Objectives:

ARM Embedded Systems and ARM Processor Fundamentals:

- Evolution of Microcontroller and Microprocessor.
- The RISC design philosophy,
- ARM design philosophy, embedded system hardware-
- AMBA bus protocol.
- ARM core data flow model.
- Registers.
- CPSR-Processor modes.
- Banked registers.
- Pipeline- Characteristic.

ARM7 Fundamentals

1. All ARM instructions are 32-bit long & stored word aligned.
2. ARM processor like all RISC processors is a Load Store architecture, Von-Neuman Architecture (same program + data memory).
3. ARM has two special instructions types for transferring data in & data out of processor.

    - Load Instruction = Copy data from memory to registers in the core.
        - (Registers in the processor core <----Memory)
    - Store Instruction = Copy data from registers to memory
        - (Registers in processor core ----> Memory)

4. There are no data processing instructions that are directly manipulate data in memory (Hence Data processing is carried out only in registers).
5. ARM core is a 32-bit bit processor most instructions treat the registers ad holding signed or unsigned 32-bit value.
6. Data Types
7. Word – 32-bit, Halfword – 16-bit, Byte – 8-bit

    - Memory is byte addressable, can hold 232 bytes (= 4 GB)
    - Word/ halfword /byte size data are placed at word/ halfword/ byte aligned addresses.
    - 32-bit ARM instructions are placed at word aligned addresses.
8. Byte order – Endian format
    - Word/halfword size data can be saved/retrieved in big endian or little endian format.
    - Big endian: MSB of word/halfword data are stored in lowest address and the data is addressed by address of MSB
    - Little endian: LSB of word/halfword data are stored in lowest address and the data is addressed by address of LSB

ARM7 based LPC2148 Microcontroller

The full form of an ARM is an advanced reduced instruction set computer (RISC) machine, and it is a 32-bit processor architecture expanded by ARM holdings. The applications of an ARM processor include several microcontrollers as well as processors. The architecture of an ARM processor was licensed by many corporations for designing ARM processor-based SoC products and CPUs

ARM7 processor is commonly used in embedded system applications. Also, it is a balance among classic as well as new-Cortex sequence. This processor is tremendous in finding the resources existing on the internet with excellence documentation offered by NXP Semiconductors. It suits completely for an apprentice to obtain in detail hardware & software design implementation.

Features of LPC2148

The main features of LPC2148 include the following.

- The LPC2148 is a 16 bit or 32 bit ARM7 family based microcontroller and available in a small LQFP64 package.
- ISP (in system programming) or IAP (in application programming) using on-chip boot loader software.
- On-chip static RAM is 8 kB-40 kB, on-chip flash memory is 32 kB-512 kB, the wide interface is 128 bit, or accelerator allows 60 MHz high-speed operation.
- It takes 400 milliseconds time for erasing the data in full chip and 1 millisecond time for 256 bytes of programming.
- Embedded Trace interfaces and Embedded ICE RT offers real-time debugging with high-speed tracing of instruction execution and on-chip Real Monitor software.
- It has 2 kB of endpoint RAM and USB 2.0 full speed device controller. Furthermore, this microcontroller offers 8kB on-chip RAM nearby to USB with DMA.
- One or two 10-bit ADCs offer 6 or 14 analogs i/ps with low conversion time as 2.44 μs/ channel.
- Only 10 bit DAC offers changeable analog o/p.
- External event counter/32 bit timers-2, PWM unit, & watchdog.
- Low power RTC (real time clock) & 32 kHz clock input.
- Several serial interfaces like two 16C550 UARTs, two I2C-buses with 400 kbit/s speed.
- 5 volts tolerant quick general purpose Input/output pins in a small LQFP64 package.
- Outside interrupt pins-21.
- 60 MHz of utmost CPU CLK-clock obtainable from the programmable-on-chip phase locked loop by resolving time is 100 μs.
- The incorporated oscillator on the chip will work by an exterior crystal that ranges from 1 MHz-25 MHz
- The modes for power-conserving mainly comprise idle & power down.
- For extra power optimization, there are individual enable or disable of peripheral functions and peripheral CLK scaling.

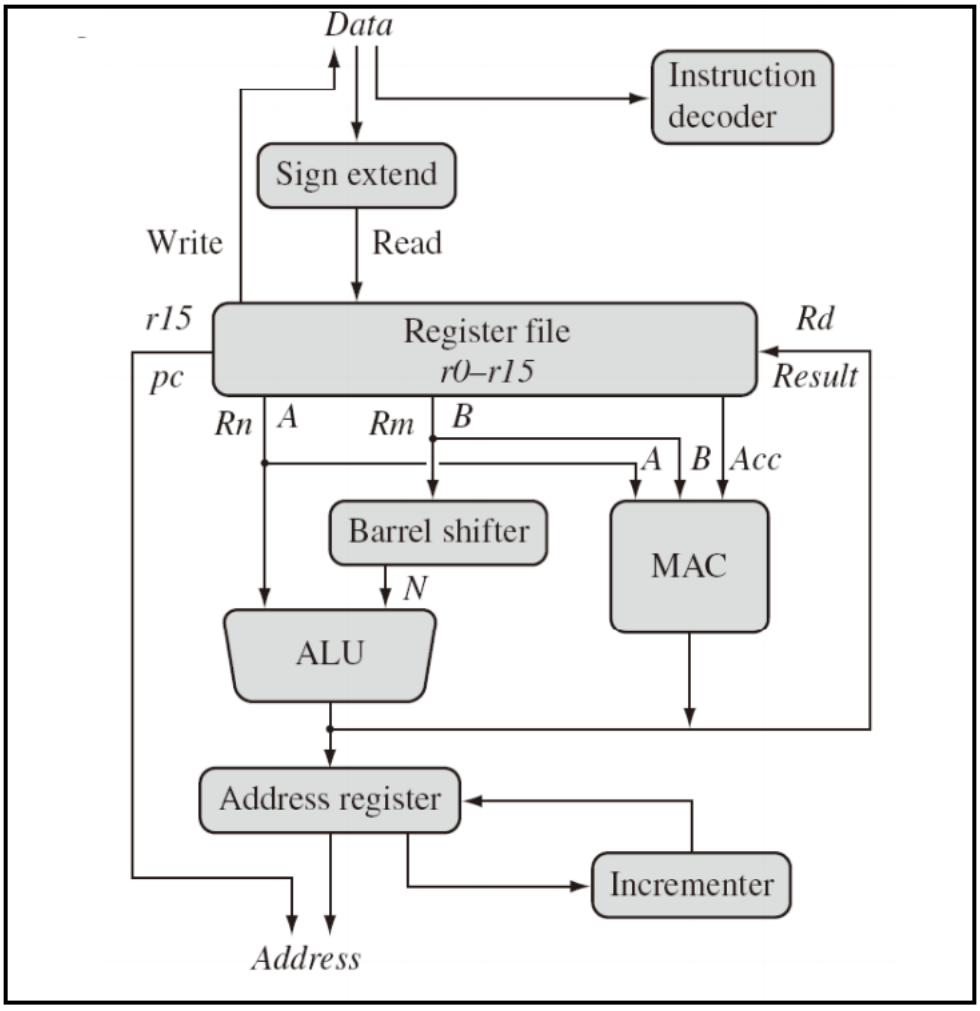
Memory

The LPC2148 microcontroller has 512-kB on-chip FLASH memory as well as 32-kB on-chip SRAM. Also, this microcontroller includes inherent support up to 2kB

finish point USB RAM. This memory is well matched for all the microcontroller applications.

1. The register set of ARM7 architecture is very large .it has sixteen registers of 32-bit each
2. It performs the low voltage operations and power consumption is also low because of very small die size (less than 0.25µm)
3. The number of addresses present in the architecture for 4 GB of linear address space are 2^{32} .
4. The ARM7 architecture can execute 300 million instructions per second. Hence its performance is very high
5. Instructions can have data operands of size 8,16 and 32 bit
6. The ALU operations in ARM7 are register based its RALU is 32 bit size
7. We can connect the DSP processors and Java accelerators to ARM7 because of the presence of coprocessor interface
8. ARM7 architecture has three stage pipelining technique.
9. The control signals of ARM memory interface are pipelined
10. It acts as a interface for direct connection to ETM(Embedded Trace Macro cell)
11. It also provides vast number of debug facilities such as , embedded ICE , RT(Real Time) debug and on-chip JTAG interface units
12. ARM7 architecture has only 2 types of interrupt requests.
    1. Fast Interrupt Request
    2. Interrupt Request

ARM core data flow model:



- **Von Neuman Architecture** Hence data coming through bus is either instruction or data (same memory).
- **The Sign extend** hardware converts signed 8-bit & 16-bit numbers to 32-bit values as they are read from memory & placed in a register (for signed values), fill zeros if unsigned.
- **Source operands (Rn & Rm)** are read from the register file using the internal buses A & B respectively & result **Rd** is written back.
- The PC value is in the address register which is fed in to the **incrementer**, then the incremented value is copied back in to r15.
- It is also written in to address register to be used as the address for the next instruction fetch.
- **ALU: (The Arithmetic & logic Unit)** or **MAC (multiply & accumulate Unit)** takes the register values Rn & Rm from A & B buses & computers a result).
- Data processing instructions write the result in Rd directly to the register file.
- Load & Store instruction use the ALU to generate on Address to be to be held in the **address register** & broadcast on the address bus.
- **Barrel shifter:**

- One important feature of the is that register Rm alternatively can be pre processed in barrel in barrel shifter before it enters the ALU [left shift , right shift , rotated etc.].
- Depending on the instruction Barrel Shifter may be used or it could be short circuit.
- Barrel shifter & ALU can calculate together a wide range of expression & address in the same cycle.

Registers

- In total 17(Visible)+20(Banked Registers) =37
- The active registers available in the user mode are shown below.
- This is protected mode which is normally used while executing applications.
- 16 Data registers & one status register
- r0 to r13 are **orthogonal general purpose register.**
- Orthogonal means, any instruction that you can apply to ro can equally be applied to any of the other register.
  - Eg. ADD ro, r1, r2
  - ADD r5, r6, r7
- **R13 (stack pointer)** and stores the top of the stack in the current processor mode.
- **R14(LR) Link Register** where the core puts the return address on executing a subroutine.
- **R15(PC) Program counter** stores the address of next instruction to be executed.
- In ARM state all ARM instruction are 32-bits wide.
- In Thumb state all instructions are 16-bit wide.
- In ARM state Instruction have to be four byte aligned in the memory. Which implies that the bottom two bits of the PC are always zero (Memory location 1000H,1004,1008H).

The processor has registers R0 through R15 and a number of special registers.

R0 through R12 are general purpose, but some of the 16-bit Thumb® instructions can only access R0 through R7 (low registers), whereas 32-bit Thumb-2 instructions can access all these registers. Special registers have predefined functions and can only be accessed by special register access instructions.

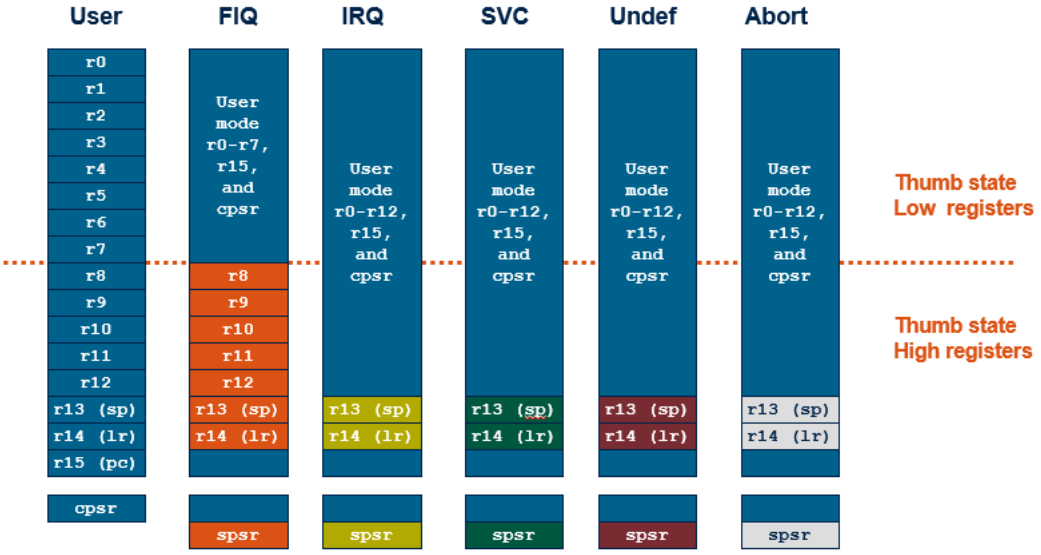
General Purpose Registers R0 through R7

The R0 through R7 general purpose registers are also called low registers. They can be accessed by all 16-bit Thumb instructions and all 32-bit Thumb-2 instructions. They are all 32 bits; the reset value is unpredictable.

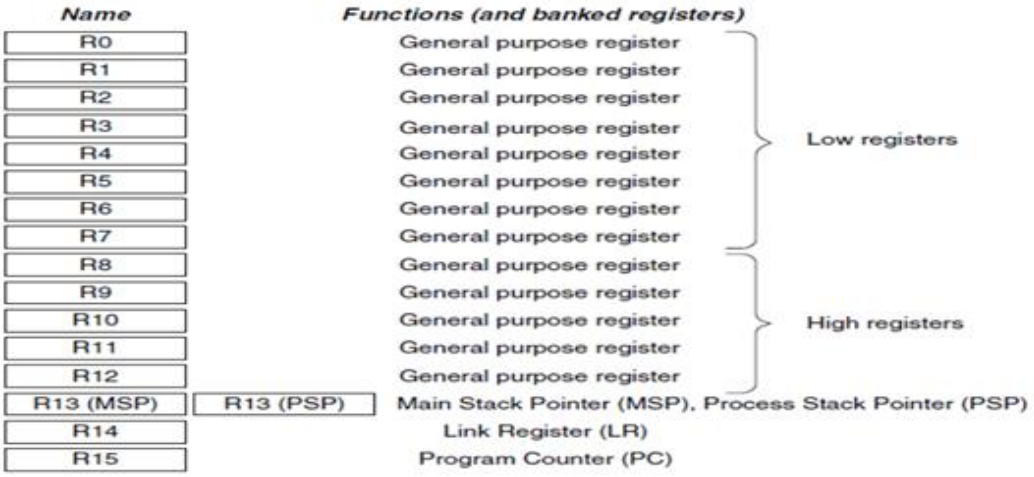
General Purpose Registers R8 through R12

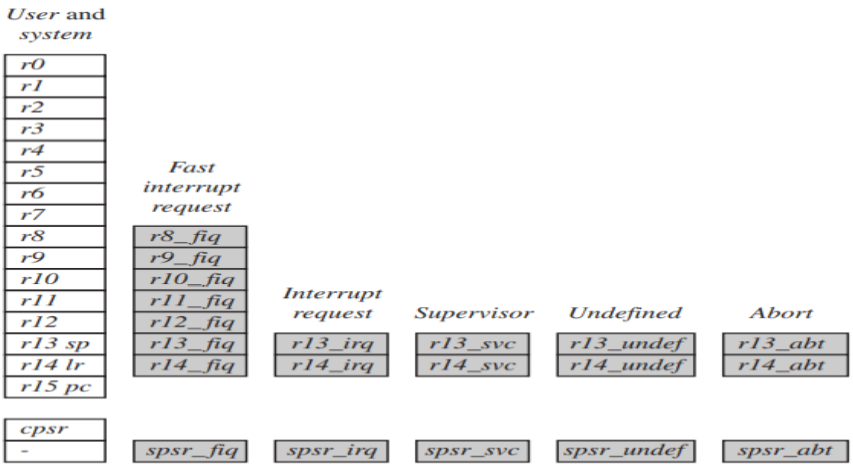
The R8 through R12 registers are also called high registers. They are accessible by all Thumb-2 instructions but not by all 16-bit Thumb instructions. These registers are all 32 bits; the reset value is unpredictable

Register Organization Summary:



Note: System mode uses the User mode register set



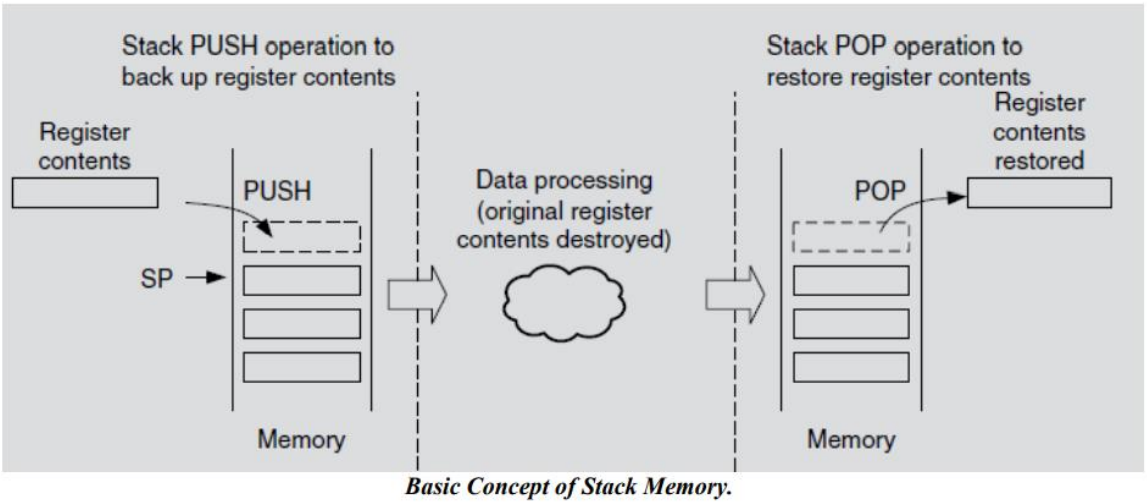


Complete ARM register set.

Stack Pointer R13

R13 is the stack pointer (SP)

Stack PUSH and POP Stack is a memory usage model. It is simply part of the system memory, and a pointer register (inside the processor) is used to make it work as a first-in/last-out buffer. The common use of a stack is to save register contents before some data processing and then restore those contents from the stack after the processing task is done. The SPs are used for accessing stack memory processes such as PUSH and POP.



Basic Concept of Stack Memory.

The assembly language syntax is as follows

PUSH {R0} ; R13=R13-4, then Memory[R13] = R0
POP {R0} ; R0 = Memory[R13], then R13 = R13 + 4

EX for subroutine

```
PUSH {R0-R7, R12, R14}          ; Save registers
POP {R0-R7, R12, R14}           ; Restore registers
BX R14                          ; Return to calling function
```

Link Register R14

R14 is the link register (LR). Inside an assembly program, you can write it as either R14 or LR. LR is used to store the return program counter (PC) when a subroutine or function is called—for example, when you're using the branch and link (BL) instruction

Program Counter R15

R15 is the PC. You can access it in assembler code by either R15 or PC.

- **When the processor is executing in ARM state:**
  - All instructions are 32 bits wide
  - All instructions must be word aligned
  - Therefore the **pc** value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be halfword or byte aligned).
- **When the processor is executing in Thumb state:**
  - All instructions are 16 bits wide
  - All instructions must be halfword aligned
  - Therefore the **pc** value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned).
- **When the processor is executing in Jazelle state:**
  - All instructions are 8 bits wide
  - Processor performs a word access to read 4 instructions at once

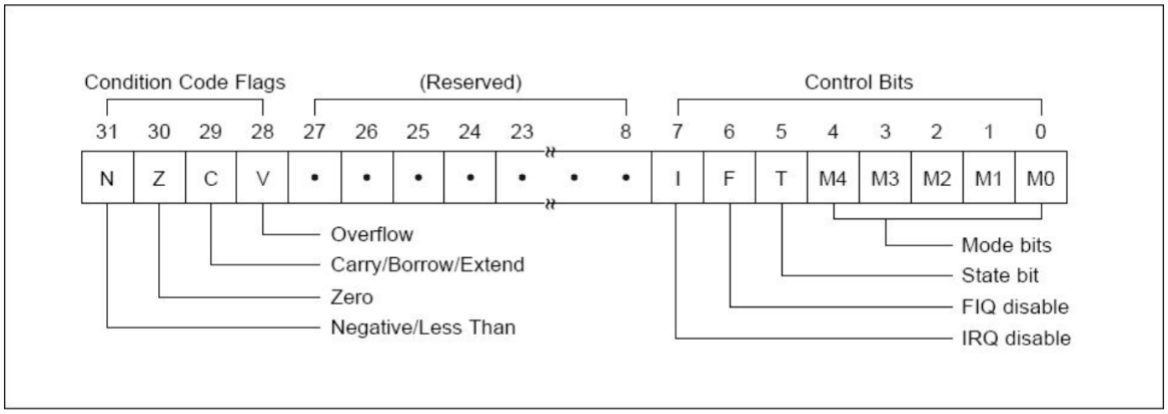
Processor Modes:

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

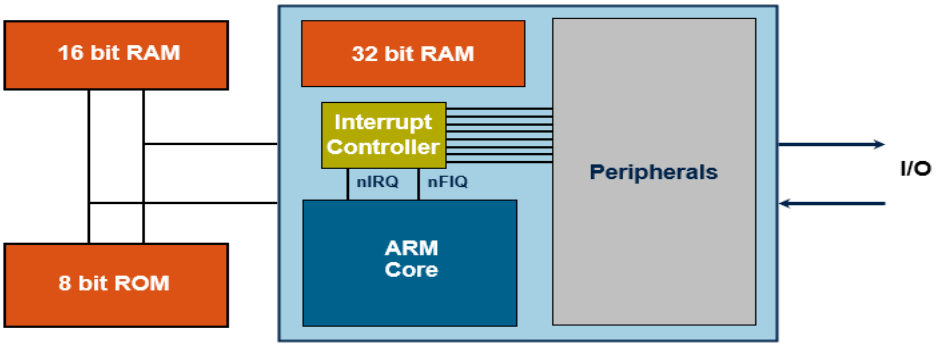
Program Status Registers

CPSR Diagram



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

ARM-based System:



AMBA: Advanced Microcontroller Bus Architecture

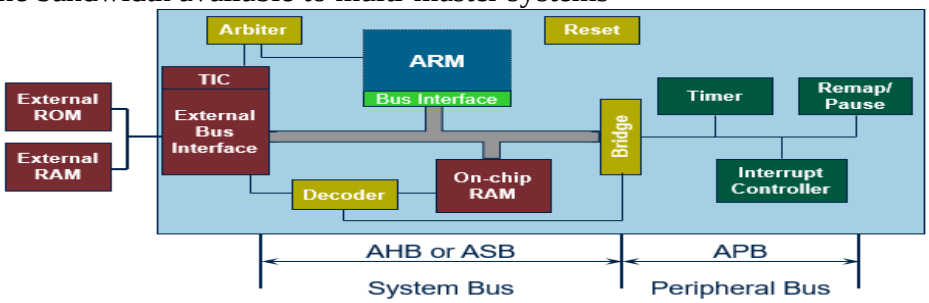
It is a specification for an on-chip bus, to enable macro cells

- (such as a CPU, DSP, Peripherals, and memory controllers) to
- be connected together to form a microcontroller or complex
- **peripheral chip.**

It defines

- A high-speed, high-bandwidth bus, the Advanced High
- Performance Bus (AHB).
- A simple, low-power peripheral bus, the Advanced
- Peripheral Bus (APB).
- Access for an external tester to permit modular testing and
- fast test of cache RAM
- Essential housekeeping operations (reset/power-up)

Provides a high-performance and fully synchronous back plane. (Backplane means additional set of controllers, which can access another common bus, which is distinct from system bus in a multilevel buses in the system.) λ Multi-layer AHB in version ARM926EJ-S and all members of the ARM10 family represents a significant advancement. It reduces access latencies and increases the bandwidth available to multi-master systems



- **AMBA**
  - Advanced Microcontroller Bus Architecture
- **ADK**
  - Complete AMBA Design Kit
- **ACT**
  - AMBA Compliance Testbench
- **PrimeCell**
  - ARM's AMBA compliant peripherals