

Aula 1

- Microprocessadores *versus* microcontroladores
- Sistemas embebidos
- Desenvolvimento de aplicações para microcontroladores
- O Microcontrolador PIC32 da Microchip
- Ferramentas de desenvolvimento para a placa DETPIC32

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

Microcontroladores *versus* Microprocessadores

- Microprocessador:
 - Circuito integrado com **um** (ou mais) **CPU**
 - **Não tem memória interna** (além do banco de registos)
 - Os barramentos estão disponíveis no exterior
 - Para obter um sistema completo é necessário acrescentar RAM, ROM e periféricos
 - Pode operar a frequências elevadas ($> 3\text{GHz}$)
 - Sistemas computacionais de uso geral
- Microcontrolador:
 - Circuito integrado inclui CPU, RAM, ROM, periféricos
 - Frequência de funcionamento normalmente baixa
 - **Baixo consumo de energia**
 - Disponibiliza uma grande variedade de periféricos e interfaces com o exterior
 - Rapidez de resposta a eventos externos (Sistemas de Tempo Real)
 - Utilizado em **tarefas específicas** (por exemplo controlo da velocidade de um motor)

Sistema embebido

- Sistema computacional especializado
 - realiza uma tarefa **específica** ou o controlo de um determinado dispositivo
- Tem requisitos próprios e executa apenas tarefas pré-definidas
- Recursos disponíveis, em geral, mais limitados que num sistema computacional de uso geral (e.g. menos memória, ausência de dispositivos de interação com o utilizador)
- Tem, em regra, um custo inferior a um sistema computacional de uso geral
- Pode ser implementado com base num microcontrolador
- Pode fazer parte de um sistema computacional mais complexo
- Exemplos de aplicação:
 - eletrónica de consumo, automóveis, telecomunicações, domótica, robótica, iot, ...

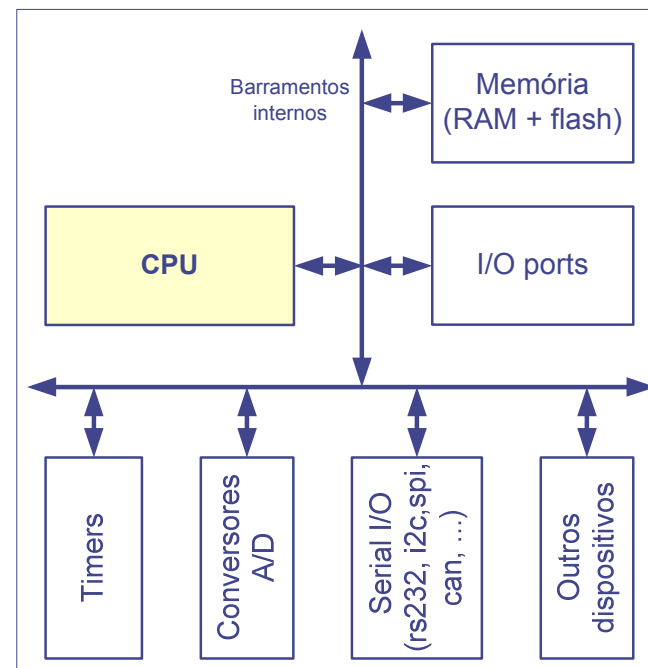
Microcontrolador – principais características

- Dispositivo programável que integra, num único circuito integrado, 3 componentes fundamentais:

- Uma Unidade de Processamento
- Memória (volátil e não volátil)
- Portos de I/O (E/S)

- Inclui outros dispositivos de suporte (periféricos), tais como:

- Timers
- Conversor A/D
- Serial I/O (rs232, i2c, spi, can, ...)
- ...



- Barramentos (dados, endereços e controlo) interligam todos estes dispositivos (não estão, geralmente, acessíveis externamente)
- Externamente há, em geral, pinos que podem ser configurados programaticamente para diferentes funções (versatilidade)

Processo de desenvolvimento de aplicações para μ C

- Computador *host* (e.g. PC) / Computador *target* (μ C)
 - Estas plataformas são, geralmente, distintas (CPU, sistema operativo, dispositivos de interface com o utilizador, ...)
- **Edição do programa** numa linguagem de alto nível (por ex. C), ou, em casos pontuais, em *assembly* do microcontrolador
- **Geração do código** usando um *cross-compiler* / *cross-assembler*
 - Um ***cross-compiler*** (compilador-cruzado) é um compilador que corre na plataforma A (o *host*, e.g. o PC) e que gera código executável para a plataforma B (o *target*, e.g. o μ C)
 - A utilização de *cross-compilers* / *cross-assemblers* é a regra no desenvolvimento de aplicações para microcontroladores uma vez que, geralmente, estes não disponibilizam os recursos necessários e as interfaces adequadas
- **Transferência para a memória do microcontrolador** (geralmente memória não volátil) do código produzido pelo *cross-compiler* / *cross-assembler*
- **Teste e depuração** (*debug*) do programa



Transferência de programas para o microcontrolador

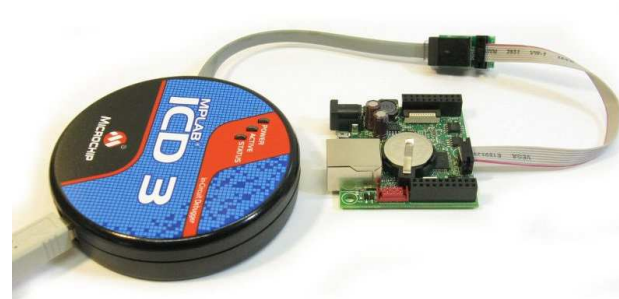
- Para a transferência de um programa executável para a memória do microcontrolador pode ser utilizado um dos seguintes métodos:
 - **Programa-monitor** (software)
 - ***Bootloader*** (software)
 - ***In-Circuit Debugger*** (hardware)
- Programas-monitor e *bootloaders*:
 - Executam no arranque do sistema
 - A comunicação com o *host* é efetuada por RS232 / USB
- *In-Circuit Debugger*
 - Dispositivo externo proprietário, i.e., específico para um dado fabricante
 - Pode usar uma interface de comunicação standard (JTAG) ou uma interface proprietária

Transferência de programas para o microcontrolador

- **Programa-monitor:** é um programa que reside, de forma permanente, na memória não volátil do microcontrolador:
 - disponibiliza funções de transferência e execução de programas
 - implementa outras funções úteis no *debug* de novos programas, como por exemplo, visualização do conteúdo de registos internos do microprocessador, visualização do conteúdo da memória, execução passo a passo, etc.
- **Bootloader:** é um programa que reside, de forma permanente, na memória do microcontrolador e que disponibiliza apenas funções básicas de transferência e execução de um programa.

Transferência de programas para o microcontrolador

- ***In-Circuit Debugger (ICD)***: um dispositivo de hardware controlado por software no *host* que permite a transferência e execução controlada de um programa num microcontrolador



- O ICD é, normalmente, necessário para a transferência inicial de um programa-monitor ou de um *bootloader*.

Tecnologias de memória não volátil

- **ROM** – programada durante o processo de fabrico
- **PROM** – *Programmable Read Only Memory*: programável uma única vez
- **EPROM** – *Erasable PROM*: escrita em segundos, apagamento em minutos (ambas efetuadas em dispositivos especiais)
- **EEPROM** – *Electrically Erasable PROM*
 - O apagamento e a escrita podem ser efetuados no próprio circuito em que a memória está integrada
 - O apagamento é feito byte a byte
 - Escrita muito mais lenta que leitura
- **Flash EEPROM** (tecnologia semelhante à EEPROM)
 - A escrita pressupõe a inicialização (*reset*) prévia das zonas de memória a escrever
 - O *reset* é feito por blocos (por exemplo, blocos de 4 kB) o que torna esta tecnologia mais rápida que a EEPROM
 - O *reset* e a escrita podem ser efetuados no próprio circuito em que a memória está integrada
 - Escrita muito mais lenta que a leitura

Microcontrolador PIC32 da Microchip

- Microcontrolador **PIC32MX795F512H**:

- CPU MIPS
- Conjunto alargado de periféricos
- Memória flash: 512 kB (+12 kB Boot flash)
- Memória RAM: 128 kB
- Versão de 64 pinos (também disponível em 100 e 121 pinos)



- **CPU:**

- **MIPS32** M4K (core 32-bits com 5 estágios de *pipeline*)
 - Com coprocessador 0 (exceções e interrupções, gestão de memória)
 - **Não** dispõe de *Floating Point Unit* (coprocessador 1)
- **32 registos de 32 bits (\$0 a \$31)**
- Espaço de endereçamento de 32 bits
- Organização de memória: *byte-addressable*
- Max. frequência de relógio: **80 MHz**

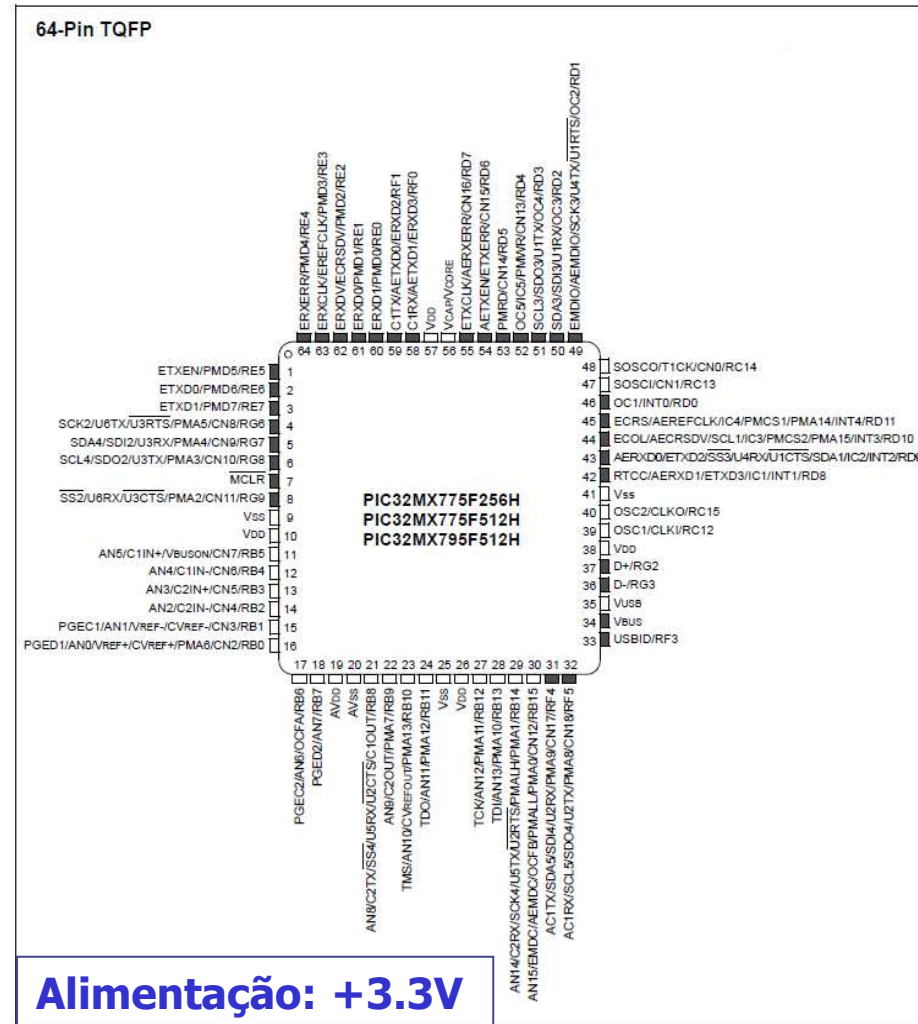
Nunca utilizamos no máx.

- Documentação completa em (link válido em 09/02/2025):

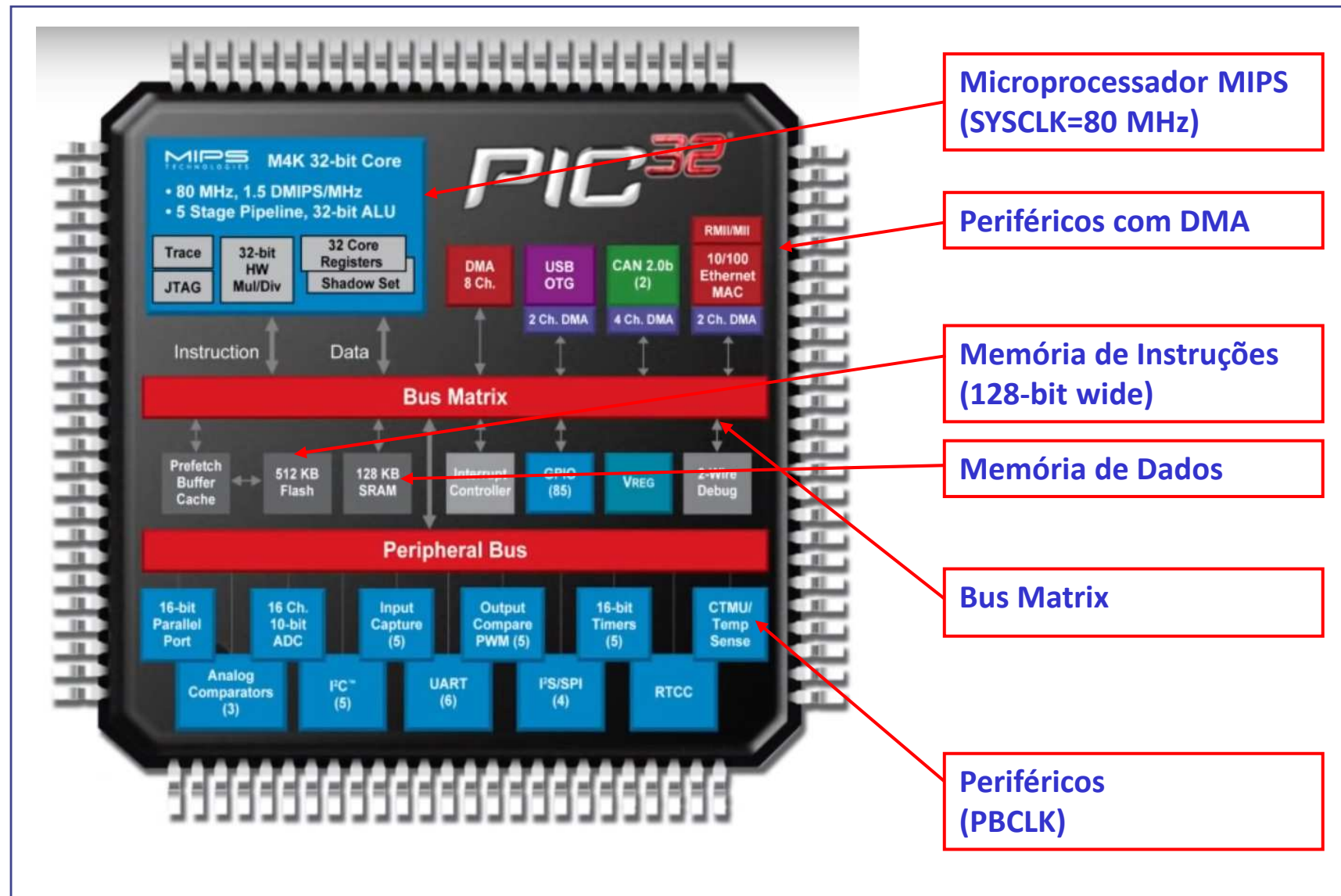
<https://www.microchip.com/en-us/product/PIC32MX795F512H>

Microcontrolador PIC32MX795F512H

- Elevado nível de multiplexagem nos pinos do circuito integrado (cada pino pode ter, na versão de 64 pinos, até 9 funções distintas)
- Função desempenhada por cada pino depende de programação



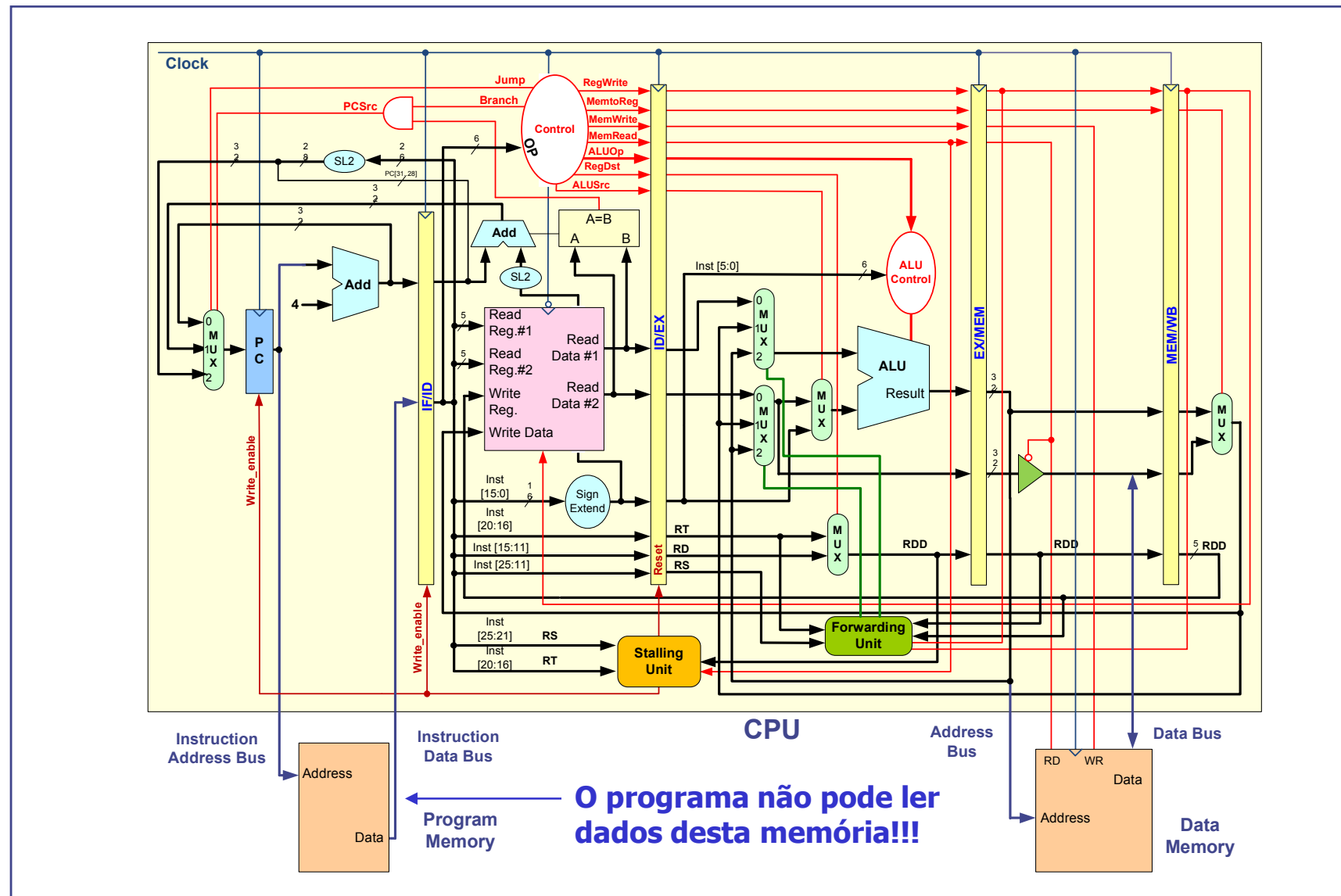
Microcontrolador PIC32MX795F512H



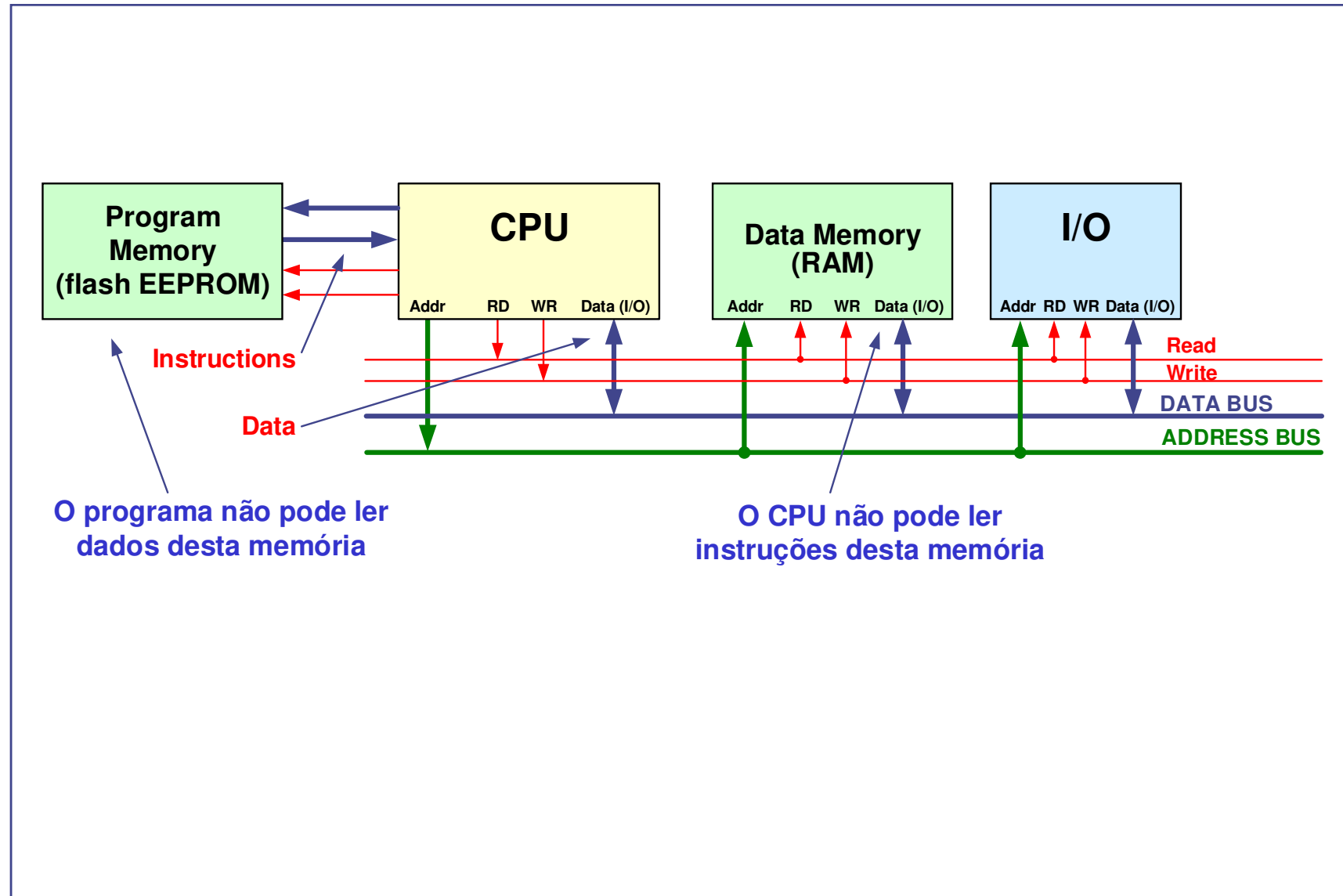
Microcontrolador PIC32MX795F512H

- O MIPS do PIC32 é baseado numa **arquitetura de Harvard** (memória de instruções e dados separadas). Esta opção evita o *hazard* estrutural na implementação *pipelined* que aconteceria com uma única memória.
- Numa arquitetura de Harvard:
 - Existem **dois** espaços de endereçamento independentes: um para o **programa** e outro para **dados**.
 - **Apenas** o bloco encarregue da **leitura das instruções da memória** (*instruction fetch*) tem **acesso** à **memória de programa**.
 - O **programa não** pode ler dados da **memória** de instruções.
 - O **CPU não** pode ler instruções da **memória** de dados
 - É difícil o tratamento das constantes (por exemplo *strings*) uma vez que estas não podem ser armazenadas juntamente com o programa na memória de instruções (tipicamente uma memória não volátil)

Versão simplificada de uma arquitetura MIPS *pipelined*

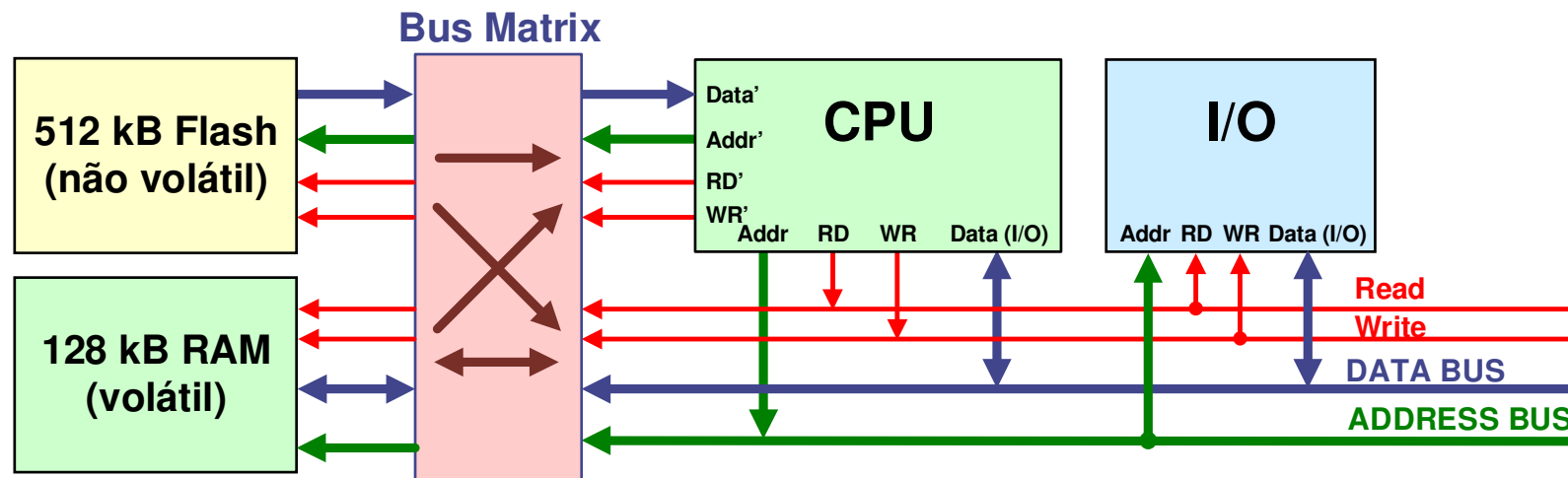


Arquitetura de Harvard convencional



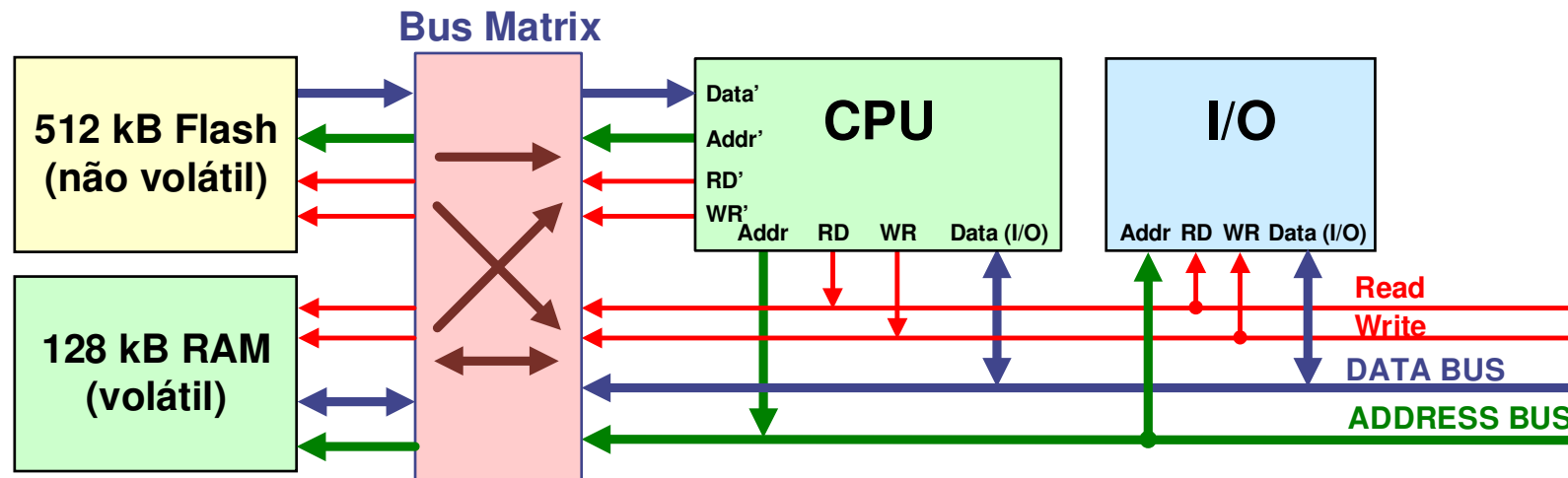
Microcontrolador PIC32MX795F512H

- Solução implementada no PIC32 que resolve o problema dos dados constantes da arquitetura de Harvard: **Bus Matrix**



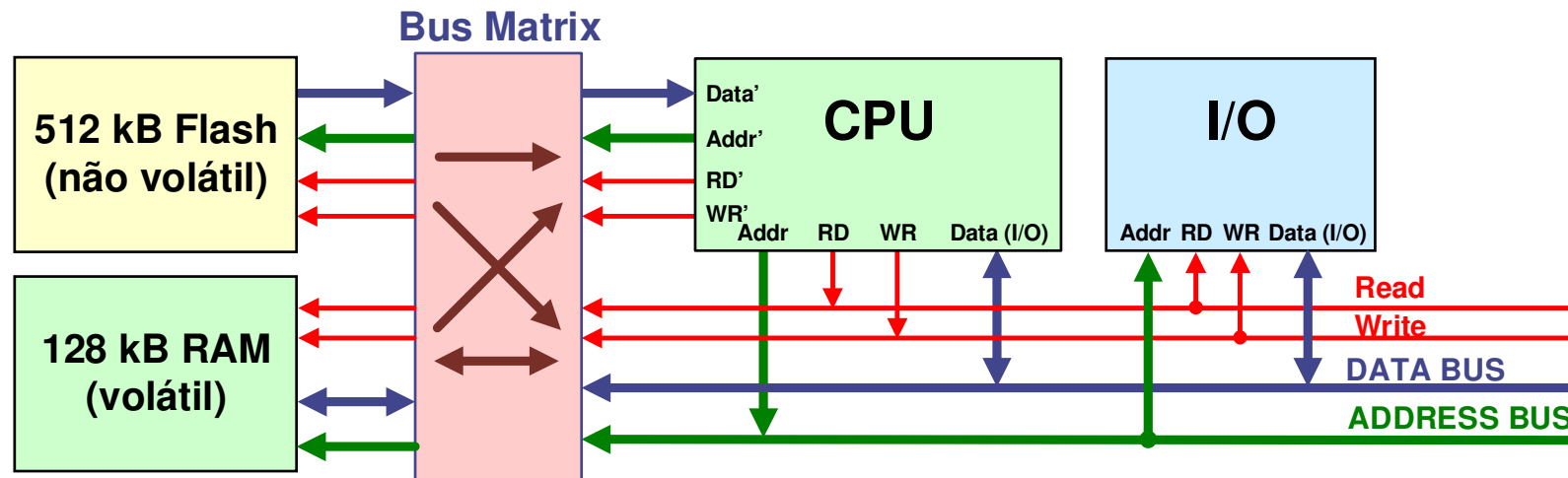
- O **Bus Matrix** é um comutador (*switch*) de alta velocidade que funciona à mesma frequência do CPU (SYSCLK)
- Estabelece ligações ponto a ponto entre os módulos do microcontrolador, em particular, entre o CPU e a memória RAM ou entre o CPU e a memória *Flash*
- O *peripheral bus* também liga ao *Bus Matrix* e pode ser configurado para trabalhar a uma frequência igual ou inferior à do CPU (PBCLK)

Microcontrolador PIC32MX795F512H



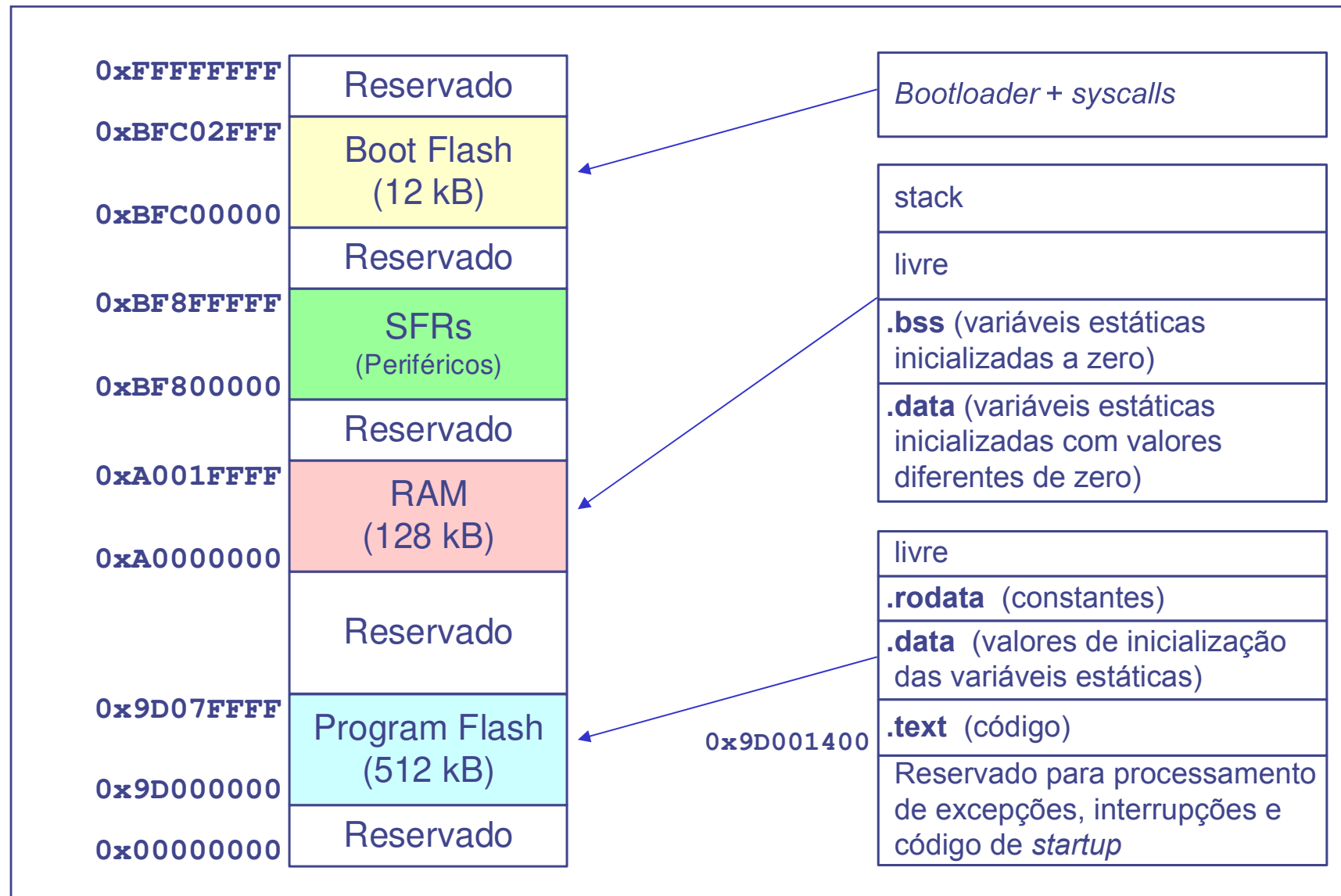
- Com o *Bus Matrix*, o espaço de endereçamento aparece, na visão do programador, como um espaço linear unificado (instruções e dados residem no mesmo espaço de endereçamento, cada um deles ocupando uma gama de endereços única)
- O CPU pode então executar programas que residem quer na *Flash* quer na RAM
- O programa gerado pelo *host* (instruções + dados constantes) pode ser armazenado na totalidade na memória *Flash* – programa pode aceder a qualquer momento à *Flash* para ler dados (por exemplo *strings*)

Microcontrolador PIC32MX795F512H



- Para o programador, o PIC32 comporta-se como uma arquitetura de *von Neumann*: um único espaço de endereçamento onde residem dados e instruções
- Para que o programa tenha acesso a qualquer zona da memória Flash (para leitura) ou da memória RAM (para leitura ou escrita), basta definir adequadamente o respetivo endereço

Mapa de memória do PIC32 (versão DETPIC32)



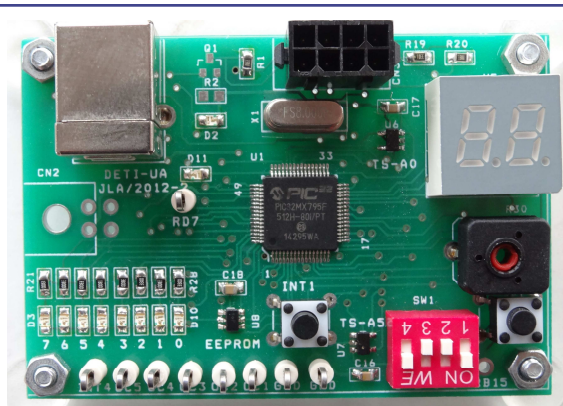
Ferramentas de desenvolvimento DETPIC32

- Edição

- GVim, gedit, geany...

- *Cross-compiler / cross-assembler*

- **gcc** com *back-end* para MIPS (gcc-pic32)
- Gera, entre outros, ficheiros ".hex" e ".map"



- Ferramentas desenvolvidas especificamente para DETPIC32

- **bootloader** – programa previamente gravado na *boot flash* do PIC32; lê informação do porto de comunicação e escreve na memória *Flash*
- **ldpic32** – programa para transferir ficheiro ".hex" para PIC32 (atua em conjunto com o *bootloader*)
- **pterm** – programa terminal para comunicação com a placa DETPIC32, permitindo a interação com o utilizador
- **hex2asm** – faz o *disassemble* do ficheiro ".hex" (utiliza o ficheiro ".map", para evidenciar secções / símbolos relevantes)

• "fcompile"