The ARM Architecture
Agenda

- Introduction to ARM Ltd
  - ARM Architecture/Programmers Model
  - Data Path and Pipelines
  - AMBA
  - 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, bus architectures, peripherals etc
ARM’s Activities

Connected Community
Development Tools
Software IP

Processors
System Level IP:
Data Engines
Fabric
3D Graphics
Physical IP
ARM Connected Community – 550+
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.
Applications
Agenda

Introduction to ARM Ltd

- ARM Architecture/Programmers Model
- Data Path and Pipelines
- AMBA
- Development Tools
Architecture Versions

- ARMv4
  - ARM7TDMI(S)
  - SC100
- ARMv5
  - ARM7EJ-S
  - ARM968E-S
  - ARM946E-S
  - ARM966E-S
  - ARM1026EJ-S
- ARMv6
  - ARM11
    - ARM1176JZ(F)-S
    - ARM1136J(F)-S
- ARMv7-Cortex
  - x1-4
    - Cortex-A9
    - Cortex-A8
    - Cortex-M3
    - Cortex-R4F
    - Cortex-R4
    - Cortex-M1
    - SC300
    - SC200
Relative Performance*

*Represents attainable speeds in 130, 90 or 65nm processes
Cortex family

**Cortex-A8**
- Architecture v7A
- MMU
- AXI
- VFP & NEON support

**Cortex-R4**
- Architecture v7R
- MPU (optional)
- AXI
- Dual Issue

**Cortex-M3**
- Architecture v7M
- MPU (optional)
- AHB Lite & APB

---

The Architecture for the Digital World®

ARM®
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 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

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

The Architecture for the Digital World®
Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM 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>

This can only be done in ARM state.
### Program Status Registers

- **Condition code flags**
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation overflowed

- **Sticky Overflow flag - Q flag**
  - Architecture 5TE/J only
  - Indicates if saturation has occurred

- **J bit**
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state

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

- **T Bit**
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state

- **Mode bits**
  - Specify the processor mode

---

![Register Diagram](image)
Cortex-M3 Programmer’s Model

- Fully programmable in C
- Stack-based exception model
- Only two processor modes
  - Thread Mode for User tasks
  - Handler Mode for OS tasks and exceptions
- Vector table contains addresses
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
```

- By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using “S”. CMP does not need “S”.

```
SUBS r1,r1,#1
BNE loop
```

**Conditional Execution and Flags**

- **CMP** r3,#0
- **BEQ** skip
- **ADD** r0,r1,r2
- **ADDNE** r0,r1,r2
Classes of Instructions (v4T)

- Load/Store
- Miscellaneous
- Data Operations
- Change of Flow
  - MOV PC, Rm
  - Bcc
  - BL
  - BLX
Branch instructions

- Branch: \( B\{<\text{cond}>\} \text{ label} \)
- Branch with Link: \( BL\{<\text{cond}>\} \text{ subroutine\_label} \)

The processor core shifts the offset field left by 2 positions, sign-extends it and adds it to the PC

- \( \pm 32 \text{ Mbyte range} \)
- How to perform longer branches?
Data processing Instructions

Consist of:
- Arithmetic: ADD, ADC, SUB, SBC, RSB, RSC
- Logical: AND, ORR, EOR, BIC
- Comparisons: CMP, CMN, TST, TEQ
- Data movement: MOV, MVN

These instructions only work on registers, NOT memory.

Syntax:

<Operation>{<cond>}{S} Rd, Rn, Operand2

- Comparisons set flags only - they do not specify Rd
- Data movement does not specify Rn
- Second operand is sent to the ALU via barrel shifter.
Using a Barrel Shifter: The 2nd Operand

Register, optionally with shift operation

- Shift value can be either be:
  - 5 bit unsigned integer
  - Specified in bottom byte of another register.
- Used for multiplication by constant

Immediate value

- 8 bit number, with a range of 0-255.
  - Rotated right through even number of positions
- Allows increased range of 32-bit constants to be loaded directly into registers
## Single register data transfer

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDR</td>
<td>Word</td>
</tr>
<tr>
<td>STR</td>
<td>Word</td>
</tr>
<tr>
<td>LDRB</td>
<td>Byte</td>
</tr>
<tr>
<td>STRB</td>
<td>Byte</td>
</tr>
<tr>
<td>LDRH</td>
<td>Halfword</td>
</tr>
<tr>
<td>STRH</td>
<td>Halfword</td>
</tr>
<tr>
<td>LDRSB</td>
<td>Signed byte load</td>
</tr>
<tr>
<td>LDRSH</td>
<td>Signed halfword load</td>
</tr>
</tbody>
</table>

- Memory system must support all access sizes

- Syntax:
  - \texttt{LDR}\{<\text{cond}>\}\{<\text{size}>\} \texttt{Rd, <address>}
  - \texttt{STR}\{<\text{cond}>\}\{<\text{size}>\} \texttt{Rd, <address>}

  e.g. \texttt{LDREQB}
Agenda

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

---

The Architecture for the Digital World®

ARM
ARM10 vs. ARM11 Pipelines

ARM10

- Branch Prediction
- Instruction Fetch
- ARM or Thumb Instruction Decode
- Reg Read
- Shift + ALU
- Memory Access
- Reg Write

FETCH | ISSUE | DECODE | EXECUTE | MEMORY | WRITE

ARM11

- Fetch 1
- Fetch 2
- Decode
- Issue
- MAC 1
- MAC 2
- MAC 3
- Write back
- Address
- Data Cache 1
- Data Cache 2

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

13-Stage Integer Pipeline

10-Stage NEON Pipeline

Instruction Fetch

Instruction Decode

Instruction Execute and LoadStore

NEON

NEON register writeback

Integer ALU pipe

Integer MUL pipe

Integer shift pipe

Non-IEEE FP ADD pipe

Non-IEEE FP MUL pipe

IEEE FP engine

LS permute pipe

NEON store data

Embedded Trace Macrocell

T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13

External trace port

L1 instruction cache miss

L1 data cache miss

L2 data

L1 data

L1 instruction cache miss

L1 data cache miss

L2 data

L3 memory system

The Architecture for the Digital World®

ARM®
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
- Data Path and Pipelines
  - AMBA
- Development Tools
AHB Structure

[Diagram of AHB Structure]

- Arbiter
- Decoder
- Masters: #1, #2, #3
- Slaves: #1, #2, #3, #4
- Address/Control
- Write Data
- Read Data
- HADDR
- HWDATA
- HRDATA

-- ARM The Architecture for the Digital World®
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
- Data Path and Pipelines
- AMBA
  - 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/community/university/
- University@arm.com
Targeting community development

- $149
- Personally affordable
- Wikis, blogs, promotion of community activity
- Freedom to innovate
- Instant access to >10 million lines of code
- Free software

> 1000 participants and growing

Active & technical community
Open access to hardware documentation
Opportunity to tinker and learn

Opportunity to tinker and learn
Fast, low power, flexible expansion

OMAP3530 Processor
- 600MHz Cortex-A8
- NEON+VFPv3
- 16KB/16KB L1$
- 256KB L2$
- 430MHz C64x+ DSP
- 32K/32K L1$
- 48K L1D
- 32K L2
- PowerVR SGX GPU
- 64K on-chip RAM

POP Memory
- 128MB LPDDR RAM
- 256MB NAND flash

Peripheral I/O
- DVI-D video out
- SD/MMC+
- S-Video out
- USB 2.0 HS OTG
- I²C, I²S, SPI, MMC/SD
- JTAG
- Stereo in/out
- Alternate power
- RS-232 serial

USB Powered
- 2W maximum consumption
- OMAP is small % of that
- Many adapter options
  - Car, wall, battery, solar, ...

3”
Peripheral I/O
- DVI-D video out
- SD/MMC+
- S-Video out
- USB HS OTG
- I²C, I²S, SPI, MMC/SD
- JTAG
- Stereo in/out
- Alternate power
- RS-232 serial

Other Features
- 4 LEDs
  - USR0
  - USR1
  - PMU_STAT
  - PWR
- 2 buttons
  - USER
  - RESET
- 4 boot sources
  - SD/MMC
  - NAND flash
  - USB
  - Serial

On-going collaboration at BeagleBoard.org
- Live chat via IRC for 24/7 community support
- Links to software projects to download
Project Ideas Using Beagle

- **OS Projects**
  - OS porting to ARM/Cortex (TI OMAP), such as open source FreeBSD
  - MythTV system
  - “Super-Beagle” – stack of Beagles as compute engine and task distribution

- **NEON Optimization Projects**
  - Codec optimization in ffmpeg (pick your favorite codec)
  - Voice and image recognition
  - Open-source Flash player optimizations (swfdec)
Fin