ARM Cortex-M3 Introduction

ARM University Relations





Agenda

Cortex-M3 Overview

v7-M Architecture/Programmers Model

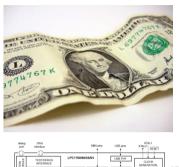
Data Path and Pipelines

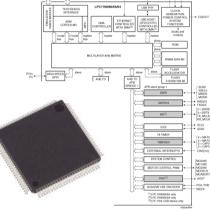
Tools and mbed Platform



What's Happening in Microcontrollers?

- Microcontrollers are getting <u>cheap</u>
 - 32-bit ARM Cortex-M3 Microcontrollers @ \$1
 - Some microcontrollers sell for as little as \$0.65
- Microcontrollers are getting powerful
 - Lots of processing, memory, I/O in one package
 - Floating-point is even available in some!
- Microcontrollers are getting interactive
 - Internet connectivity, new sensors and actuators
 - LCD and display controllers are common

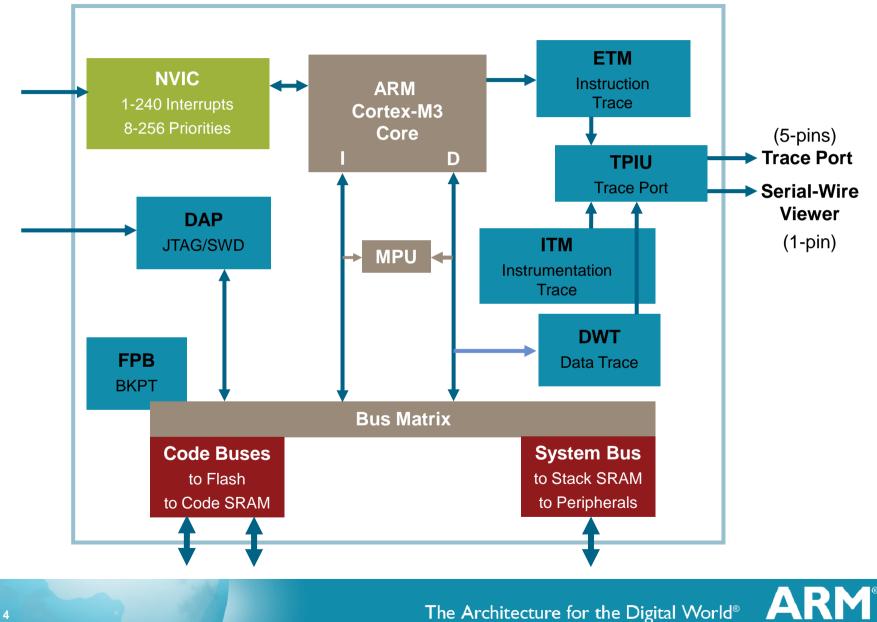






Creates new opportunities for microcontrollers

ARM Cortex-M3 Processor



The Architecture for the Digital World®

ARM Cortex-M3 Microcontroller

- 18 x 32-bit registers
- Excellent compiler target
- Reduced pin count requirements
- Efficient interrupt handling
- Power management
- Efficient debug and development support features
 - Breakpoints, Watchpoints,
 - Flash Patch support,
 - Instruction Trace
- Strong OS support
 - User/Supervisor model
 - OS support features
- Designed to be fully programmed in C (even reset, interrupts and exceptions)

ARM Cortex-M3 Microcontroller

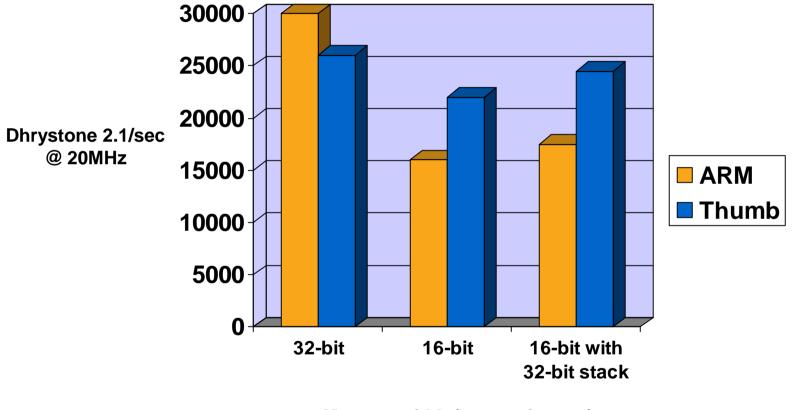
- ARMv7M Architecture
- No Cache No MMU
- Debug is optimized for microcontroller applications
- Vector table contains addresses, not instructions
- DIV instruction
- Interrupts automatically save/restore state
- Exceptions programmed in C (No Coprocessor 15 All registers are memory-mapped)
- Interrupt controller is part of Cortex-M3 macrocell
- Fixed memory map
- Bit-banding

6

- Non-Maskable Interrupt (NMI)
- Only one processor status reg
- Thumb-2 processing core
 - Mix of 16 and 32 bit instructions for very high code density
 - Gives complete Thumb compatibility



ARM and Thumb Performance

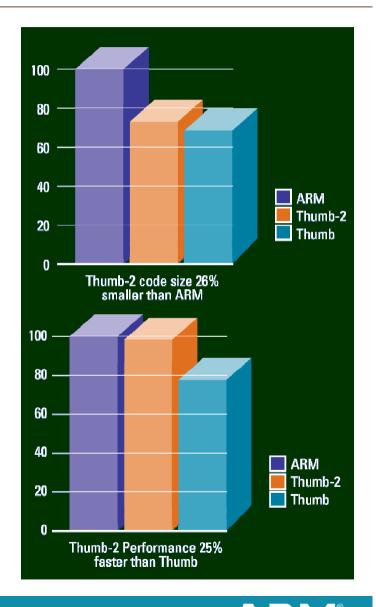


Memory width (zero wait state)



The Thumb-2 instruction set

- Variable-length instructions
 - ARM instructions are a fixed length of 32 bits
 - Thumb instructions are a fixed length of 16 bits
 - Thumb-2 instructions can be either 16-bit or 32-bit
- Thumb-2 gives approximately 26% improvement in code density over ARM
- Thumb-2 gives approximately 25% improvement in performance over Thumb



Agenda

Cortex-M3 Overview

v7-M Architecture/Programmers Model

Data Path and Pipelines

Tools and mbed Platform



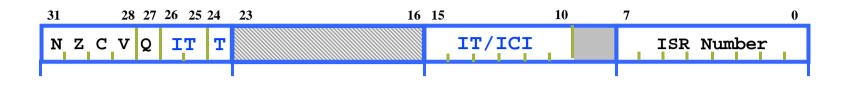
Cortex-M3 Register Set

- Very compiler friendly
 - Load/Store Architecture
 - 32-bit registers
 - Flexible register scheme
 - Linear 32-bit address space



	_
r0	
r1	
r2	
r3	
r4	
r5	
r6	
r7	
r8	
r9	
r10	
r11	
r12	Process
sp	
lr	sp
r15 (pc)	
	1
xPSR	

Program Status Register



One Status Register consisting of

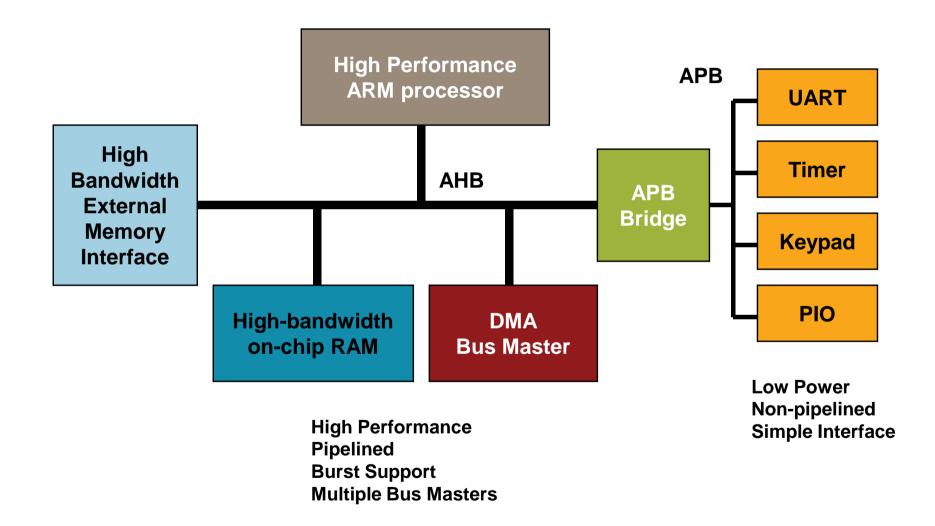
- APSR Application Program Status Register ALU flags
- IPSR Interrupt Program Status Register Interrupt/Exception No.
- EPSR Execution Program Status Register
 - IT field If/Then block information
 - ICI field Interruptible-Continuable Instruction information

xPSR

- Composite of the 3 PSRs
- Stored on the stack on exception entry



An Example AMBA System

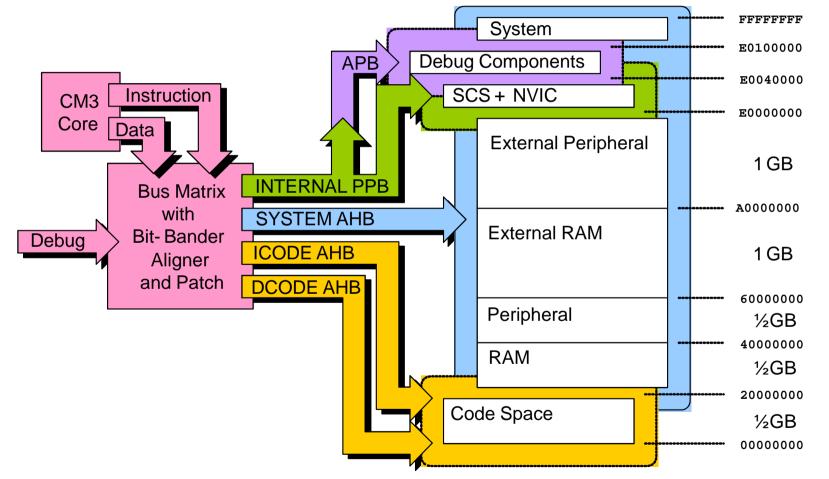


The Architecture for the Digital World®

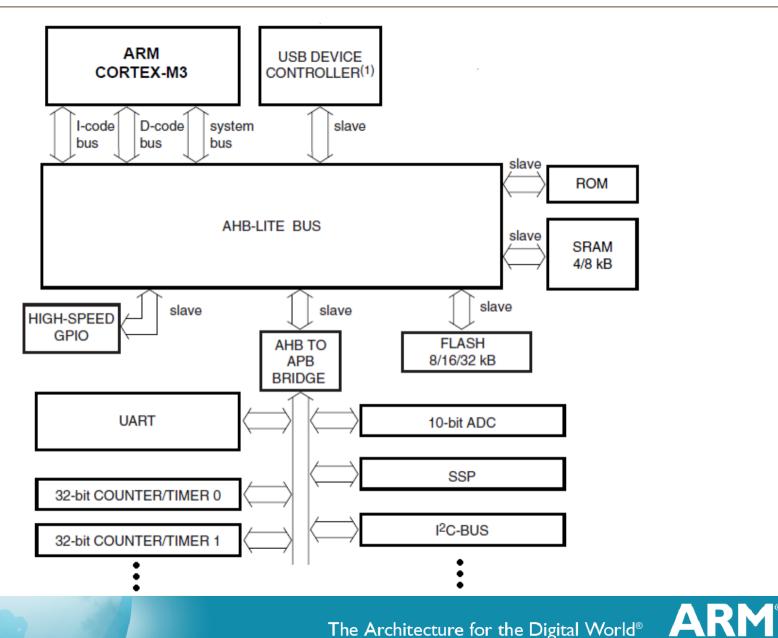
ARM

Memory Map

- Very simple linear 4GB memory map
- The Bus Matrix partitions memory access via the AHB and PPB buses

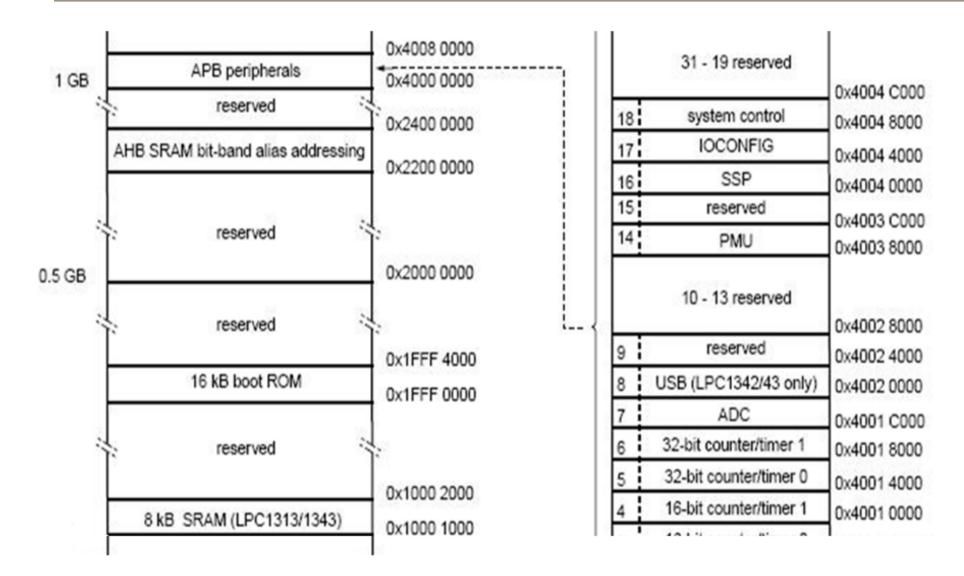


NXP LPC1311/13/42/43 Block Diagram

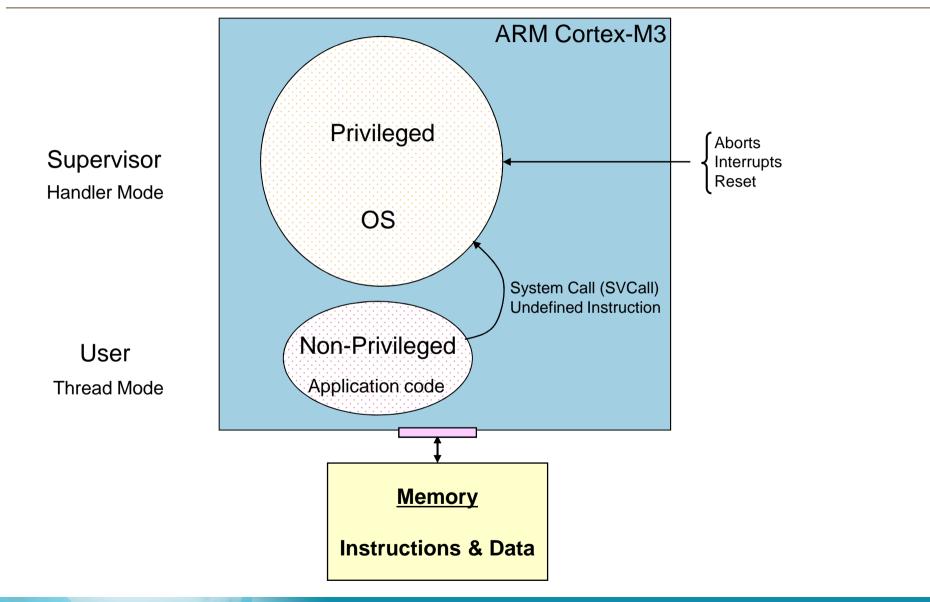


The Architecture for the Digital World®

NXP LPC1311/13/42/43 Memory Map



Processor Privilege



The Architecture for the Digital World®

ARM

Memory Protection Unit (MPU)

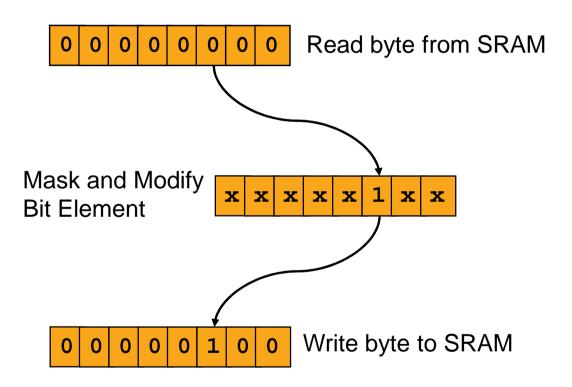
MPU provides access control for various memory regions

- Zero Latency Memory Protection
 - 8 register-stored regions
 - Same regions used for instructions and data
 - Minimum region size 32 Bytes (max 4GB)
 - No address translation or page tables
- Configured via memory-mapped control registers



Cortex-M3 Bit Banding

Traditional Method of Atomic Manipulation



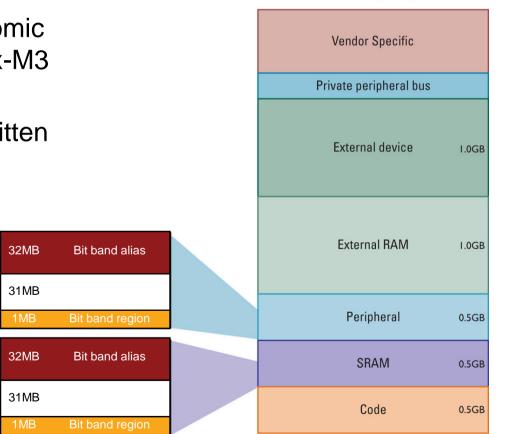


Cortex-M3 Bit Banding

- Writes to a word address in the bit band alias affect a single bit in the bit band region
- The write is translated to an atomic read-modify-write by the Cortex-M3 bus matrix
- Bit 0 of the stored register is written to the appropriate bit

Word alias

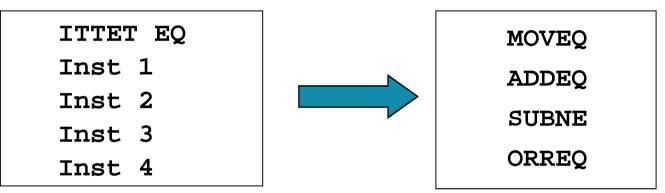
Physical bit



Memory map

Conditional Execution

- If Then (IT) instruction added (16 bit)
 - Up to 3 additional "then" or "else" conditions maybe specified (T or E)
 - Makes up to 4 following instructions conditional

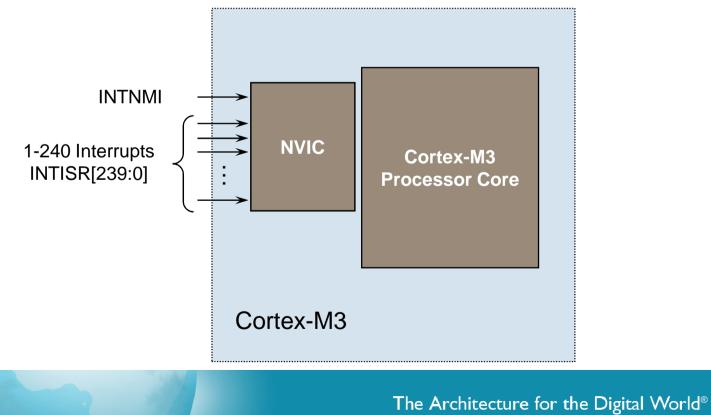


- Any normal ARM condition code can be used
- 16-bit instructions in block do not affect condition code flags
 - Apart from comparison instruction
 - 32 bit instructions may affect flags (normal rules apply)
- Current "if-then status" stored in CPSR
 - Conditional block maybe safely interrupted and returned to
 - Must NOT branch into or out of 'if-then' block

Interrupt Handling

- One Non-Maskable Interrupt (INTNMI) supported
- 1-240 prioritizable interrupts supported
 - Interrupts can be masked
 - Implementation option selects number of interrupts supported
- Nested Vectored Interrupt Controller (NVIC) is tightly coupled with processor core
- Interrupt inputs are active HIGH

21



Exception Handling

- Reset
- NMI

Faults

- Hard Fault
- Memory Manage
- Bus Fault
- Usage Fault
- SVCall
- Debug Monitor
- PendSV
- SysTick Interrupt
- External Interrupt

Power Management

- Multiple sleep modes supported
 - Controlled by NVIC
 - Sleep Now Wait for Interrupt/Event instructions
 - Sleep On Exit Sleep immediately on return from last ISR
 - Deep Sleep
 - Long duration sleep, so PLL can be stopped
 - Exports additional output signal SLEEPDEEP
- Cortex-M3 system is clock gated in all sleep modes
 - Sleep signal is exported allowing external system to be clock gated also
 - NVIC interrupt Interface stays awake
- Wake-Up Interrupt Controller (WIC)
 - External wake-up detector allows Cortex-M3 to be fully powered down
 - Effective with State-Retention / Power Gating (SRPG) methodology



Cortex-M3 Debug

- Single stepping
- ITM (Instrumentation Trace Module)
 - Support for instrumented code
 - Like "printf" debugging, but single cycle writes to ITM module can be exported via serial interface
- Optional ETM (Embedded Trace Module) which provides instruction trace
- Access to all memory and registers via Debug Access Port (DAP)
- Profiling Support
 - Helps you optimize your code
 - A hardware triggered PC-sampler is provided by the Cortex-M3 core
- Flash patch / Breakpoints
 - 6 instruction comparators, ARMv5T BKPT instruction is supported
 - 2 literal comparators
 - Allows flash code to be remapped into SRAM or system address space
- Data Watchpoint and Trace (DWT) that implements 4 h/w watchpoints



Agenda

Cortex-M3 Overview

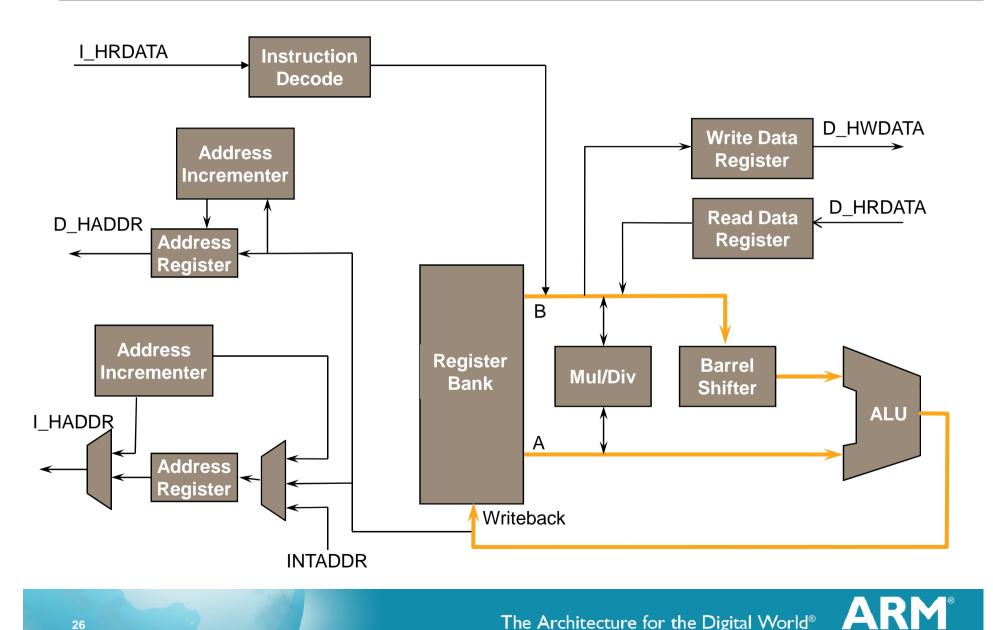
v7-M Architecture/Programmers Model

Data Path and Pipelines

Tools and mbed Platform



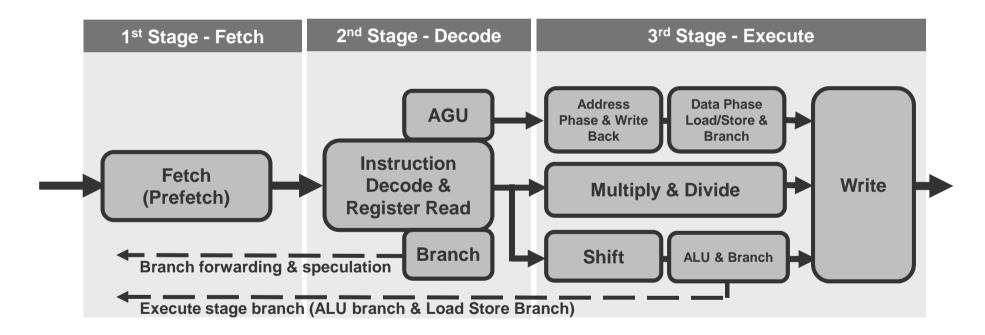
Cortex-M3 Datapath



The Architecture for the Digital World®

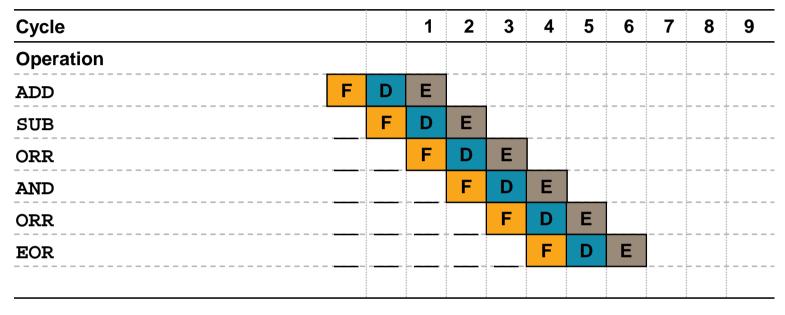
Cortex-M3 Pipeline

- Cortex-M3 has 3-stage fetch-decode-execute pipeline
 - Similar to ARM7
 - Cortex-M3 does more in each stage to increase overall performance



The Architecture for the Digital World[®]

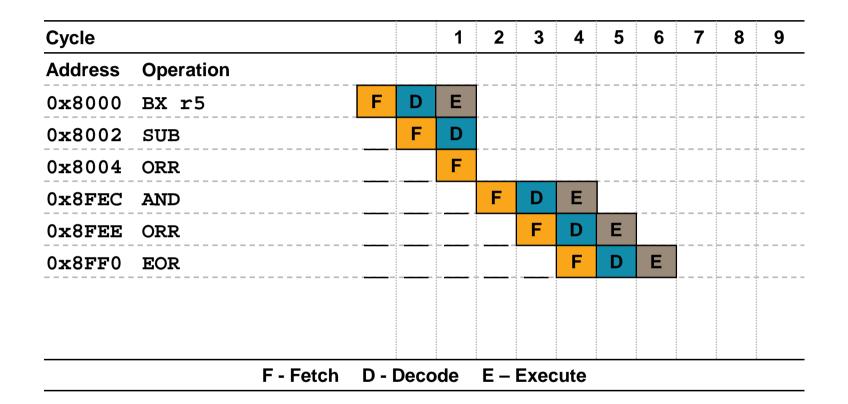
Optimal Pipelining



F - Fetch D - Decode E - Execute

- All operations here are on registers (single cycle execution)
- In this example it takes 6 clock cycles to execute 6 instructions
- Clock cycles per Instruction (CPI) = 1

Branch Pipeline Example



- It takes 3 cycles to complete the branch
- Worst case scenario indirect branch taken

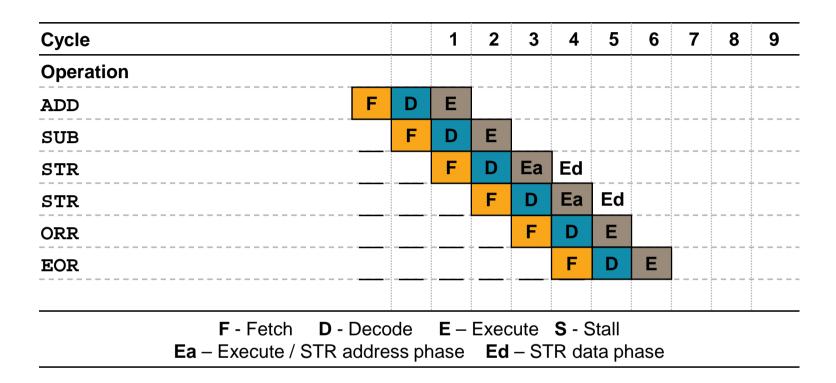
ARI

LDR Pipeline Example

Cycle			1	2	3	4	5	6	7	8	9
Operation		-			_						
ADD	 F	D	Е								
SUB	 	F	D	Е							
LDR	 		F	D	Ea	Ed					
AND	 			F	D	S	Е				
ORR	 				F	S	D	Е			
EOR	 						F	D	Е		
	 				_						

- In this example it takes 7 clock cycles to execute 6 instructions
- Clock cycles per Instruction (CPI) = 1.2
- The read cycle must complete on the bus before the LDR instruction can complete since there is only one writeback port in the register file

STR-STR Pipeline Example



- Store buffer allows STR instruction to finish before store cycle completes on the bus
- Back-to-back STR instructions pipeline on the AHB-Lite bus
 Also works for LDR-STR-LDR-STR



Agenda

Cortex-M3 Overview

v7-M Architecture/Programmers Model

Data Path and Pipelines

Tools and mbed Platform



RVMDK Software Development Tools



- 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 Version
 - 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/

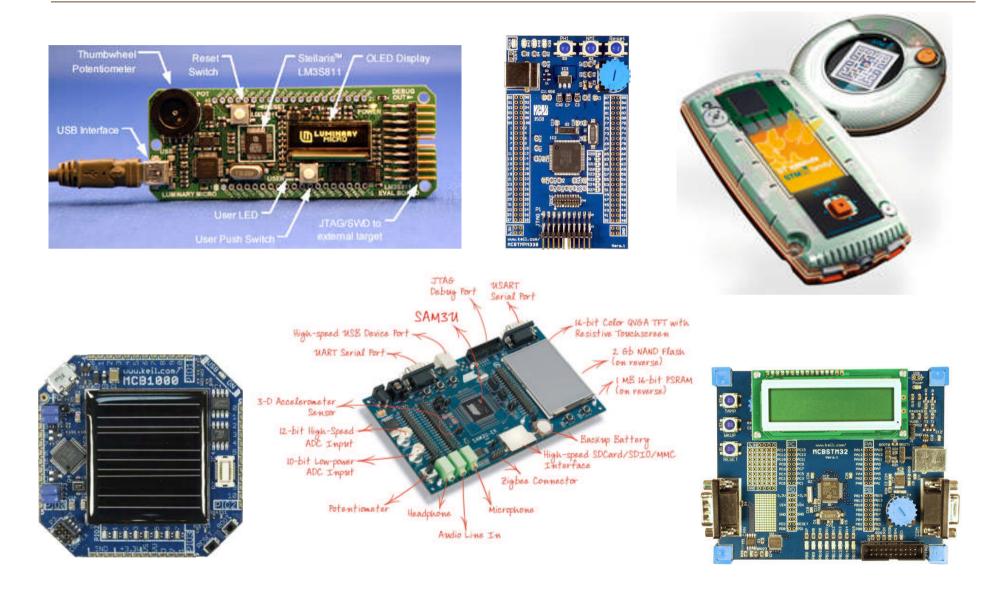


RVMDK Software Development Tools

<pre>pier Vake Vake Vake Control Vake Control Vake Contrent Vaked Control Vake Contro</pre>	🔽 Hello - µVision3 - [C:\Keil\ARM\	Examples\Hello\Hello.c]	
Image: Section State Image: Section State Image: Sectio	File Edit View Project Debug F	il <u>a</u> sh Pe <u>ri</u> pherals <u>T</u> ools <u>S</u> VCS <u>W</u> indow <u>H</u> elp	_ 8 ×
Image: Section State Image: Section State Image: Sectio	참 술 🖬 🗿 👗 🖻 🛍 그 으 -	╪╪ <u>ѧ%%%%</u> , ⋈	
Image: Strike PPC2100 Image: Strike PPC2100 Image: Strike Version Image: Strike Version Image: Version Image: Strike Version Image: Strike Version Image: Strike Version			
<pre>et Workspee Vake</pre>			
piper Value 02 This file is part of the Wission/ARM derelopment tools wind 00000000 - Copyright KEIL ELEXITORIAR derelopment tools - / / / / / / / / / / / / / / / / / / /			
Image: Constraints 00 /* Constraints */* 100 000000000000000000000000000000000000	Register Value 🔺		_
11 000000000 1/2 12 000000000 1/2 13 000000000 1/2 14 000000000 1/2 15 0.00000000 1/2 16 0 1/2 17 0.00000000 1/2 18 0.00000000 1/2 19 #include 1/2 11 #include 1/2 12 1/2 1/2 19 1/2 1/2 10 #include 1/2 11 #include 1/2 12 1/2 1/2 11 #include 1/2 12 1/2 1/2 13 1/2 1/2 14 #include 1/2 15 1/2 1/2 16 1/2 1/2 17 1/2 1/2 17 1/2 1/2 17 1/2 1/2 17 1/2 1/2 17 1/2 1/2			
32 Decomposed */ 33 Decomposed */ 34 Decomposed */ 35 Decomposed */ 36 Decomposed */ 36 Decomposed */ 37 Decomposed */ 38 Decomposed */ 39 # # /* 30 # # /* 30 # /* # 31 # /* # 31 # /* # 31 # /* # 31 # /* # 31 # /* # 32 Decomposed */ 33 # /* # 34 # # # # 34 # # # # # 34 # # # # # # 35 # # # # # #	R0 0x000000c		
B3 0x0000000 B4 0x0000000 B5 0x0000000 B6 0x0000000 B7 0x00000000 B8 0x00000000 B9 0x00000000 B10 0x00000000 B11 ************************************			
R4 Docococcocc R5 Docococcocc R5 Dococcoccocc R5 Dococcoccocc R5 Dococcoccocc R7 Dococcoccocc R7 Dococcoccocc Dococcoccocc */ R7 Dococcoccocc Dococcoccocc */ R7 Dococcoccocc Dococcoccoccoccocc */ R7 Dococcoccocc Dococcoccoccocc */ R7 Dococcoccoccocc Dococcoccocc */ R7 Dococcoccocc Dococcoccoccocc */ R7 Dococcoccocc Dococcoccoccoccocc */ R7 Dococcoccoccocc Dococcoccoccocc */ R7 Dococcoccoccoccoccoccoccoccocc D000 Coccoccoccoccoccoccoccoccoccoccoccoccocc			
FS 000000000000000000000000000000000000			
Bit #include <stdio.b> /* prototype declarations for I/O functions */ B7 000000000 */ B7 000000000 */ B1 */ */</stdio.b>			
B7 0x0000000 B80 11 #include <lpc21xx.b> /* LPC21xx definitions */ B8 0x0000000 B10 0x0000000 B10 0x0000000 B10 */ */ */ B1 0x0000000 B10 0x0000000 B10 */ */ */ */ B1 0x0000000 B10 0x0000000 B10 */ */ */ */ B1 */ */ */ */ */ */ B1 */ */ */ */ */ */ B1 */ */ */ */ */ */ */ */ */ */ */ */ */ */ */ */ */ */ */ */<!--</td--><td></td><td>10 #include <stdio.h> /* prototype declarations for I/O functions */</stdio.h></td><td></td></lpc21xx.b>		10 #include <stdio.h> /* prototype declarations for I/O functions */</stdio.h>	
Bin Decomposition Bin PinNSELO = 0x00050000; PinNSELO = 0x00050000; /* Enable RxD1 and TxD1 PinNSELO = 0x00050000; /* Binis, no Parity, 1 Stop bit Dilcc = 0x03; /* Binis, no Parity, 1 Stop bit PinNSELO = 0x0005000; /* Binis, no Parity, 1 Stop bit PinNSELO = 0x000500; /* Binis, no Parity, 1 Stop bit PinNSELO = 0x003; /* Binis, no Parity, 1 Stop bit PinNSELO = 0x003; /* Binis, no Parity, 1 Stop bit PinNSELO = 0x003; /* DLAB = 0 PinnesdSFR Pinintf ("Hello World\n"); ALDOW uchar ALMON uchar ALMON uchar ALMON uchar Alloc E Statup: Disastembly Steid E Modol Coa		11 #include <lpc21xx.h> /* LPC21xx definitions */</lpc21xx.h>	
Bin Nonononon Hin Nononononon Hin Nonononononon Hin Nononononononononononononononononononon			
Immunit 15 /* main program */ Immunit Immunit 15 /* main (void) /* execution starts here */ Immunit Immunit Immunit Immunit /* /* /* Immunit Immunit Immunit /* /* /* /* Immunit Immunit Immunit /* /* /* /* Immunit Immunit Immunit /* /* /* /* /* Immunit Immunit Immunit /*			
Image: Structure with the second of the s			
bols bols	🖹 🗮 🚺 👫 F 🗮 🌄 T		
tools ** is initialize the serial interface */ PINSELD = 0x00050000; /* Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Enable RxD1 and TxD1 PINSELD = 0x00050000; /* & Bots, no Parity, 1 Stop bit ** ALDOM uchar ** ALDOW uchar ** ALDOW uchar ** ALMON uchar ** Aldon uchar ** Action ** ** Action ** ** Action ** ** Auton ** ** Aloo ** <t< td=""><td></td><td>17 int main (void) { /* execution starts here */</td><td></td></t<>		17 int main (void) { /* execution starts here */	
PinsEL0 = 0x00050000; /* Enable RxD1 and TxD1 */ Wi Simulator VIREG UILCR = 0x83; /* 8 bits, no Parity, 1 Stop bit */ Peripheral SFR UILCR = 0x83; /* 9600 Baud Rate 6 15MHz VFB Clock */ * ALDOW uchar printf ("Hello World\n"); /* the 'printf' function call */ * ALDOW uchar uchar while (1) (/* An embedded program does not stop and */ */ * ALMON uchar uchar while (1) (/* An embedded program does not stop and */ */ * ALMON uchar uchar While (1) (/* An embedded program does not stop and */ */ * ALMON uchar uchar While (1) (/* An embedded program does not stop and */ */ * ALMON uchar uchar While (1) (/* Addess: 0x4000 */ */ Running in Eval Mode C:\\Keil\\ARM\\Examples\\Hello\\Obj\\Hello.ELF" */ Addess: 0x4000 */ */ *** Currently used: 1980 Bytes (12%) 12% */ */ */ */ */ **** SSIGN BreakDisable BreakEnable BreakKill BreakList */ */ */ *			
me Upe 21 U1LCR = 0x83; /* 8 bits, no Parity, 1 Stop bit */ Peripheral SFR U1DLL = 97; /* 9600 Baud Rate @ 15MHz VPB Clock */ ALDOW uchar uchar 22 printf ("Hello World\n"); /* the 'printf' function call */ ALDOW uchar uchar uchar */ */ */ ALDOW uchar uchar uchar */ */ */ */ ALMIN uchar uchar uchar */ */ */ */ ALMIN uchar uchar uchar */ */ */ */ */ ALMON uchar uchar uchar uchar */ */ */ */ ALMON uchar uchar uchar */ */ */ */ */ ALMON uchar uchar uchar * */ */ */ */ ALMON uchar uchar uchar */ */ */ */ */ ALMON uchar uchar uchar */	A <u>a</u> sk: * □ <u>C</u> ase Sensitive		
Vir Simulator VTREG 21 UILCR = 0x83; /* 8 bits, no Parity, 1 stop bit */ Perpheral SFR - ALDOM uchar 10LCR = 0x03; /* 9600 Bauk Rate @ 15MHz VPB Clock */ • ALDOW uchar uchar printf ("Hello World\n"); /* the 'printf' function call */ • ALDOW uchar uchar uchar */ */ • ALMUN uchar uchar ** */ */ */ • ALMON uchar uchar */ */ */ */ • ALMON uchar uchar */ */ */ */ • ALMON uchar uchar */ */ */ */ • ALSEC uchar */ */ */ */ */ • Almon uchar */ */ */ */ */ */ • Almon uchar */ */ */ */ */ */ • Almon uchar */ */ */ */ */ */ C:::::::::::::::::::::::	Name Type 🔺		
Perpheral SH • ALDOM • ALDOW • ALDOW • ALDOW • ALDOW • ALHOUR • ALHOUR • ALHOUR • ALHOUR • ALHOUR • ALMON • ALMON • ALSEC • Order • Order • ALSEC • Order • Order	E VI Simulator VTREG		
ALDDM uchar 24 printf ("Hello World\n"); /* the 'printf' function call */ ALDOW uchar uchar uchar */ */ ALDOW uchar uchar */ */ */ ALDON uchar uchar */ */ ALMON uchar uchar */ */ ALMON uchar uchar */ */ ALSEC uchar */ */ */ MISSING DEVICE (R003: SECURITY KEY NOT FOUND) */ */ */ Running in Eval Mode */ */ */ */ Load "C:\\Keil\\ARM\\Examples\\Hello\\Obj\\Hello.ELF" */ */ */ **** Currently used: 1980 Bytes (12%) */ */ */ */ **** Currently used: 1980 Bytes (12%) */ */ */ */ **** Currently used: 1980 Bytes (12%) */ */ */ */ **** Currently used: 1980 BreakEnable BreakKill BreakList */ */ */ */ **** ********************************	Peripheral SFR		
• ALDOW • ALDOW • ALDOW • ALDOY • ALDOY • ALDOY • ALDOY • ALDOY • ALHOUR • ALHOUR • ALHOUR • ALHOUR • ALHOUR • ALMON • ALMON • ALMON • ALSEC • ALSEC • ALMON • ALSEC • ALMON • ALSEC • ALMON • ALSEC • ALMON • ALSEC • ALSEC • ALMON • ALSEC • ALSEC • ALMON • ALSEC • ALMON • ALSEC • ALSEC • ALSEC • ALSEC • ALMON • ALSEC • ALSEC • ALMON • ALSEC • ALSEC • ALMON • ALSEC • ALGE • ALGE • Address: 0x4000 • Ox00004000: 00 00 00 00 00 00 00 00 00 00 00			
ALDOY ushar 26 while (1) (/* An embedded program does not stop and */ ALMON uchar uchar Image: Constraint of the stop and the stop an	→ 🔿 ALDOW uchar		
•• ALMIN uchar			
• ALMON • ALSEC uchar		, 27 while (1) (/* An embedded program does not stop and */	
•• ALSEC uchar • I Hello.c Startup.s Oisassembly Serial #2 Alssec uchar • I Hello.c Startup.s Oisassembly Serial #2 Alssec Uchar • I Hello.c Startup.s Oisassembly Serial #2 Address: Ox4000 Ox00004000: O0 O0 O0			
Running in Eval Mode Load "C:\Keil\ARM\Examples\Hello\Obj\Hello.ELF" 0x00004000: 00 00 00 00 00 00 00 00 00 00 00 00	· · · ·	📄 Hello.c 🖹 Startup.s 🕵 Disassembly <i>ৰ্ভ</i> Serial #2	
Running in Eval Mode Load "C:\Keil\ARM\Examples\Hello\Obj\Hello.ELF" 0x00004000: 00 00 00 00 00 00 00 00 00 00 00 00	MISSING DEVICE (ROO3: SEC	URITY KEY NOT FOUND)	
<pre>*** Restricted Version with 16384 Byte Code Size Limi *** Currently used: 1980 Bytes (12%) > SSIGN BreakDisable BreakEnable BreakKill BreakList *** Memory #1 Memory #1 Memory #3 Memory #4 /</pre>	Running in Eval Mode		
<pre>*** Restricted Version with 16384 Byte Code Size Limi *** Currently used: 1980 Bytes (12%) *** Currently used: 1980 B</pre>	Load "C:\\Keil\\ARM\\Exam	Free	
*** Currently used: 1980 Bytes (12%) > 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			
ASSIGN BreakDisable BreakEnable BreakKill BreakList			
▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲			
	ASSIGN BreakDisable Break		
		Memory #1 / Memory #2 / Memory #3 / Memory #4 /	
dy Simulation [11: 0.72642057 sec]: 29 C:1	eady	Simulation t1: 0.72642057 sec L:29 C:1	

ARM

Cortex-M3 Development Platforms



The Architecture for the Digital World®

ARM

Rapid Prototyping

Rapid Prototyping helps industries create new products

- Control, communication and interaction increasingly define products
- Development cycles for microelectronics have not kept pace

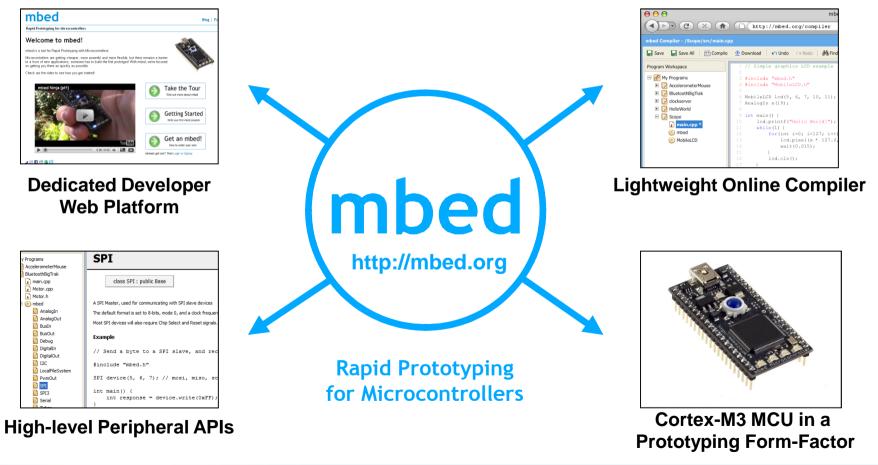




mbed

Getting Started and Rapid Prototyping with ARM MCUs

Complete Targeted Hardware, Software and Web 2.0 Platform





The Architecture for the Digital World®

mbed Motor Controller Example

```
#include "mbed.h"
DigitalOut leftfwd(p16);
DigitalOut leftrev(p17);
PwmOut pwm(p21);
AnalogIn pot(p20);
int main(){
  pwm.period_ms(1);
  while(1){
  float potmod = pot - 0.5;
         if (potmod >= 0) {
                 leftfwd = 1;
                 leftrev = 0;
                 pwm=((1.8 * potmod) + 0.1);
         } else {
                  leftfwd = 0;
                 leftrev = 1;
                 pwm = fabs((1.8 * potmod) - 0.1);
```

Documentation

- ARM v7-M Architecture Reference Manual (ARM v7-M ARM)
- Cortex-M3 Technical Reference Manual (TRM)
- ARM Debug Interface V5 Architecture Spec (ADIv5)
- "The Definitive Guide To The ARM Cortex-M3" by Joseph Yiu (Elsevier)



University Resources

http://www.arm.com/support/university/

university@arm.com

http://www.mbed.org/

http://www.keil.com/



