Conceitos gerais sobre sistemas baseados em arquitetura de computadores

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

José Luís Azevedo, Bernardo Cunha

Engenharia reversa ao nível do sistema

- Objetivos possíveis:
  - Reconstruir integralmente um sistema completo incluindo o hardware
  - Obter o esquema interno de um único circuito integrado
  - Identificar os elementos (circuitos) que constituem um sistema
  - Reconstituir o software que é executado num módulo computacional
  - Obter um modelo funcional parcial ou completo de um sistema por forma a reproduzir esse modelo mesmo que com base em elementos físicos distintos.

Engenharia reversa ao nível do sistema

Objetivo no âmbito de ER

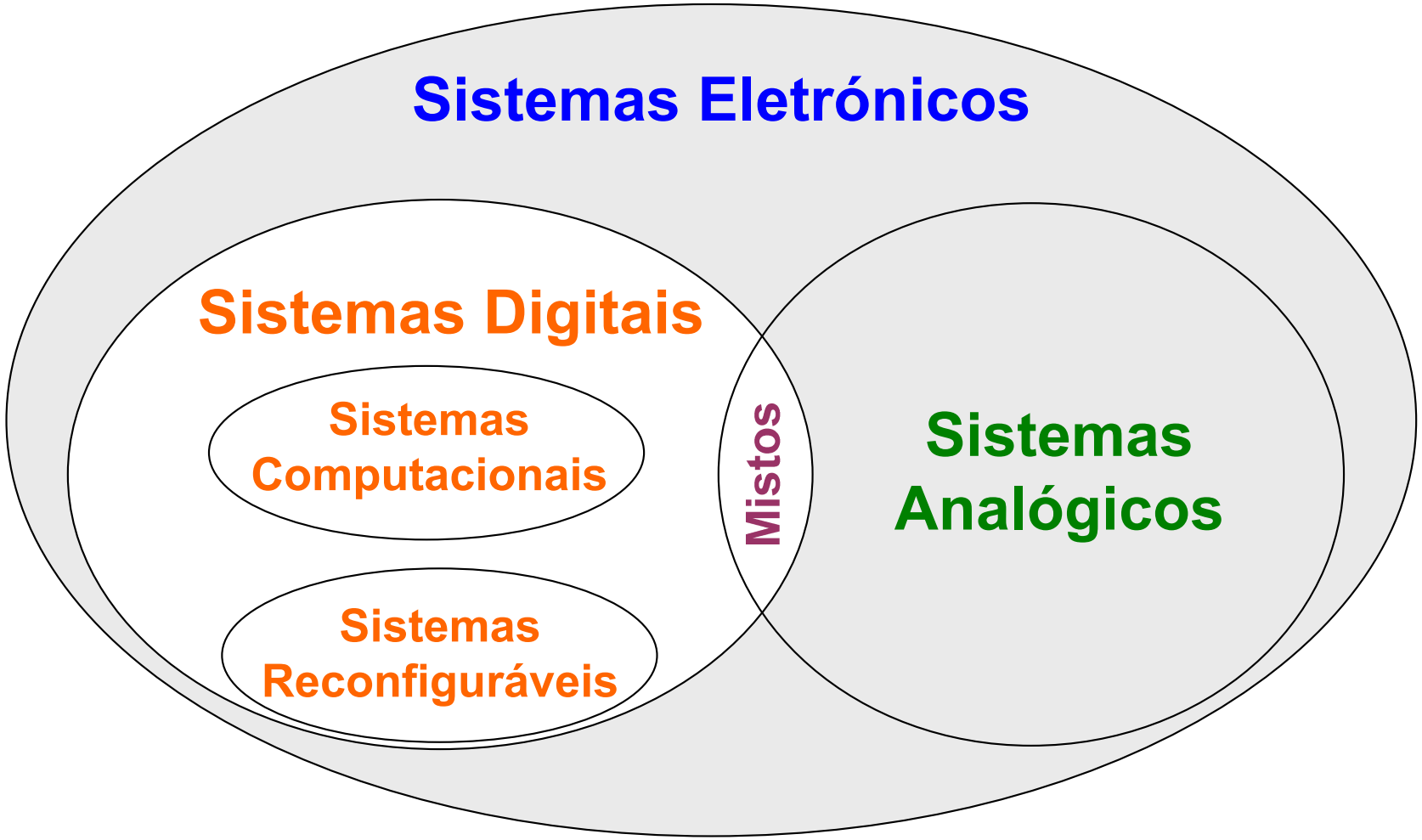
- Assim:
  - Embora seja possível fazer a reconstituição de circuitos de elevado nível de integração isso requer métodos invasivos e destrutivos e tecnologia de elevado custo
  - Em grande parte das situações o objetivo é replicar a funcionalidade, mesmo que com recurso a componentes distintos
  - Muitos circuitos/componentes *off the shelf* não protegem o software built-in e disponibilizam mecanismos de interação para fazer o *dump* da memória interna.
- Pelo que o foco na parte que se segue visa fundamentalmente:
  - Reconstituir o software que é executado num módulo computacional de um sistema
  - Obter um modelo funcional parcial ou completo dum sistema por forma a reproduzir esse modelo mesmo que com base em elementos físicos distintos.

Engenharia reversa ao nível do sistema

Abordagem

- Para cumprir estes objetivos há dois níveis de conhecimento básico que é necessário assegurar
  - Perceber os conceitos (a um nível muito básico) da arquitetura de um sistema computacional e dos seus principais componentes
  - Compreender algumas (poucas) das normas de comunicação série que permitem interligar localmente (ao nível do circuito impresso), ou em sistemas distribuídos de dimensão local, os blocos funcionais que possam compor um determinado circuito computacional dedicado.

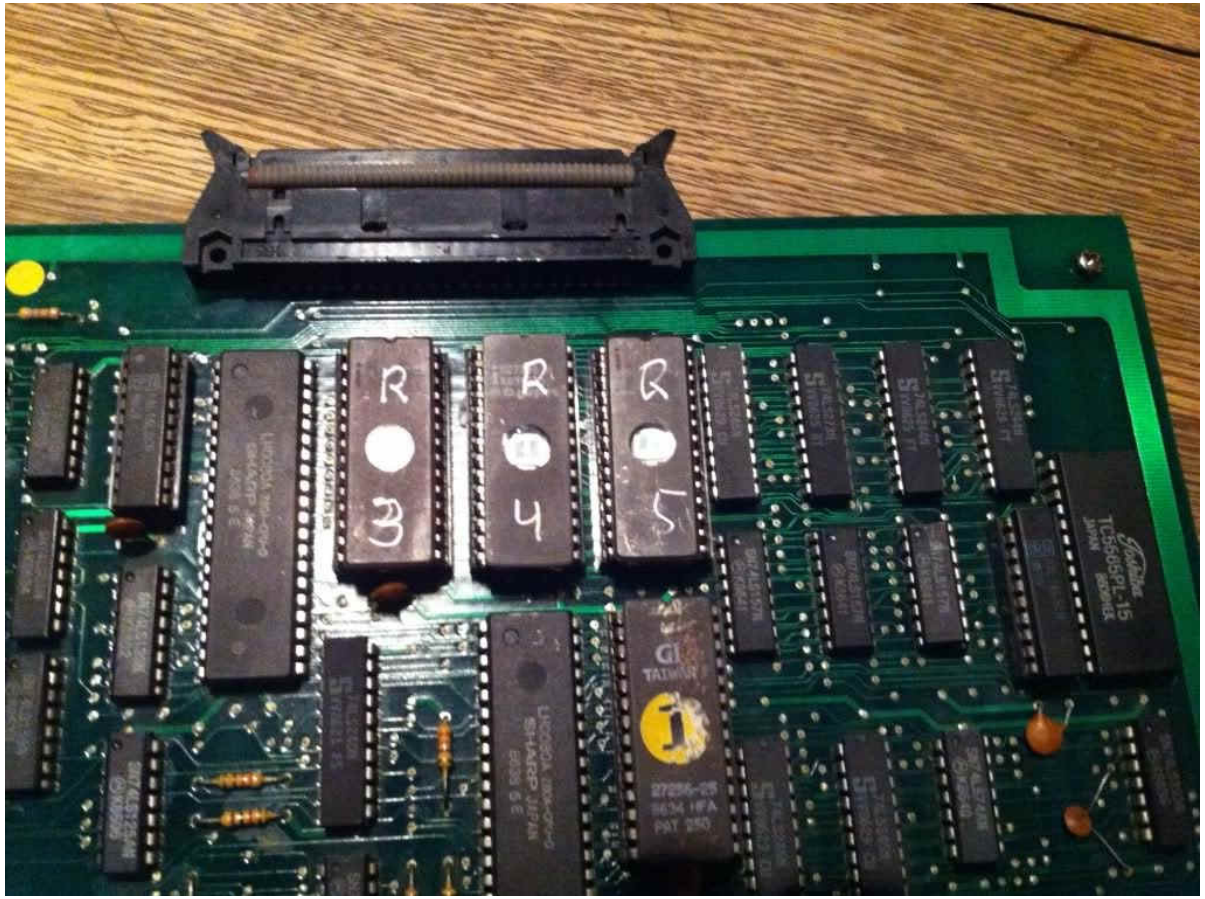
Engenharia reversa ao nível do sistema



Engenharia reversa ao nível do sistema

Breve abordagem histórica

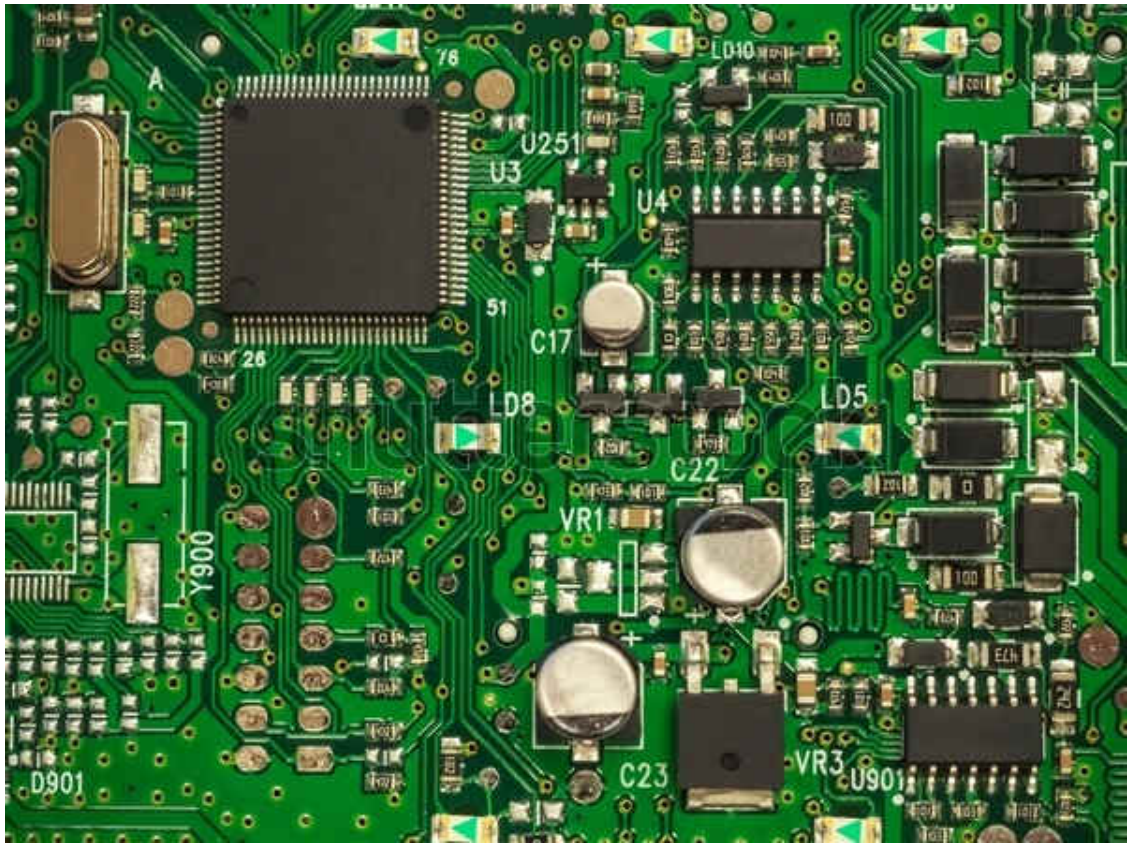
- Há cerca de 30/40 anos:
  - Sistemas digitais (computacionais ou não) baseados com componentes de baixa integração
  - Fáceis de identificar
  - Interligação física dos componentes acessível com ferramentas simples
  - Software permanente em componentes que podiam ser facilmente lidos
  - Reprodução parcial ou total do circuito acessível



Engenharia reversa ao nível do sistema

Breve abordagem histórica

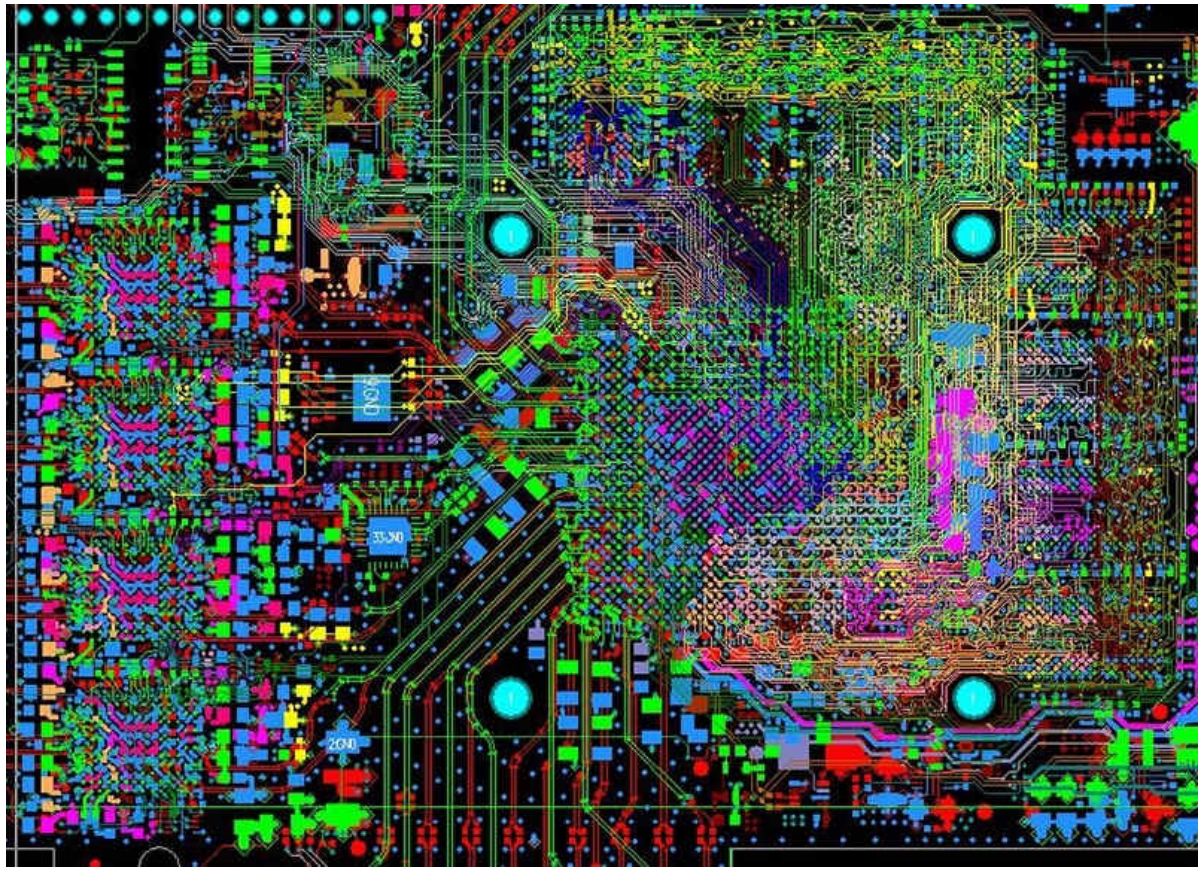
- Com o crescimento do nível de integração:
  - Parte do componentes passíveis de identificação
  - Acesso à ligação exterior dos componentes razoavelmente geral
  - Interligação física entre componentes acessível mas mais complexo com o aumento do número de camadas dos PCBs
  - Aumento do número de protocolos de comunicação (nomeadamente série) permite o *tracing* de interligação entre circuitos.



Engenharia reversa ao nível do sistema

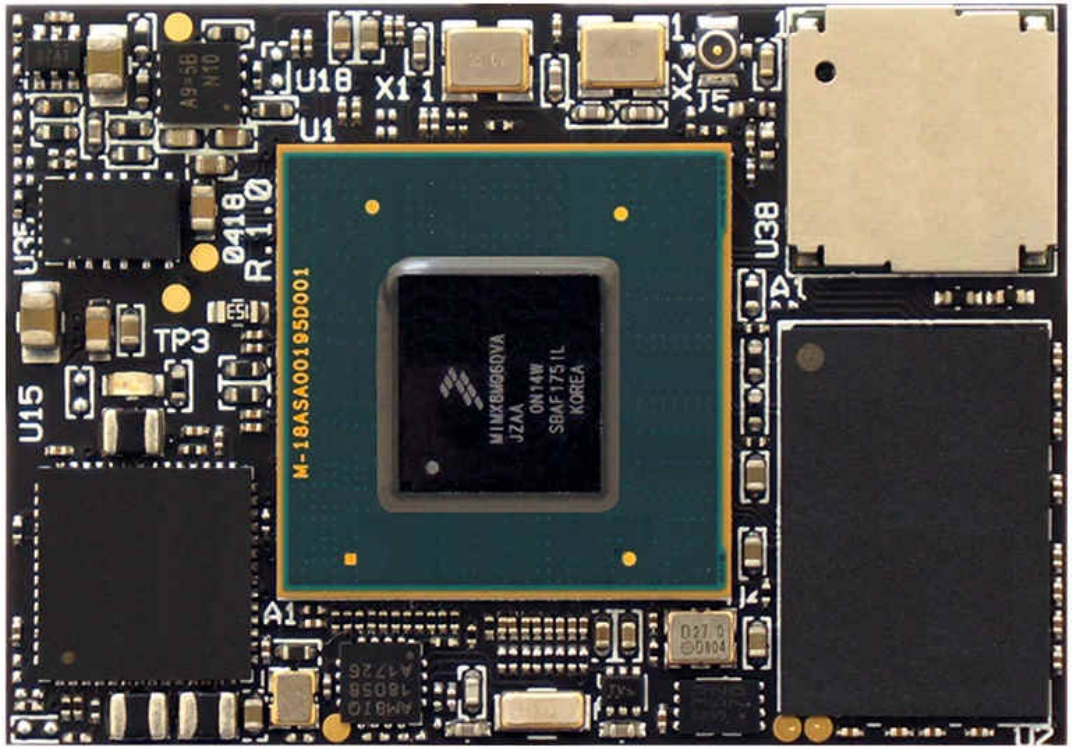
Breve abordagem histórica

- Situação atual:
  - Muito alta densidade dos PCBs (em excesso de 20 camadas)
  - *Route tracing* apenas possível com ferramentas industriais.

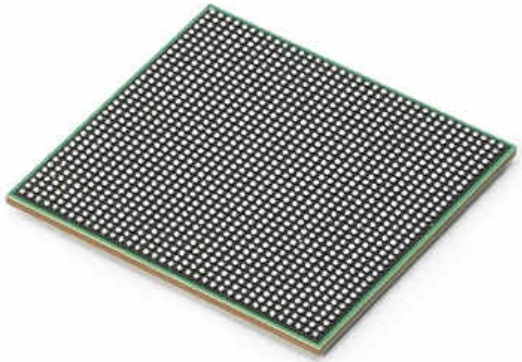


Engenharia reversa ao nível do sistema

Breve abordagem histórica



ASE Group Fan Out Chip on Substrate - BGA
1296 pin

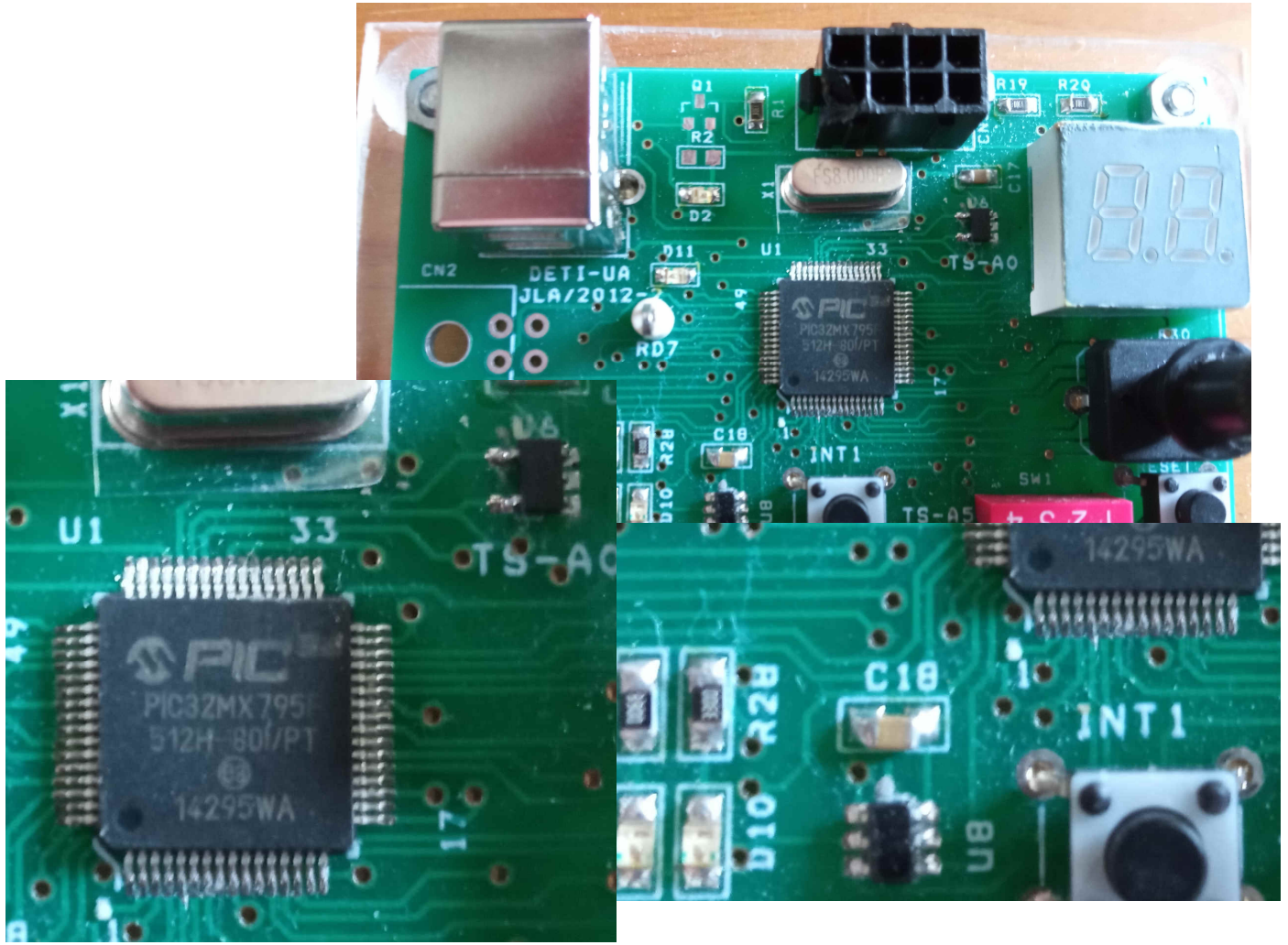


- Situação atual:
  - Circuitos de altíssima densidade (em excesso de 50 Gtransistores
  - Configuração *BGA* (impossível o acesso em circuito)
  - Identificação de componentes nem sempre possível

Engenharia reversa ao nível do sistema

Breve abordagem histórica

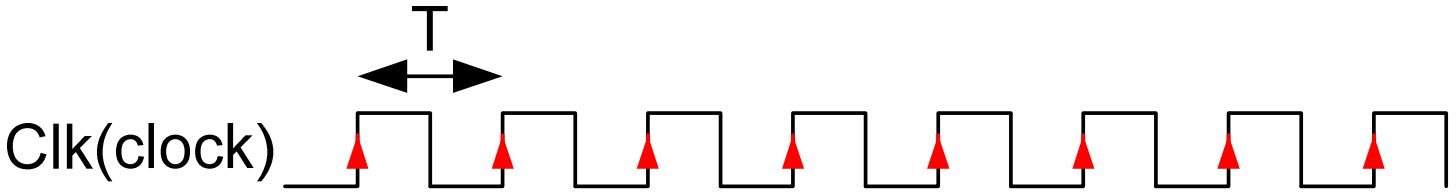
- Mesmo em circuitos de relativamente baixa complexidade:
  - Apenas parte do componentes passíveis de identificação
  - Em circuitos com *pin count* baixo pode tentar fazer-se uma interpretação básica funcional com analisadores lógicos e ou osciloscópios
  - Muitos sistemas são acompanhados de uma descrição mais ou menos detalhada das suas funcionalidades



Engenharia reversa ao nível do sistema

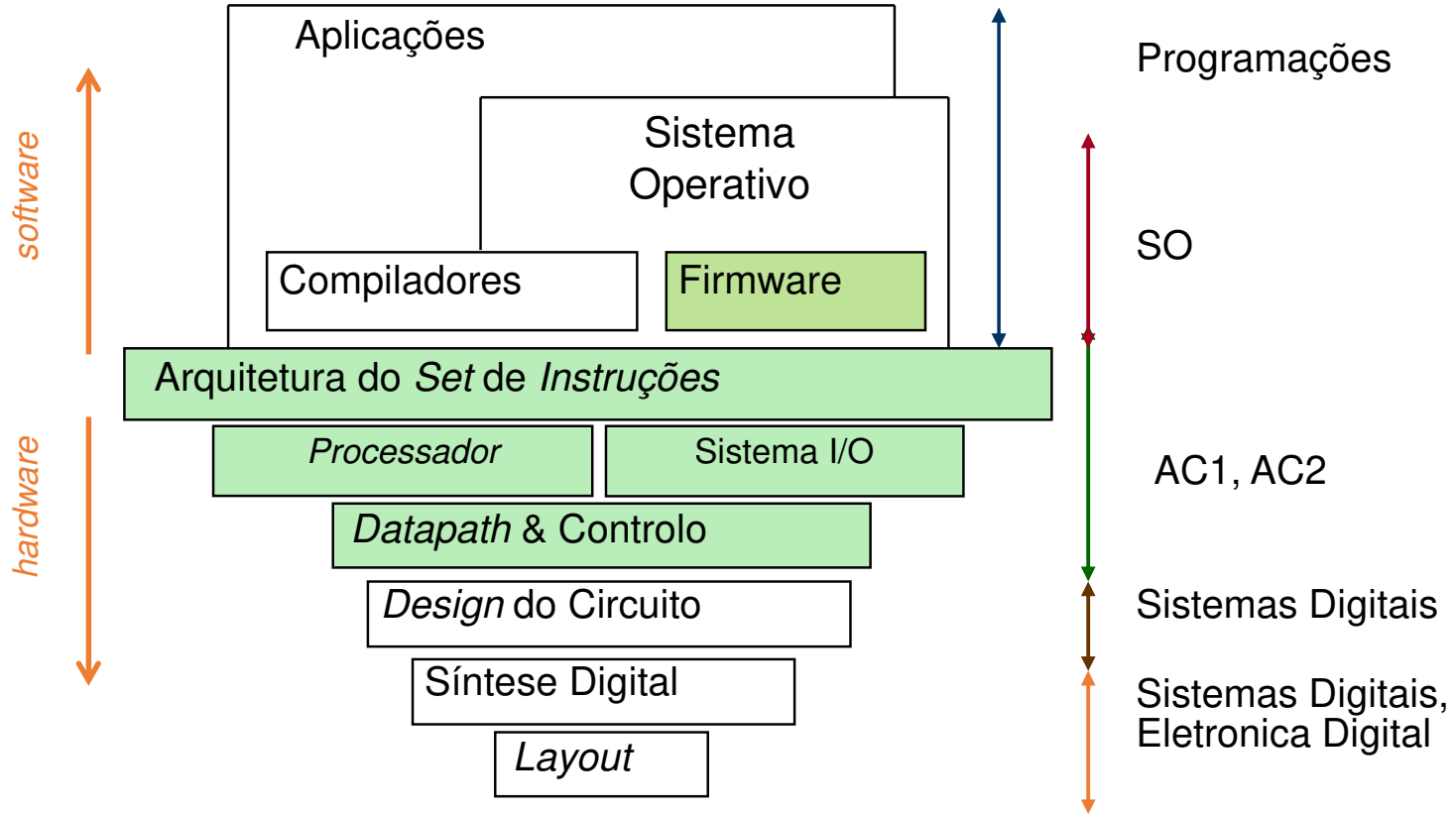
Arquitetura de Computadores

- Princípios básicos de qualquer arquitetura de computadores
  - circuitos digitais binários (alfabeto tem apenas dois símbolos - *bits*)
  - circuitos digitais síncronos (ações desencadeadas pela transição ativa de um sinal de relógio)
  - agrupamento de símbolos formam palavras
  - informação representada por palavras de dimensão limitada
    - nº máximo de palavras (# de combinações possíveis para uma dada dimensão)
  - número limitado de endereços (determinado pelo nº de símbolos usados no endereço)
    - dimensão máxima de memória limitada (ao nível do sistema)
  - número de instruções máquina limitado pelo ISA e pela dimensão das instruções



Engenharia reversa ao nível do sistema

Níveis da Arquitetura de Sistemas Computacionais



Engenharia reversa ao nível do sistema

Representação e codificação de informação digital

- A extração de informação resulta da combinação de uma palavra digital com o conhecimento do código que lhe deu origem
- A mesma informação com diferentes métodos de codificação resultam em palavras diferentes…

10000100 00100101 01000110 01010111 a/ (signed int) = **-2077931945_{10}**

10000100 00100101 01000110 01010111 a/ (ext. 8 bit ASCII) = **"à%FW"**

10000100 00100101 01000110 01010111 a/ (unsigned int) = **$+2217035351_{10}$**

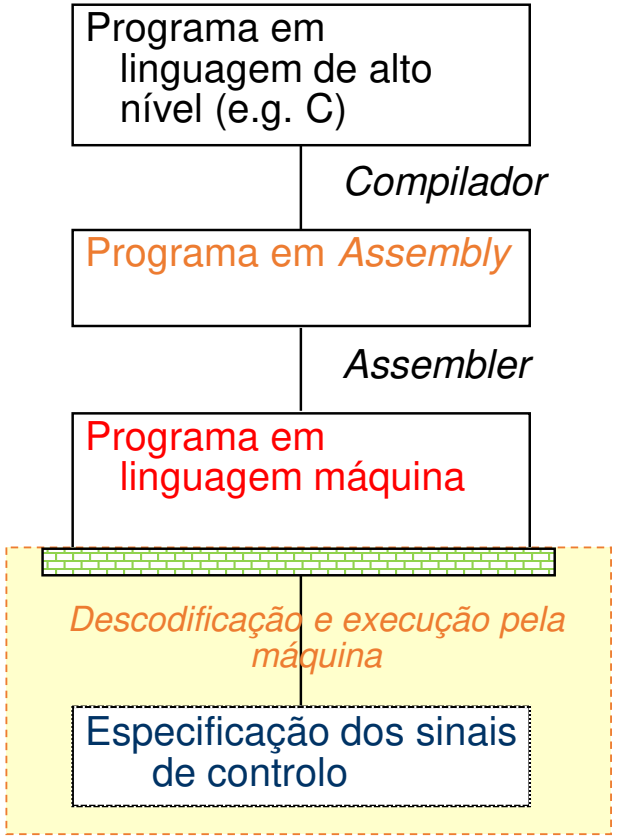
10000100 00100101 01000110 01010111 a/ (float) IEE754 = **$-1.9427955 * 10^{-36}$**

10000100 00100101 01000110 01010111 a/ BCD = **+84254657**

10000100 00100101 01000110 01010111 a/ Gray Code = **+4164519013**

Engenharia reversa ao nível do sistema

Níveis de Representação de informação




```
a = b + c + d;
```

```
add  $10, $11, $12
add  $10, $10, $13
```

```
00000001011101100101000000100000
00000001010011010101000000100000
```

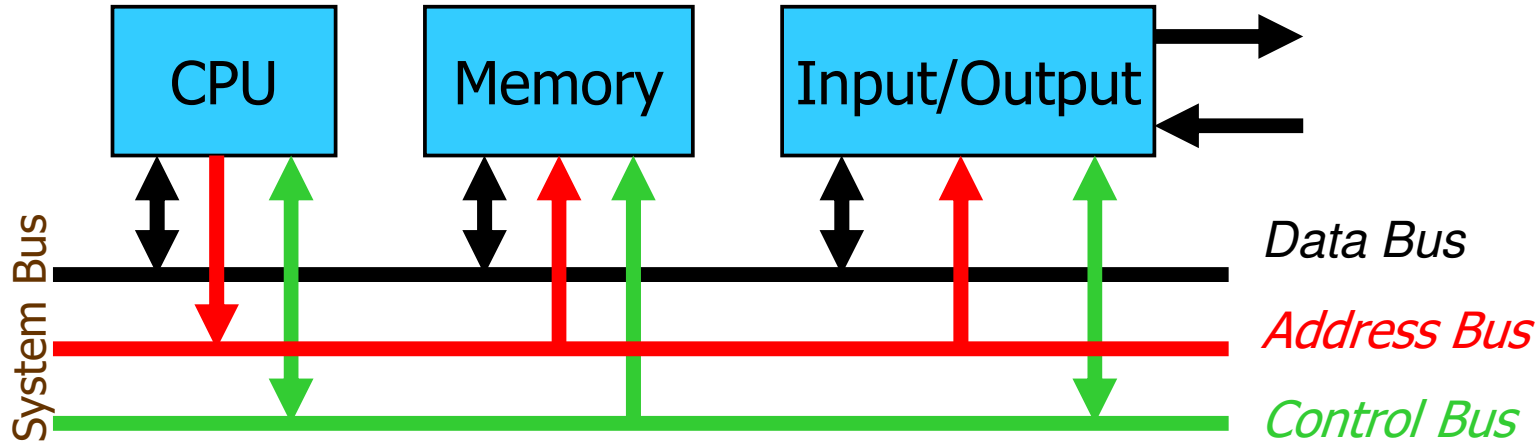
(32 bits)

```
ALUOp <= "10"
ALUIn1 <= Reg[Inst(25:21)]
ALUIn2 <= Reg[Inst(20:16)]
```

Engenharia reversa ao nível do sistema

Modelo básico da arquitetura de um sistema Computacional

- Modelo de von Neumann

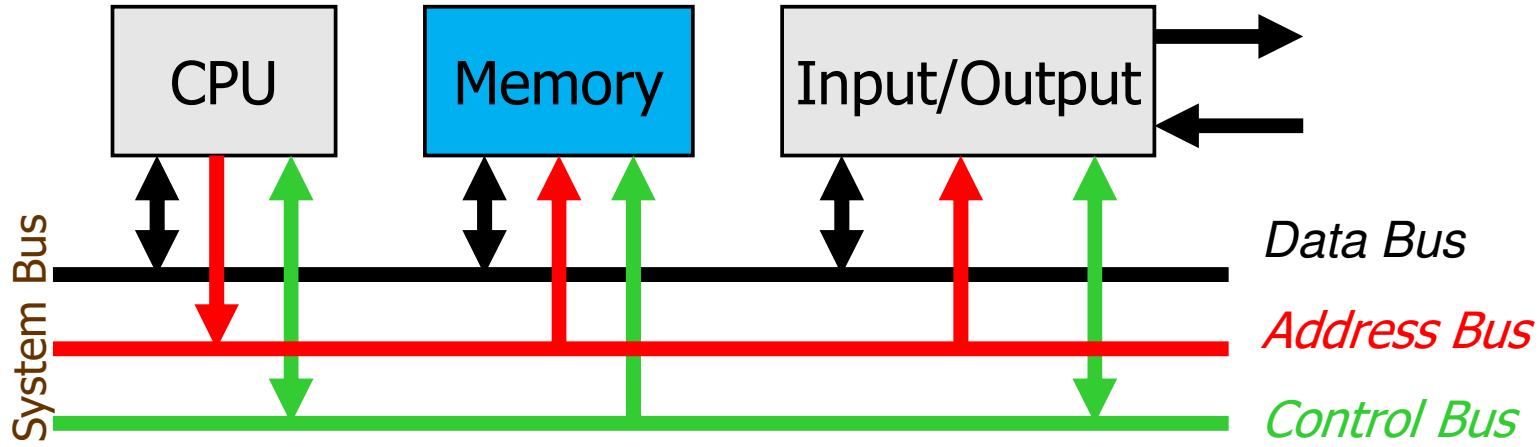


- Data Bus: barramento de transferência de informação (CPU↔memória, CPU↔Input/Output)
- Address Bus: identifica a origem/destino da informação
- Control Bus: sinais de protocolo que especificam o modo como a transferência de informação deve ser feita

Engenharia reversa ao nível do sistema

Modelo básico da arquitetura de um sistema Computacional (Memória)

- Modelo de von Neumann

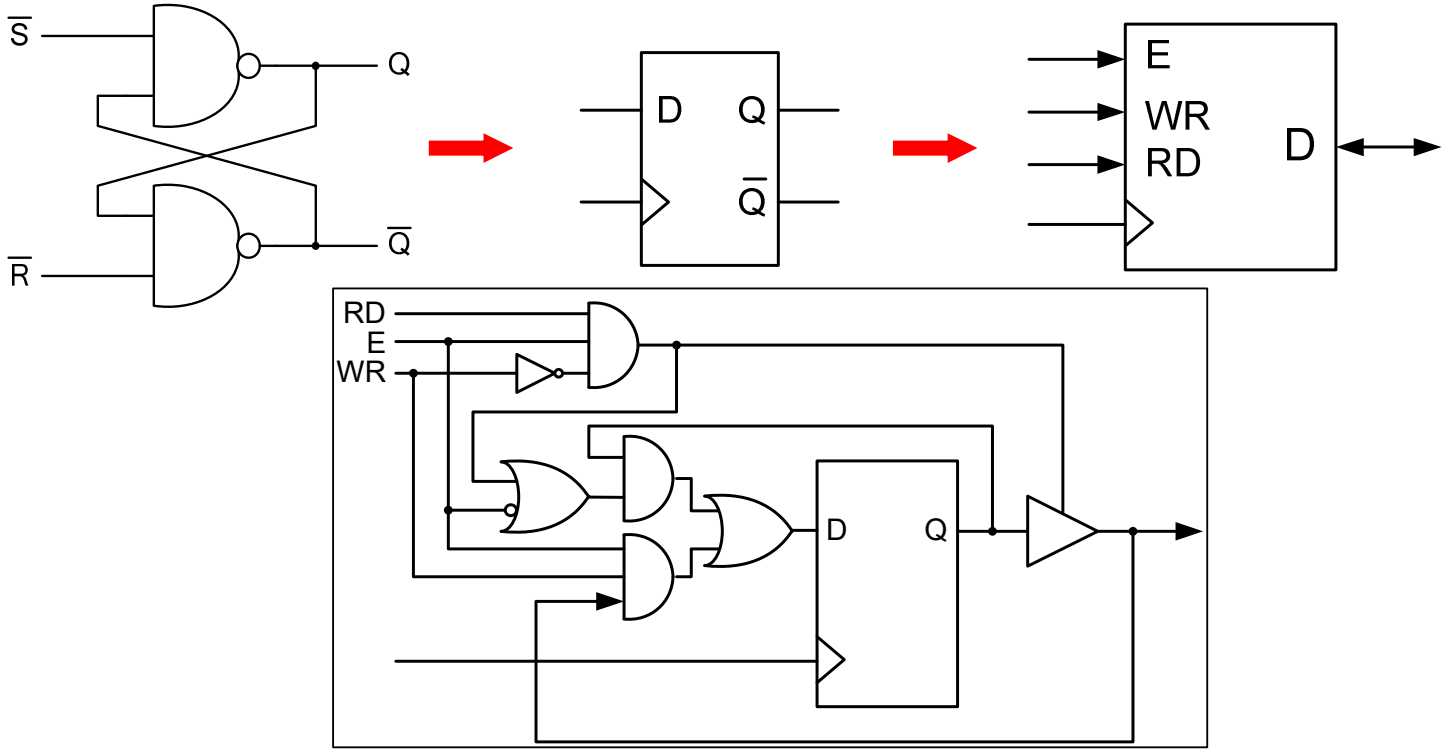


- Data Bus: barramento de transferência de informação (CPU↔memória, CPU↔Input/Output)
- Address Bus: identifica a origem/destino da informação
- Control Bus: sinais de protocolo que especificam o modo como a transferência de informação deve ser feita

Engenharia reversa ao nível do sistema

Memória

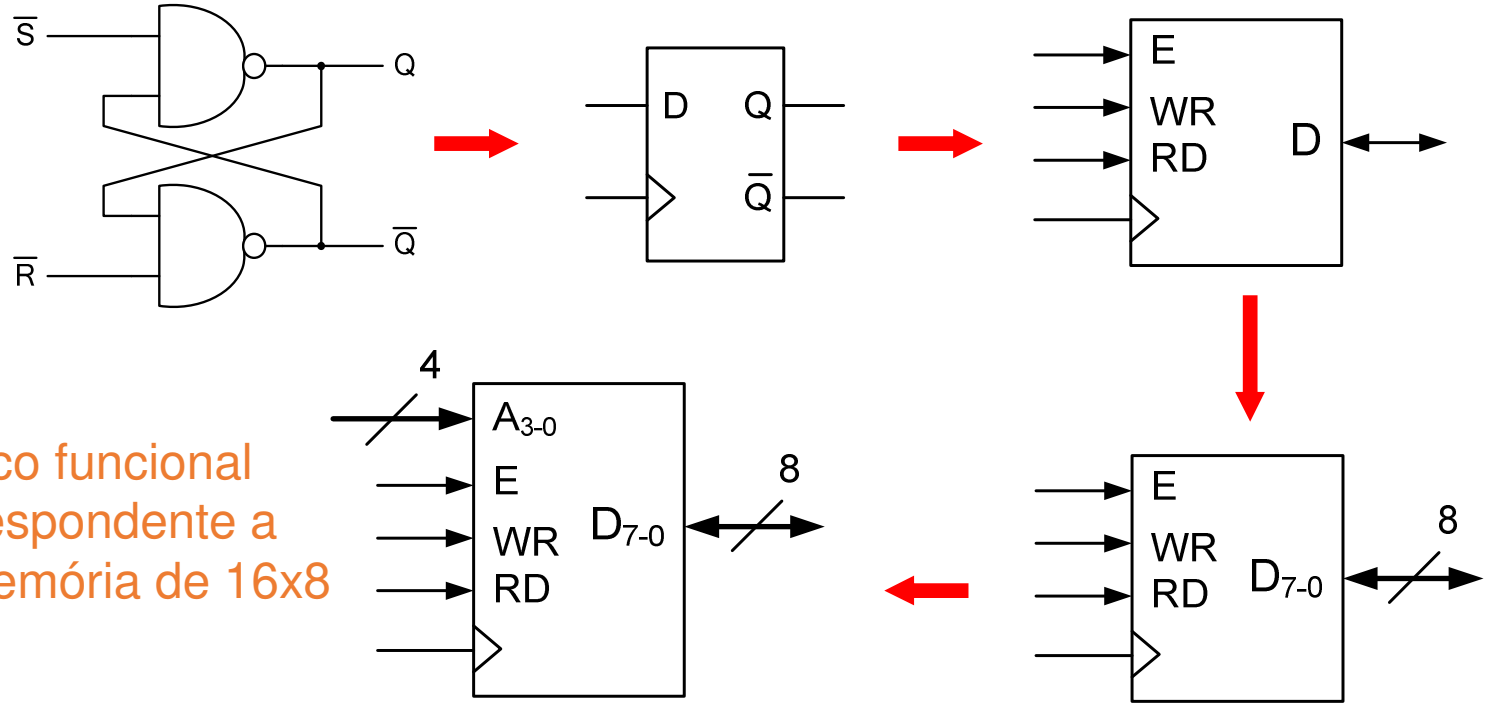
- Uma "Memória“ (um dispositivo com capacidade para armazenar informação digital binária) pode ser construída à custa de blocos básicos bem conhecidos dos sistemas digitais: **flip-flops**



Engenharia reversa ao nível do sistema

Memória

- Exemplo simples de uma memória RAM com uma dimensão de 16x8



Bloco funcional correspondente a uma memória de 16x8

Engenharia reversa ao nível do sistema

Memória (endereçamento)

- **Endereço** (*address*) – um número (único) que identifica univocamente cada registo de memória. Os endereços são contados sequencialmente, começando em 0
  - Exemplo: o conteúdo da posição de memória 0x2000 é 0x32 – (0x2000 é o endereço, 0x32 o valor armazenado)
- **Espaço de endereçamento** (*address space*) – a gama total de endereços que o CPU consegue referenciar (depende da dimensão do barramento de endereços).
  - Exemplo: um CPU com um barramento de endereços de 16 bits pode gerar endereços na gama: 0x0000 a 0xFFFF (i.e., 0 a 2^{16}-1)

Engenharia reversa ao nível do sistema

Memória – tipos básicos

- RAM – Random Access Memory
  - Designação para memória volátil que pode ser lida e escrita
  - Acesso "random“

- ROM – Read Only Memory
  - Memória não volátil que apenas pode ser lida (ou escrita em condições particulares)
  - Acesso "random“

(Acesso "random" - tempo de acesso é o mesmo para qualquer posição de memória)

Engenharia reversa ao nível do sistema

Memória – tipos básicos

- Tecnologias:
  - Semicondutor
  - Magnética
  - Ótica
  - Magneto-ótica

- Memória volátil:
  - Informação armazenada perde-se quando o circuito é desligado da alimentação: RAM (SRAM e DRAM)

- Memória não volátil:
  - A informação armazenada mantém-se até ser deliberadamente alterada: EEPROM, Flash EEPROM, tecnologias magnéticas e óticas

Engenharia reversa ao nível do sistema

Tecnologias de memória não volátil (perspetiva histórica)

- **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

Engenharia reversa ao nível do sistema

Memória – tecnologias de memória volátil

Tecnologia	Tempo Acesso	$ / GB
SRAM	0,5 – 2,5 ns	$500 - $1000
DRAM	35 - 70 ns	$10 - $20
Flash (SSD)	5 – 50 us	$0,75 - $1
Magnetic Disk	5 - 20 ms	$0,005 - $0,1

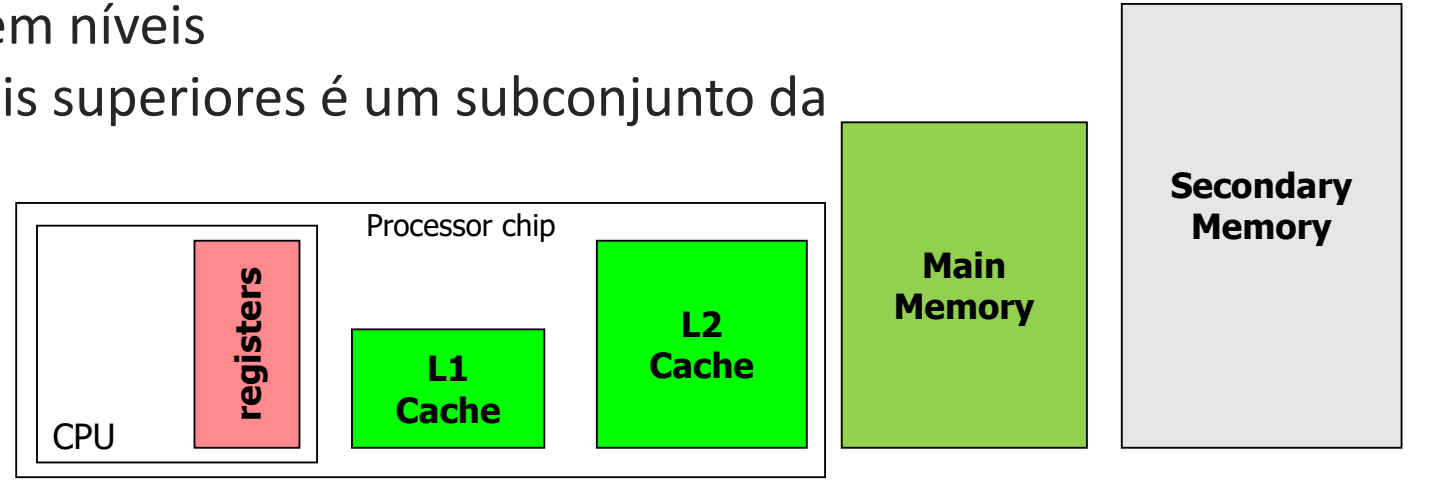
- SRAM - Static Random Access Memory
- DRAM - Dynamic Random Access Memory

- Dadas estas diferenças de custo e de tempo de acesso, é vantajoso construir o sistema de memória como uma hierarquia onde se utilizem todas estas tecnologias

Engenharia reversa ao nível do sistema

Hierarquia de memória

- Memória organizada em níveis
- A informação nos níveis superiores é um subconjunto da dos níveis inferiores

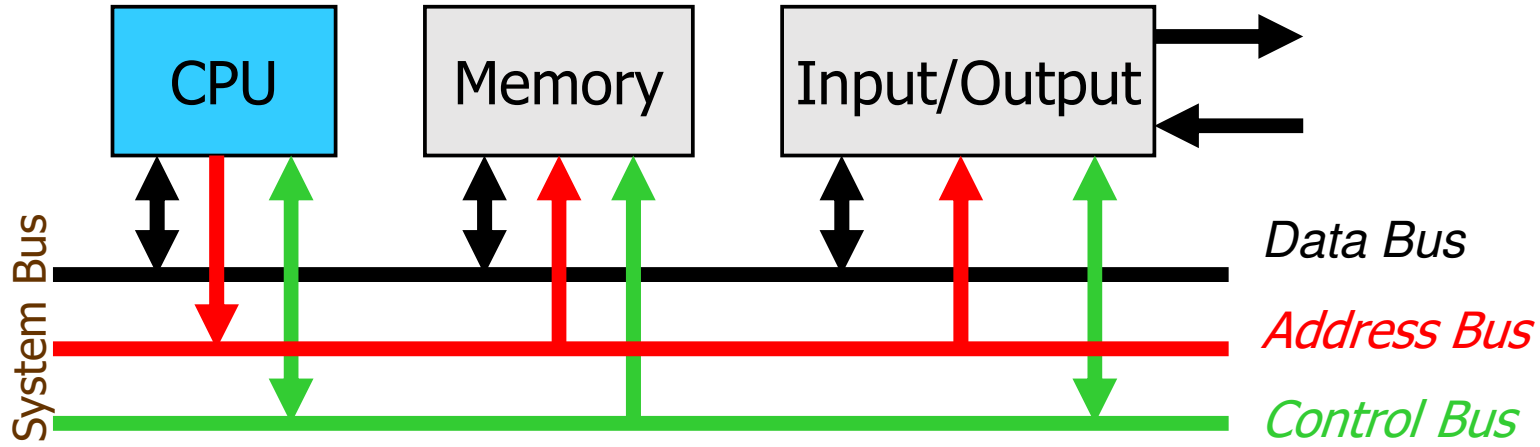


- A informação circula apenas entre níveis adjacentes da hierarquia
- **Bloco** – quantidade de informação que circula entre níveis adjacentes (n bytes)

Engenharia reversa ao nível do sistema

Modelo básico da arquitetura de um sistema Computacional (CPU)

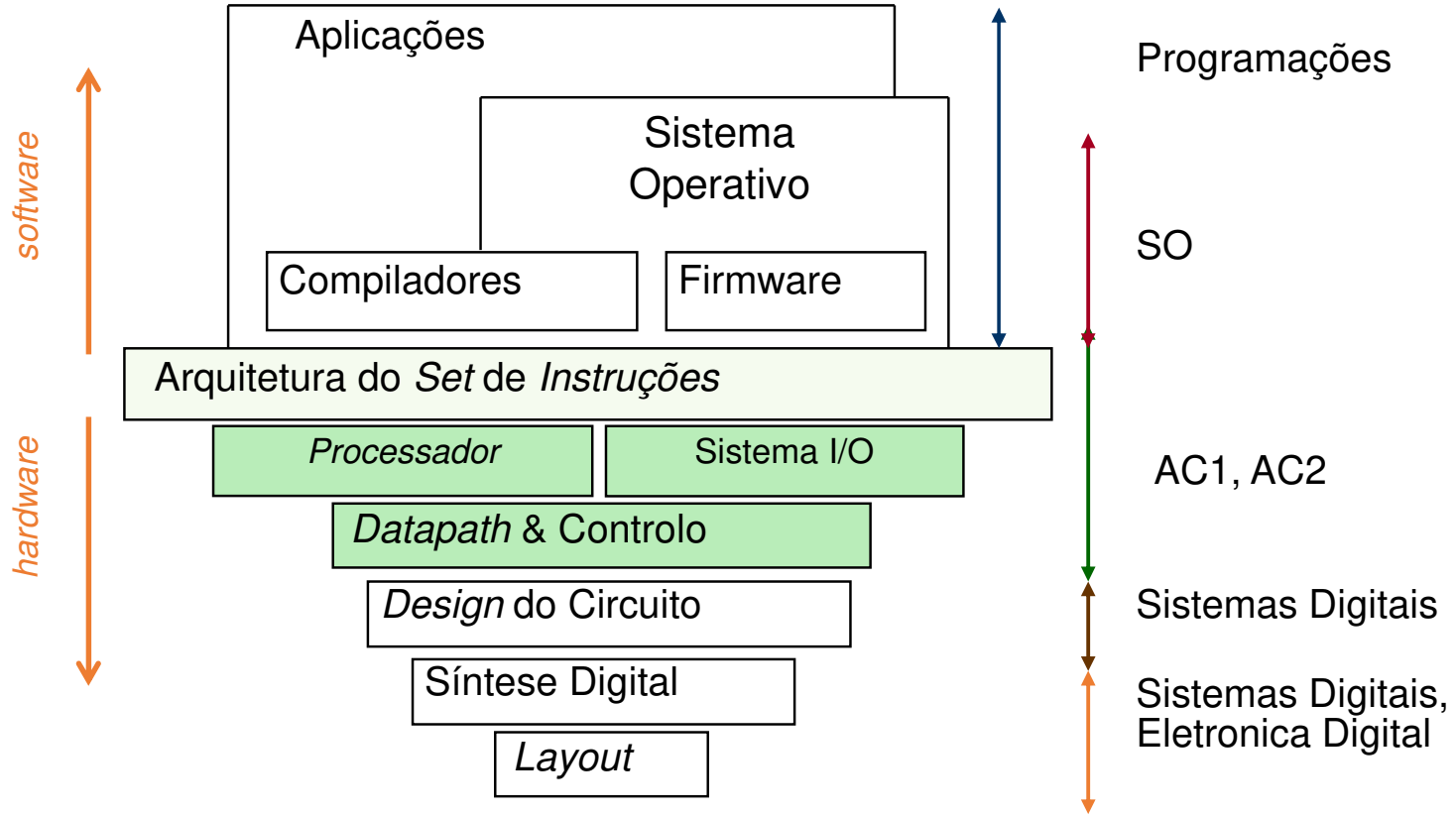
- Modelo de von Neumann



- Data Bus: barramento de transferência de informação (CPU↔memória, CPU↔Input/Output)
- Address Bus: identifica a origem/destino da informação
- Control Bus: sinais de protocolo que especificam o modo como a transferência de informação deve ser feita

Engenharia reversa ao nível do sistema

Níveis da Arquitetura de Sistemas Computacionais



Engenharia reversa ao nível do sistema

ISA

- **ISA**: Instruction Set Architecture
- **Instruction Set**: coleção de todas as operações/instruções que o processador pode executar

- Arquitetura de Computadores =
  Arquitetura do Conjunto de Instruções (ISA) +
  Organização da Máquina

Engenharia reversa ao nível do sistema

Instruções e classes de instruções

- Logo em 1948 a equipa de *Burks, Goldstine and von Neumann* concluíram que, independentemente do número de instruções máquina que um processador possa executar, estas podem ser sempre classificadas em um de três grupos
- Este princípio mantem-se válido até à atualidade

- Classes de instruções:
  - Processamento (aritméticas e lógicas)
  - Transferência de informação
  - Controlo de fluxo de execução

- A terceira classe é crítica na recuperação do modelo estrutural de um programa…

Engenharia reversa ao nível do sistema

ISA – Um interface crítico

- **Formato e codificação das instruções**
  - como são descodificadas?
- **Operandos das instruções e resultados**
  - onde podem residir?
  - quantos operandos explícitos?
  - como localizar?
  - quais podem residir na memória externa?
- **Tipo e dimensão dos dados**
- **Operações**
  - quais são suportadas?
- **Instruções auxiliares**
  - jumps, conditions, branches

Engenharia reversa ao nível do sistema

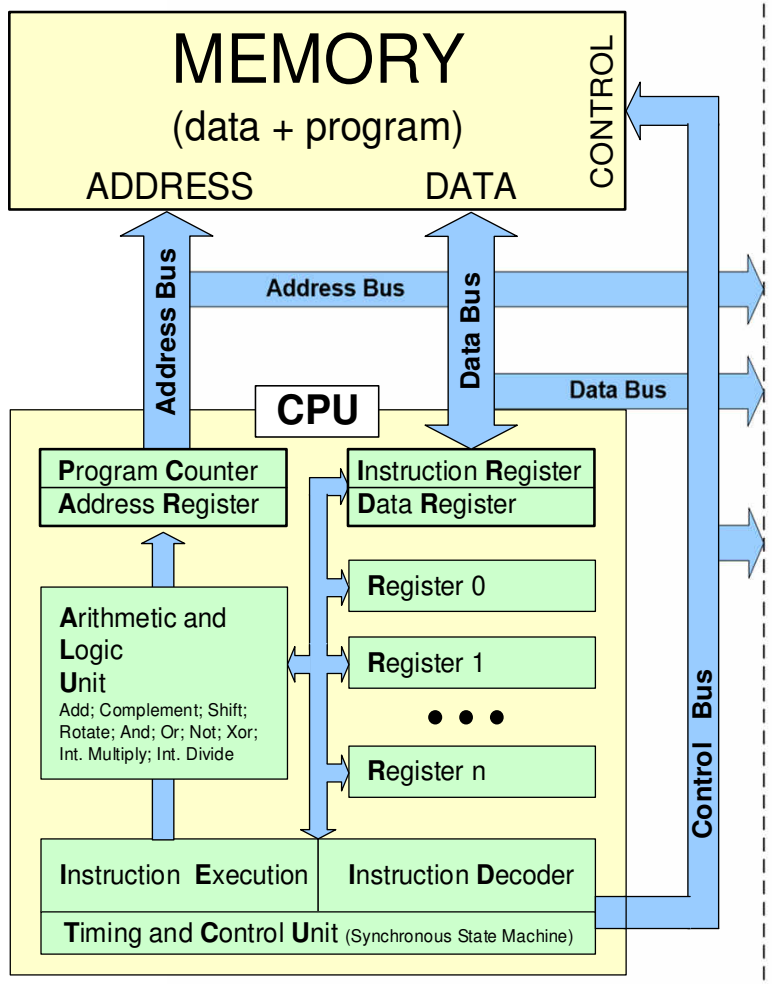
ISA

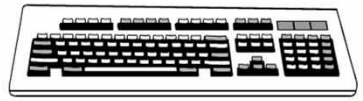
- Também designada por "modelo de programação"
- Uma importante abstração que representa a **interface entre o h/w e o nível mais básico de s/w**
- Descreve tudo o que o programador necessita de saber para programar corretamente, em linguagem máquina, um determinado processador
- Descreve a funcionalidade, independentemente do h/w que a implementa. Pode assim falar-se de "**arquitetura**" e "**implementação de uma arquitetura**"
- Exemplo em que a mesma arquitetura do conjunto de instruções tem 2 implementações distintas:
  - Processadores AMD compatíveis com Intel x86

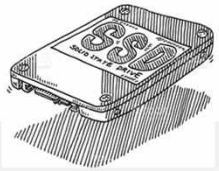
Engenharia reversa ao nível do sistema

Modelo de von Neuman

No Modelo de von Neumann existe apenas uma memória partilhada por instruções e por dados

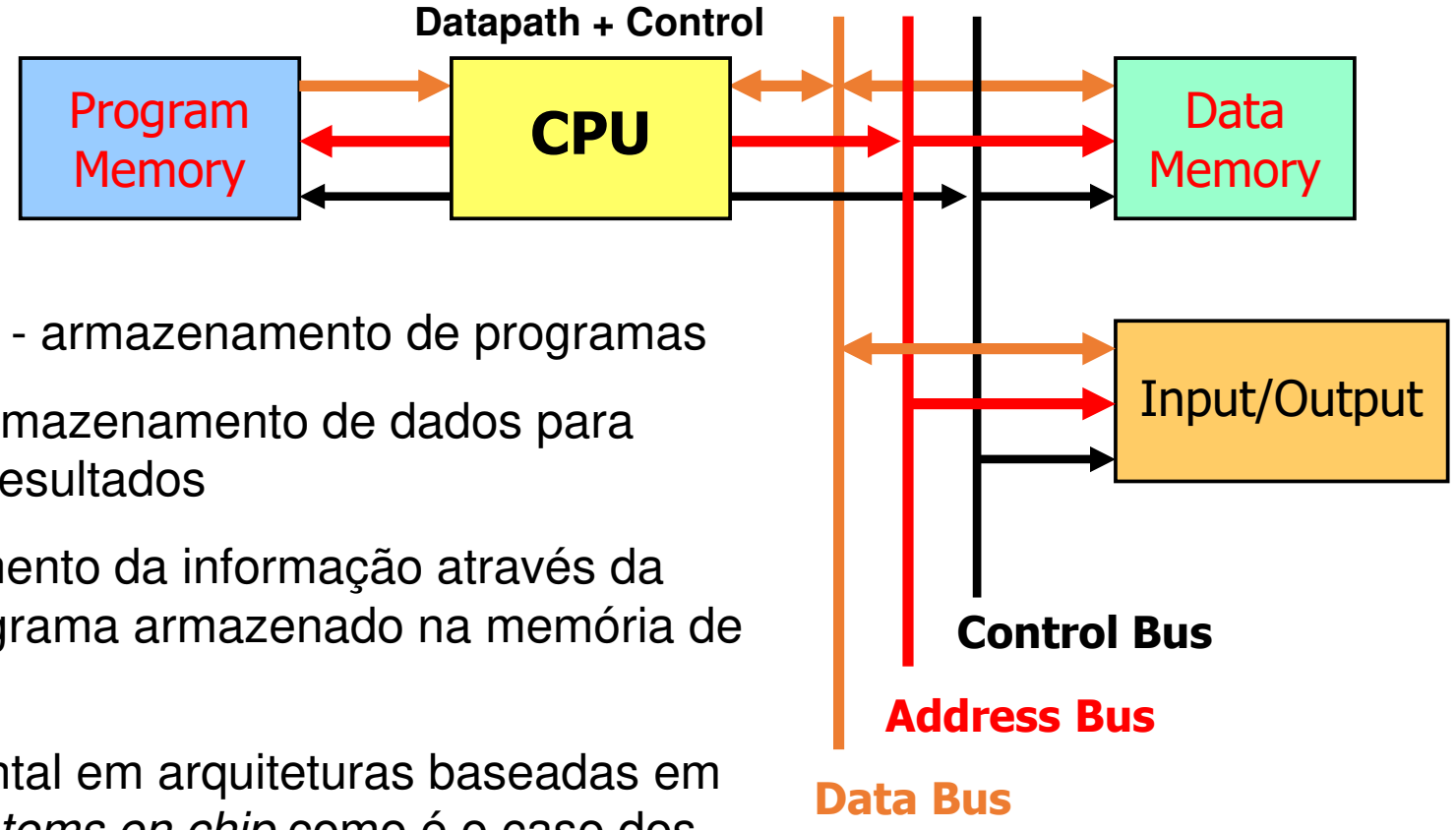






Engenharia reversa ao nível do sistema

Modelo de Harvard

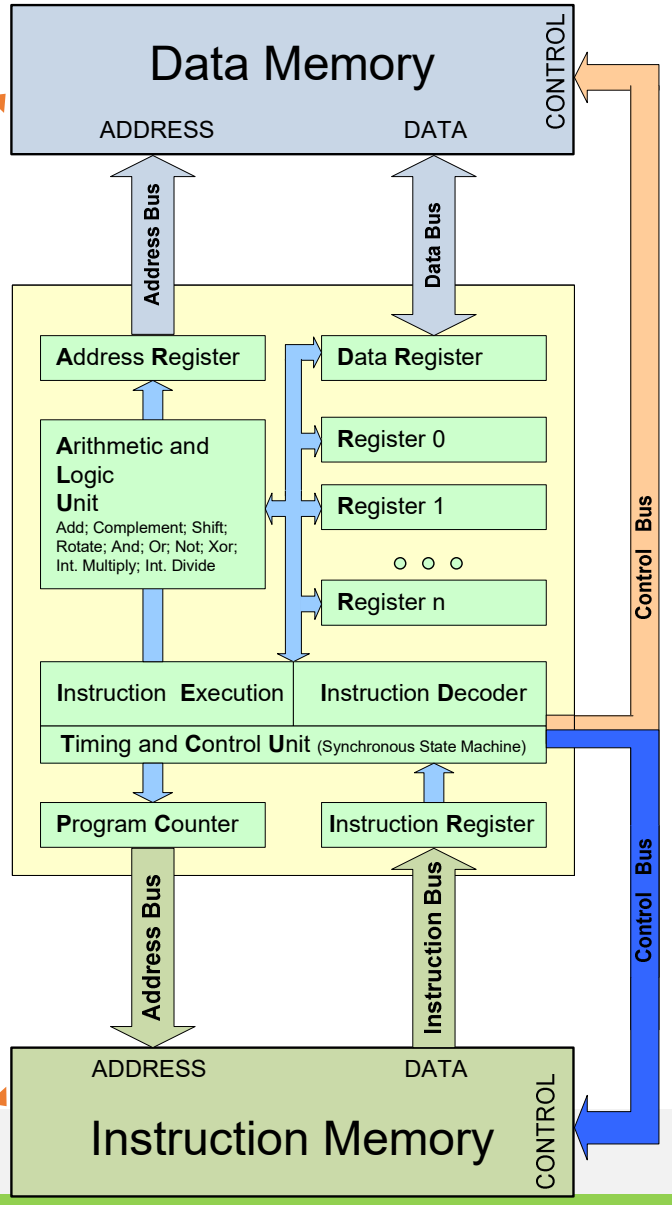


- Program Memory - armazenamento de programas
- Data Memory - armazenamento de dados para processamento, resultados
- CPU - processamento da informação através da execução do programa armazenado na memória de programa
- Modelo fundamental em arquiteturas baseadas em pipeline e em *systems on chip* como é o caso dos micro-controladores

Engenharia reversa ao nível do sistema

Modelo de Harvard

No Modelo de Harvard existem duas memórias independentes (uma para instruções e outra para dados) que podem ser acedidas simultaneamente pelo CPU



Engenharia reversa ao nível do sistema

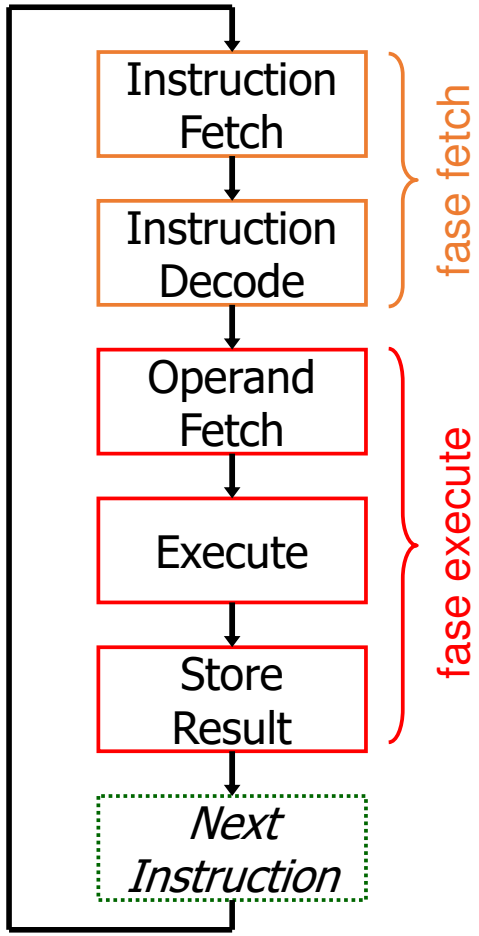
Arquitetura básica do CPU

- **Secção de dados** (*datapath*) – elementos operativos/funcionais para encaminhamento, processamento e armazenamento de informação
  - Multiplexers
  - Unidade Aritmética e Lógica (ALU) – Add, Sub, And, Or...
  - Registos internos
- **Unidade de controlo** – responsável pela coordenação dos elementos do *datapath*, durante a execução de um programa
  - Gera os sinais de controlo que adequam a operação de cada um dos recursos da secção de dados às necessidades da instrução que estiver a ser executada
  - Dependendo da arquitetura, pode ser uma máquina de estados ou um elemento meramente combinatório
- Independentemente da Unidade de Controlo ser combinatória ou sequencial, **o CPU é sempre uma máquina de estados síncrona**

Engenharia reversa ao nível do sistema

Ciclo-base de execução de uma instrução

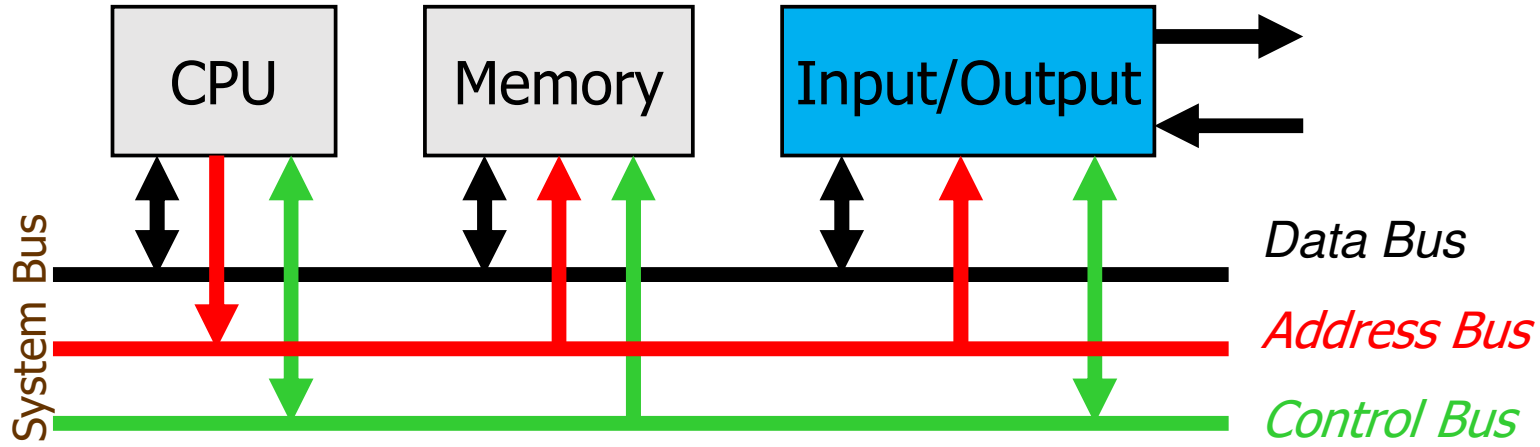
- **Instruction fetch:** leitura do código máquina da instrução (instrução reside em memória)
- **Instruction decode:** descodificação da instrução pela unidade de controlo
- **Operand fetch:** leitura do(s) operando(s)
- **Execute:** execução da operação especificada pela instrução
- **Store result:** armazenamento do resultado da operação no destino especificado na instrução



Engenharia reversa ao nível do sistema

Modelo básico da arquitetura de um sistema Computacional (I/O)

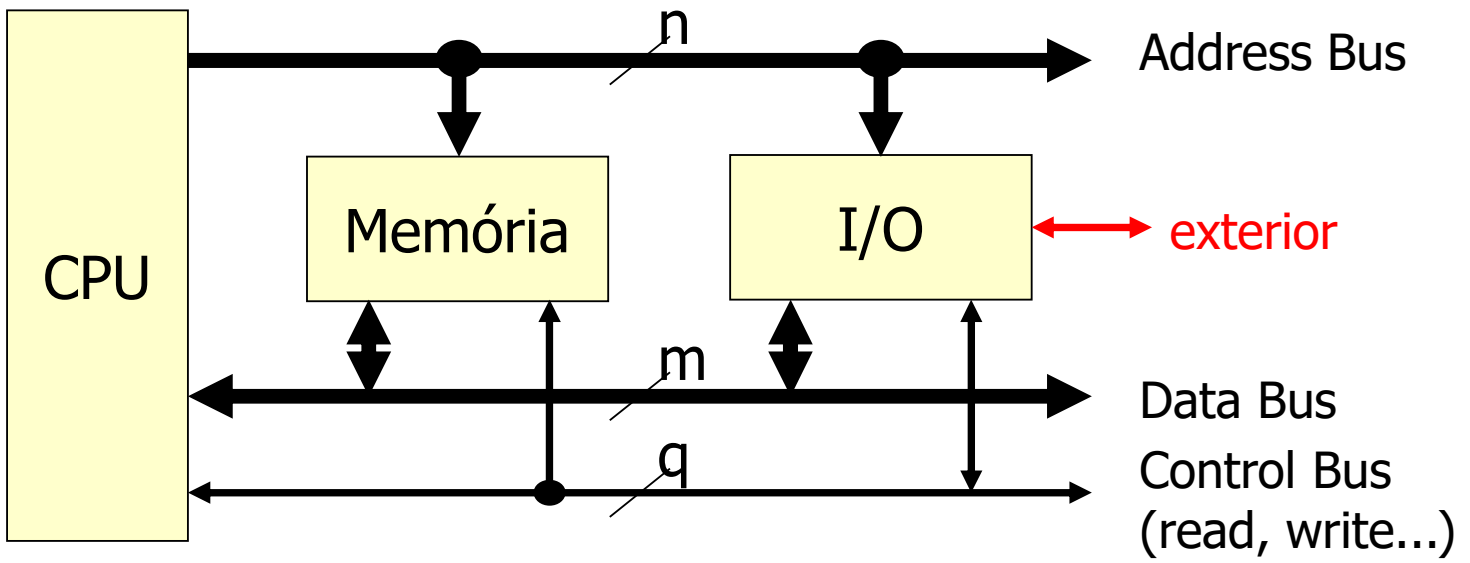
- Modelo de von Neumann



- Data Bus: barramento de transferência de informação (CPU↔memória, CPU↔Input/Output)
- Address Bus: identifica a origem/destino da informação
- Control Bus: sinais de protocolo que especificam o modo como a transferência de informação deve ser feita

Engenharia reversa ao nível do sistema

Dispositivos periféricos - introdução



- 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

Engenharia reversa ao nível do sistema

Dispositivos periféricos - introdução

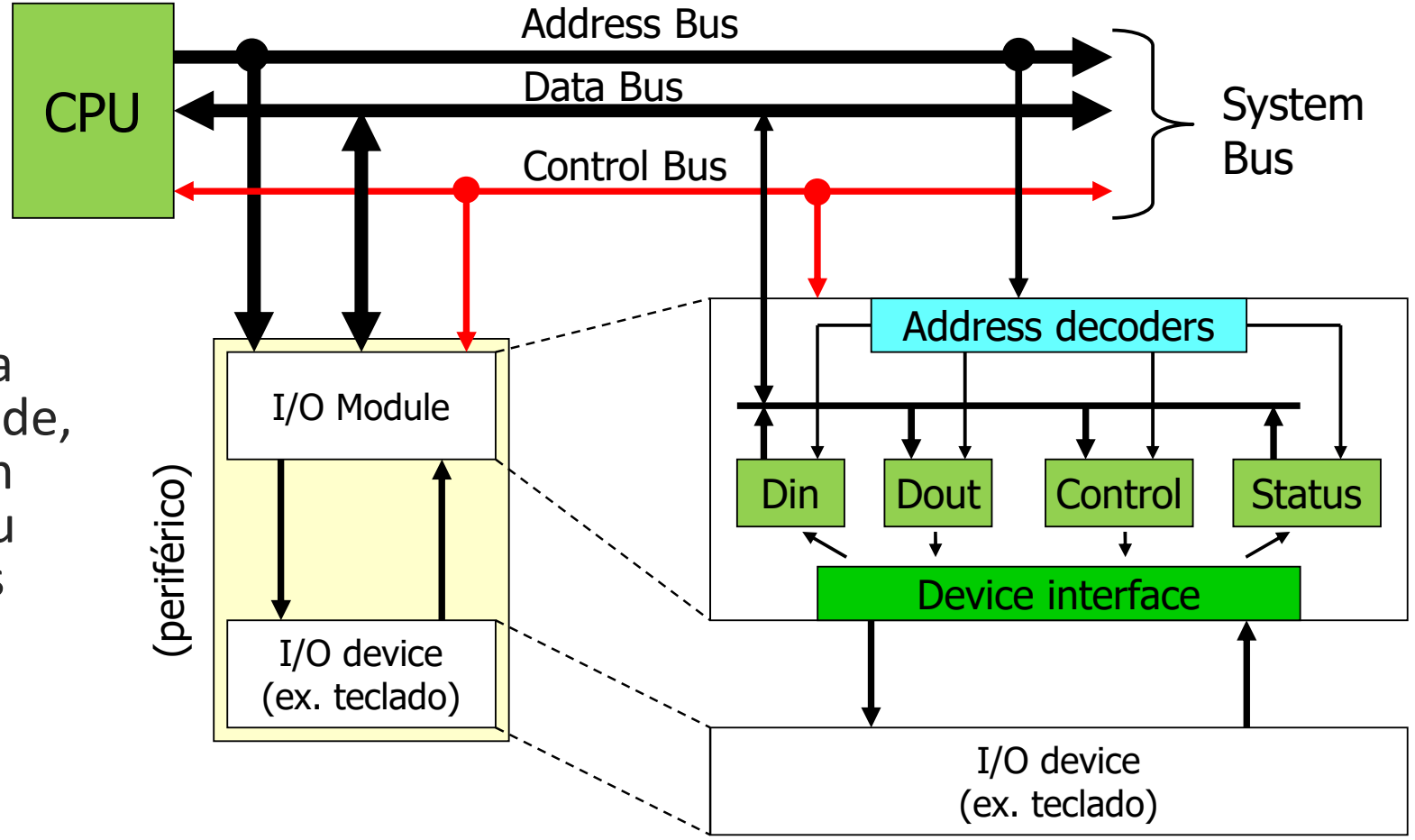
- Dispositivos periféricos:
  - grande variedade [centenas] (por exemplo: teclado, rato, unidade de disco ...)
  - com métodos de operação diversos
  - assíncronos relativamente ao CPU
  - geram diferentes quantidades de informação com diferentes formatos a diferentes velocidades (de alguns bits/s a centenas de Megabyte/s)
  - É muito comum serem mais lentos que o CPU e a memória

- É necessária uma interface que providencie a adaptação entre as características intrínsecas do dispositivo periférico e as do CPU / memória

Engenharia reversa ao nível do sistema

Dispositivos periféricos - interface

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



Engenharia reversa ao nível do sistema

Módulo de I/O

- O módulo de I/O pode ser visto como um módulo de compatibilização entre as características e modo de funcionamento do sistema computacional e o 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. O periférico liga-se ao sistema através dos barramentos, do mesmo modo que todos os outros dispositivos (ex. memória)
- Interação com o dispositivo físico:
  - Lida com as particularidades do dispositivo, nomeadamente, 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.

Engenharia reversa ao nível do sistema

Módulo de I/O

- O **módulo de I/O** é composto por um conjunto de registos. A descrição de cada um deles é específico para cada periférico. Estes registos constituem o que se designa por **modelo de programação do periférico**.

- **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*)

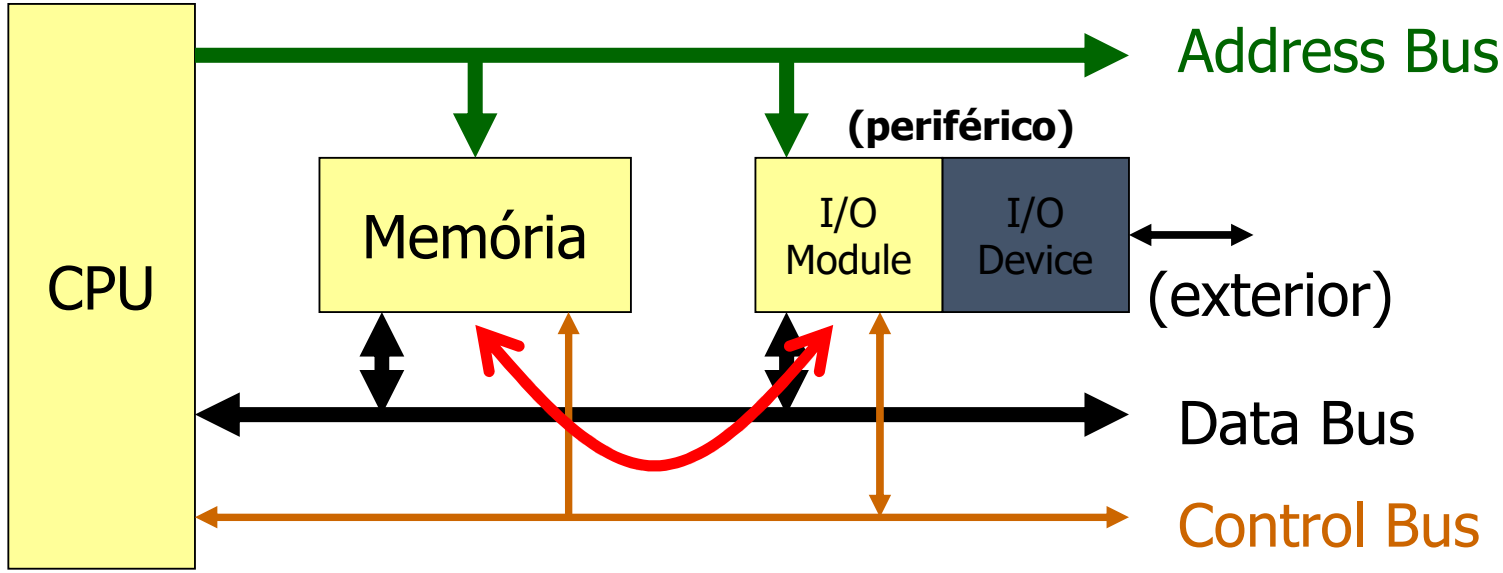
- **Status Register**(s) (*Read only*)
  - Registo(s) que engloba(m) um conjunto de bits que dão informação sobre o estado do periférico (ex. operação terminada, informação disponível, situação de erro, ...)

- **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*)

Engenharia reversa ao nível do sistema

Periféricos – transferência de informação



- CPU – processamento
- Memória – armazenamento (instruções e dados)
- Unidades de I/O – comunicação com dispositivos externos
- Que métodos podem ser usados para transferir informação entre os periféricos e a memória?

Engenharia reversa ao nível do sistema

Periféricos – técnicas de transferência de informação

1. O CPU inicia e controla a transferência de informação:
   - E/S programada (***programmed I/O***)
     - O CPU toma a iniciativa – aguarda se necessário, inicia e controla a transferência de informação (**POLLING**)
   - E/S por interrupção (***interrupt driven I/O***)
     - O periférico sinaliza o CPU de que está pronto para trocar informação (leitura ou escrita). O CPU inicia e controla a transferência

2. O CPU não toma parte na transferência de informação:
   - E/S por acesso direto à memória (**DMA**)
     - Um dispositivo hardware externo ao CPU (DMA) faz a transferência de informação diretamente entre a memória e o periférico; o CPU não toma parte no processo de transferência
     - O CPU apenas configura inicialmente o periférico e o DMA; no final o DMA sinaliza o CPU que a transferência terminou

Engenharia reversa ao nível do sistema

Transferência de informação por interrupção

- Na técnica de E/S por interrupção quando o periférico está pronto para disponibilizar/receber informação informa o CPU desse facto
- Uma interrupção, depois de reconhecida, faz com que o CPU abandone temporariamente a execução do programa em curso para executar a rotina que dá seguimento à interrupção gerada
  - A rotina associada à interrupção designa-se por **rotina de serviço à interrupção** ou ***interrupt handler***

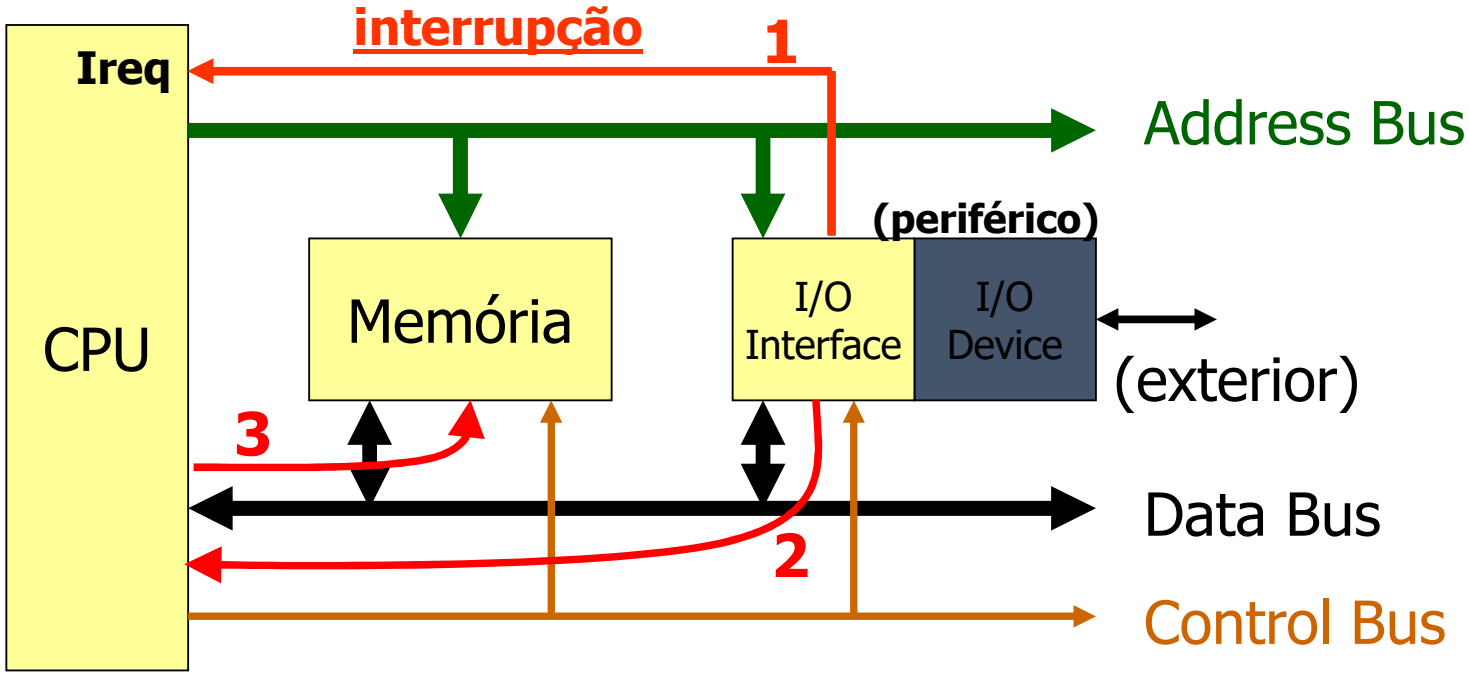
Exemplo: leitura de dados de um periférico

- CPU envia pedido de informação ao periférico (escrita num registo de controlo do periférico)
- CPU continua a execução do programa (com outras tarefas)
- Quando tiver informação disponível, o periférico gera um pedido de interrupção ao CPU

- CPU atende a interrupção:
  - Suspende a execução do programa corrente
  - Salta para a **rotina de atendimento à interrupção** (*interrupt handler*) que transfere a informação
  - Retoma a execução do programa suspenso

Engenharia reversa ao nível do sistema

Transferência de informação por interrupção



- O periférico envia pedido de interrupção ao CPU quando tiver informação disponível

Engenharia reversa ao nível do sistema

Sistemas embebidos (*embedded systems*)

Engenharia reversa ao nível do sistema

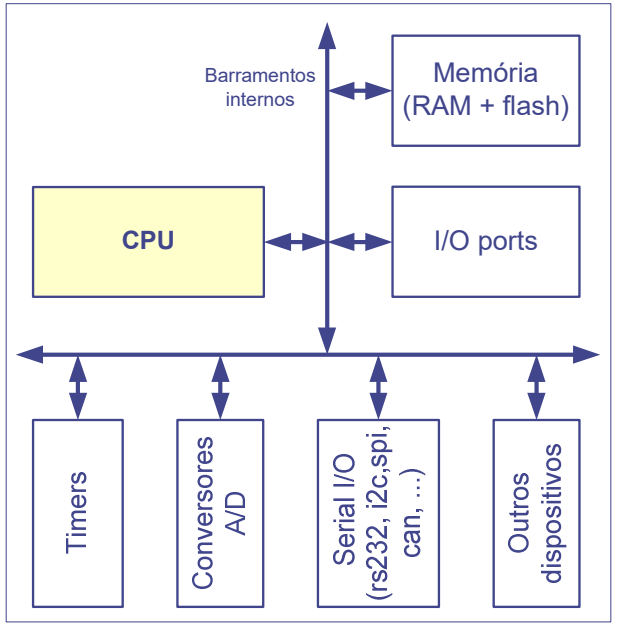
Sistemas embebidos (*embedded systems*)

- 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, são mais (ou muito 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 com frequência 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, …

Engenharia reversa ao nível do sistema

Microcontrolador – principais caraterísticas

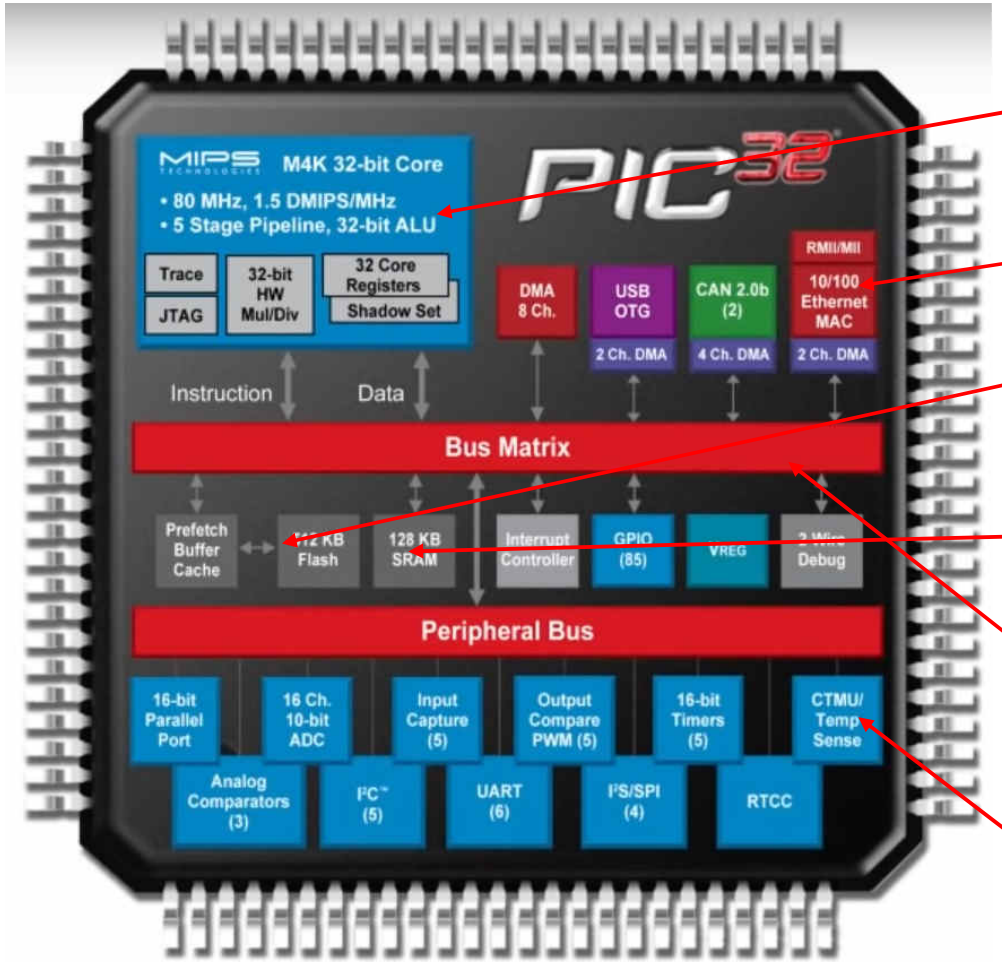
- Dispositivo computacional programável que integra, num único circuito integrado, 3 componentes fundamentais:
  - Uma Unidade de Processamento
  - Memória (volátil e não volátil)
  - Portos de I/O (E/S)
- Inclui outros dispositivos de suporte (periféricos), tais como:
  - Timers
  - Conversor A/D
  - 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).

Engenharia reversa ao nível do sistema

Exemplo de micro-controlador



Microprocessador MIPS (SYSCLK=80 MHz)

Periféricos com DMA

Memória de Instruções (128-bit wide)

Memória de Dados

Bus Matrix

Periféricos (PBCLK)

Modelo de Harvard

Engenharia reversa ao nível do sistema

Famílias de micro-controladores

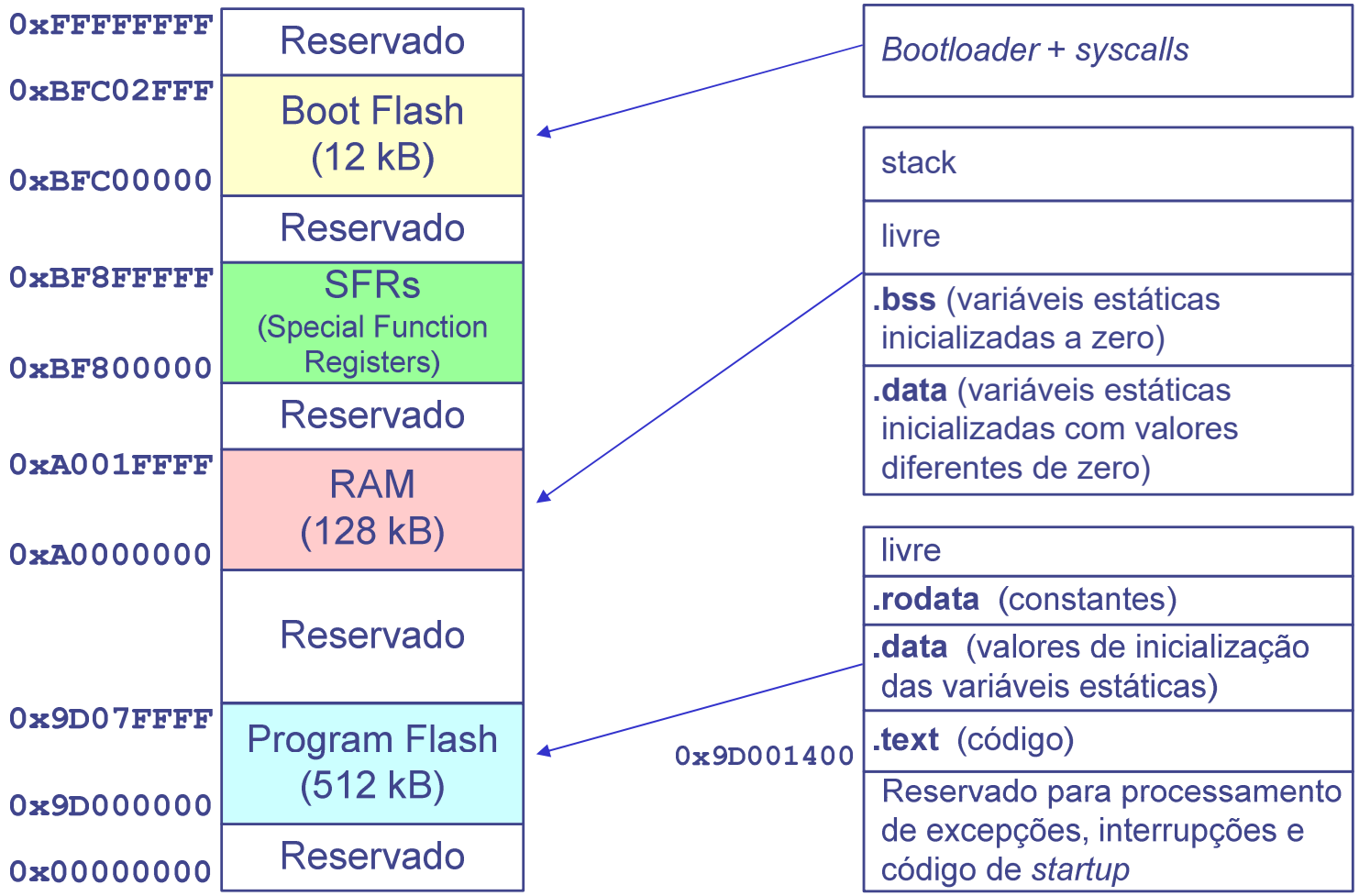
	PIC32MZ2048EFH144	PIC32MX110F016B
Core	252MHz 32 bit M-Class (144 pin) 16 KB I-Cache, 4 KB D-Cache	40MHz 32 bit MIPS (28 pin)
Mem	Flash: 2048 kB + RAM: 512 kB	Flash: 16 kB + RAM: 4 kB
FPU	32/64-bit IEEE 754-compliant	---
ADC	12 bit – 48 analog inputs	10 bit – 10 analog inputs
Timers	9 x 16 bit or 4 x 32bit	5 x 16 bit or 2 x 32bit
UART	6 @ 25Mbps	2 @ 12.5Mbps
SPI	6 4-wire (up to 50 MHz)	2 4-wire (up to 50 MHz)
I2C	5 modules (up to 1 Mbaud)	2 modules (up to 1 Mbaud)
USB	2.0 Hi-Speed (dedicated DMA)	---
Eth	10/100 Mbps Ethernet MAC	---
CAN	2 modules (dedicated DMA)	---
DMA	8 channels	4 channels (GP)

~11€

~2€

Engenharia reversa ao nível do sistema

Mapa de memória do **PIC32MX795F512H**



Engenharia reversa ao nível do sistema

Programação e acesso à informação

- Para a transferência de um programa executável para a memória do microcontrolador pode utilizar-se 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 (Microchip ICSP).
  - O uso de JTAG e ou interfaces proprietárias pode permitir igualmente a leitura (dump) do espaço de memória do micro-controlador.