

Sistemas Microcontrolados

Arquitetura

Prof. Guilherme Peron

Microprocessador

- Analogia: vamos fazer um bolo



Microprocessador

- O que é precisamos?
 1. Livro de receitas → passo-a-passo de como fazer (rotina);
 2. Definir o sabor;
 3. Acessar a receita do livro;
 4. Descobrir os ingredientes;
 5. Ir até a página correspondente da receita → Software;
 6. Em qual página está? “XYZ” (1kg de bolo)

Microprocessador

- Qual é a minha atitude neste instante?

Resp: Somente ler a receita!!!

- Analisando o caso, qual seria a nossa memória?

Microprocessador

- Memória ROM (*Read-only Memory*);
- Receita (Página “XYZ”):
 1. 3 xícaras de farinha;
 2. 2 xícaras de açúcar;
 3. Cobertura: (vide página “ABC”).
- **Por que a cobertura não está escrita aqui também?**

Microprocessador

Resp: Várias receitas utilizam! (isso seria uma subrotina)

- O que devemos fazer?

Microprocessador

Resp: Vamos fazer a cobertura e depois voltamos e damos continuidade a nossa receita.

- Sequência:
 1. Executar XYZ;
 2. Executar ABC;
 3. Retornar XYZ e continuar.
- E se eu quiser fazer um bolo de 2kg?

Microprocessador

Resp: Podemos utilizar uma folha de rascunho → RAM (*Random Access Memory*)

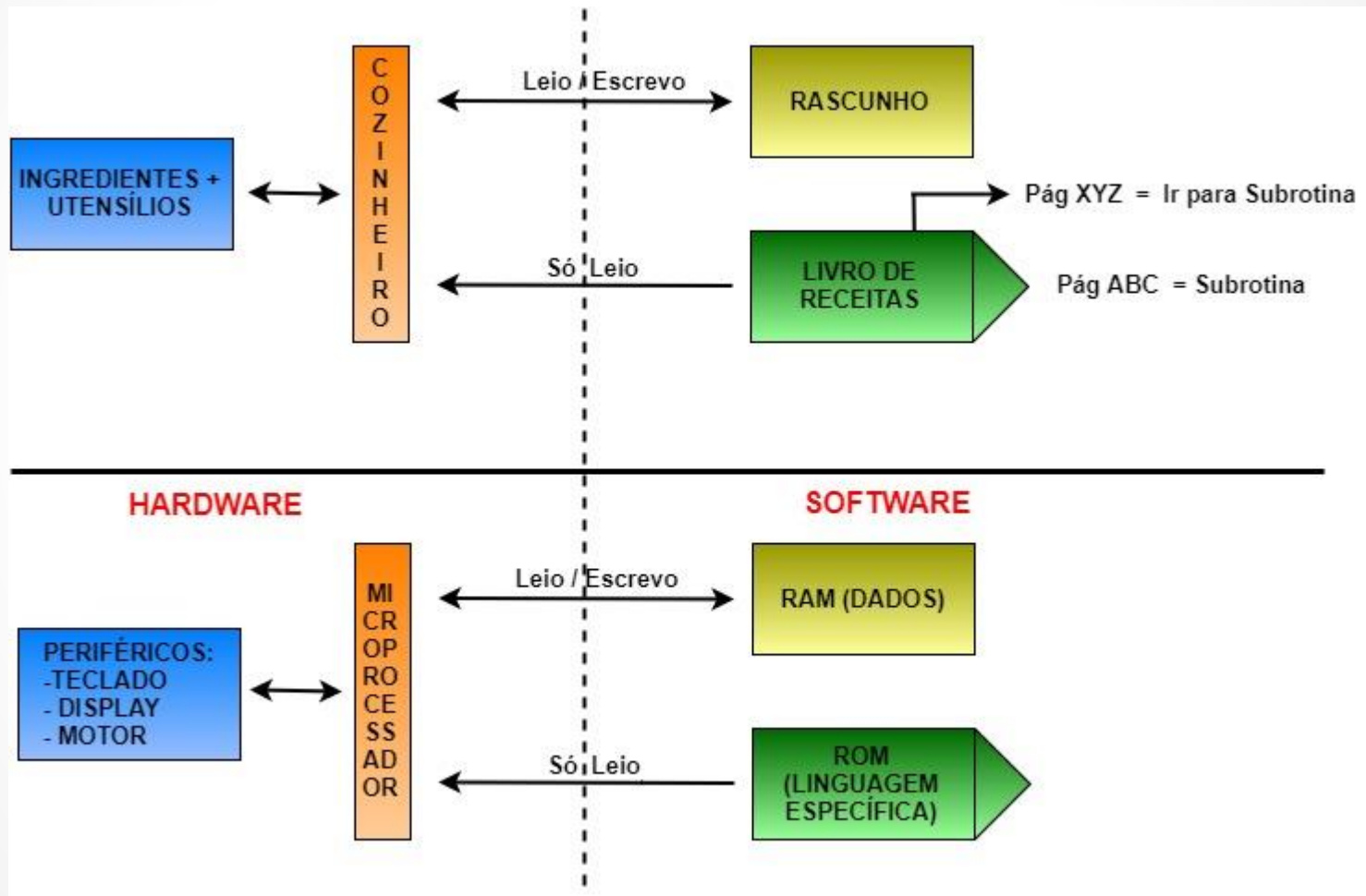
- Resumindo:
 1. Cozinheiro - microprocessador (Hardware);
 2. Ingredientes - eletrônica (Hardware);
 3. Livro de receitas - ROM (Software);
 4. Rascunho - RAM (Software).
- Qual a diferença entre o microprocessador e o cozinheiro?

Microprocessador

Resp: Vocabulário.

- Resultado → precisamos entender o vocabulário do microprocessador.

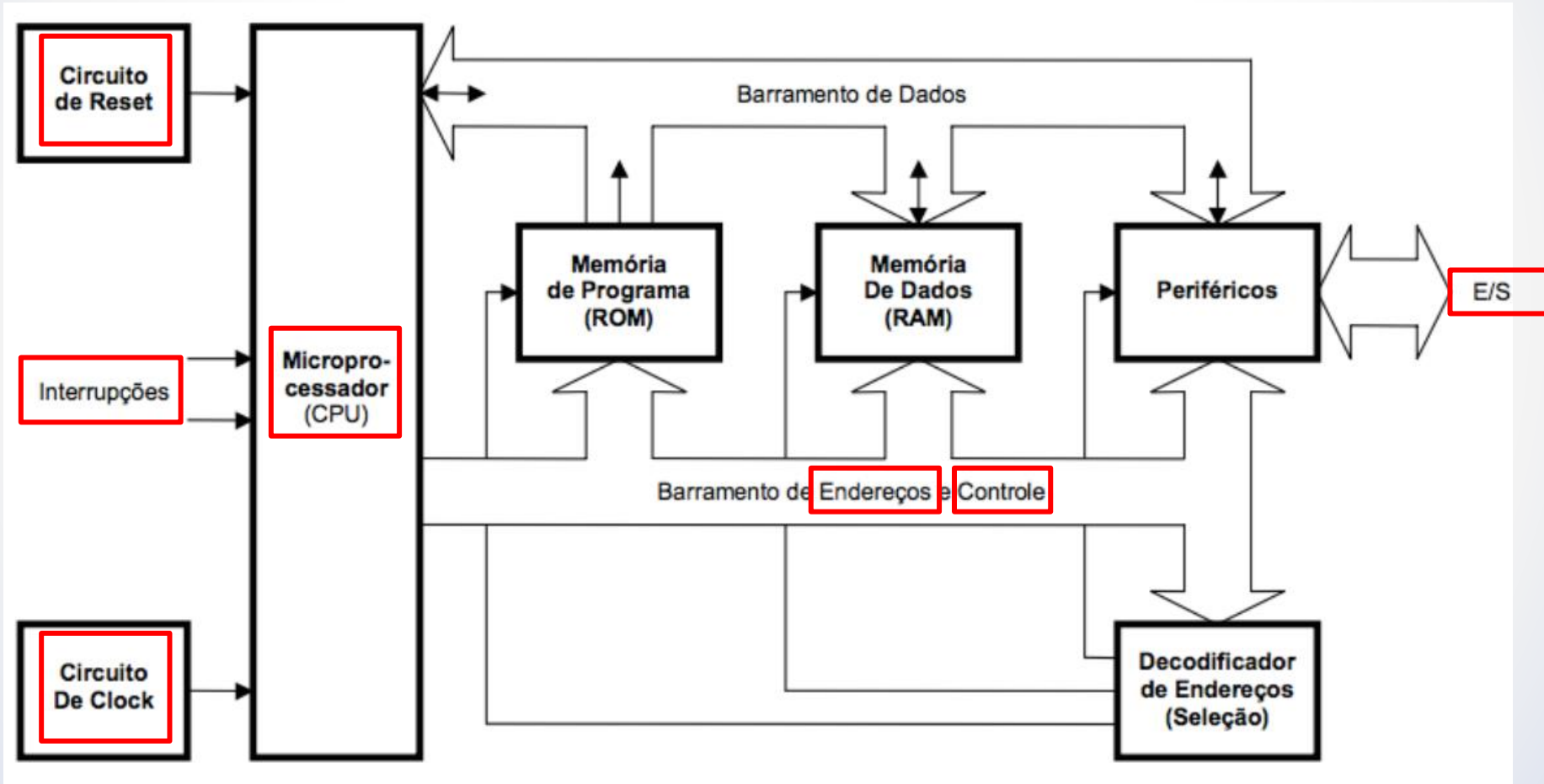
Microprocessador



Microprocessador

- Componente Eletrônico;
- Composição Básica:
 1. ROM: Ler INSTRUÇÕES que devem ser executadas;
 2. RAM: Armazenar temporariamente informações de uso das INSTRUÇÕES;
 3. Barramentos de Dados e Endereços.
- Atribuições:
 1. Executar tarefa da ROM;
 2. Se comunicar com o mundo (teclado, impressora, LCD...)

Diagrama Genérico



Funções

- **Barramento de Endereços:** Selecionar com qual posição de memória ou periférico deseja se comunicar;
- **Controle:** Permitem o microprocessador acionar a RAM e a ROM em um certo tempo específico e vice-versa (ligar/desligar);
- **Barramento de E/S (I/O):** Comunicação com o mundo externo;
- **CPU (Engenheiro formado e casado):** Se comunicar e acionar todos os barramentos, obedecendo a ROM;

Funções da CPU

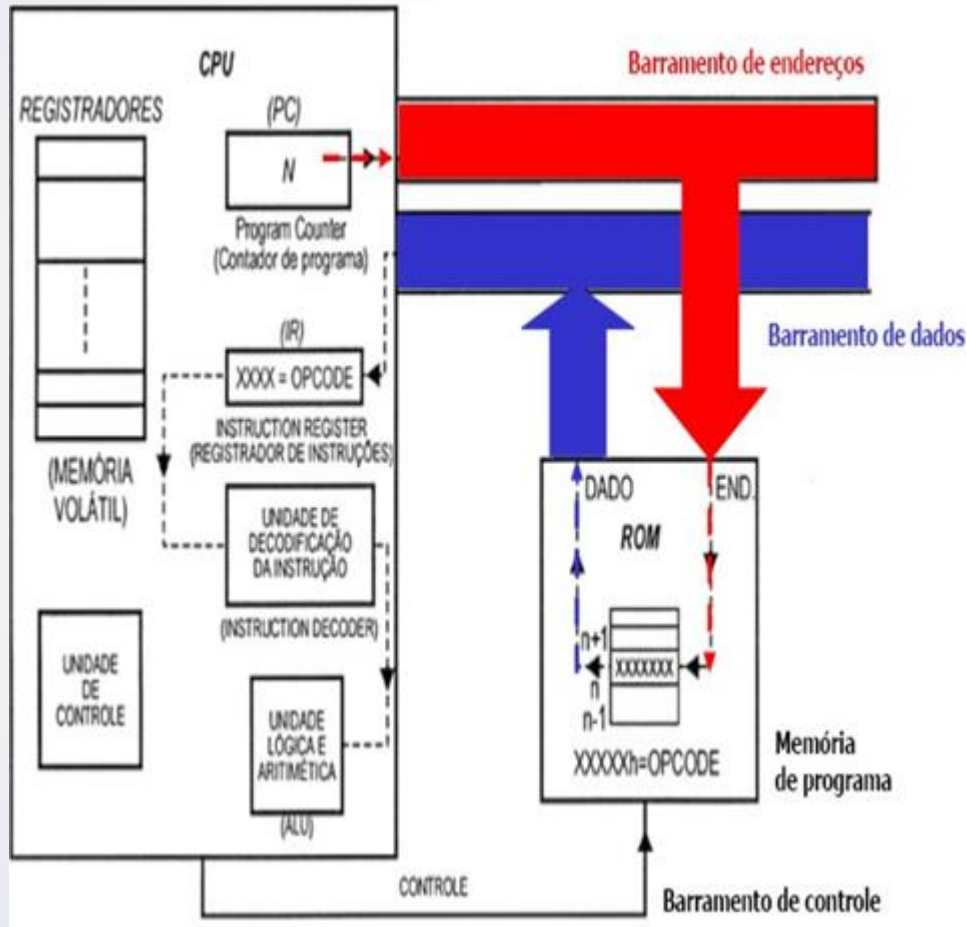
- a) Buscar instruções continuamente na ROM;
- b) Executar essas instruções;
- c) Executar funções lógicas;
- d) Executar funções aritméticas;
- e) Executar transferências de dados (internas e externas);
- f) Executar comparação de dados -> tomada de decisão;

Meio “Tapada” → Só entende binário!!!

Funções

- **Oscilador:** Tarefas internas e externas sincronizadas e com uma velocidade predeterminada;
- **Reset:** Iniciar as rotinas e realizar a leitura no primeiro endereço;
- **Interrupções:** Pinos de acesso externo que interrompem o microprocessador.
- **Registradores:** Armazenamento de alta velocidade dentro do microprocessador
- **Barramento:** Conjunto de fios utilizados para passar informação entre os módulos

Microprocessador



1. O endereço de PC vai para BUS;
2. Ativa o sinal de controle da ROM;
3. Ciclo de busca:
 - a. lê da ROM;
 - b. no endereço dado pelo PC;
 - c. lê/transmite pelo BUS dados.
4. Instrução carregada e armazenada no IR;
5. Incrementa o PC (próxima inst.)
6. Inicia o ciclo de execução.

Microprocessador x Microcontrolador

- Qual a diferença?

Microprocessador x Microcontrolador

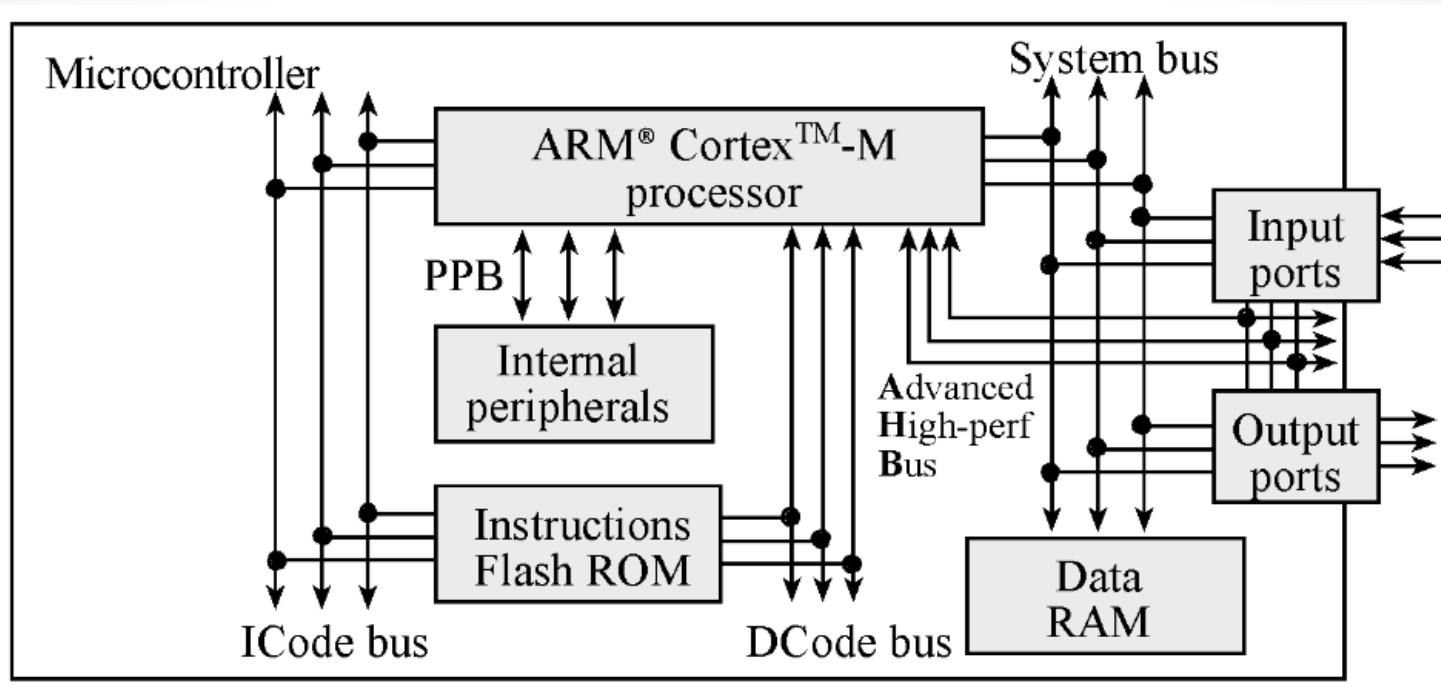
- Hardware interno diferente:
 - Microprocessador (-)
 - Microcontrolador (+)
- Microprocessador contém:
 - IR, PC, ALU...
- Microcontrolador contém:
 - Tudo o que o microprocessador tem
 - + Periféricos

Periféricos

- Exemplos:
 - Memórias;
 - Temporizadores;
 - Portas de Entrada/Saída;
 - Teclados;
 - *Displays*;
 - Impressoras;
 - Sensores;
 - Motores;

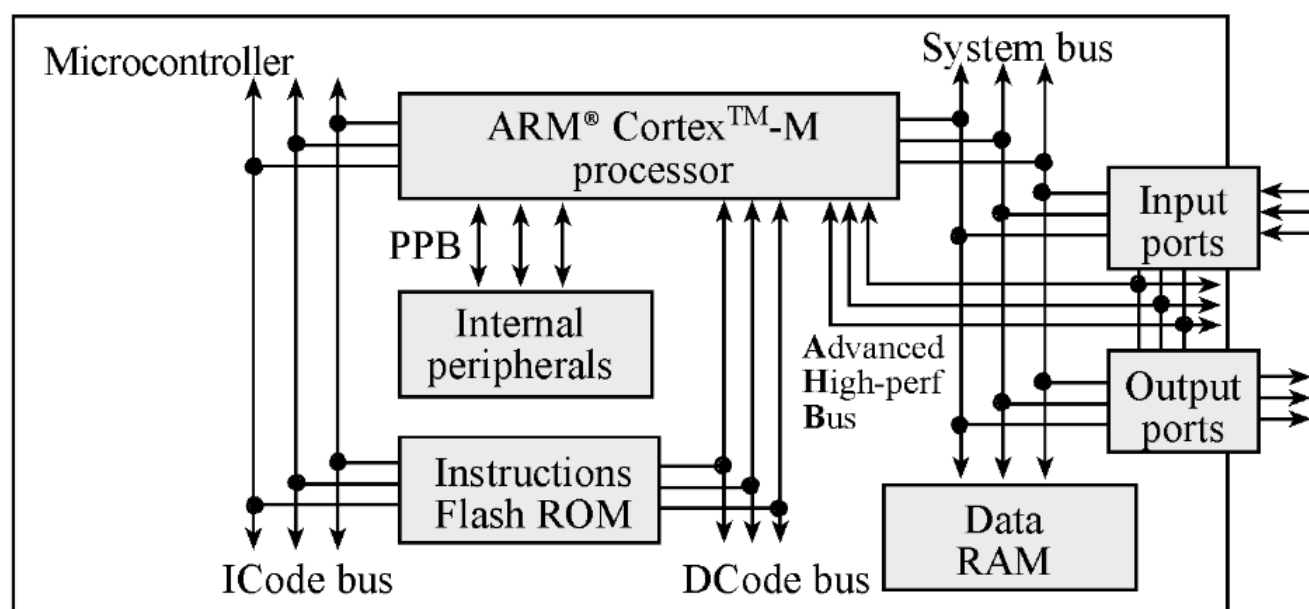
Arquitetura ARM Cortex M4

Arquitetura ARM Cortex M4



- Processador ARM Cortex-M4
 - ARMv7-ME
- Arquitetura **Harvard**
 - Diferentes barramentos para instruções e dados

Arquitetura ARM Cortex M4

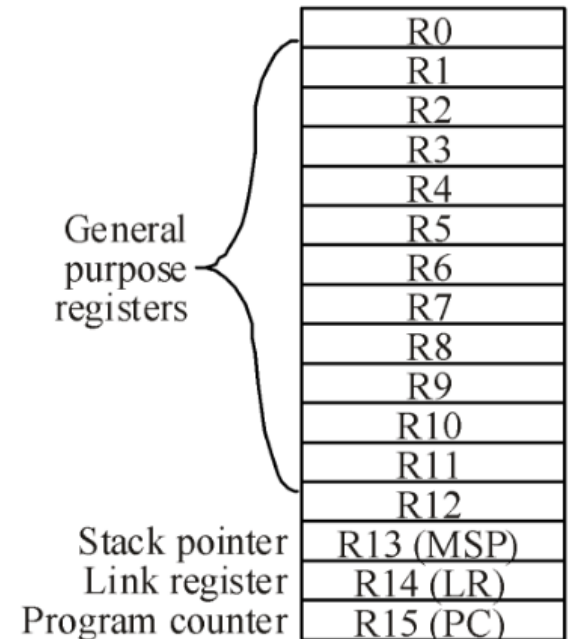


○ Barramentos:

- ICode: busca *opcodes* da ROM;
- DCode: lê dados constantes da ROM;
- System: lê/escreve dados da RAM ou I/O, busca *opcode* da RAM;
- PPB: lê/escreve dados de periféricos internos;
- AHB: lê/escreve dados de portas I/O de alta velocidade;

Registradores (32 bits)

- Registradores R0 a R15
 - 13 de uso geral;
 - R13 → MSP (*Main Stack Pointer*)
 - Aponta para o topo da pilha
 - R14 → LR (*Link Register*)
 - Guarda o endereço de retorno para funções
 - R15 → PC (*Program Counter*)
 - Aponta para a próxima instrução a ser buscada da memória;
 - Processador busca a instrução que está no PC e incrementa o PC.

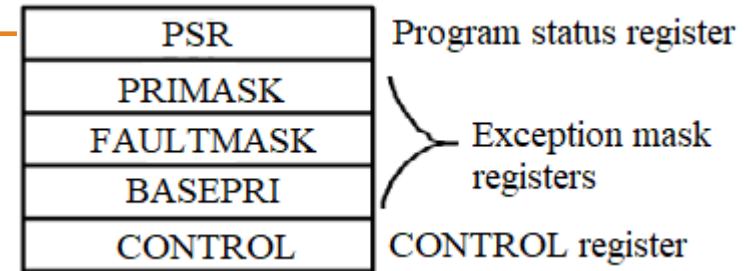


Registradores (32 bits)

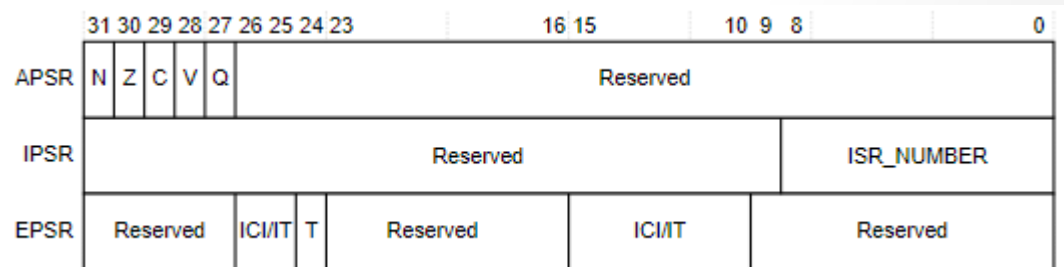
- Registradores Especiais

- PSR (*Program Status Register*);

- APSR → *Application*
 - IPSR → *Interrupt*
 - EPSR → *Execute*



Bits do APSR	Indicação
N	Resultado é negativo
Z	Resultado é zero
V	<i>Overflow</i> com sinal
C	<i>Carry</i> ou <i>overflow</i> sem sinal
Q	Saturação



Memória

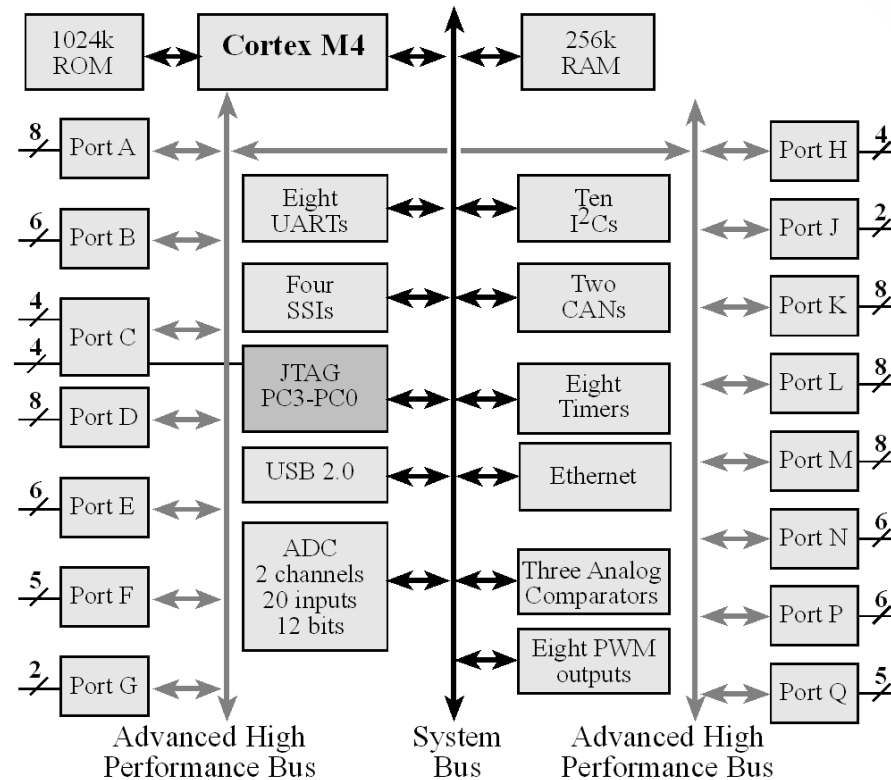
- Endereça até 4GB de memória
 - 32 bits;
 - Em geral:
 - Flash ROM → começa em 0x0000.0000
 - RAM → começa em 0x2000.0000
 - I/O → entre 0x4000.0000 e 0x5FFF.FFFF
 - I/O PPB Interno → entre 0xE000.0000 e 0xE004.1FFF

Memória

- TM4C1294

1024k Flash ROM	0x0000.0000 0x000F.FFFF
256k RAM	0x2000.0000 0x2003.FFFF
I/O	0x4000.0000 0x400F.FFFF
I/O PPB Interno	0xE000.0000 0xE004.1FFF

Periféricos TM4C1294



- Cada *port* tem um número de pinos:
 - Exemplo: PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7

Exercícios

- Abrir o *datasheet* do TM4C1294
 - Verificar a seção dos Registradores (páginas 85 a 99)
 - Verificar a seção das Memórias (seção 2.4).

Paradigma CISC x RISC

RISC

- *Reduced Instruction Set Computer*
 - Instruções simples que executam rápido
 - Elevado número de registradores de uso geral
 - Decodificação de instruções com lógica combinacional (tabela)
 - Execução utilizando *pipeline* → um ciclo de clock por instrução

RISC

- *Reduced Instruction Set Computer*
 - Regularidade de tempo de execução
 - Regularidade de tamanho de instrução
 - Redução da área de silício e tempo de projeto
 - Efeito final: melhor desempenho, apesar do número de instruções ser maior por programa

CISC

- *Complex Instruction Set Computer*
 - Conjunto de instruções inicialmente simples
 - Avanços tecnológicos permitiram a fabricação de computadores com mais transistores e menor custo
 - Projetistas optaram por conjuntos de instruções cada vez mais complexos
 - **Intenção:** reduzir a distância semântica entre Assembly e linguagens de alto nível

CISC

- *Complex Instruction Set Computer*
 - Instruções com elevado grau semântico
 - Elevado número de modos de endereçamento
 - Ex: endereçamento indireto em memória
 - Elevado número de ciclos de clock por instrução → redução da frequência de clock
 - Menor número de instruções por programa → menor uso de memória de código
 - Decodificação através de microcódigo → dificulta/impossibilita o uso de *pipeline*

RISC x CISC

RISC	CISC
Conjunto de instruções reduzido	Conjunto de instruções extenso
Instruções semanticamente simples	Instruções semanticamente complexas
Instruções de tamanho fixo	Instruções de tamanho variável
Decodificação simplificada (tabela)	Decodificação complexa (microcódigo)
Execução regular	Cada instrução executa à sua maneira
Instruções requerem o mesmo número de ciclos de clock para executar	Grande variação no número de ciclos de clock por instrução
Possibilita o uso de pipeline	Extremamente difícil/impossível o uso de pipeline

Pipeline (3 Estágios)

1. Busca (*Fetch*)

- Busca da instrução na memória

2. Decodificação (*Decode*)

- Decodificação dos registradores usados na instrução

3. Execução (*Execute*)

- Leitura de registradores;
- Operações lógicas, aritméticas e de deslocamento;
- Escrita em registradores

Pipeline (3 Estágios)

- Situação Ideal

Cycle		1	2	3	4	5	6	7	8	9
Operation										
ADD	F	D	E							
SUB		F	D	E						
ORR			F	D	E					
AND				F	D	E				
ORR					F	D	E			
EOR						F	D	E		

F - Fetch D - Decode E - Execute

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

Pipeline (3 Estágios)

- Efeito de Saltos

Cycle		1	2	3	4	5	6	7	8	9
Address	Operation									
0x8000	BX r5	F	D	E						
0x8002	SUB		F	D						
0x8004	ORR			F						
0x8FEC	AND				F	D	E			
0x8FEE	ORR					F	D	E		
0x8FF0	EOR						F	D	E	
F - Fetch D - Decode E - Execute										

- Pior caso: salto indireto, 3 ciclos de clock para completar o salto (ARM Cortex-M4)