Newton never dies
It only gets new hardware

Paul Guyot
Worldwide Newton Conference 2004

Introduction

Newton underground
for the MP2x00 and eMate....

Introduction (i)

- Decreasing, limited supply of hardware
- What is so great about the Newton cannot be found in other PDAs and Operating Systems :
  - Best industrial handwriting recognition
  - Data-centered
  - Newton Intelligence
- Apple will not license NewtonOS to the community

Introduction (ii)

- NewtonOS begins to be a little bit rusty for today's uses and this is going to worsen:
  - No decent Java virtual machine
  - No CSS2 web browser and no cryptographic package (SSL/SSH)
  - Poor IPv4 (current version of Internet) and no IPv6 (next generation of Internet)

Introduction (iii)

- In spite of huge efforts to keep the Newton up to date
  - WiFi driver with many supported 802.11b cards
  - Bluetooth drivers
  - Phone-like exchange capabilities (V-Cards...)

What kind of future?

“Let’s rewrite NewtonOS”

- “Don’t you think we could rewrite a new operating system with all the good things of NewtonOS?”
  - Huge task (think about the cost of the Newton)
  - No immediate reward until the project is well advanced
  - It cannot be too close to the Newton because of Apple’s patents
  - No good handwriting recognition
  - NewtonScript is a key element of NewtonOS

GNUton (NewtonScript everywhere)

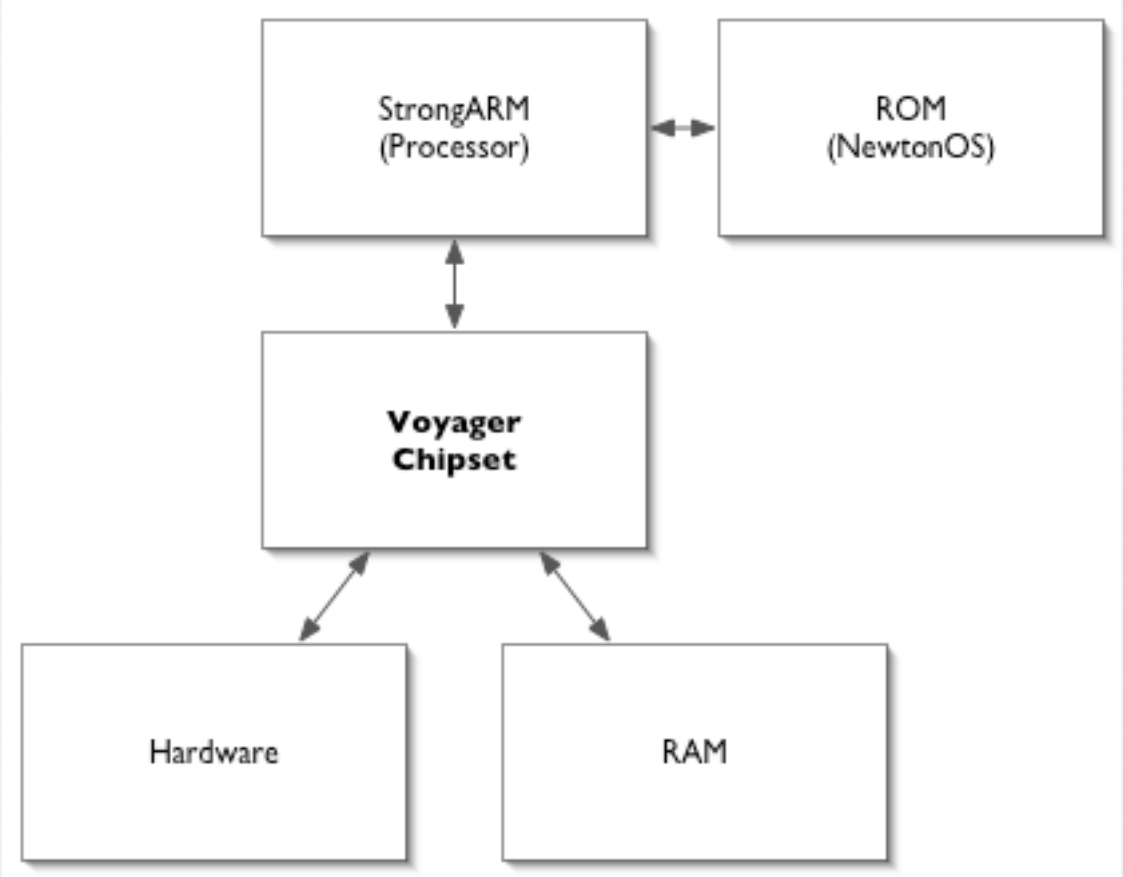
- "The Newton is NewtonScript, let's write a NewtonScript environment"
  - Since NewtonOS 2.0, less and less NewtonOS code is in NewtonScript.
  - NewtonScript only is for the interface and it only works on top of native code.
  - There is a huge amount of code to rewrite.
  - Many features such as the handwriting recognition are not written in NewtonScript.

The Einstein Project

- 1- Write a Newton Emulator with all the hardware
- 2- Port NewtonOS to ARM-powered PDAs
- 3- Extend NewtonOS on this new hardware

- Each small step is important for the platform

The architecture of the Newton (2.1 units)



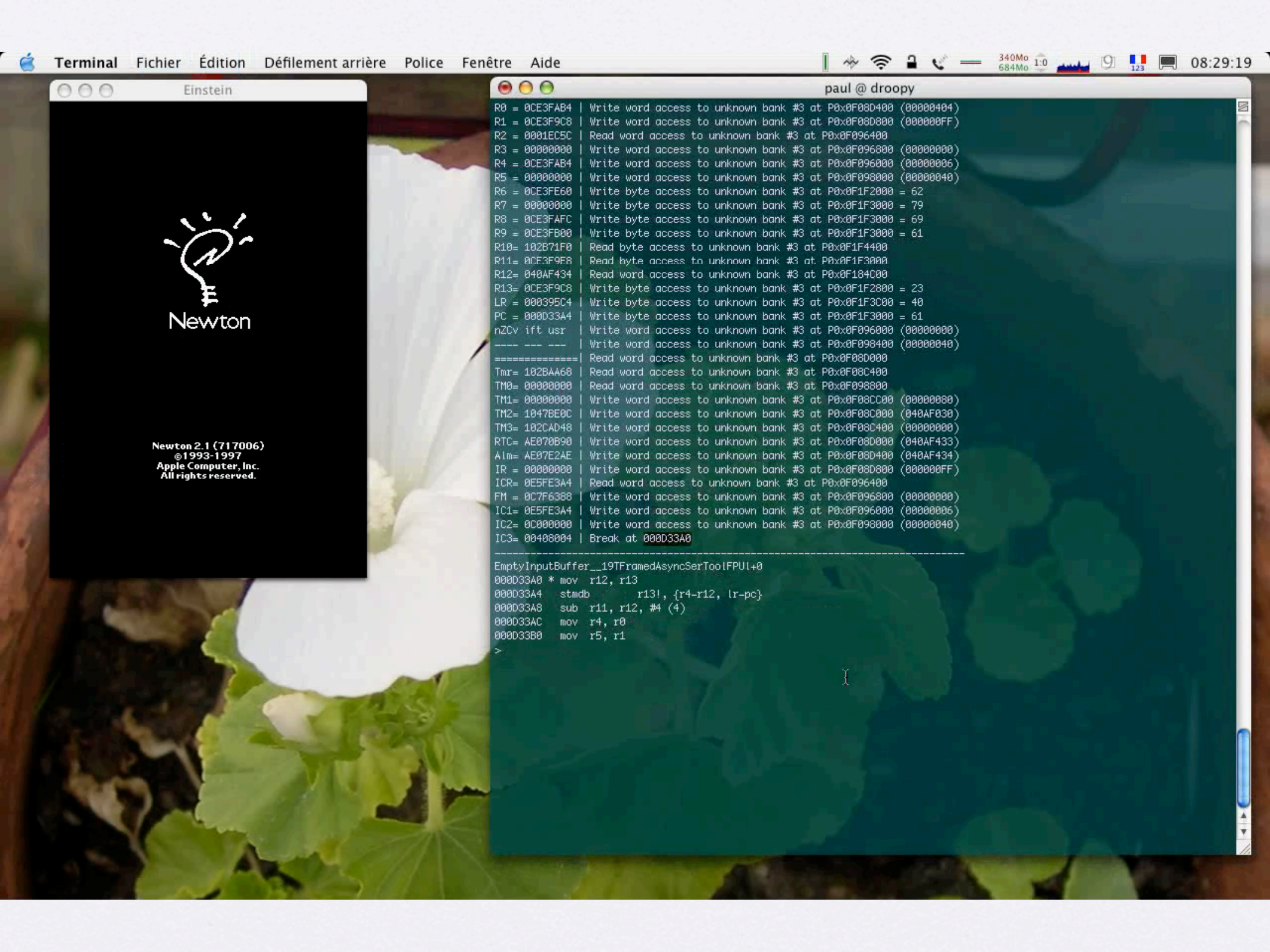
The problem of the Voyager Chipset

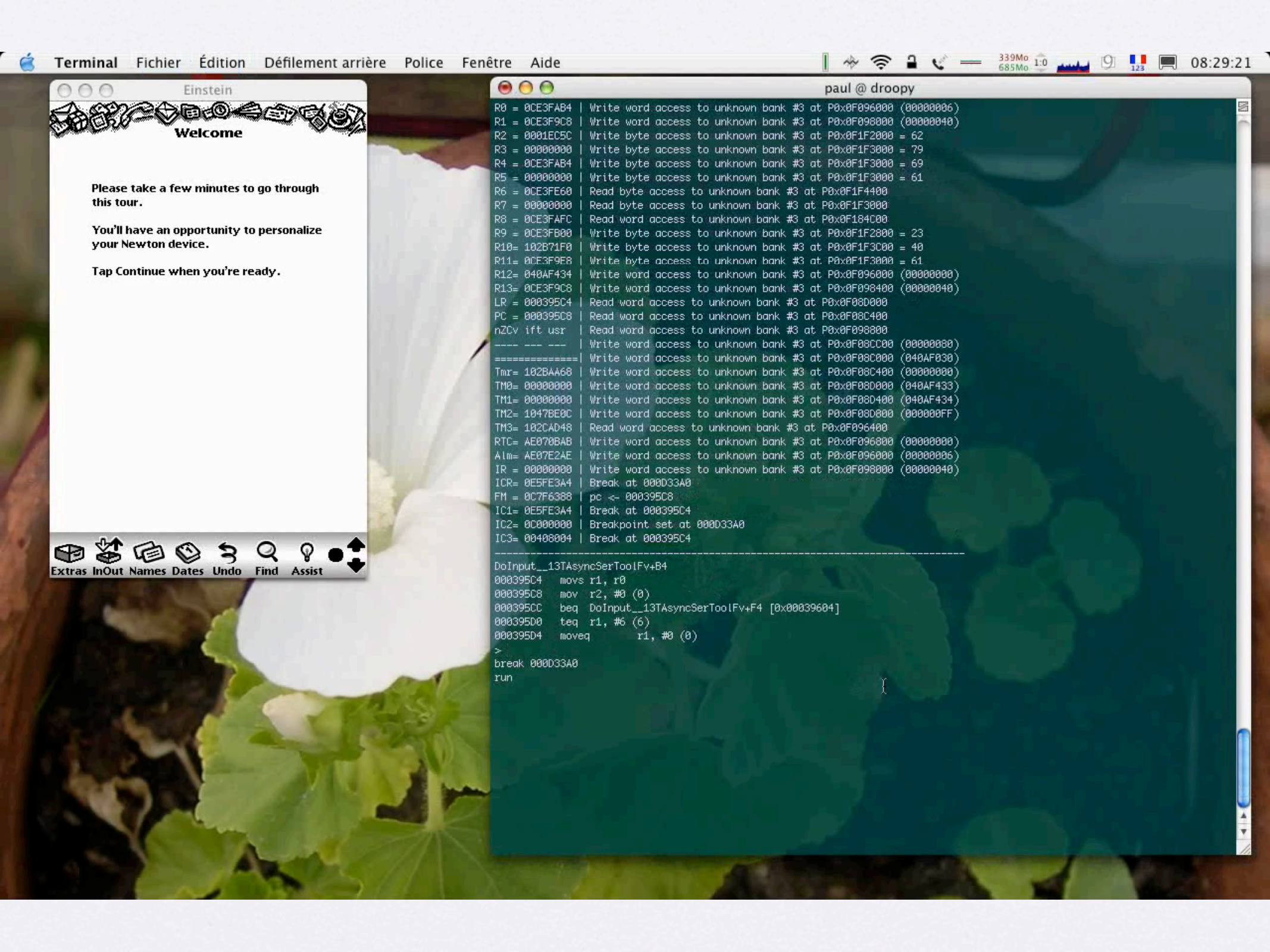
- “Perhaps one day, [Brian Parker (of Palm Emulator fame)] and Paul Guyot will team up and write the Newton emulator for the rest of us. Where oh where can we get the Voyager chipset instructions?” (Adam Tow, 2004)
  - Cirrus developed it with Apple and will not produce any or give us the specifications.

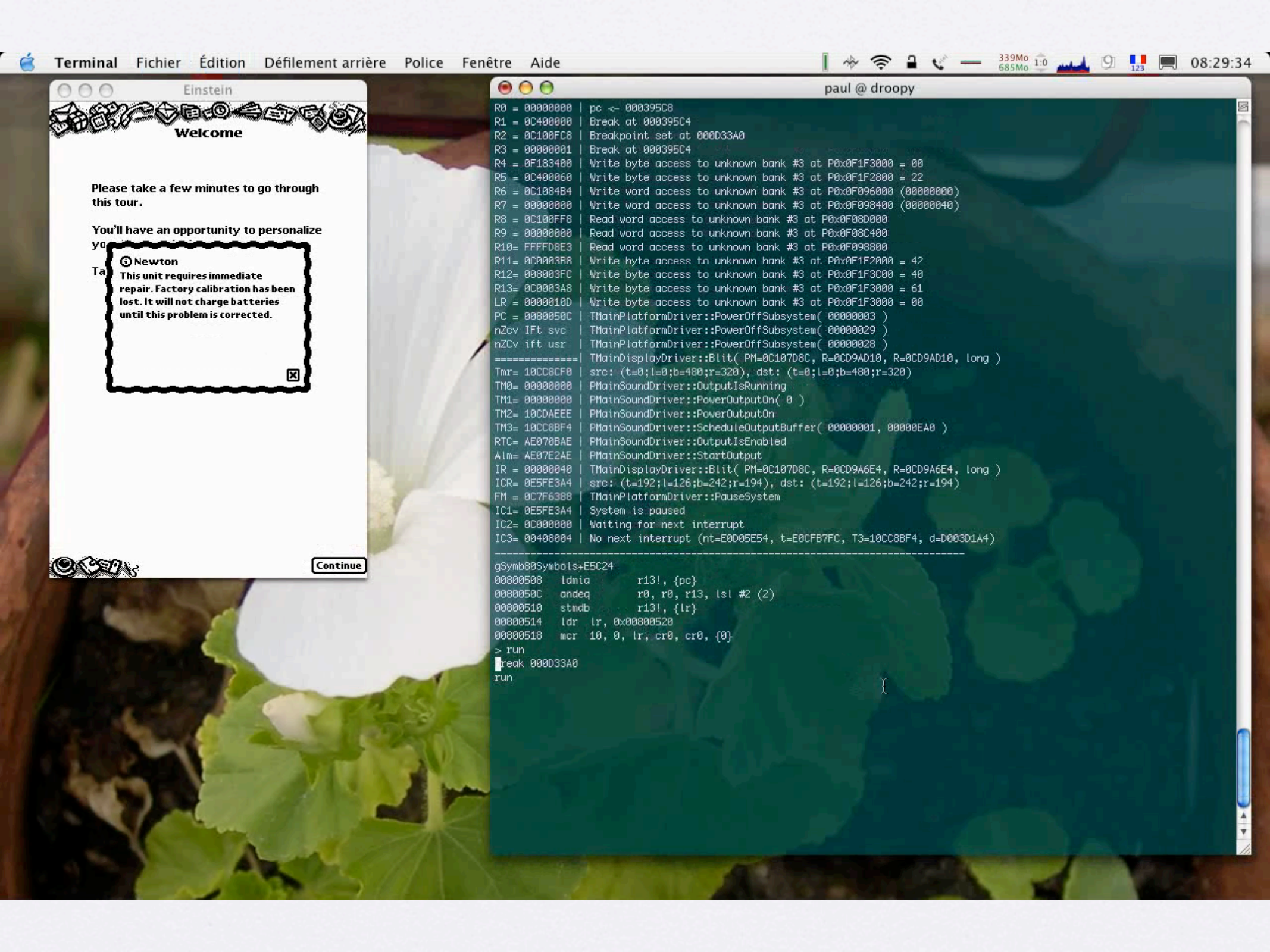
The complexity of an emulator

- “A Newton emulator has to emulate not only the ARM CPU but all the specialized ARM MMU functions and the custom hardware for DMA, IR, Flash, display, etc, etc. Not a simple job. I would estimate (and I have 30 years experience in systems and software engineering) that this is at least 5 man years of effort; attempting it without the detailed hardware documentation for a Newton MP2000 would almost certainly fail.” (John Arkley, Apple Employee #88, 04/1999)

Is it pointless?


Terminal Fichier Édition Défilement arrière Police Fenêtre Aide
08:29:19
Einstein
Newton
Newton 2.1 (717006)
©1993-1997
Apple Computer, Inc.
All rights reserved.
paul @ droopy
R0 = 0CE3FAB4 | Write word access to unknown bank #3 at P0x0F08D400 (00000404)
R1 = 0CE3F9C8 | Write word access to unknown bank #3 at P0x0F08D800 (000000FF)
R2 = 0001EC5C | Read word access to unknown bank #3 at P0x0F096400
R3 = 00000000 | Write word access to unknown bank #3 at P0x0F096800 (00000000)
R4 = 0CE3FAB4 | Write word access to unknown bank #3 at P0x0F096000 (00000006)
R5 = 00000000 | Write word access to unknown bank #3 at P0x0F098000 (00000040)
R6 = 0CE3FE60 | Write byte access to unknown bank #3 at P0x0F1F2000 = 62
R7 = 00000000 | Write byte access to unknown bank #3 at P0x0F1F3000 = 79
R8 = 0CE3FAFC | Write byte access to unknown bank #3 at P0x0F1F3000 = 69
R9 = 0CE3FB00 | Write byte access to unknown bank #3 at P0x0F1F3000 = 61
R10= 102B71F0 | Read byte access to unknown bank #3 at P0x0F1F4400
R11= 0CE3F9E8 | Read byte access to unknown bank #3 at P0x0F1F3000
R12= 040AF434 | Read word access to unknown bank #3 at P0x0F184C00
R13= 0CE3F9C8 | Write byte access to unknown bank #3 at P0x0F1F2800 = 23
LR = 000395C4 | Write byte access to unknown bank #3 at P0x0F1F3C00 = 40
PC = 000D33A4 | Write byte access to unknown bank #3 at P0x0F1F3000 = 61
nZCv ift usr | Write word access to unknown bank #3 at P0x0F096000 (00000000)
---- --- --- | Write word access to unknown bank #3 at P0x0F098400 (00000040)
==========| Read word access to unknown bank #3 at P0x0F08D000
Tmr= 102BAA68 | Read word access to unknown bank #3 at P0x0F08C400
TM0= 00000000 | Read word access to unknown bank #3 at P0x0F098800
TM1= 00000000 | Write word access to unknown bank #3 at P0x0F08CC00 (00000080)
TM2= 1047BE0C | Write word access to unknown bank #3 at P0x0F08C000 (040AF030)
TM3= 102CAD48 | Write word access to unknown bank #3 at P0x0F08C400 (00000000)
RTC= AE070B90 | Write word access to unknown bank #3 at P0x0F08D000 (040AF433)
Alm= AE07E2AE | Write word access to unknown bank #3 at P0x0F08D400 (040AF434)
IR = 00000000 | Write word access to unknown bank #3 at P0x0F08D800 (000000FF)
ICR= 0E5FE3A4 | Read word access to unknown bank #3 at P0x0F096400
FM = 0C7F6388 | Write word access to unknown bank #3 at P0x0F096800 (00000000)
IC1= 0E5FE3A4 | Write word access to unknown bank #3 at P0x0F096000 (00000006)
IC2= 0C000000 | Write word access to unknown bank #3 at P0x0F098000 (00000040)
IC3= 00408004 | Break at 000D33A0
EmptyInputBuffer__19TFramedAsyncSerToolFPUl+0
000D33A0 * mov r12, r13
000D33A4 stmdb r13!, {r4-r12, lr-pc}
000D33A8 sub r11, r12, #4 (4)
000D33AC mov r4, r0
000D33B0 mov r5, r1
>


Terminal Fichier Édition Défilement arrière Police Fenêtre Aide
08:29:21
Einstein
Welcome
Please take a few minutes to go through this tour.
You'll have an opportunity to personalize your Newton device.
Tap Continue when you're ready.
Extras InOut Names Dates Undo Find Assist
paul @ droopy
R0 = 0CE3FAB4 | Write word access to unknown bank #3 at P0x0F096000 (00000006)
R1 = 0CE3F9C8 | Write word access to unknown bank #3 at P0x0F098000 (00000040)
R2 = 0001EC5C | Write byte access to unknown bank #3 at P0x0F1F2000 = 62
R3 = 00000000 | Write byte access to unknown bank #3 at P0x0F1F3000 = 79
R4 = 0CE3FAB4 | Write byte access to unknown bank #3 at P0x0F1F3000 = 69
R5 = 00000000 | Write byte access to unknown bank #3 at P0x0F1F3000 = 61
R6 = 0CE3FE60 | Read byte access to unknown bank #3 at P0x0F1F4400
R7 = 00000000 | Read byte access to unknown bank #3 at P0x0F1F3000
R8 = 0CE3FAFC | Read word access to unknown bank #3 at P0x0F184C00
R9 = 0CE3FB00 | Write byte access to unknown bank #3 at P0x0F1F2800 = 23
R10= 102B71F0 | Write byte access to unknown bank #3 at P0x0F1F3C00 = 40
R11= 0CE3F9E8 | Write byte access to unknown bank #3 at P0x0F1F3000 = 61
R12= 040AF434 | Write word access to unknown bank #3 at P0x0F096000 (00000000)
R13= 0CE3F9C8 | Write word access to unknown bank #3 at P0x0F098400 (00000040)
LR = 000395C4 | Read word access to unknown bank #3 at P0x0F08D000
PC = 000395C8 | Read word access to unknown bank #3 at P0x0F08C400
nZCv ift usr | Read word access to unknown bank #3 at P0x0F098800
---- --- --- | Write word access to unknown bank #3 at P0x0F08CC00 (00000080)
==========| Write word access to unknown bank #3 at P0x0F08C000 (040AF030)
Tmr= 102BAA68 | Write word access to unknown bank #3 at P0x0F08C400 (00000000)
TM0= 00000000 | Write word access to unknown bank #3 at P0x0F08D000 (040AF433)
TM1= 00000000 | Write word access to unknown bank #3 at P0x0F08D400 (040AF434)
TM2= 1047BE0C | Write word access to unknown bank #3 at P0x0F08D800 (000000FF)
TM3= 102CAD48 | Read word access to unknown bank #3 at P0x0F096400
RTC= AE070BAB | Write word access to unknown bank #3 at P0x0F096800 (00000000)
Alm= AE07E2AE | Write word access to unknown bank #3 at P0x0F096000 (00000006)
IR = 00000000 | Write word access to unknown bank #3 at P0x0F098000 (00000040)
ICR= 0E5FE3A4 | Break at 000D33A0
FM = 0C7F6388 | pc <- 000395C8
IC1= 0E5FE3A4 | Break at 000395C4
IC2= 0C000000 | Breakpoint set at 000D33A0
IC3= 00408004 | Break at 000395C4
DoInput__13TAsyncSerToolFv+B4
000395C4 movs r1, r0
000395C8 mov r2, #0 (0)
000395CC beq DoInput__13TAsyncSerToolFv+F4 [0x00039604]
000395D0 teq r1, #6 (6)
000395D4 moveq r1, #0 (0)
>
break 000D33A0
run


Terminal Fichier Édition Défilement arrière Police Fenêtre Aide
339Mo 685Mo
08:29:34
Einstein
Welcome
Please take a few minutes to go through this tour.
You'll have an opportunity to personalize yo
Ta
Newton
This unit requires immediate repair. Factory calibration has been lost. It will not charge batteries until this problem is corrected.
Continue
paul @ droopy
R0 = 00000000 | pc <- 000395C8
R1 = 0C400000 | Break at 000395C4
R2 = 0C100FC8 | Breakpoint set at 000D33A0
R3 = 00000001 | Break at 000395C4
R4 = 0F183400 | Write byte access to unknown bank #3 at P0x0F1F3000 = 00
R5 = 0C400060 | Write byte access to unknown bank #3 at P0x0F1F2800 = 22
R6 = 0C1084B4 | Write word access to unknown bank #3 at P0x0F096000 (00000000)
R7 = 00000000 | Write word access to unknown bank #3 at P0x0F098400 (00000040)
R8 = 0C100FF8 | Read word access to unknown bank #3 at P0x0F08D000
R9 = 00000000 | Read word access to unknown bank #3 at P0x0F08C400
R10= FFFFD8E3 | Read word access to unknown bank #3 at P0x0F098800
R11= 0C0003B8 | Write byte access to unknown bank #3 at P0x0F1F2000 = 42
R12= 000003FC | Write byte access to unknown bank #3 at P0x0F1F3C00 = 40
R13= 0C0003A8 | Write byte access to unknown bank #3 at P0x0F1F3000 = 61
LR = 0000010D | Write byte access to unknown bank #3 at P0x0F1F3000 = 00
PC = 0080050C | TMainPlatformDriver::PowerOffSubsystem(00000003)
nZcv IFt svc | TMainPlatformDriver::PowerOffSubsystem(00000029)
nZCv ift usr | TMainPlatformDriver::PowerOffSubsystem(00000028)
==========| TMainDisplayDriver::Blit(PM=0C107D8C, R=0CD9AD10, R=0CD9AD10, long)
Tmr= 10CC8CF0 | src: (t=0;l=0;b=480;r=320), dst: (t=0;l=0;b=480;r=320)
TM0= 00000000 | PMainSoundDriver::OutputIsRunning
TM1= 00000000 | PMainSoundDriver::PowerOutputOn(0)
TM2= 10CDAEEE | PMainSoundDriver::PowerOutputOn
TM3= 10CC8BF4 | PMainSoundDriver::ScheduleOutputBuffer(00000001, 00000EA0)
RTC= AE070BAE | PMainSoundDriver::OutputIsEnabled
Alm= AE07E2AE | PMainSoundDriver::StartOutput
IR = 00000040 | TMainDisplayDriver::Blit(PM=0C107D8C, R=0CD9A6E4, R=0CD9A6E4, long)
ICR= 0E5FE3A4 | src: (t=192;l=126;b=242;r=194), dst: (t=192;l=126;b=242;r=194)
FM = 0C7F6388 | TMainPlatformDriver::PauseSystem
IC1= 0E5FE3A4 | System is paused
IC2= 0C000000 | Waiting for next interrupt
IC3= 00408004 | No next interrupt (nt=E0D05E54, t=E0CFB7FC, T3=10CC8BF4, d=D003D1A4)
gSymb80Symbols+E5C24
00800508 ldmia r13!, {pc}
0080050C andeq r0, r0, r13, lsl #2 (2)
00800510 stmdb r13!, {lr}
00800514 ldr lr, 0x00800520
00800518 mcr 10, 0, lr, cr0, cr0, {0}
> run
break 000D33A0
run

How Einstein Emulator works and how we will be able to port NewtonOS

The N2 Platform (2.1 units)

- Apple meant to license the N2 platform (2.1 units) to third party hardware manufacturer
  - Schlumberger's Watson has custom hardware
- There is room for ROM Extensions (REXes) coming after Apple's ROM.
  - Drivers for custom hardware
  - Patches
  - Packages

The P-Class mechanism (i)

- NewtonOS is one of the first operating system written in an object oriented programming language (C++, not NewtonScript)
- C++ lacks dynamism with interfaces and implementations being chosen at runtime. In other words, drivers cannot be written without some glue.
- NewtonOS team defined P-Classes with drivers in a registry that can be replaced (overridden)

The P-Class mechanism (ii)

- There is an interface called TStore for storage of objects on flash memory.
- There are two implementations in NewtonOS: TPackageStore (for read-only stores in Newton packages) and TFlashStore (for internal memory and linear cards)
- One can provide a new implementation called TATAStore for ATA cards
- Same with, e.g., sound codecs (built-in: GSM, DTMF, provided by packages: MP3, Macintalk)

The P-Class mechanism (iii)

```
#include <Newton.h>
#include <Protocols.h>
#include <NewtQD.h>

struct ScreenInfo {
    ULong   fScreenHeight;   // 00000140 (320) height
    ULong   fScreenWidth;    // 000001E0 (480) width
    ULong   fBitsPerPixel;   // 00000004 (depth?)
    ULong   fUnknown_0C;     // 00000037 55 (?)
    UShort  fResolutionX;    // 00640064 100/100 (bpi?)
    UShort  fResolutionY;    //
    ULong   fUnknown_14;     // 00000020 32 (?)
    ULong   fUnknown_18;     // 00000020 32 (?)
};

///
/// Protocol for the screen driver.
///
/// \author Paul Guyot <pguyot@kallisys.net>
/// \author Nicolas Zinovieff <krugazor@poulet.org>
///
PROTOCOL TScreenDriver : public TProtocol
{
public:
    void    Delete();
    ULong   ScreenSetup();
    ULong   GetScreenInfo(ScreenInfo*);
    ULong   PowerInit();
    ULong   PowerOn();
    ULong   PowerOff();
    ULong   Blit(PixelMap*, Rect*, Rect*, long);
    long    GetFeature(long);
    void    SetFeature(long, long);
    ULong   AutoAdjustFeatures();
    ULong   DoubleBlit(PixelMap*, PixelMap*, Rect*, Rect*, long);
    ULong   EnterIdleMode();
    ULong   ExitIdleMode();
};

#endif
        // _TSCREENDRIVER_H
```

- There is a P-Class interface for the platform driver and an implementation for the Voyager Chipset. Hence we do not need to know the exact Voyager specifications
- There are P-Classes for other hardware elements : battery, screen, tablet, sound, etc.

What's next?


Einstellungen
Ausschalten
Backup Restore
Erscheinungsbild
Datum/Zeit
Kalibrieren
Beam Receive
Internet Wizard
Sicherheitseinstellungen
Helligkeit
Netzwerk
SHARP
Cancel
On/Off
OK
Space
Fn
Tab

Future of the Emulator itself

- The emulator allowed/will allow us to :
  - Understand the boot problems of some units
  - Understand and fix the calibration problem
  - Improve and develop more easily programs that access to the bowels of the NewtonOS
  - Read flash cards directly on a laptop and exchange data more easily (maybe coupled with the DCL)
  - Work & test system patches without damaging any unit

Porting NewtonOS

- The emulator allowed us to discover the basic hardware locations required to run NewtonOS (e.g. timers) and the specifications of the drivers (e.g. we know how to turn the backlight)
- We could either
  - run the emulator (with JIT for a decent speed) on a PDA
  - or run NewtonOS natively with custom drivers and little modifications

My suggestion

- The emulator solves the hardware problem. However, NewtonOS lacks up-to-date software for web browsing, cryptographic, modern networking.
  - We could put NewtonOS on top of Unix and take advantage of all the open source software running on Unix
  - We then could extend NewtonOS with hooks to allow applications to take advantage of this code

Conclusion

Conclusion

- Einstein is a set of propositions for a community-run future for the platform.
- These propositions have in common a migration to new hardware and no complete rewrite of NewtonOS.
- The emulator is just the first step.

New post-Apple era?

Questions ?

pguyot@kallisys.net
http://www.kallisys.com/