## Operação em Tempo Real

- Devem finalizar operações com <u>restrições temporais</u>
  - *Hard real time:* o não cumprimento das restrições causa a falha da ação
  - *Soft real time:* o não cumprimento das restrições causa redução da performance

15

15

## Requisitos Não-Funcionais

- Muitos SE são itens de mercado de massa que devem ter baixo custo de fabrico.
  - Memória limitada, baixa freq. processador…
- O consumo de energia é crítico em dispositivos alimentados por bateria.
  - O consumo excessivo de energia aumenta o custo do sistema (melhores sistemas de alimentação).

16

16

## Equipas de projeto

- Normalmente desenvolvidos por equipas de poucos elementos
- Habitualmente devem cumprir prazos apertados
  - Janela de mercado de 6 meses é comum
  - Pouco espaço para falhas

17

17

## Porque utilizar Microprocessadores?

- Alternativas: *field-programmable gate arrays* (*FPGAs*), lógica customizada, etc.
- Microprocessadores são muito eficientes: podem utilizar a mesma lógica para realizar diferentes funções
- Microprocessadores simplificam o desenvolvimento de famílias de produtos

18

18

# O Paradoxo da Performance

- Microprocessadores utilizam muito mais lógica para implementar uma função do que a lógica customizada
- Contudo os microprocessadores são tipicamente mais rápidos:
  - Fortemente *pipelined*;
  - Concebidos por grandes equipas de desenvolvimento;
  - Tecnologia *VLSI* agressiva.

*VLSI = Very-Large-Scale Integration*



19

# Energia

- A lógica customizada requer menos energia, mas os *CPUs* possuem vantagens:
  - Microprocessadores modernos possuem caraterísticas que permitem controlar o consumo de energia
  - Técnicas de design de software podem ajudar a reduzir o consumo de energia
- Sistemas heterogéneos: lógica customizada para funções bem definidas, *CPUs* + software para tudo o resto.



20

## Plataformas

- Plataforma de computação embebida:
  - arquitetura de hardware + software associado
- Muitas plataformas integram vários microprocessadores

Exemplos:

- Multiprocessadores em telemóveis
- Redes de processadores em automóveis

21



## O Que Significa "Performance"?

- Na computação de propósito geral, performance significa *throughput* médio, que pode não ser bem definido.
- Em sistemas tempo real, performance significa cumprir as restrições
  - Falhar o *deadline* pode resultar em consequências graves
  - Término após o *deadline* pode ser igual a não executar!

22

## Caraterização da Performance

- Necessitamos de analisar o sistema a vários níveis de abstração para compreender a performance:
  - CPU
  - Plataforma
  - Programa
  - Tarefas
  - Multiprocessadores





## Desafios no desenvolvimento de Sistemas Embebidos (1)

- Qual a quantidade de hardware necessária?
  - Qual a frequência do CPU? Memória?
- Como cumprir as restrições temporais?
  - Hardware rápido ou software inteligente?
- Como minimizar o consumo de energia?
  - Desativar lógica desnecessária? Reduzir acessos à memória?

## Desafios no desenvolvimento de Sistemas Embebidos (2)

- Funciona realmente?
  - A especificação está correta?
  - A implementação está de acordo com a especificação?
  - Como testar para características de tempo real?
  - Como testar com dados reais?
- Qual o princípio de ação no sistema?
  - Monitorização, controlo?
  - Qual é a plataforma de desenvolvimento?



25

## Metodologias de conceção

- Procedimento para projetar um sistema
- Compreender a metodologia de desenvolvimento ajuda a garantir que nada é suprido
- Compiladores, ferramentas de engenharia de software, ferramentas de CAD, etc., podem ser utilizadas para:
  - Automatizar as etapas da metodologia;
  - acompanhar a própria metodologia.



26

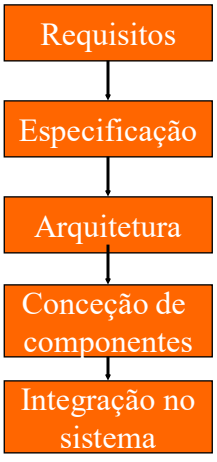
# Objetivos de conceção

- Performance
  - Velocidade, restrições temporais
- Funcionalidade e interface com o utilizador
- Custo de fabrico
- Consumo de energia
- Outros requisitos (dimensão física, materiais, etc.)





# Níveis de Abstração



## *Top-down* vs. *bottom-up*

- Conceção *top-down*:
  - Começa com a descrição mais abstrata;
  - Alcança a descrição mais detalhada.
- Conceção *bottom-up* :
  - Inicia com componentes pequenos até atingir um sistema de grandes dimensões.
- Tipicamente são utilizadas ambas as técnicas!





## Refinar Passo-a-Passo

- A cada nível de abstração deve-se:
  - Analisar o design para determinar características do estado corrente de design;
  - Refinar o design para acrescentar detalhe.

## Requisitos

- Descrição em linguagem corrente das necessidades e expetativas do utilizador.
- Podem ser obtidos de várias formas:
  - Contacto direto com os clientes;
  - Contacto direto com os representantes de marketing;
  - Disponibilizar protótipos para comentário dos utilizadores.

31

31

## Requisitos Funcionais vs. Não-Funcionais

- Requisitos funcionais:
  - indica o que o sistema deve fazer, em termos de tarefas e serviços - *output* como uma função de *input*
- Requisitos não-funcionais:
  - Tempo necessário para processar o *output*;
  - Tamanho, peso, etc.;
  - Consumo de energia;
  - Fiabilidade, etc.

32

32

# Formulário de Requisitos

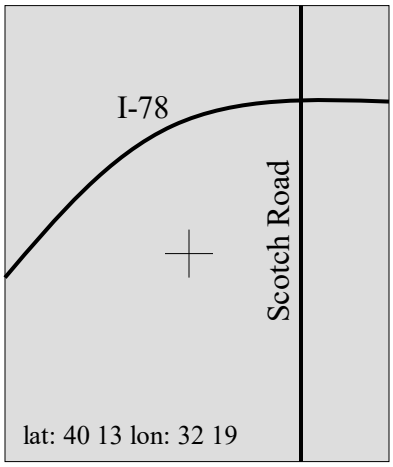
nome
propósito
inputs
outputs
funções
performance
custo fabrico
consumo energia
dimensão/peso

33

33

# Exemplo: requisitos de um mapa de GPS (1)

- As coordenadas GPS identificam a posição no mapa, definem o traçado a partir de uma base de dados local.




34

34