


iPhone Autopsy

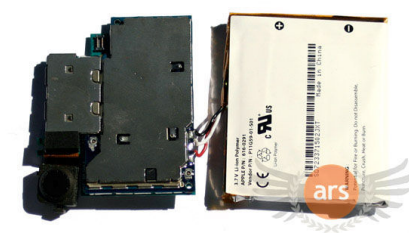
- Consist mainly battery pack and display unit



©2004 Morgan Kaufmann Publishers 1

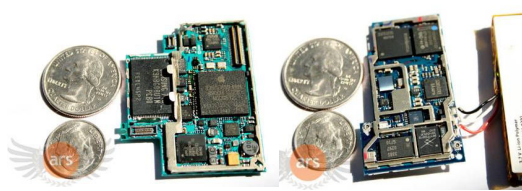
Mother and Daughter

- Mother board and daughter board snapped together
 - And put under an RF shielding



©2004 Morgan Kaufmann Publishers 2

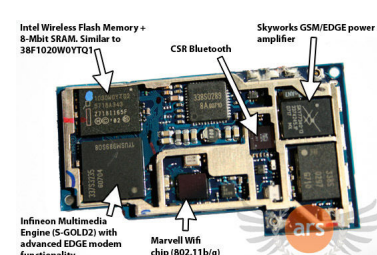
Size of processor board and daughter board



©2004 Morgan Kaufmann Publishers 3


Possible identification of the components

- Marvell Wifi (ARM based 802.11 controller)
- Intel Wireless Flash (32M) + 8-M SRAM
- Infineon Multimedia
 - Encode/decode Audio/Video, drive camera, bluetooth, and radio



©2004 Morgan Kaufmann Publishers 4

Micron image sensor

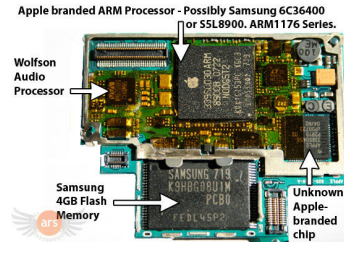


©2004 Morgan Kaufmann Publishers 5

iPhone Autopsy

- Wolfson Audio codec (sam as IPOD)
- Samsung 4GB flash

Apple branded ARM Processor - Possibly Samsung GC36400 or SSL8900, ARM1176 Series.



©2004 Morgan Kaufmann Publishers 6

Common Intermediate Language

- Lowest-level human readable language used in the .net framework
- Primary .net language are C#, Visual Basic .net, C++/CIL, j#
- Why not go directly to natively compiled binaries?
 - Verified for safety during runtime
 - Provide better security and reliability
- Just-In-Time Compilation
- Native Image Generator Compilation

©2004 Morgan Kaufmann Publishers 7

Chapter 2

Instructions: Language of the Computer

©2004 Morgan Kaufmann Publishers 8

Instructions:

- Language of the Machine
- More primitive than higher level languages
- Very restrictive
- Easy for machine, harder for human

- We'll be working with the MIPS instruction set architecture
 - "1st" RISC architecture
 - similar to ARM
 - Almost 100 million MIPS processors manufactured in 2002
 - used by NEC, Nintendo, Cisco, Silicon Graphics, Sony, ...

©2004 Morgan Kaufmann Publishers 9

General design principles to live by

- Simplicity favors regularity
 - Being regular make things easier to deal with
- Smaller is faster
 - Smaller area is easier to handle
- Make the common case fast
 - No sense to work on rare cases
- Good design demands good compromises
 - A catchall...

©2004 Morgan Kaufmann Publishers 10

MIPS arithmetic

- All instructions have 3 operands
- Operand order is fixed (destination first)
 - Make it easier for the hardware...
 - Design Principle: simplicity favors regularity
- Example:


```
C code:      a = b + c
MIPS 'code': add $s0, $s1, $s2
```

\$s1 denotes register

©2004 Morgan Kaufmann Publishers 11

MIPS arithmetic

- Of course this complicates some things...

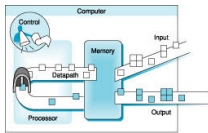

```
C code:      a = b + c + d;
MIPS code:   add $s0, $s1, $s2
              add $s0, $s0, $s3

C code:      a = (b+c) - (d+e);
MIPS code:   add $s0, $s1, $s2
              add $s3, $s4, $s5
              sub $s6, $s0, $s3
```
- Arithmetic operands must be registers
- Registers are all 32 bits
 - Regularity
- Quick Question: Why not 31 bits?

©2004 Morgan Kaufmann Publishers 12

Registers vs. Memory

- Only 32 registers provided
- Design Principle: smaller is faster.
 - Faster decoding, less "capacitance", more specialized...
- Compiler associates variables with registers
- What about programs with lots of variables?

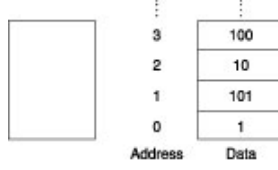


COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 13

Memory Organization

- Viewed as a large, single-dimension array, with an address.
- A memory address is an index into the array
- "Byte addressing" means that the index points to a byte of memory.




COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 14

Memory Organization

- Bytes are nice, but most data items use larger "words"
- For MIPS, a word is 32 bits or 4 bytes.



Registers hold 4 bytes of data

- 2^{32} bytes with byte addresses from 0 to $2^{32}-1$
- 2^{30} words with byte addresses 0, 4, 8, ... $2^{32}-4$
- Words are aligned

Quick question:

- what are the least 2 significant bits of a word address?

COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 15

To and from main memory

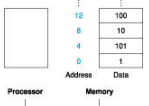
- Load and store instructions
- Example: (A->\$s3, h->\$s2)
 - C code: $A[12] = h + A[8];$
 - MIPS code:


```
lw $t0, 32($s3)
add $t0, $s2, $t0
sw $t0, 48($s3)
```
- Store word has destination last
- Remember arithmetic operands are registers, not memory!

Can't write: `add 48($s3), $s2, 32($s3)`

Quick Question:

- Why 32, 48, instead of 8, 12?



COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 16

Constants

- Add immediate
 - Very frequently occurring
 - increment, decrement, offset,...

```
lw $t0, AddrConstant4($s1)
add $s3, $s3, $t0
```

- Too complicated for such common case, use another instruction:

```
addi $s3, $s3, 4
```

Design principle

Make the common case fast

COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 17

MIPS instruction so far...

MIPS operands		
Name	Example	Comments
32 registers	<code>\$t0, \$s1, ...</code>	Fast locations for data. In MIPS, data must be in registers to perform arithmetic.
2^{32} memory words	<code>Memory[0], ...</code> <code>Memory[4], ...</code> <code>Memory[4294967292]</code>	Accessed only by data transfer instructions in MIPS. MIPS uses byte addresses, so sequential word addresses differ by 4. Memory holds data structures, arrays, and spilled registers.

MIPS assembly language				
Category	Instruction	Example	Meaning	Comments
Arithmetic	add	<code>add \$s1, \$s2, \$s3</code>	$\$s1 = \$s2 + \$s3$	Three operands; data in registers
	subtract	<code>sub \$s1, \$s2, \$s3</code>	$\$s1 = \$s2 - \$s3$	Three operands; data in registers
	add immediate	<code>addi \$s1, \$s2, 100</code>	$\$s1 = \$s2 + 100$	Used to add constants
Data transfer	load word	<code>lw \$s1, 100(\$s2)</code>	$\$s1 = \text{Memory}[\$s2 + 100]$	Data from memory to register
	store word	<code>sw \$s1, 100(\$s2)</code>	$\text{Memory}[\$s2 + 100] = \$s1$	Data from register to memory

COMPUTER SCIENCE & ENGINEERING

©2004 Morgan Kaufmann Publishers 18

Quick questions

- For a given function, which programming language likely takes the most lines of code, C or MIPS assembly language?
- MIPS has only 32 registers, why so few?

Quick questions

- For a given function, which programming language likely takes the most lines of code, C or MIPS assembly language?
- MIPS has only 32 registers why so few?
- (multiple choices)
- Given the importance of registers, what is the rate of increase in the number of registers in a chip over time?
 - Very fast: Moore's law double # transistors every 1.5 years
 - Very slow: program/compiler are optimized for a fixed ISA