The ARM Architecture

Agenda

- Introduction to ARM Ltd

  ARM Architecture/Programmers Model

  Data Path and Pipelines

  AMBA/GPU

  IEM

  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

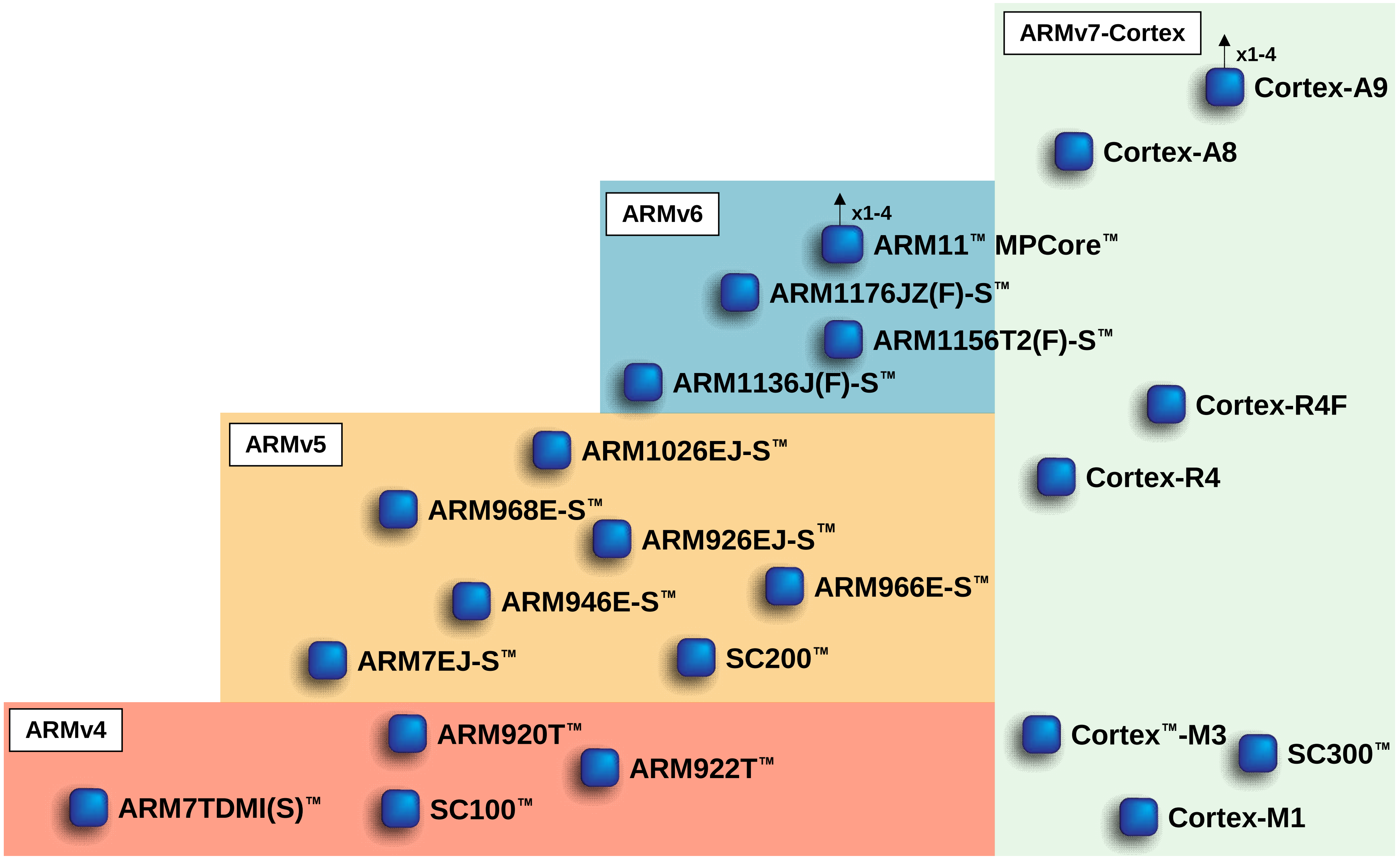
Data Path and Pipelines

AMBA/GPU

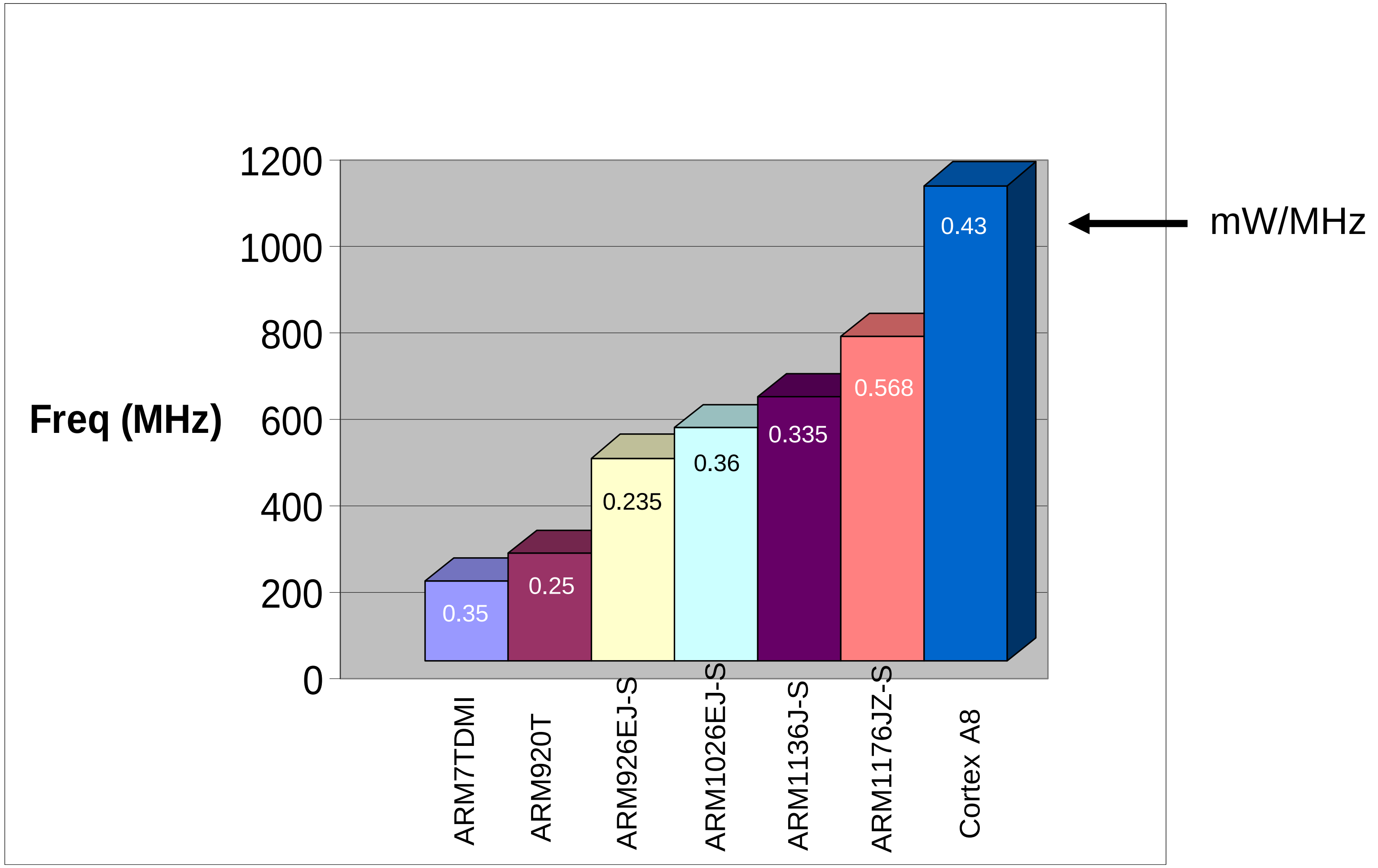
IEM

Development Tools

Architecture Versions

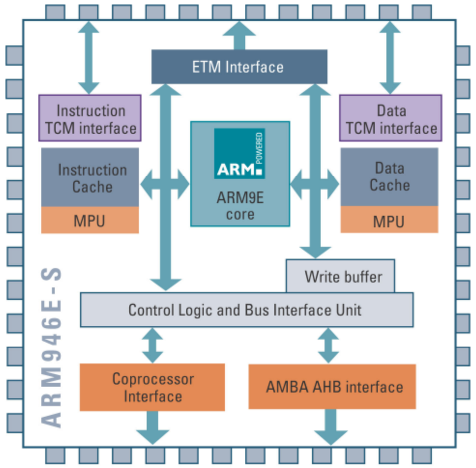


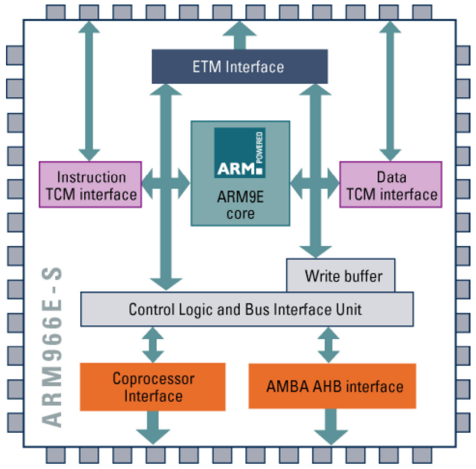
Relative Performance*



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

ARM9E Processor Core





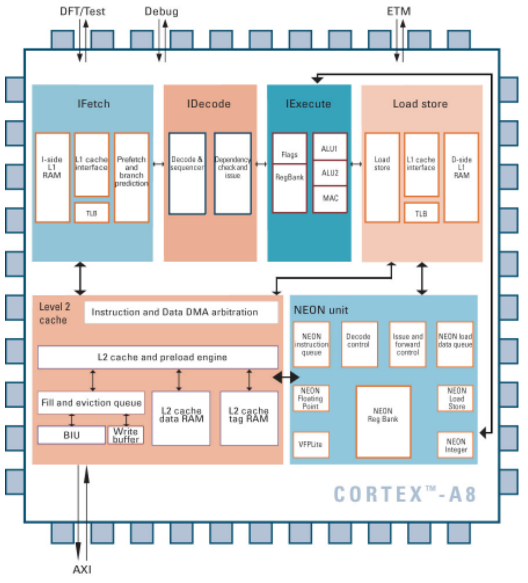
- **ARM9E is based on the ARM9TDMI core**
- **Core implementation differences**
  - Architecture V5TE support
  - Single cycle 32x16 multiplier implementation
  - EmbeddedICE Logic RT
- **ARM926EJ-S / ARM946E-S**
  - Configurable Instruction and Data caches
  - Instruction and Data TCM Interfaces
  - AHB bus interface
  - ARM926EJ-S has MMU
  - ARM946E-S has MPU
- **ARM966E-S**
  - Instruction and Data TCM Interfaces
  - No Cache or MPU/MMU

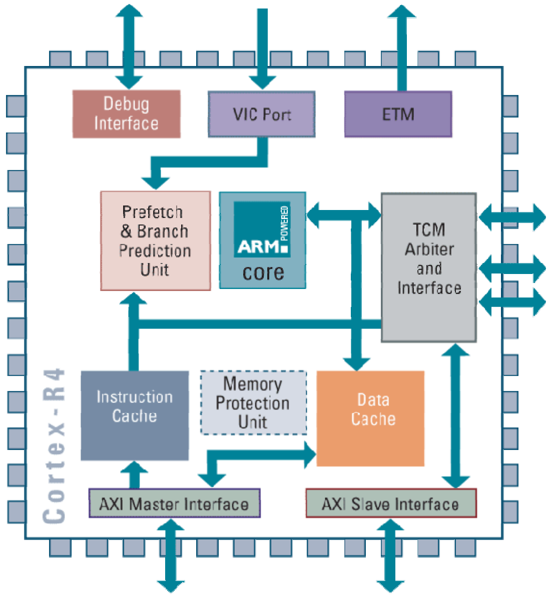
Cortex family

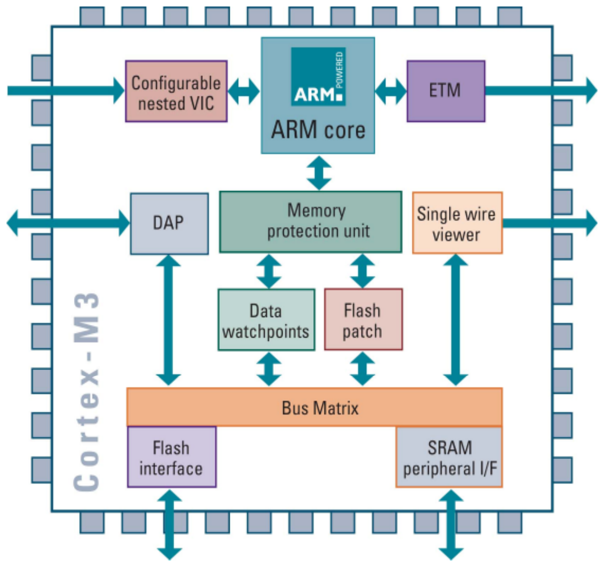
Cortex-A8
- Architecture v7A
- MMU
- AXI
- VFP & NEON support

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

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





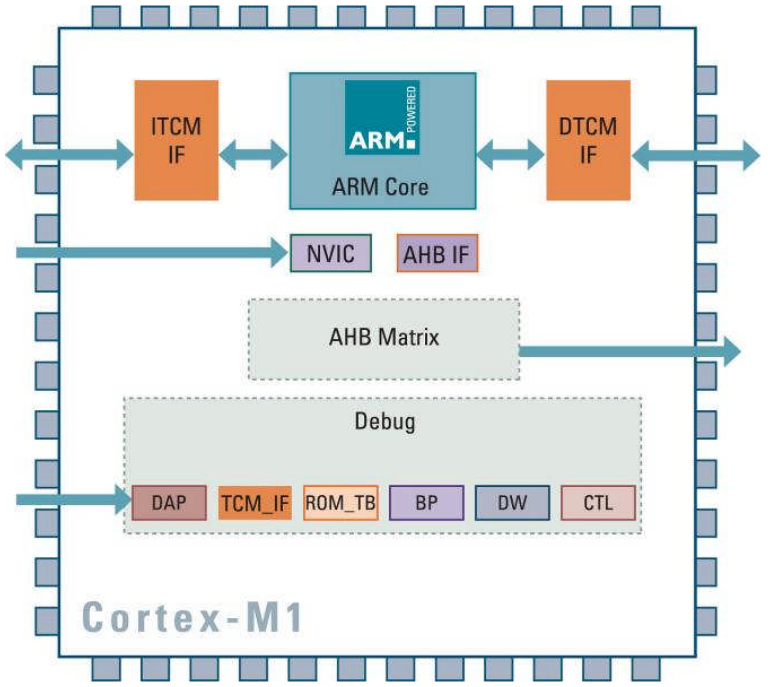


ARM Cortex-M1 Processor

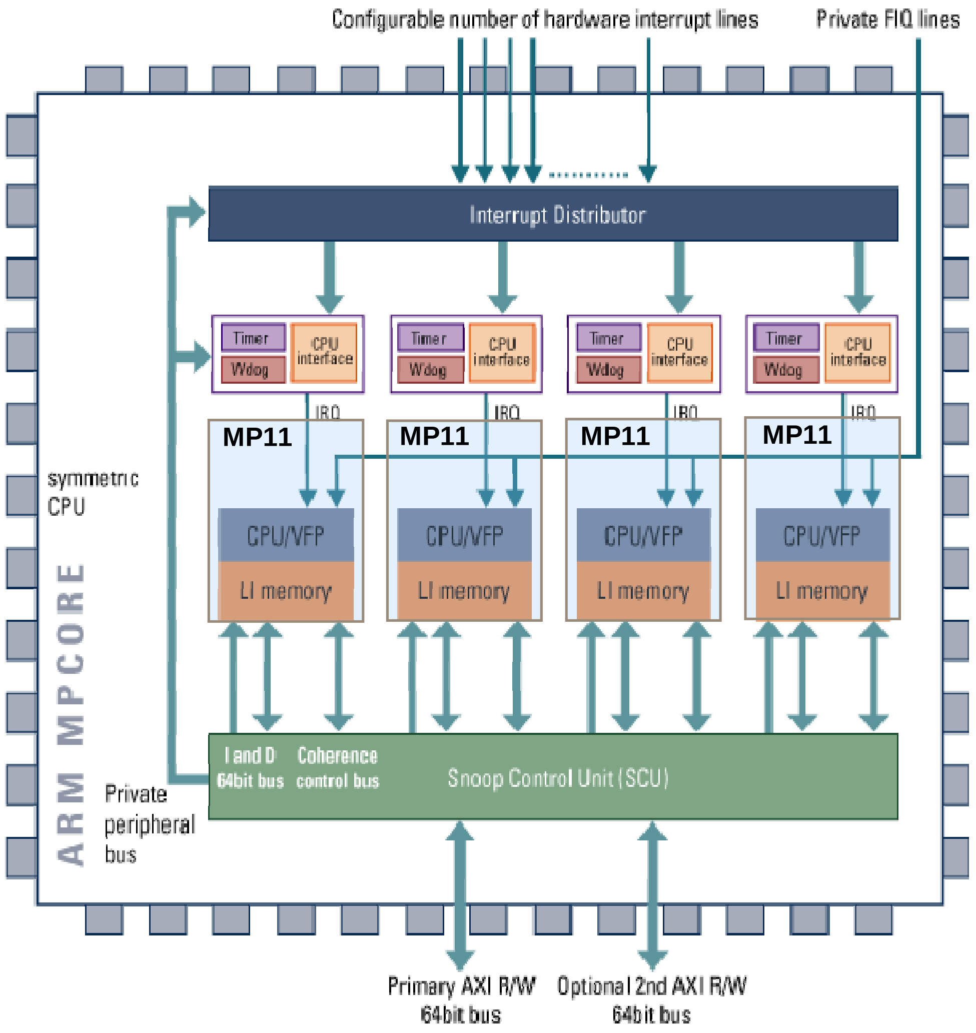
- High frequency, low area microcontroller processor for FPGA
  - Between 70MHz – 200MHz (depending on FPGA device)
  - Occupies less than 15% area on the most popular FPGA device sizes
  - Cortex-M1 upwards compatible with Cortex family on ASIC/ASSP/MCU
  - Performance will continue to increase as FPGA technology progresses

- Optimized for synthesis on multiple FPGA types
  - Xilinx (e.g. Spartan-3, Virtex-5)
  - Altera (e.g. Cyclone-II, Stratix-III)
  - Actel (M1 ProASIC3 and M1 Fusion)

XILINX® ALTERA® Actel Lattice Semiconductor Corporation

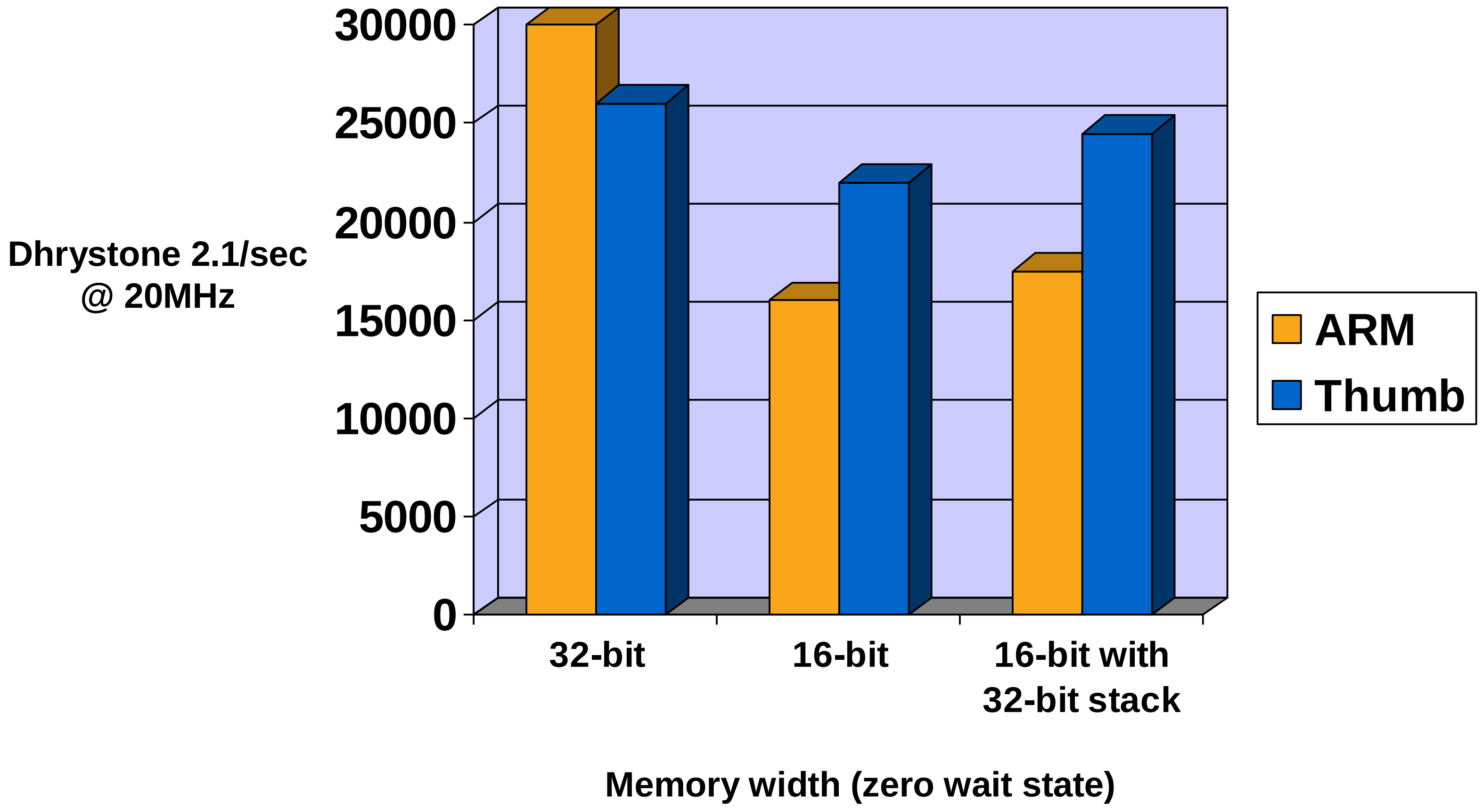


ARM11 MPCore



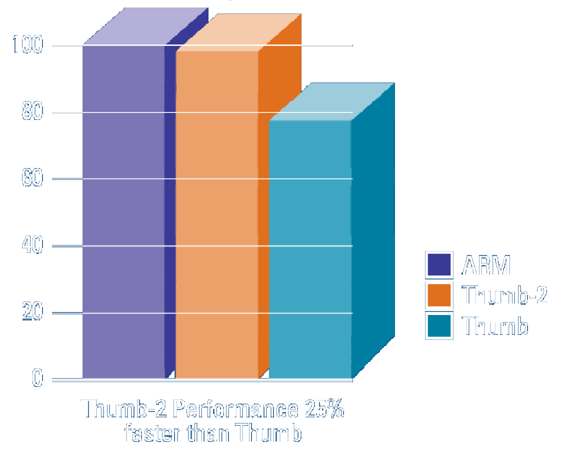
- **Synthesizable**
- **1 – 4 MP11 processors**
  - With associated timers & interfaces
  - With or without VFP11 coprocessor
- **ARM v6K compliant**
- **Configurable interrupt inputs**
  - 0 – 224 in steps of 32
  - Programmable distribution to MP11s
- **Support for SMP or AMP**
- **MESI-based cache coherency**
- **1 or 2 AXI interfaces to level 2**
  - 64-bit data buses
- **IEM Ready**
- **Program Trace using ETMs**

ARM and Thumb Performance

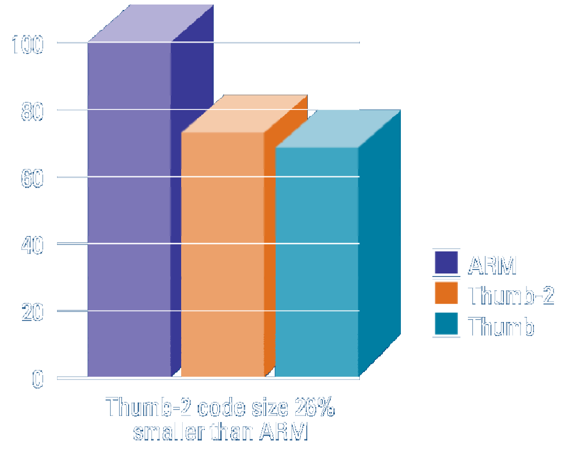


Thumb-2 Instruction Set

EEMBC Analysis - Performance



EEMBC Analysis – Code Size

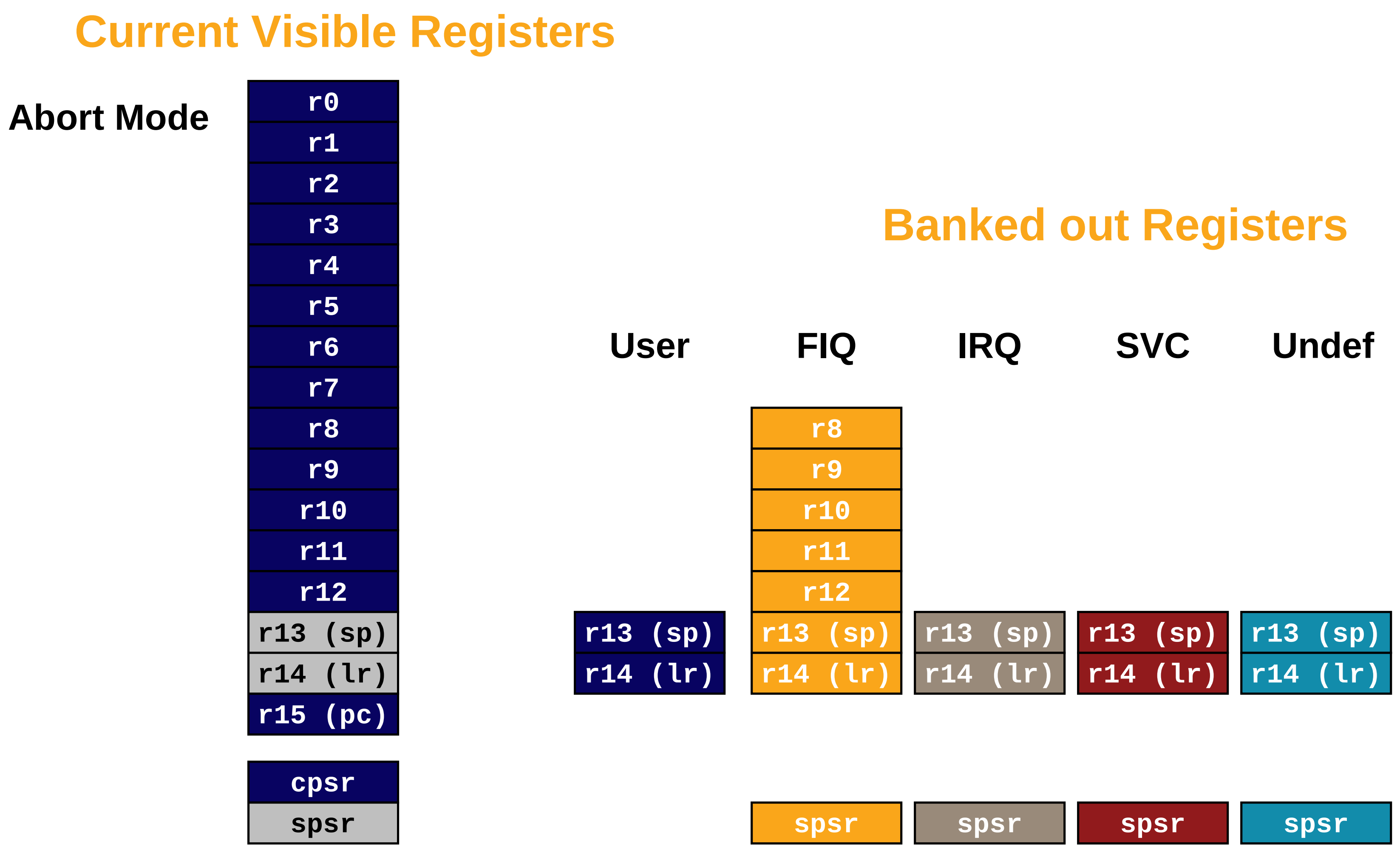


- Second generation of the Thumb architecture
  - Blended 16-bit and 32-bit instruction set
  - 25% faster than Thumb
  - 30% smaller than ARM
- Increases performance but maintains code density
- Maximizes cache and tightly coupled memory usage

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

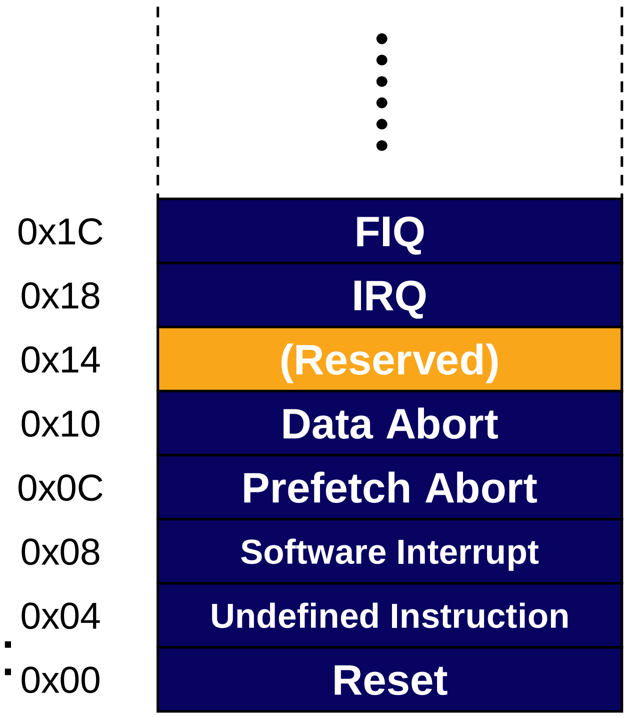
The ARM Register Set



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>

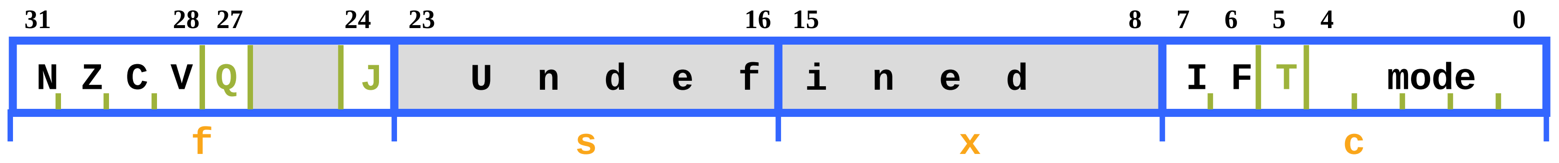
This can only be done in ARM state.



Vector Table

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

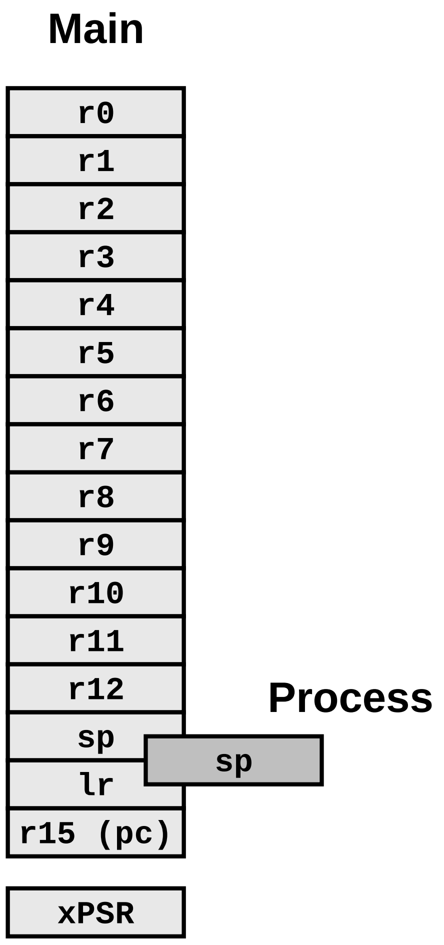
Program Status Registers



- 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

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



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
```

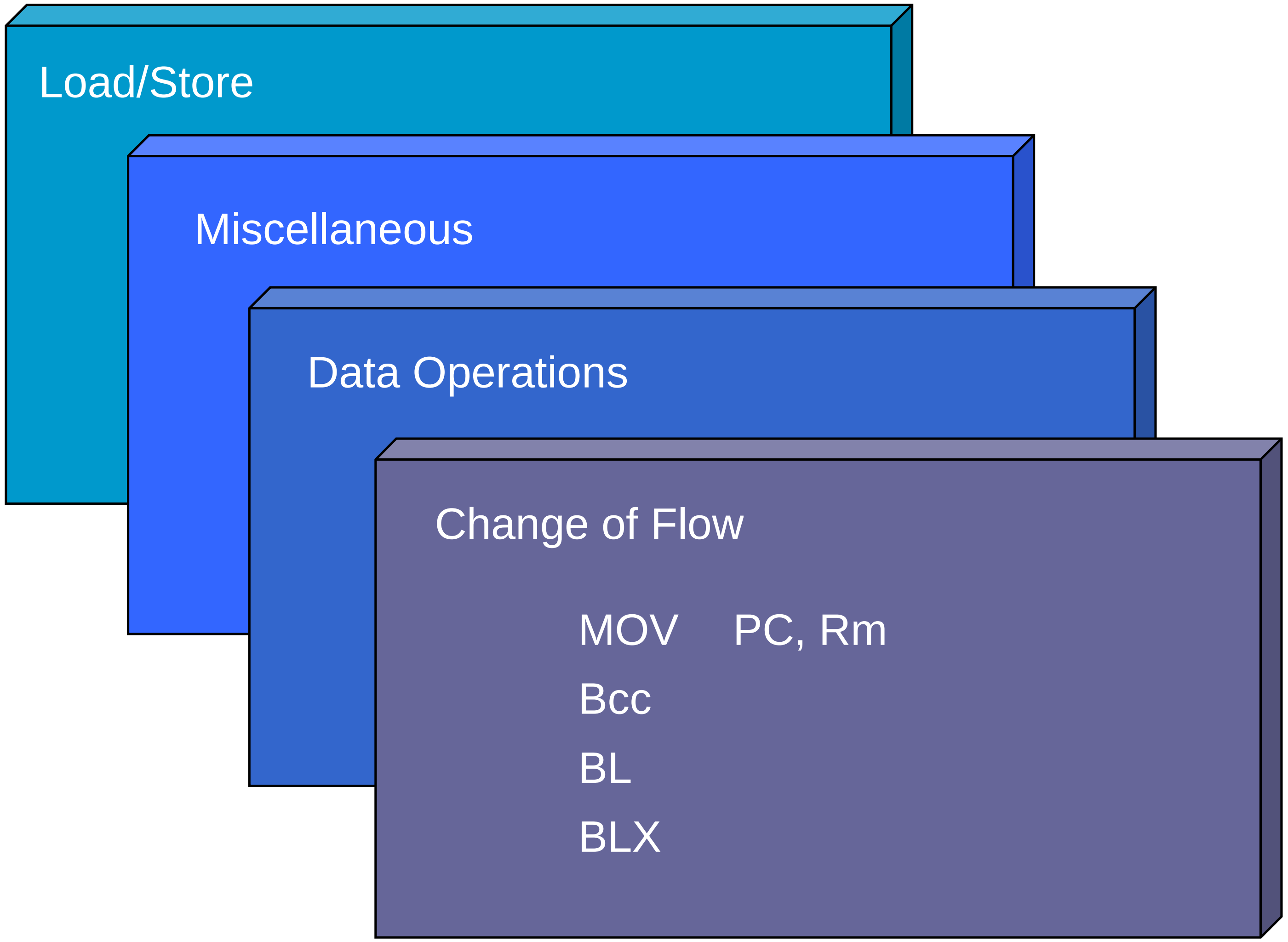
- By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using “S”. CMP does not need “S”.

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

decrement r1 and set flags

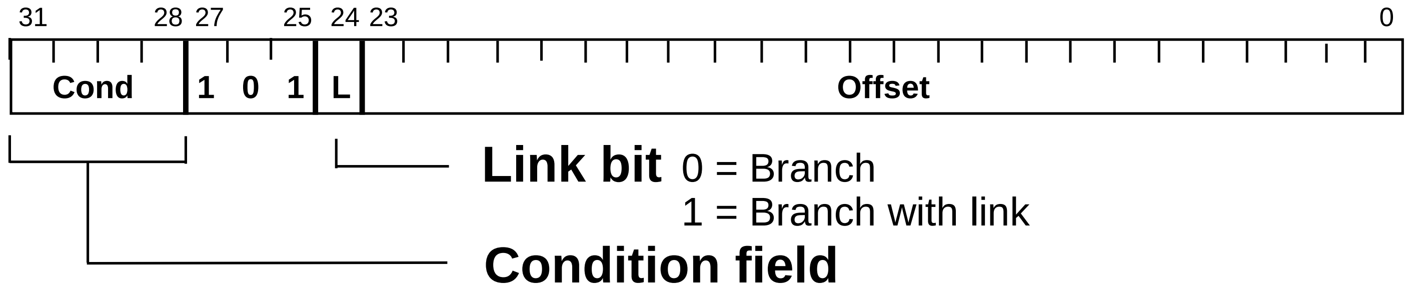
if Z flag clear then branch

Classes of Instructions (v4T)



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`

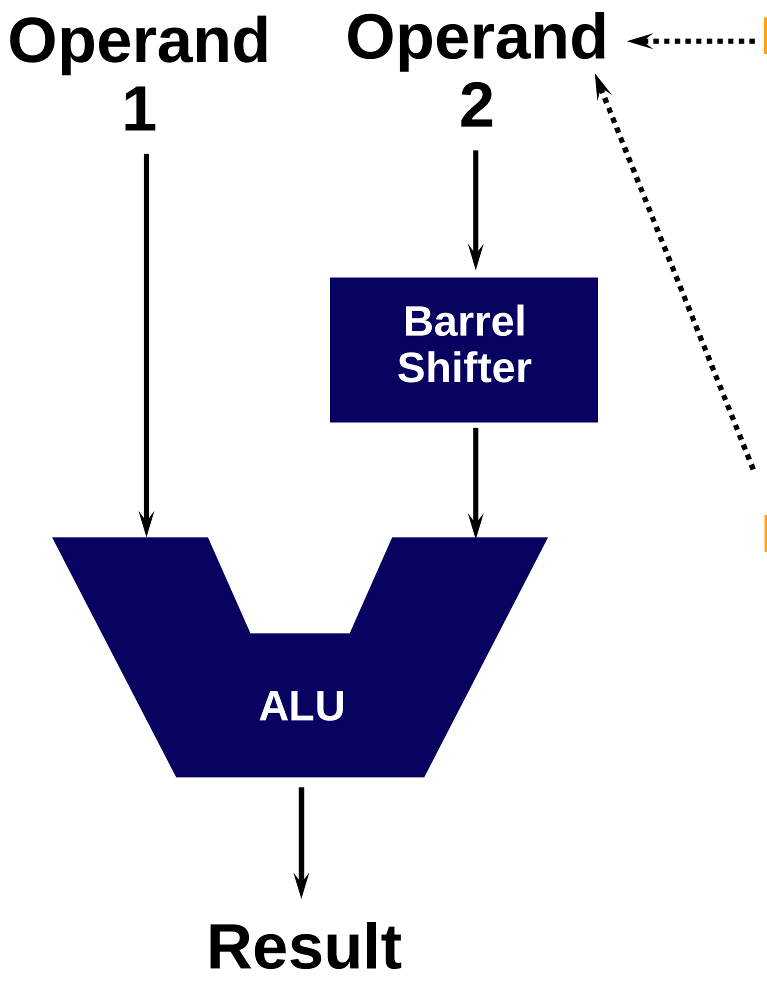
- These instructions only work on registers, NOT memory.

- 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

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

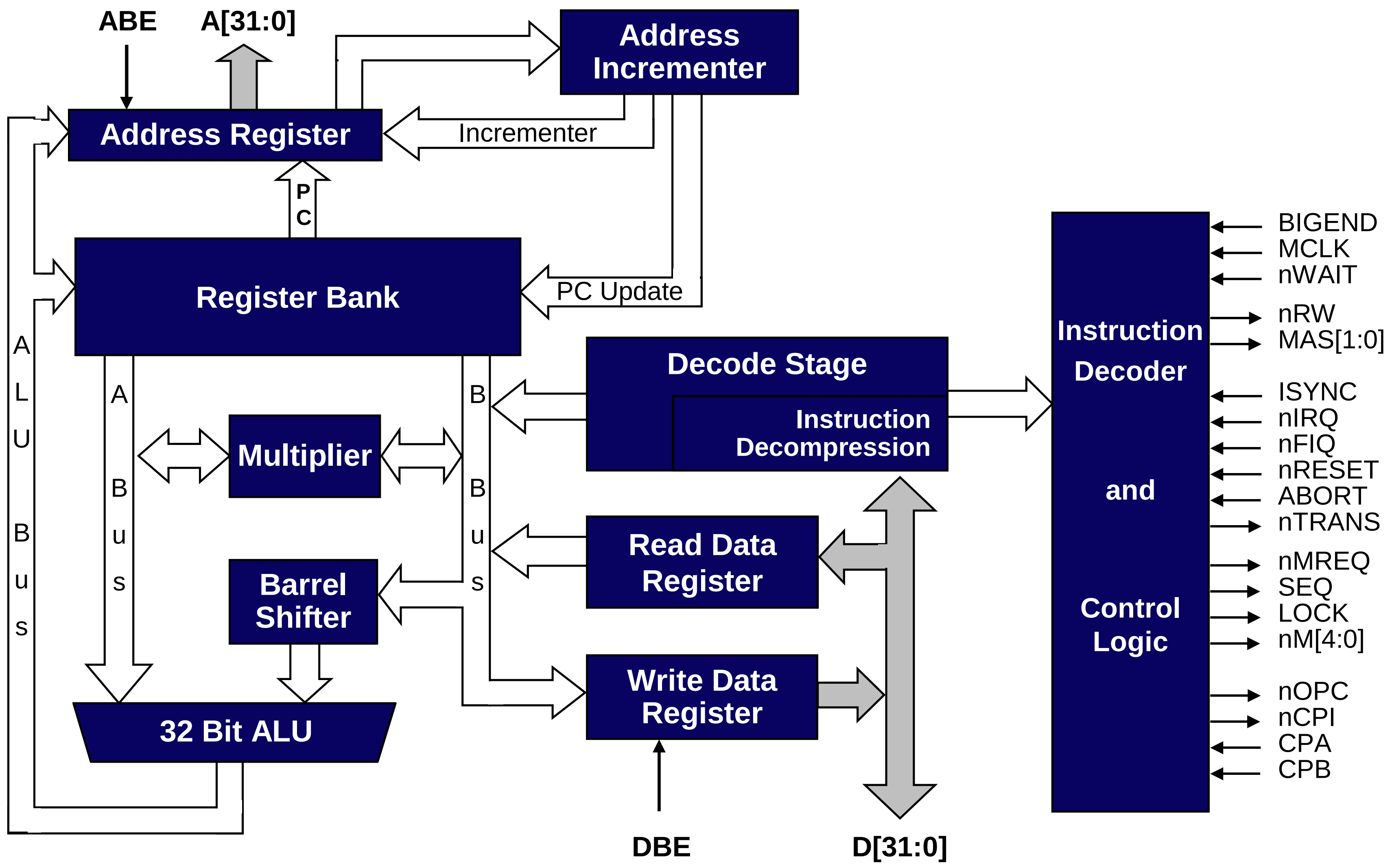
- Data Path and Pipelines

AMBA/GPU

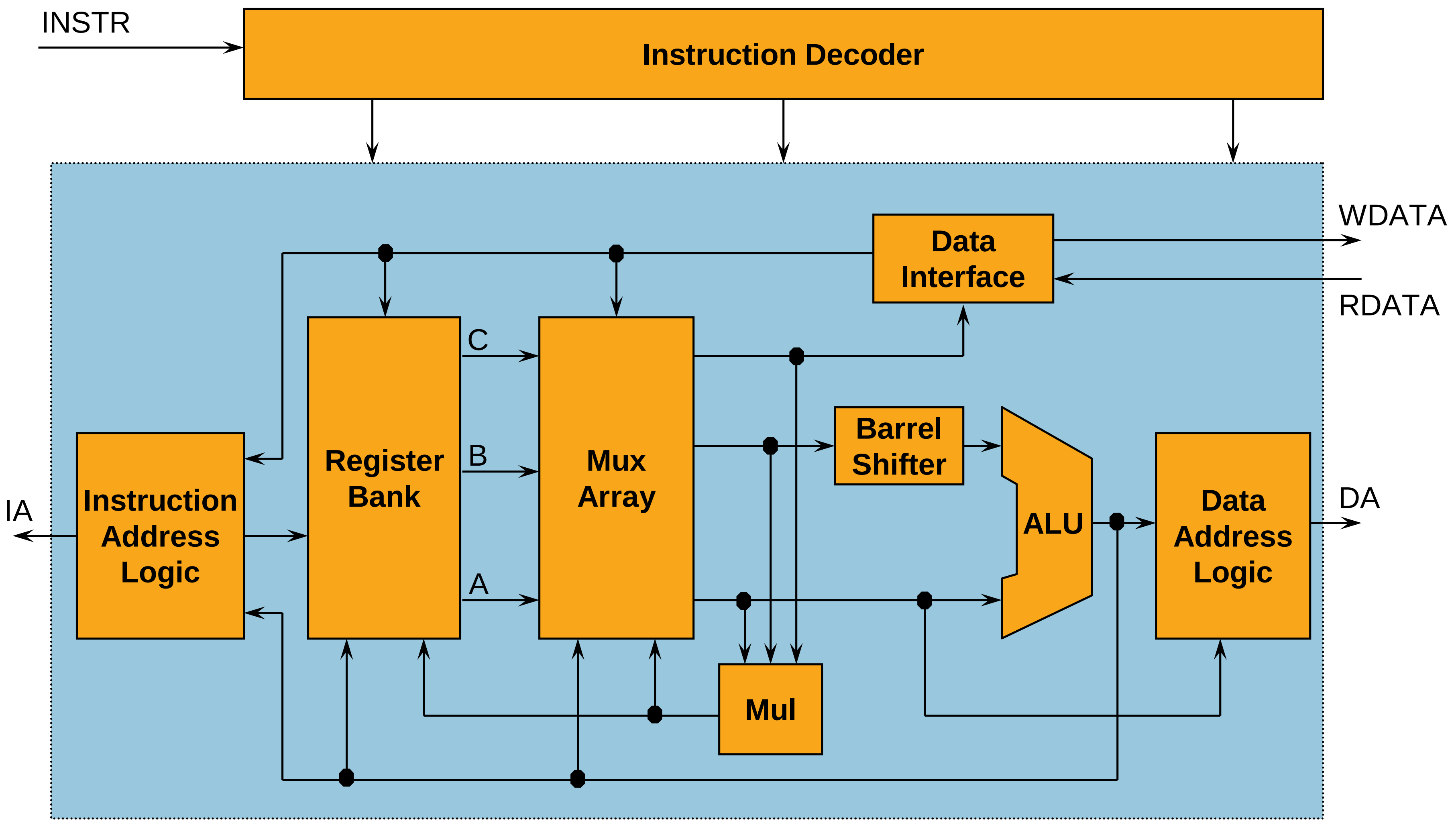
IEM

Development Tools

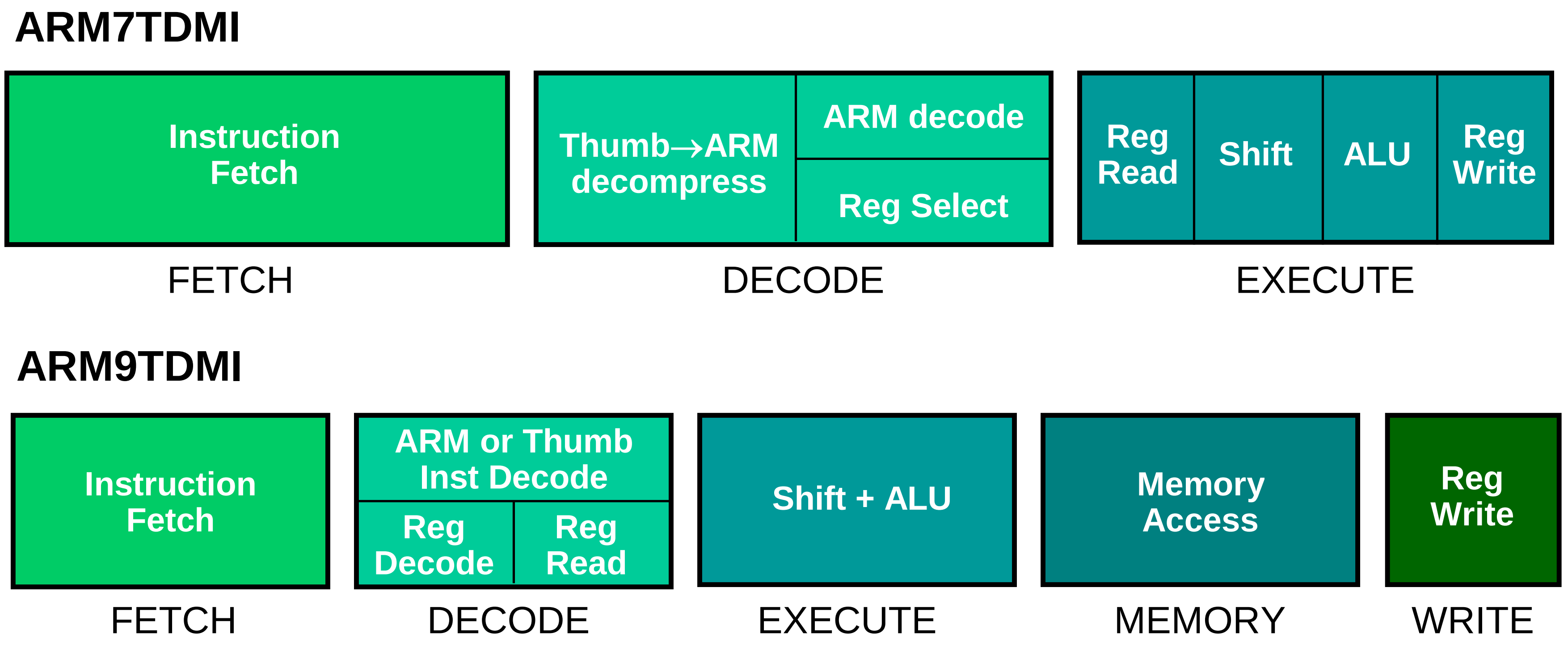
The ARM7TDM Core



ARM9E-S Datapath

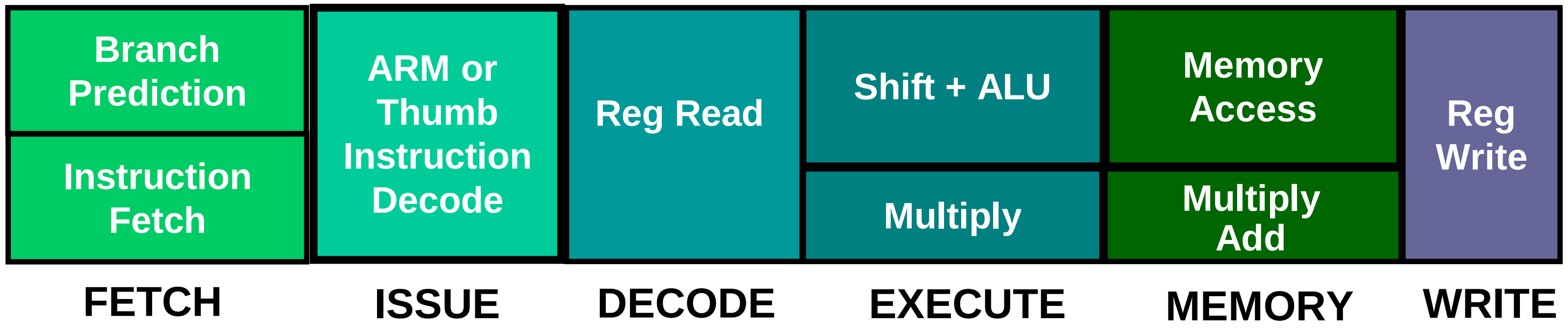


Pipeline changes for ARM9TDMI

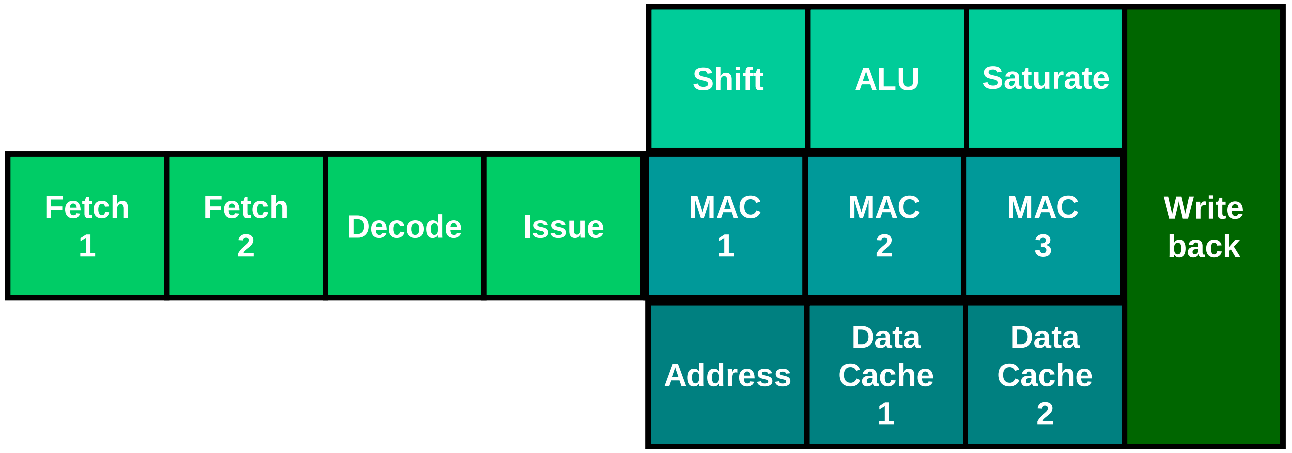


ARM10 vs. ARM11 Pipelines

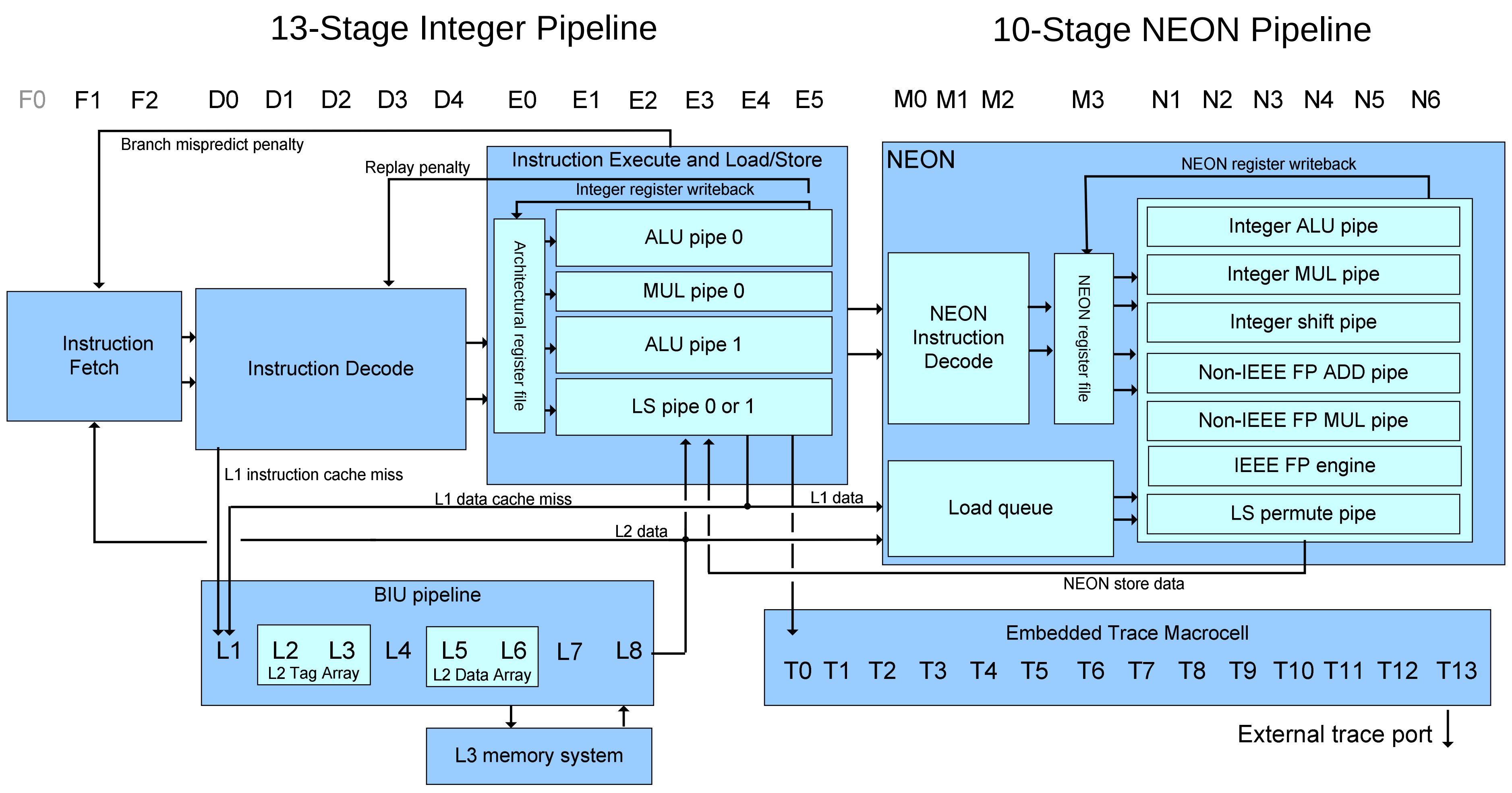
ARM10



ARM11



Full Cortex-A8 Pipeline Diagram



Agenda

Introduction to ARM Ltd

ARM Architecture/Programmers Model

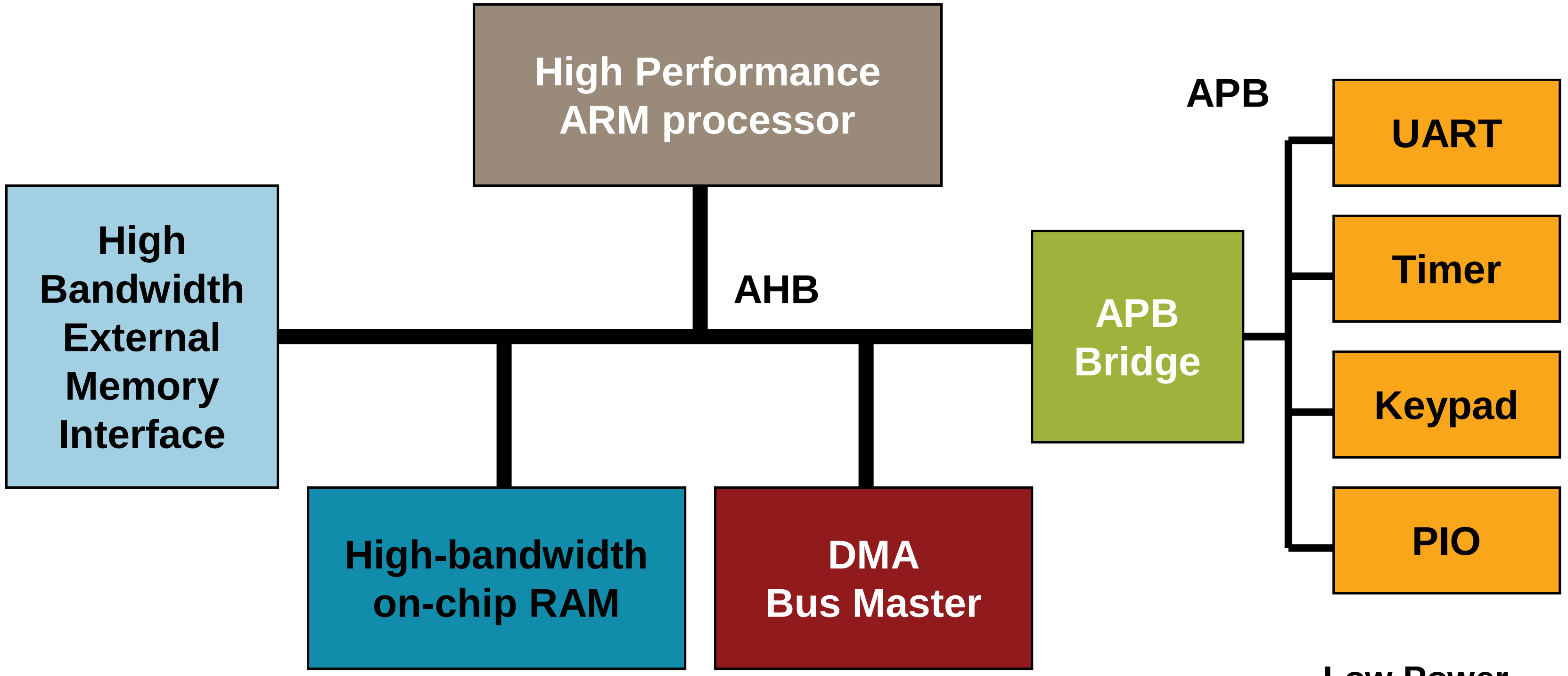
Data Path and Pipelines

- AMBA/GPU

IEM

Development Tools

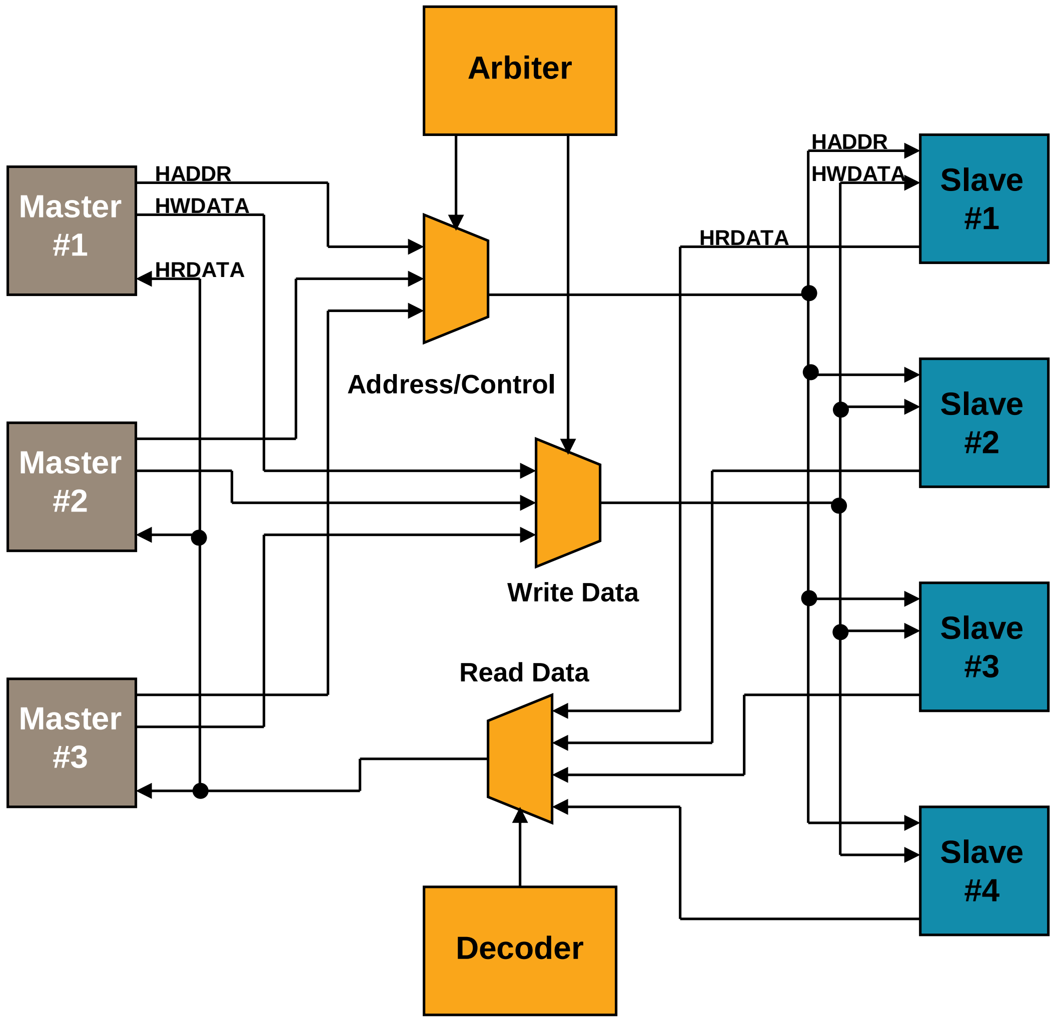
An Example AMBA System


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

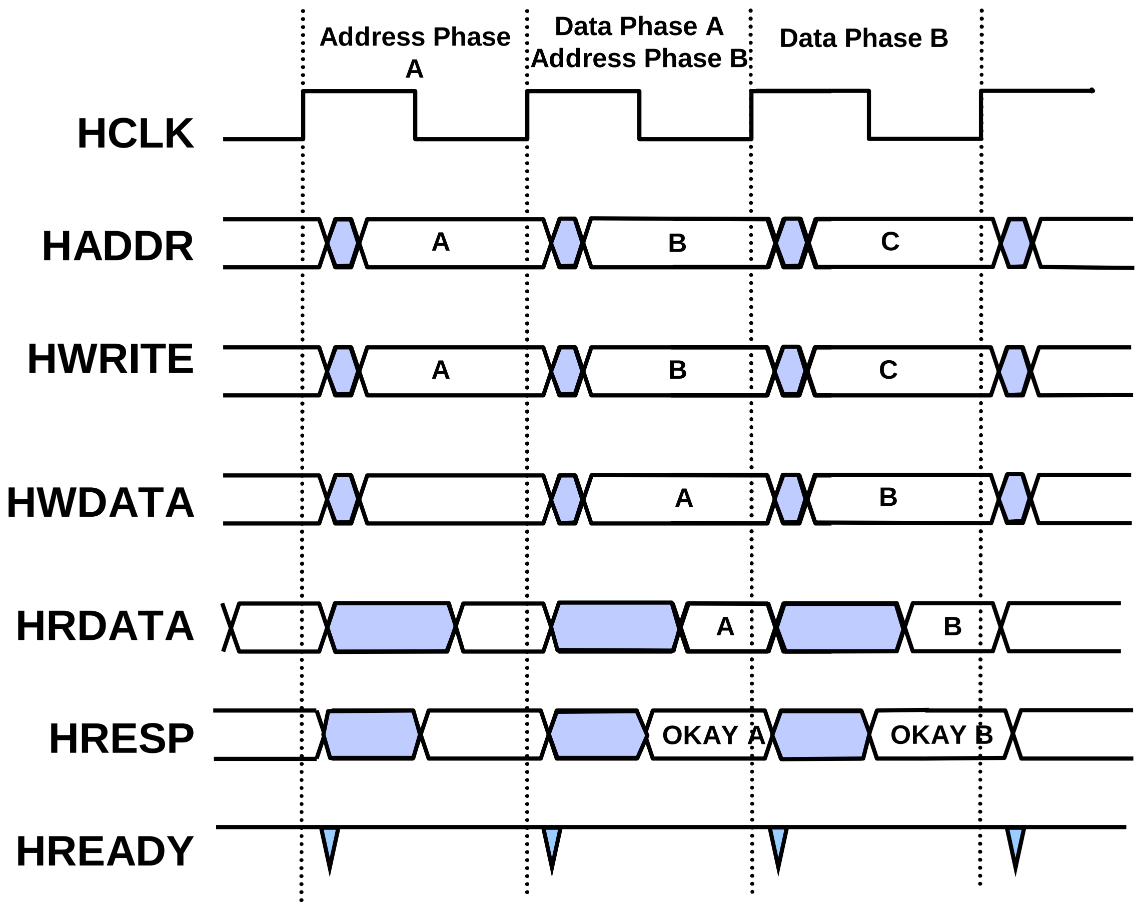
High Performance
Pipelined
Burst Support
Multiple Bus Masters

Low Power
Non-pipelined
Simple Interface

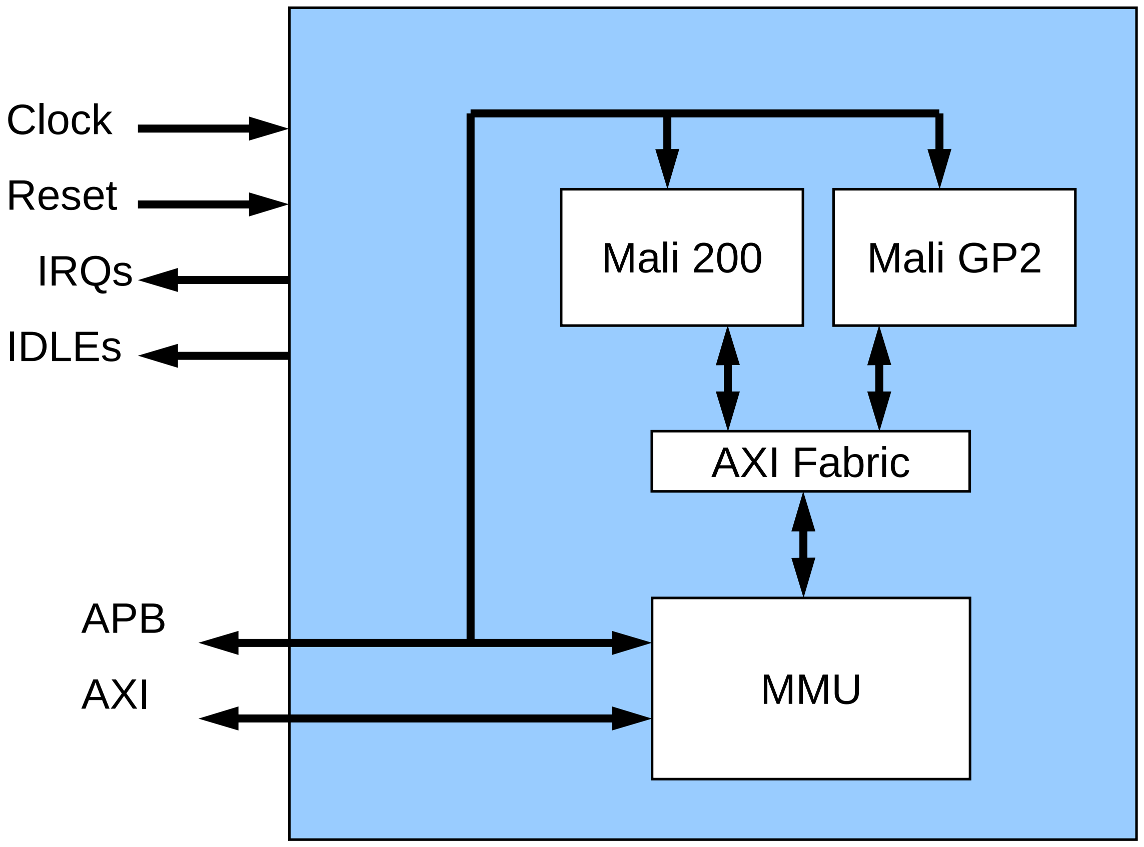
AHB Structure



AHB basic signal timing

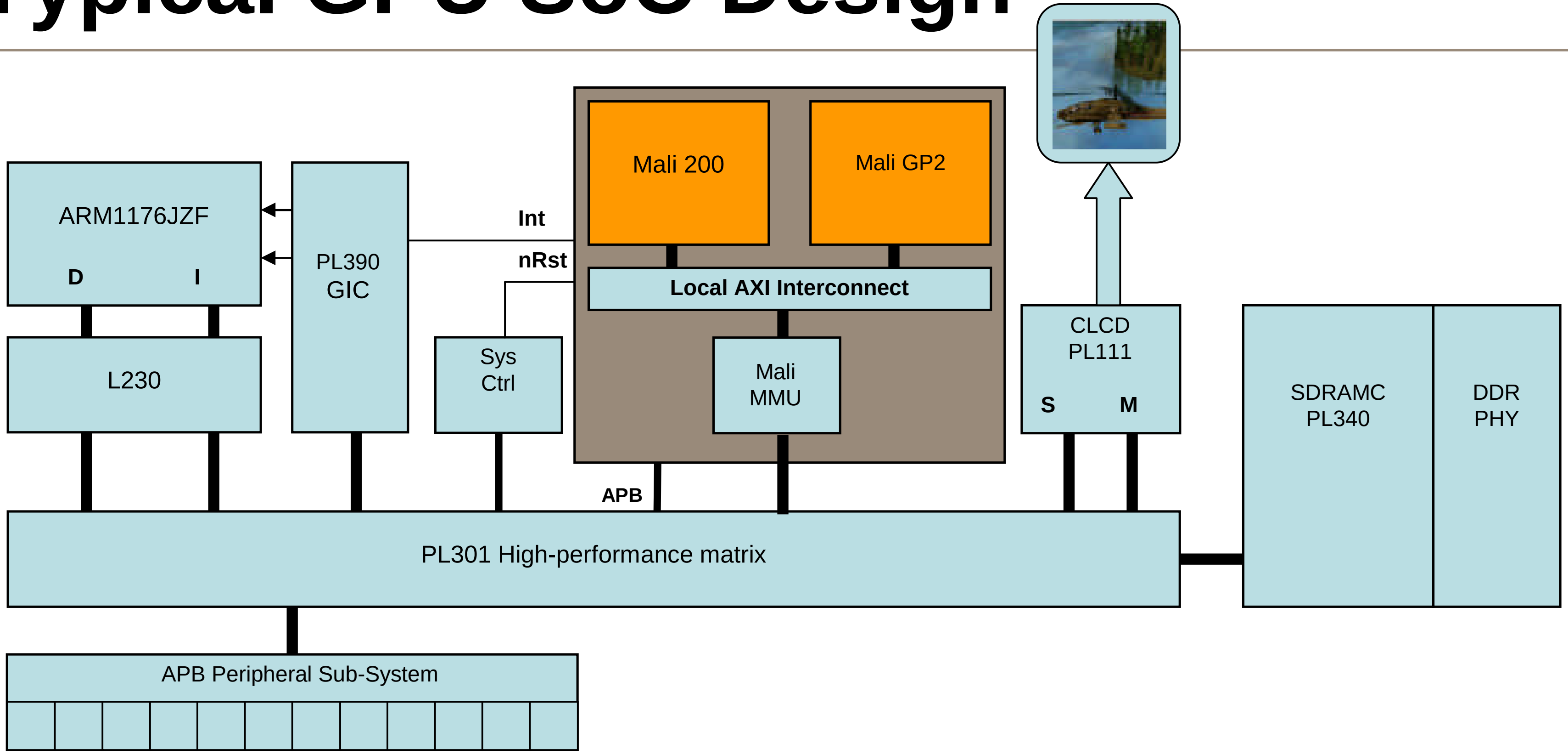


Mali200 + GP2 SoC Integration



- Shipped as synthesizable Verilog
- Mali 200 + GP2 requires a single instant in the SoC, with a small number of connections to be made.
- IDLES can be used for gating the Mali200 and GP2 core clock

Typical GPU SoC Design



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

Agenda

Introduction to ARM Ltd

ARM Architecture/Processors/Programmers Model

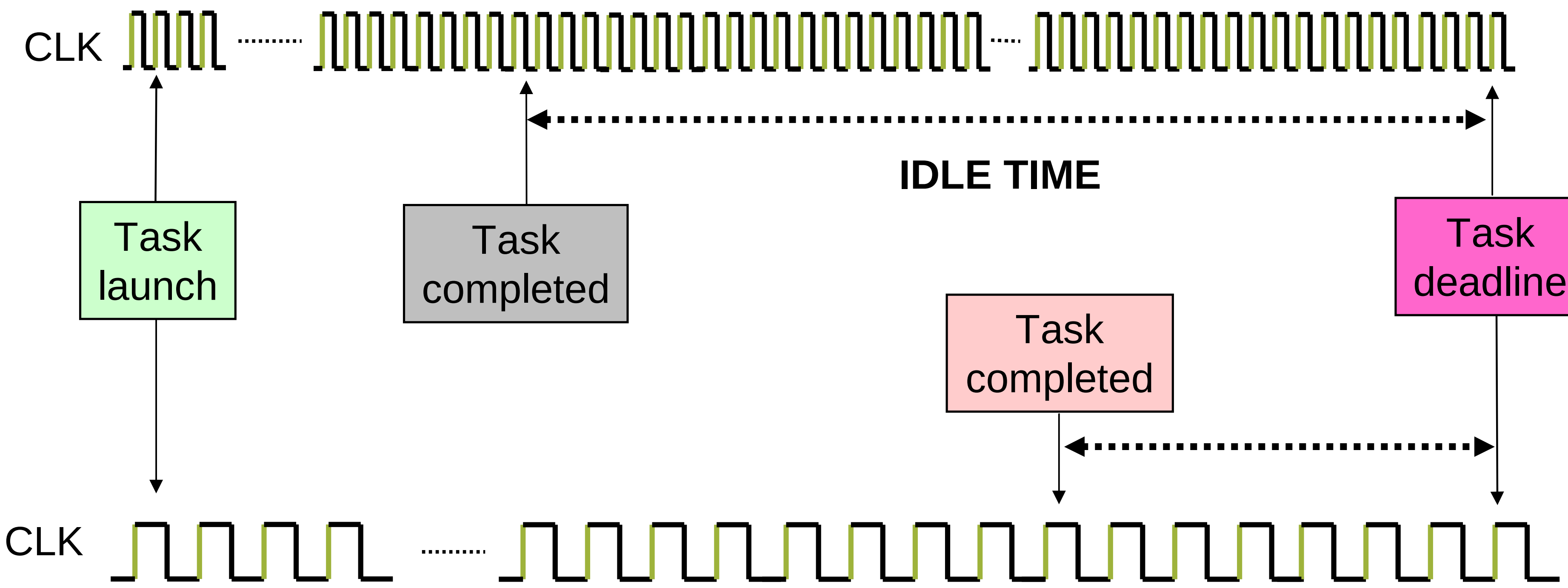
Data Path and Pipelines

AMBA/GPU

- IEM

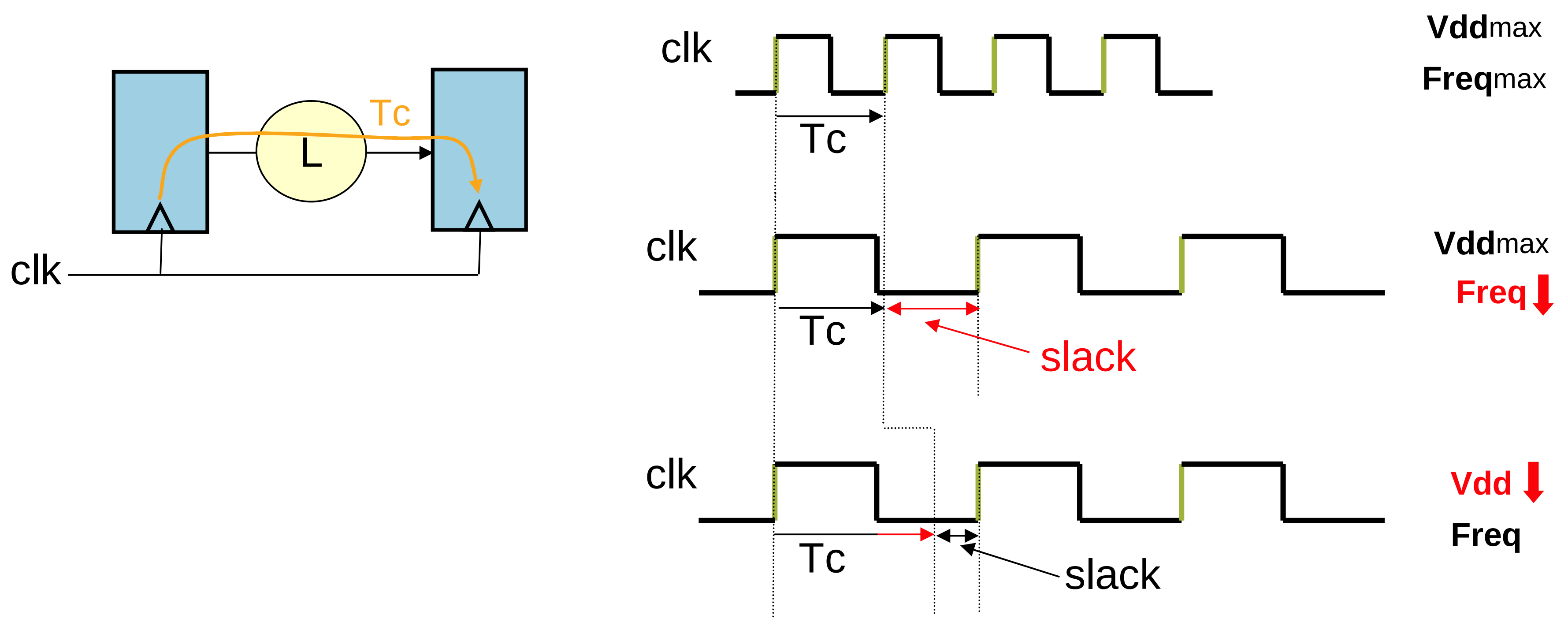
Development Tools

Clocking



- Systems are usually designed for maximum speed but this might only be utilized for certain tasks

Voltage

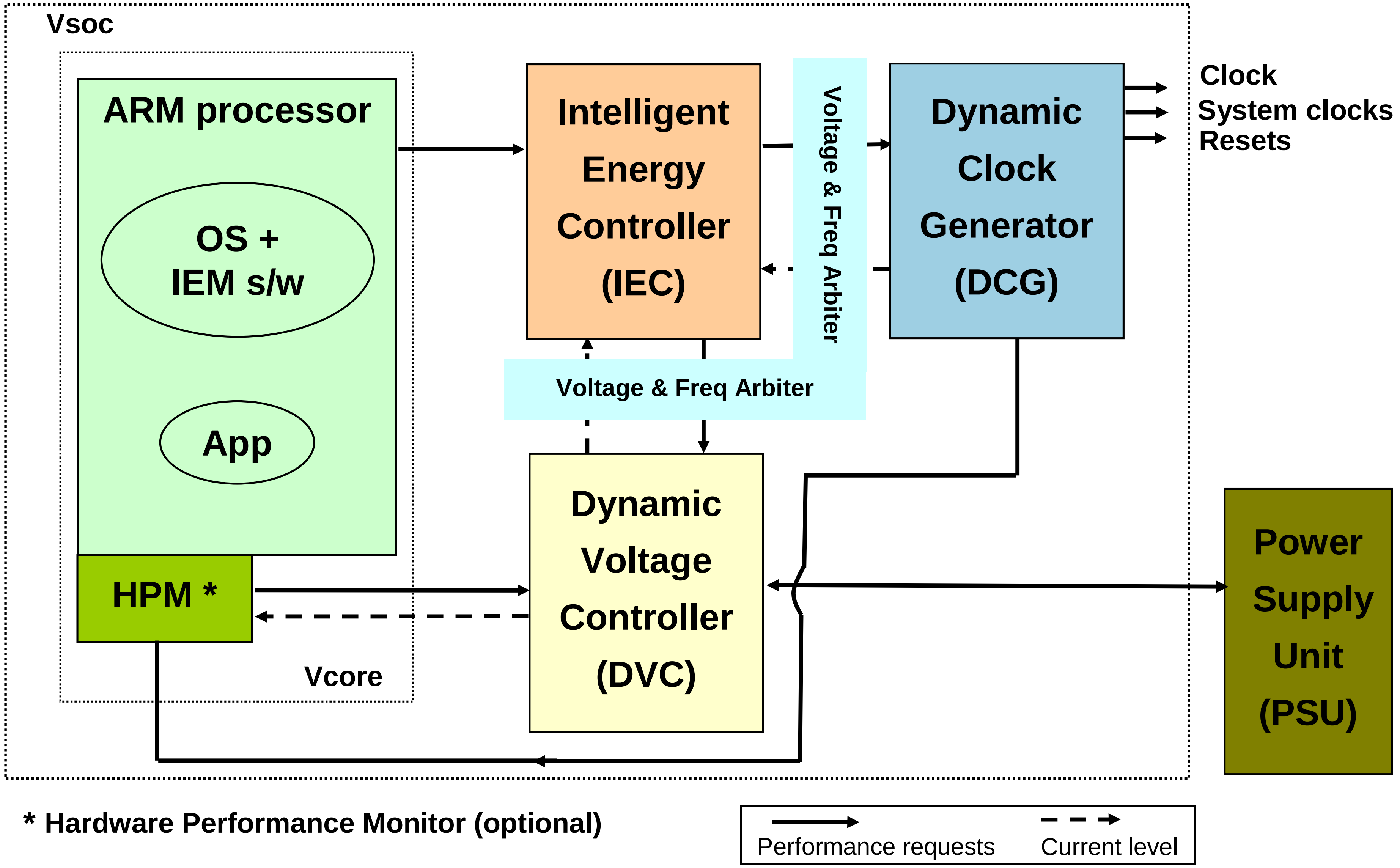


- Lowering clock frequency introduces more slack into register-to-register timing
- Slack can be utilized by lower voltage for system causing Tc to increase but energy usage to decrease

IEM Software

- IEM-enabled OS
- Analyses historical performance required for tasks
- Policies and algorithms
- Performance targets forward to IEM hardware as percentage of maximum

IEM Infrastructure



* Hardware Performance Monitor (optional)

IEM

- **Intelligent Energy Manager works by changing voltage and clock rate to match the performance required to complete the task**
- **Can yield a quadratic saving in energy usage for a given task**
  - Better than just clock gating/scaling
    - Saving in leakage current from voltage reduction

$$P = Cv_{dd}^2 f + v_{dd} I_{leak} \qquad E = \int P\,dt$$

where $Cv_{dd}^2 f$ is the dynamic component due to switching

where $v_{dd} I_{leak}$ is the static component due to leakage

where E = ENERGY

Agenda

Introduction to ARM Ltd

ARM Architecture/Programmers Model

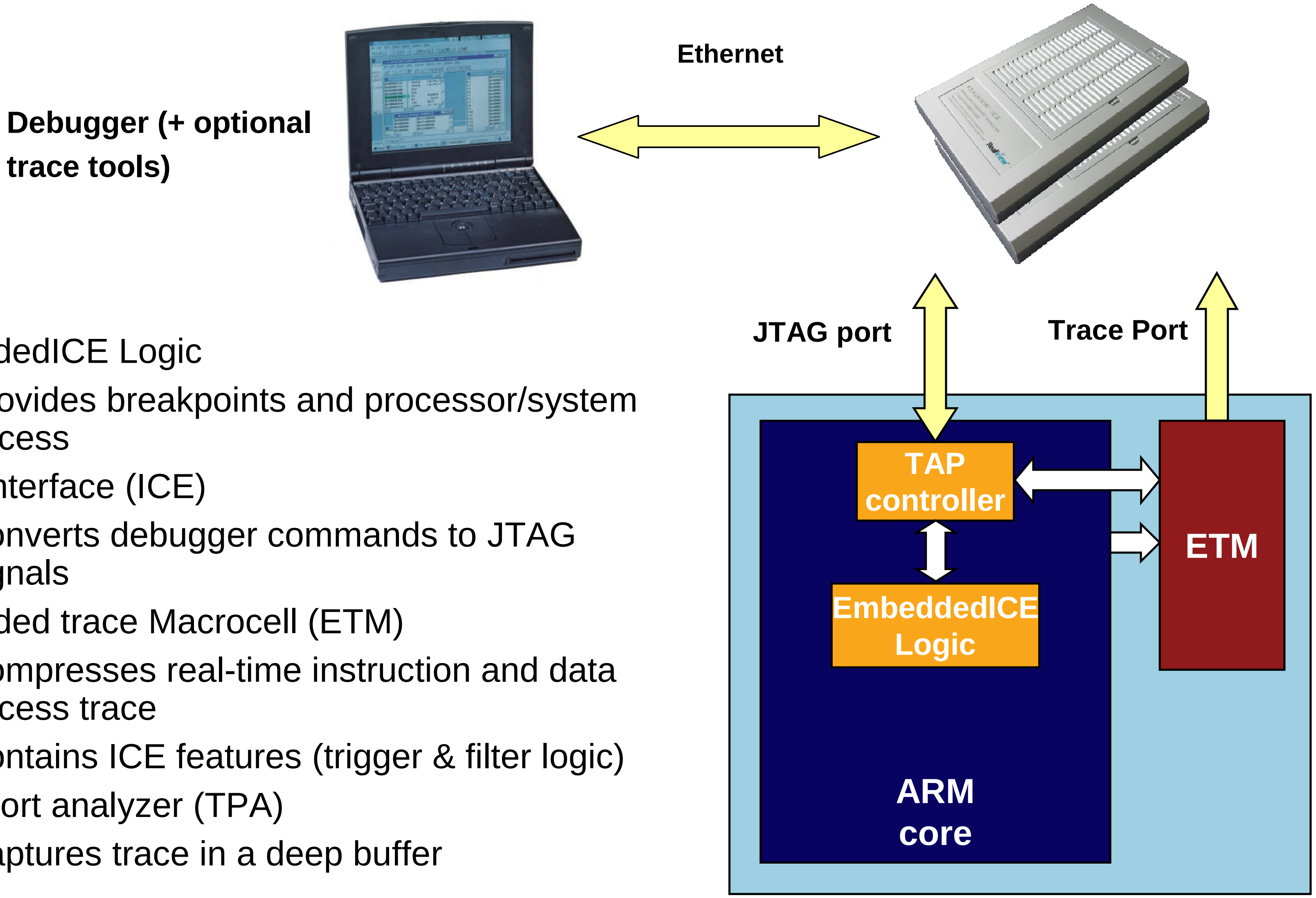
Data Path and Pipelines

AMBA/GPU

IEM

- Development Tools

ARM Debug Architecture



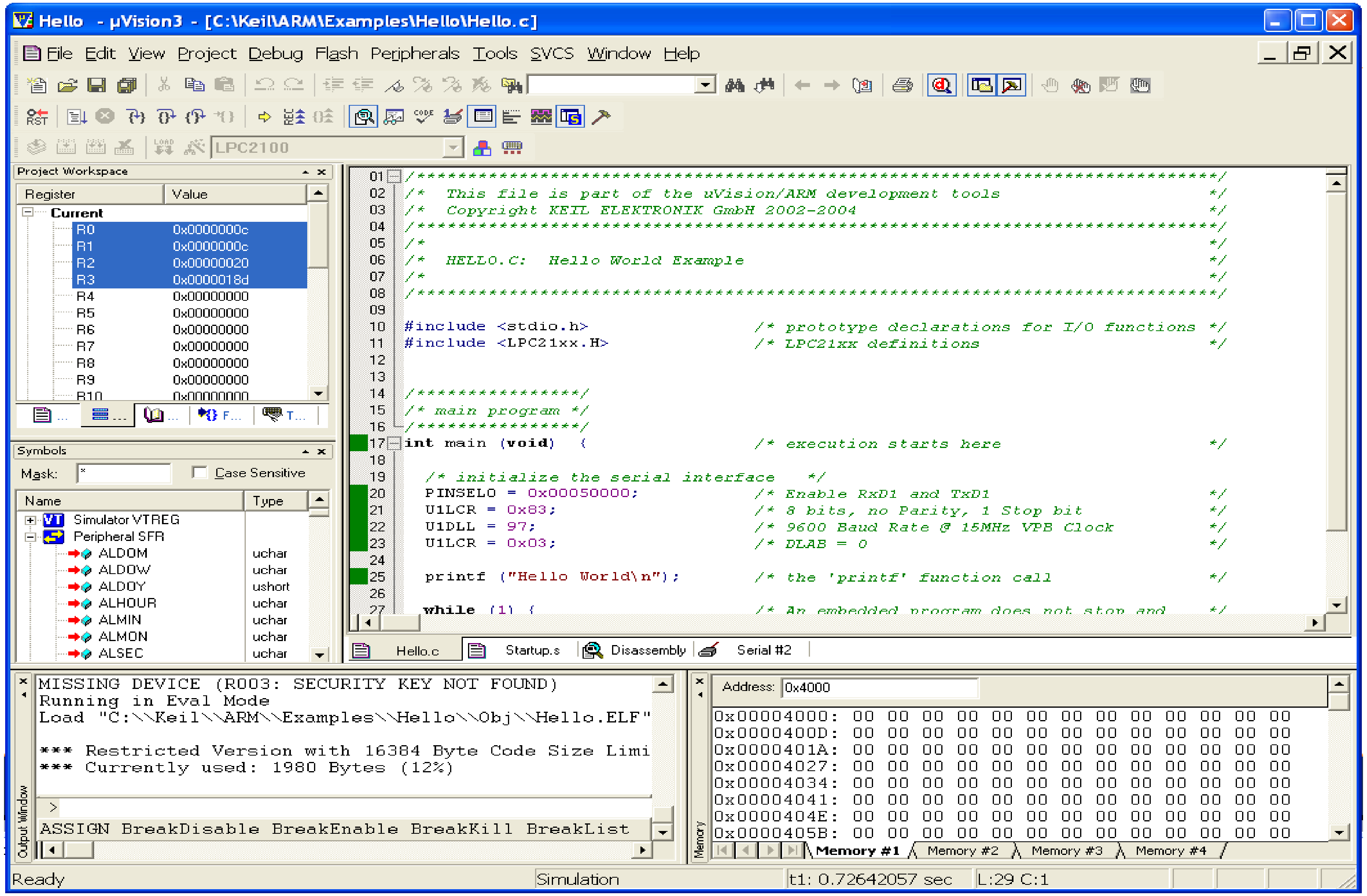
- EmbeddedICE Logic
  - Provides breakpoints and processor/system access
- JTAG interface (ICE)
  - Converts debugger commands to JTAG signals
- Embedded trace Macrocell (ETM)
  - Compresses real-time instruction and data access trace
  - Contains ICE features (trigger & filter logic)
- Trace port analyzer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

Keil Development Tools for ARM

KEIL™ An ARM® Company

- 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 + 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



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    */
```

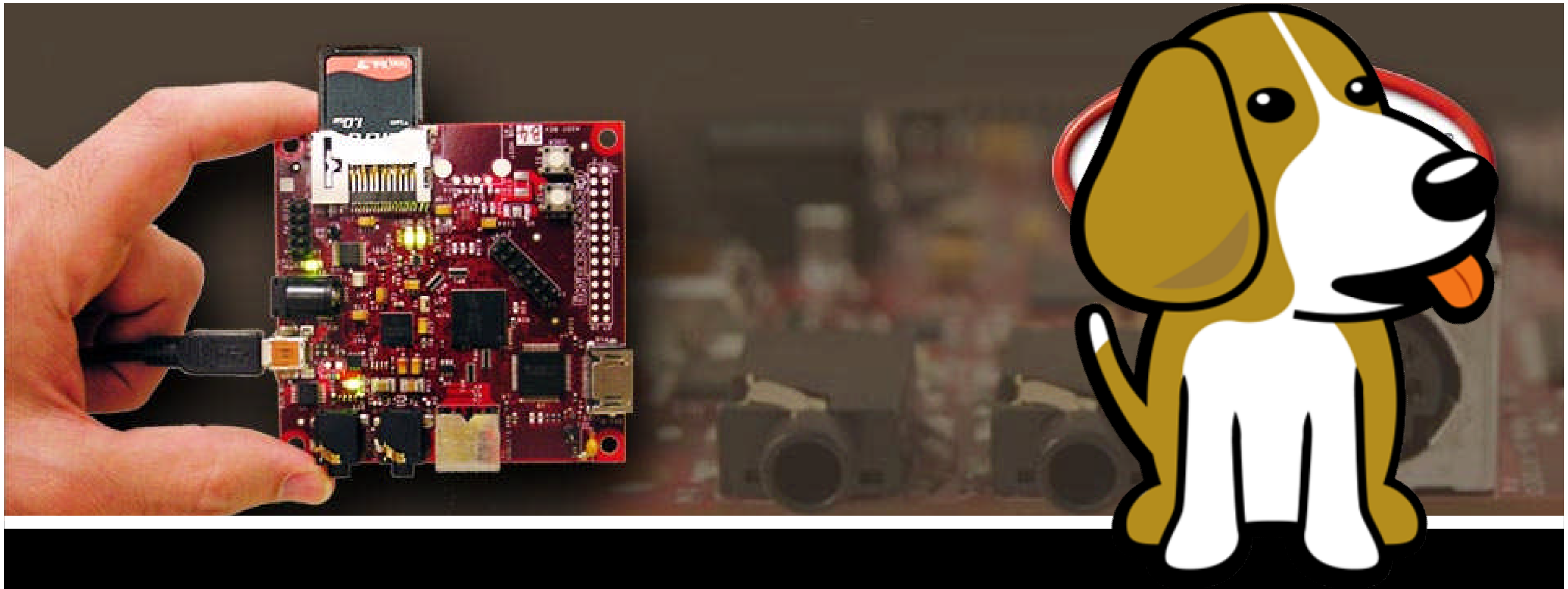
```
MISSING DEVICE (R003: SECURITY KEY NOT FOUND)
Running in Eval Mode
Load "C:\\Keil\\ARM\\Examples\\Hello\\Obj\\Hello.ELF"

*** 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
```

Ready | Simulation | t1: 0.72642057 sec | L:29 C:1

University Resources

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



Beagle Board

Targeting community development

$149

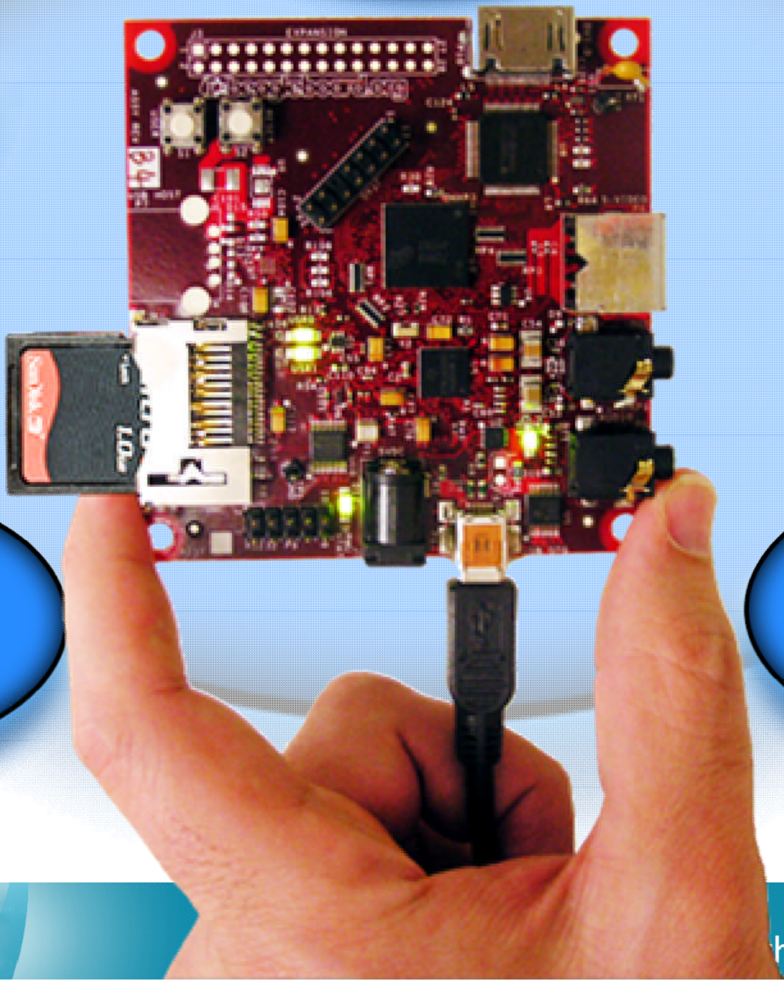
Personally affordable

> 1000 participants and growing

Wikis, blogs, promotion of community activity

Active & technical community

Freedom to innovate



Open access to hardware documentation

Instant access to >10 million lines of code

Opportunity to tinker and learn

Free software

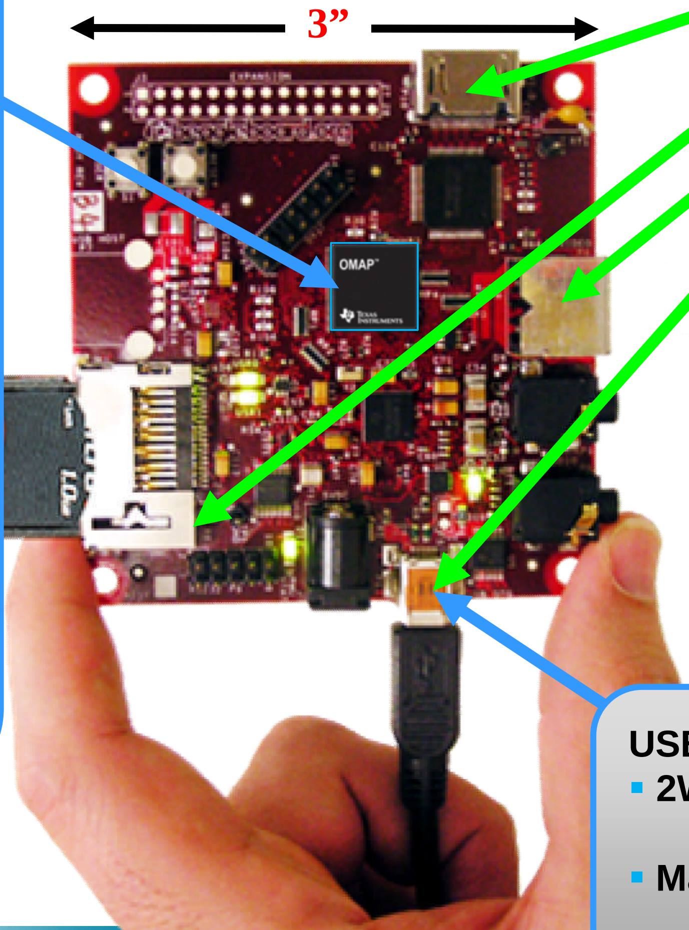
Fast, low power, flexible expansion

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

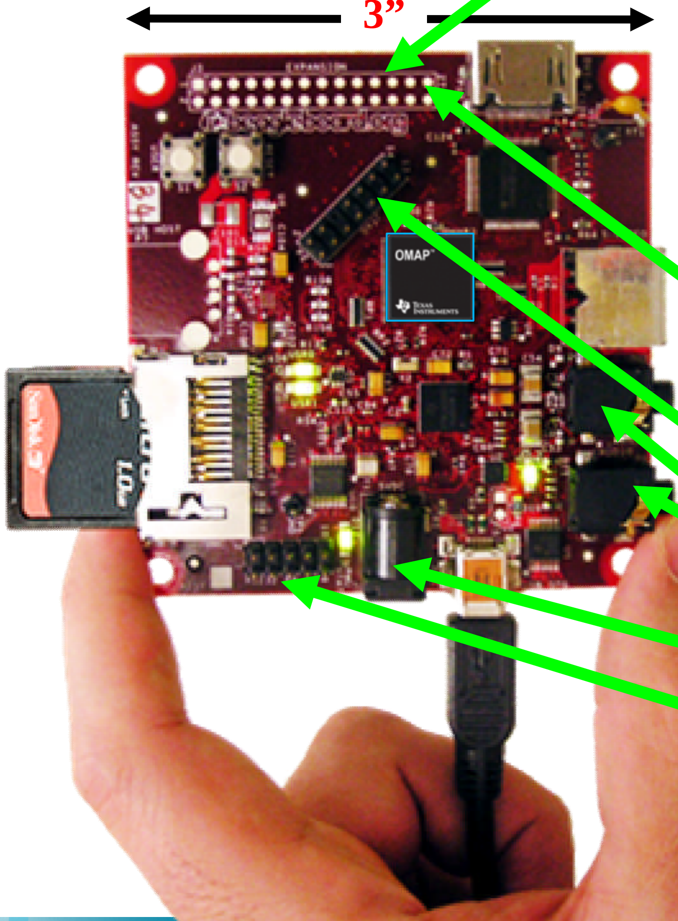
USB Powered

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

And more…

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

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



Peripheral I/O
- DVI-D video out
- SD/MMC+
- S-Video out
- USB HS OTG
- I^2C, I^2S, SPI, MMC/SD
- JTAG
- Stereo in/out
- Alternate power
- RS-232 serial

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