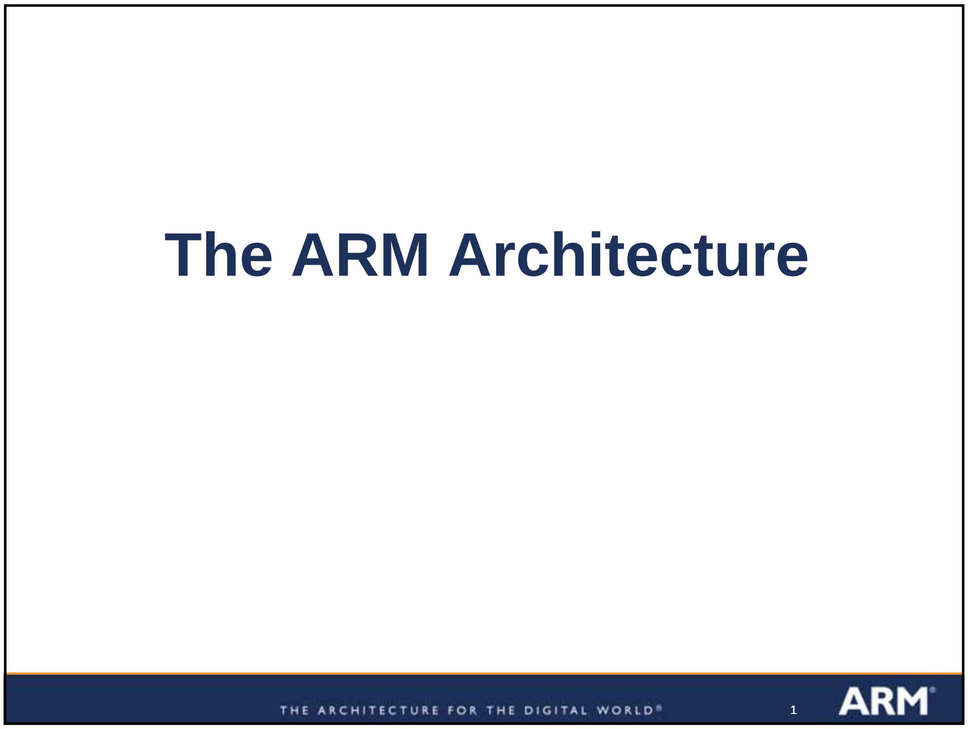
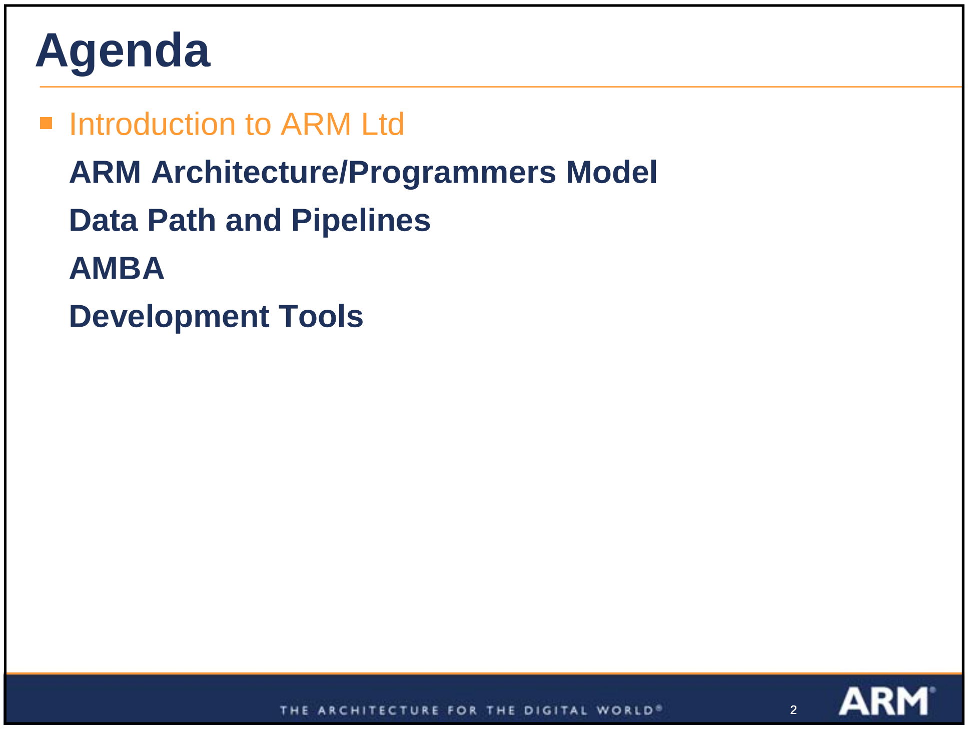


The ARM Architecture



Agenda

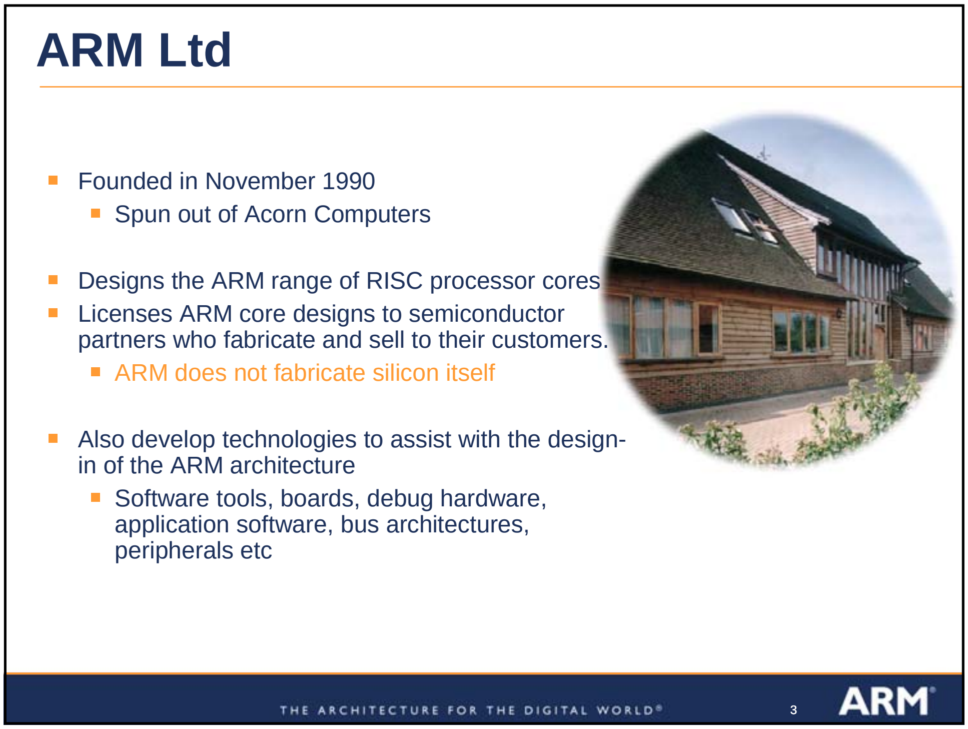
- Introduction to ARM Ltd

  ARM Architecture/Programmers Model

  Data Path and Pipelines

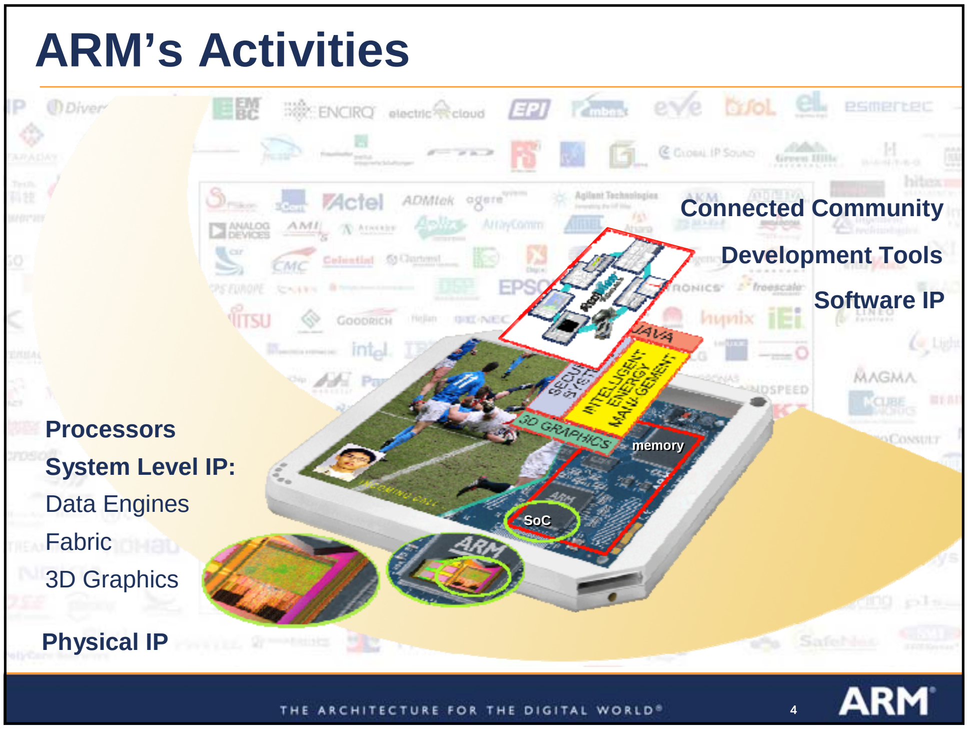
  AMBA

  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

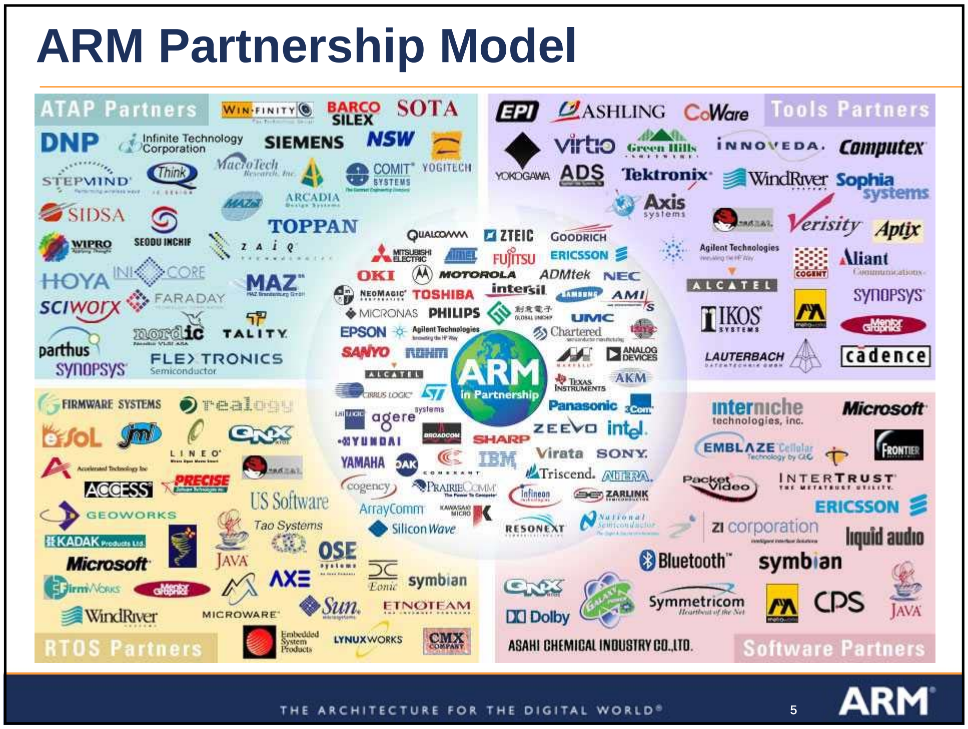
System Level IP:

Data Engines

Fabric

3D Graphics

Physical IP



ARM Partnership Model

ARM Powered Products

How many ARM processors?

Agenda

Introduction to ARM Ltd

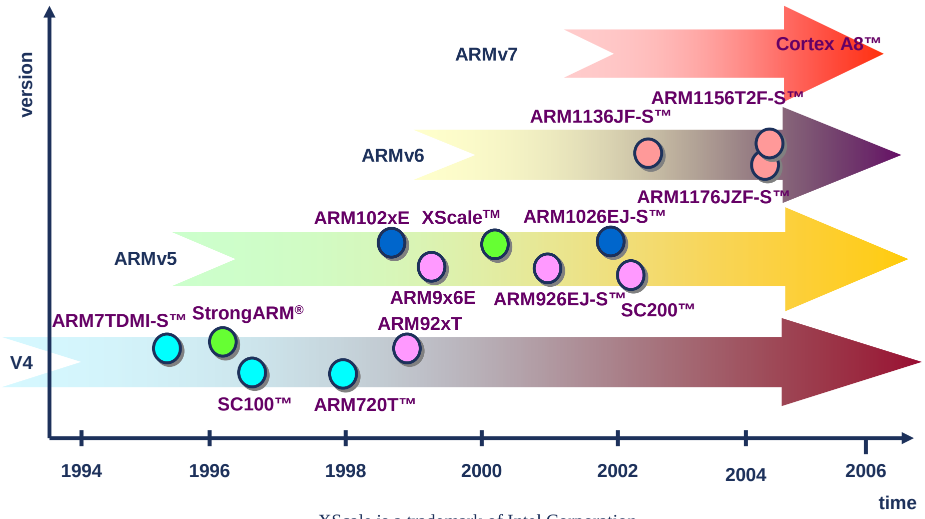
- ARM Architecture/Programmers Model

Data Path and Pipelines

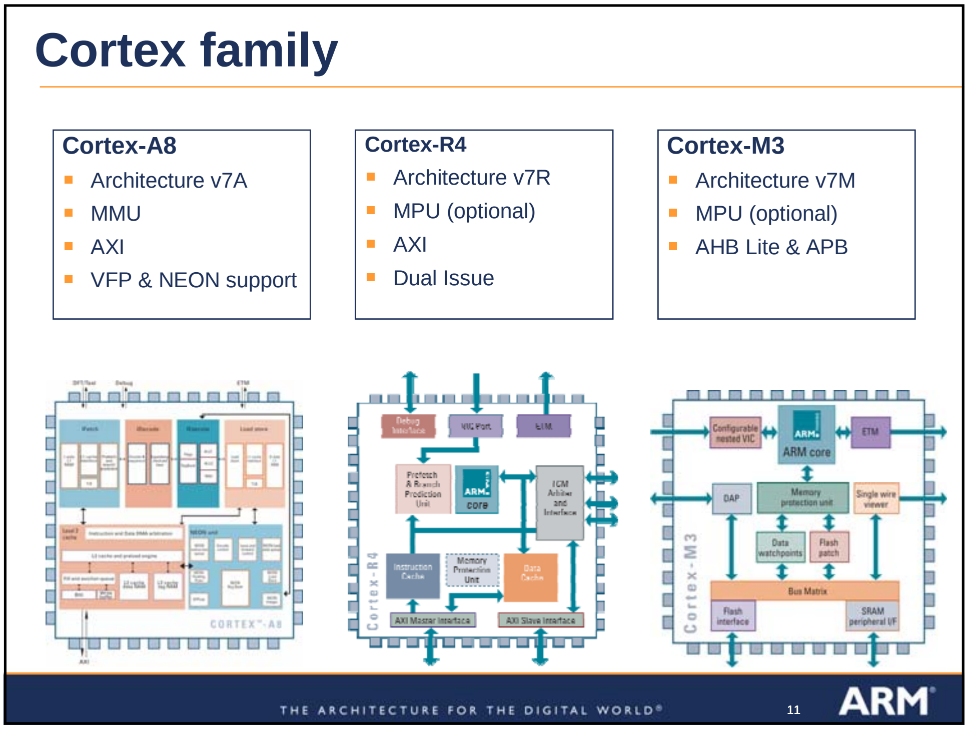
AMBA

Development Tools

Architecture Revisions



XScale is a trademark of Intel Corporation



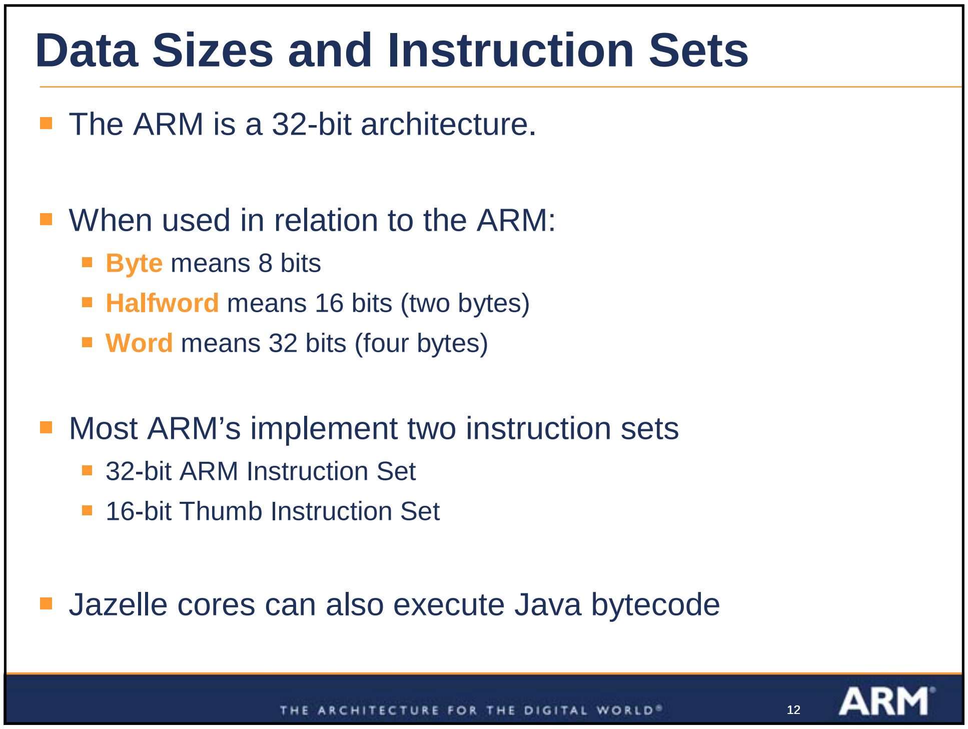
Cortex family

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

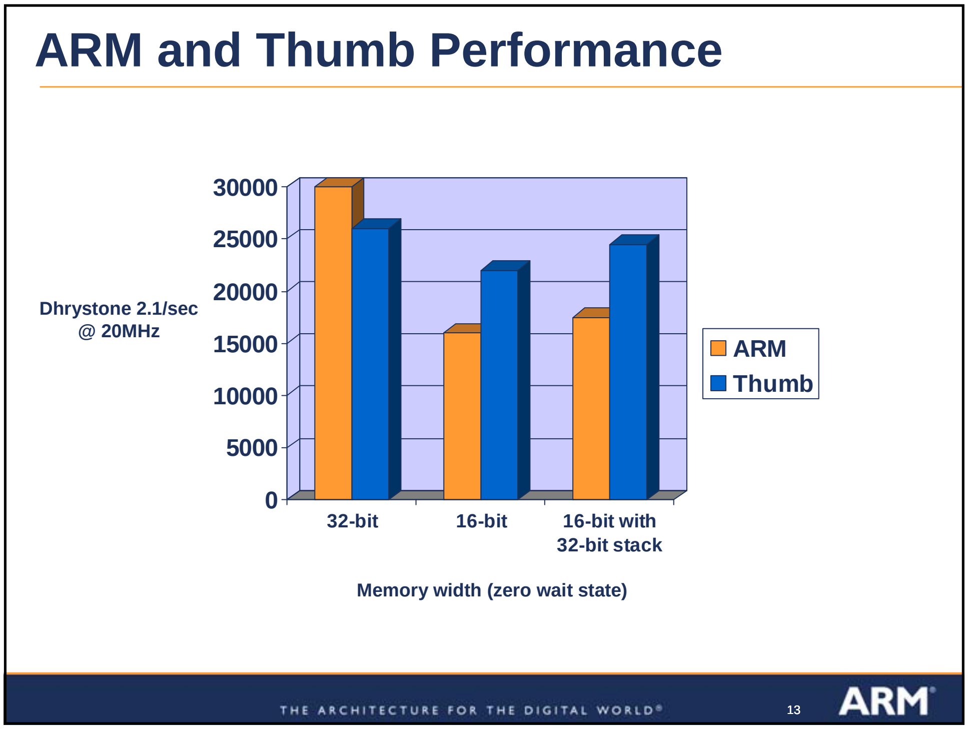
THE ARCHITECTURE FOR THE DIGITAL WORLD® 11 ARM®



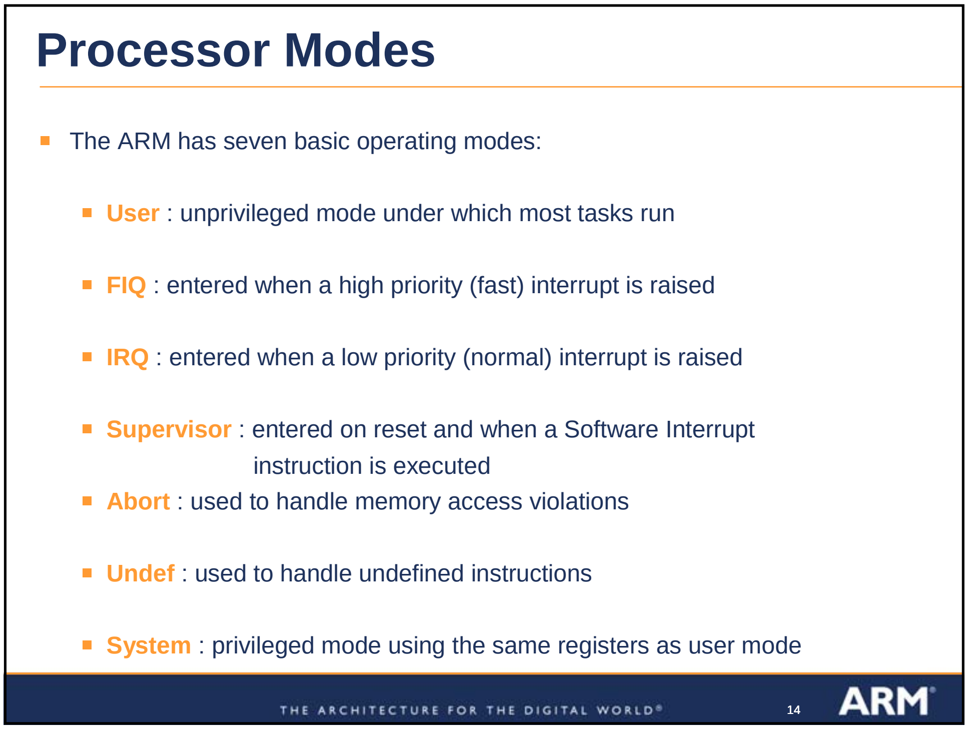
Data Sizes and Instruction Sets

- 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

THE ARCHITECTURE FOR THE DIGITAL WORLD® 12 ARM®

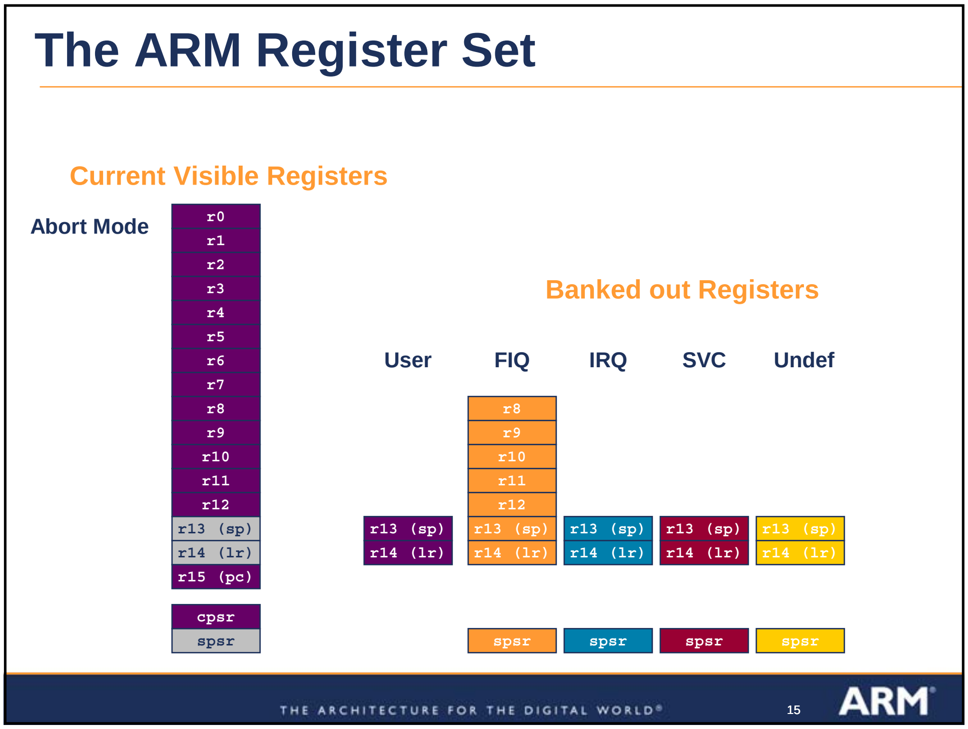


ARM and Thumb Performance



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

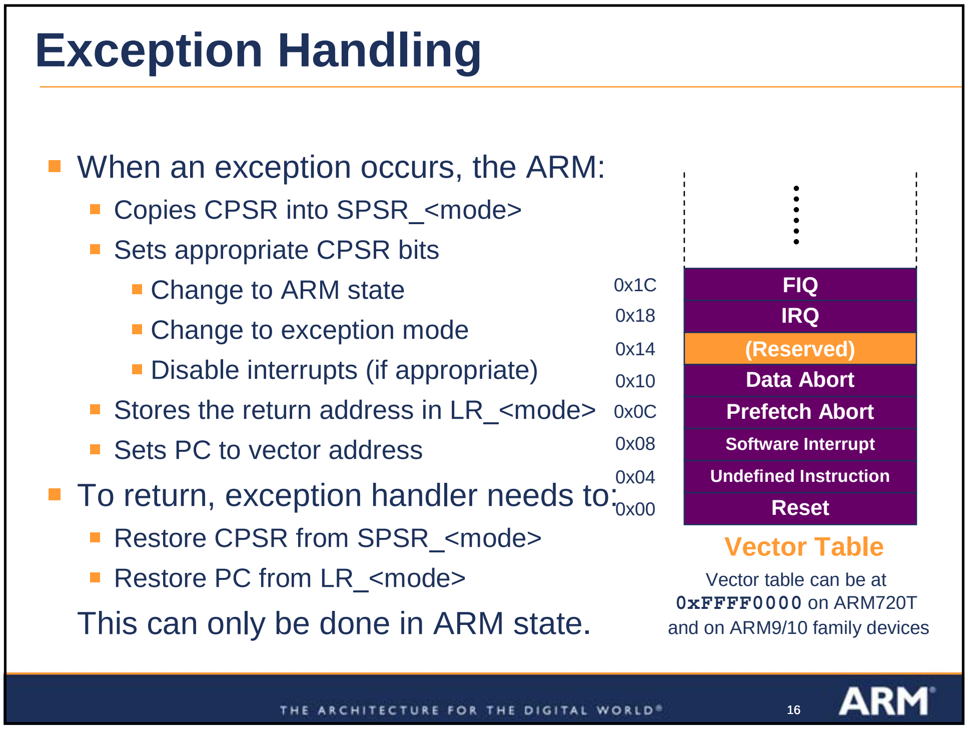


The ARM Register Set

Current Visible Registers

Banked out Registers

Abort Mode	User	FIQ	IRQ	SVC	Undef
r0					
r1					
r2					
r3					
r4					
r5					
r6					
r7					
r8		r8			
r9		r9			
r10		r10			
r11		r11			
r12		r12			
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
r15 (pc)					
cpsr					
spsr		spsr	spsr	spsr	spsr



Exception Handling

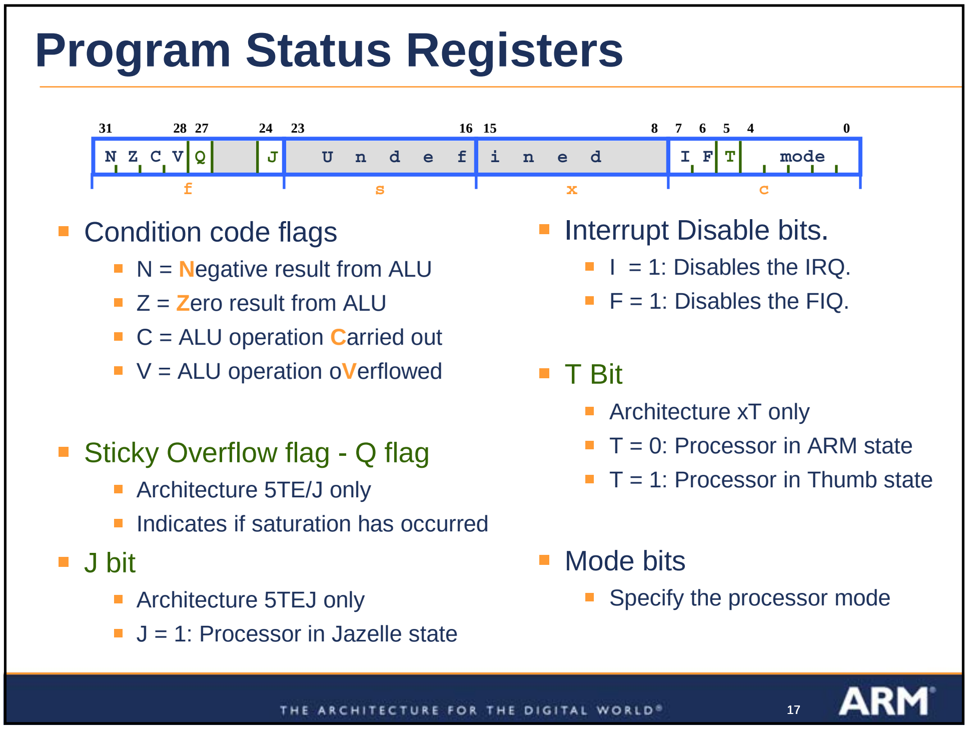
- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler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

This can only be done in ARM state.

0x1C	FIQ
0x18	IRQ
0x14	(Reserved)
0x10	Data Abort
0x0C	Prefetch Abort
0x08	Software Interrupt
0x04	Undefined Instruction
0x00	Reset

Vector Table

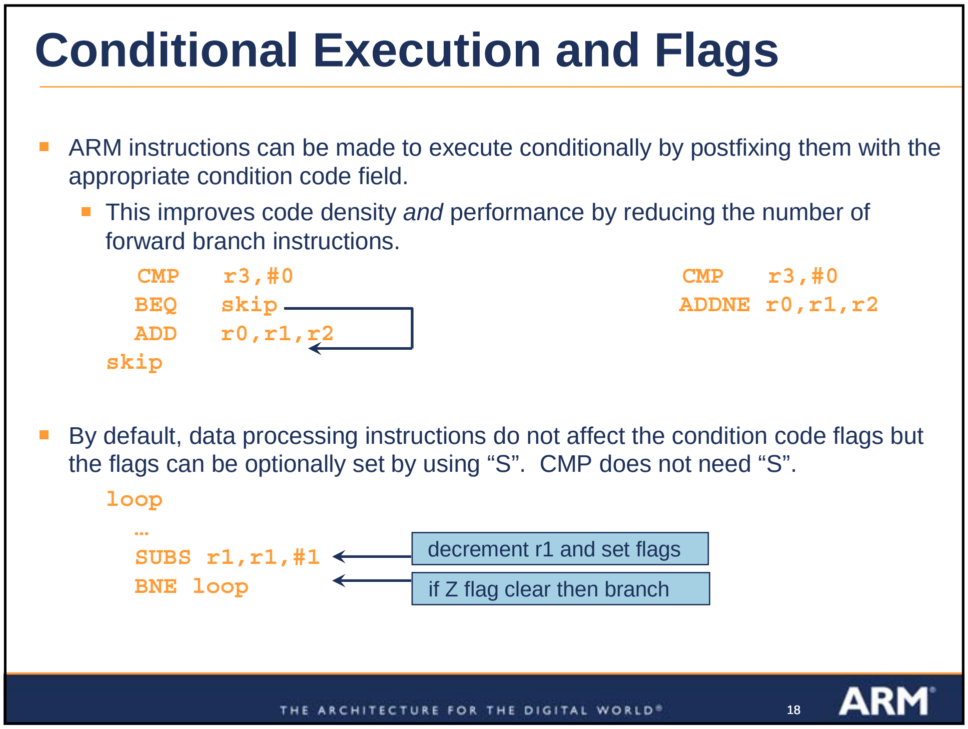
Vector table can be at `0xFFFF0000` on ARM720T and on ARM9/10 family devices



Program Status Registers

31	30	29	28	27	26	25	24	23 16	15 8	7	6	5	4 0
N	Z	C	V	Q			J	U n d e f	i n e d	I	F	T	mode
f								s	x	c			

- Condition code flags
  - N = **N**egative result from ALU
  - Z = **Z**ero result from ALU
  - C = ALU operation **C**arried out
  - V = ALU operation o**V**erflowed
- 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



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
```

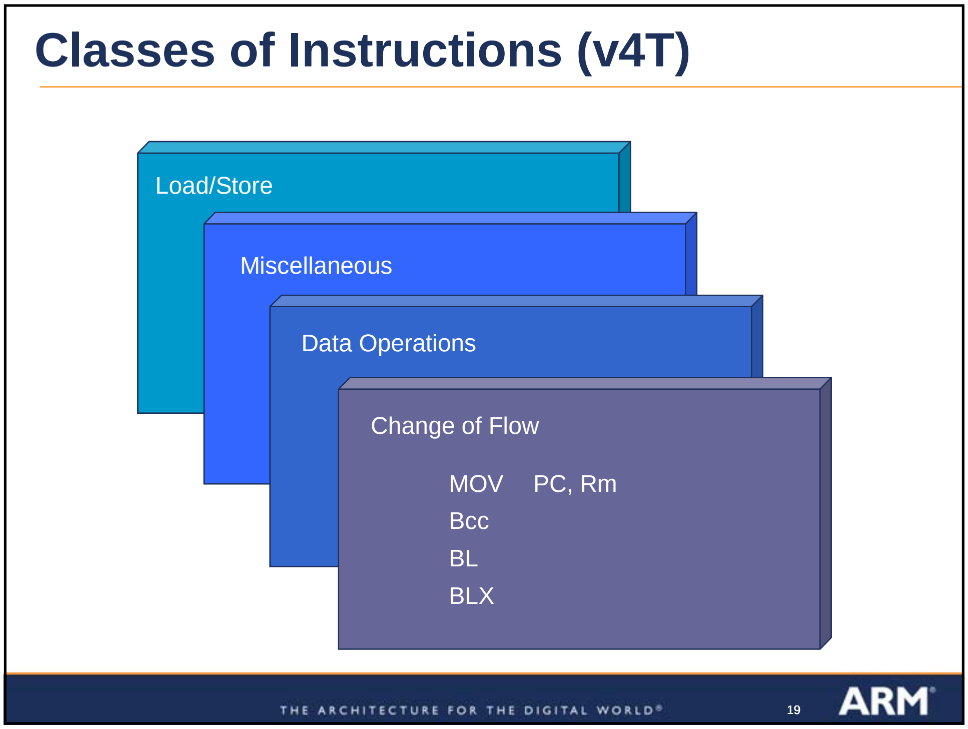
```
CMP    r3,#0
ADDNE  r0,r1,r2
```

- 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
```

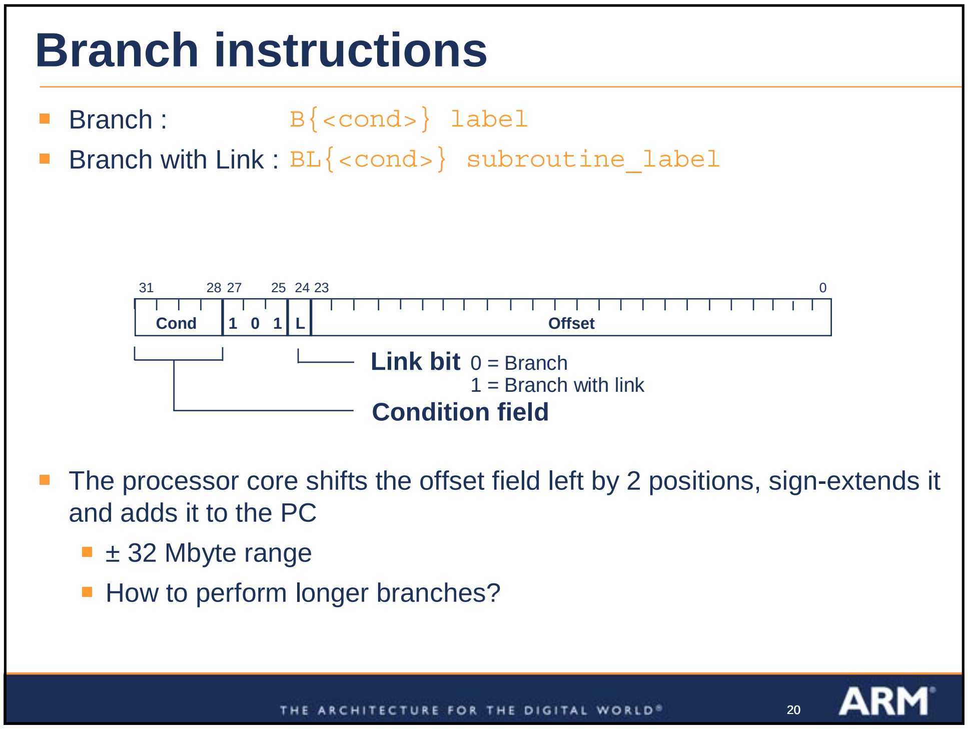
decrement r1 and set flags

if Z flag clear then branch



Classes of Instructions (v4T)

```
MOV    PC, Rm
Bcc
BL
BLX
```

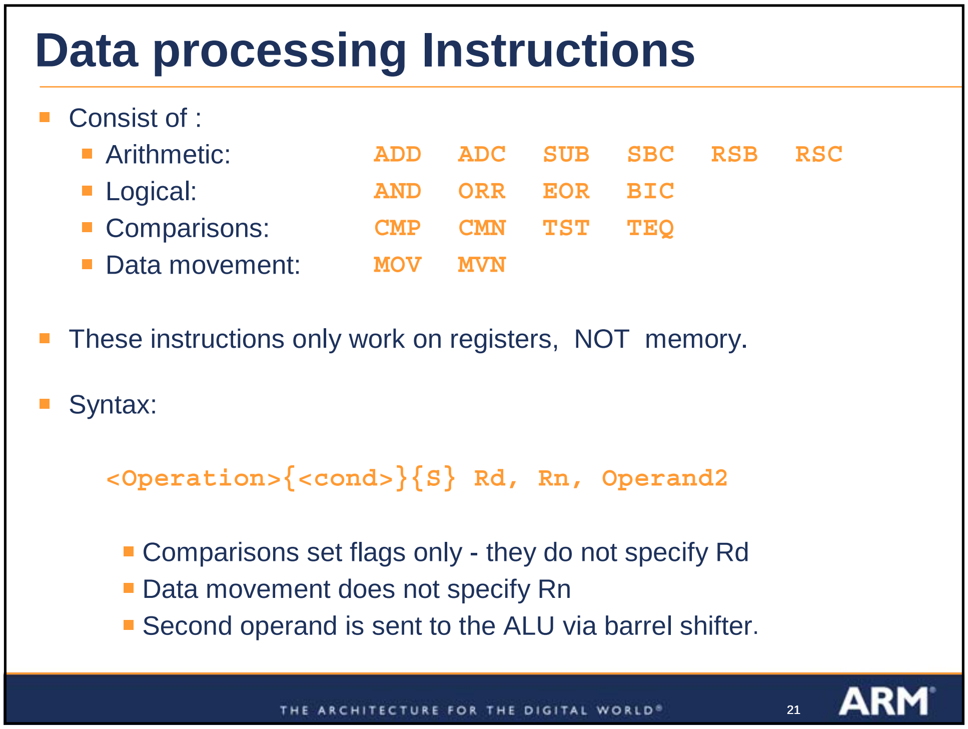


Branch instructions

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

31–28	27–25	24	23–0
Cond	1 0 1	L	Offset

- 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 :

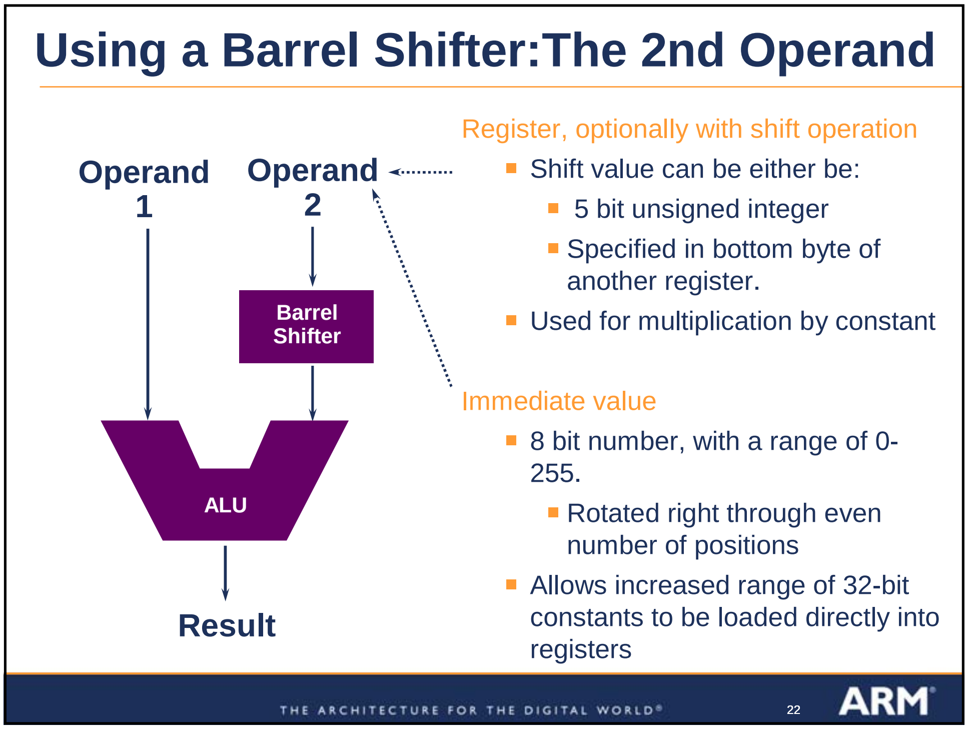
Arithmetic:	ADD	ADC	SUB	SBC	RSB	RSC
Logical:	AND	ORR	EOR	BIC		
Comparisons:	CMP	CMN	TST	TEQ		
Data movement:	MOV	MVN				

- 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.



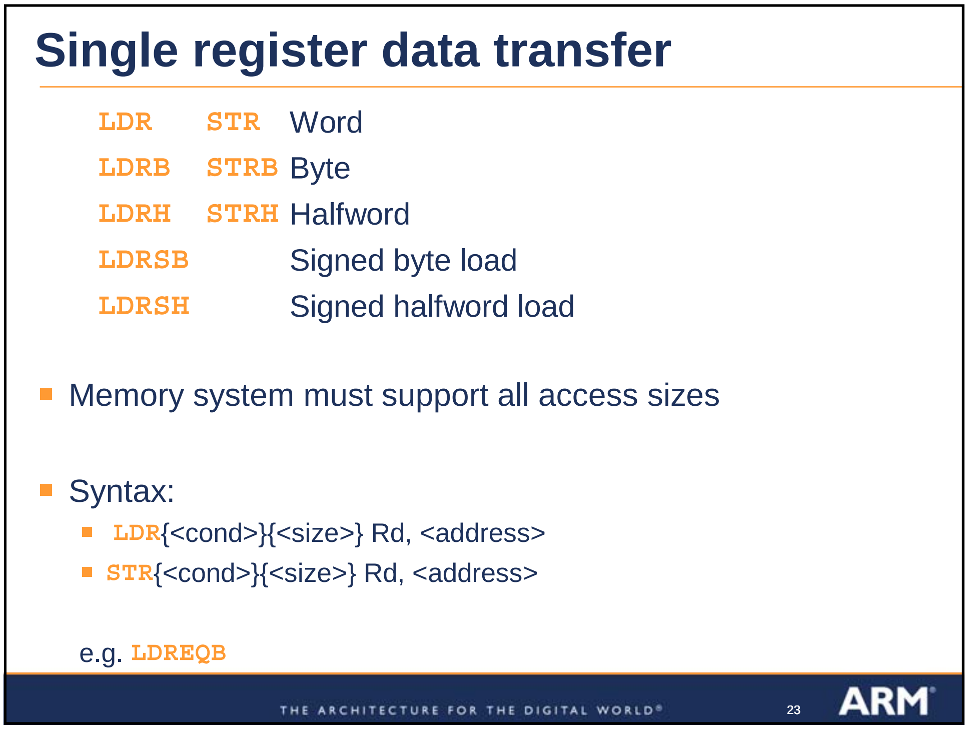
Using a Barrel Shifter:The 2nd Operand

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



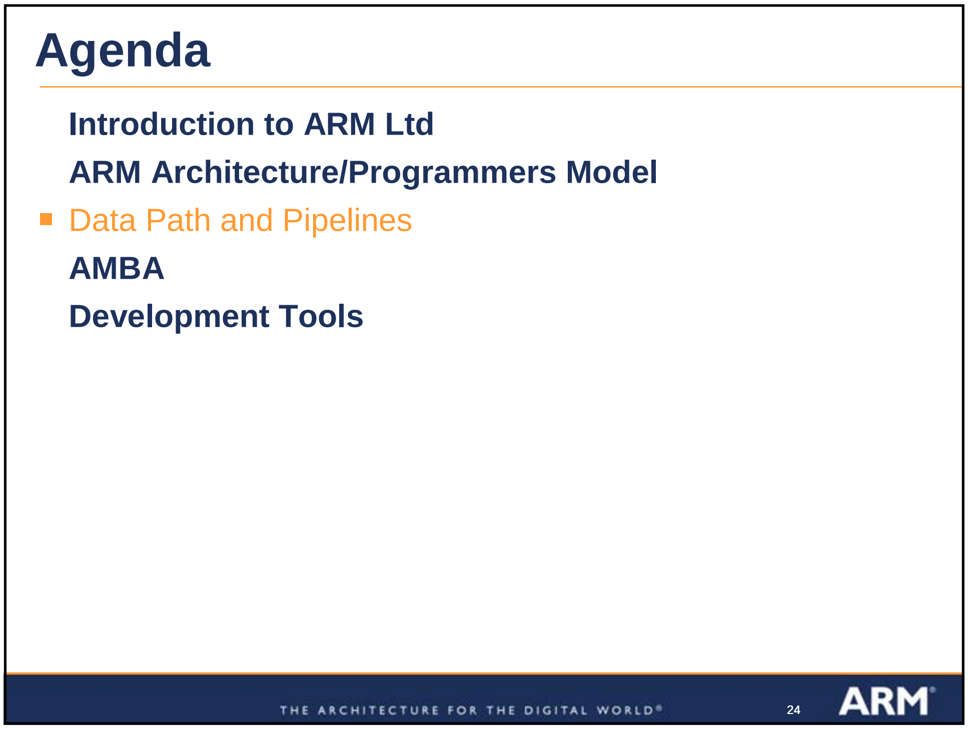
Single register data transfer

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

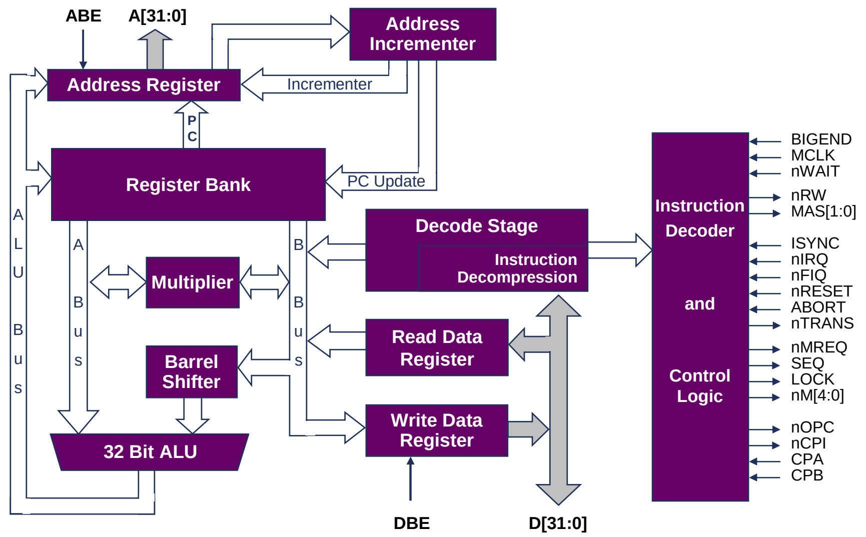
ARM Architecture/Programmers Model

- Data Path and Pipelines

AMBA

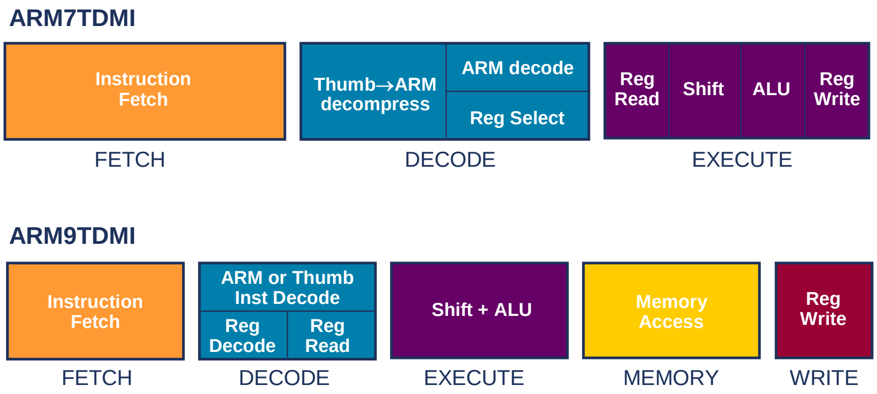
Development Tools

The ARM7TDM Core



ABE
A[31:0]
Address Incrementer
Address Register
Incrementer
P C
Register Bank
PC Update
A L U B u s
A B u s
B B u s
Multiplier
Barrel Shifter
32 Bit ALU
Decode Stage
Instruction Decompression
Read Data Register
Write Data Register
DBE
D[31:0]
Instruction Decoder and Control Logic
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

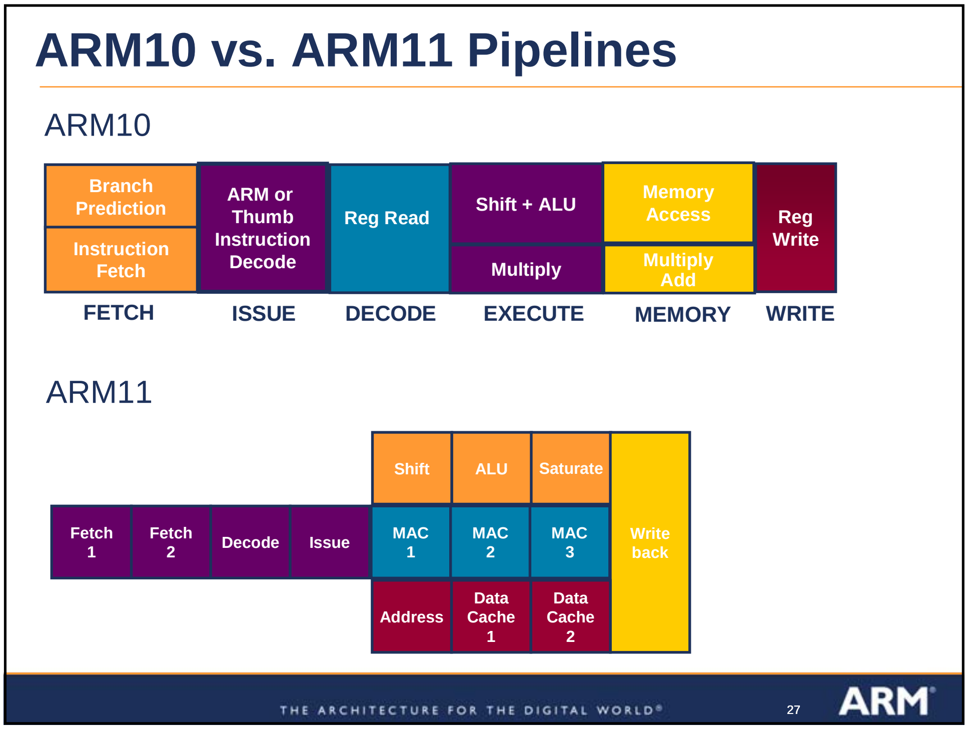
Instruction Fetch
Thumb→ARM decompress
ARM decode
Reg Select
Reg Read
Shift
ALU
Reg Write

FETCH
DECODE
EXECUTE

ARM9TDMI

Instruction Fetch
ARM or Thumb Inst Decode
Reg Decode
Reg Read
Shift + ALU
Memory Access
Reg Write

FETCH
DECODE
EXECUTE
MEMORY
WRITE

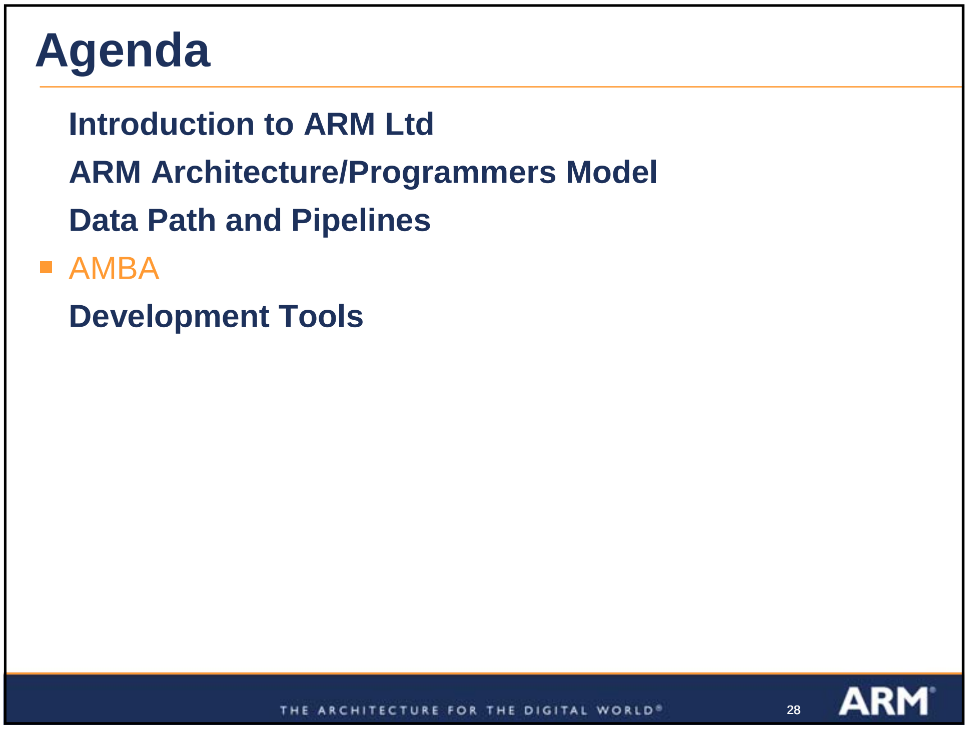


ARM10 vs. ARM11 Pipelines

ARM10

ARM11

THE ARCHITECTURE FOR THE DIGITAL WORLD® 27 ARM



Agenda

Introduction to ARM Ltd

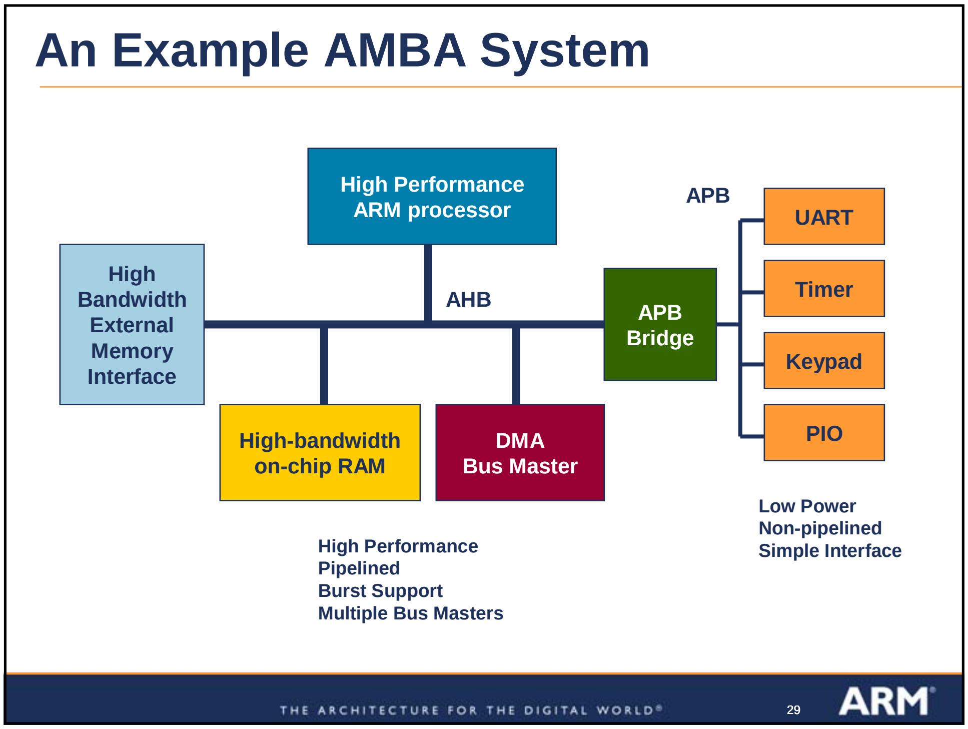
ARM Architecture/Programmers Model

Data Path and Pipelines

- AMBA

Development Tools

THE ARCHITECTURE FOR THE DIGITAL WORLD® 28 ARM

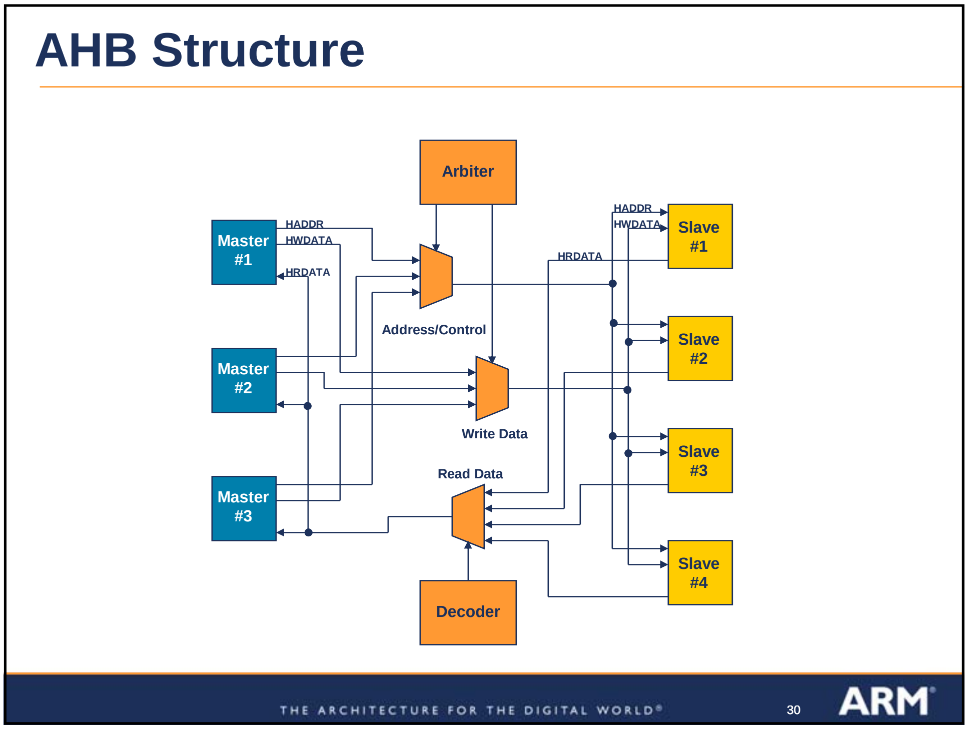


An Example AMBA System

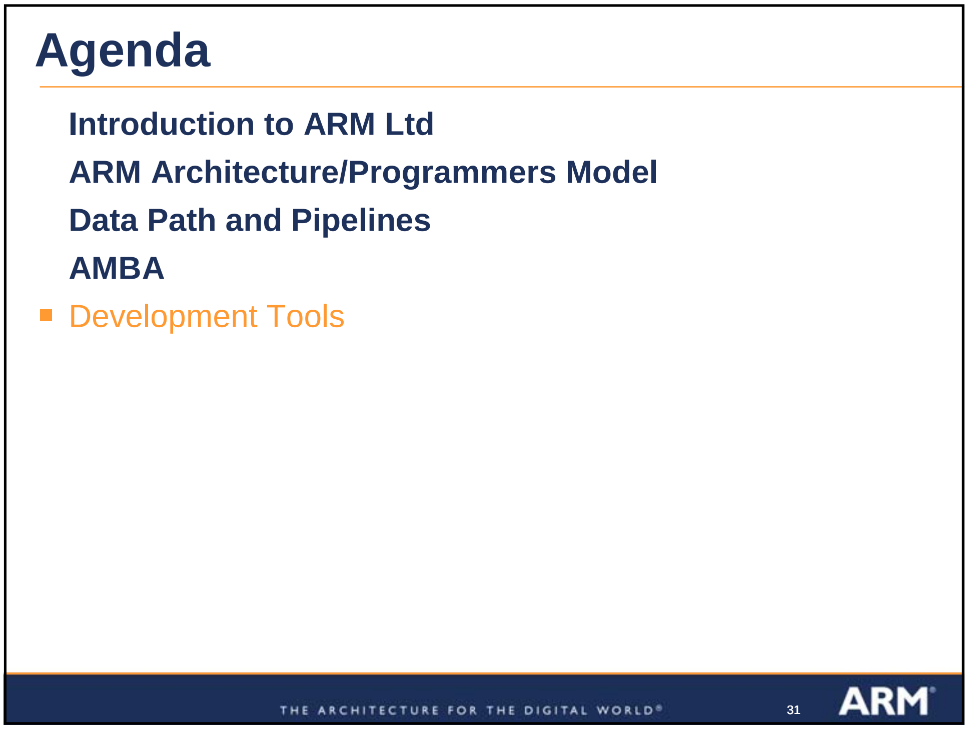
High Performance ARM processor
High Bandwidth External Memory Interface
AHB
APB Bridge
APB
UART
Timer
Keypad
PIO
High-bandwidth on-chip RAM
DMA Bus Master

Low Power
Non-pipelined
Simple Interface

High Performance
Pipelined
Burst Support
Multiple Bus Masters



AHB Structure



Agenda

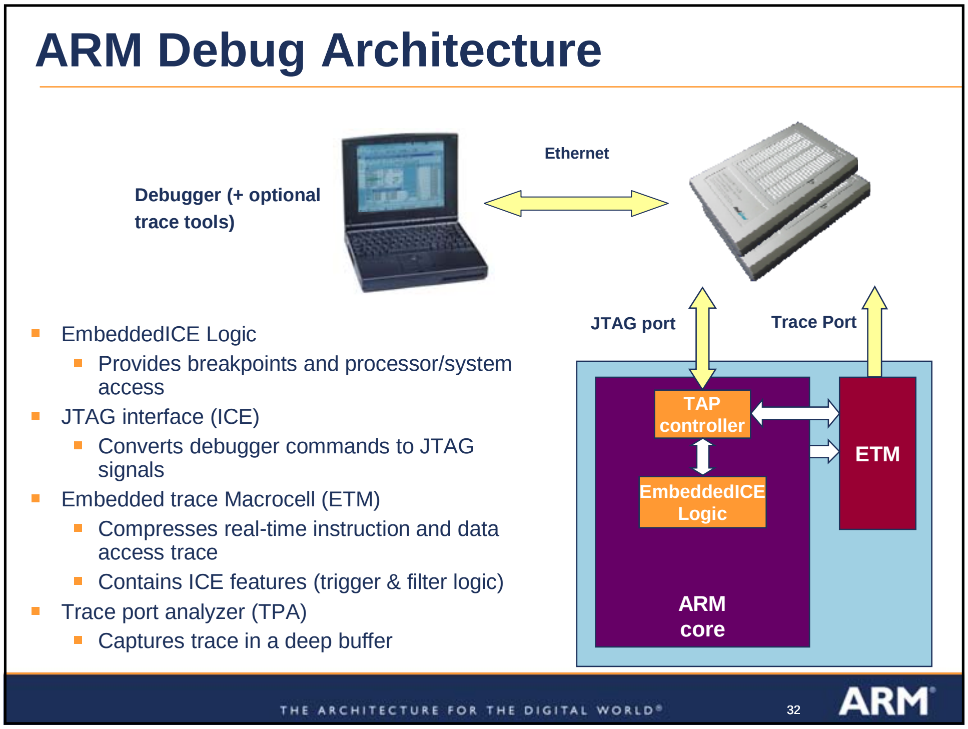
Introduction to ARM Ltd

ARM Architecture/Programmers Model

Data Path and Pipelines

AMBA

- Development Tools



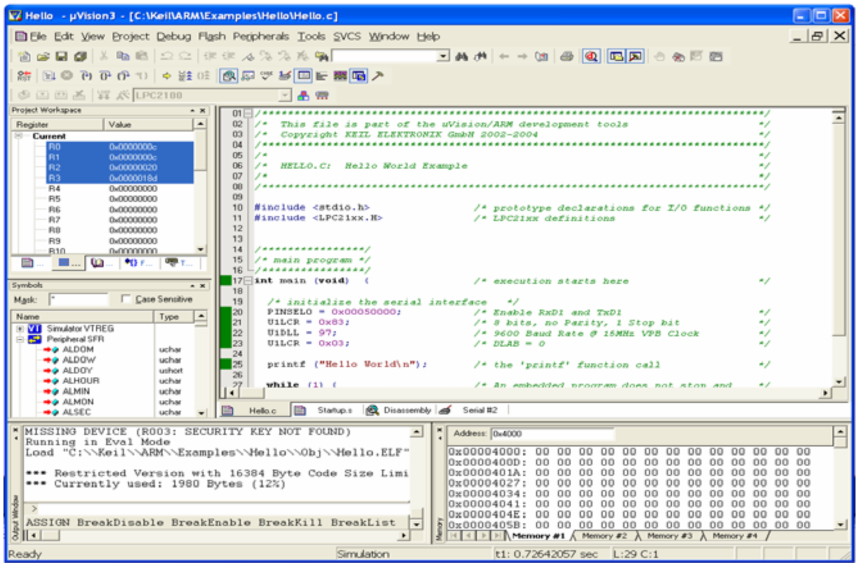
ARM Debug Architecture

- 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 (I^2C, 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



Register	Value
Current	
R0	0x0000000c
R1	0x0000000c
R2	0x00000020
R3	0x0000018d
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000

Name	Type
Simulator VTREG	
Peripheral SFR	
ALDOM	uchar
ALDOW	uchar
ALDOY	ushort
ALHOUR	uchar
ALMIN	uchar
ALMON	uchar
ALSEC	uchar

```
/******************************************************************************/
/*  This file is part of the uVision/ARM development tools                    */
/*  Copyright KEIL ELEKTRONIK GmbH 2002-2004                                  */
/******************************************************************************/
/*                                                                            */
/*  HELLO.C:  Hello World Example                                             */
/*                                                                            */
/******************************************************************************/

#include <stdio.h>                /* prototype declarations for I/O functions */
#include <LPC21xx.H>              /* LPC21xx definitions                      */


/****************/
/* main program */
/****************/
int main (void)  {                /* execution starts here                    */

  /* initialize the serial interface   */
  PINSEL0 = 0x00050000;           /* Enable RxD1 and TxD1                     */
  U1LCR = 0x83;                   /* 8 bits, no Parity, 1 Stop bit            */
  U1DLL = 97;                     /* 9600 Baud Rate @ 15MHz VPB Clock         */
  U1LCR = 0x03;                   /* DLAB = 0                                 */

  printf ("Hello World\n");       /* the 'printf' function call               */

  while (1) {                     /* An embedded program does not stop and    */
```

```
MISSING DEVICE (R003: SECURITY KEY NOT FOUND)
Running in Eval Mode
Load "C:\\Keil\\ARM\\Examples\\Hello\\Obj\\Hello.ELF"

*** Restricted Version with 16384 Byte Code Size Limi
*** Currently used: 1980 Bytes (12%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x4000
0x00004000: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000400D: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000401A: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004027: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004034: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004041: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000404E: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000405B: 00 00 00 00 00 00 00 00 00 00 00 00 00
```

ARM®

THE ARCHITECTURE
FOR THE DIGITAL WORLD™

THE ARCHITECTURE FOR THE DIGITAL WORLD®

35

ARM®