Opening the Box

- Capacitive multitouch LCD screen
- 3.8 V, 25 Watt-hour battery
- Computer board
Inside the Processor

- Apple A5
What is Computer Architecture?

- **Computer Architecture** is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.

Diagram:

- Application
- Algorithm
- Programming Language
- Operating System
- **Computer Architecture**
- Circuits
- Devices
- Technology
How Computer Is Made

https://www.youtube.com/watch?v=UvluuAliA50
Relative Performance

- Define: Performance = 1/Execution Time
- “X is $n$ time faster than Y”

\[
\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n
\]

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$
    \[
    = \frac{15\text{s}}{10\text{s}} = 1.5
    \]
  - So A is 1.5 times faster than B
Instruction Count and CPI

CPU Time = Instruction Count × CPI × Clock Cycle Time

- Instruction Count for a program
  - Determined by program, ISA and compiler

- Average cycles per instruction (CPI)
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix
Uniprocessor Performance

Constrained by power, instruction-level parallelism, memory latency

Intel Tick-Tock Model
Embedded processors

- Broadcom XLP-II: 20-core
- Cavium Octeon: 48-core
- Tilera Tile-Gx8072: 72-core

Mobile devices – MPSoCs
- CPU, GPU, DSP, etc.

(GP)GPUs
- Nvidia Kepler: 192x15 cores
- AMD Liverpool: 1152 cores

Challenge: On-chip communication for parallel computing


Intel Xeon Phi: 60-core[5]

[1] http://www.theregister.co.uk/2010/02/03/intel_westmere_ep_preview/
[2] http://www.theregister.co.uk/2012/10/03/ibm_power7_plus_server_launch/
Instruction Set

- The repertoire of instructions of a computer
- Different computers have different instruction sets
  - But with many aspects in common
- Early computers had very simple instruction sets
  - Simplified implementation
- Many modern computers also have simple instruction sets
  - CISC vs. RISC
The MIPS Instruction Set

- Large share of embedded core market
  - Applications in consumer electronics, network/storage equipment, cameras, printers, ...
  - Typical of many modern ISAs (see Appendixes E)
- We will examine two implementations of MIPS ISA
  - A simplified version
  - A more realistic pipelined version
- Simple subset, shows most aspects
  - Memory reference instructions: \texttt{lw, sw}
  - Arithmetic-logical instructions: \texttt{add, sub, and, or, s	exttt{lt}}
  - Control transfer instructions: \texttt{beq, j}
MIPS Instruction Examples

- C code:
  \[ g = h + A[8]; \]
  - G in $s1, h in $s2, base address of A in $s3

- Compiled MIPS code:
  - Index 8 requires offset of 32
  - 4 bytes per word
  \[
  \text{lw} \quad $t0, \; 32($s3) \quad \# \text{ load word}
  \text{add} \quad $s1, \; $s2, \; $t0
  \]
MIPS Instruction Examples

- C code:
  \[ A[12] = h + A[8]; \]
  - \( h \) in \$s2, base address of \( A \) in \$s3

- Compiled MIPS code:
  - Index 8 requires offset of 32
  
  \[
  \begin{align*}
  \text{lw} & \quad \$t0, \ 32($s3) \quad \# \text{ load word} \\
  \text{add} & \quad \$t0, \ \$s2, \ \$t0 \\
  \text{sw} & \quad \$t0, \ 48($s3) \quad \# \text{ store word}
  \end{align*}
  \]
MIPS Instruction Examples

- **Conditional**
  - Branch to a labeled instruction if a condition is true; otherwise, continue sequentially
  - `beq rs, rt, L1`
  - If (rs == rt) branch to instruction labeled L1;

- **Unconditional**
  - `j L1`
  - Unconditional jump to instruction labeled L1
MIPS R-format Instructions

- Instruction fields
  - op: operation code (opcode)
  - rs: first source register number
  - rt: second source register number
  - rd: destination register number
  - shamt: shift amount (00000 for now)
  - funct: function code (extends opcode)
R-format Example

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>

add $t0, $s1, $s2

<table>
<thead>
<tr>
<th>0</th>
<th>17</th>
<th>18</th>
<th>8</th>
<th>0</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>10001</td>
<td>10010</td>
<td>01000</td>
<td>00000</td>
<td>100000</td>
</tr>
</tbody>
</table>

```
000000100011001001000000001000000_2 = 02324020_{16}
```
MIPS I-format Instructions

- Load/store instructions
  - lw $t0, 32($s3)
  - rs/rt: source/destination register number
  - Address: offset added to base address in rs

- Immediate arithmetic
  - addi $s1, $s2, 20
  - rs/rt: source/destination register number
  - Constant: $-2^{15}$ to $+2^{15} - 1$

<table>
<thead>
<tr>
<th></th>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>constant or address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>
MIPS J-format Instructions

- Jump (j) targets could be anywhere in text segment
  - j L1
- Encode full address in instruction

(Pseudo)Direct jump addressing
- Target address = \( PC_{31...28} \times (address \times 4) \)
Branch Addressing

- Branch instructions specify
  - `beq rs, rt, L1`
  - Opcode, two registers, target address

- Most branch targets are near branch
  - Forward or backward

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>constant or address</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

- PC-relative addressing
  - Target address = PC + offset $\times$ 4
  - PC already incremented by 4 by this time
Addressing Mode Summary

1. Immediate addressing
   \[
   \text{op} \quad \text{rs} \quad \text{rt} \quad \text{Immediate}
   \]

2. Register addressing
   \[
   \text{op} \quad \text{rs} \quad \text{rt} \quad \text{rd} \quad \ldots \quad \text{func}
   \]

3. Base addressing
   \[
   \text{op} \quad \text{rs} \quad \text{rt} \quad \text{Address}
   \]

4. PC-relative addressing
   \[
   \text{op} \quad \text{rs} \quad \text{rt} \quad \text{Address}
   \]

5. Pseudodirect addressing
   \[
   \text{op} \quad \text{Address}
   \]

Diagram:

- Immediate addressing
- Register addressing
- Base addressing
- PC-relative addressing
- Pseudodirect addressing