18-447
ARM ISA
Recitation #1

Prof. Onur Mutlu
Carnegie Mellon University
Spring 2014, 1/14/2013
What is Computer Architecture?

- **ISA+implementation definition**: The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

- **Traditional (only ISA) definition**: “The term **architecture** is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.” *Gene Amdahl*, IBM Journal of R&D, April 1964
ISA vs. Microarchitecture

ISA
- Agreed upon interface between software and hardware
  - SW/compiler assumes, HW promises
- What the software writer needs to know to write and debug system/user programs

Microarchitecture
- Specific implementation of an ISA
- Not visible to the software

Microprocessor
- ISA, uarch, circuits
- “Architecture” = ISA + microarchitecture
ISA

- Instructions
  - Opcodes, Addressing Modes, Data Types
  - Instruction Types and Formats
  - Registers, Condition Codes

- Memory
  - Address space, Addressability, Alignment
  - Virtual memory management

- Call, Interrupt/Exception Handling
- Access Control, Priority/Privilege
- I/O: memory-mapped vs. instr.
- Task/thread Management
- Power and Thermal Management
- Multi-threading support, Multiprocessor support
ARM ISA – What is it?

- ARM is a RISC architecture
- Used for mobile computing
- Features
  - Load/Store Architecture
  - Conditional Execution
  - Inline barrel shifter
  - Multiple execution modes involving banked registers
  - Many different addressing modes
  - Thumb mode (16-bit mode)
ARM Basics - Registers

- 16 General Purpose Registers
  - R15 is the PC
  - R14 is the linker addr
  - R13 is the stack pointer
- CPSR and SPSP
## ARM Basics – Instruction Encodings

<table>
<thead>
<tr>
<th>Condition</th>
<th>Opcode</th>
<th>Shift Amount</th>
<th>Shift</th>
<th>Register</th>
<th>Immediate</th>
<th>Rotate</th>
<th>Media Instructions</th>
<th>Architecturally Undefined</th>
<th>Load/Store Multiple</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move immediate to status register</td>
<td>0 0 0 1 0 1 0</td>
<td>0 1 0</td>
<td>Mask</td>
<td>SBO</td>
<td>rotate</td>
<td>immediate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load/store immediate offset</td>
<td>0 0 0 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load/store register offset</td>
<td>0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media instructions [4]</td>
<td>0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load/store multiple</td>
<td>0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Branch and branch with link</td>
<td>0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coprocessor load/store and double register transfers</td>
<td>0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coprocessor data processing</td>
<td>0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coprocessor register transfers</td>
<td>0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software interrupt</td>
<td>0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unconditional instructions:</td>
<td>0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure A3-1 ARM instruction set summary**
ARM ISA - Barrel Shifter

- 4 Different types of shifts: LSL, LSR, ASR, ROR (special case RRX)
- Assembler code:

```
<opcode>{<cond>}{S} <Rd>, <Rn>, <shifter_operand>
<Rm>, LSL #<shift_imm>
<Rm>, LSL <Rs>
```

- Encoding:

32-bit immediate

<table>
<thead>
<tr>
<th>31</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>16</th>
<th>15</th>
<th>12</th>
<th>11</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>cond</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>opcode</td>
<td>S</td>
<td>Rn</td>
<td>Rd</td>
<td>rotate_imm</td>
<td>immed_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Immediate shifts

| 31 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 16 | 15 | 12 | 11 | 7  | 6  | 5  | 4  | 3  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| cond | 0 | 0 | 0 | opcode | S | Rn | Rd | shift_imm | shift | 0 | Rm |

Register shifts

| 31 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 16 | 15 | 12 | 11 | 8  | 7  | 6  | 5  | 4  | 3  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| cond | 0 | 0 | 0 | opcode | S | Rn | Rd | Rs | 0 | shift | 1 | Rm |

ARM ISA - Addressing Mode

- Many different modes (about 9)
  - Only worry about 2 for this class
- Assembler Code:

  ```assembly
  LDR|STR{cond}{B}{T} <Rd>, <addressing_mode>
  [Rn], #/+/-<offset_12>
  [Rn], +/-<Rm>
  [Rn], +/-<Rm>, <shift> #/shift_imm>
  [Rn], #/+/-<offset_12>!!
  [Rn], #/+/-<offset_12>
  ```

- Encoding:

  ![Addressing Mode Encoding Diagram]

More info on page A5-19 of *ARM Architecture Reference Manual*