Arquitetura de Microcontroladores

EMB5642 – Aula 2

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Memórias : Tecnologias

Tipo de Memória	Categoria	Apagamento	Mecanismo de Escrita	Volatilidade
RAM	Escrita e Leitura	Eletricamente, em nível de byte	Eletricamente	Volátil
EEPROM	Principalmente leitura			Não Volátil
Flash		Eletricamente em nível de bloco		

µprocessadores x µcontroladores

- Microprocessadores são circuitos integrados que possuem apenas a CPU(unidade central de processamento) dentro deles, os microprocessadores não possuem memória RAM, ROM ou outros periféricos no chip, por isso, eles precisam de um circuito externo com estes periféricos para funcionarem. Ex I7 – Intel e o Turion – AMD.

µprocessadores x µcontroladores

- Os microcontroladores são CIs que possuem uma CPU juntamente com memória RAM, ROM e outros periféricos, todos embarcados em um único chip. Microcontroladores são projetados para aplicações específicas, sendo que cada modelo pode possuir quantidades distintas de memória e periféricos. Ex: 8051, PIC, MSP430 e ARM

µprocessadores x µcontroladores

- Como o microcontrolador (MCU) é desenvolvido para aplicações específicas, eles necessitam de poucos recursos, como quantidade de RAM/ROM e portas de I/O, determinando diretamente o tamanho e influenciando no custo, que é reduzido.
- Os microprocessadores (CPU) são desenvolvidos para aplicações generalistas, por isso precisam de poder de processamento e o circuito externo deve ter flexibilidade, possibilitando upgrades nos componentes externos.

µprocessadores x µcontroladores

- Normalmente a velocidade do clock de um microprocessador é consideravelmente maior do que as dos MCUs, por exemplo: as MCUs operam numa faixa que vai de poucos MHz até 50-80MHz, os microprocessadores operam acima de 1.5 GHz. ARMs A15, que são MCUs ou até SoC (system on a chip) operam na faixa dos 1.5 GHz também.

Memórias e MCUs

- Nas MCUs estão presentes todos os tipos de memória já citados em apenas um chip. Internamente a CPU da MCU estão os registrados, porém em menor número.
- Em alguns MCUs da ARM (ARM920T) existem memórias do tipo cache separadas (1 dado + 1 instrução), outras famílias com PIC16 ou PIC18 e MSP não possuem cache.

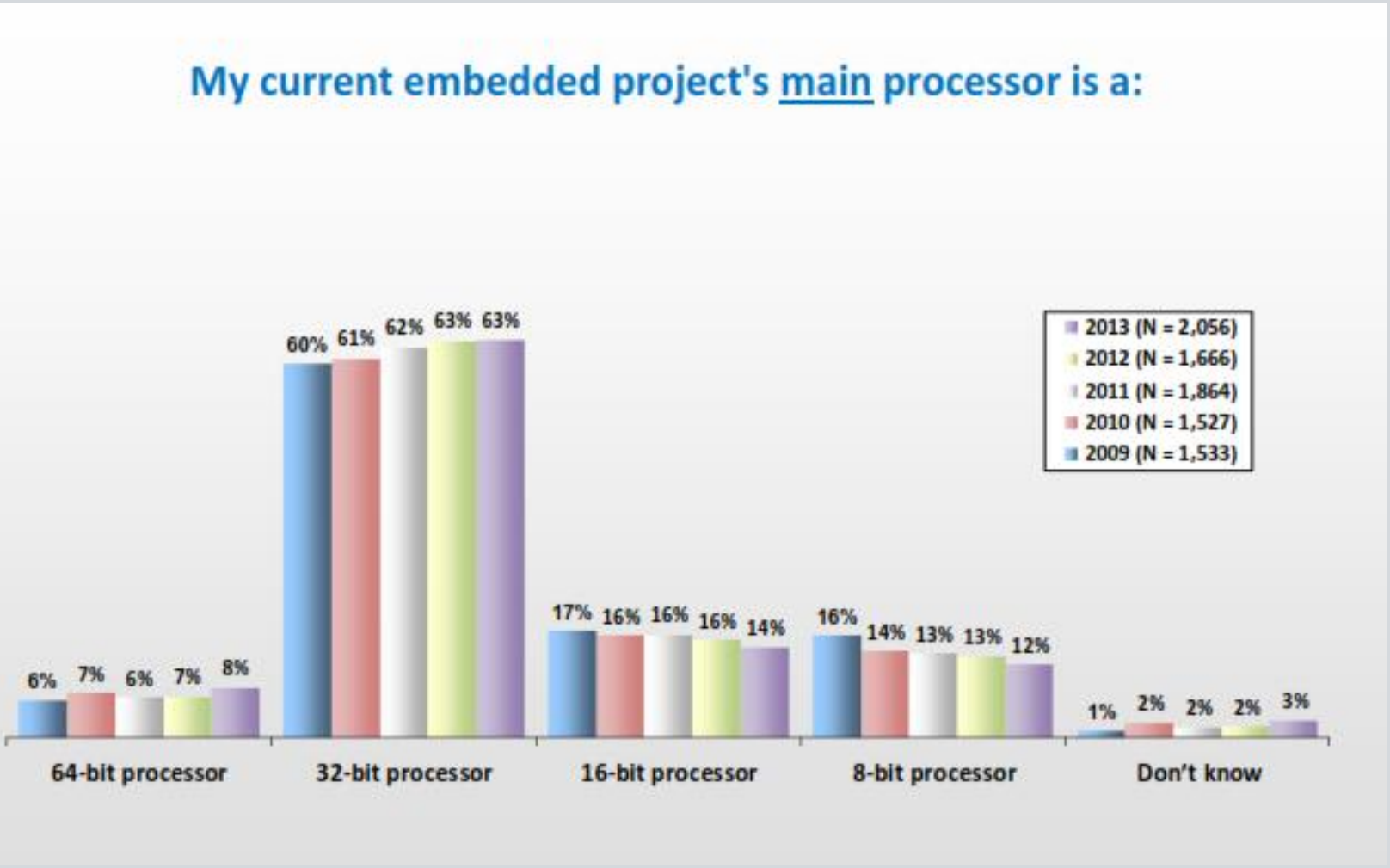
Memórias e MCUs

- O papel da memória principal é desempenhado por memória SRAM existentes no chip em quantidades reduzidas, indo da ordem de unidades de kB até centenas de kB normalmente.
- Já para o armazenamento permanente é utilizado uma memória flash, no caso do ARM7 (indo de 32kB até 512kB), onde são armazenados os dados e sequência de instruções.

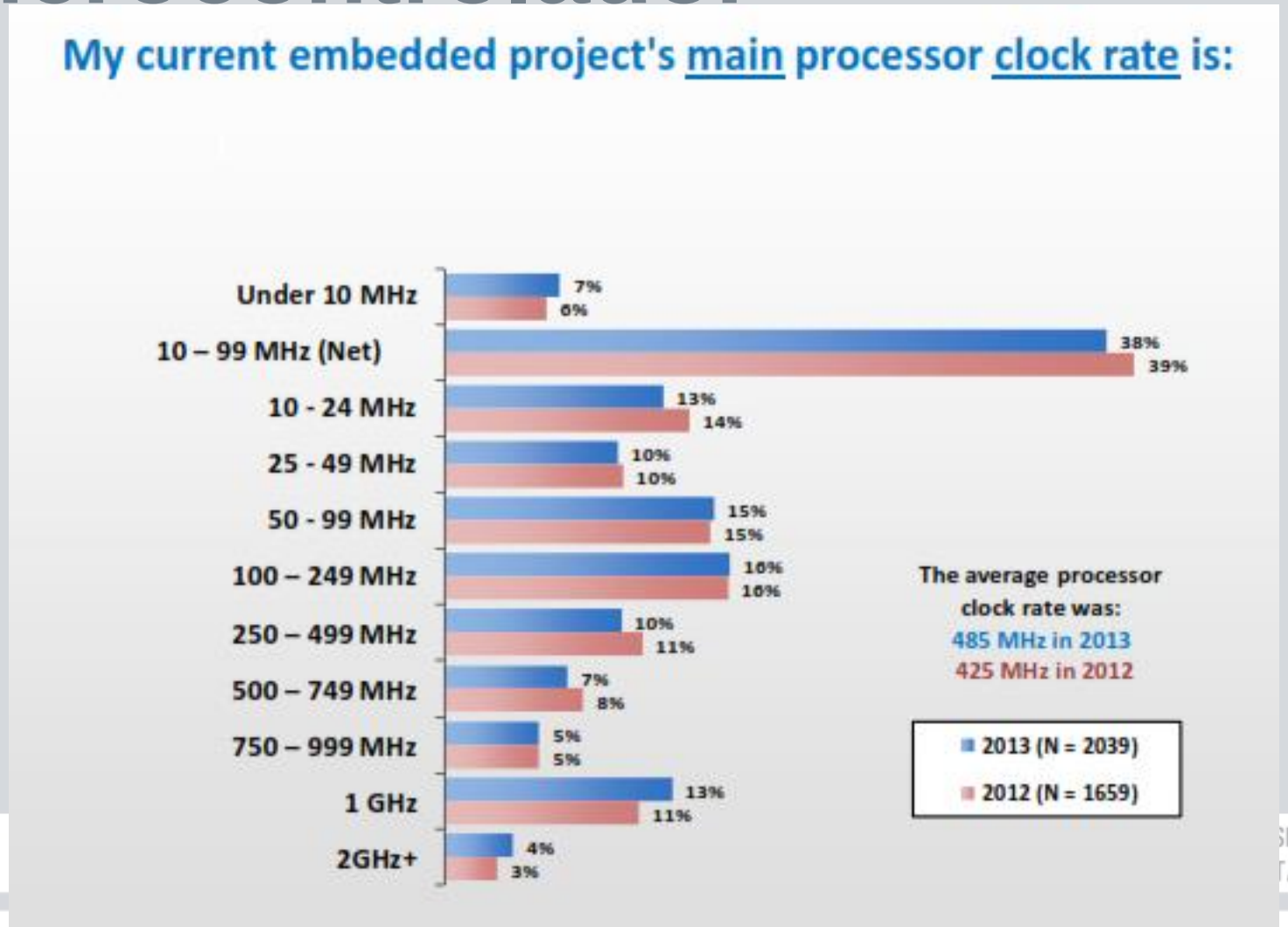
Memórias e MCUs

- Há alguns MCUs que dispõe de uma quantidade pequena de EEPROM (dezenas a centenas de bytes) para o usuário armazenar dados, caso do PIC16 e alguns modelos de MCU ARM.

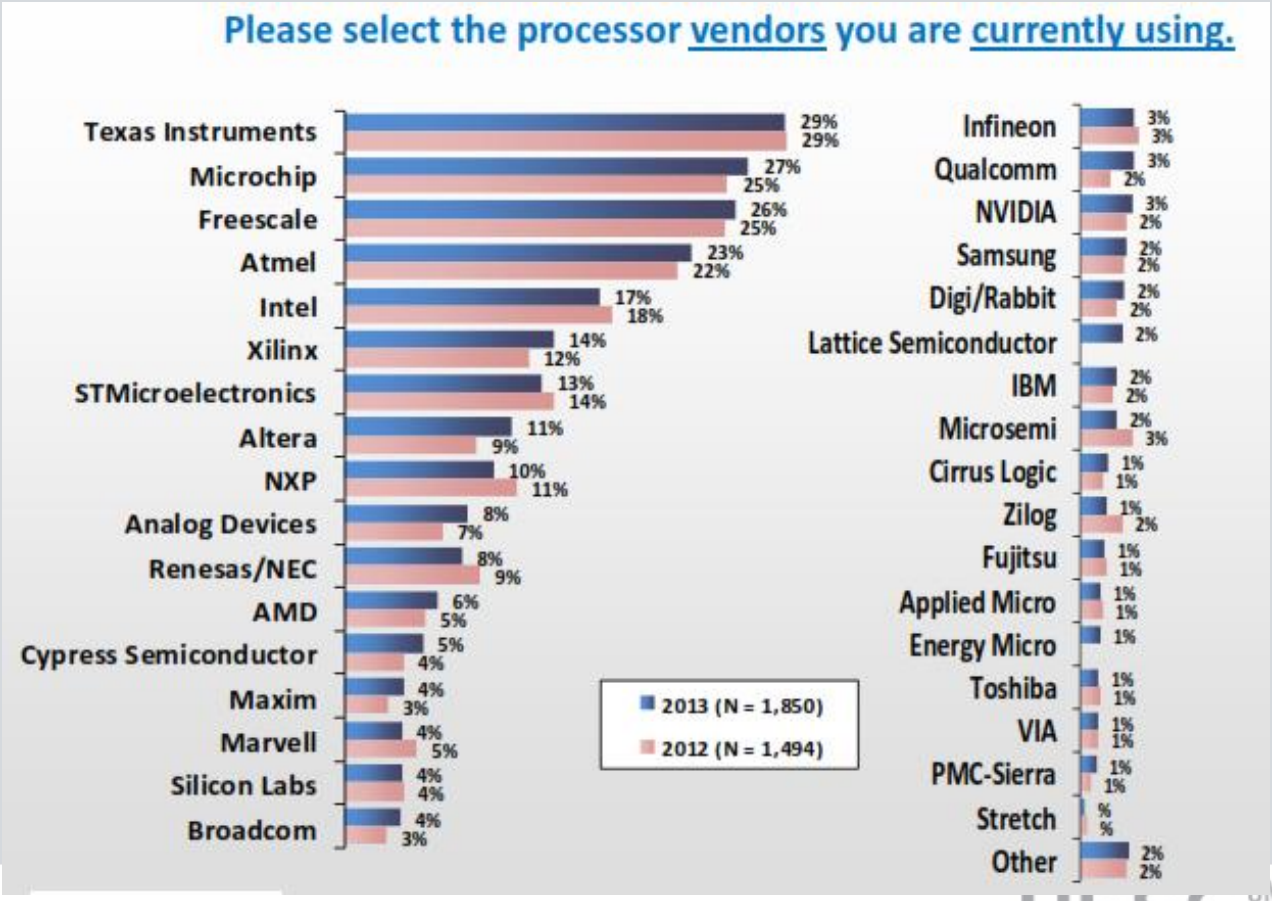
Uso de microcontroladores



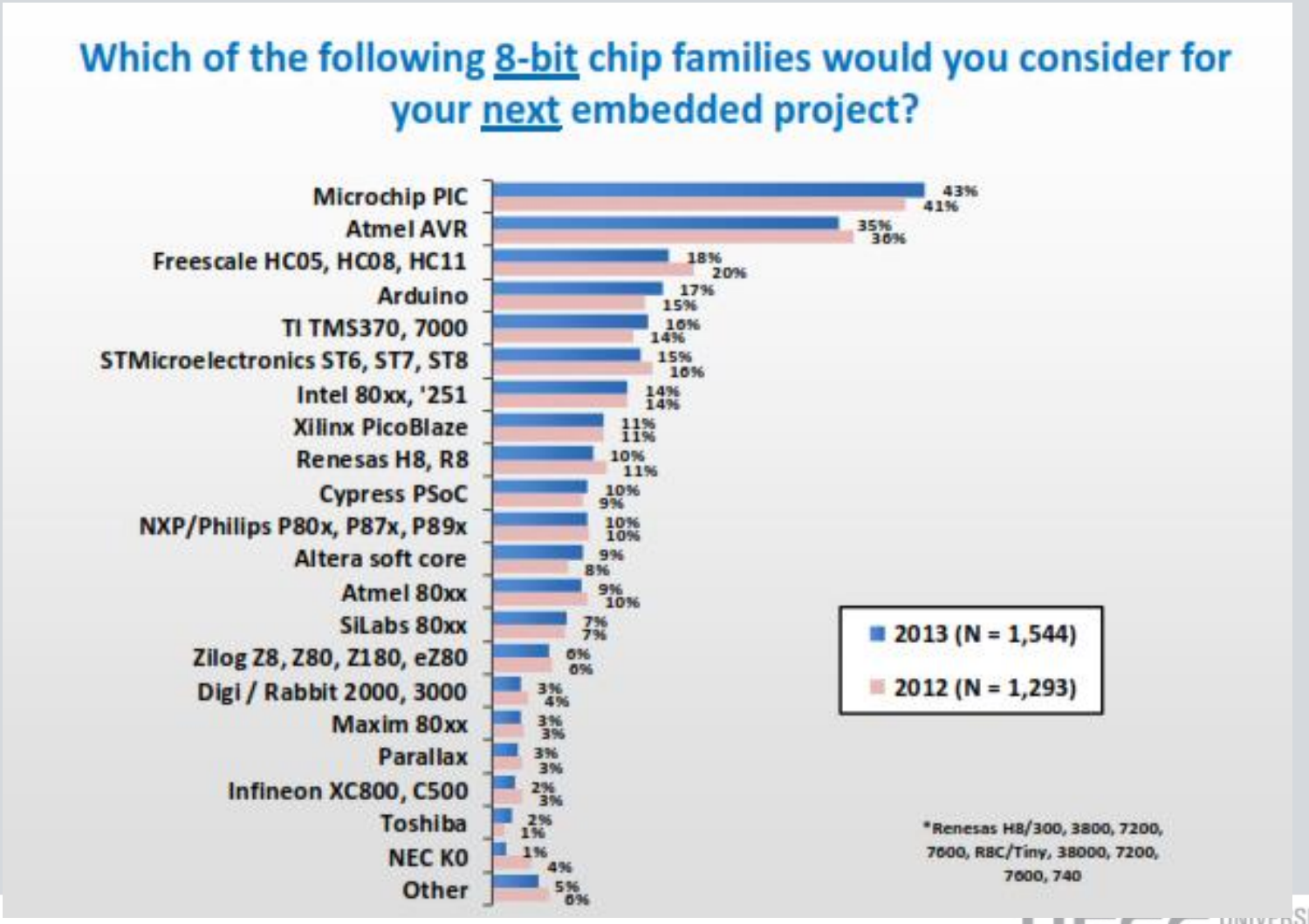
Velocidade do microcontrolador



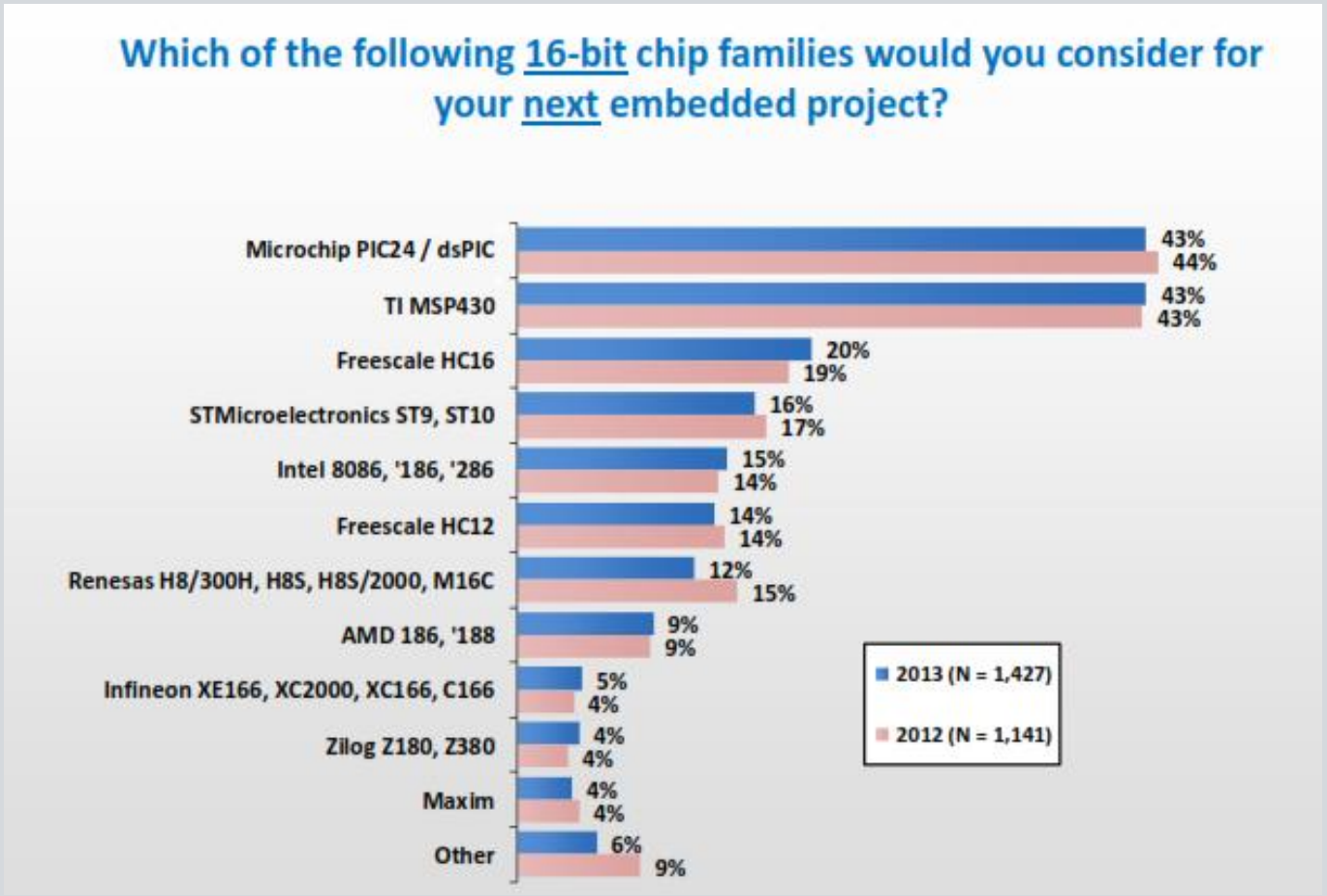
Fabricante de microcontroladores



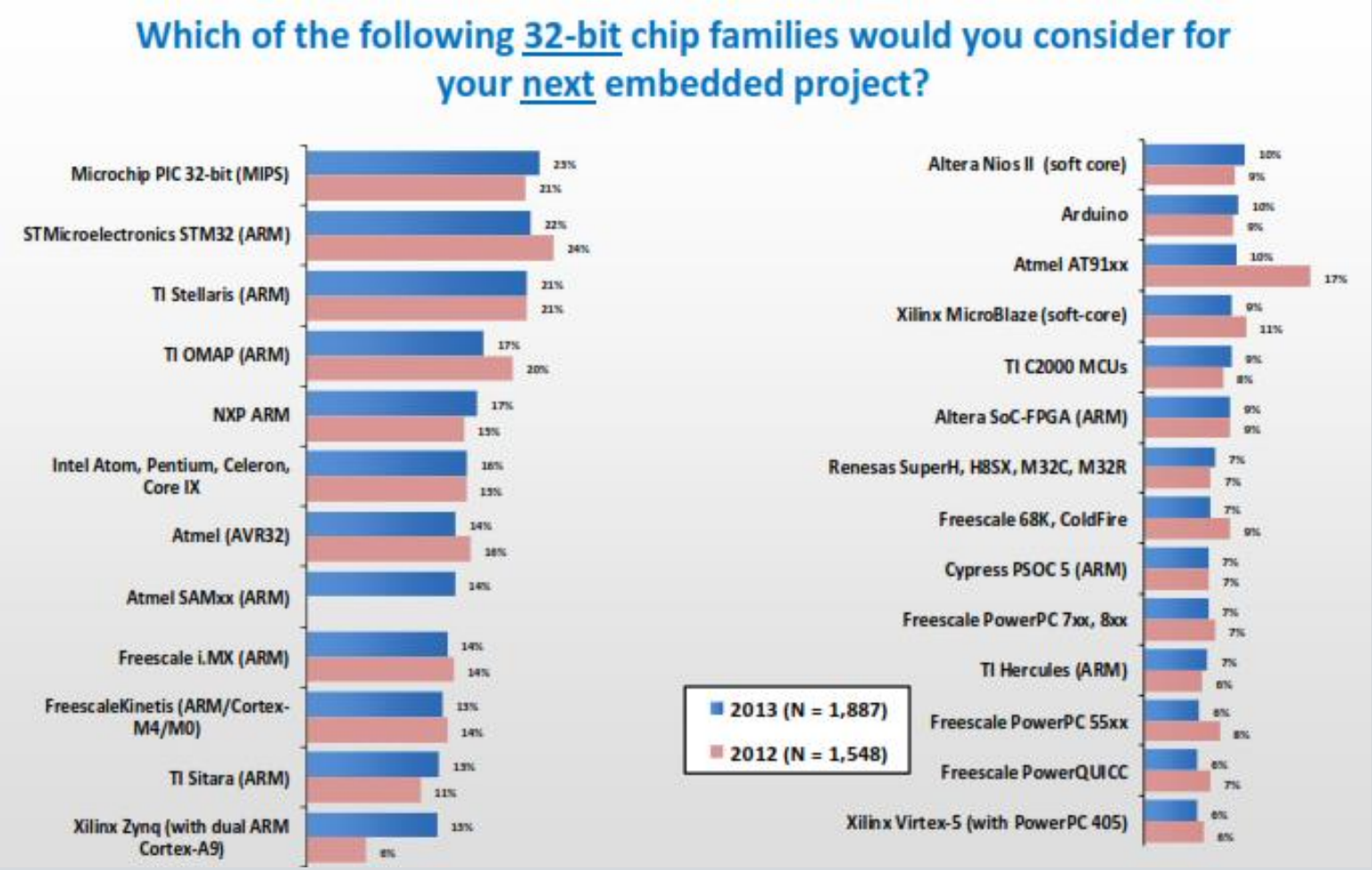
Microcontrolador de 8 bits



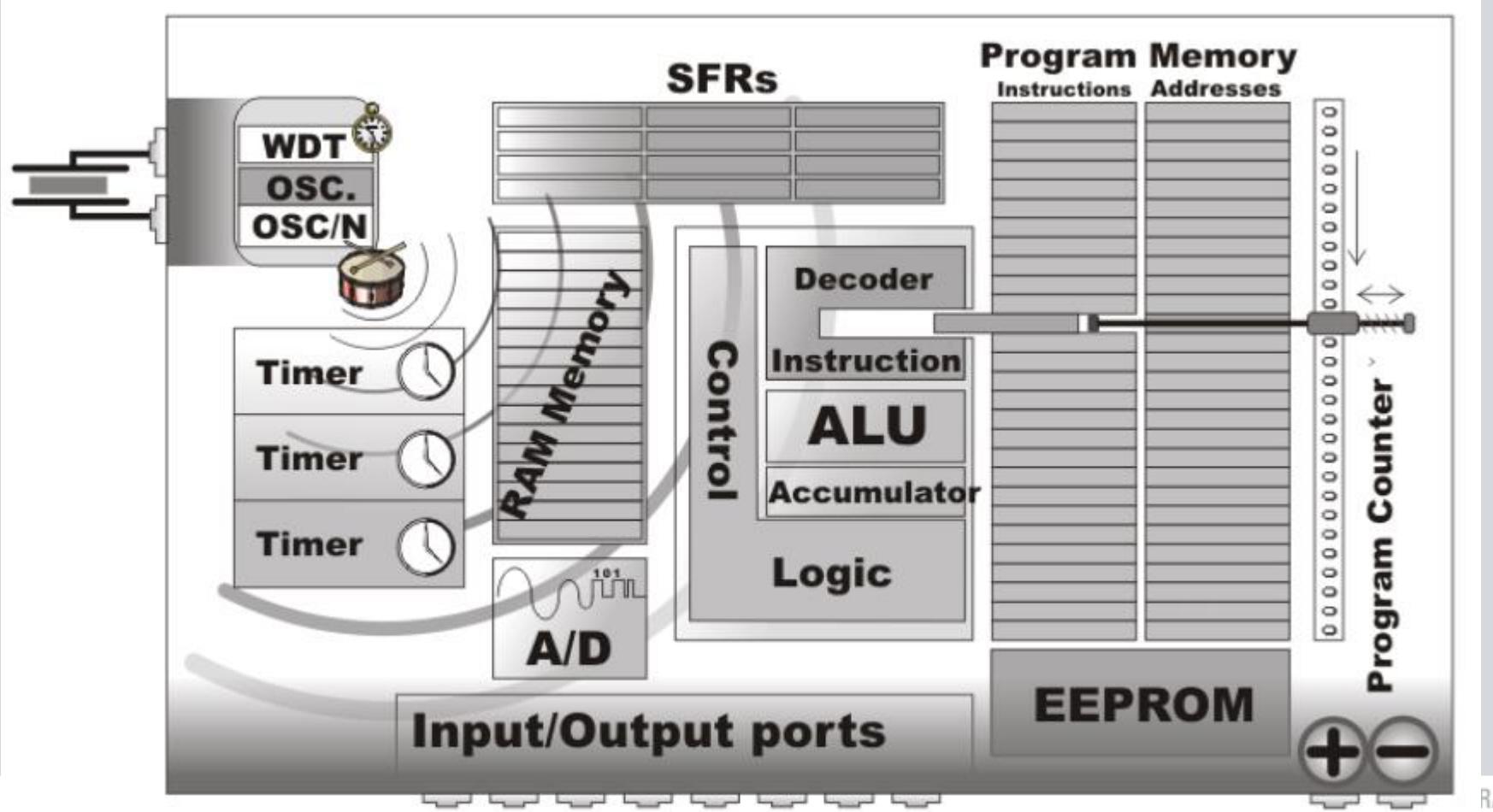
Microcontrolador de 16 bits



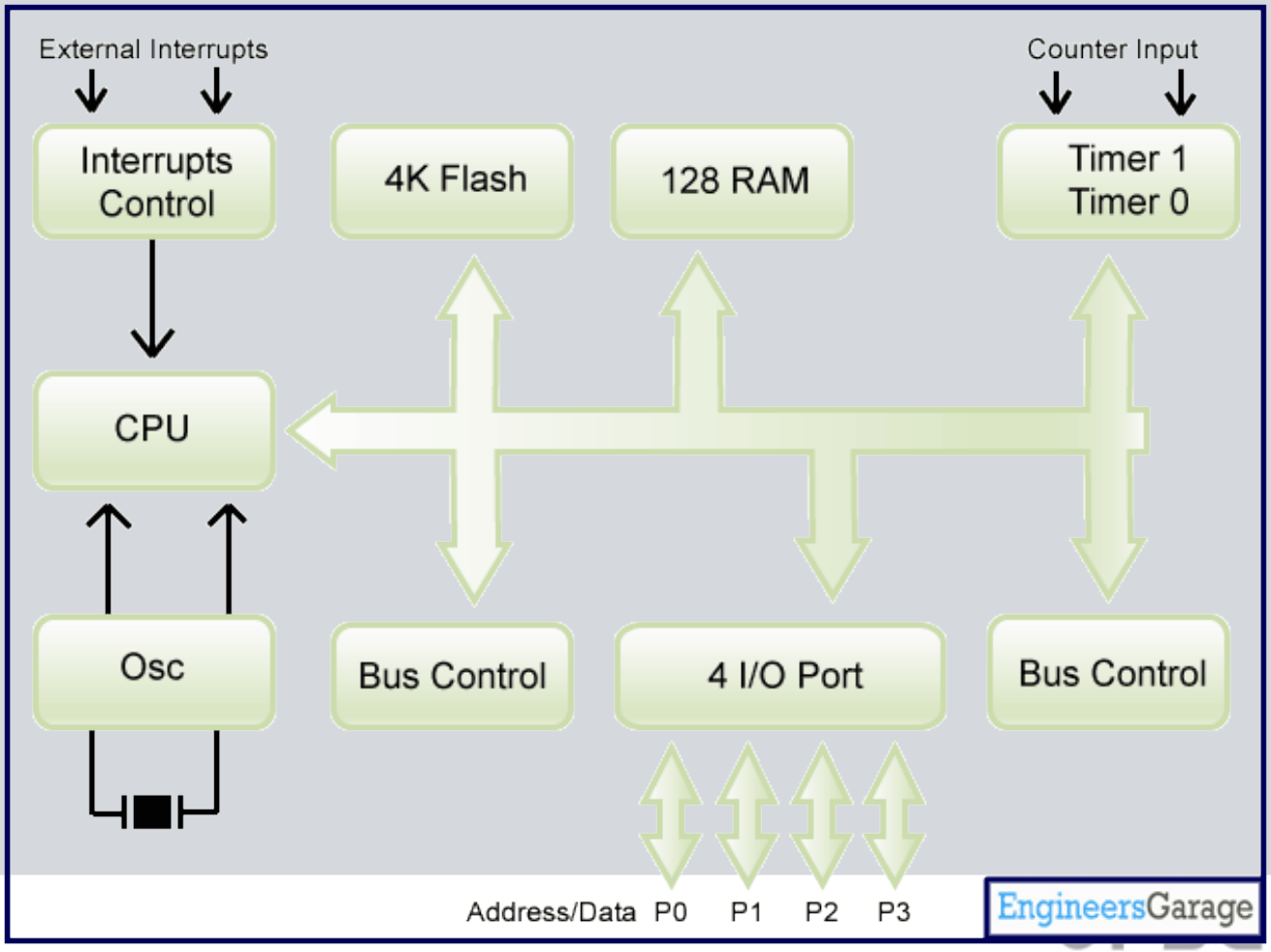
Microcontrolador de 32 bits



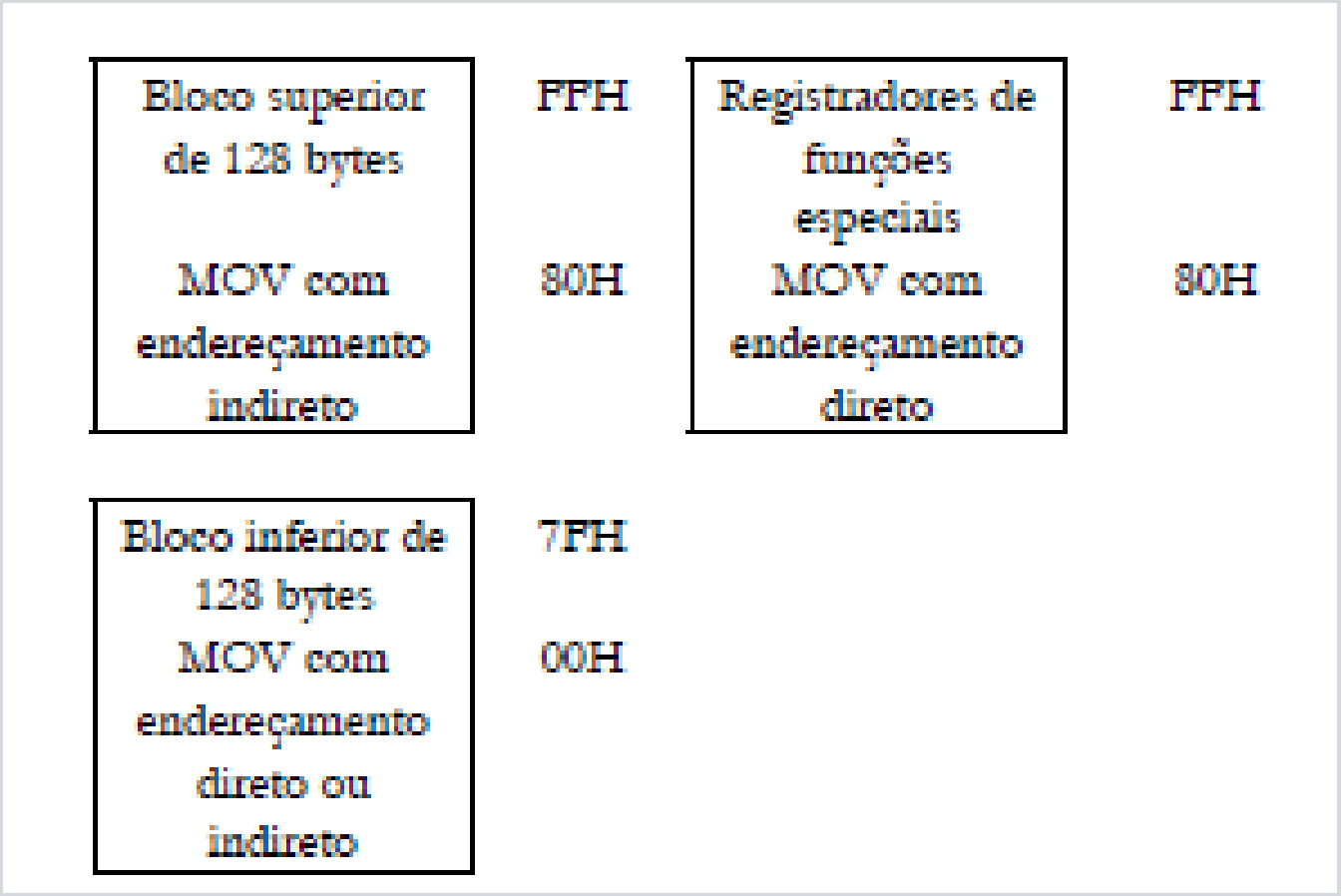
Arquitetura Completa Genérica de um MCU



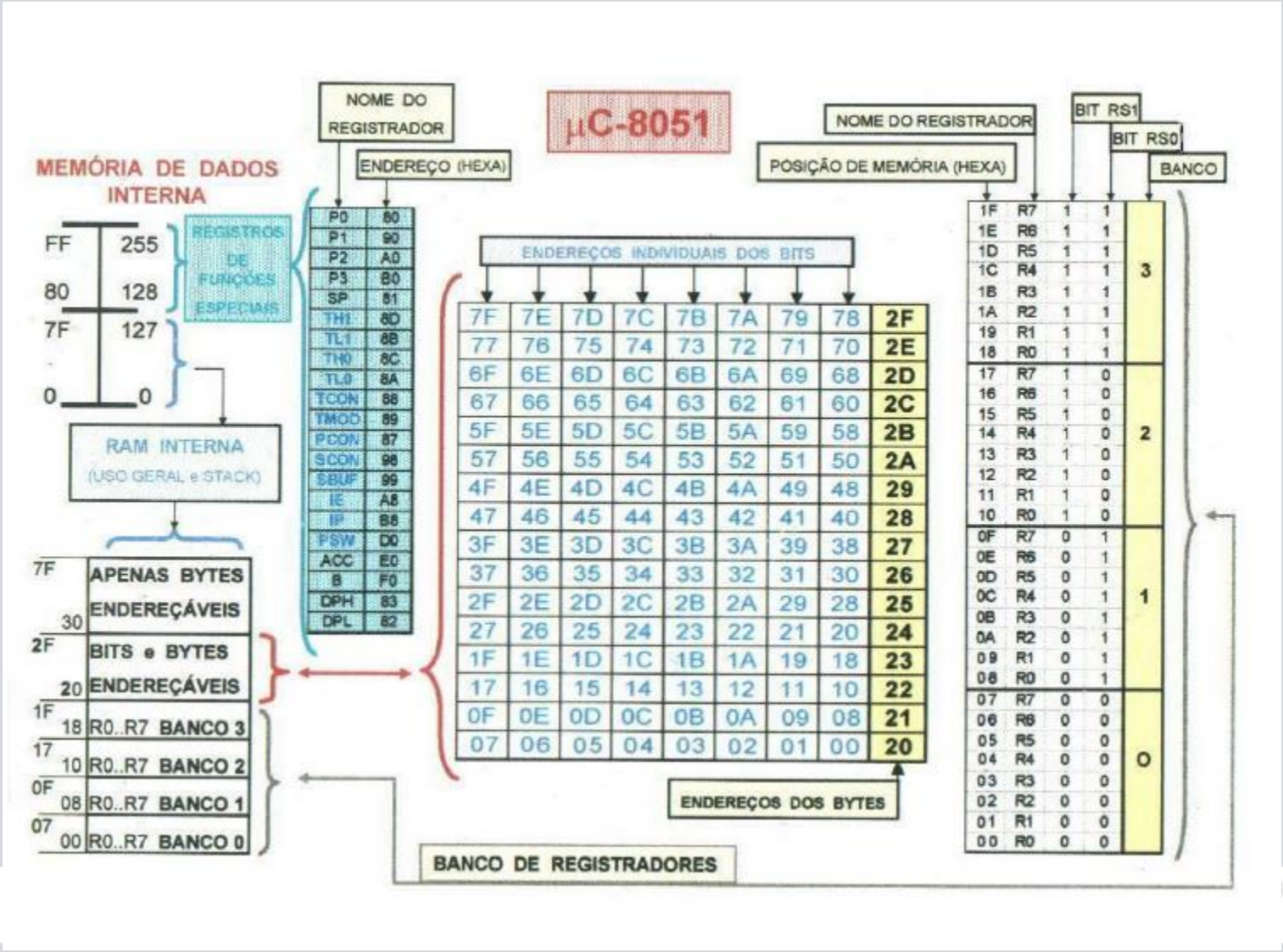
Arquitetura de MCUs – 8051



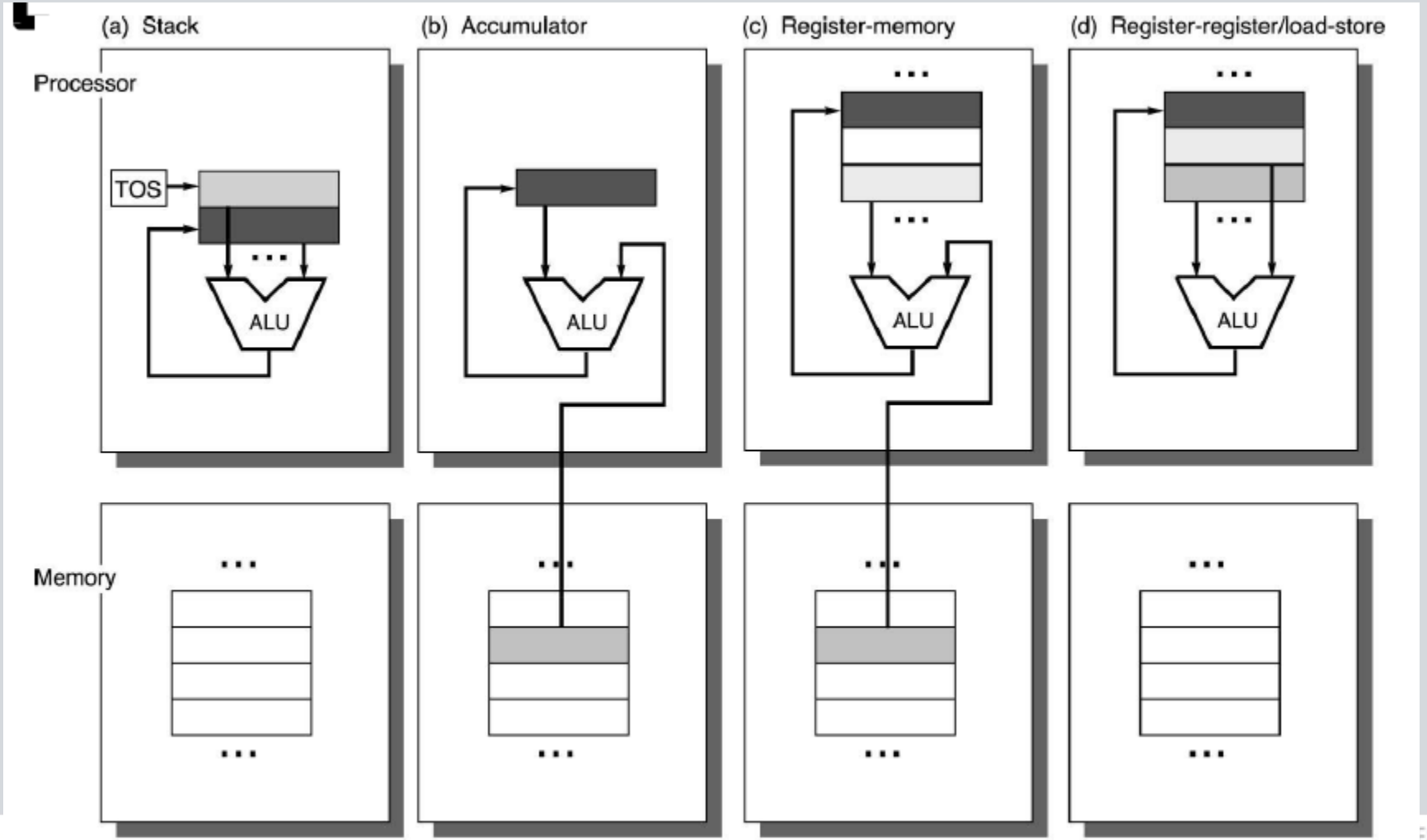
Organização Memória 8051



Organização RAM 8051



Arquitetura Processador



Endereçamento de memória

- Registrador
- Imediato
- Deslocamento
- Indireto de registrador indexado
- Direto ou absoluto
- Indireto de memória

Modo registrador

Exemplo:

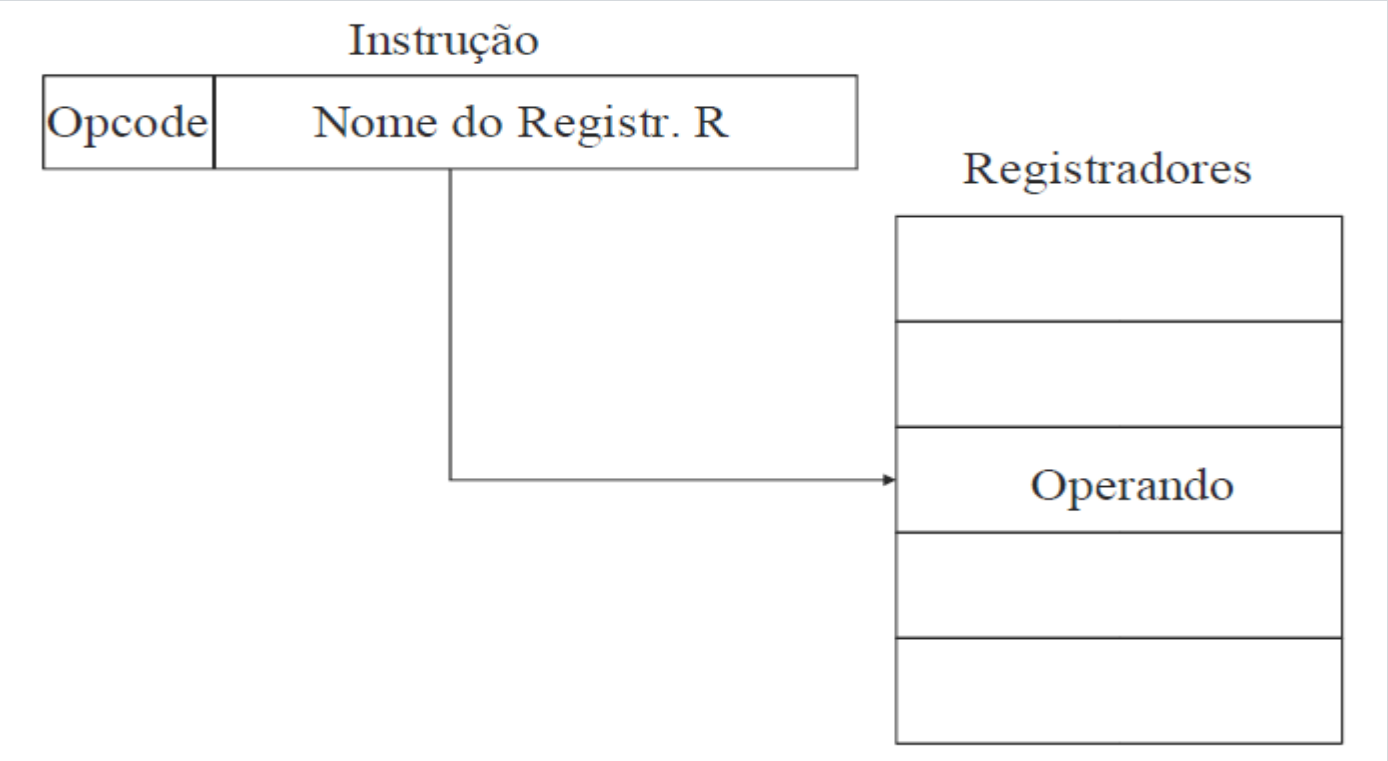
```
Add R4, R3
```

Significado:

```
Regs[R4] ← Regs[R4] + Regs[R3]
```

Usando quando um ou mais operandos está em registradores (inclusive o operando de destino)

Modo registrador



Modo registrador

- **Modo por registrador direto**
  O operando aponta para um registrador, o qual contém o dado.

  Modo por registrador indireto
  O operando aponta para um registrador, o qual contém um endereço de memória (ponteiro) para o dado.

Endereçamento de memória

Vantagens

Maior velocidade / rapidez de execução - o acesso ao registrador é muito mais rápido que o acesso à memória. Economia de espaço de armazenamento de instrução (o tamanho da instrução é menor porque como são poucos registradores, são menos bits para seus endereços).

Desvantagem

Pequeno número de registradores - se forem muitos os dados endereçados por registrador, os registradores disponíveis podem não ser suficientes.

Modo Imediato

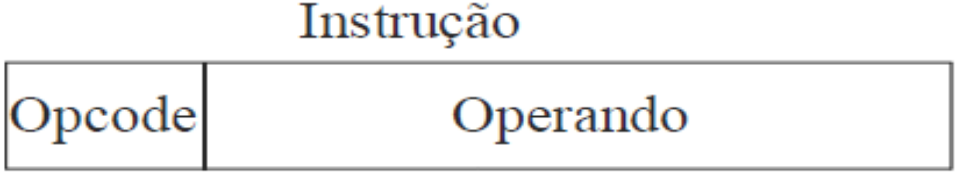
Exemplo:

```
Add R4, #3
```

Significado:

```
Regs[R4] ← Regs[R4] + 3
```

Usando quando um dos operandos é uma constante (codificada no programa)



Modo Imediato

- O valor do campo operando é o próprio dado.

- É usado para trabalhar com valores constantes. O operando é dito operando imediato (o operando é o próprio valor a ser operado, ou seja, é o próprio dado a ser processado).

Modo Imediato

Vantagem

O operando é obtido durante o ciclo de busca, em apenas 1 acesso. Não é necessário fazer nenhum acesso à MP no ciclo de execução, acarretando maior rapidez na execução.

Modo Imediato

Desvantagens

a) Este modo de endereçamento não permite flexibilidade para alterar dados que variam a cada execução do programa, portanto não é adequado para variáveis repetidamente operadas com diferentes valores a cada execução do programa.

b) O tamanho do dado fica limitado ao número de bits do operando (campo operando da instrução). A limitação de tamanho do campo operando reduz o valor máximo do dado que pode ser armazenado.

Modo Direto ou Absoluto

Exemplo:

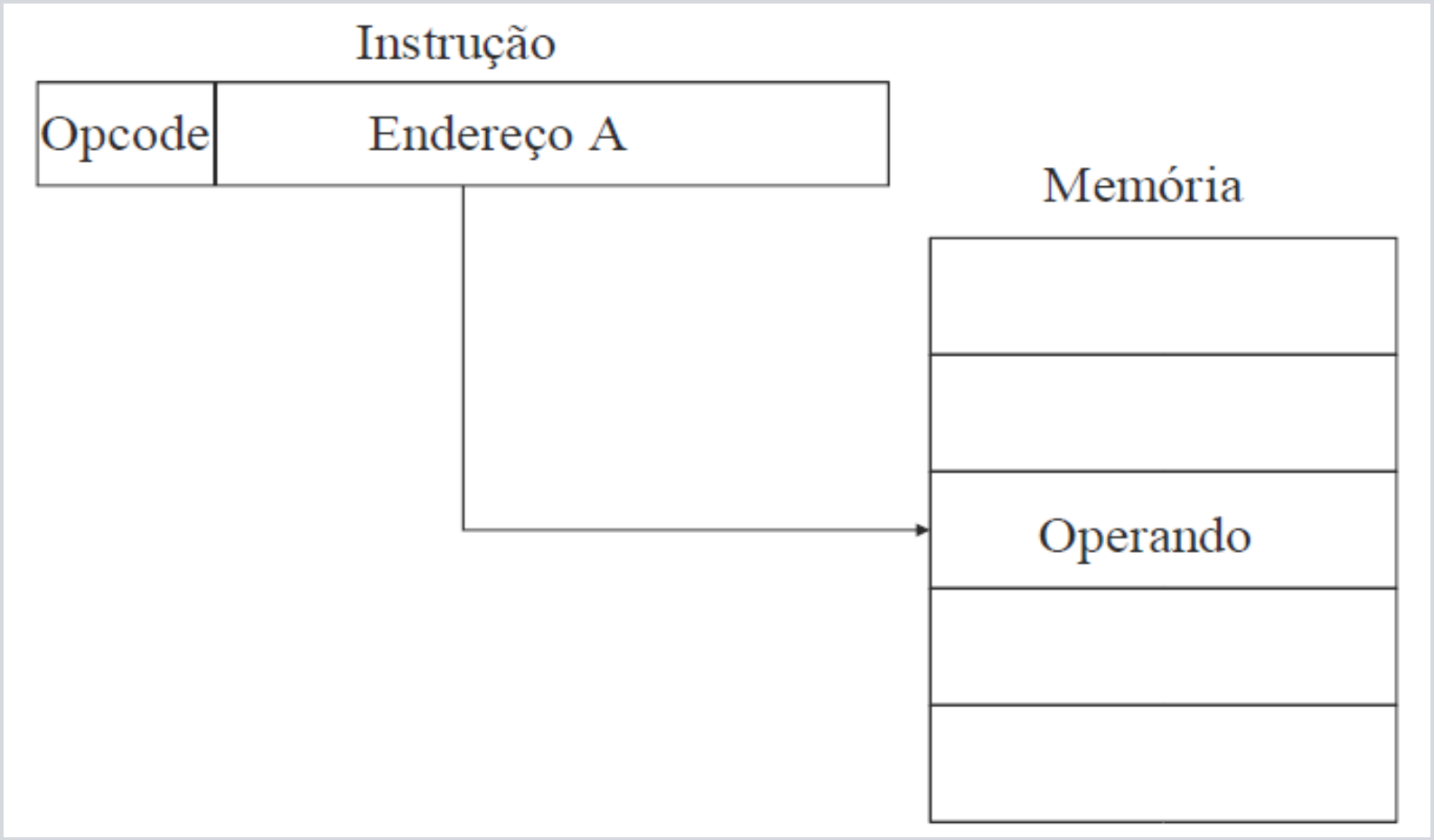
```
Add R1, (1001)
```

Significado:

```
Regs[R1] ← Regs[R1] +
           Mem[1001]
```

Usando no acesso a dados alocados estaticamente na memória (endereço constante)

Modo Direto ou Absoluto



Modo Direto ou Absoluto

Vantagens

a) É aplicado em mais situações que o modo imediato;
b) Requer apenas uma referência à memória para busca do dado (além de uma para a busca da instrução), sendo mais rápido que o modo indireto.

Desvantagens

a) Limitação do endereço da MP que pode ser indicado pelo tamanho do campo operando.
b) É mais lento que o modo imediato.

Utillização

Quando o dado varia de valor a cada execução

Modo indireto de memória

Exemplo:

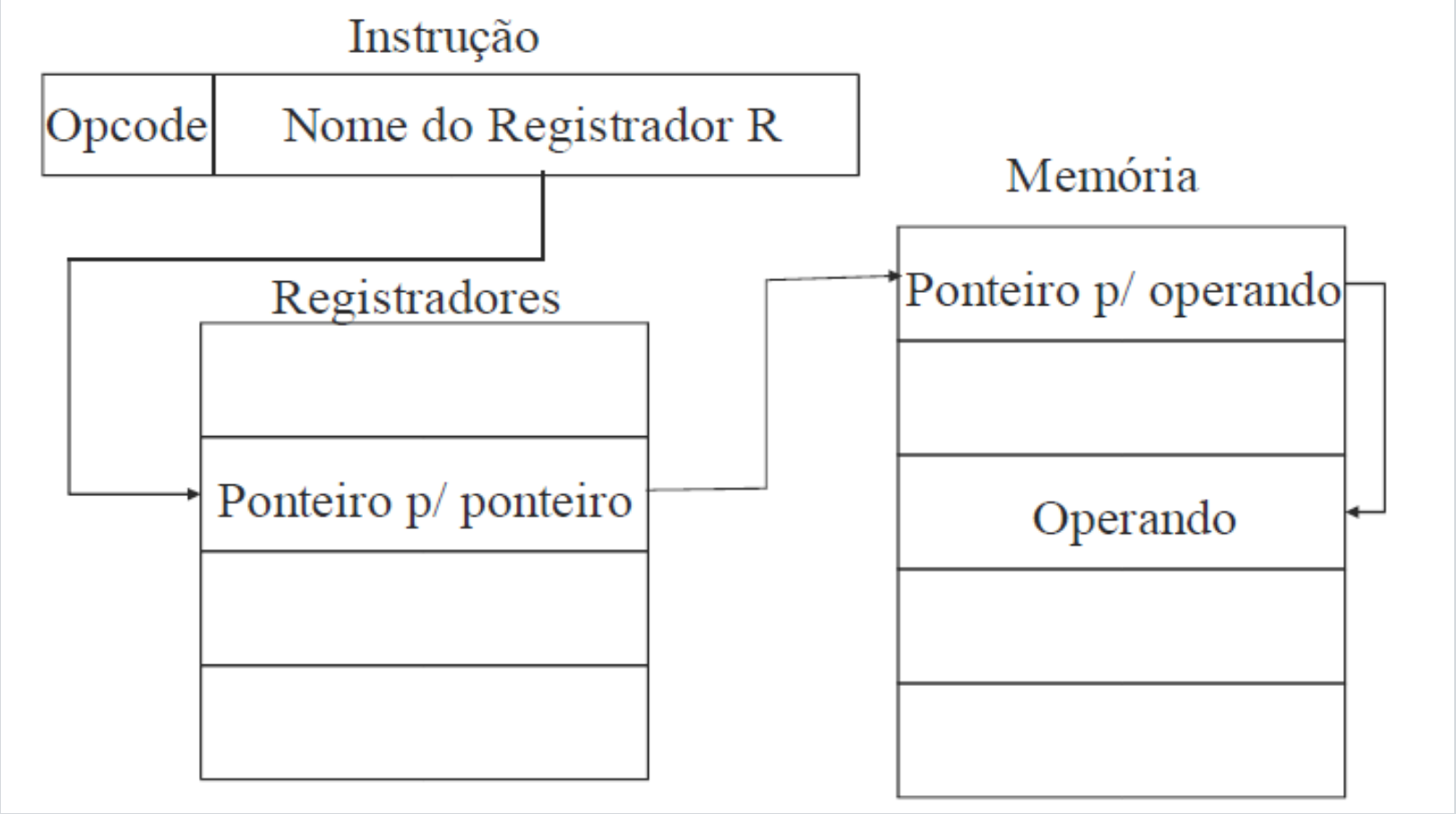
```
Add R1, @(R3)
```

Significado:

```
Regs[R1] ← Regs[R1] +
           Mem[Mem[Regs[R3]]
```

Usando para de-referenciar um ponteiro

Modo indireto de memória



Endereçamento de memória

Vantagem

a) Permite implementar estruturas de organização de dados mais complexas, mais sofisticadas.
b) Elimina a limitação de células endereçáveis.

Desvantagem

Requer maior quantidade de acessos à MP para completar o ciclo de execução da instrução, acarretando que o tempo requerido para a execução da instrução é maior.

Obs.1: É possível haver várias indireções. Em algumas máquinas, existe 1 bit que sinaliza no caso de existirem várias Indireções. Enquanto este bit for 0, continua com as indireções, até encontrá-lo ligado.

Modo indireto de memória

Utilização
Manutenção de ponteiro de dados

Exemplo:
Relação de dados a serem movimentados para novas posições de memória (por exemplo, elementos de vetores), basta modificar o valor da célula endereçada pela instrução (não é necessário mudar o valor do operando).

Modo indireto registrador

Exemplo:

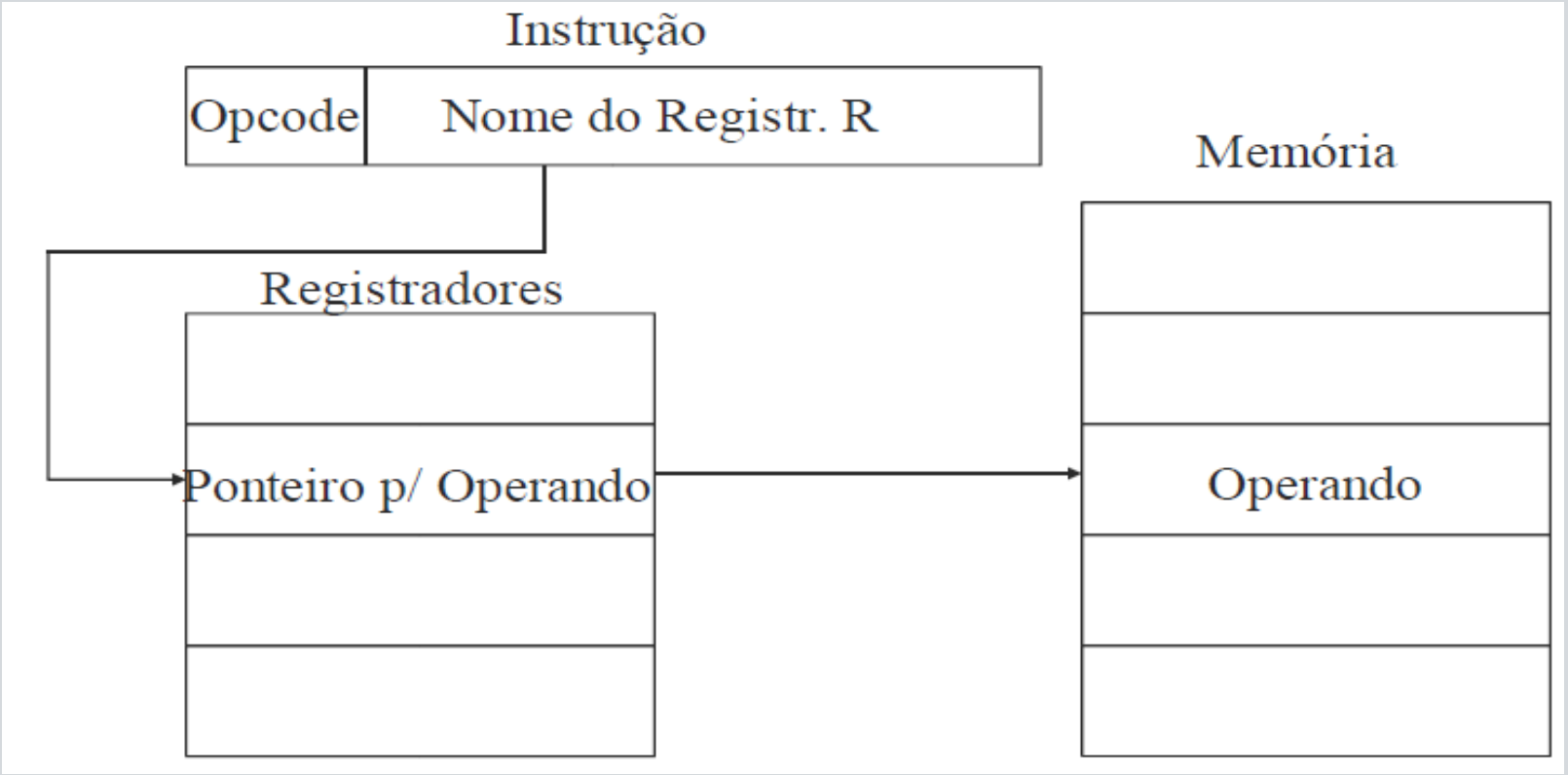
```
Add R4, (R1)
```

Significado:

```
Regs[R4] ← Regs[R4] +
           Mem[Regs[R1]]
```

Usando à memória com uso de ponteiros para cálculo de endereço

Modo indireto registrador



Modo indireto registrador

Vantagens

Maior velocidade / rapidez de execução - o acesso ao registrador é muito mais rápido que o acesso à memória.
Economia de espaço de armazenamento de instrução (o tamanho da instrução é menor porque como são poucos registradores, são menos bits para seus endereços).

Desvantagem

Não são adequados para transferência de variáveis da MP para ULA.
Pequeno número de registradores - se forem muitos os dados endereçados por registrador, os registradores disponíveis podem não ser suficientes

Modo deslocamento

Exemplo:

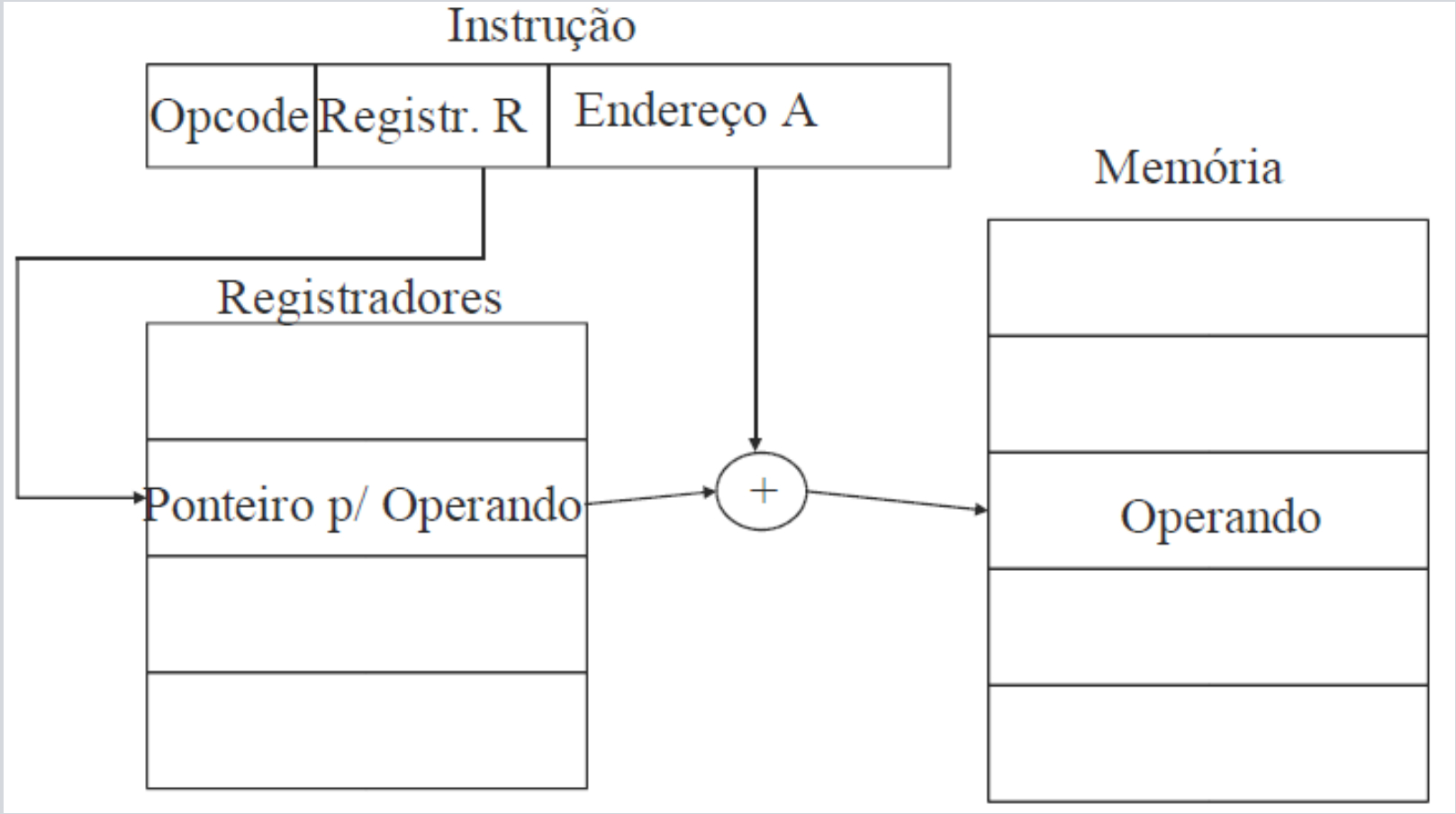
```
Add R4, 100(R1)
```

Significado:

```
Regs[R4] ← Regs[R4] +
           Mem[100+Regs[R1]]
```

Usando para acesso a variáveis locais

Modo deslocamento

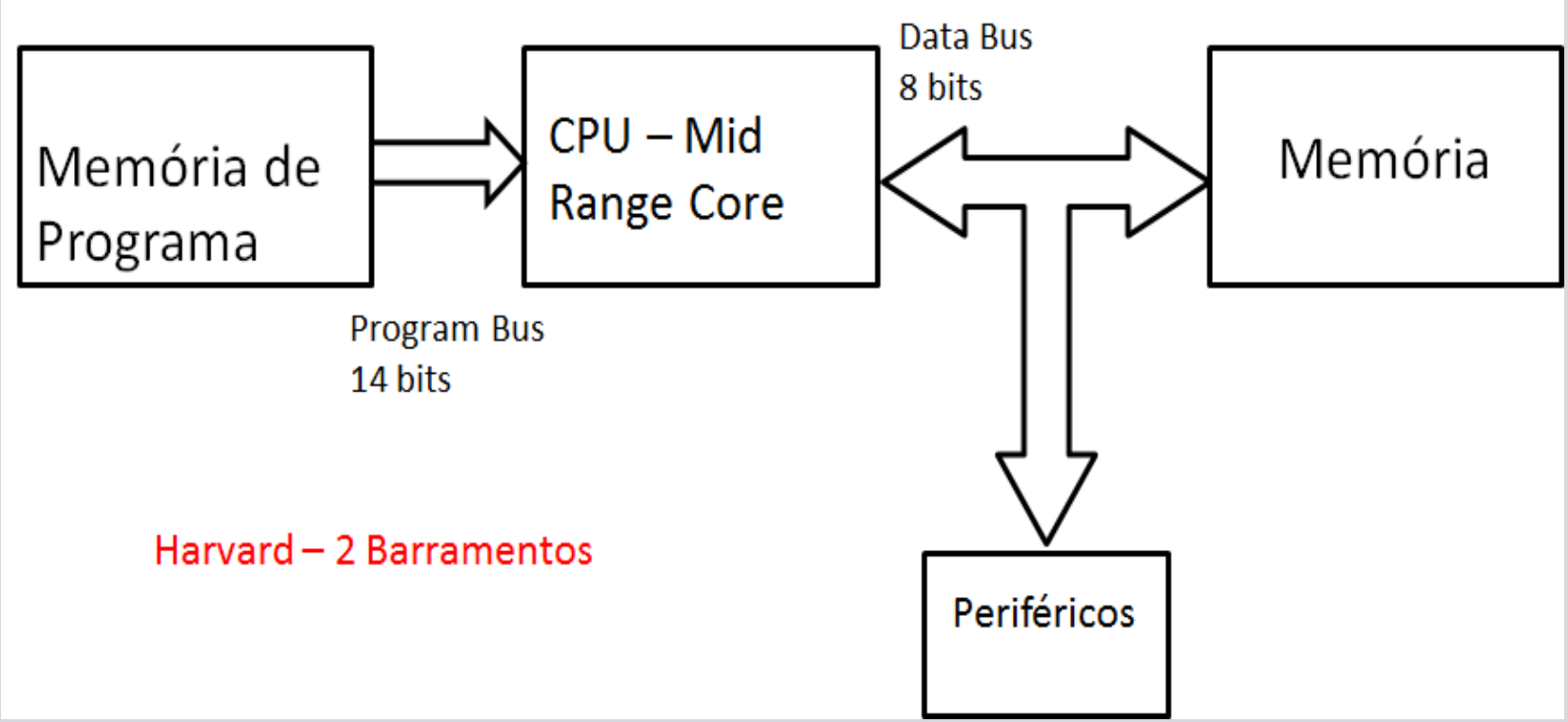


Endereçamento de memória

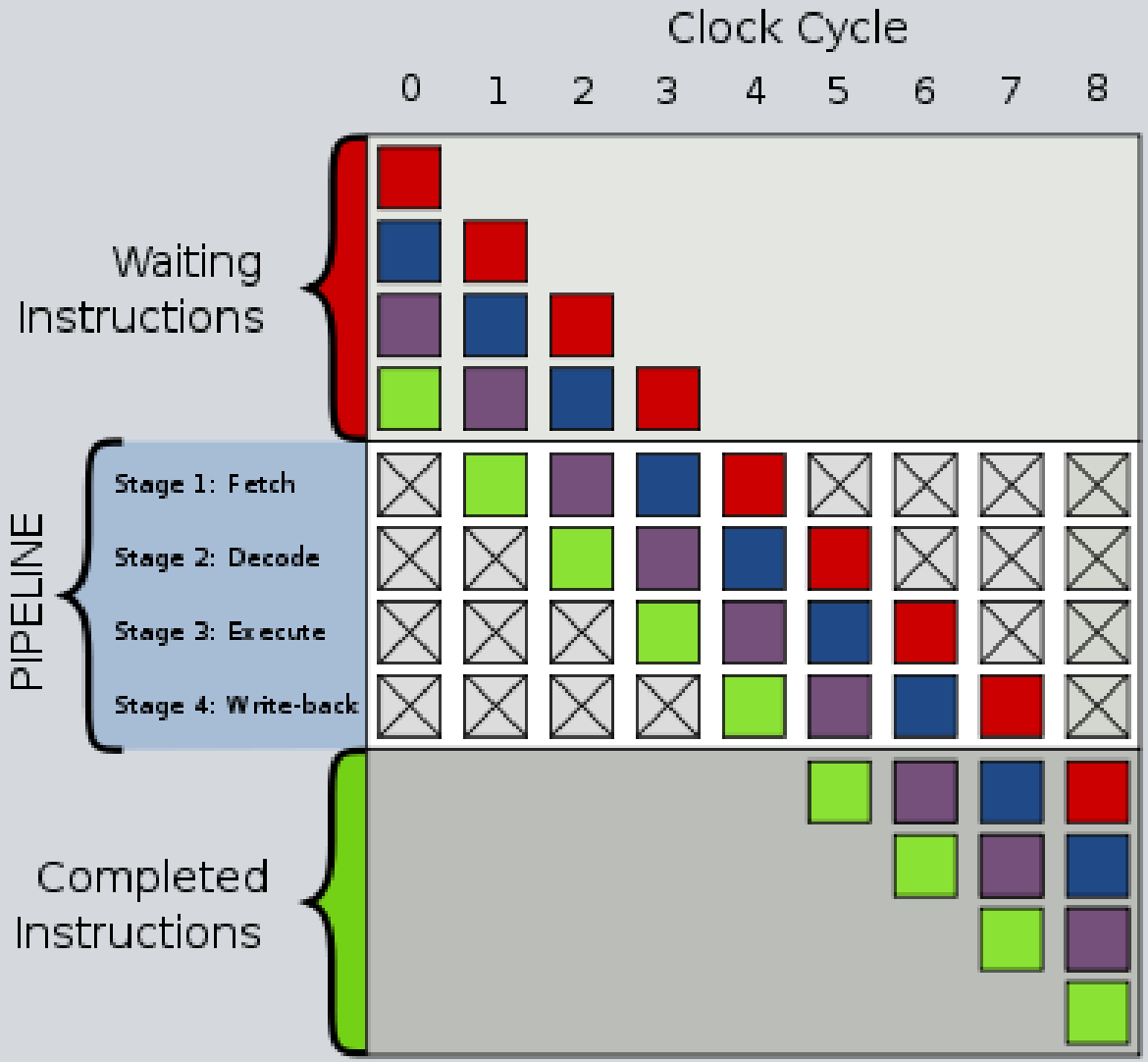
O modo de endereçamento afeta o tempo requerido para executar uma instrução e a memória requerida para seu armazenamento

Instruções que usam endereçamento implícito ou por registrador são executadas muito rápido, pois trabalham direto com o *hardware* do processador e seus registradores. A instrução toda pode ser buscada (*"fetched"*) em um único ciclo de busca (um único acesso à memória). O número de acessos à memória é o mais importante fator no consumo de tempo de execução da instrução.

Arquitetura de MCUs – PIC16

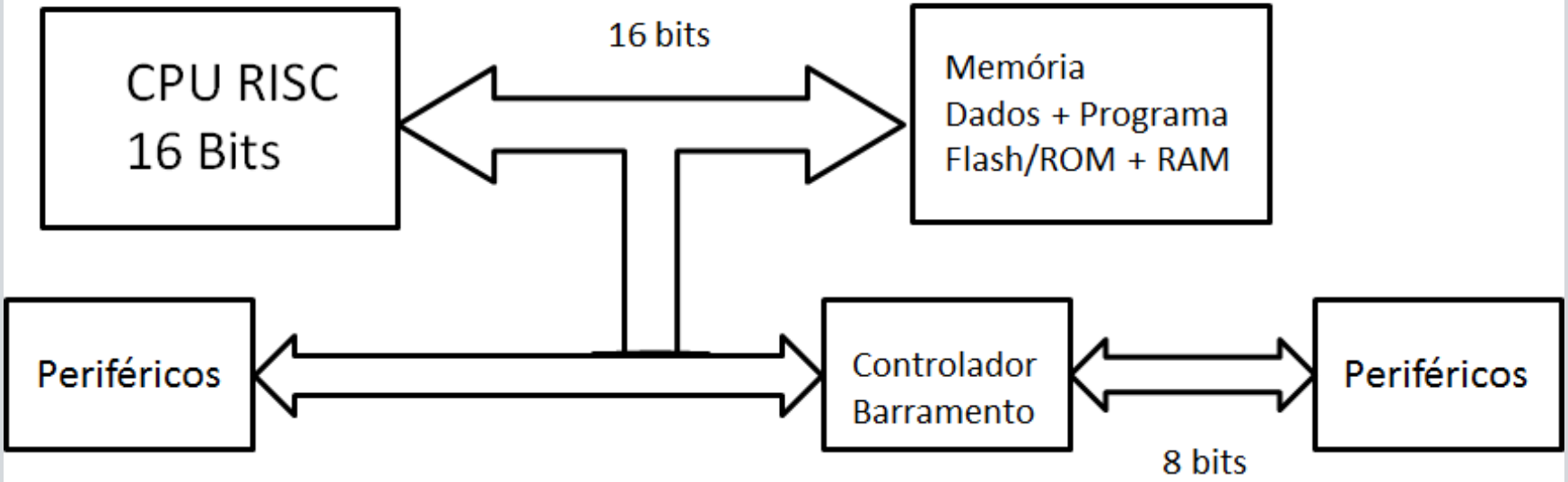


Pipeline


Clock Cycle
0
1
2
3
4
5
6
7
8
Waiting Instructions
PIPELINE
Stage 1: Fetch
Stage 2: Decode
Stage 3: Execute
Stage 4: Write-back
Completed Instructions

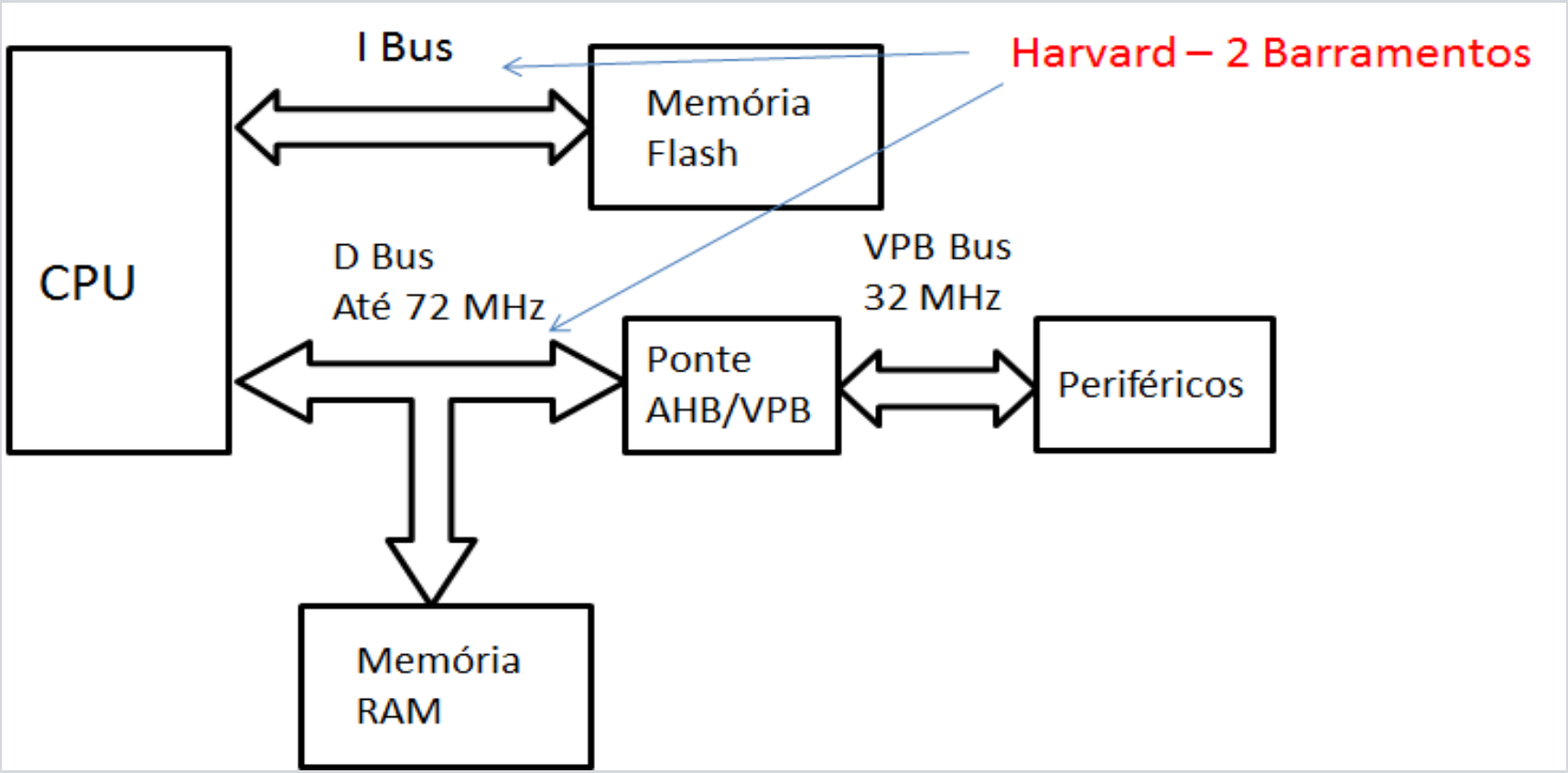
Arquitetura de MCUs – MSP430

Von Neumann – 1 Barramento de dados



Barramento de dados 16 bits
Barramento endereço 16 bits
Barramento controle 8 bits

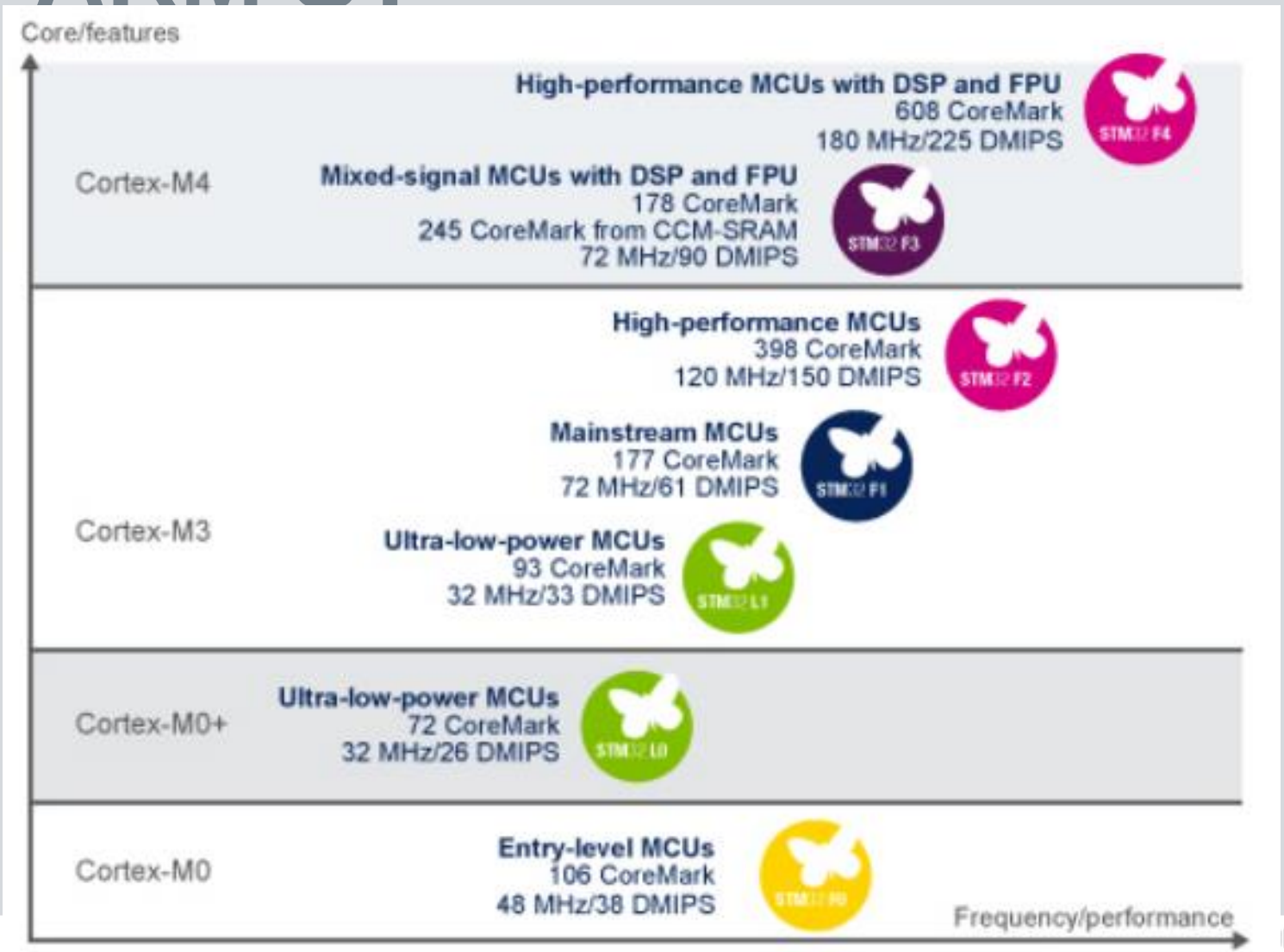
Arquitetura de MCUs – ARM M4F



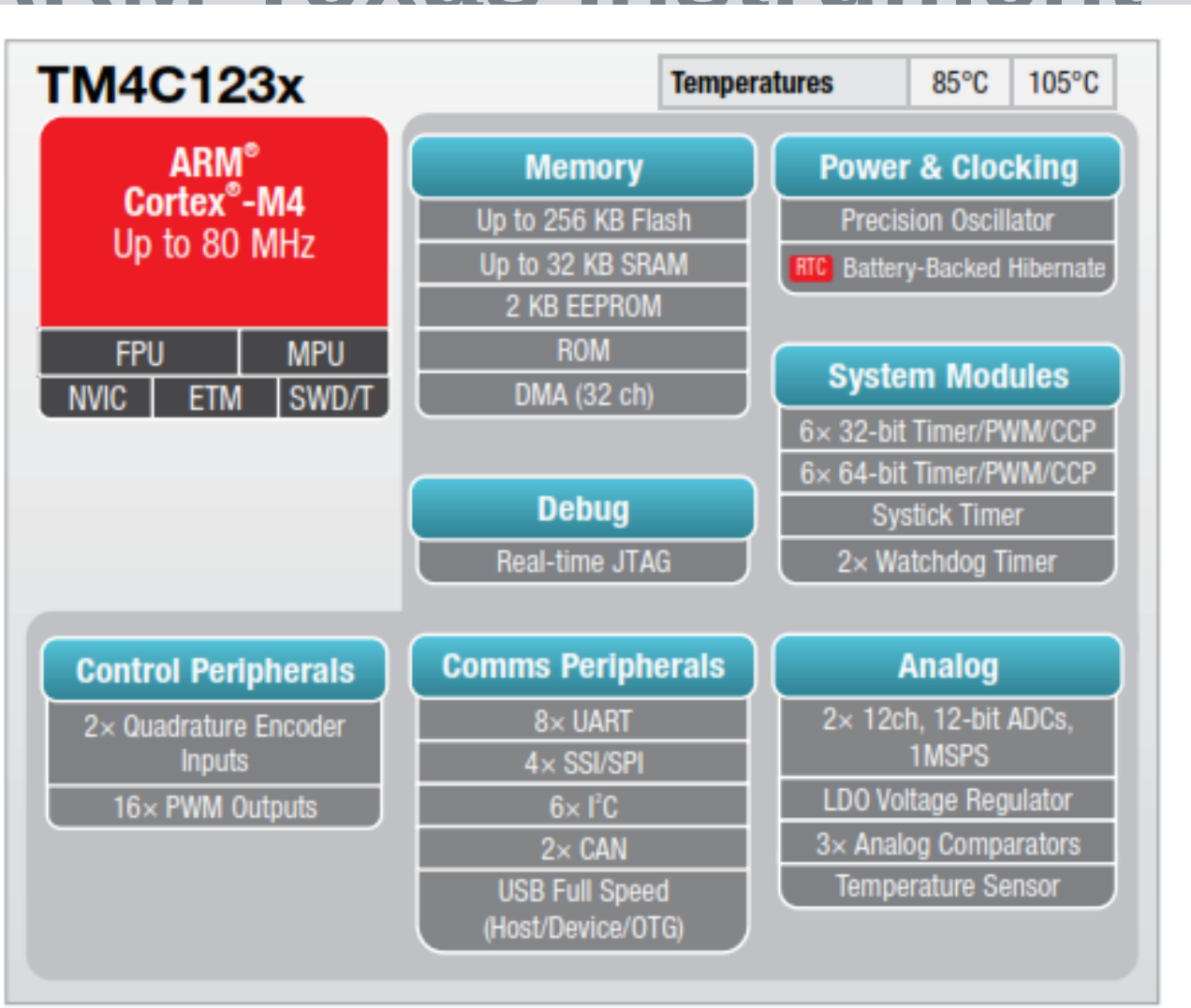
Evolução ARMs

Architecture	Bit width	Cores designed by ARM Holdings	Cores designed by third parties	Cortex profile	References
ARMv1	32/26	ARM1			
ARMv2	32/26	ARM2, ARM3	Amber, STORM Open Soft Core[25]		
ARMv3	32	ARM6, ARM7			
ARMv4	32	ARM8	StrongARM, FA526		
ARMv4T	32	ARM7TDMI, ARM9TDMI			
ARMv5	32	ARM7EJ, ARM9E, ARM10E	XScale, FA626TE, Feroceon, PJ1/Mohawk		
ARMv6	32	ARM11			
ARMv6-M	32	ARM Cortex-M0, ARM Cortex-M0+, ARM Cortex-M1		Microcontroller	
ARMv7-M	32	ARM Cortex-M3		Microcontroller	
ARMv7E-M	32	ARM Cortex-M4		Microcontroller	
ARMv7-R	32	ARM Cortex-R4, ARM Cortex-R5, ARM Cortex-R7		Real-time	
ARMv7-A	32	ARM Cortex-A5, ARM Cortex-A7, ARM Cortex-A8, ARM Cortex-A9, ARM Cortex-A12, ARM Cortex-A15, ARM Cortex-A17	Krait, Scorpion, PJ4/Sheeva, Apple A6/A6X (Swift)	Application	
ARMv8-A	64/32	ARM Cortex-A53, ARM Cortex-A57[26]	X-Gene, Denver, Apple A7 (Cyclone), K12	Application	[27][28]
ARMv8-R	32	No announcements yet		Real-time	[29][30]

uC ARM ST


Core/features
High-performance MCUs with DSP and FPU
608 CoreMark
180 MHz/225 DMIPS
Cortex-M4
Mixed-signal MCUs with DSP and FPU
178 CoreMark
245 CoreMark from CCM-SRAM
72 MHz/90 DMIPS
High-performance MCUs
398 CoreMark
120 MHz/150 DMIPS
Mainstream MCUs
177 CoreMark
72 MHz/61 DMIPS
Cortex-M3
Ultra-low-power MCUs
93 CoreMark
32 MHz/33 DMIPS
Cortex-M0+
Ultra-low-power MCUs
72 CoreMark
32 MHz/26 DMIPS
Cortex-M0
Entry-level MCUs
106 CoreMark
48 MHz/38 DMIPS
Frequency/performance

uC ARM Texas Instrument



Anderson Wedderhoff Spengler

E-mail: anderson.spengler@ufsc.br

Telefone: +55 (48) 3721 7489

UNIVERSIDADE FEDERAL
DE SANTA CATARINA