The ARM Architecture

Ali Saidi
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
- Data Path and Pipelines
- System Design
- Development Tools
ARM Ltd

- Founded in November 1990
  - Spun out of Acorn Computers

- Designs the ARM range of RISC processor cores
- Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers.
  - ARM does not fabricate silicon itself

- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware, application software, graphics, bus architectures, peripherals, cell libraries
The Architecture for the Digital World

ARM designs technology that lies at the heart of advanced digital products
ARM’s Activities

Connected Community
Development Tools
Software IP

Processors
System Level IP:
Data Engines
Fabric
3D Graphics

Physical IP
Product Areas

- Cortex-M0 to Cortex-A9
- Compilers & OS profiling
- 180nm to 28nm
- GPU to HD video
ARM Business Today

- Processor Shipped Last Year: ~4 Billion
- Processor Shipped In Total: >24 Billion
- Processor Licenses: 500+
- Semiconductor Partners: 200+
- Process Technology: 28 – 250 nm
- Connected Community Members: 700+
ARM Business Model Drivers

- Deliver more functionality to the end-user sooner and more cost-effectively
  - Integration
  - Economics
  - Focus
  - Ecosystem
  - Choice
  - Power efficiency
Global Company
**Nokia N95 Multimedia Computer**

**OMAP™ 2420**
Applications Processor
ARM1136™ processor-based
SoC, developed using Magma ®
Blast® family and winner of 2005
INSIGHT Award for ‘Most
Innovative SoC’

**Symbian OS™ v9.2**
Operating System supporting ARM
processor-based mobile devices,
developed using ARM® RealView®
Compilation Tools

**S60™ 3rd Edition**
S60 Platform supporting ARM
processor-based mobile devices

**Mobiclip™ Video Codec**
Software video codec for ARM
processor-based mobile devices

**ST WLAN Solution**
Ultra-low power 802.11b/g WLAN
chip with ARM9™ processor-based
MAC

Connect. Collaborate. Create.
ARM Processor Applications
World’s Smallest ARM Computer?

Wireless Sensor Network
- Cortex-M0 +16KB RAM 65nm UWB Radio antenna
- 10 kB Storage memory ~3fW/bit
- 12μAh Li-ion Battery

Battery
Solar Cells
Processor, SRAM and PMU

Wirelessly networked into large scale sensor arrays

Cortex-M0; 65¢
University of Michigan

The Architecture for the Digital World®
World’s Largest ARM Computer?

4200 ARM powered Neutrino Detectors

70 bore holes 2.5km deep

60 detectors per string starting 1.5km down

1km$^3$ of active telescope

Work supported by the National Science Foundation and University of Wisconsin-Madison
From $1\text{mm}^3$ to $1\text{km}^3$
Agenda

- Introduction to ARM Ltd
  - ARM Architecture/Programmers Model
  - Data Path and Pipelines
  - System Design
  - Development Tools
Architecture Versions

ARMv4
- ARM7TDMI(S)™
- SC100™

ARMv5
- ARM7EJ-S™
- SC200™

ARMv6
- ARM1026EJ-S™
- ARM968E-S™
- ARM926EJ-S™
- ARM946E-S™
- ARM966E-S™

ARMv7-Cortex
- ARM11™ MPCore™
- ARM1176JZ(F)-S™
- ARM1136J(F)-S™
- ARM1156T2(F)-S™

ARMv7-Cortex
- Cortex-A9
- Cortex-A8
- Cortex-R4F
- Cortex-R4
- Cortex™-M3
- SC300™
- Cortex-M1/M0 (v6-M)
Relative Performance*

*Represents attainable speeds in 130, 90 or 65nm processes
ARM Cortex Advanced Processors

Architectural innovation, compatibility across diverse application spectrum

- **ARM Cortex-A family:**
  - Applications processors for feature-rich OS and 3rd party applications

- **ARM Cortex-R family:**
  - Embedded processors for real-time signal processing, control applications

- **ARM Cortex-M family:**
  - Microcontroller-oriented processors for MCU, ASSP, and SoC applications

Unparalleled Applicability
Data Sizes and Instruction Sets

- The ARM is a 32-bit architecture.

- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)

- Most ARM’s implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit/32bit Thumb Instruction Set

- Jazelle cores can also execute Java bytecode
ARM and Thumb Performance

Dhrystone 2.1/sec @ 20MHz

Memory width (zero wait state)
Thumb-2 Instruction Set

- Second generation of the Thumb architecture
  - Blended 16-bit and 32-bit instruction set
  - 25% faster than Thumb
  - 30% smaller than ARM

- Increases performance but maintains code density

- Maximizes cache and tightly coupled memory usage
Processor Modes – A Class

- The ARM has seven basic operating modes:
  - **User**: unprivileged mode under which most tasks run
  - **FIQ**: entered when a high priority (fast) interrupt is raised
  - **IRQ**: entered when a low priority (normal) interrupt is raised
  - **Supervisor**: entered on reset and when a Software Interrupt instruction is executed
  - **Abort**: used to handle memory access violations
  - **Undef**: used to handle undefined instructions
  - **System**: privileged mode using the same registers as user mode
The ARM Register Set

Current Visible Registers

<table>
<thead>
<tr>
<th>Abort Mode</th>
<th>r0</th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
<th>r5</th>
<th>r6</th>
<th>r7</th>
<th>r8</th>
<th>r9</th>
<th>r10</th>
<th>r11</th>
<th>r12</th>
<th>r13 (sp)</th>
<th>r14 (lr)</th>
<th>r15 (pc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undef</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Banked out Registers

<table>
<thead>
<tr>
<th>User</th>
<th>FIQ</th>
<th>IRQ</th>
<th>SVC</th>
<th>Undef</th>
</tr>
</thead>
<tbody>
<tr>
<td>r8</td>
<td>r8</td>
<td>r13 (sp)</td>
<td>r13 (sp)</td>
<td>r13 (sp)</td>
</tr>
<tr>
<td>r9</td>
<td>r9</td>
<td>r10</td>
<td>r11</td>
<td>r12</td>
</tr>
<tr>
<td>r10</td>
<td>r10</td>
<td>r11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>FIQ</th>
<th>IRQ</th>
<th>SVC</th>
<th>Undef</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>spsr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>spsr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spsr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spsr</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Exception Handling

<table>
<thead>
<tr>
<th>Exception</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIQ</td>
<td>0x1C</td>
</tr>
<tr>
<td>IRQ</td>
<td>0x18</td>
</tr>
<tr>
<td>(Reserved)</td>
<td>0x14</td>
</tr>
<tr>
<td>Data Abort</td>
<td>0x10</td>
</tr>
<tr>
<td>Prefetch Abort</td>
<td>0x0C</td>
</tr>
<tr>
<td>Software Interrupt</td>
<td>0x08</td>
</tr>
<tr>
<td>Undefined Instruction</td>
<td>0x04</td>
</tr>
<tr>
<td>Reset</td>
<td>0x00</td>
</tr>
</tbody>
</table>

### When an exception occurs, the ARM:
- Copies CPSR into SPSR_<mode>
- Sets appropriate CPSR bits
  - Change to ARM or Thumb state
  - Change to exception mode
  - Disable interrupts (if appropriate)
- Stores the return address in LR_<mode>
- Sets PC to vector address

### To return, exception handler needs to:
- Restore CPSR from SPSR_<mode>
- Restore PC from LR_<mode>

Vector table can be at 0xFFFF0000 on ARM720T and on ARM9/10 family devices.
Program Status Registers

- **Condition code flags**
  - $N = \text{Negative result from ALU}$
  - $Z = \text{Zero result from ALU}$
  - $C = \text{ALU operation Carried out}$
  - $V = \text{ALU operation Overflowed}$

- **Sticky Overflow flag - Q flag**
  - Architecture v5+ only
  - Indicates if saturation has occurred

- **J bit**
  - Architecture v5+ only
  - $J = 1$: Processor in Jazelle state

- **Interrupt Disable bits.**
  - $I = 1$: Disables the IRQ.
  - $F = 1$: Disables the FIQ.

- **T Bit**
  - Architecture v5+ only
  - $T = 0$: Processor in ARM state
  - $T = 1$: Processor in Thumb state

- **Mode bits**
  - Specify the processor mode
Conditional Execution and Flags

- ARM instructions can be made to execute conditionally by postfixing them with the appropriate condition code field.
  - This improves code density and performance by reducing the number of forward branch instructions.

```
CMP   r3,#0
BEQ   skip
ADD   r0,r1,r2

CMP   r3,#0
ADDNE r0,r1,r2
```

- Why was this developed?
  - When would you want to use it? Always? Any downsides?
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
  - Data Path and Pipelines
- System Design
- Development Tools
The ARM7TDM Core

[Diagram of ARM7TDM Core]

- Address Register
- Register Bank
- Multiplier
- Barrel Shifter
- 32 Bit ALU
- Address Incrementer
- Decode Stage
- Instruction Decoder
- Control Logic
- Instruction Decompression
- Read Data Register
- Write Data Register
- DBE
- D[31:0]

Signals:
- ABE
- A[31:0]
- BIGEND
- MCLK
- nWAIT
- nRW
- MAS[1:0]
- ISYNC
- nIRQ
- nFIQ
- nRESET
- ABORT
- nTRANS
- nMREQ
- SEQ
- LOCK
- nM[4:0]
- nOPC
- nCPI
- CPA
- CPB
Pipeline changes for ARM9TDMI

ARM7TDMI

FETCH
- Instruction Fetch

DECODE
- Thumb→ARM decompress
- ARM decode
- Reg Select

EXECUTE
- Reg Read
- Shift
- ALU
- Reg Write

ARM9TDMI

FETCH
- Instruction Fetch

DECODE
- ARM or Thumb Inst Decode
- Reg Decode
- Reg Read

EXECUTE
- Shift + ALU

MEMORY
- Memory Access

WRITE
- Reg Write

ARM®
The Architecture for the Digital World®
ARM10 vs. ARM11 Pipelines

ARM10

- Branch Prediction
- Instruction Fetch
- ARM or Thumb Instruction Decode
- Reg Read
- Shift + ALU
- Memory Access
- Multiply Add
- Multiply
- Write back
- MAC 1
- MAC 2
- MAC 3
- Data Cache 1
- Data Cache 2
- Reg Write
- Fetch 1
- Fetch 2
- Decode
- Issue
- Address
- Write back

ARM11

- FETCH
- ISSUE
- DECODE
- EXECUTE
- MEMORY
- WRITE

The Architecture for the Digital World®
Full Cortex-A8 Pipeline Diagram

13-Stage Integer Pipeline

10-Stage NEON Pipeline
Agenda

Introduction to ARM Ltd
ARM Architecture/Programmers Model
Data Path and Pipelines
- System Design
Development Tools
Typical SoC w/GPU

- Designed and optimised for AMBA: provides easier integration with ARM cores and fabric IP
- Unified Memory Architecture
Agenda

Introduction to ARM Ltd
ARM Architecture/Programmers Model
Data Path and Pipelines
System Design

Development Tools
**ARM Debug Architecture**

- **EmbeddedICE Logic**
  - Provides breakpoints and processor/system access
- **JTAG interface (ICE)**
  - Converts debugger commands to JTAG signals
- **Embedded trace Macrocell (ETM)**
  - Compresses real-time instruction and data access trace
  - Contains ICE features (trigger & filter logic)
- **Trace port analyzer (TPA)**
  - Captures trace in a deep buffer
Keil Development Tools for ARM

- Includes ARM macro assembler, compilers (ARM RealView C/C++ Compiler, Keil CARM Compiler, or GNU compiler), ARM linker, Keil uVision Debugger and Keil uVision IDE

- Keil uVision Debugger accurately simulates on-chip peripherals (I2C, CAN, UART, SPI, Interrupts, I/O Ports, A/D and D/A converters, PWM, etc.)

- Evaluation Limitations
  - 16K byte object code + 16K data limitation
  - Some linker restrictions such as base addresses for code/constants
  - GNU tools provided are not restricted in any way

- http://www.keil.com/demo/
Keil Development Tools for ARM
University Resources

- http://www.arm.com/support/university/
- University@arm.com