Aula 1

- Microprocessadores *versus* microcontroladores
- Sistemas embebidos
- Desenvolvimento de aplicações para microcontroladores
- O Microcontrolador PIC32 da Microchip
- Noção de periférico; estrutura básica de um módulo de I/O; registos de controlo, status e dados; modelo de programação

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

Microcontroladores *versus* Microprocessadores

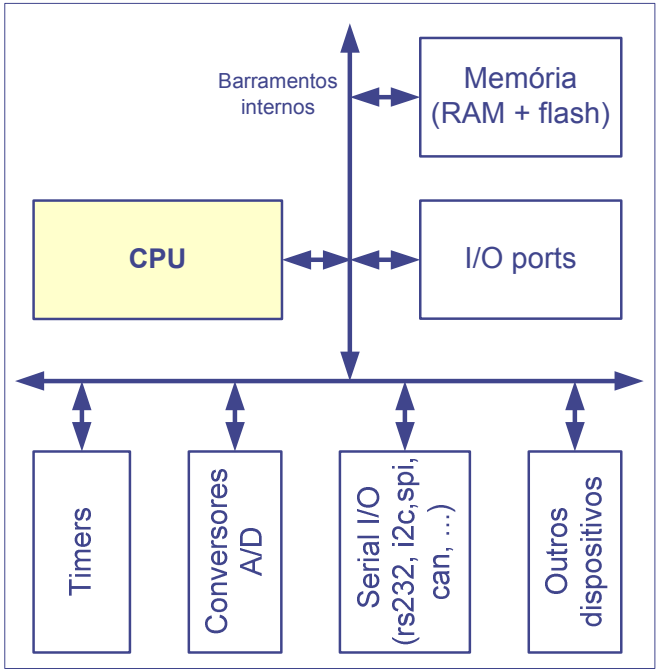
- Microprocessador:
  - Circuito integrado com um (ou mais) CPU
  - Não tem memória interna (além do banco de registos)
  - Os barramentos estão disponíveis no exterior
  - Para obter um sistema completo é necessário acrescentar RAM, ROM e periféricos
  - Pode operar a frequências elevadas (> 3GHz)
  - Sistemas computacionais de uso geral
- Microcontrolador:
  - Circuito integrado que inclui CPU, RAM, ROM e 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 caraterísticas

- 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 μC

- Computador *host* (e.g. PC) / Computador *target* (μC)
  - 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 μC)
  - 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

- **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 a visualização do conteúdo de registos internos do CPU e 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
- ***In-Circuit Debugger***: é um dispositivo de hardware controlado por software no *host* que permite a transferência e execução controlada de um programa num microcontrolador
  - específico para um dado fabricante
  - pode usar uma interface de comunicação standard (JTAG) ou uma interface proprietária

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

S

- **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

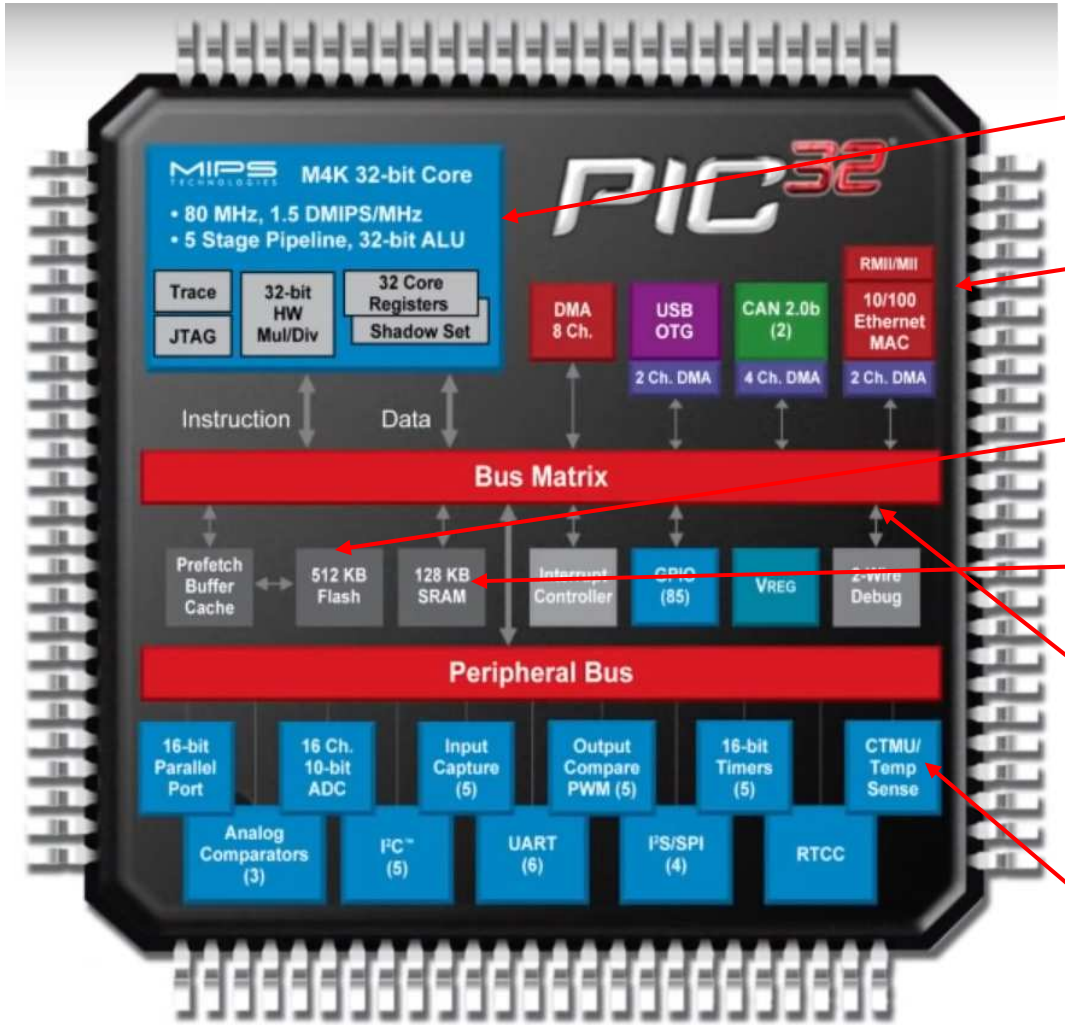
Exemplo de Microcontrolador – PIC32 da Microchip

- 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
- Documentação completa em (link válido em 26/01/2026):

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

Microcontrolador PIC32MX795F512H



Microprocessador MIPS (SYSCLK=80 MHz)

Periféricos com DMA

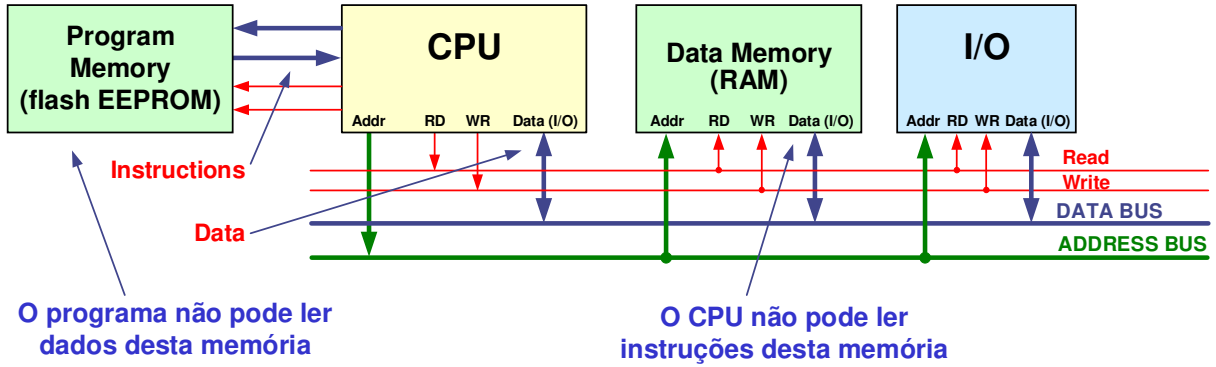
Memória de Instruções (128-bit wide)

Memória de Dados

Bus Matrix

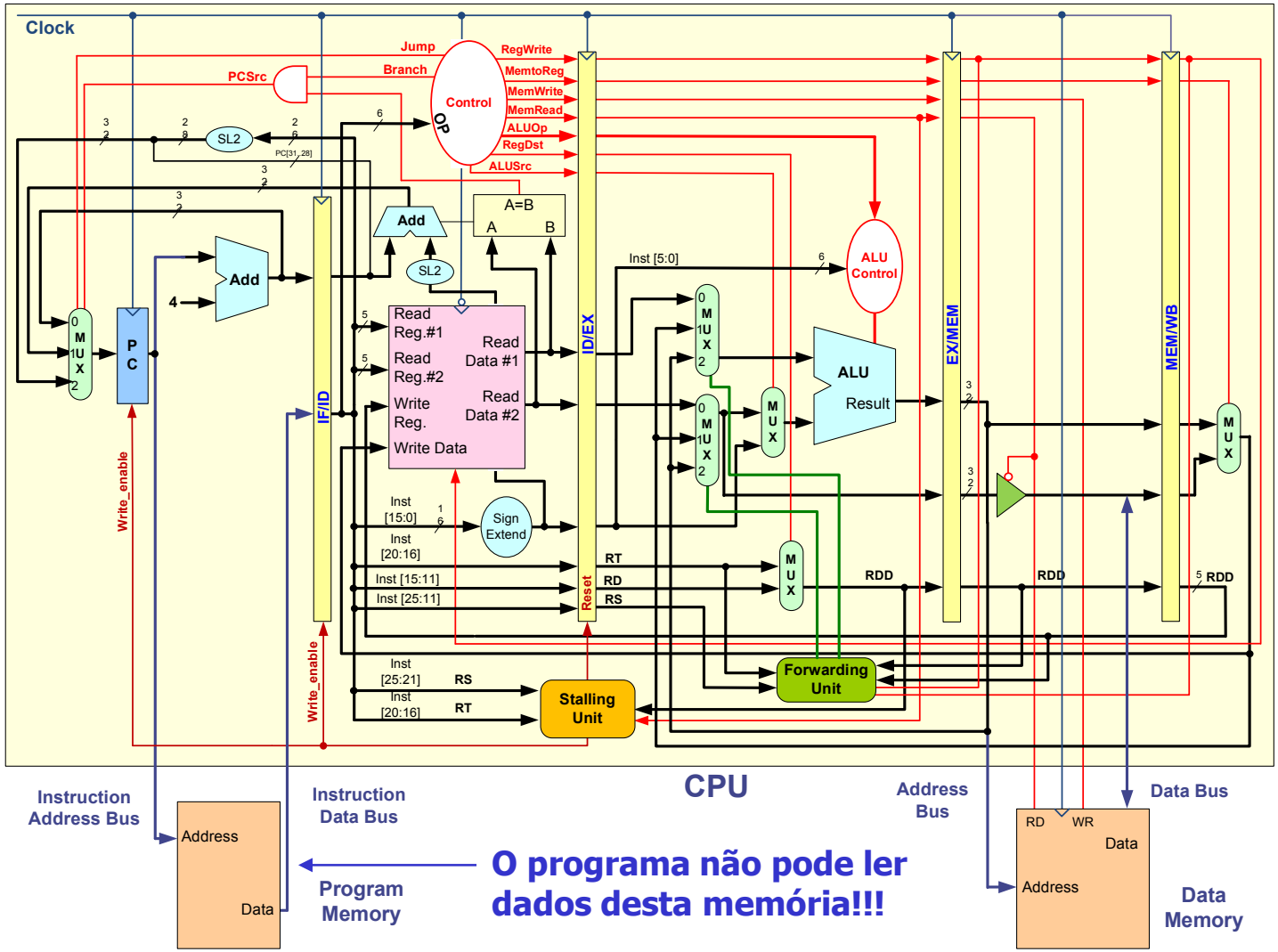
Periféricos (PBCLK)

Microcontroladores baseados em arquitetura de Harvard



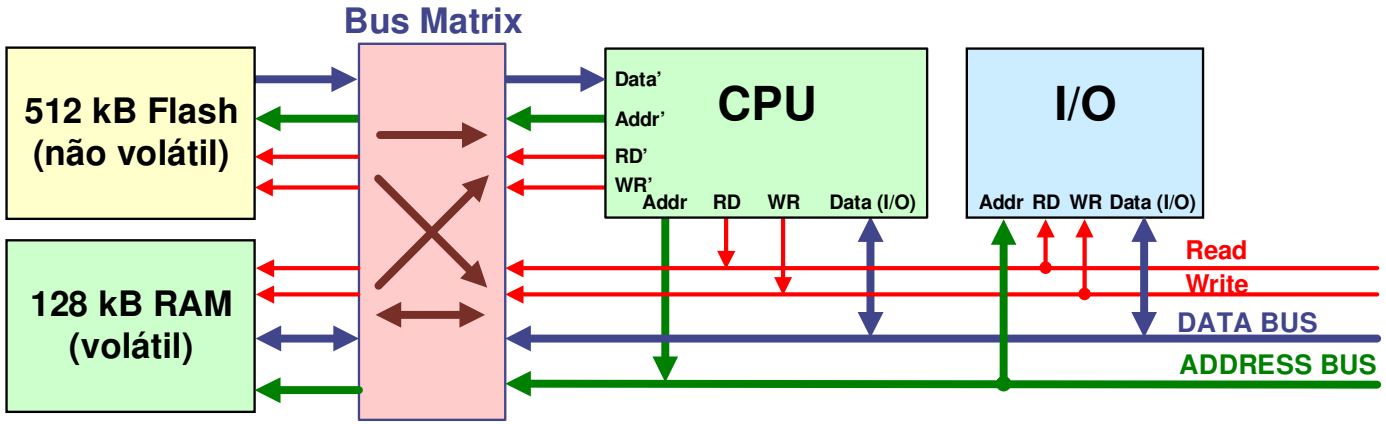
- Dois espaços de endereçamento independentes: um para instruções e outro para dados
- Apenas o bloco encarregue da leitura das instruções da memória (*instruction fetch*) tem acesso à memória de programa
- O programa não pode aceder a dados armazenados na memória de instruções
- O CPU não pode executar instruções armazenadas na memória de dados
- O tratamento das constantes (ex. *strings*) é dificultado pois não podem ser armazenadas juntamente com as instruções (tipicamente uma memória não volátil, ex. *flash*)
- O MIPS é baseado numa **arquitetura de Harvard** (evita o *hazard* estrutural na implementação *pipelined* que aconteceria com uma única memória)

Versão simplificada de uma arquitetura MIPS *pipelined*



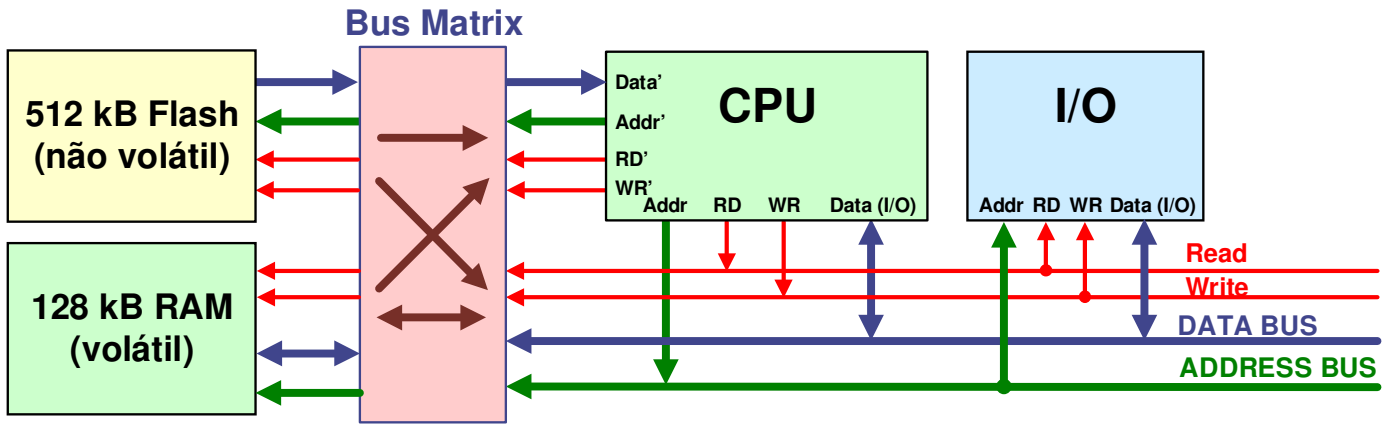
Microcontrolador PIC32MX795F512H – *Bus Matrix*

- Solução implementada no PIC32 para ultrapassar a limitação da arquitetura de Harvard no acesso a dados constantes (armazenados na memória *flash*): ***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 diferentes blocos do microcontrolador, nomeadamente:
  - entre o bloco IF ("Instruction Fetch") do CPU e a memória Flash
  - entre o bloco IF e a memória RAM
  - entre o bloco MEM ("Data Memory Access") do CPU e a memória RAM
  - entre o bloco MEM e a memória Flash

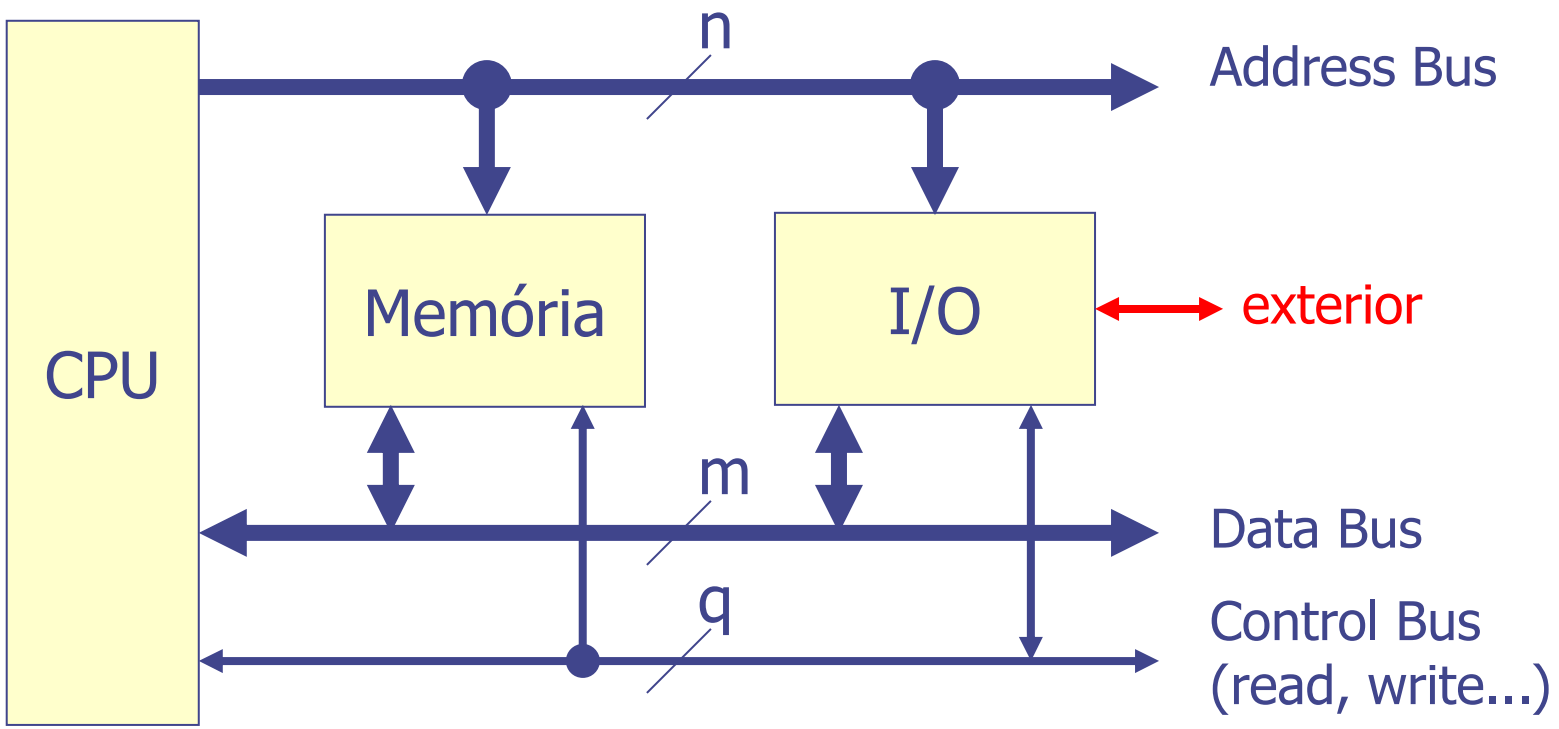
Microcontrolador PIC32MX795F512H – *Bus Matrix*



- 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)
- Para o programador, o PIC32 comporta-se como uma arquitetura de *von Neumann*: um único espaço de endereçamento onde residem dados e instruções
- 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*)

- Noção de periférico
  - estrutura básica de um módulo de I/O
  - registos de controlo, status e dados
  - modelo de programação

Troca de informação com o exterior - periférico

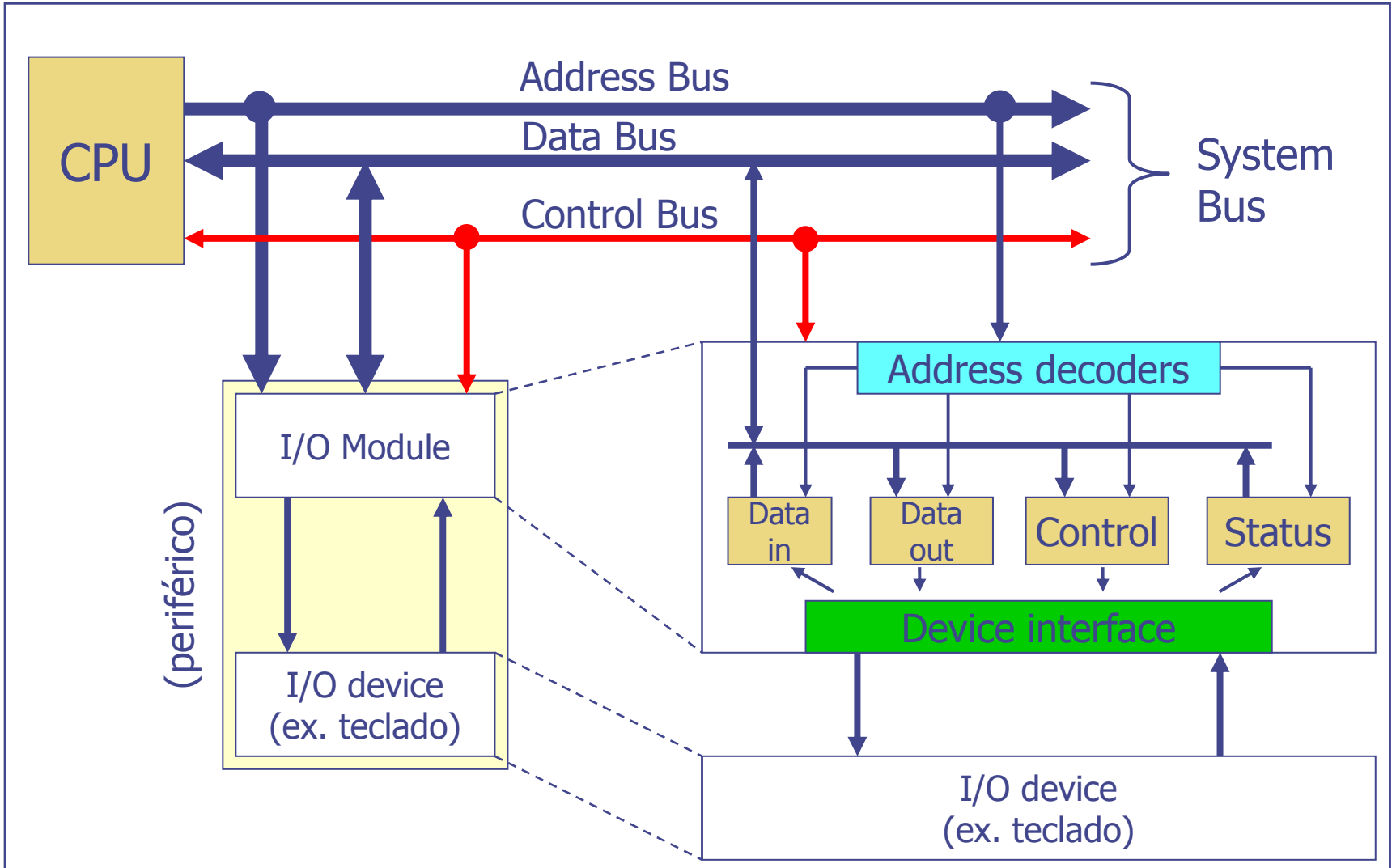


- O "sistema de entradas/saídas" providencia os mecanismos e os recursos para a comunicação entre o sistema computacional e o exterior
- Troca de informação com o exterior:
  - Processo é designado, genericamente, como Input/Output (I/O)
  - O dispositivo que assegura a comunicação designa-se por periférico

Troca de informação com o exterior - periférico

- Dispositivos periféricos:
  - grande variedade (por exemplo: conversor A/D, timer, controlador Ethernet, controlador USB, …)
  - com métodos de operação específicos
  - assíncronos relativamente ao CPU
  - geram diferentes quantidades de informação com diferentes formatos a diferentes velocidades (de alguns bits/s a dezenas de Megabyte/s)
  - mais lentos que o CPU e a memória
- É necessária uma interface que adapte as características do dispositivo periférico às do CPU/memória
  - **Módulo de I/O**

Módulo de I/O



- Dependendo da sua complexidade, o periférico tem atribuído um ou mais endereços

Módulo de I/O

- O módulo de I/O pode ser entendido como o elemento que assegura a compatibilização entre as características (e o modo de funcionamento) do sistema computacional e as do dispositivo físico propriamente dito
- Ao nível do hardware:
  - Adequa as características do dispositivo físico de I/O às características do sistema digital ao qual tem que se ligar; tal como a memória, o periférico comunica com o sistema através dos barramentos
- Na interação com o dispositivo físico:
  - Lida com as particularidades do dispositivo, por exemplo, formatação de dados, deteção e gestão de situações de erro, ...
- Ao nível do software:
  - Adequa o dispositivo físico à forma de organização do sistema computacional, disponibilizando e recebendo informação através de registos; esta solução esconde do programador a complexidade e os detalhes de implementação do dispositivo periférico

Módulo de I/O

- O módulo de I/O permite ao processador ver um modelo simplificado do periférico, escondendo os detalhes de funcionamento interno
- Com a adoção do módulo de I/O, o dispositivo periférico, independentemente da sua natureza e função, passa a ser encarado pelo processador como uma coleção de registos de dados, de controlo e de *status*
- A comunicação entre o processador e o periférico é assegurada por operações de escrita e de leitura, em tudo semelhantes a um acesso a uma posição de memória (por ex., LW e SW no caso do MIPS)
  - Ao contrário do que acontece na memória, o valor associado a estes endereços pode mudar sem intervenção do CPU
- O conjunto de registos e a descrição de cada um deles são específicos para cada periférico e constituem o que se designa por **modelo de programação do periférico**

Módulo de I/O – modelo de programação

- **Data Register**(s) (*Read/Write*)
  - Registo(s) onde o processador coloca a informação a ser enviada para o periférico (*write*) e de onde lê informação proveniente do periférico (*read*)
- **Status Register**(s) (*Read only*)
  - Registo(s) que engloba(m) um conjunto de bits que dão informação sobre o estado do periférico (ex. operação terminada, informação disponível, situação de erro, ...)
- **Control Register**(s) (*Write only* ou *Read/Write*)
  - Registo(s) onde o CPU escreve informação sobre o modo de operação do periférico (comandos)
- É comum um só registo incluir as funções de controlo e de *status*. Nesse caso, um conjunto de bits desse registo está associado a funções de controlo (*read/write* ou *write only bits*) e outro conjunto a funções de status (*read only bits*)

Comunicação entre o CPU e outros dispositivos

- No modelo mais simples (sem DMA), a iniciativa da comunicação é sempre do CPU, no contexto da execução das instruções
- A comunicação entre o CPU e um dispositivo externo obedece a um **protocolo** definido que envolve sinais de controlo e sincronização
- As duas operações possíveis são:
  - ***Write*** (transferência de dados do CPU para o dispositivo externo: CPU $\rightarrow$ dispositivo externo)
  - ***Read*** (transferência de dados do dispositivo externo para o CPU: CPU $\leftarrow$ dispositivo externo)
- Uma operação de acesso do CPU a um dispositivo externo envolve usar:
  - O barramento de endereços para especificar o **endereço do registo do módulo de I/O do dispositivo a aceder**
  - O barramento de controlo para indicar qual a operação a realizar (*read* ou *write*)
  - O barramento de dados para transportar a informação, no sentido correspondente à operação

Seleção do dispositivo externo

- **Operação de escrita** (CPU → dispositivo externo)
  - o CPU coloca dados no barramento de dados
  - apenas o dispositivo endereçado deve armazenar esses dados
- **Operação de leitura** (CPU ← dispositivo externo)
  - apenas o dispositivo endereçado pode colocar dados no barramento de dados
  - todos os outros dispositivos devem permanecer em estado *Tri-State* (alta impedância), de modo a não interferirem na comunicação
- **Partilha do barramento de dados**
  - num sistema computacional vários circuitos estão ligados ao barramento de dados (memória e unidades de I/O)
  - apenas o dispositivo endereçado pelo CPU deve ser selecionado (ativado) para comunicar; esta função de seleção é assegurada pela **descodificação de endereços** (ver Aula 6)