# Arquitetura ARM Cortex-M4

Prof. Hugo Vieira Neto
2020/1

## Objetivo

- Revisar os principais conceitos do núcleo ARM Cortex-M4:
  - Modelo de programação
  - Modelo de memória
  - Arquitetura Load/Store
  - Pipeline e execução condicional
  - Padrão AAPCS

## Arquitetura x Organização

- Arquitetura = documento de especificação
  - Conjunto de Instruções
  - Exceções/Interrupções
  - Modelo de Memória
  - Registradores
  - Ex: ARMv4, ARMv7, etc.
- Não tem custo, pode ser obtido diretamente do website da ARM

## Arquitetura x Organização

- Organização = implementação física (silício)
  - Ex: ARM7TDMI, ARM Cortex-M4, etc.
- ARM vende implementações de núcleos em VHDL ou máscaras de difusão para empresas licenciadas

## ARM Cortex-M4

- Arquitetura ARMv7E-M
- Conjunto de instruções Thumb-2
- 21 registradores (32 bits)
- Um único registrador de estado do programa
- Mapa de memória fixo
  - Periféricos mapeados em memória
- Sem MMU (Unidade de Gerenciamento de Memória) ou memória cache

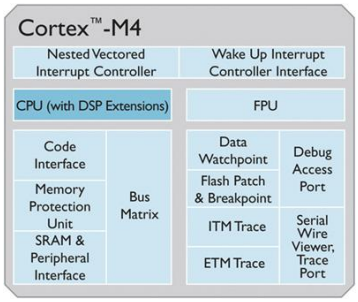
## ARM Cortex-M4

- Controlador de interrupções faz parte da macrocélula do núcleo do processador
- Tabela de vetores de interrupção contém endereços, não instruções
- Interrupções automaticamente salvam e recuperam o estado do processador
  - Tratamento muito eficiente de interrupções
- Gerenciamento de consumo

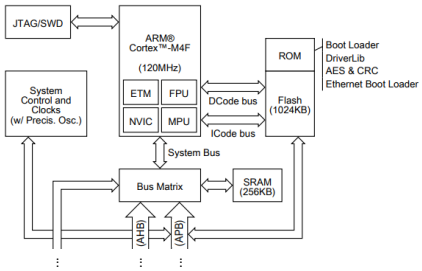
## ARM Cortex-M4

- Projetado para ser programado em C
  - Até mesmo o tratamento de interrupções
- Suporte para sistemas operacionais (RTOS)
  - Modelo Usuário/Supervisor
  - Exceções SVC, PendSV e SysTick
  - Unidade de Proteção de Memória (MPU)
- Cortex-M4F possui uma Unidade de Ponto Flutuante (FPU)

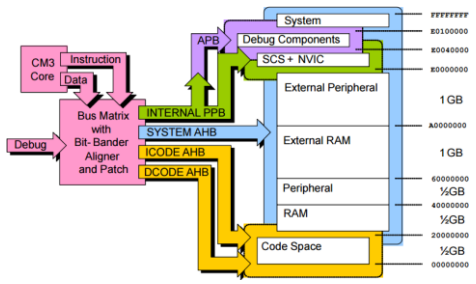
## Visão Simplificada do Cortex-M4




## Diagrama em Blocos – TM4C1294




## Mapa de Memória do Cortex-M4




## Mapa de Memória – TM4C1294

| Início | Final | Descrição |
|---|---|---|
| 0x0000 0000 | 0x000F FFFF | Flash interna (1MiB) |
| 0x0010 0000 | 0x01FF FFFF | Reservado |
| 0x0200 0000 | 0x02FF FFFF | ROM interna (16 MiB) |
| 0x0300 0000 | 0x1FFF FFFF | Reservado |
| 0x2000 0000 | 0x2003 FFFF | SRAM interna (256KiB) |
| 0x2004 0000 | 0x21FF FFFF | Reservado |
| 0x2200 0000 | 0x2234 FFFF | Cópia da SRAM (bit-band) |
| 0x2235 0000 | 0x3FFF FFFF | Reservado |
| 0x4000 0000 | 0xDFFF FFFF | Periféricos |
| 0xE000 0000 | 0xFFFF FFFF | Periféricos Privados |

## Registradores do Núcleo (32 bits)

- 13 registradores de propósito geral
  - R0 a R7 (low registers)
  - R8 a R12 (high registers)
- 3 registradores de propósito específico
  - R13 = Stack Pointer (SP)
  - R14 = Link Register (LR)
  - R15 = Program Counter (PC)

## Registradores do Núcleo (32 bits)

- 5 registradores especiais
  - xPSR = Program Status Register
  - PRIMASK = Priority Mask Register
  - FAULTMASK = Fault Mask Register
  - BASEPRI = Base Priority Mask Register
  - CONTROL = Control Register

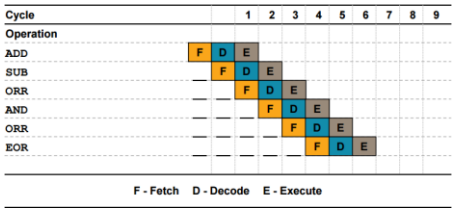
## Arquitetura Load/Store

- Acesso à memória:
  - Somente instruções LD leem dados da memória
  - Somente instruções ST escrevem dados na memória
  - Instruções de processamento de dados não acessam a memória
- Operações sobre dados em memória requerem:
  - Leitura da memória
  - Operação (em registrador)
  - Escrita na memória

## Pipeline em Três Estágios

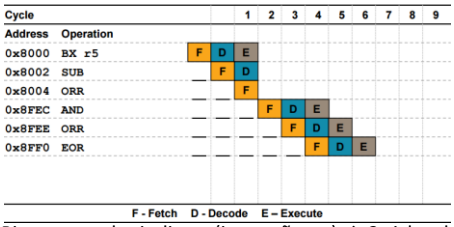
1. Busca (*fetch*)
   - Busca instrução na memória
2. Decodificação (*decode*)
   - Decodifica registradores usados na instrução
3. Execução (*execute*)
   - Lê dos registradores
   - Realiza operações
   - Escreve nos registradores

## Pipeline: Situação Ideal




- Todas as operações realizadas em registradores → 6 instruções em 6 ciclos de clock (Cortex-M4)

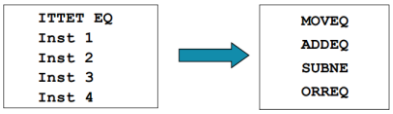
## Pipeline: Efeito de Saltos




- Pior caso: salto indireto (instrução `BX`) → 3 ciclos de clock para completar o salto (Cortex-M4)

## Execução Condicional

- Bloco If-Then (IT)
  - Até 3 instruções condicionais "then" (T) ou "else" (E) podem ser adicionadas ao bloco
  - Condiciona a execução de até 4 instruções consecutivas



## Exceções e Interrupções

- Exceções (faltas, Debug, SVC, PendSV)
- 1 interrupção não-mascarável (NMI)
- 1 interrupção de SysTick
- 1 a 240 interrupções externas com controle de prioridade
  - Implementação define número de interrupções
- Controlador de interrupções (NVIC) acoplado fortemente ao núcleo do processador

## Gerenciamento de Consumo

- Modos de baixo consumo (*sleep modes*)
  - Sleep Now
    - Wait for Interrupt (WFI) / Wait for Event (WFE)
  - Sleep On Exit
    - Imediatamente após a saída da interrupção de menor prioridade
  - Deep Sleep
    - Longa duração, PLL desligado
- Controlados pelo NVIC

## Padrão AAPCS

- Padroniza as chamadas de função na arquitetura ARM (*ARM Architecture Procedure Call Standard*)
- Passagem de parâmetros para a função:
  - Primeiros parâmetros em R0, R1, R2 e R3
  - Demais parâmetros na pilha
- alor de retorno da função:
  - R0 (32 bits)
  - R1:R0 (64 bits)

## Padrão AAPCS

- Fortemente relacionado com o salvamento de contexto automático em interrupções
- Registradores R0, R1, R2, R3 e R12 pertencem à função chamada
- Demais registradores (R4, R5, R6, R7,R8, R9, R10 e R11) pertencem à função chamadora
- Alinhamento da pilha deve ser de 64 bits:
  - Número par de registradores em PUSH/POP

## Padrão AAPCS

- Quem tem a obrigação de salvar o conteúdo dos registradores utilizados numa função?
  - Função chamada deve salvar R4 a R11 na pilha antes de alterar seus conteúdos (atenção para o alinhamento em 64 bits!)
  - Função chamada pode usar livremente R0 a R3 e R12 (devem ser salvos na pilha, se necessário, pela função chamadora)

## Atividade Extra-classe

- Leitura:
  - Padrão ARM: “Procedure Call Standard for the ARM Architecture (AAPCS)” (Seção 5)
  - Livro: “The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors” (Capítulo 8)
  - Livro: “The Designer’s Guide to The Cortex-M Processor Family” (Capítulo 3, com atenção especial às Seções “Priority and Preemption” e “Exception Model”)

## Atividade Extra-classe

- Analisar o projeto “c_asm” da área de trabalho “EK-TM4C1294XL_IAR”
  - Verificar com o depurador como são passados os parâmetros à função chamada (Assembly) e como é passado o valor de retorno à função chamadora (linguagem C)
  - Experimentar passar diferentes quantidades de parâmetros, com diferentes tamanhos (8, 16, 32, 64 bits), à função implementada em Assembly