The Story so far:

- Instruction Set Architectures
- Performance issues
- ALU
- Single Cycle CPU
- Multicycle CPU: datapath, control
- Microprogramming
- Exceptions

---

Mid-term: 6pm-8pm; break 15 minutes.
Lecture on "real-world processor architectures"

**Question 1:** 1.25 * 2^16
Viz. 1110011111
**Question 3:** a. \[ @1 \rightarrow \text{Memory}[@0] \text{; $@0 = $@0 + 4} \]
**Question 3:** c. To calculate the number of cycles it takes, assume that $@0$ was "all ones".
**Question 3:** d. \[ @1 \rightarrow \text{Mem}[@0] \]
Also: there should be "breaks" after every case.

**Question 4:** a. Feel free to draw on either the picture in the back or the picture in the question. Just make sure the TA can find it!!!

---

Real-World Architectures?
ARM 7 Datapath

Die-Size => Cheap

ARM7TDMI
32/16 bit RISC architecture (ARM v4T)
32-bit ARM instruction set for best performance and flexibility
16-bit Thumb instruction set for increased code density
Unified bus interface, 32-bit data bus serves both instructions and data
Three-stage pipeline: IF, ID, OF, Ex, MEM
20-bit A11 and high-performance multiplier
Very small die size and low power consumption
Fully static operation
Coprocessor interface

MIPS/Watt => Low Power
StrongARM110 => 185mips/450mW

Instruction Set

ARM7 Datapath

Von Neuman Architecture: Single bus for data and instructions: ARM7
Harvard Architecture: Dual bus: ARM9/10 most modern processors

Instruction Set

Figure 1: POM714 Chip Logical View

Figure 6: Data Processing Instructions

Figure 7: Instruction Set

Figure 8: Coprocessor

Figure 9: Instruction Set

InSTRUCTION SET

ARM7 takes 4CPS (0000) whenever it needs to execute a coprocessor (or undefined) instruction. This will not happen if the instruction field is to be executed because of the condition codes.

Each coprocessor will have a copy of the instruction, and may inspect the CPA field to see which coprocessor it is

Every coprocessor in a system will have a unique number to identify it. If the coprocessor reaches the contents of the CPA field, it will know where the in the CPA is the coprocessor that that number.

If no coprocessor has a number which matches the CPA field, CPA and CPA will remain HIGH, and ARM will take the undefined instruction trap.

Otherwise ARM7 assumes the CPA line going LOW, and waits until the coprocessor is not busy.

Coprocessor "Hard" and "Airen" operations show similarity
Branch Prediction:
- In each cycle, up to eight instructions are fetched from the direct-mapped 64 KB instruction cache.
- The branch prediction logic scans the fetched instructions looking for up to two branches each cycle.
- Depending upon the branch type found, various branch prediction mechanisms engage to help predict the branch direction or the target address of the branch or both.
- Branch direction for unconditional branches are not predicted.
- All conditional branches are predicted, even if the condition register bits upon which they are dependent are known at instruction fetch time.
- Branch target addresses for the PowerPC branch to link register (btl) and branch to counter register (bct) instructions can be predicted using a hardware implemented branch target buffer and branch count cache mechanism, respectively.
- Target addresses for absolute and relative branches are computed directly as part of the branch scan function.

As branch instructions flow through the rest of the pipeline, the branch leads execute in the branch evaluation until the actual outcome of the branch is determined. At that point, if the prediction was found to be incorrect, the branch instructions are simply completed like all other instructions.

In the event that a prediction is found to be incorrect, the instruction fetch logic causes the mispredicted instructions to be discarded and starts refetching instructions along the corrected path.
Very-long instruction words (VLIW) (256 bits wide) => supply up to eight 32-bit instructions:

- The first bit of every 32-bit instruction determines if the next instruction belongs to the same execute packet as the previous instruction, or whether it should be executed in the following clock as a part of the next execute packet.
- Two sets of functional units. Each set contains four units, and a register file:
  - One set contains functional units .L1, .S1, .M1, and .D1;
- The two register files each contain 16 32-bit registers for a total of 32 general-purpose registers.
- All eight functional units (.L1, .S1, .M1, .M2, .S2, and .L2) also execute floating-point instructions.
- Two sets of data-addressing units (.D1 and .D2) are responsible for all data transfers between the register file and the memory.
- The two .L and .S functional units perform a general set of arithmetic, logical, and branch instructions.

The processing flow begins when a 256-bit-wide instruction fetch packet is fetched. Each 32-bit instruction declared for the individual functional units is "linked" together by "1" bits in the least significant bit (LSB) position of the instruction.

Execute packets are dispatched to their respective functional units at the rate of one per clock cycle.

After decoding, the instructions simultaneously drive all active functional units.

---

**The Texas Instruments C6x Series: VLIW Performance**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Peak MIPS</th>
<th>BDTI</th>
<th>ISR latency</th>
<th>Power @ 5V</th>
<th>Unit Price</th>
<th>Area (mm²)</th>
<th>Volume (in³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium MMX 233</td>
<td>466</td>
<td>49</td>
<td>1.14 µs</td>
<td>2.25 W</td>
<td>$213</td>
<td>5.5&quot; x 2.5&quot;</td>
<td>8.789 in</td>
</tr>
<tr>
<td>Pentium MMX 266</td>
<td>632</td>
<td>56</td>
<td>0.99 µs</td>
<td>4.85 W</td>
<td>$348</td>
<td>5.5&quot; x 2.5&quot;</td>
<td>8.789 in</td>
</tr>
<tr>
<td>C62x 150 MHz</td>
<td>1200</td>
<td>74</td>
<td>0.12 µs</td>
<td>1.45 W</td>
<td>$225</td>
<td>1.3&quot; x 1.3&quot;</td>
<td>0.118 in</td>
</tr>
<tr>
<td>C62x 200 MHz</td>
<td>1600</td>
<td>99</td>
<td>0.09 µs</td>
<td>1.94 W</td>
<td>$866</td>
<td>1.3&quot; x 1.3&quot;</td>
<td>0.118 in</td>
</tr>
</tbody>
</table>

---

**Algorithm Kernel Benchmarks**

The BDTI Benchmarks™ are based on DSP algorithm kernels.
- DSP algorithm kernels are the most computationally intensive portions of DSP applications.
- Example algorithm kernels include FFT, IIR filters, and Viterbi decoders.
- Application-relevant algorithm kernels are strong predictors of overall performance.
| ARM/Thumb:       | Small/Simple/Cheap       |
| ARM/Thumb:       | Thumb mode               |
| ARM/Thumb:       | Conditional Execution    |
| ARM/Thumb:       | Co-processor hooks       |
| Power4:          | Supercomputing at its best|
| Power4:          | 200 instructions in flight|
| Power4:          | Detailed SMP hooks       |
| TI C6x:          | Signal Processing        |
| TI C6x:          | Optimized for arithmetic |
| TI C6x:          | VLIW                     |
| Trends:          | Superscalar/Deep pipes?  |
| Trends:          | Out-of-order?            |
| Trends:          | Instruction groups/packets?|
| Trends:          | Need: compiler technology!|