Table of Contents

Cortex M4 Block Diagram

† For the Cortex-M4F processor, the core includes a Floating Point Unit (FPU)

Cortex M4 Features

- 32 bit RISC processor: 32-bit register, 32-bit internal data path, 32-bit bus interface
- handle 16, 32, 64 bit data
- Thumb-2 Technology
- Configurable Nested Vectored Interrupt Controller (NVIC): closely integrated with the processor core to achieve low latency interrupt processing
- Better debugging support
  - breakpoints support
  - watchpoints
  - support printf() style debugging.
- Multiple high-performance bus interfaces
- Hardware Divide (2-12 Cycles)
- Barrel shifter
- DSP and SIMD extensions
- Single-cycle multiply-accumulate (MAC) unit(Up to 32 x 32 + 64 -¿ 64)
- Optional Memory Protection Unit (MPU)
- Optional Floating Point Unit (FPU) unit

Floating Point Unit (FPU), Memory Protection Unit (MPU)

- Floating Point Unit (FPU)
  - 32-bit instructions for single-precision (C float) data-processing operations.
  - Combined Multiply and Accumulate instructions for increased precision.
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root.
  - Hardware support for denormals and all IEEE rounding modes.
  - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers.
  - Decoupled three stage pipeline.
- Memory Protection Unit (MPU)
  - An optional MPU for memory protection
  - Eight memory regions.
  - Sub Region Disable (SRD), enabling efficient use of memory regions.
  - The ability to enable a background region that implements the default memory map attributes.

Nested Vectored Interrupt Controller (NVIC)

- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing.
  - External interrupts, configurable from 1 to 240.
  - Bits of priority, configurable from 3 to 8.
  - Dynamic reprioritization of interrupts.
  - Priority grouping. This enables selection of preempting interrupt levels and non preempting interrupt levels.
  - Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
  - Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.
  - Optional Wake-up Interrupt Controller (WIC), providing ultra-low power sleep mode support.

Bus Matrix

- Bus Matrix
  - The bus matrix connects the processor and debug interface to the external buses.
  - The bus matrix interfaces to the following external buses
    - **ICode bus**: This is for instruction and vector fetches from code space. This is a 32-bit AHB-Lite bus.
    - **DCode bus**: This is for data load/stores and debug accesses to code space. This is a 32-bit AHB-Lite bus.
    - **System bus**: This is for instruction and vector fetches, data load/stores and debug accesses to system space. This is a 32-bit AHB-Lite bus.
    - **Private Peripheral Bus (PPB)**. This is for data load/stores and debug accesses to PPB space. This is a 32-bit APB bus.
  - The bus matrix also controls the following:
    - Unaligned accesses. The bus matrix converts unaligned processor accesses into aligned accesses.
    - Bit-band support that includes atomic bit-band write and read operations.
    - Write buffer for buffering of write data.

Bus Interfaces

- ICode memory interface
  - Instruction fetches from Code memory space, 0x00000000 to 0x1FFFFFFC , are performed over the 32-bit AHB-Lite bus.
  - All fetches are word-wide.
  - The number of instructions fetched per word depends on the code running and the alignment of the code in memory.
- DCode memory interface
  - Data and debug accesses to Code memory space, 0x00000000 to 0x1FFFFFFF , are performed over the 32-bit AHB-Lite bus.
  - Core data accesses have a higher priority than debug accesses on this bus. This means that debug accesses are waited until core accesses have completed when there are simultaneous core and debug access to this bus.
  - Control logic in this interface converts unaligned data and debug accesses into two or three aligned accesses, depending on the size and alignment of the unaligned access. This stalls any subsequent data or debug access until the unaligned access has completed.
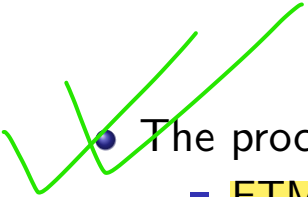
Bus Interfaces

- **System interface**
  - Instruction fetches, and data and debug accesses, to address ranges 0x20000000 to 0xDFFFFFFF and 0xE0100000 to 0xFFFFFFFF are performed over 32-bit AHB-Lite bus.
  - For simultaneous accesses to this bus, the arbitration order in decreasing priority is:
    - data accesses
    - instruction and vector fetches
    - debug.
  - The system bus interface contains control logic to handle unaligned accesses,bit-band accesses, and pipelined instruction fetches.
- **Private Peripheral Bus (PPB)**
  - Data and debug accesses to external PPB space, 0xE0040000 to 0xE00FFFFF , are performed over the 32-bit Advanced Peripheral Bus (APB) bus.
  - The Trace Port Interface Unit (TPIU) and vendor specific peripherals are on this bus.
  - Core data accesses have higher priority than debug accesses, so debug accesses are waited until core accesses have completed when there are simultaneous core and debug access to this bus.
  - Only the address bits necessary to decode the External PPB space are supported on this interface.

External Bus Interfaces

- The processor incorporates three external bus interfaces
  - ETM interface that allows the connection of an Embedded Trace Macrocell
  - AHB Trace Macrocell interface that enables simple connection of an ETM to the processor
  - Advanced High-performance Bus Access Port (AHB-AP) interface for debug accesses
- The processor incorporates the following external interfaces:
  - Multiple memory and device bus interfaces.
  - ETM Interface
  - Trace Port Interface
  - Debug Port Interface

Bus Interfaces

- **ETM interface**: The ETM interface enables simple connection of the ETM-M4 to the processor, and provides a channel for instruction trace to the ETM.
- The bus matrix also controls the following:
  - Unaligned accesses. The bus matrix converts unaligned processor accesses into aligned accesses.
  - Bit-band support that includes atomic bit-band write and read operations.
  - Write buffer for buffering of write data.

Debugger

- Low-cost debug solution that features:
    - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESET is asserted.
    - Serial Wire Debug Port (SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP) debug access.
    - Optional Flash Patch and Breakpoint (FPB) unit for implementing breakpoints and code patches.
    - Optional Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling.
    - Optional Instrumentation Trace Macrocell (ITM) for support of printf() style debugging.
    - Optional Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode.
    - Optional Embedded Trace Macrocell (ETM) for instruction trace.

Configurable Options in Cortex- M4

- MPU : Memory Protection Unit
- FPB: Flash Patch and Breakpoint
- DWT: Data Watchpoint and Trace Unit
- ITM: Instrumentation Trace Macrocell Unit
- ETM: Embedded Trace Macrocell
- AHB-AP: Advanced High-performance Bus Access Port
- HTM Interface: AHB Trace Macrocell interface —item TPIU: Trace Port Interface Unit
- WIC: Wake-up Interrupt Controller
- Debug Port AHB-AP interface
- FPU: Floating Point Unit
- Bit-banding
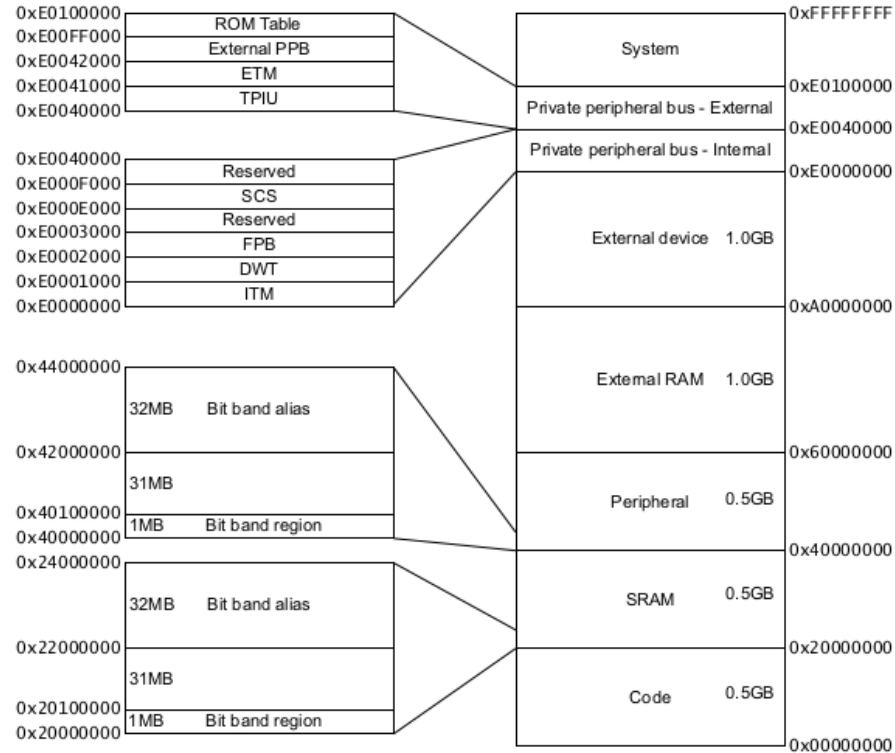- Constant AHB control

System address map

Figure 3-1 System address map
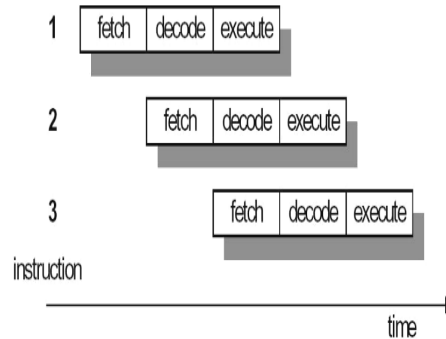
Figure 1

Cortex M4 Features

Figure 2

- Fetch : The instruction is fetched from memory and placed in the instruction pipeline
- Decode : The instruction is decoded and the datapath control signals prepared for the next cycle
- Execute : The register bank is read, an operand shifted, the ALU result generated and written back into destination register
- At any time slice, 3 different instructions may occupy each of these stages, so the hardware in each stage has to be capable of independent operations

Data Size and Instruction Set

- When used in relation to the ARM:
  - Halfword means 16 bits (two bytes)
  - Word means 32 bits (four bytes)
  - Doubleword means 64 bits (eight bytes)
- Three instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set
  - 32-bit Thumb Instruction Set

Registers

- ARM has 37 registers all of which are 32-bits long.
  - 1 dedicated program counter ≻ c
  - 1 dedicated current program status register *CPSR*
  - 5 dedicated saved program status registers
  - 30 general purpose registers
- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr
  - Privileged modes (except System) can also access a particular spsr (saved program status register)
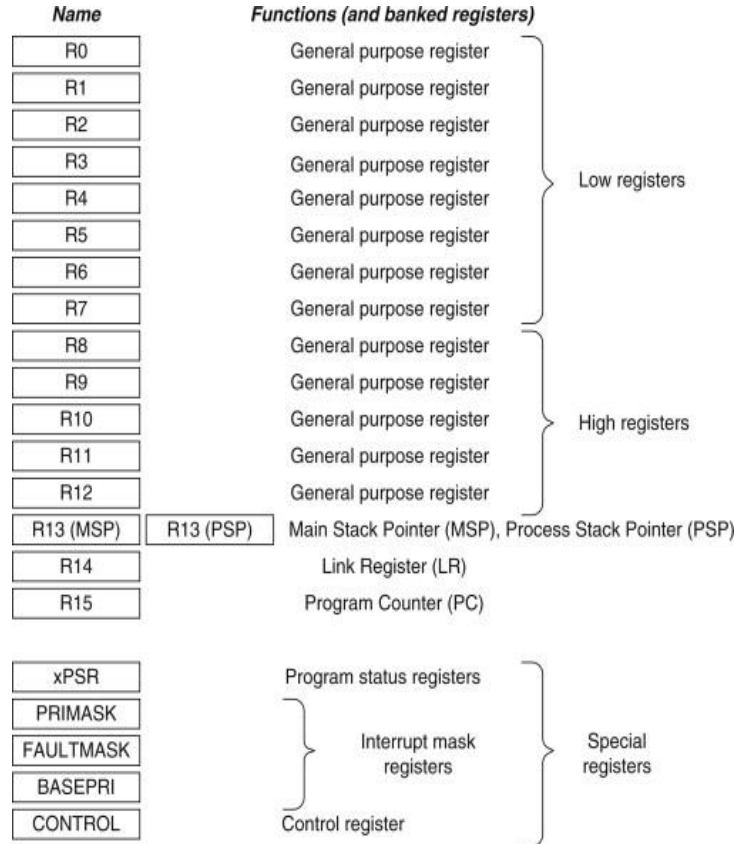
Register Sets

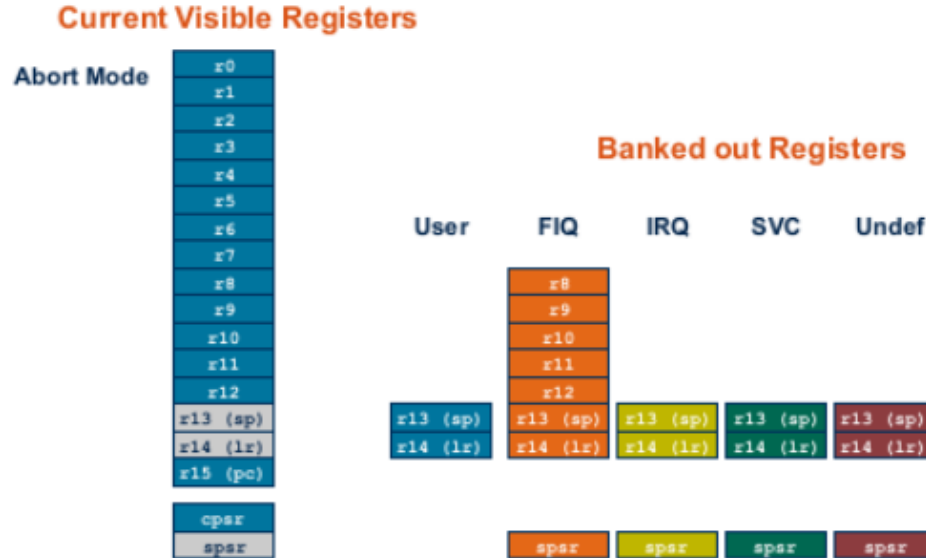Figure 3

Register Sets

Figure 4

Register Sets

- R0–R12:
  - General-Purpose Registers
  - 32-bit general-purpose registers for data operations or for addressing
  - R0-R7: Low registers, 16-bit Thumb instructions can only access to these registers
  - R8-R12: High registers, accessible by all 32-bit instructions

- Register R13
  - used as the Stack Pointer (SP)
  - two modes
  - SP_main: main stack pointer (MSP)
    - default SP
    - used by OS-kernel, exception handler, application code with privileged access
  - SP_process: process stack pointer (PMSP)
    - usually in Thread mode
    - used by application code not running exception handler
  - PUSH and POP instruction used for access stack memory
  - full descending stack arrangement
  - SP decrements when new data is stored in stack
  - PUSH and POP are always word-aligned
  - Example: PUSH RO ; Memory[R13]=R0
    POP R0 ; R0=memory[R13], R13=R13+4

Register Sets

- R14:
    - Link Registers
    - Store the return address when a function/subroutine/interrupt is called and updates automatically
    - At the end of the function, PC is loaded with the value from LR and resume the calling program
    - For nested function, the value of LR needs to be stored in stack
    - Bit 0 is readable and writable, bit0=1 to indicate Thumb state
- Example
main
....
BL func; Call func using Brunch with link instruction
; PC= func
; LR= the next instruction in main
....
func
.... ; code for func
....

Register Sets

- R15:
  - Program Counter (PC)
  - Contains the address of the next instruction to be executed
  - Readable and writeable
  - A read operation returns the current instruction address plus 4
  - Writing to PC causes a branch operation
  - LSB is set zero (default) to be aligned with half-word or word addresses. It is set to 1 to indicate Thumb state.
  - Example: 0x1008 : MOV R0, PC ; R0=0x1012