can play a *video* of me playing Pinball-easy.run (4.6MB MNG) on Salto. I implemented a subset of MNG-LC to be able to record a series of deltas between screenshots (one per frame) and store them to the MNG stream as PNGs.

In Salto you can now take single screenshots with the `PRTSCR` key and you can start/stop MNG recording with `CTRL + PRTSCR`. The screenshots get 4 digit hex serial numbers, always beginning with *alto0000.png*. The MNG file is always *alto.mng* for now. Be sure to stop recording before you break or crash salto, because closing it may not work reliably in all cases.

In the video you can see some of the remaining issues in Salto regarding the mouse cursor. This game is using the mouse cursor bitmask to draw the ball, and you will notice a jiggling of +/- 1 scanline up and down that (most probably) isn't supposed to be there.

July 12th, 2007
Here's another MNG of me booting into and playing one wave of Astro Roids (1.6MB). I played without the saucers, because the keys are weird enough for a PC keyboard *CTRL* and *A* rotate the ship, *RETURN* fires bullets, *SPACE* is hyperspace. I haven't found the thrust key yet.

And one more: Missile Command. The jumpy mouse cursor sucks, and pressing the mouse keys isn't always detected by the game. Perhaps the two issues are even related. I really have to find out why the cursor isn't stable; and I have to find a way to synchronize the Alto's cursor X and Y with the absolute mouse X and Y that SDL gives me. The problem is: Alto's mouse coordinates are always only relative.

Getting the Alto Ethernet up and running

July 15th, 2007
I just got the Alto Ethernet card simulated well enough so that the system can receive packets (duckbreath broadcasted internally) and transmit packets (e.g. after a disk boot). Now the question is how to connect some instances of Salto, or even connect Salto with a real Alto II ethernet.
BTW: If you want to test some of the things described above on Salto, here's a games.dsk.Z and a diag.dsk.Z image. You just specify the name of an image to insert into the drive on the Salto command line like ./salto games.dsk.Z.

And then *maze.run* isn't the maze game shown in the screenshot at the top of the page. Look at this MNG. It's a game to test your mouse control abilities it seems :-P

Where to go?

You of course want to go to Eric Smith's Altogether page, if you didn't just surf on in from there.

Ciao,
Juergen