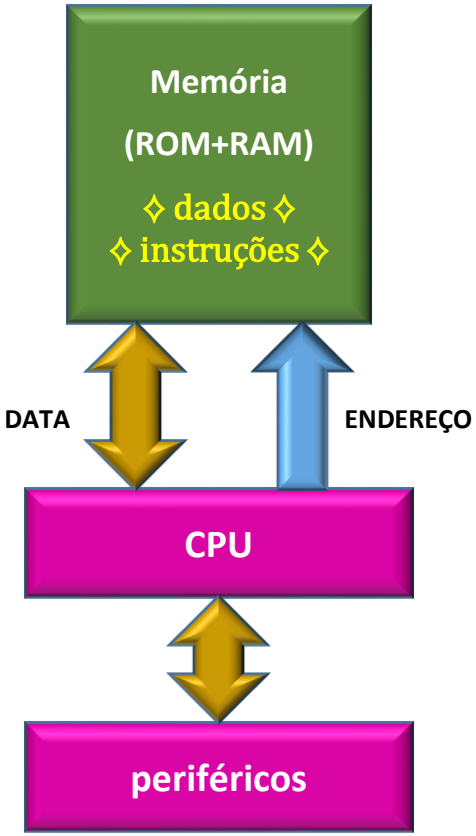
Von Neumann

(Princeton architecture)



O projeto da Unidade de Controle na CPU deve levar em conta que o barramento de dados ora transporta INSTRUÇÕES e ora transporta DADOS, multiplexados no tempo.

A memória externa à CPU tem instruções e dados intercalados, portanto, a unidade de controle tem que coordenadar quando/onde está buscando instrução e quando/onde está buscando ou gravando dados.

Exemplos mais conhecidos de computador x86 e AMD

Vantagens:
→ Não precisa de blocos de memória separados (mais barato)

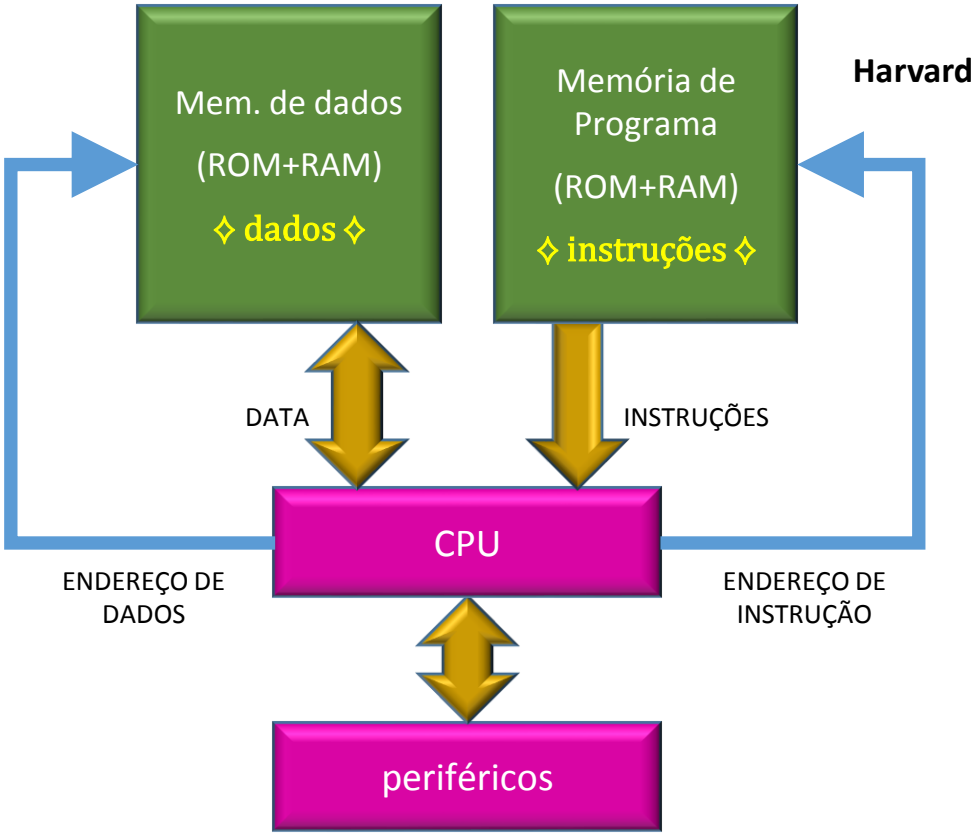
Gargalos:
→ Dados e instruções compartilham o mesmo barramento, portanto há necessidade de dividirem o tempo de uso. Isso diminui a taxa de transferência de dados/instruções (throughput).

Desvantagens:
Por compartilharem o mesmo bus, tendem a ser mais lentos nos algoritmos que fazem busca/gravação constante na memória, algumas vezes solucionados por memória CACHE.

A Unidade de Controle na CPU pode operar os barramentos simultaneamente.

Isso significa que pode ocorrer busca de INSTRUÇÕES ao mesmo tempo em que dados são buscados ou gravados na memória de dados. Operação em "pipeline" pode aumentar a eficiência do processamento.

Por outro lado, implica que a unidade de controle tem que lidar com dois endereçamentos.

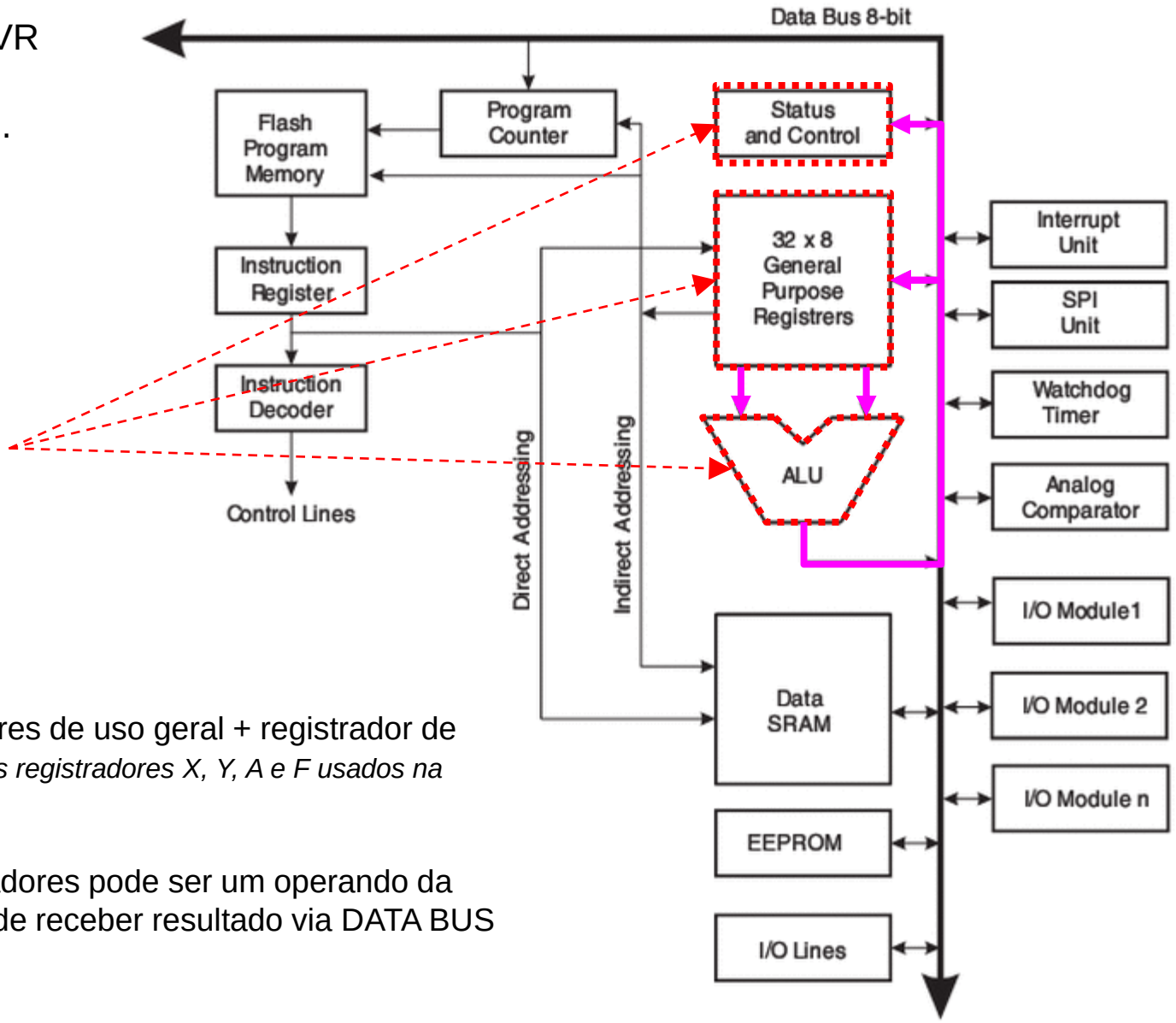


- Diagrama blocos AVR

Identificando Harvard...

Registradores

ULA (ou ALU)



O AVR tem 32 registradores de uso geral + registrador de status (*em vez de apenas os registradores X, Y, A e F usados na aula passada...*)

Qualquer desses registradores pode ser um operando da ALU e qualquer outro pode receber resultado via DATA BUS

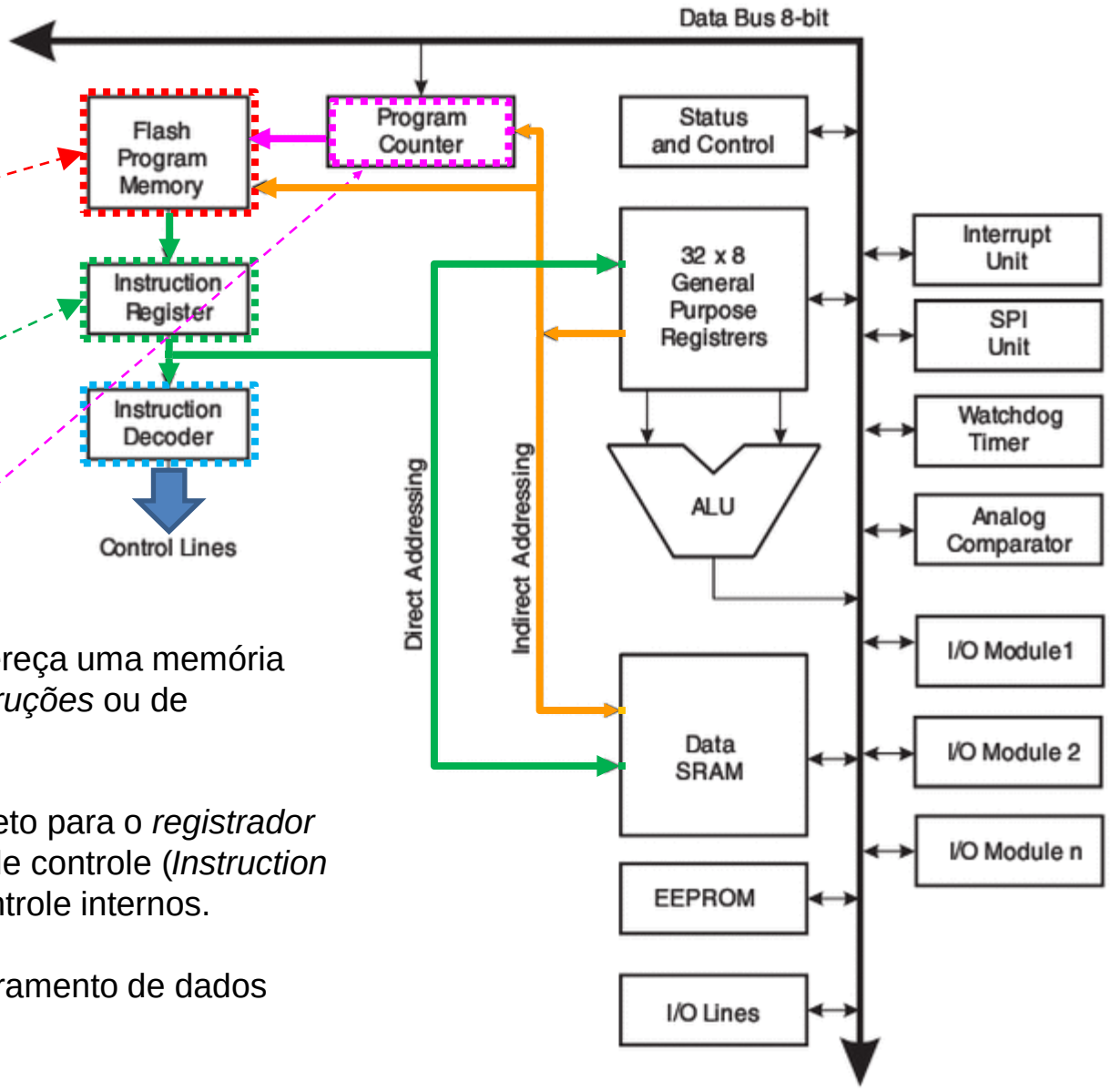
- Diagrama blocos AVR

Identificando Harvard...

Memória de Instruções

Registrador de instruções

Contador de Programa



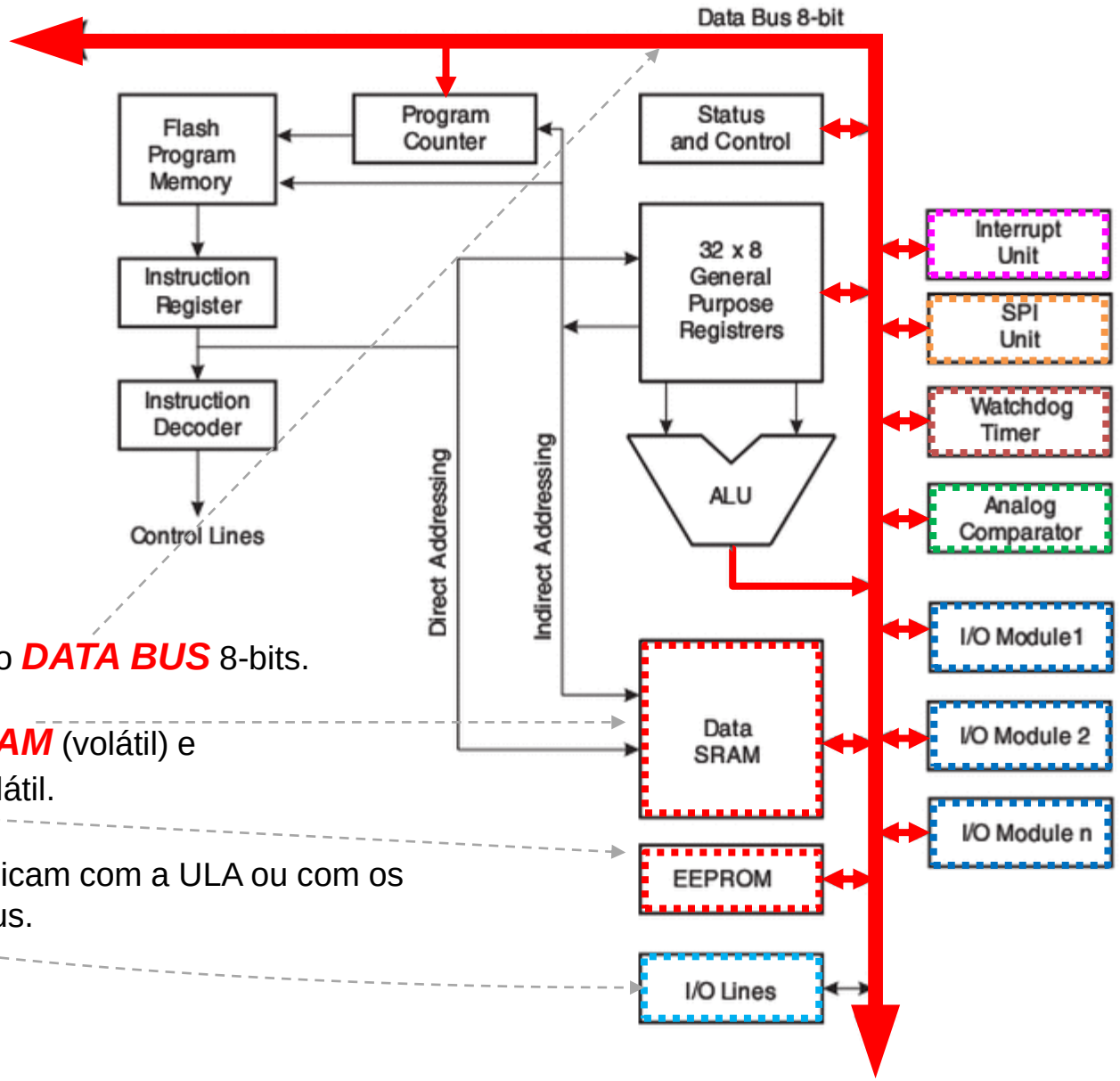
O contador de programa (PC) endereça uma memória separada (*Flash – memória de instruções* ou de *programa...*)

A instrução (saída da Flash) vai direto para o *registrador de instrução* e daí para a unidade de controle (*Instruction Decoder*) que gera as linhas de controle internos.

Esse caminho não depende do barramento de dados (*data bus* de 8-bits).

- Diagrama blocos AVR

Identificando Harvard...



DADOS fluem no controlador pelo ***DATA BUS*** 8-bits.

Há uma memória de rascunho ***RAM*** (volátil) e uma memória ***EEPROM*** não volátil.

Todos os ***periféricos*** se comunicam com a ULA ou com os registradores (e RAM) via data bus.

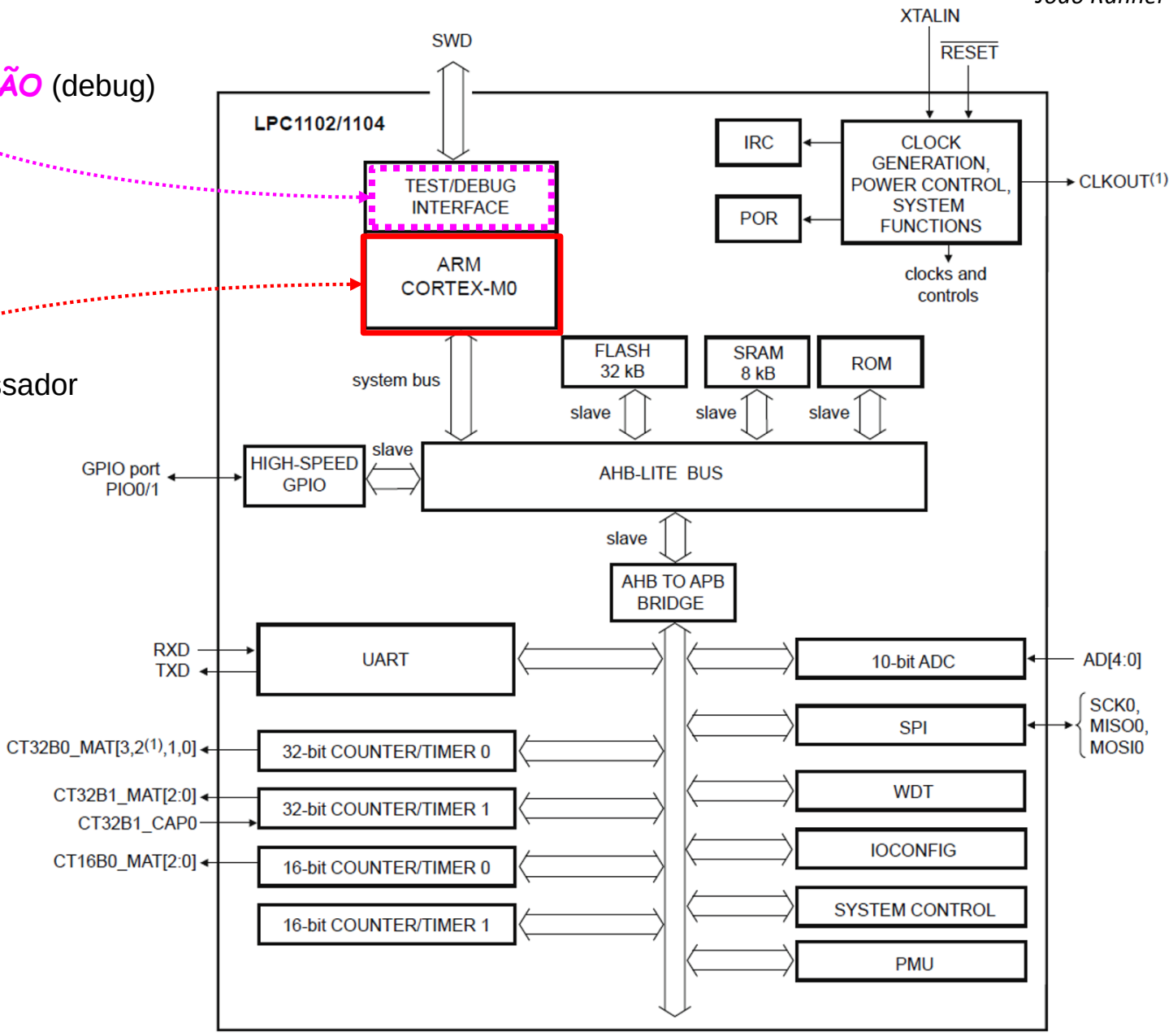
UFABC

✧ Diagrama de blocos de um SoC baseado em ARM (NXP – LPC1102)

João Ranhel

Interface p/ **DEPURAÇÃO** (debug)

núcleo **(CORE)** processador



UFABC

✧ Diagrama de blocos de um SoC baseado em ARM (NXP – LPC1102)

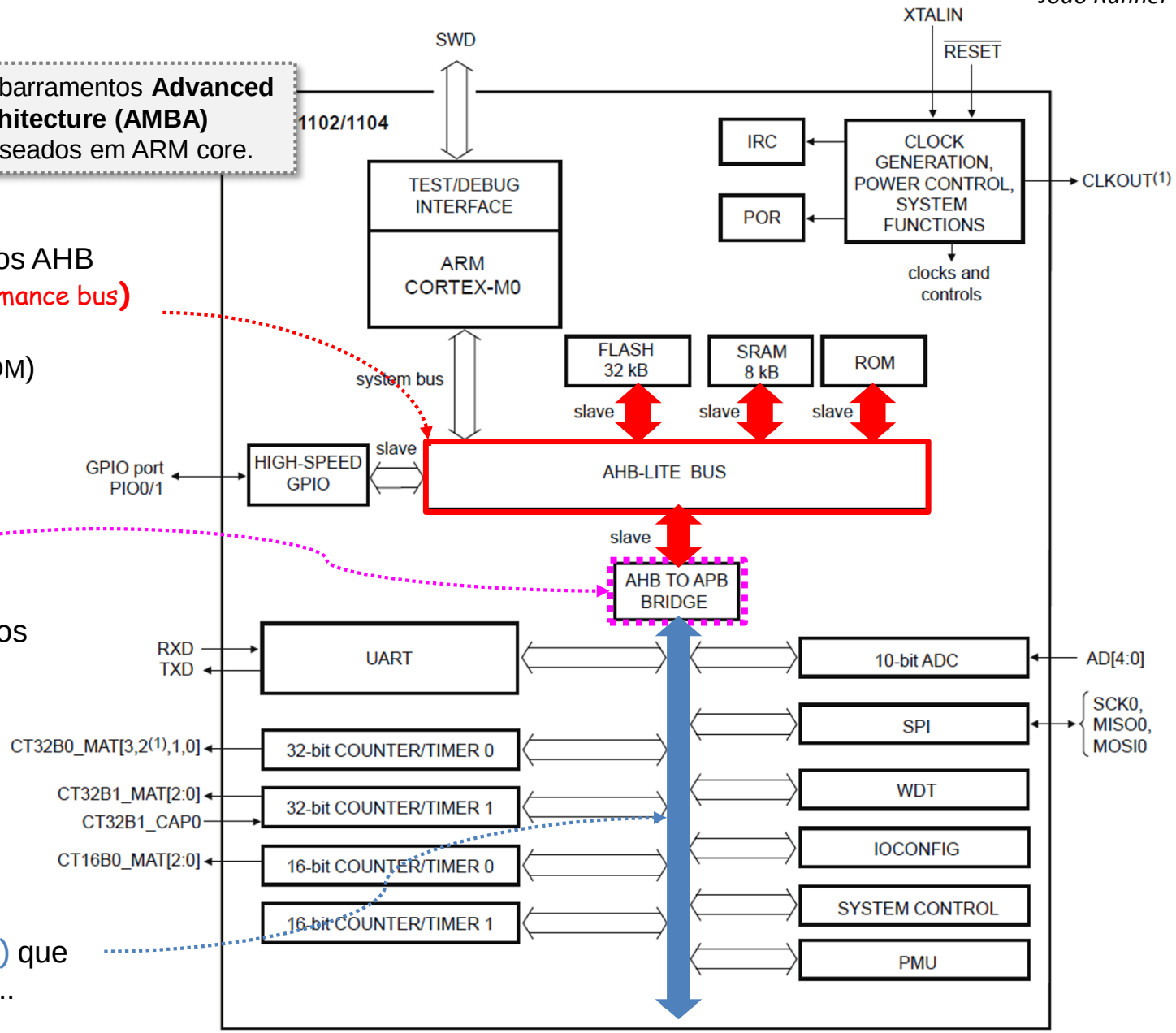
João Ranhel

A ARM criou o padrão de barramentos **Advanced Microcontroller Bus Architecture (AMBA)** usado em todos os μC baseados em ARM core.

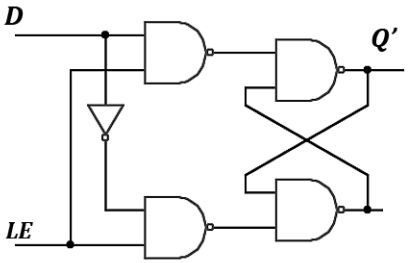
Matriz de barramentos AHB (advanced high-performance bus) de alta velocidade... (liga FLASH, RAM e ROM)

Ponte (bus bridge) com bus de periféricos de baixa velocidade

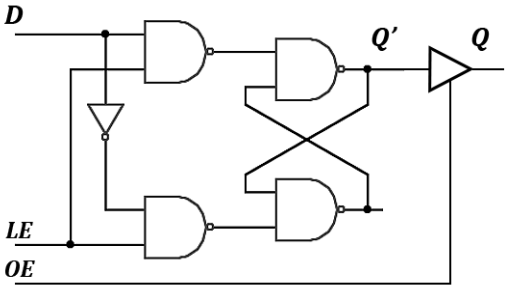
Barramento APB (advanced peripheral bus) que interliga os periféricos...



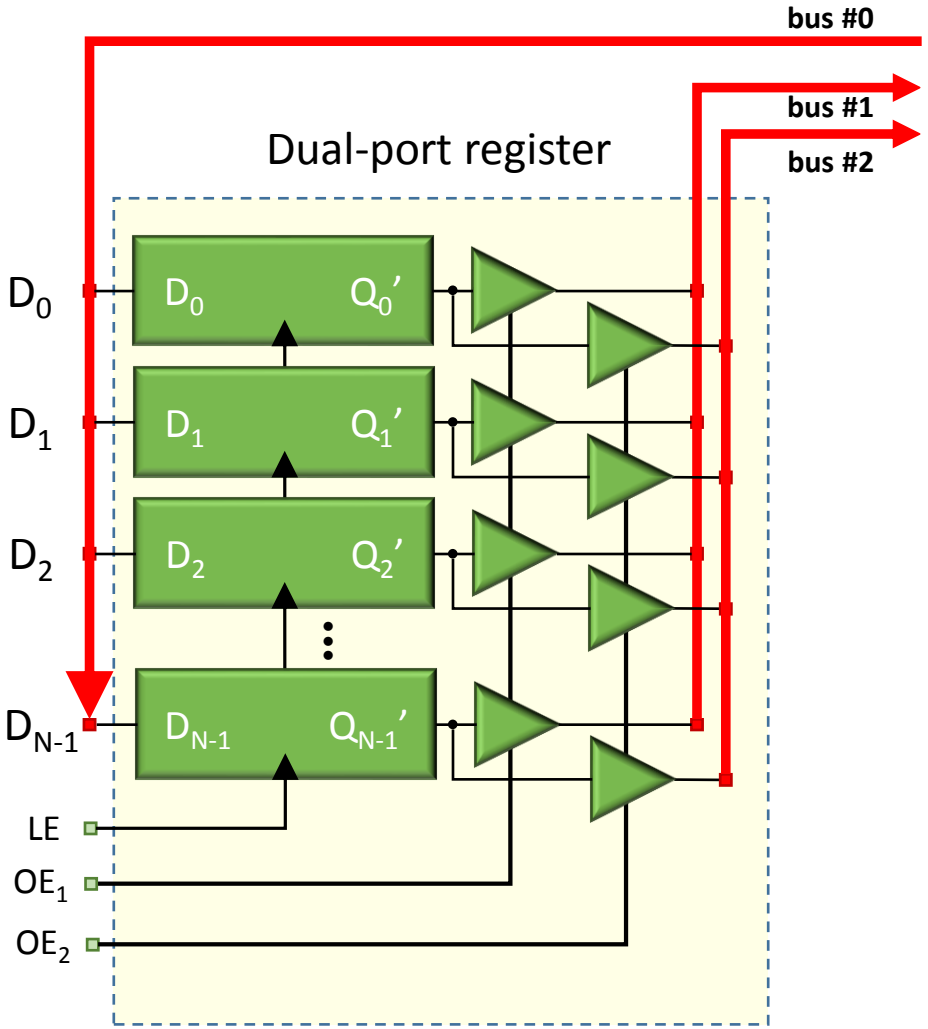
➢ CONCEITO NOVO: ***DUAL-PORT REGISTER*** (DUAL-PORT RAM)



Latch tipo D
→ armazena 1 bit na descida do comando LE



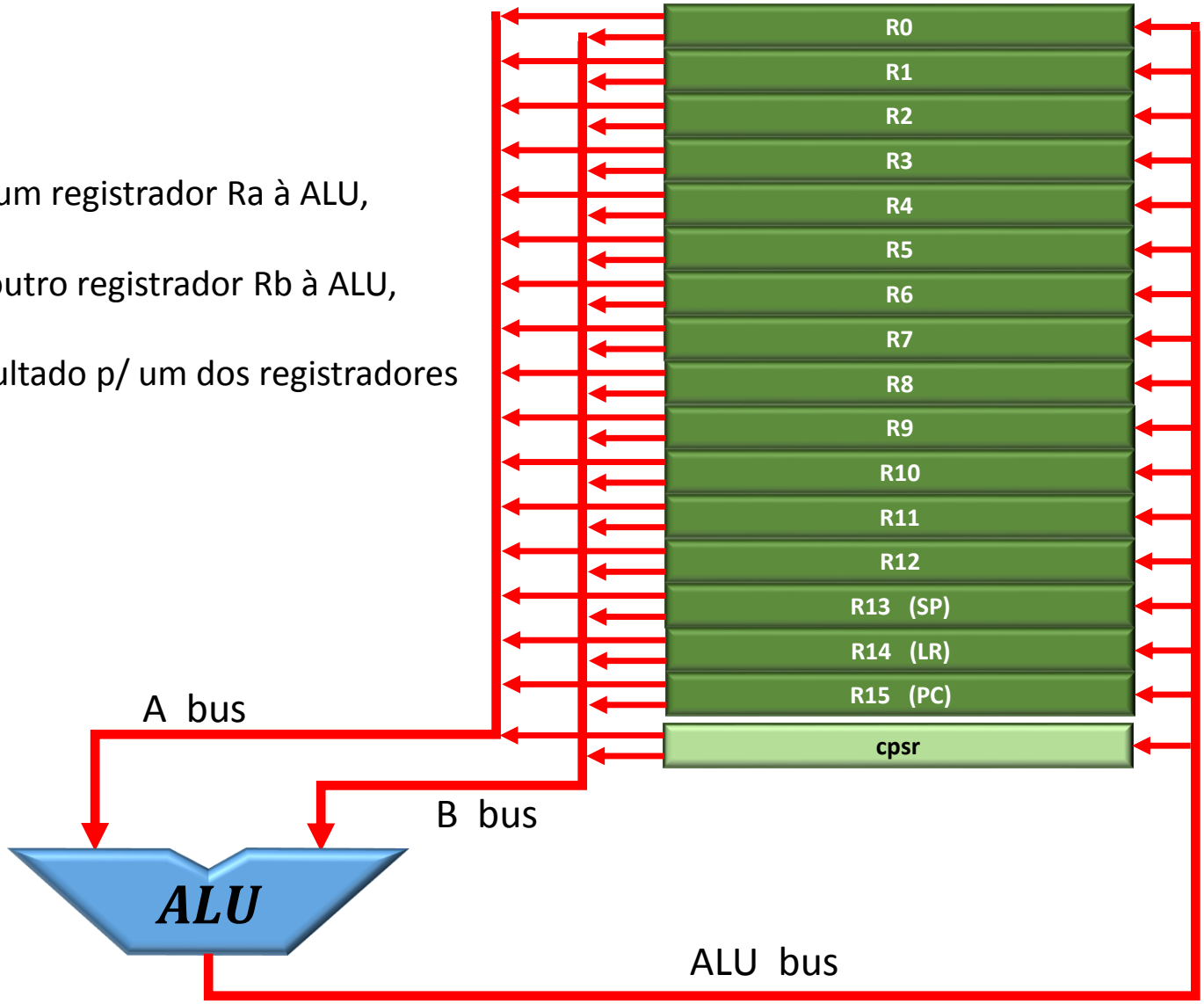
Latch-D 3-STATE
→ Permite a saída do bit memorizado no FF-D quando o sinal OE = '1'.



→ **Bus A** – leva o valor de um registrador Ra à ALU,

→ **Bus B** - leva o valor de outro registrador Rb à ALU,

→ **Bus Alu** – retorna o resultado p/ um dos registradores



➢ RESPOSTAS:

a) Como se **controla** qual registrador vai usar o barramento A ou o barramento B ?

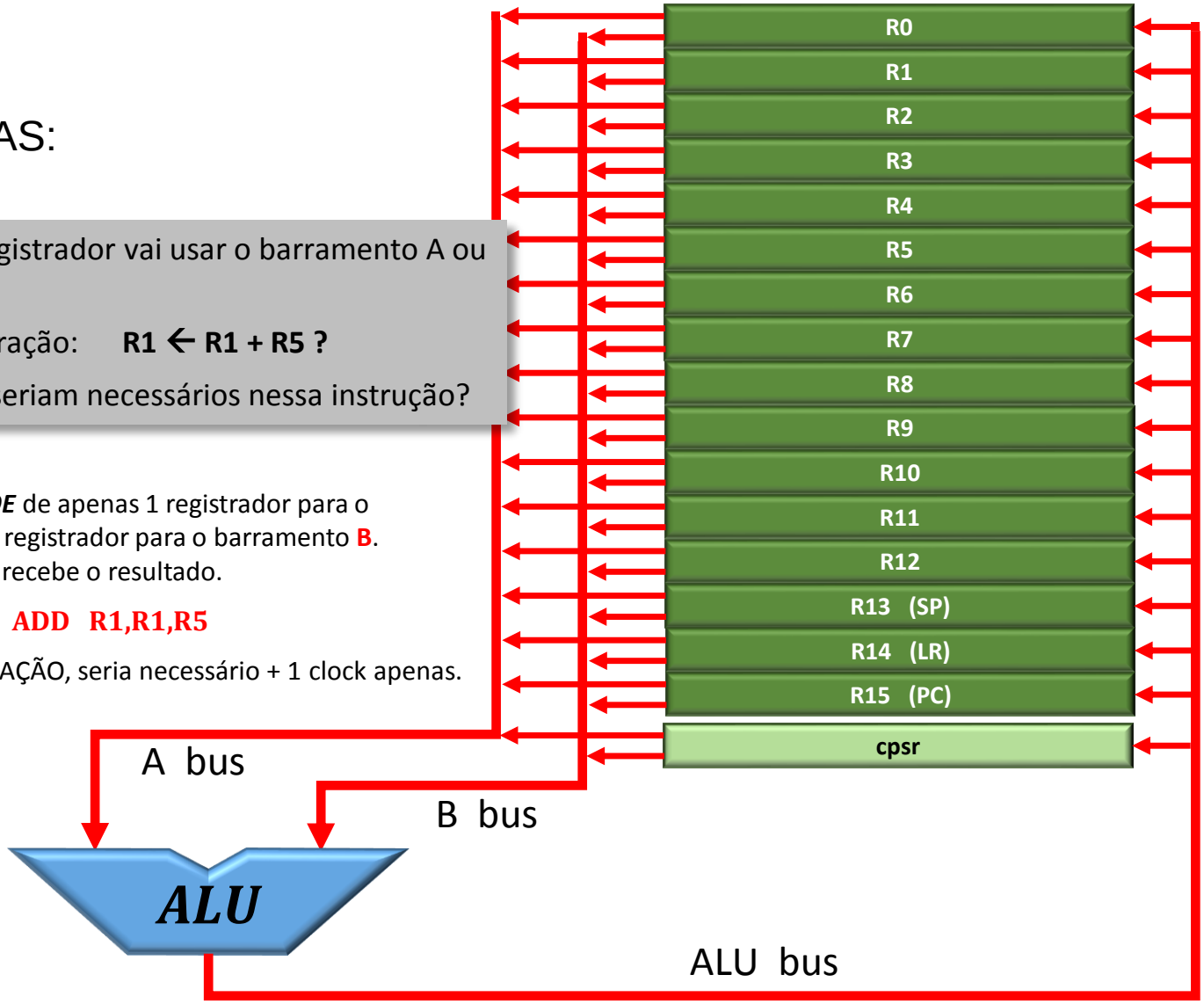
b) É possível executar a operação: **R1 ← R1 + R5 ?**

c) Quantos pulsos de clock seriam necessários nessa instrução?

R **a)** A unidade de controle liga o ***OE*** de apenas 1 registrador para o barramento **A**, e o ***OE*** de apenas 1 registrador para o barramento **B**. Então, liga o ***LE*** do registrador que recebe o resultado.

R **b)** Sim, o mnemônico pode ser: **ADD R1,R1,R5**

R **c)** Depois do FETCH e DECODIFICAÇÃO, seria necessário + 1 clock apenas.



Recordando... Registrador Universal de deslocamento

R **a)** Considerando que o bit mais significativo (MSB) fica à esquerda, deslocar 2 bits para a esquerda significa multiplicar o número BCD por 4

R **b)** Considerando MSB à esquerda, deslocar 1 bit para a direita significa dividir o número por 2

a) O que significa DESLOCAR 2 bits para a esquerda em um número binário sem sinal?

b) O que significa deslocar 1 bit para a direita?

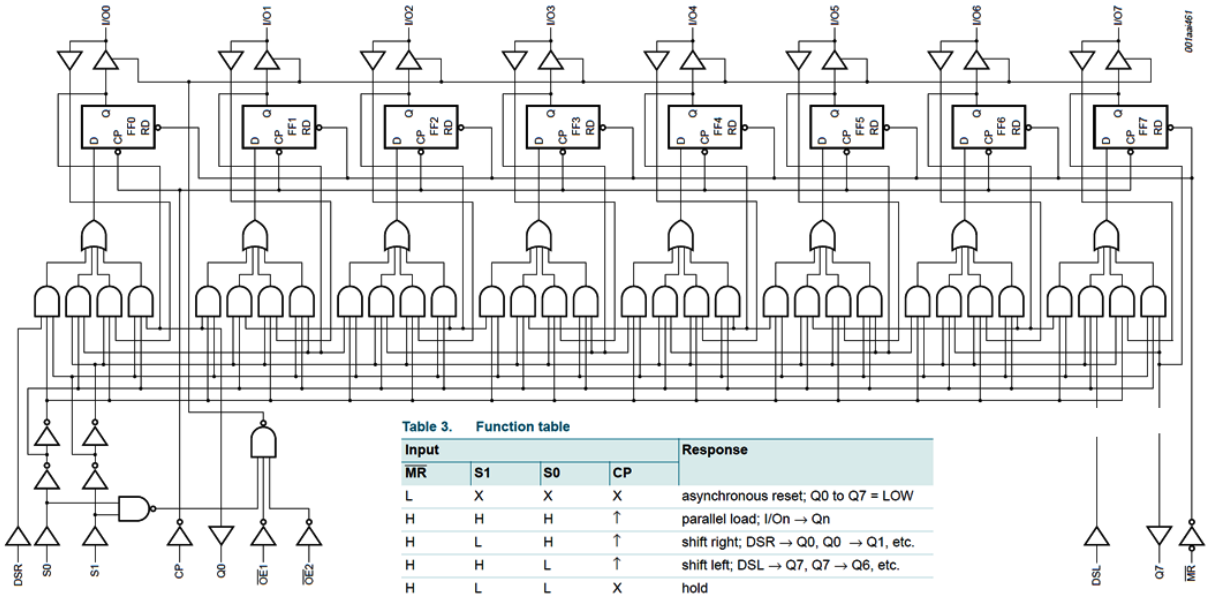
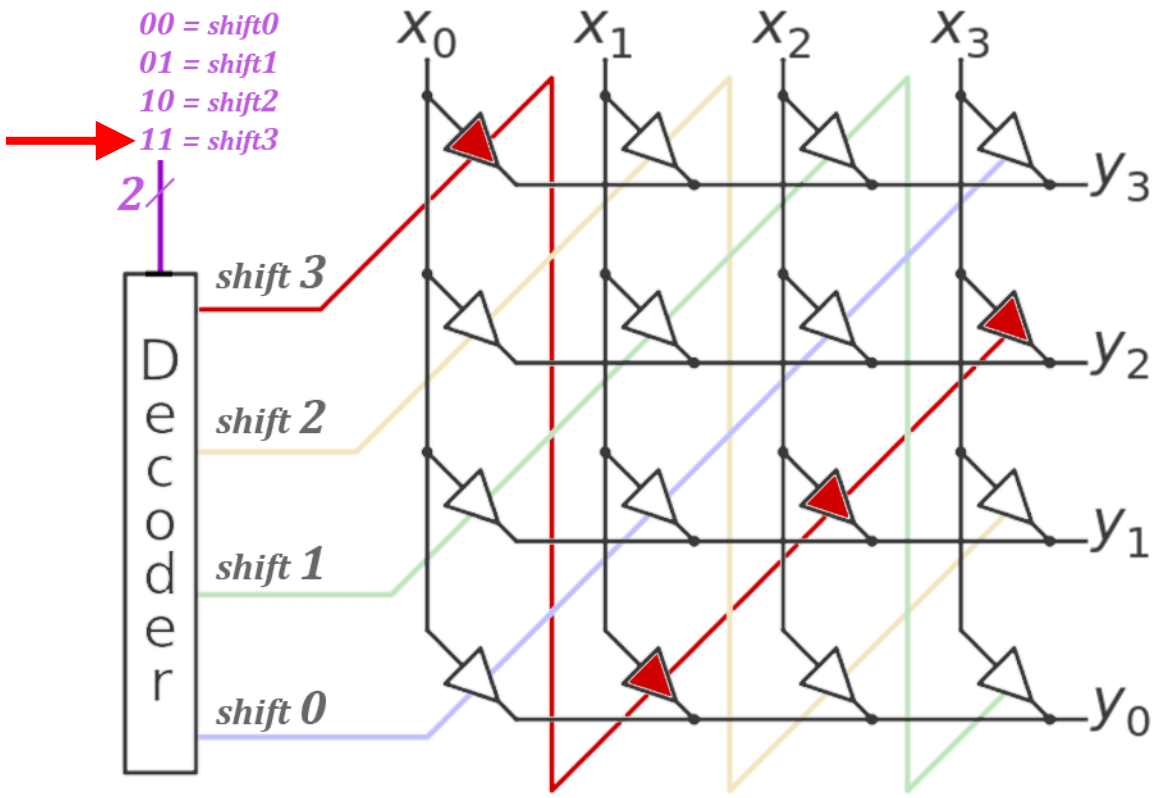


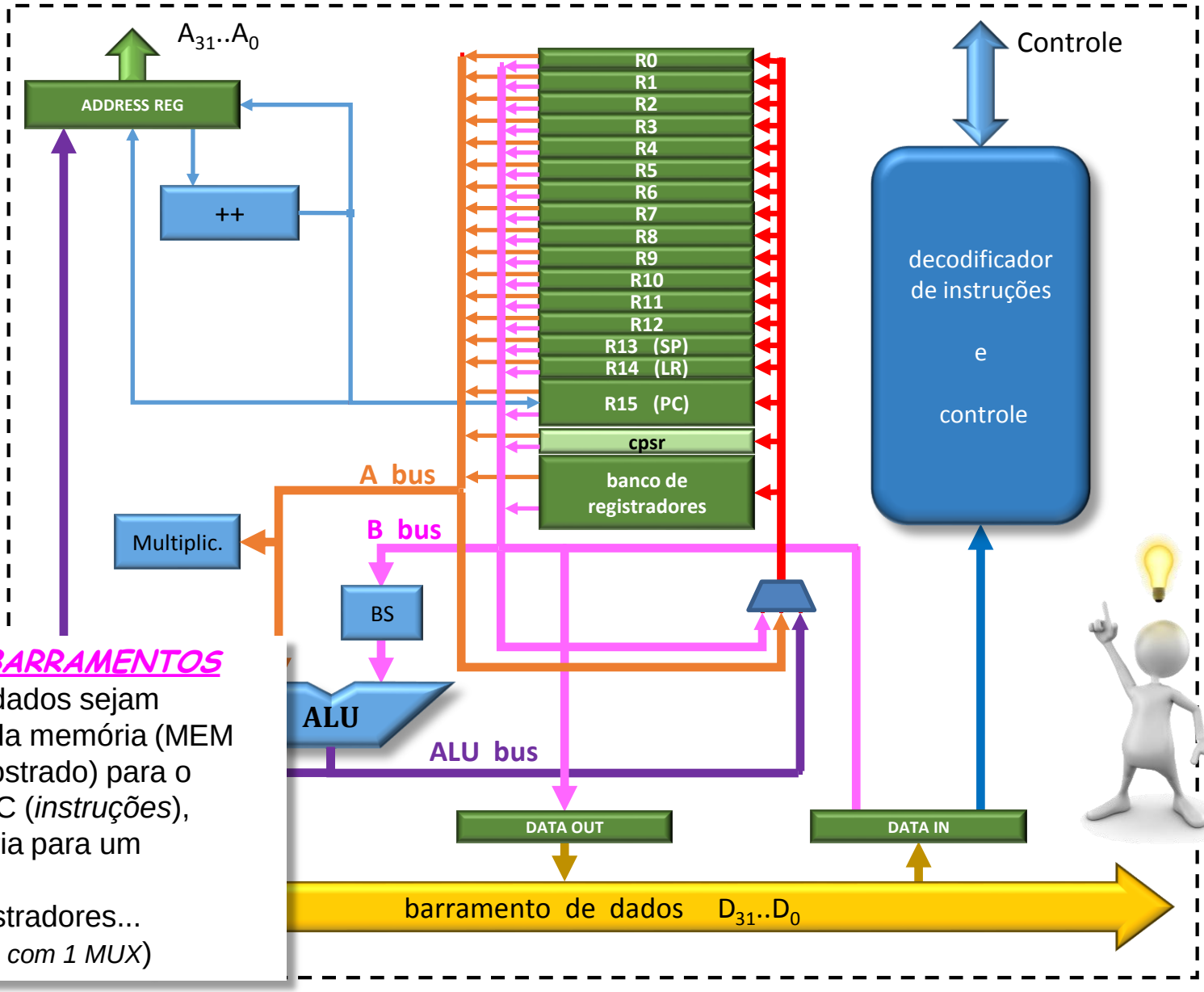
Table 3. Function table

Input				Response
$\overline{MR}$	S1	S0	CP	
L	X	X	X	asynchronous reset; Q0 to Q7 = LOW
H	H	H	↑	parallel load; I/On → Qn
H	L	H	↑	shift right; DSR → Q0, Q0 → Q1, etc.
H	H	L	↑	shift left; DSL → Q7, Q7 → Q6, etc.
H	L	L	X	hold

Problema com o registrador de deslocamento é que cada bit deslocado usa 1 clock.
Criou-se então o circuito BARREL SHIFTER (feito apenas com lógica combinacional)!



NÚCLEO (**CORE**) simplificado de um processador **ARM**



Matriz de BARRAMENTOS

- ✓ permite que dados sejam transferidos da memória (MEM bus – não mostrado) para o registrador PC (*instruções*),
- ✓ ou da memória para um registrador,
- ✓ ou entre registradores...

(*exemplo com 1 MUX*)

Todos os registradores são de 32 bits !

porém, cada registrador pode conter:

Byte *= 8 bits*

Half-word *= 16 bits*

Word *= 32 bits*

- R13 – Stack Pointer

*(apontador da pilha – guarda o último ENDEREÇO da **pilha** na RAM)*

- R14 – Link Register

(guarda o endereço de RETORNO quando uma subrotina é chamada)

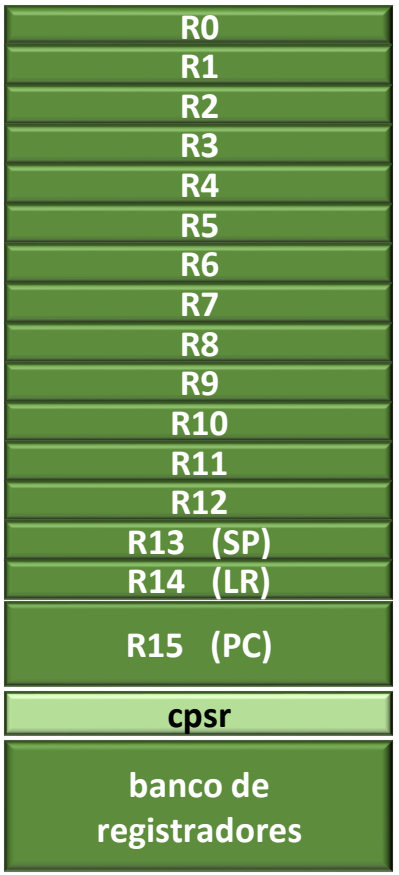
- R15 – Program Counter

(guarda o endereço da próxima INSTRUÇÃO a ser executada)

- CPSR – reg de flags de status do programa

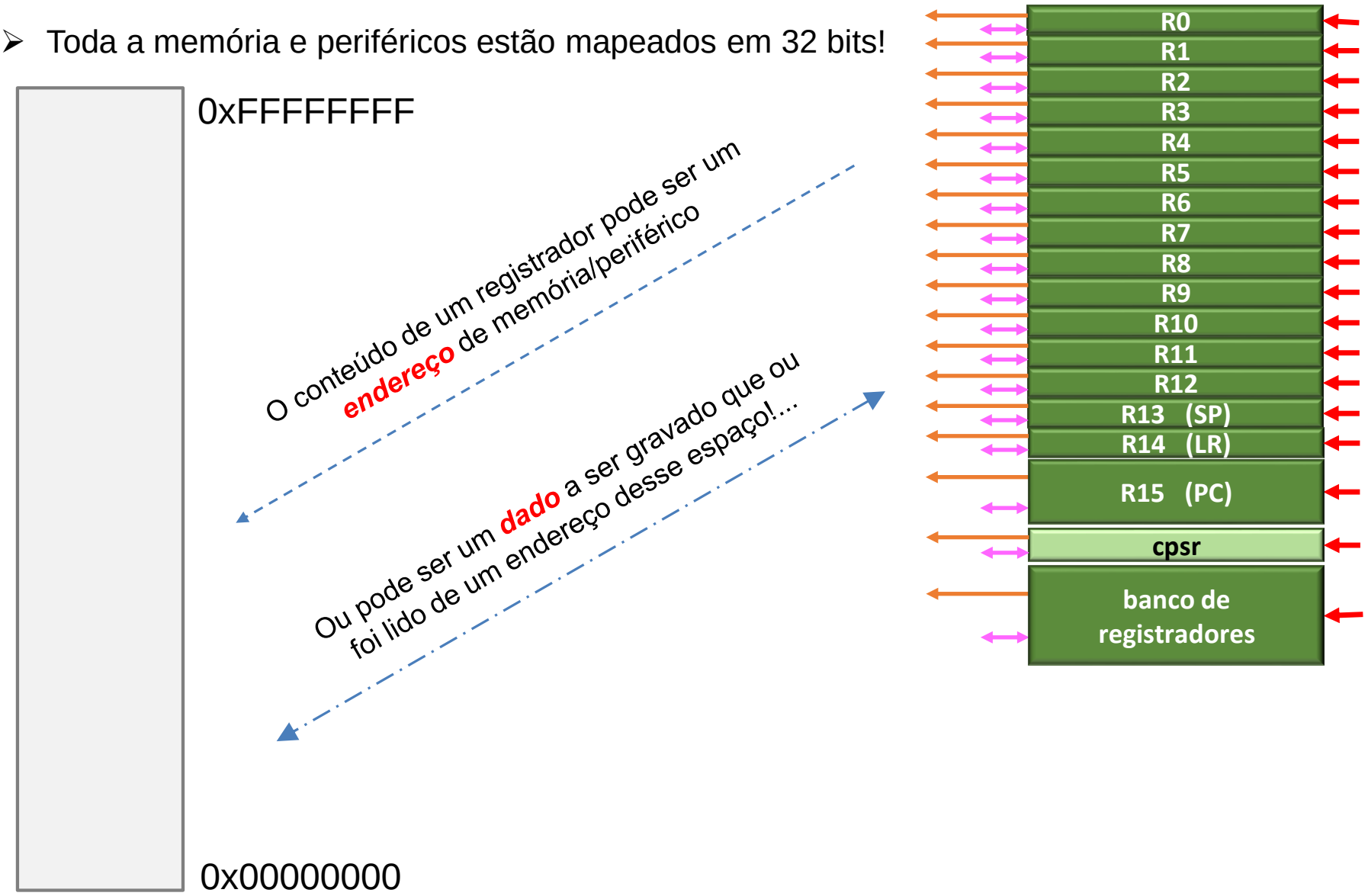
(Current Program Status Register).

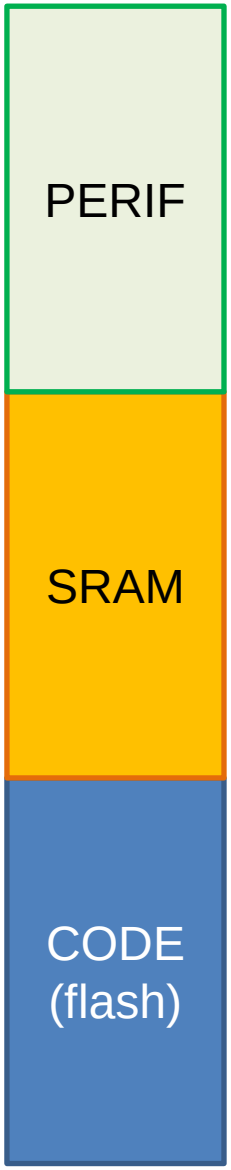
- Centenas de registradores *são usados para configurar dispositivos I/0 dentro do CHIP, como veremos. Contudo, não são registradores de uso geral, portanto não possuem instruções que os identifiquem. As instruções do ARM têm como operando apenas estes registradores*.



- Endereço de memória é relativo a 1 byte (para endereçar words = ++ 4 bytes)

- Toda a memória e periféricos estão mapeados em 32 bits!





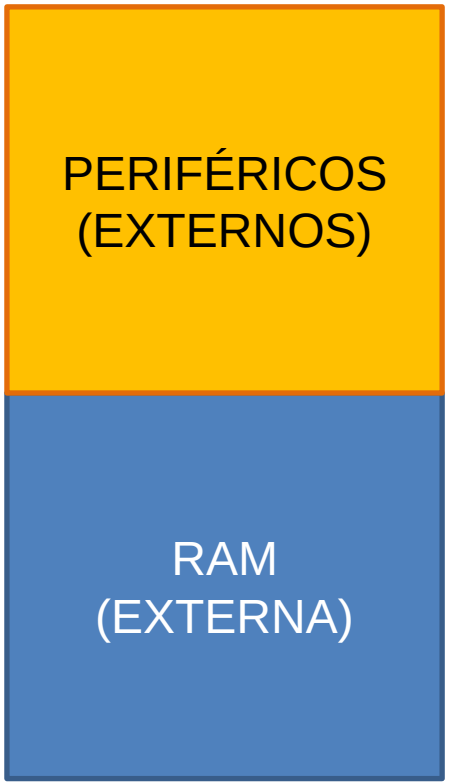
ESPAÇO DE MEMÓRIA ON-CHIP

0,5 Gbytes endereços de periféricos

0,5 Gbytes endereços de memória RAM estática

0,5 Gbytes endereços de memória RAM estática

ESPAÇO DE MEMÓRIA OFF-CHIP



0xDFFFFFFF

0,5 Gbytes endereços de periféricos EXTERNOS

0xA0000000

0x9FFFFFFF

0,5 Gbytes endereços de memória RAM EXTERNA

0x60000000

ESPAÇO DE MEMÓRIA DO FABRICANTE
(e espaço reservado para funções específicas – 1MB)

0xFFFFFFFF

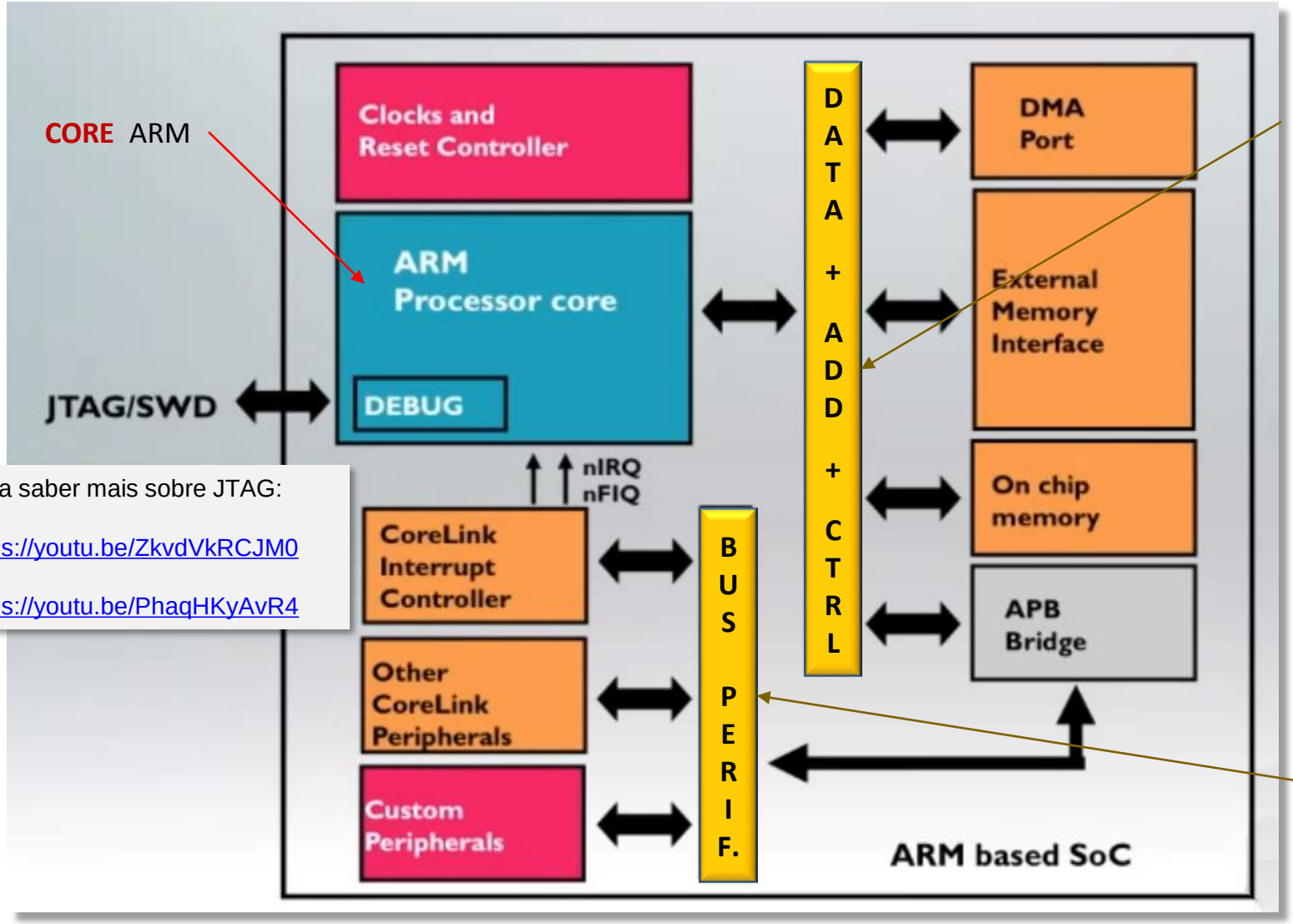
0,5 Gbytes endereços liberados para um fabricante...

0xE0100000
0xE00FFFFF

1 Mbytes barramento periférico privado
- *Nested Vector Interrupt Controller – NVIC*
- *System Tick Timer – SysTick*
- *Fault Status and Control*
- *Depurador do processador (debugging)*

0xE0000000

✧ **SoC** (System on Chip) baseado em core ARM

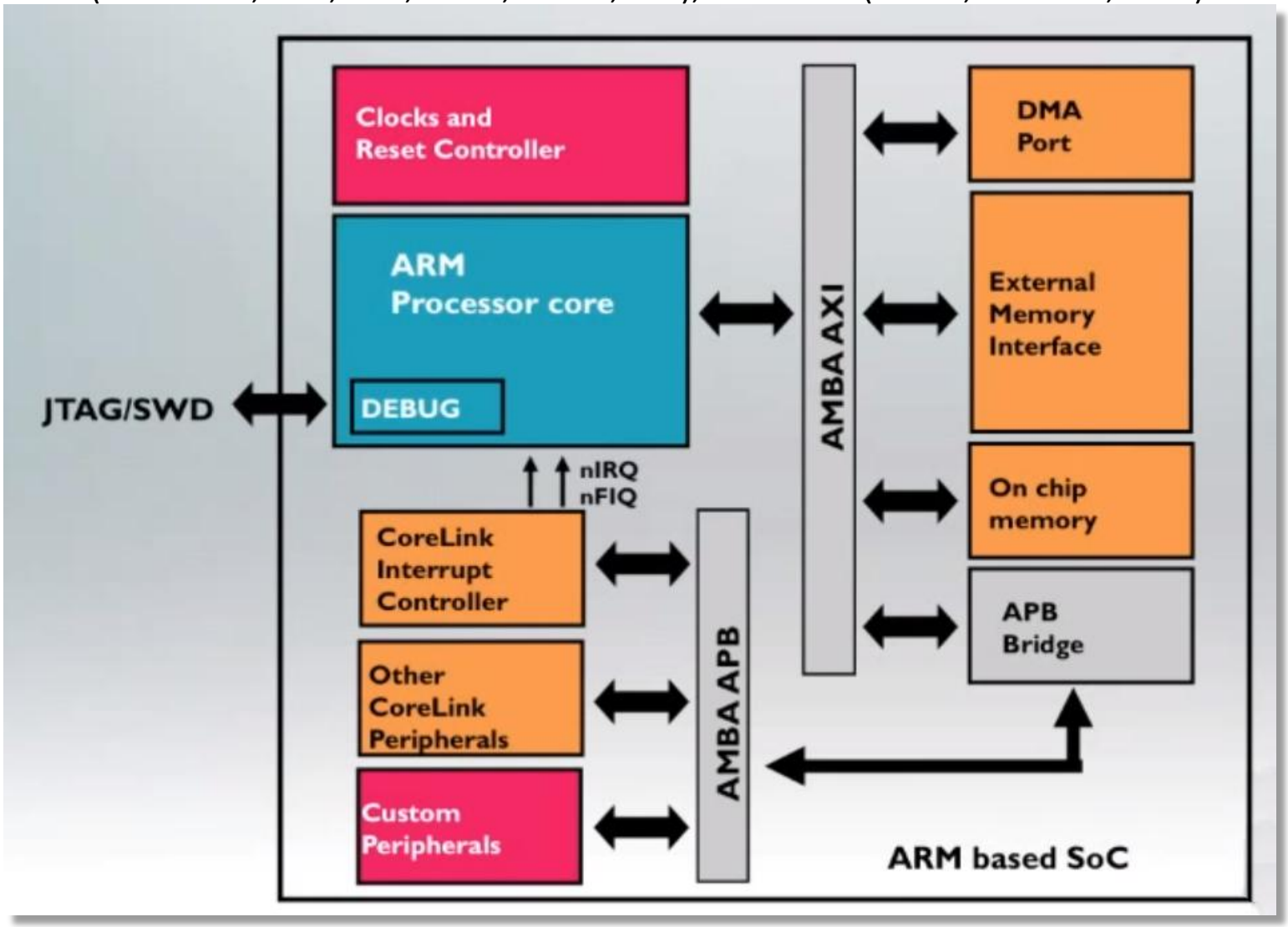


Para saber mais sobre JTAG:

https://youtu.be/ZkvdVkRCJM0

https://youtu.be/PhaqHKyAvR4

✧ Várias empresas licenciam o "core" ARM e seus barramentos, e inserem processadores (periféricos) adicionais (como ADC, DAC, USB, PWM, timers, etc.); memórias (FLASH, EEPROM, RAM) e bus-bridges.

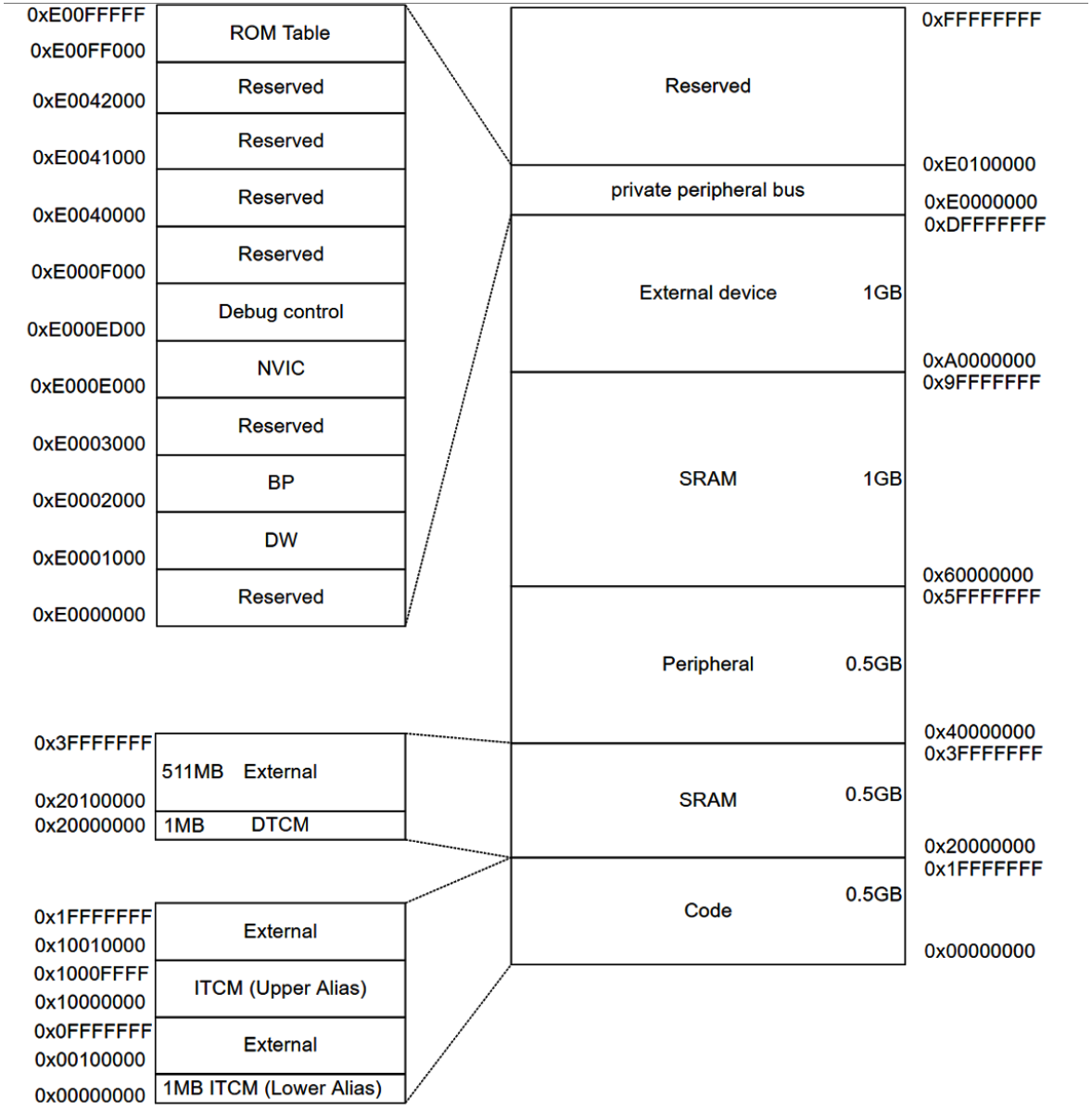


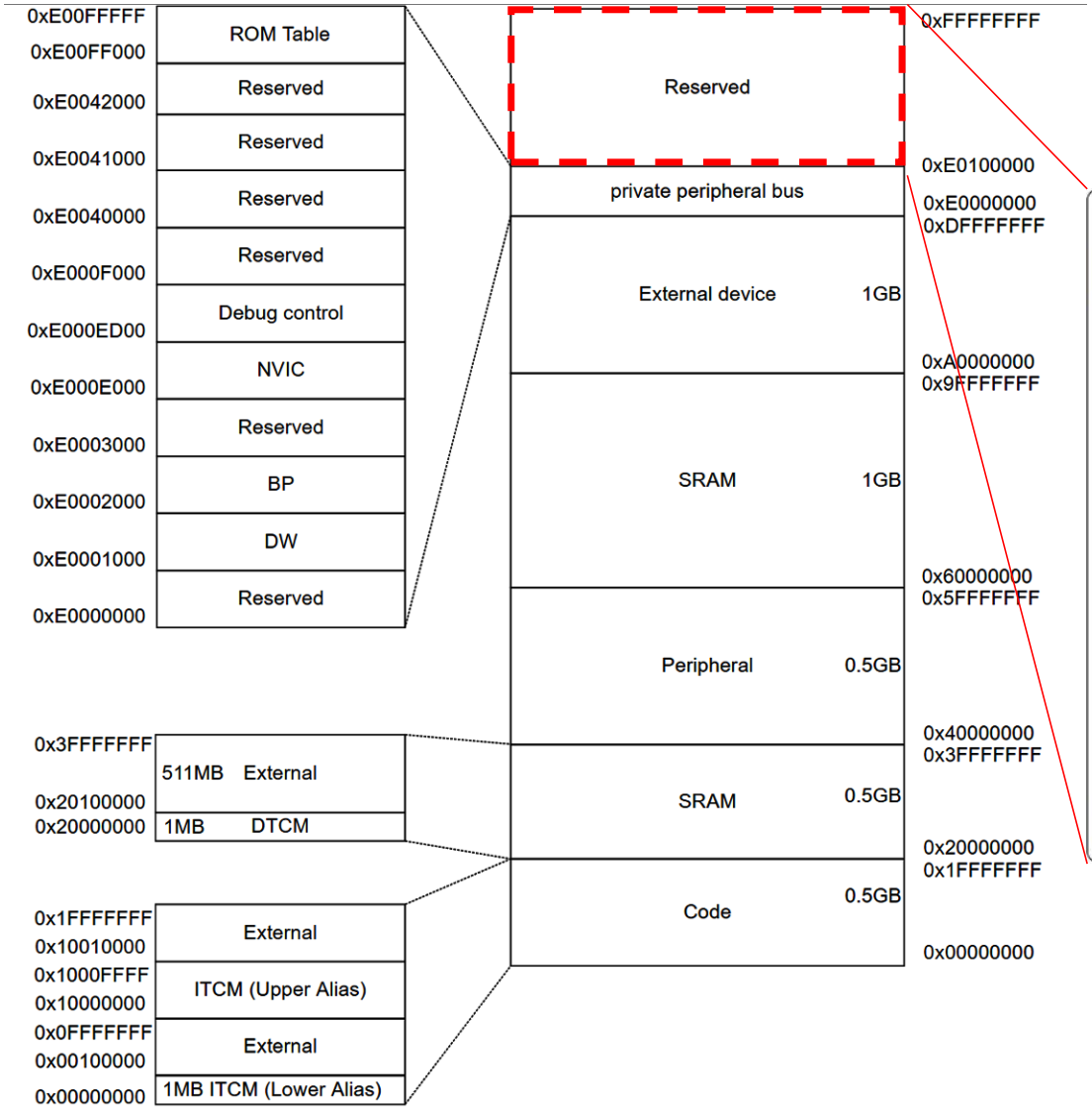
Os *registradores* podem conter:

Dados
Endereços

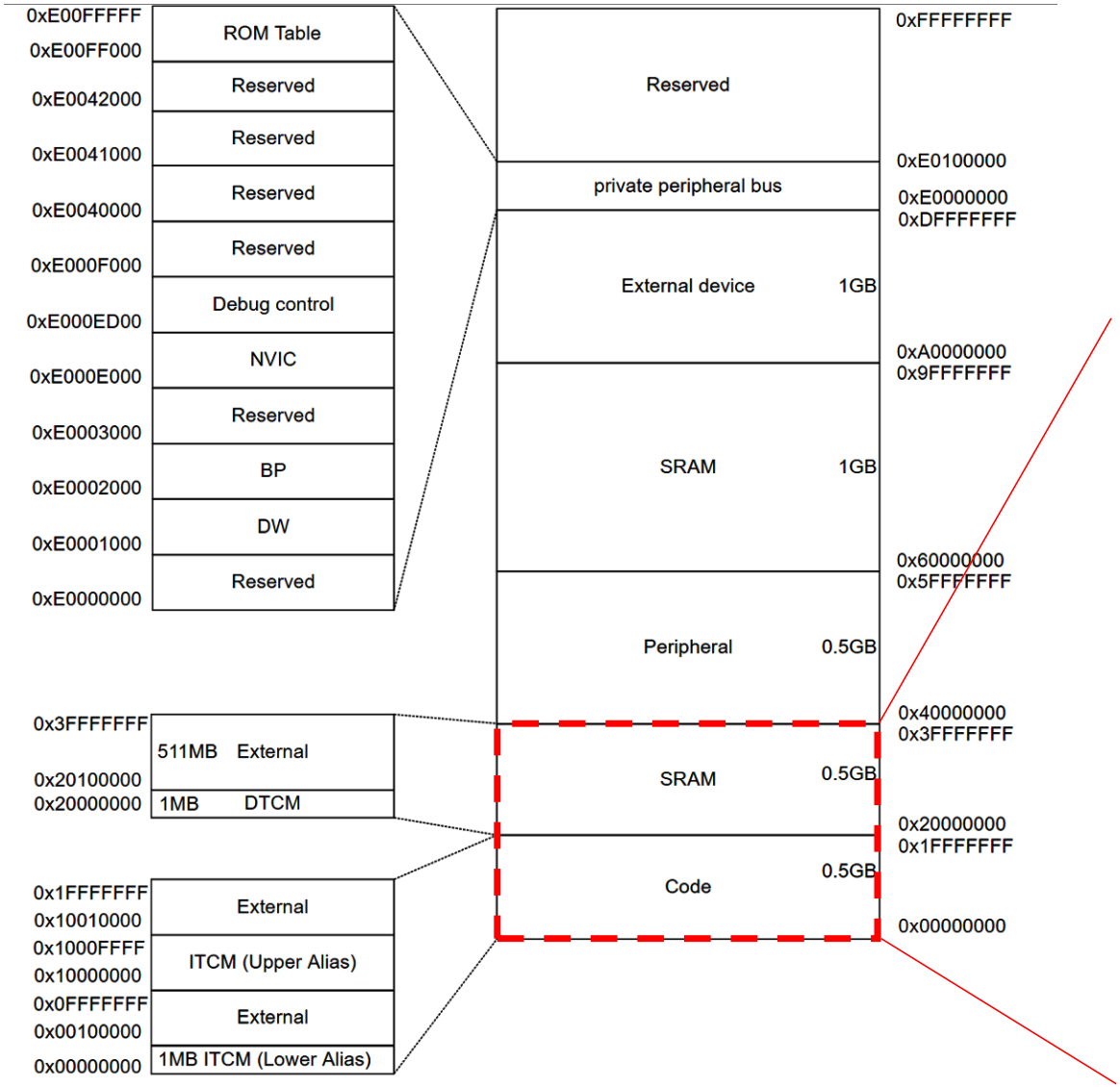
Todos de 32 bits!

É tarefa do programador interpretar se o conteúdo do registrador é dado ou endereço!





Periféricos internos
mapeados pelo fabricante do chip
(timers, watchdog, UART, SPI, I2C...)



Memória ROM e RAM interna mapeados pelo fabricante do chip (flash, eeprom, SRAM...)

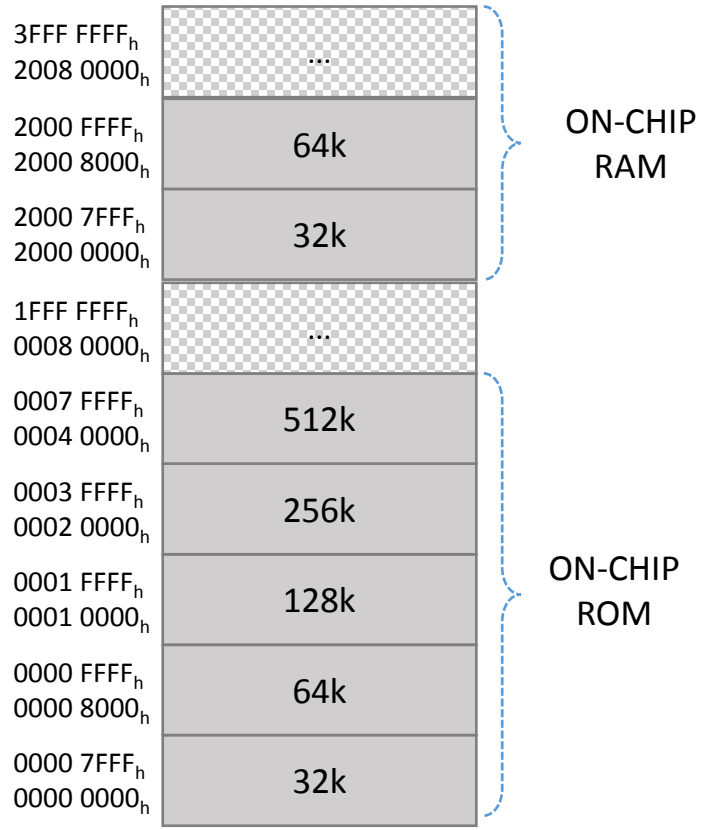
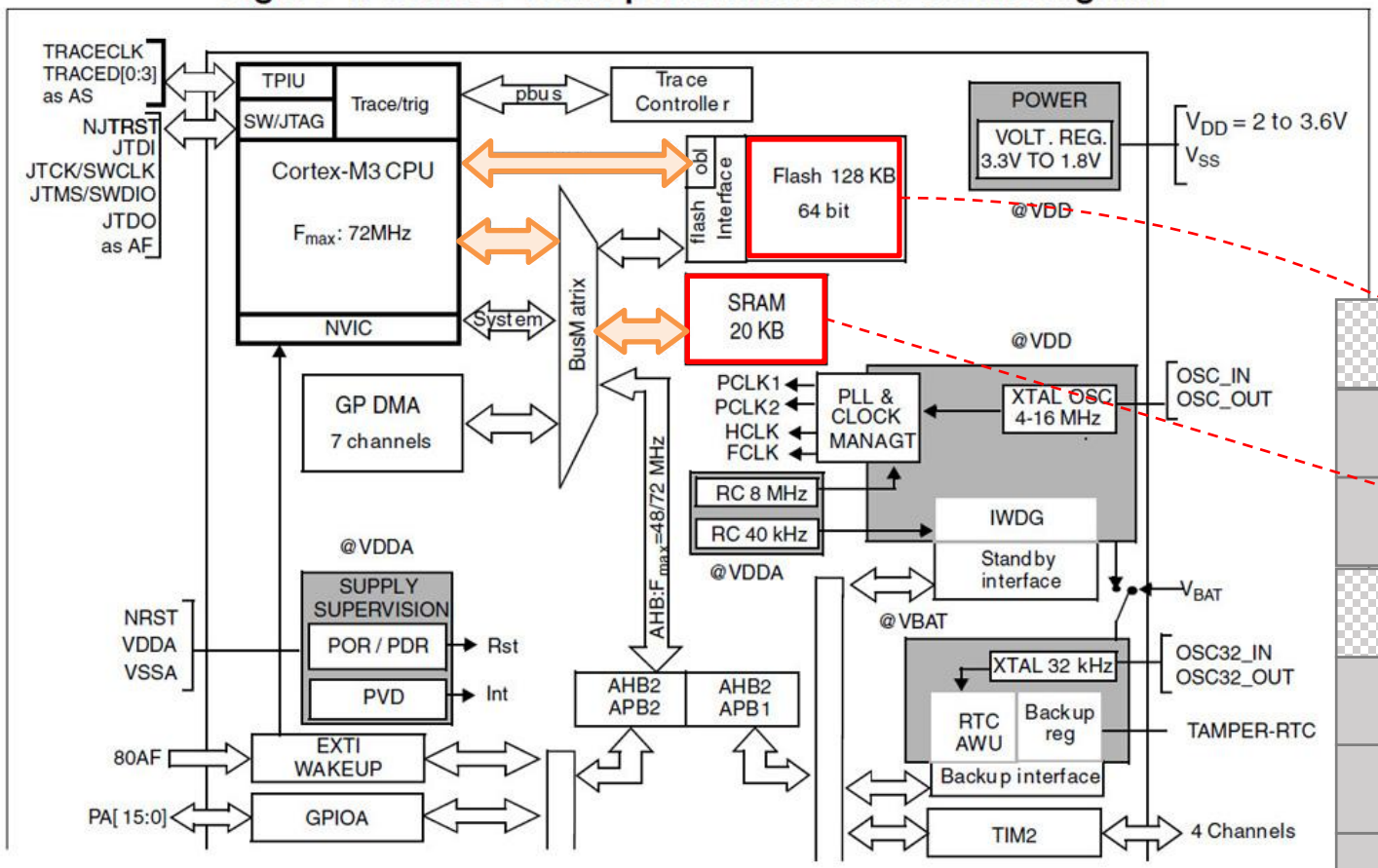
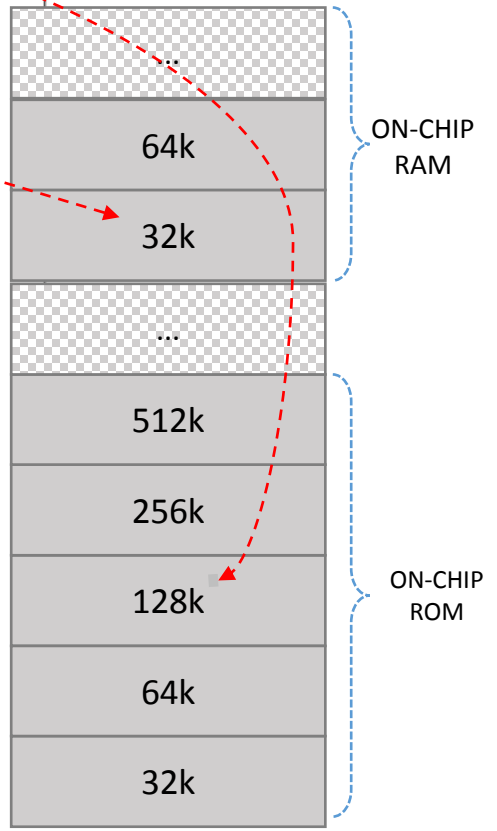


Figure 1. STM32F103xx performance line block diagram



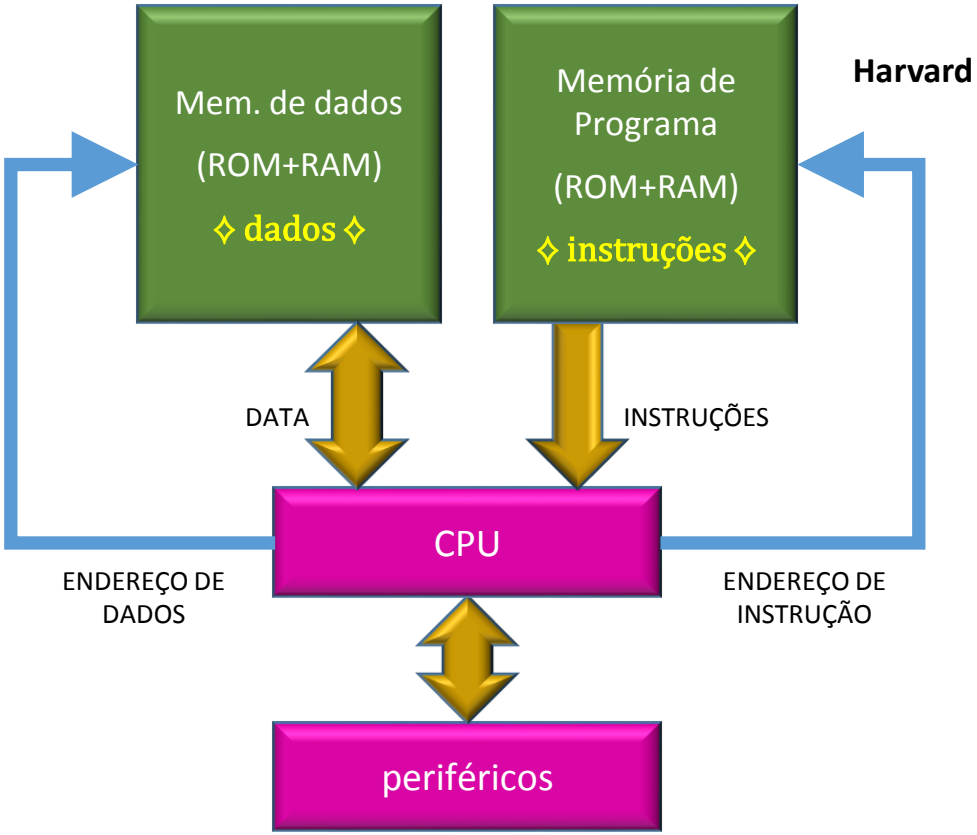


➢ Na arquitetura Harvard a memória de dados é separada da de instruções *!!!*

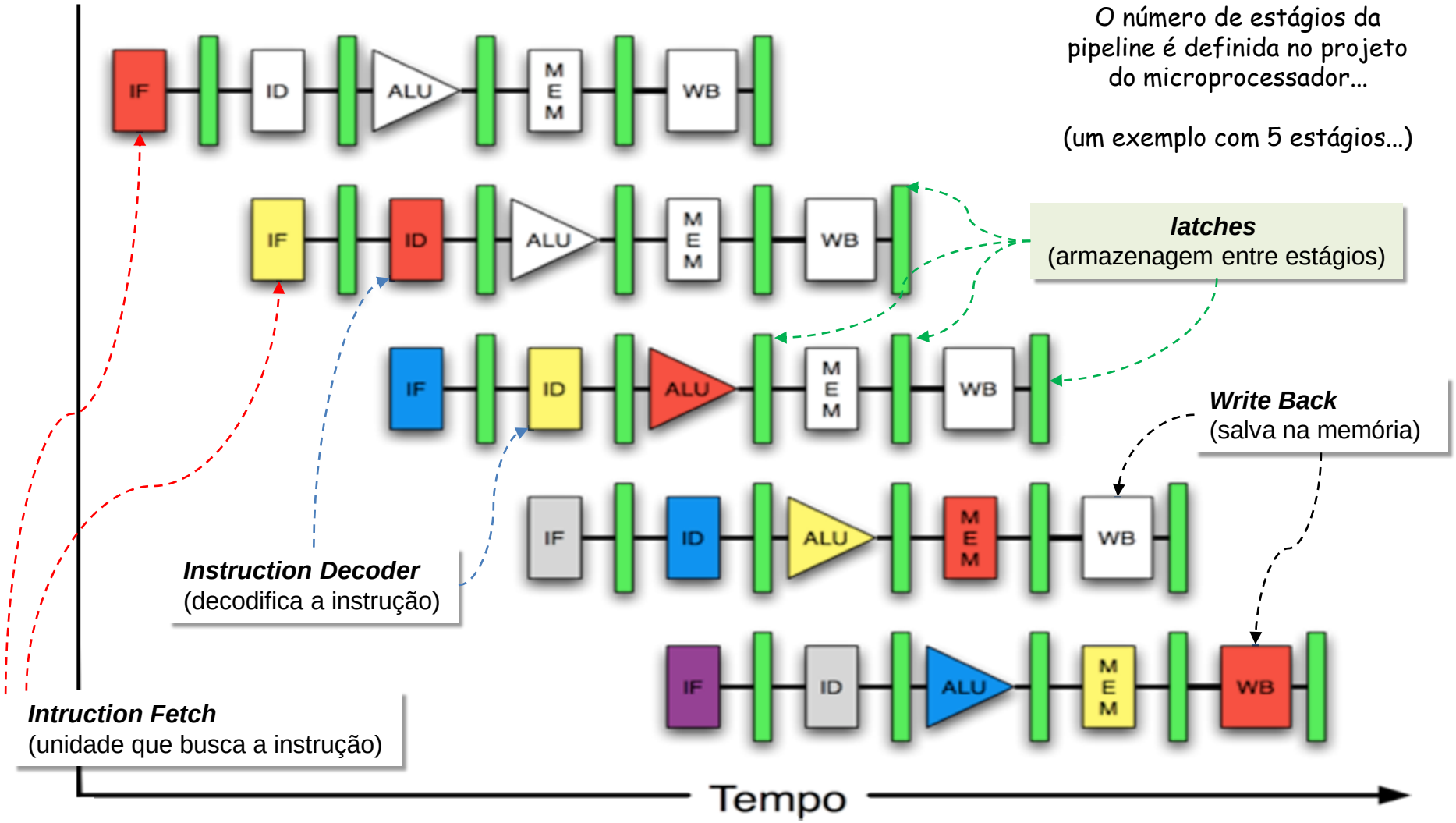
ENTÃO...

ENQUANTO A CPU EXECUTA UMA INSTRUÇÃO, PODEMOS BUSCAR OUTRA...

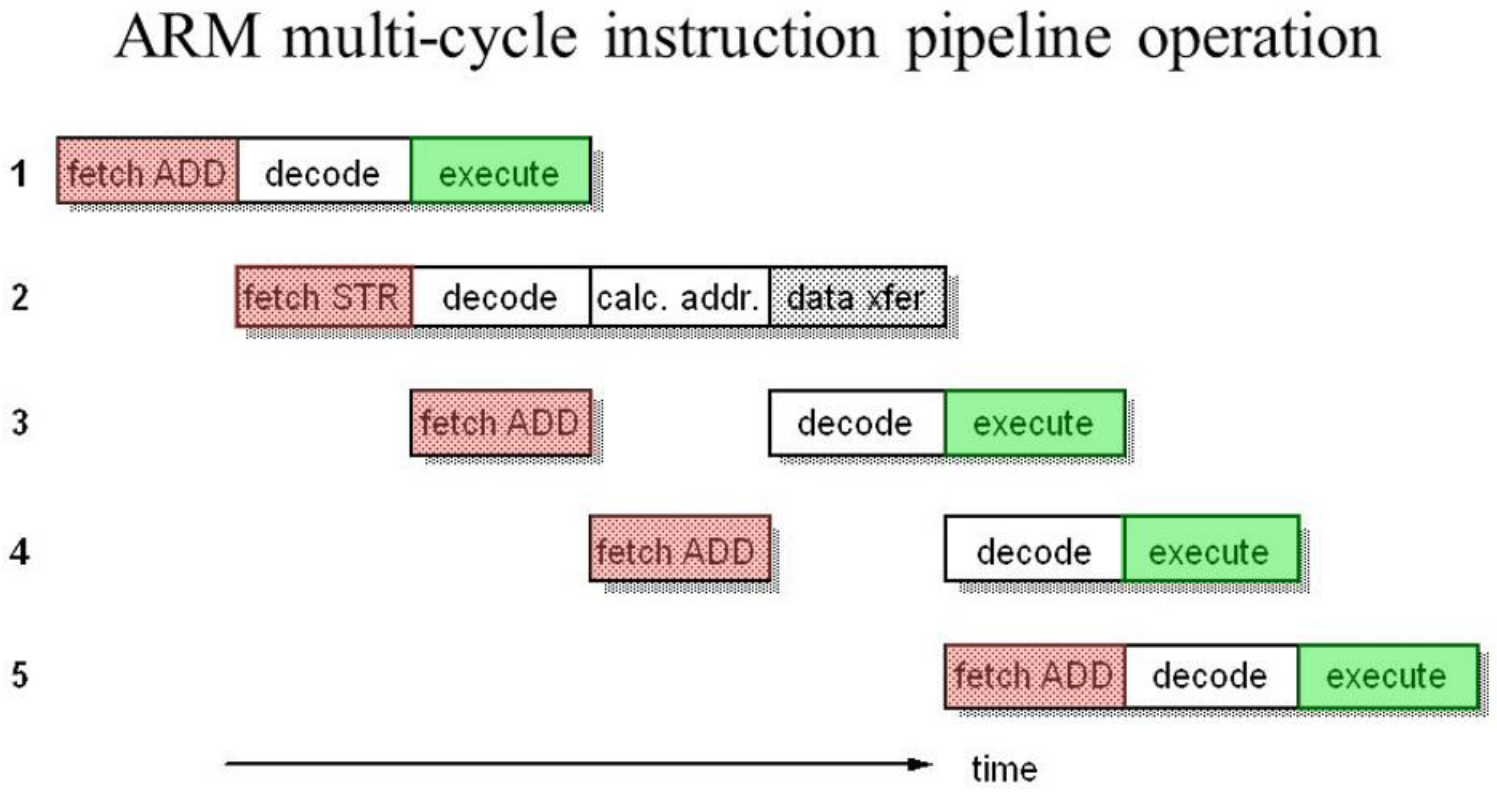
ESSE É O CONCEITO DE PIPELINE !



- Pipeline permite buscar instruções (***fetch – IF***) e executá-las simultaneamente *!!!*



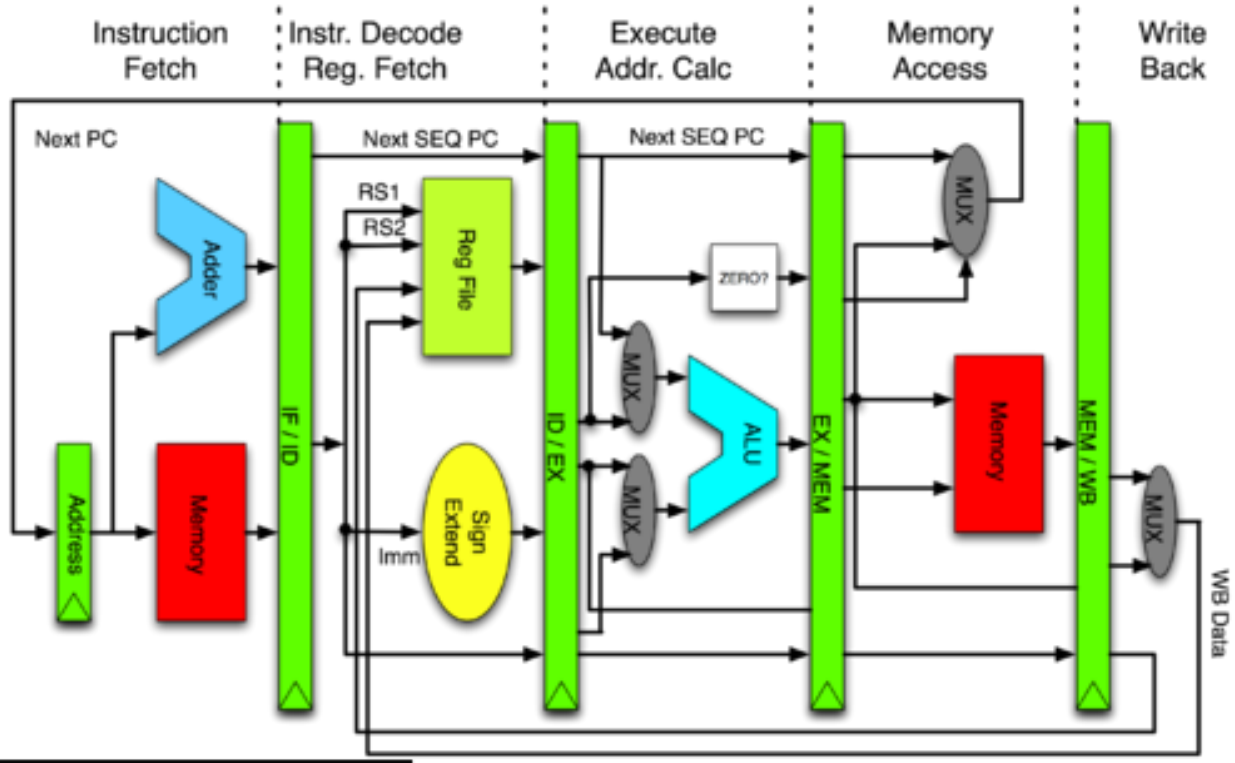
No ***modelo de 3 estágios***, algumas instruções requerem cálculo de *endereço* de memória para terminar a instrução. Isso pode causar um *delay* nas instruções seguintes...



➢ *CONCEITO NOVO:* ***PIPELINE*** *(cadeia simultânea de operações)*

✧ Arquitetura ARM – ***harzards*** em pipeline de 5 estágios...

João Ranhel



Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

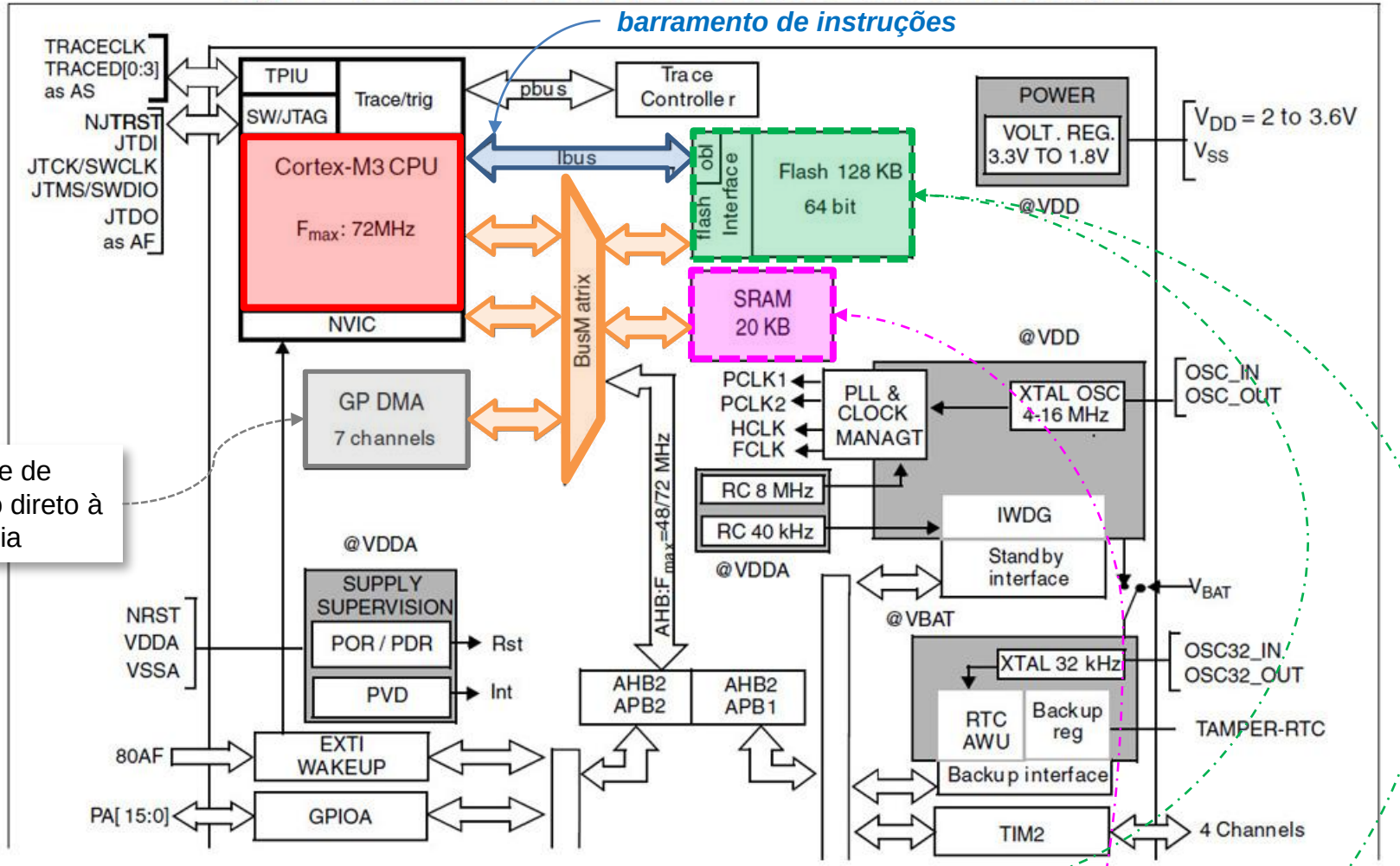
3 tipos de “hazards” (ameaças) à execução correta do programa:

Structural hazard: quando 2 ou mais instruções precisam usar o mesmo recurso de hardware (estágio do pipeline) ao mesmo tempo;

Data hazard: quando uma operação em uma instrução depende de um dado que ainda está sendo calculado na pipeline;

Control hazard: uma ameaça surge quando se faz uma transferência (e.g. JUMP) dependente de informações ainda em processamento na pipeline.

Figure 1. STM32F103xx performance line block diagram



controle de acesso direto à memória

- ✓ A **CPU** busca instruções na ***FLASH*** via ***barramento de instruções***;
- ✓ grava/lê dados da SRAM também pelo barramento de dados na bus matrix...
- ✓ grava/lê dados na ***FLASH*** via barramento de ***dados***, pela ***matriz de barramentos***

Figure 1. STM32F103xx performance line block diagram

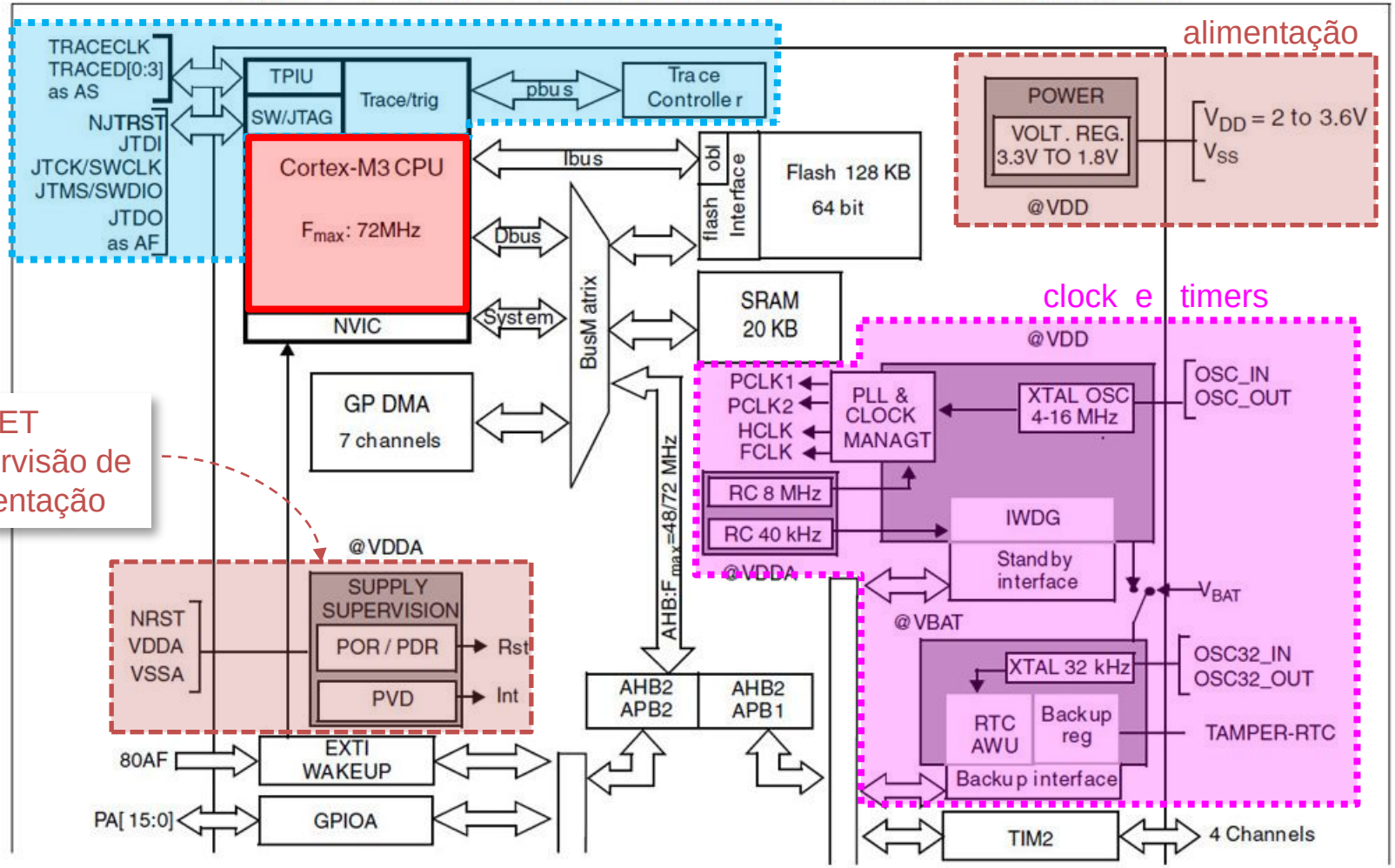
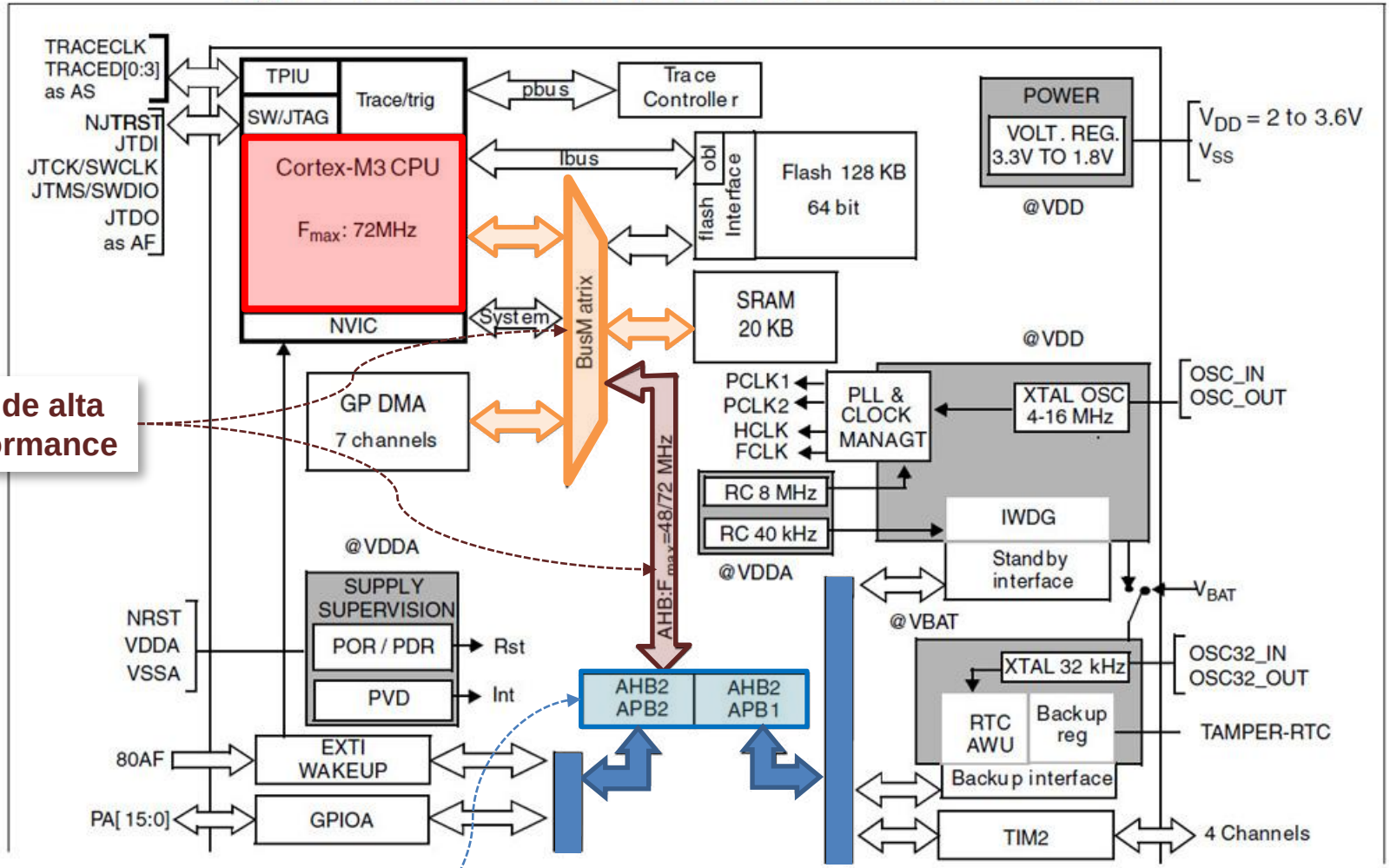


Figure 1. STM32F103xx performance line block diagram



bus de alta performance

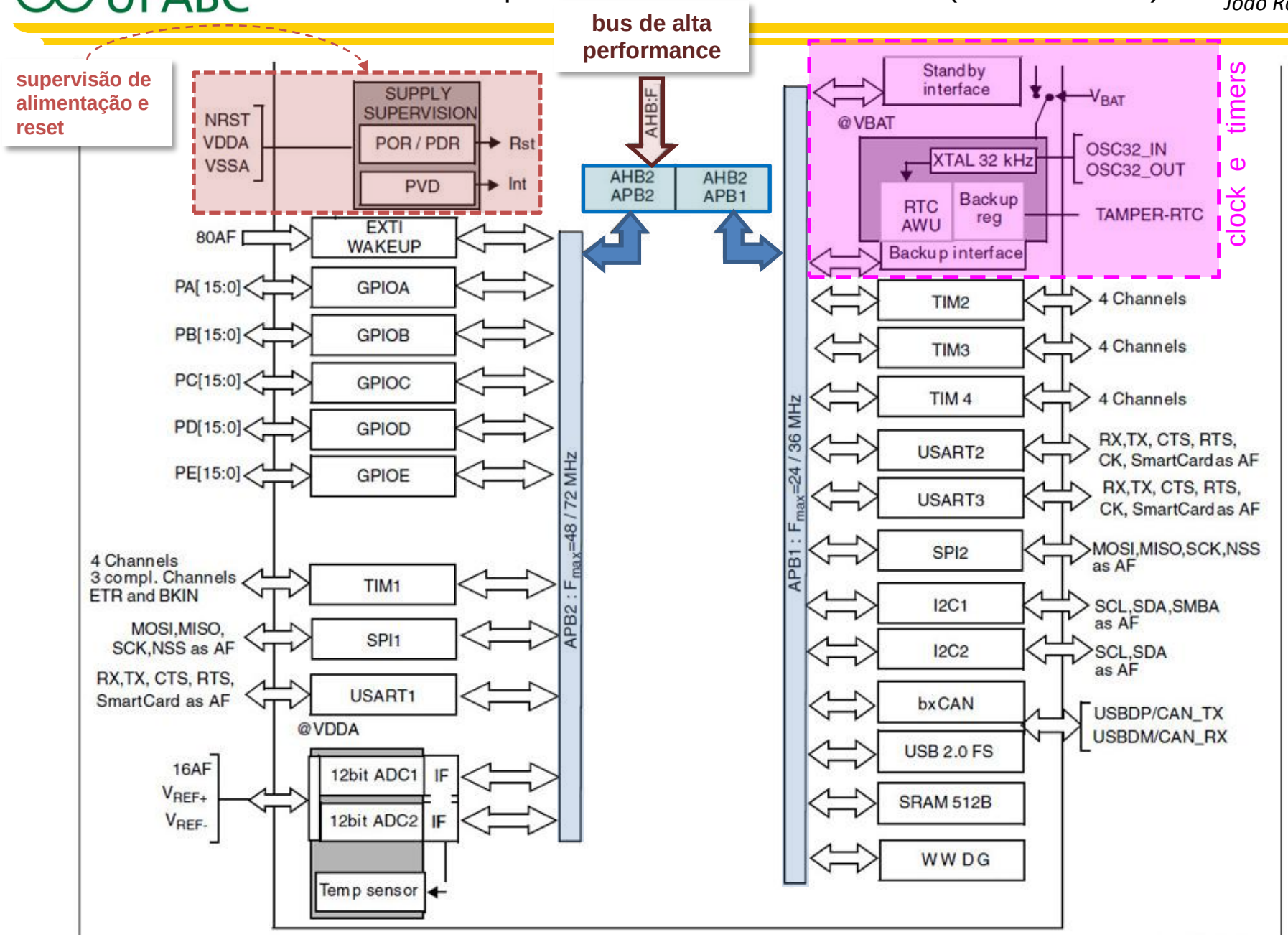
(ponte) bus-bridge

barramento de periféricos

UFABC

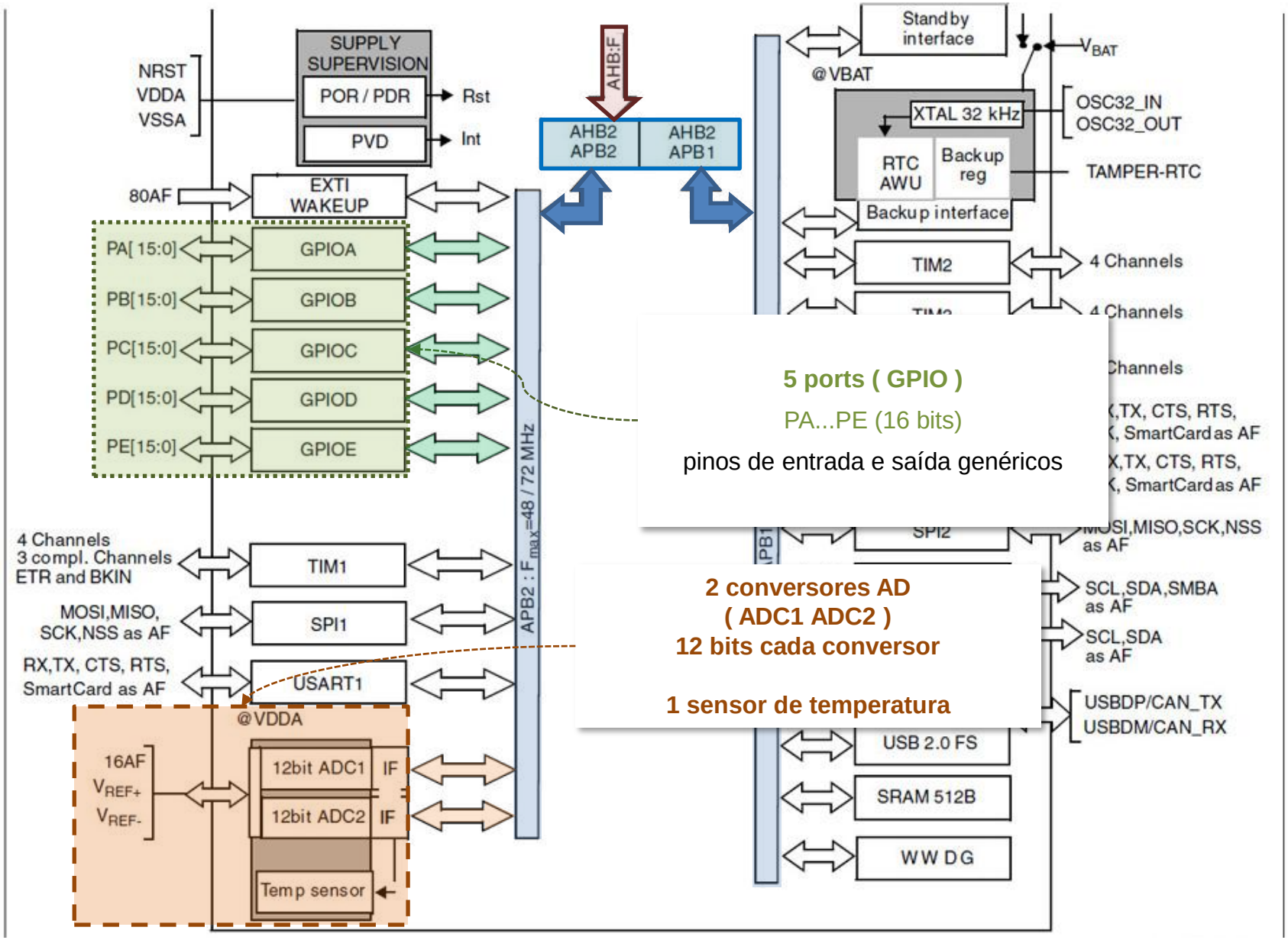
✧ CORE do microprocessador **ARM CORTEX M3** (STM32F103...)

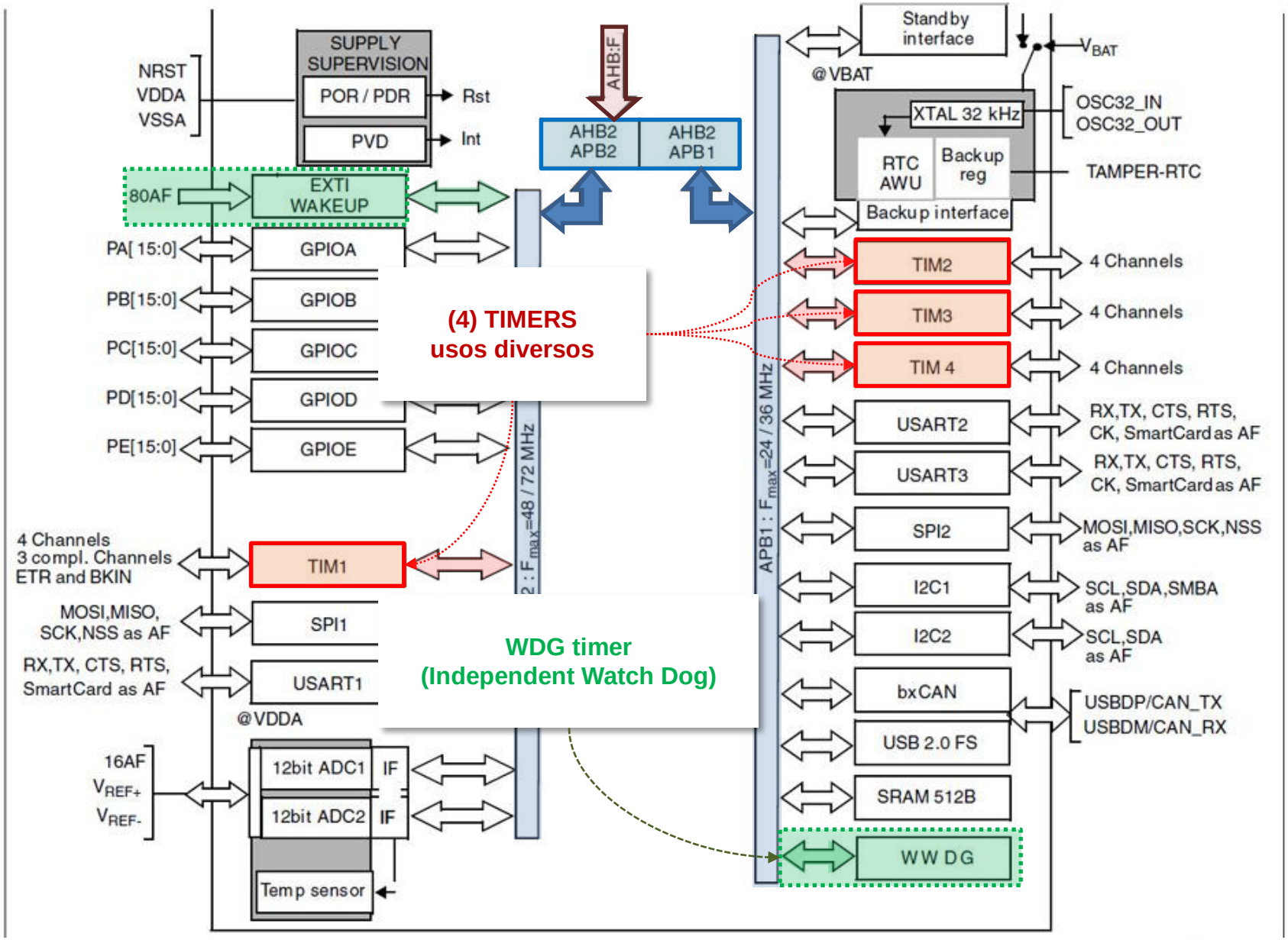
João Ranhel

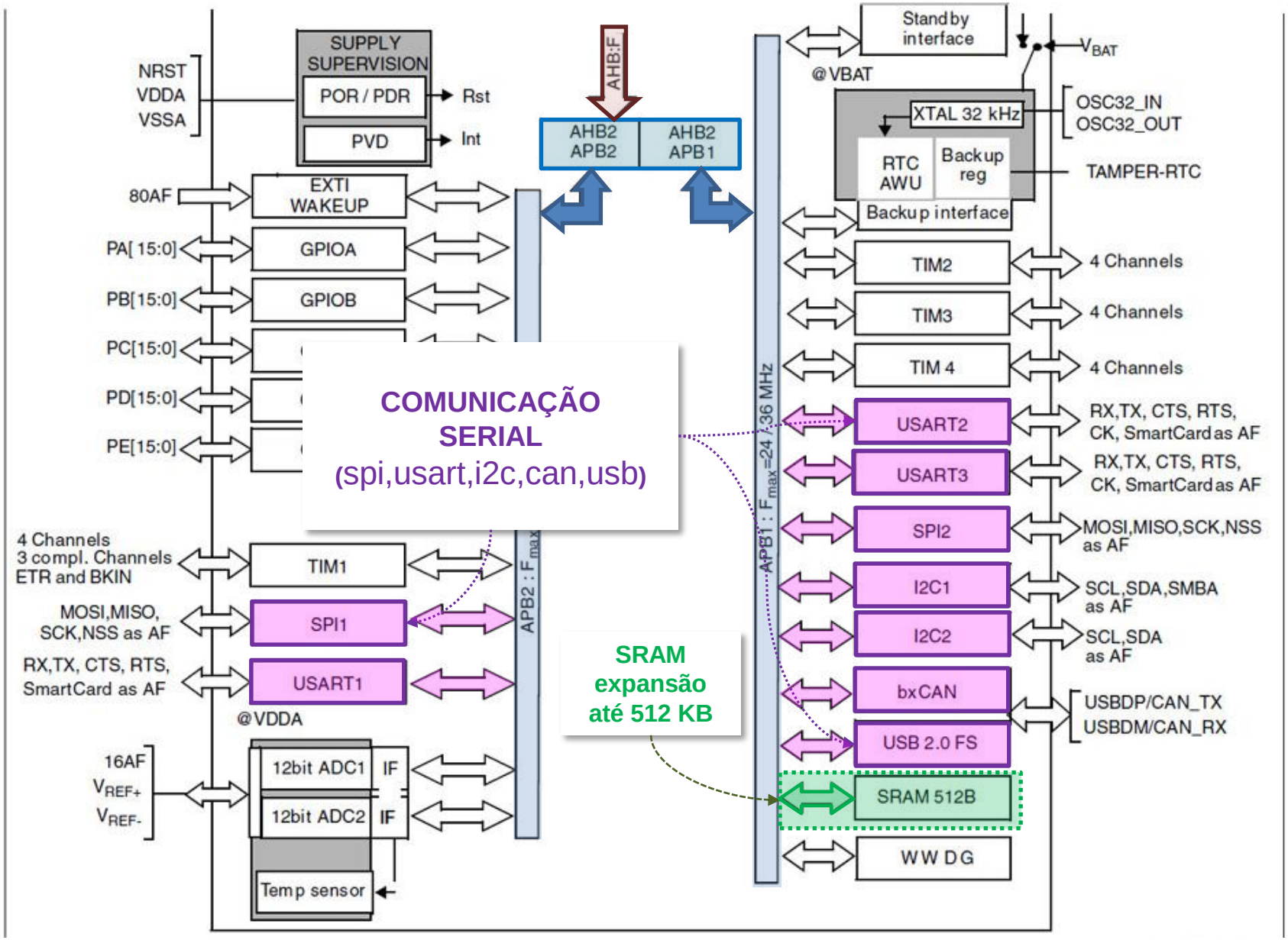


✧ CORE do microprocessador **ARM CORTEX M3** (STM32F103...)

João Ranhel




SUPPLY SUPERVISION
NRST
VDDA
VSSA
POR / PDR
PVD
Rst
Int
AHB:F
AHB2 APB2
AHB2 APB1
80AF
EXTI WAKEUP
PA[15:0]
GPIOA
PB[15:0]
GPIOB
PC[15:0]
GPIOC
PD[15:0]
GPIOD
PE[15:0]
GPIOE
(4) TIMERS
usos diversos
WDG timer
(Independent Watch Dog)
4 Channels
3 compl. Channels
ETR and BKIN
TIM1
MOSI,MISO,
SCK,NSS as AF
SPI1
RX,TX, CTS, RTS,
SmartCard as AF
USART1
@VDDA
16AF
VREF+
VREF-
12bit ADC1
12bit ADC2
IF
Temp sensor
APB2 : Fmax=48 / 72 MHz
APB1 : Fmax=24 / 36 MHz
Standby interface
@VBAT
VBAT
XTAL 32 kHz
OSC32_IN
OSC32_OUT
RTC
AWU
Backup reg
TAMPER-RTC
Backup interface
TIM2
TIM3
TIM 4
4 Channels
USART2
USART3
RX,TX, CTS, RTS,
CK, SmartCard as AF
SPI2
MOSI,MISO,SCK,NSS
as AF
I2C1
I2C2
SCL,SDA,SMBA
as AF
SCL,SDA
as AF
bxCAN
USB 2.0 FS
USBDP/CAN_TX
USBDM/CAN_RX
SRAM 512B
WWDG


SUPPLY SUPERVISION
NRST
VDDA
VSSA
POR / PDR
PVD
Rst
Int
AHB:F
AHB2 APB2
AHB2 APB1
80AF
EXTI WAKEUP
PA[15:0]
GPIOA
PB[15:0]
GPIOB
PC[15:0]
PD[15:0]
PE[15:0]
COMUNICAÇÃO SERIAL
(spi,usart,i2c,can,usb)
4 Channels
3 compl. Channels
ETR and BKIN
TIM1
MOSI,MISO,
SCK,NSS as AF
SPI1
RX,TX, CTS, RTS,
SmartCard as AF
USART1
@VDDA
16AF
VREF+
VREF-
12bit ADC1
12bit ADC2
IF
Temp sensor
APB2 : Fmax
SRAM expansão até 512 KB
Standby interface
@VBAT
VBAT
XTAL 32 kHz
OSC32_IN
OSC32_OUT
RTC AWU
Backup reg
TAMPER-RTC
Backup interface
TIM2
TIM3
TIM 4
4 Channels
USART2
USART3
RX,TX, CTS, RTS,
CK, SmartCard as AF
SPI2
MOSI,MISO,SCK,NSS
as AF
I2C1
I2C2
SCL,SDA,SMBA
as AF
SCL,SDA
as AF
bxCAN
USB 2.0 FS
USBDP/CAN_TX
USBDM/CAN_RX
SRAM 512B
WWDG
APB1 : Fmax=24 /36 MHz

✧ Resumo da aula de hoje...

- Vimos os conceitos básicos do núcleo do processador ARM;
- ARM possui 16 registradores de uso geral (*3 reservados: PC, LR, SP*)
- Os registradores são interligados entre si, com a RAM e a FLASH via matriz de barramentos de alta performance (*AMBA-AXI*);
- Também são interligados com periféricos via barramento mais lento (*AMBA-APB*)
- Registradores são de 32 bits – podem conter *endereços* ou *dados*!
- Mapa de memória é predefinido (2^{32} = 4 Gbytes de endereços)
- Memória e Periféricos estão mapeados em um único espaço de endereços
- A ARM licencia o núcleo para que fabricantes agreguem periféricos/memória