Einstein *Roadmap*

Einstein Roadmap
a presentation by
Paul Guyot (pguyot@kallisys.net)
WWNC 2007 - 2007/7/8

Part I: Prologue

Many thanks to:

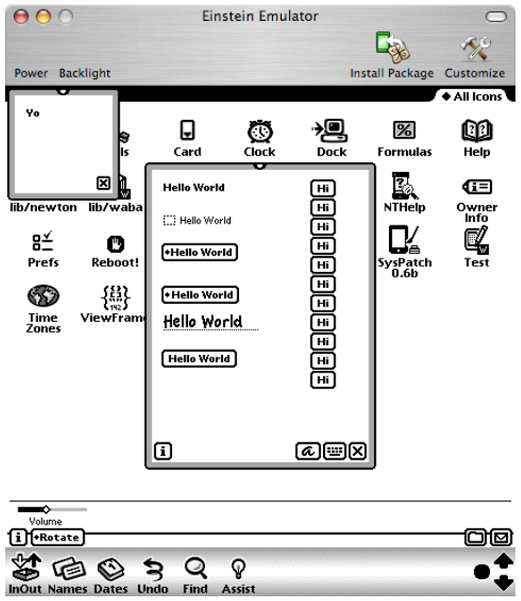
Basuke
Simon Bell
Martin Buis
Andy Diller
Frank Gründel
Lars Immisch
Eckhart Köppen
Sean Luke
Matthias Melcher
Makoto Nukui
Sylvain Pilet
Victor Rehorst
Walter Smith
Adam Tow
Michael Vacík
David Watson
Larry Yaeger
Nicolas Zinovieff

You, and many others…

Einstein is NewtonOS 2.1 running on new hardware

Einstein is NewtonOS 2.1 running on new hardware

- Mac (2004/9)



Einstein is NewtonOS 2.1 running on new hardware

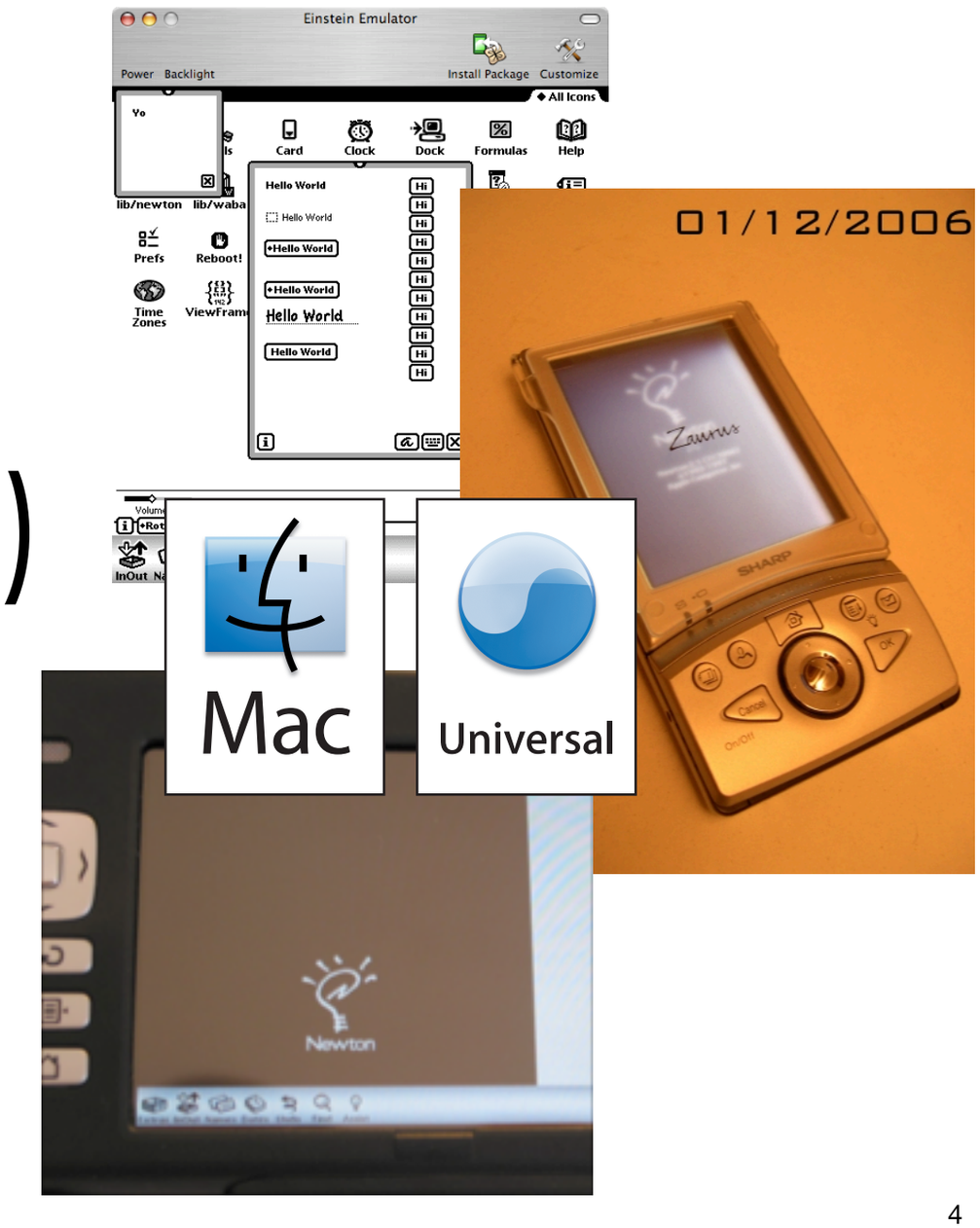
- Mac (2004/9)
- Zaurus (2006/1)

Einstein is NewtonOS 2.1 running on new hardware

- Mac (2004/9)
- Zaurus (2006/1)
- Nokia 770 (2006/1)

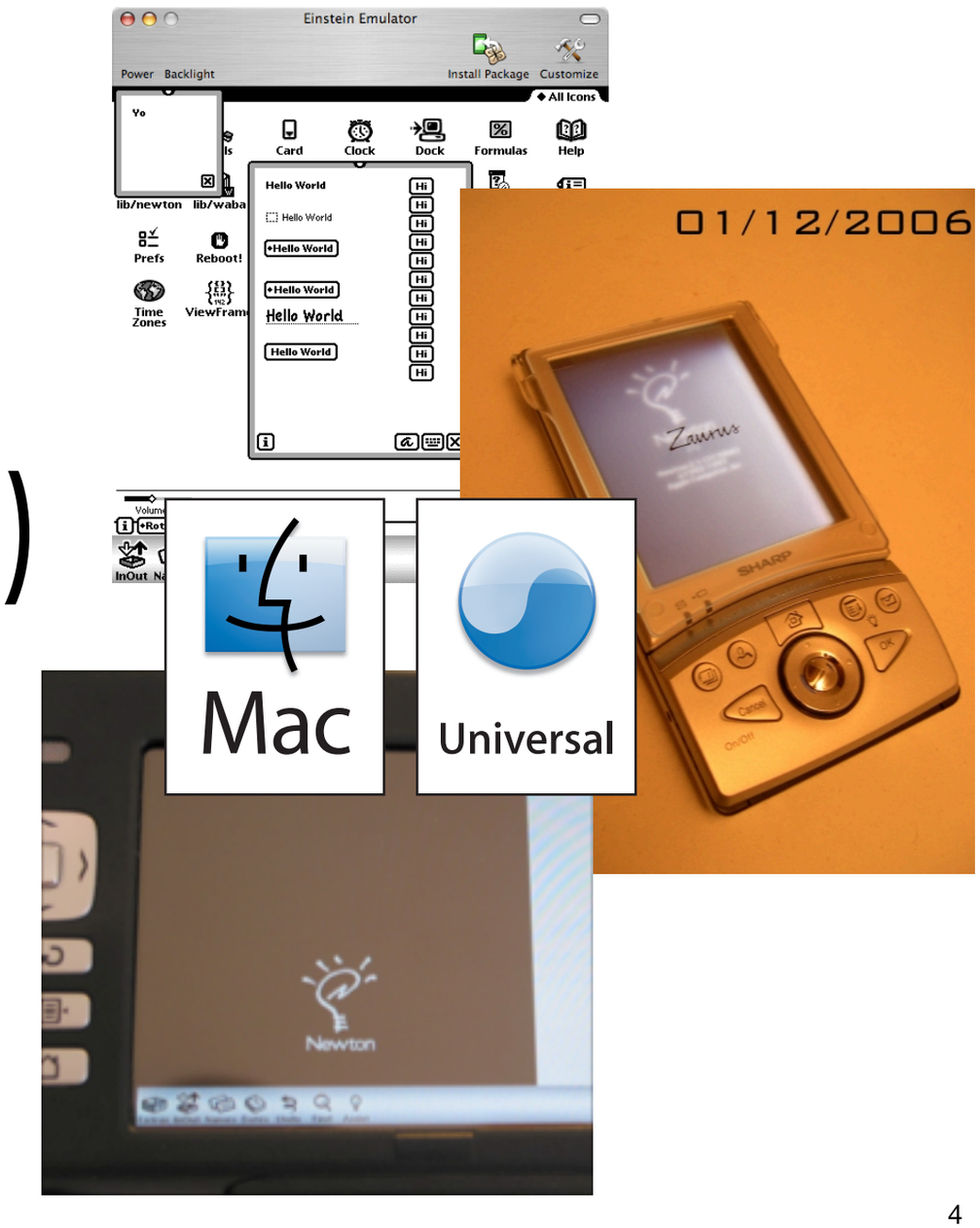
Einstein is NewtonOS 2.1 running on new hardware

- Mac (2004/9)
- Zaurus (2006/1)
- Nokia 770 (2006/1)
- MacIntel (2006/5)



Einstein is NewtonOS 2.1 running on new hardware

- Mac (2004/9)
- Zaurus (2006/1)
- Nokia 770 (2006/1)
- MacIntel (2006/5)
- Linux x86 (2006/5)

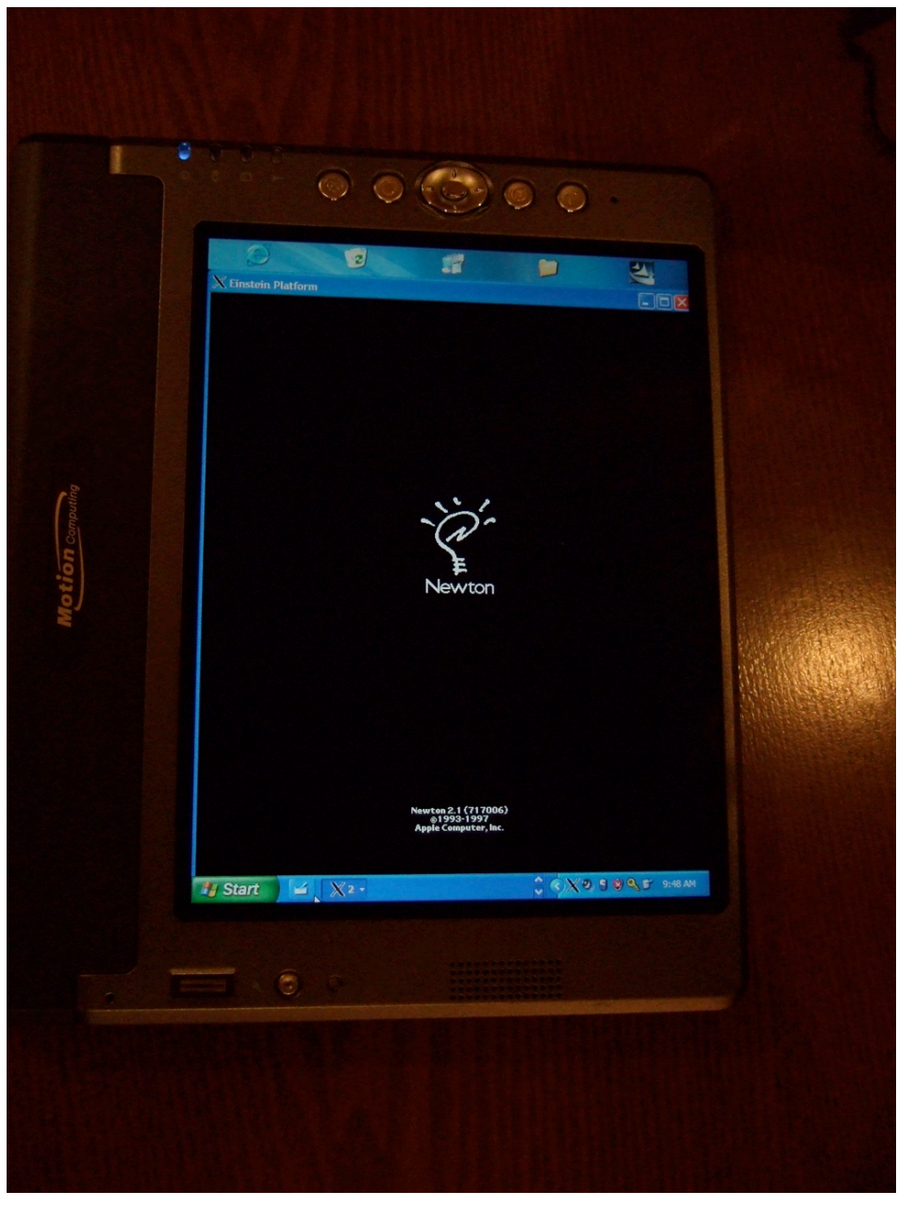


Einstein is NewtonOS 2.1 running on new hardware

- Mac (2004/9)
- Zaurus (2006/1)
- Nokia 770 (2006/1)
- MacIntel (2006/5)
- Linux x86 (2006/5)
- Nokia 800 (2007/7)

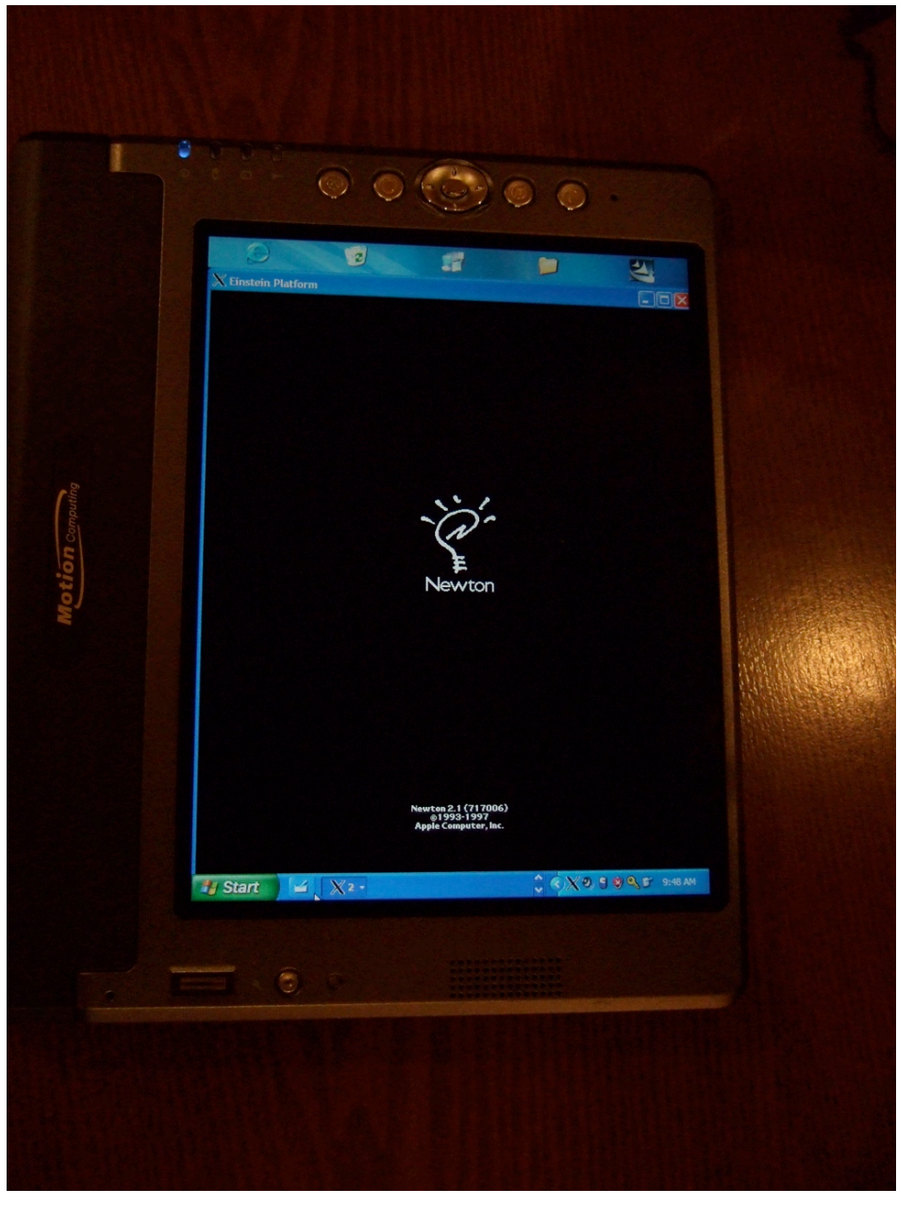
Einstein is NewtonOS 2.1 running on new hardware

- Windows (2007/7)



Einstein is NewtonOS 2.1 running on new hardware

- Windows (2007/7)
- iPhone (?)



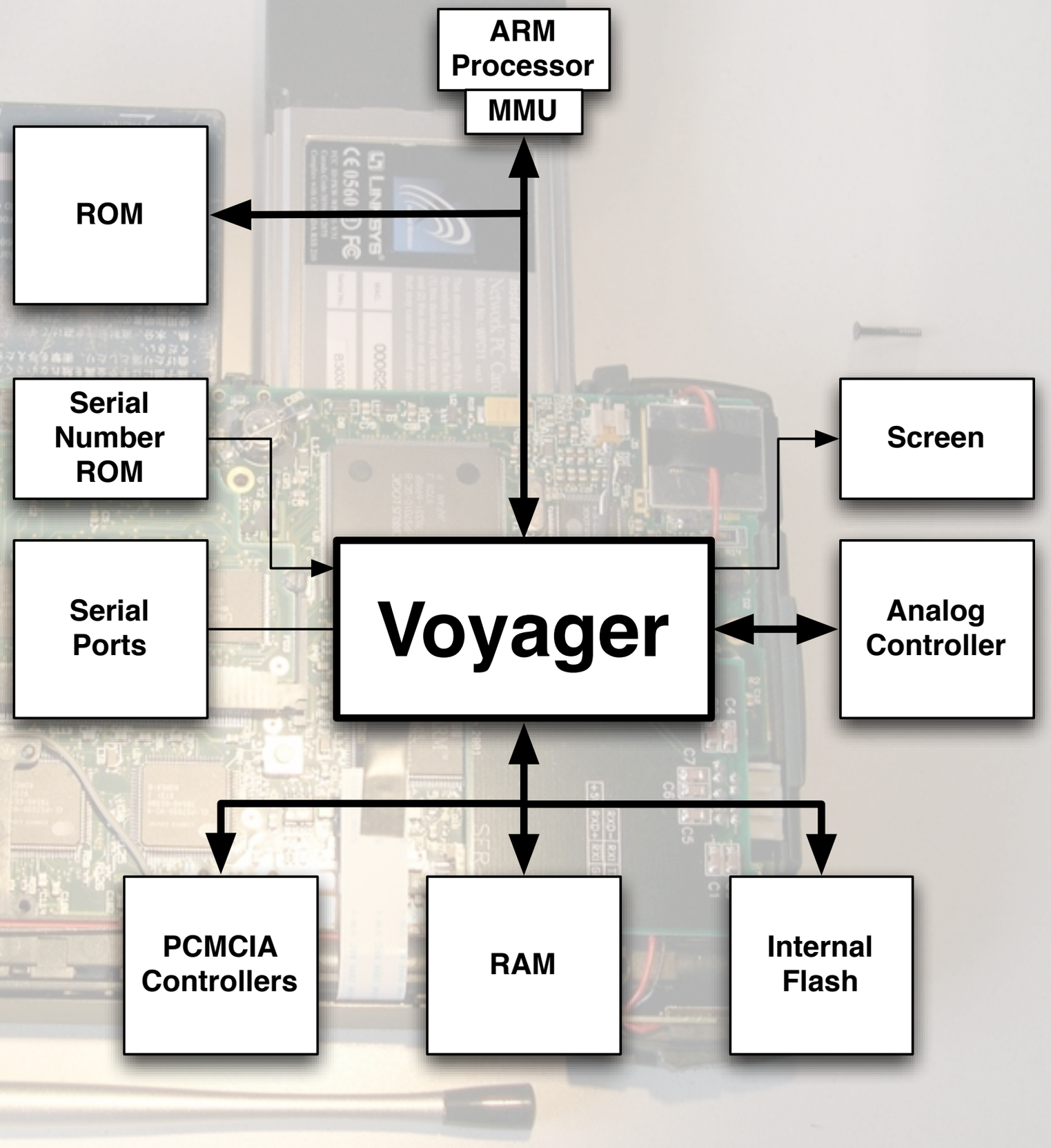
Three goals in 2004

- ✓Get an emulator for NewtonOS development (2005)
- ✓Extend NewtonOS with modern technologies (2006)
- -Replace Newton MessagePads with modern PDAs

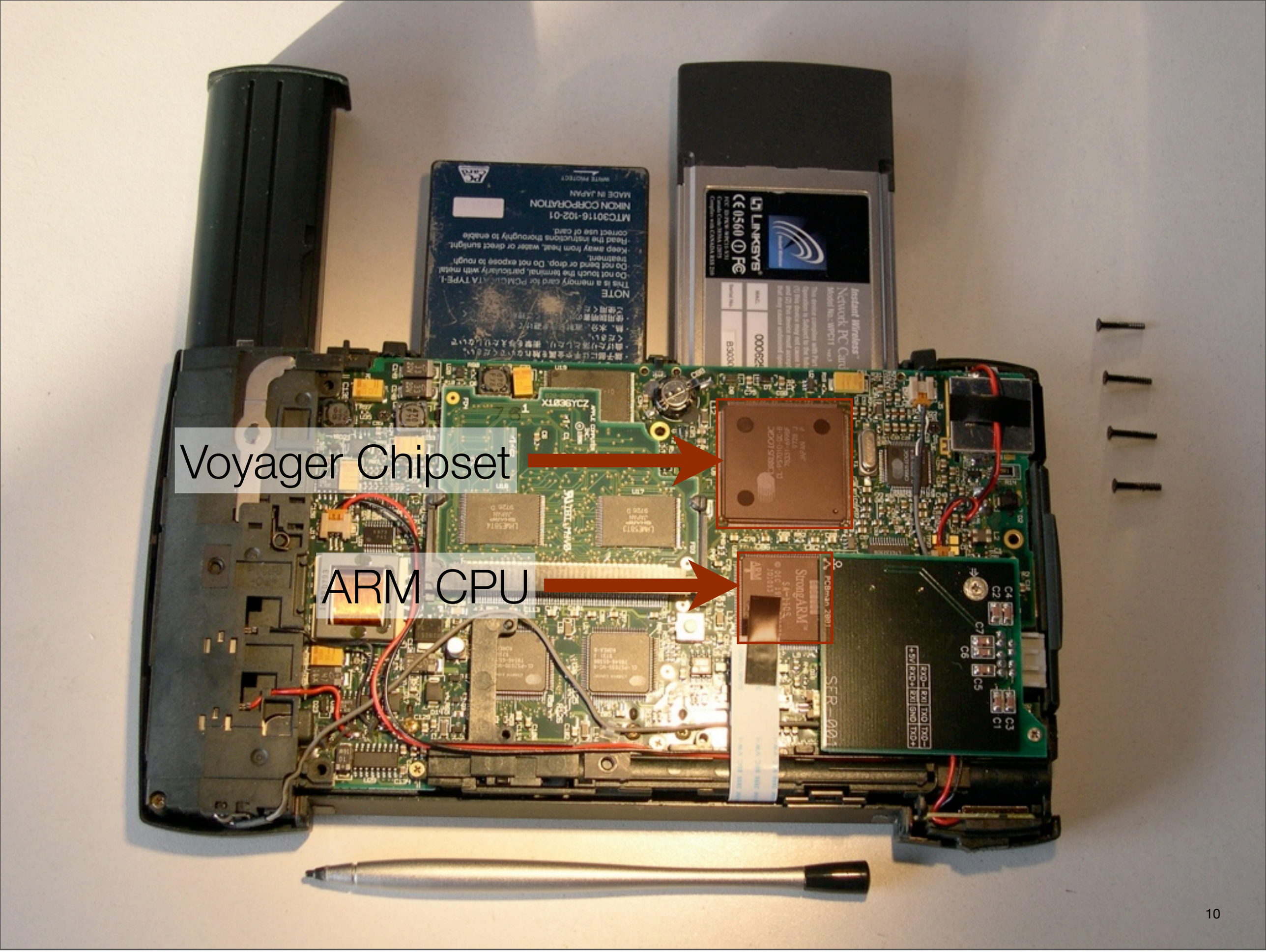
Part II: Inside Einstein

How Einstein was born

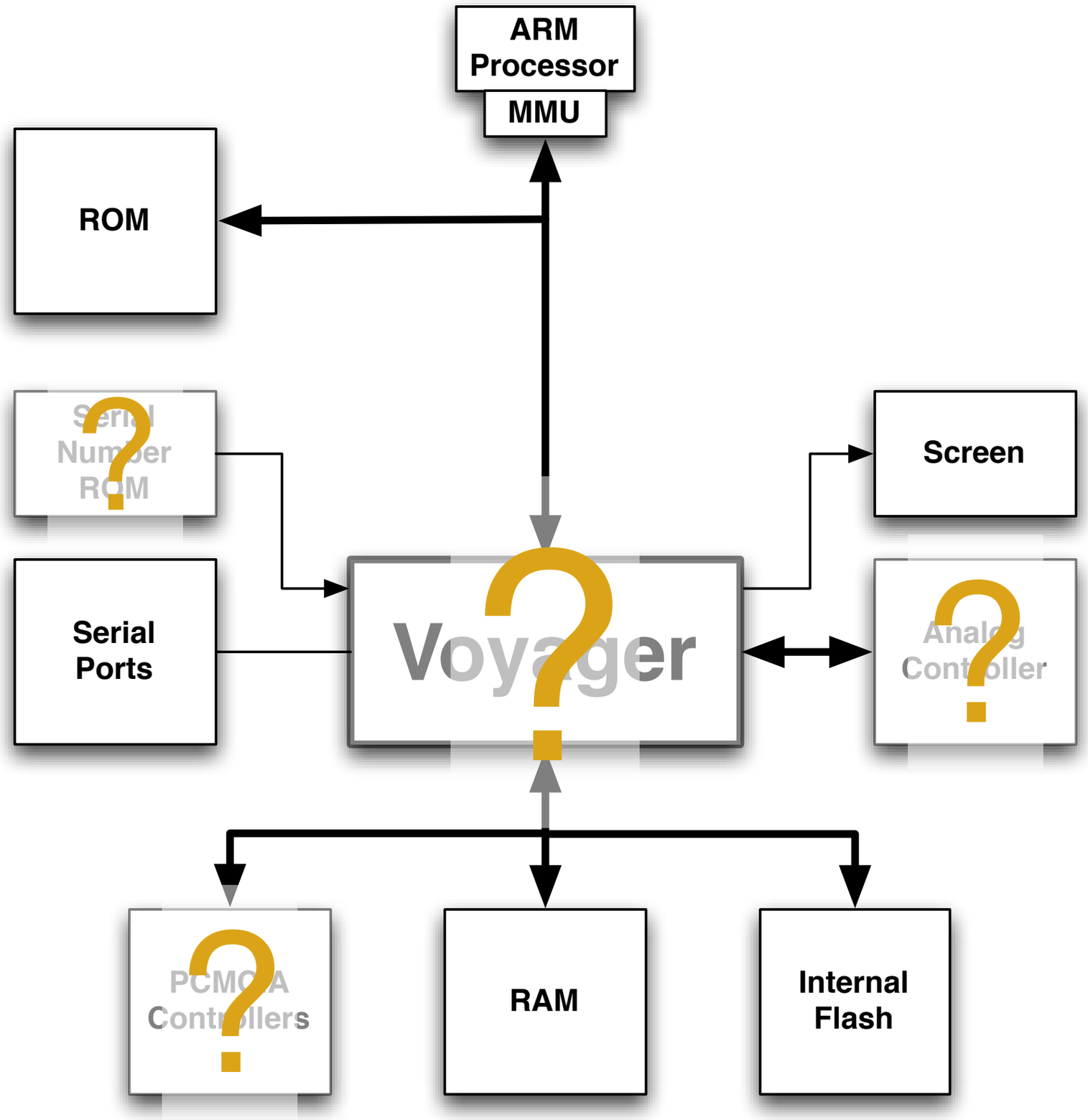
Newton 2.1 hardware is based on the Voyager Chipset



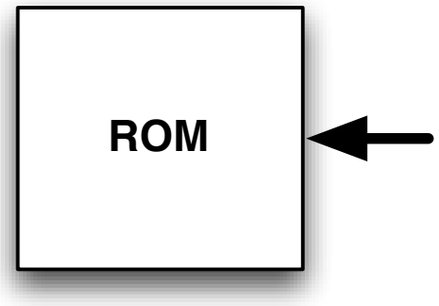

LINKSYS
CE 0560
Instant Wireless
Network PC Card
Model No. WPC11
NOTE
MADE IN JAPAN
StrongARM
SER_001


Voyager Chipset
ARM CPU

We do not know how the Voyager Chipset works



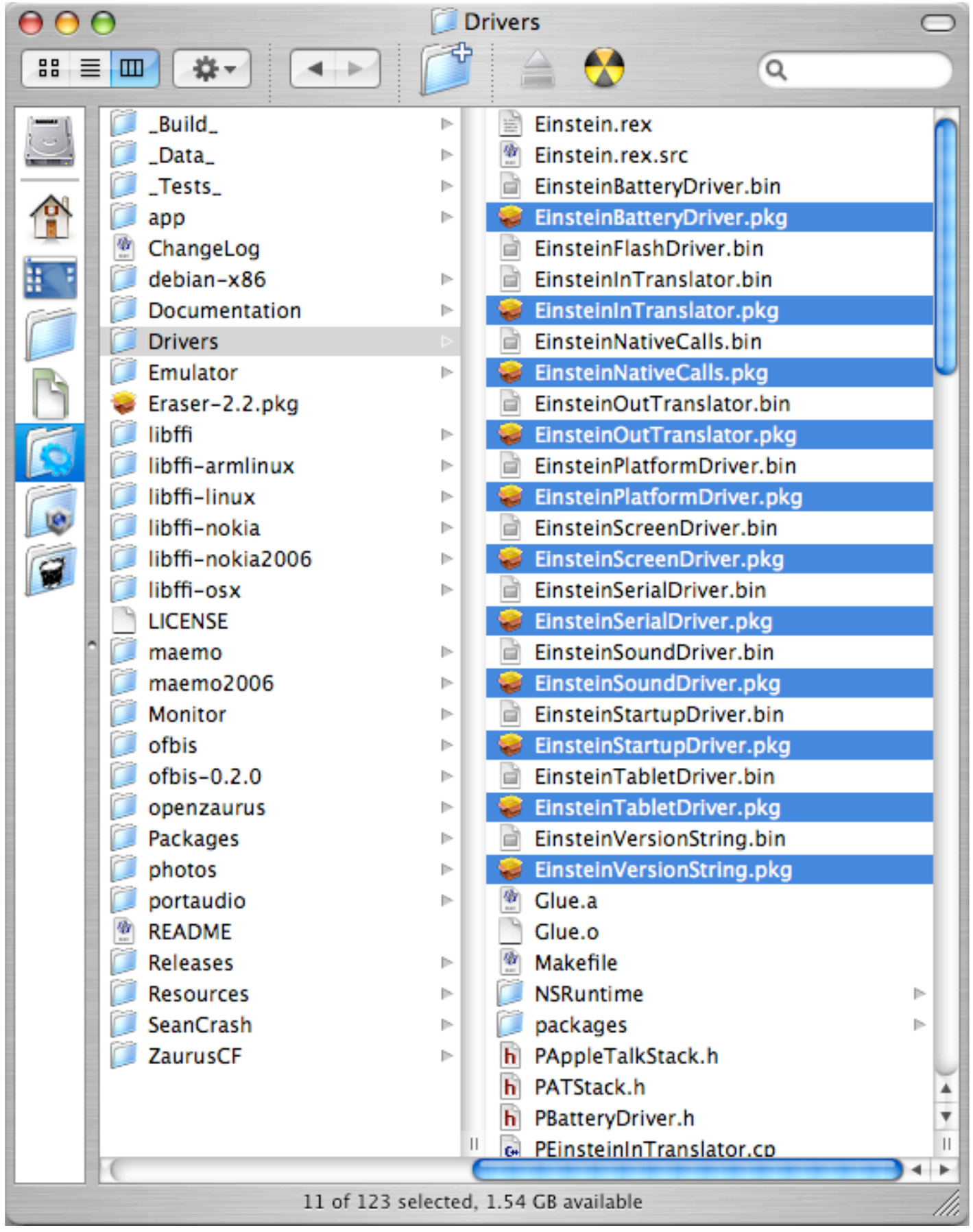
The ARM Processor in NewtonOS 2.1 mostly accesses the Voyager Chipset through **drivers** called “**PClasses**”



It accesses it directly for the timer (interruptions)

Einstein provides its own drivers inside the **Einstein REX**

(ROM Extension)



The Speed Challenge

In 1997, Newtons were **very fast**:
StrongARM 110 at 161.9MHz.

In 1997, Newtons were **very fast**:
StrongARM 110 at 161.9MHz.

1997: Pentium II (233-266MHz)
1997: PowerPC G3 (366MHz)

In 1997, Newtons were **very fast**:
StrongARM 110 at 161.9MHz.

1997: Pentium II (233-266MHz)
1997: PowerPC G3 (366MHz)

2007: MacPro: 3 GHz
2007: iPhone: 620 Mhz (ARM!)

The principle of emulation (the slow way, Einstein 2004)

1. Update the Program Counter (PC)
2. Get the physical address of the instruction
3. Read the instruction
4. Analyze it
5. Do what it should do
6. Determine if there is any interruption
7. Repeat

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=0001868C

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=00018690

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=00018694

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=00018698

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=0001869C

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

PC=000186A0

```
BootOS:
00018688 mov     r0, #0xB0
0001868C orr     r0, r0, #0x00001000
00018690 mcr     15, 0, r0, cr1, cr1, {0}
00018694 mrc     15, 0, r0, cr0, cr0, {0}
00018698 bic     r0, r0, #0xF
0001869C eor     r0, r0, #0x44000000
000186A0 eor     r0, r0, #0x00010000
000186A4 eors    r0, r0, #0x0000A100
```

The Program Counter

The program counter is the (virtual) address of the current instruction (+4). It is where the software is currently executing.

Updating the PC is expensive.

Einstein 2007 only updates it when necessary.

Translating addresses: MMU

Modern computers have virtual and physical addresses
virtual: what the software sees
physical: what the hardware sees

NewtonOS makes a heavy use of MMU for **packages**, **virtual memory**, **memory protection.**

Einstein uses a cache for MMU since 2005

Analyzing instructions

Analyzing instructions takes a lot of time. This can be done by translating at runtime into references to native code (Just In Time, JIT).

Einstein uses threaded emulation since 2005

Einstein 2007 includes a new module to **directly translate** ARM code into ARM code (work in progress: only few instructions for now)

Interruptions

Interruptions are hardware signals sent to the processor to interrupt what it was doing and do something else instead.

Examples:

the alarm fires off -> show a dialog

you press on the screen -> do something

preemptive multithreading

Emulating interruptions is very expensive.

Interruptions

In the future, we can replace interruptions with **virtualizations**:

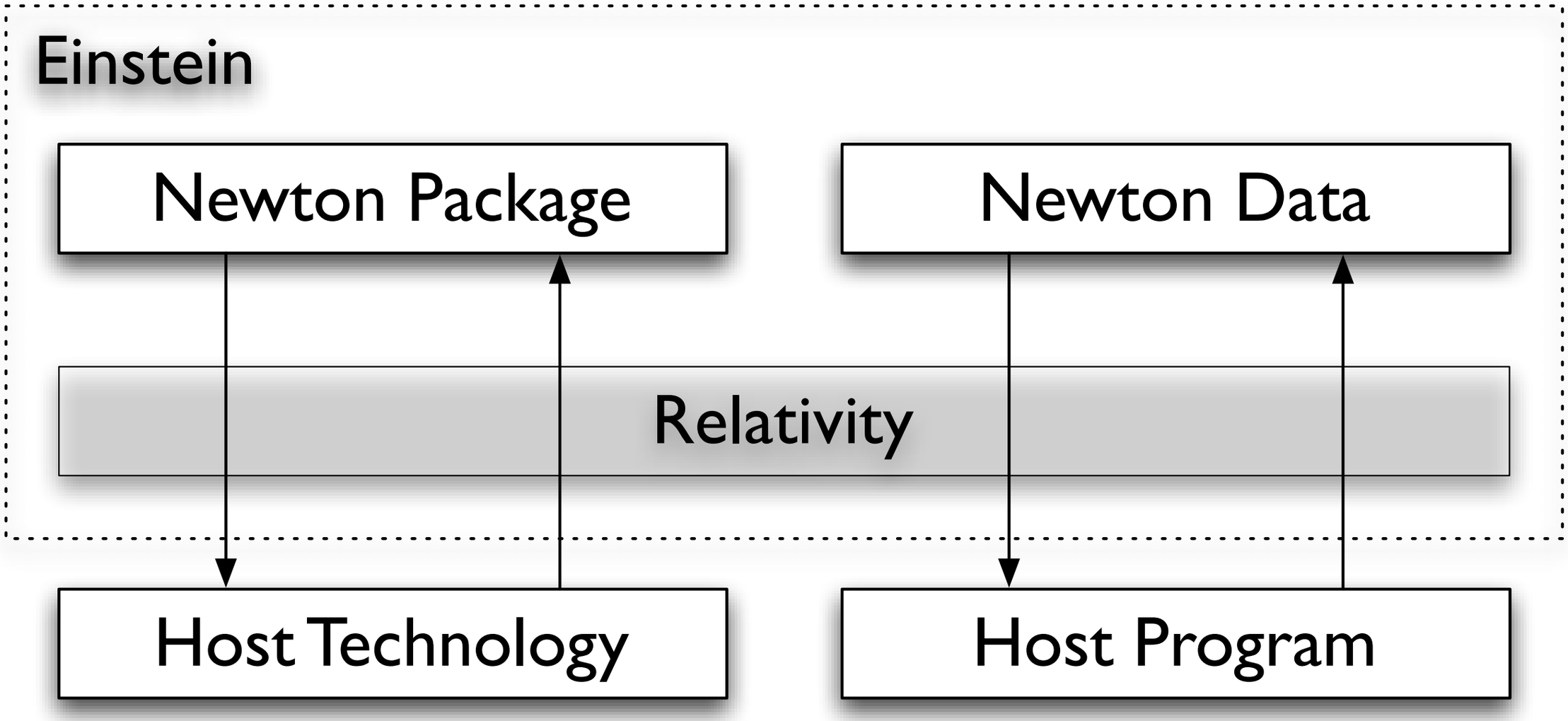
When you press the screen on a Zaurus, the Zaurus is interrupted.

— Fast interruptions (FIQ) can be virtualized as **they *do not* influence** the main process.
— Regular interruptions (IRQ) are used for preemptive multithreading. **(難しい)**

Relativity

Relativity was introduced here:

Relativity is the **integration** of host and Newton data and technologies



Examples:

— **Control host applications** (iTunes)
— Use **other languages** within Einstein (Python, Ruby, …)
— Use **Zaurus kanji handwriting recognition** within Einstein
— Use **Newton handwriting recognition** in host operating system
— **Share** Einstein and Host address books

Relativity is only limited by **your imagination**.

Part III: Epilogue

Today, Einstein **becomes**...

Open Einstein

*Open*Einstein

http://code.google.com/p/einstein/

*Open*Einstein

It will always belong to the community

GNU General Public License v2

Sharing the effort:

- Access to **more hardware** (Nokia 800, ***iPhone?***, recent Zaurus, Windows Mobile)
- **Ported on Windows yesterday with Matthias!**
- Work can be **distributed** on different modules

Together we can...

Work on **speed**…

— Make it much faster on the Zaurus/Nokia/ iPhone with **direct translation of ARM instructions**

— **Virtualize** fast interruptions, regular interruptions, memory accesses, NewtonScript bytecode interpreter (NEWT/0), and more…

Work on **host integration**…

— Emulating **serial ports**

— Sharing host **internet** access

— Integrating **soups** (data exchange)

— Emulating ATA cards to **provide storage**

— Providing **color** (cf the VGA card)

— Integrating NewtonOS windows and host windows (**NewtView**)

…and imagine the next

Einstein

…and imagine the next

Open Einstein

Thank you for your attention!