

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 Partnership Model



Agenda

Introduction to ARM Ltd

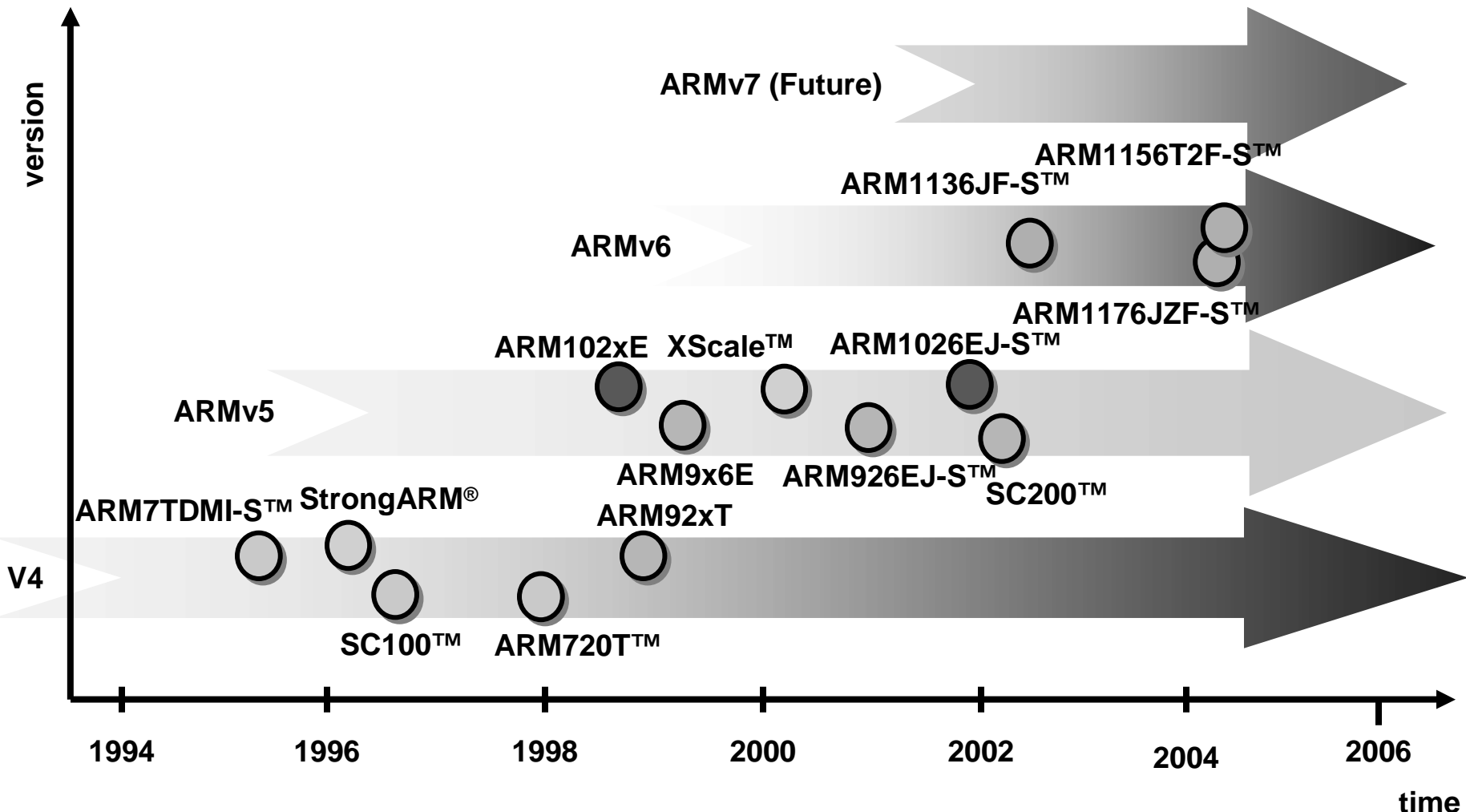
- **ARM Architecture/Programmers Model**

Data Path and Pipelines

AMBA

Development Tools

Architecture Revisions



XScale is a trademark of Intel Corporation

Data Sizes and Instruction Sets

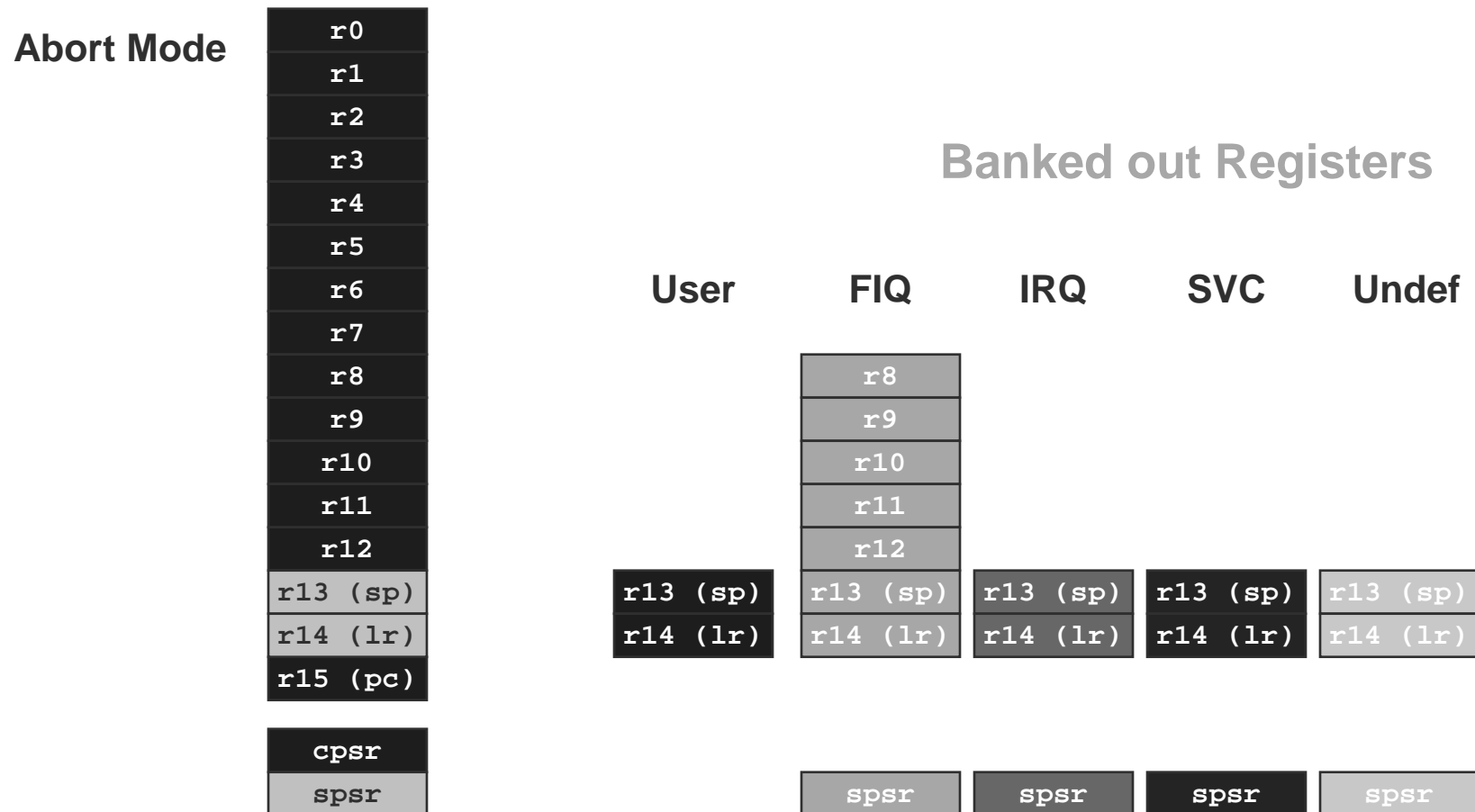
- The ARM is a 32-bit architecture.
- When used in relation to the ARM:
 - **Byte** means 8 bits
 - **Halfword** means 16 bits (two bytes)
 - **Word** means 32 bits (four bytes)
- Most ARM's implement two instruction sets
 - 32-bit ARM Instruction Set
 - 16-bit Thumb Instruction Set
- Jazelle cores can also execute Java bytecode

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

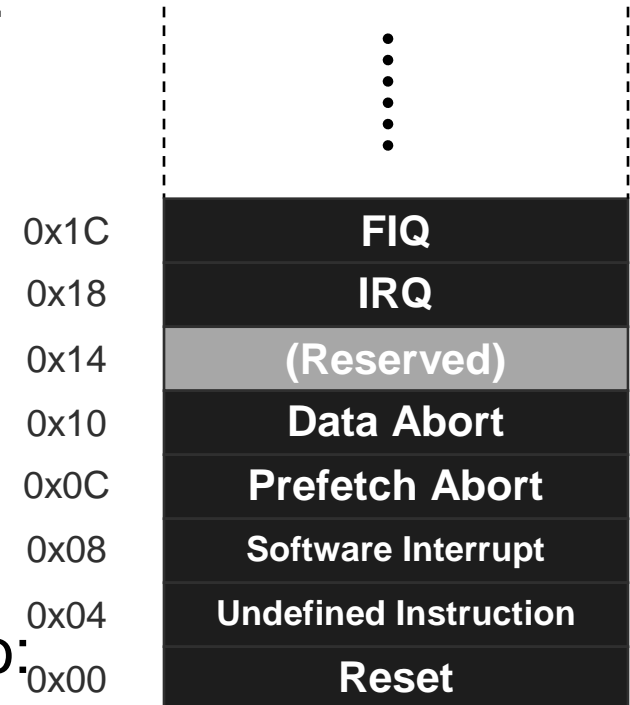
Current Visible Registers



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.



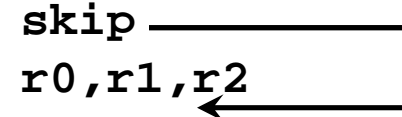
Vector Table

Vector table can be at
0xFFFF0000 on ARM720T
and on ARM9/10 family devices

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
skip
```

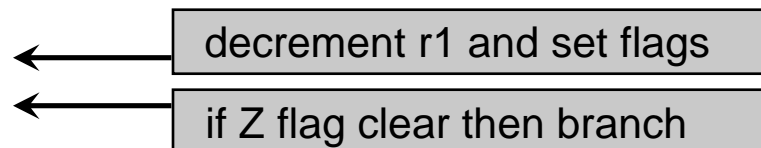


```
CMP    r3,#0
ADDNE  r0,r1,r2
```

- 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”.

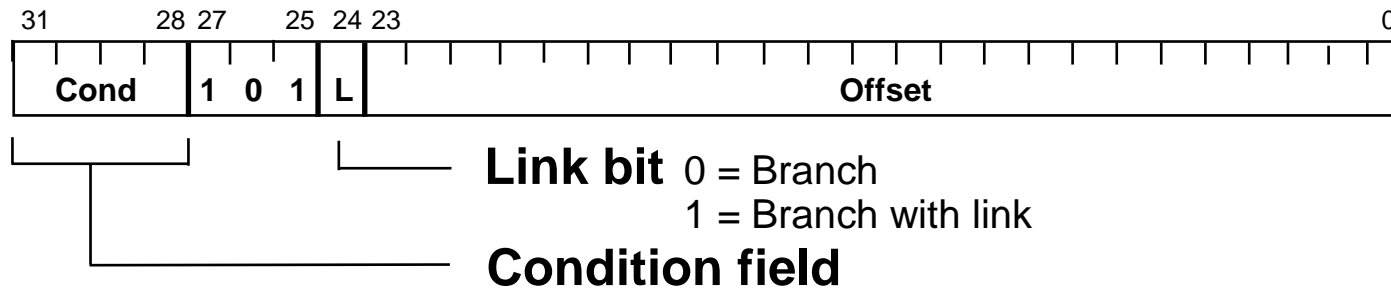
loop

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



Branch instructions

- Branch : `B{<cond>} label`
- Branch with Link : `BL{<cond>} subroutine_label`



- The processor core shifts the offset field left by 2 positions, sign-extends it and adds it to the PC
 - ± 32 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.

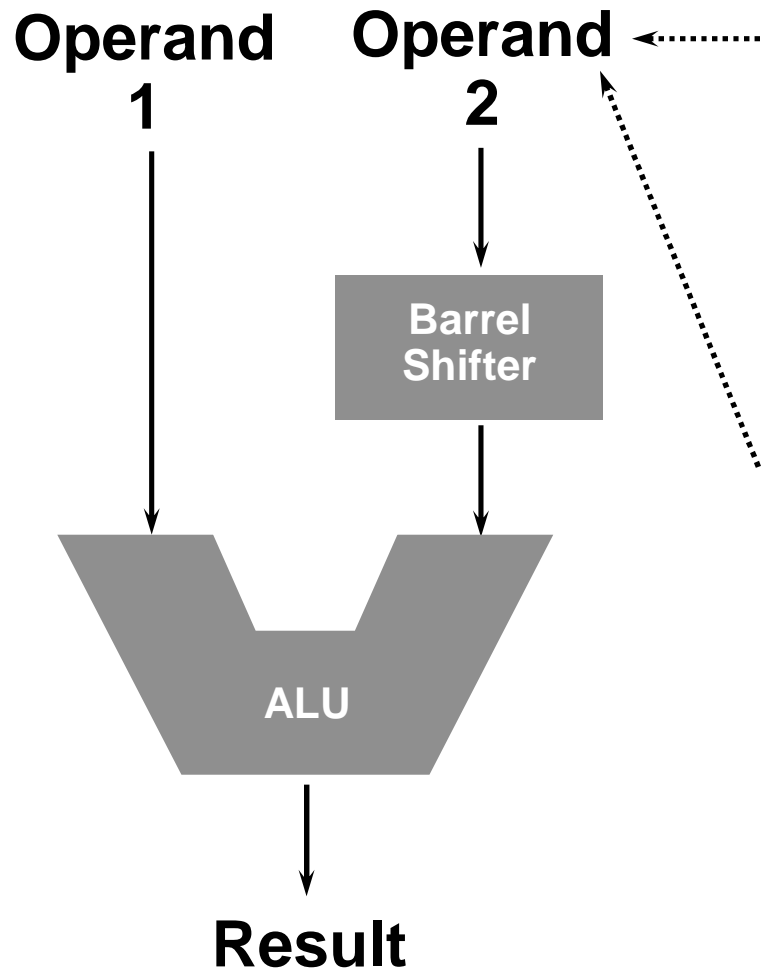
- Syntax:

<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

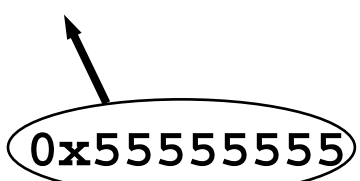
- 8 bit number, with a range of 0-255.
 - Rotated right through even number of positions
- Allows increased range of 32-bit constants to be loaded directly into registers

Loading 32 bit constants

- To allow larger constants to be loaded, the assembler offers a pseudo-instruction:
 - `LDR rd, =const`
- This will either:
 - Produce a `MOV` or `MVN` instruction to generate the value (if possible).or
 - Generate a `LDR` instruction with a PC-relative address to read the constant from a *literal pool* (Constant data area embedded in the code).

- For example

■ <code>LDR r0, =0xFF</code>	=>	<code>MOV r0, #0xFF</code>
■ <code>LDR r0, =0x55555555</code>	=>	<code>LDR r0, [PC, #Imm12]</code>
		...
		...
		<code>DCD 0x55555555</code>



- This is the recommended way of loading constants into a register

Single register data transfer

LDR STR Word

LDRB STRB Byte

LDRH STRH Halfword

LDRSB Signed byte load

LDRSH Signed halfword load

- Memory system must support all access sizes
- Syntax:
 - LDR{<cond>}{<size>} Rd, <address>
 - STR{<cond>}{<size>} Rd, <address>

e.g. LDREQB

Agenda

Introduction to ARM Ltd

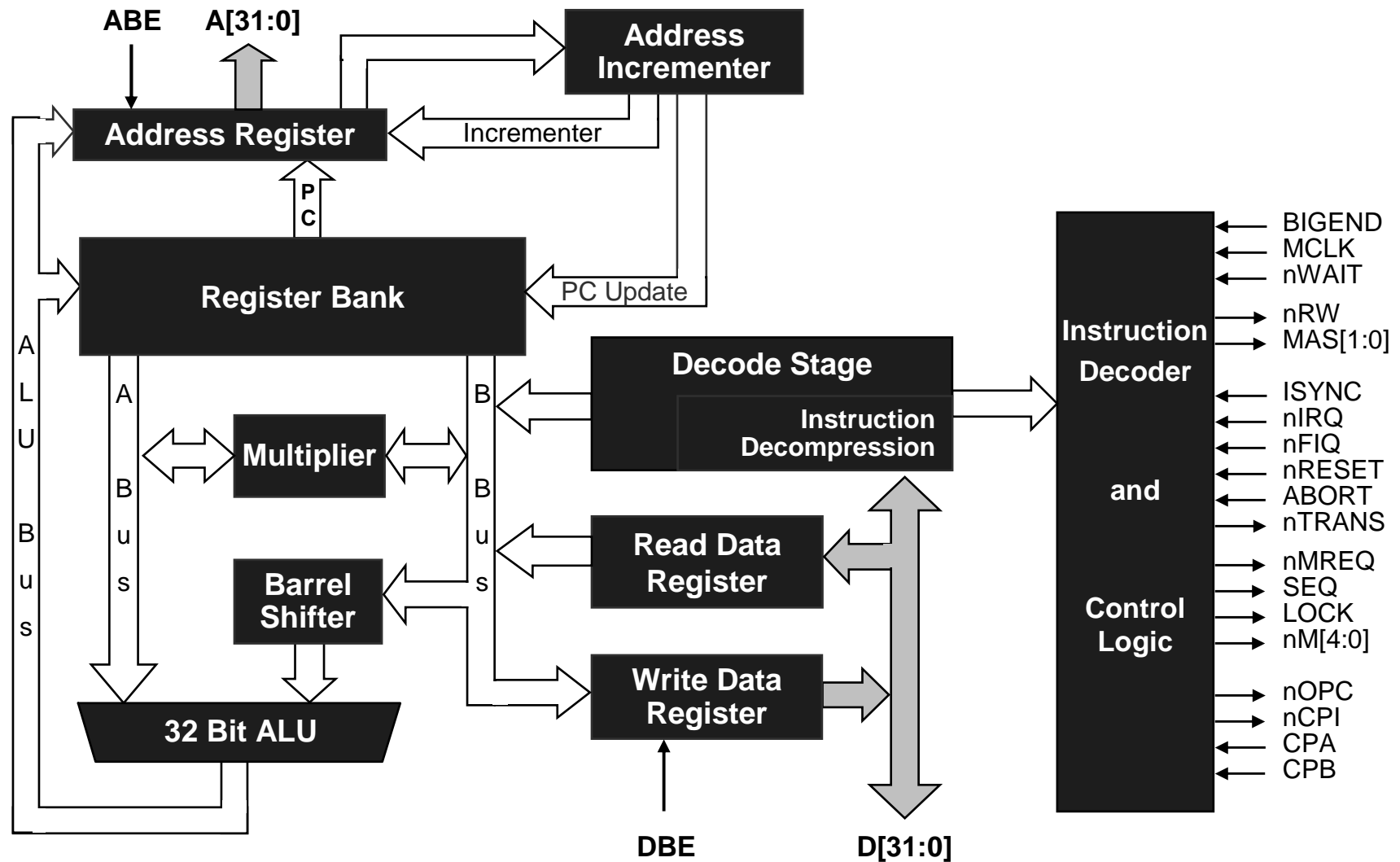
ARM Architecture/Programmers Model

- **Data Path and Pipelines**

AMBA

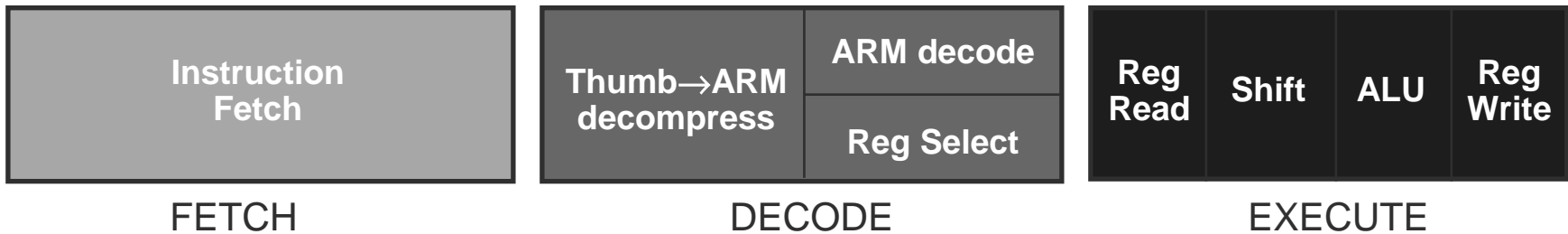
Development Tools

The ARM7TDM Core

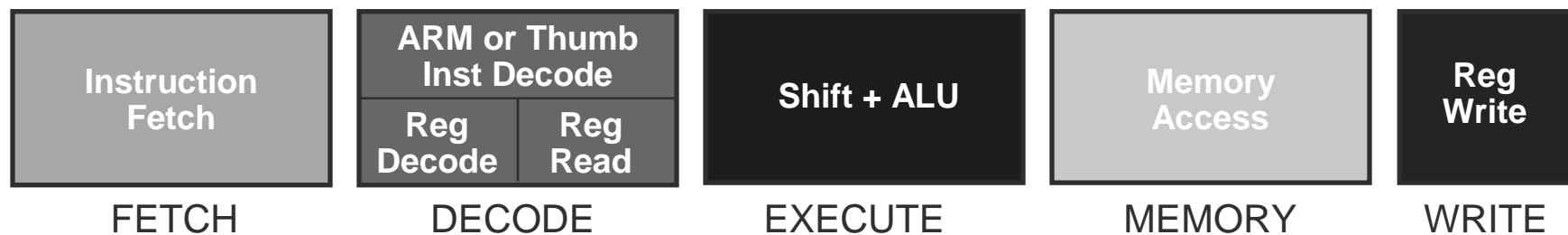


Pipeline changes for ARM9TDMI

ARM7TDMI

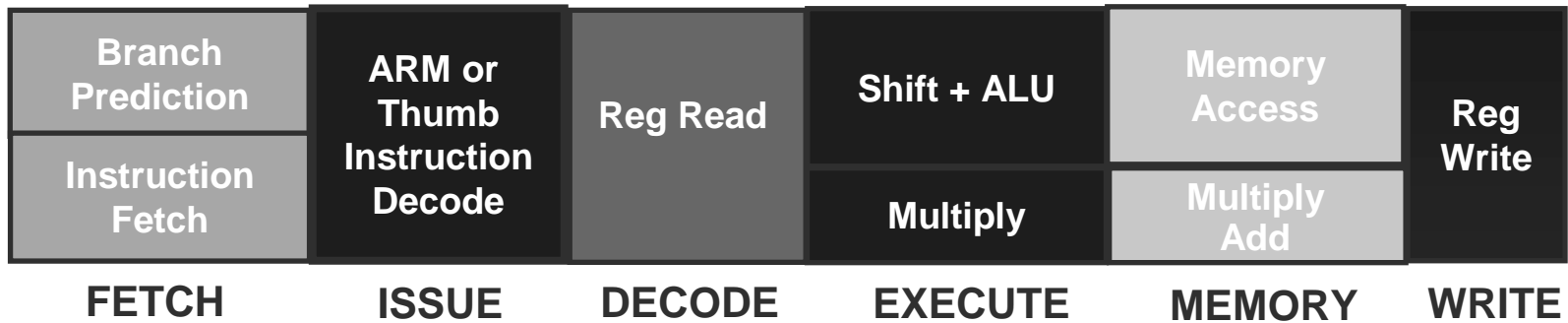


ARM9TDMI

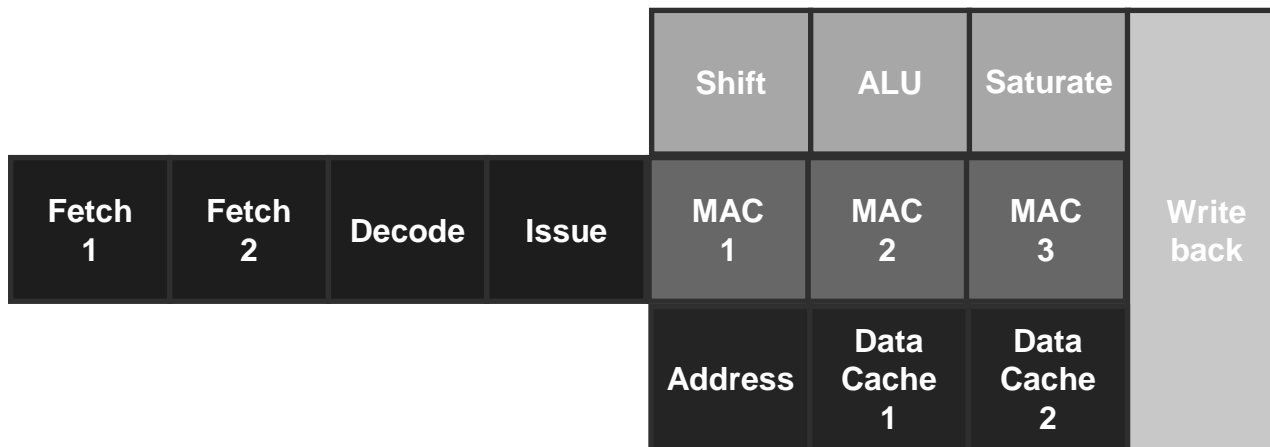


ARM10 vs. ARM11 Pipelines

ARM10



ARM11



Agenda

Introduction to ARM Ltd

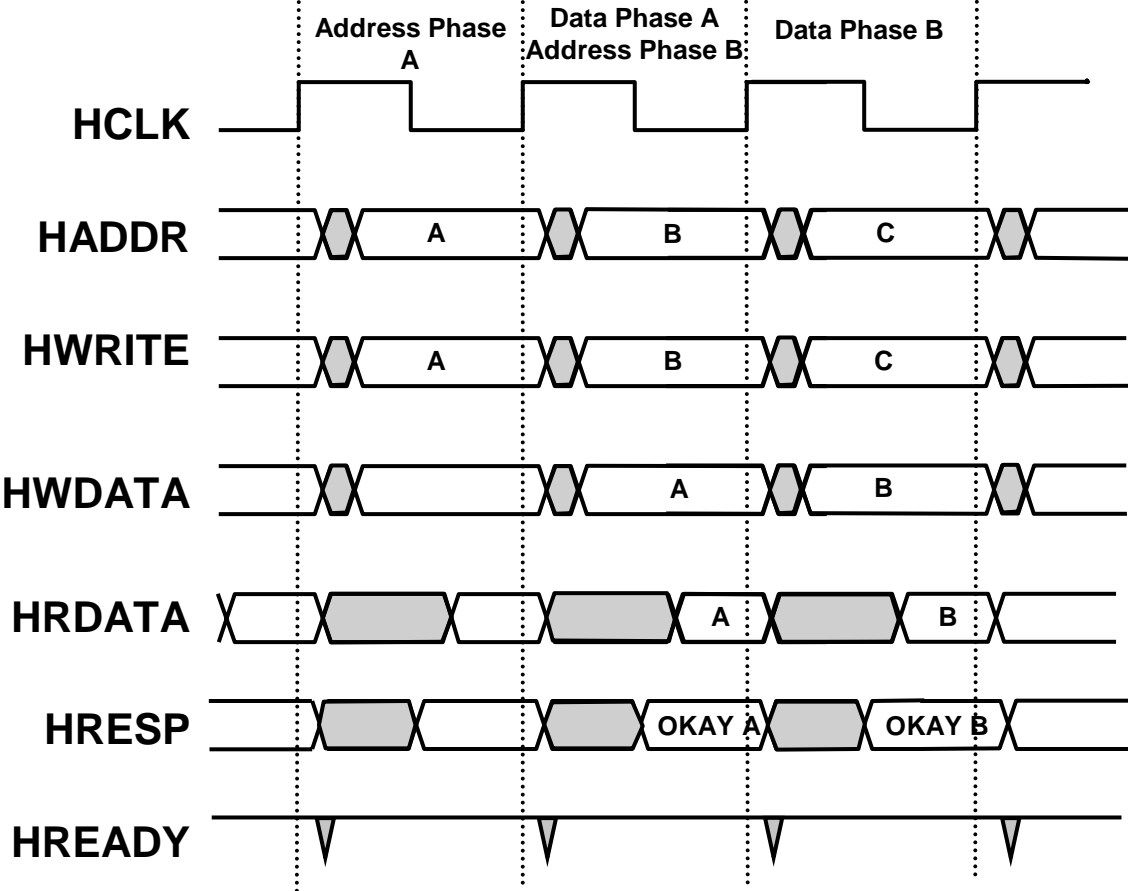
ARM Architecture/Programmers Model

Data Path and Pipelines

- **AMBA**

Development Tools

AHB basic signal timing



Agenda

Introduction to ARM Ltd

ARM Architecture/Programmers Model

Data Path and Pipelines

AMBA

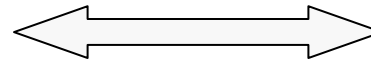
- **Development Tools**

ARM Debug Architecture

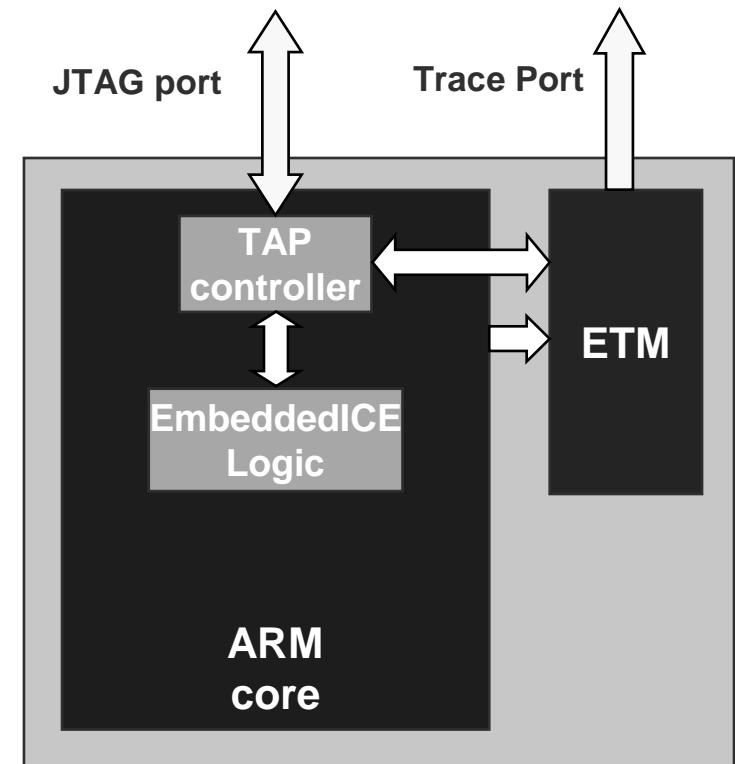
Debugger (+ optional trace tools)



Ethernet



- 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 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

The screenshot displays the Keil uVision3 IDE with the following components:

- Project Workspace:** Shows the current register values for R0 through R10.
- Register Window:**

Register	Value
R0	0x0000000c
R1	0x0000000c
R2	0x00000020
R3	0x0000018d
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
- Symbols Window:** Lists symbols for the Simulator (VTREG) and Peripheral SFRs (ALDOM, ALDOW, ALDOY, ALHOUR, ALMIN, ALMON, ALSEC).
- Source Code Editor:** Displays the C code for `Hello.c`, including headers, comments, and the `main` function which prints "Hello World".
- Output Window:** Shows a warning: "MISSING DEVICE (R003: SECURITY KEY NOT FOUND) Running in Eval Mode" and code size information: "Restricted Version with 16384 Byte Code Size Limit, Currently used: 1980 Bytes (12%)".
- Memory Window:** Shows memory dump for address 0x4000, displaying hexadecimal values for addresses 0x00004000 through 0x0000405B.
- Status Bar:** Shows "Ready", "Simulation", and timing information: "t1: 0.72642057 sec L:29 C:1".

ARM[®]

THE ARCHITECTURE
FOR THE DIGITAL WORLD™