IM420 Sistemas Embarcados de Tempo Real

Notas de Aula – Semana 02

Prof. Denis Loubach
dloubach@fem.unicamp.br

Programa de Pós-Graduação em Eng. Mecânica / Área de Mecatrônica
Faculdade de Engenharia Mecânica - FEM
Universidade Estadual de Campinas - UNICAMP

1^o Semestre de 2018

Tópicos

Por volta de 1946 ...

Electronic Numerical Integrator And Computer (ENIAC)

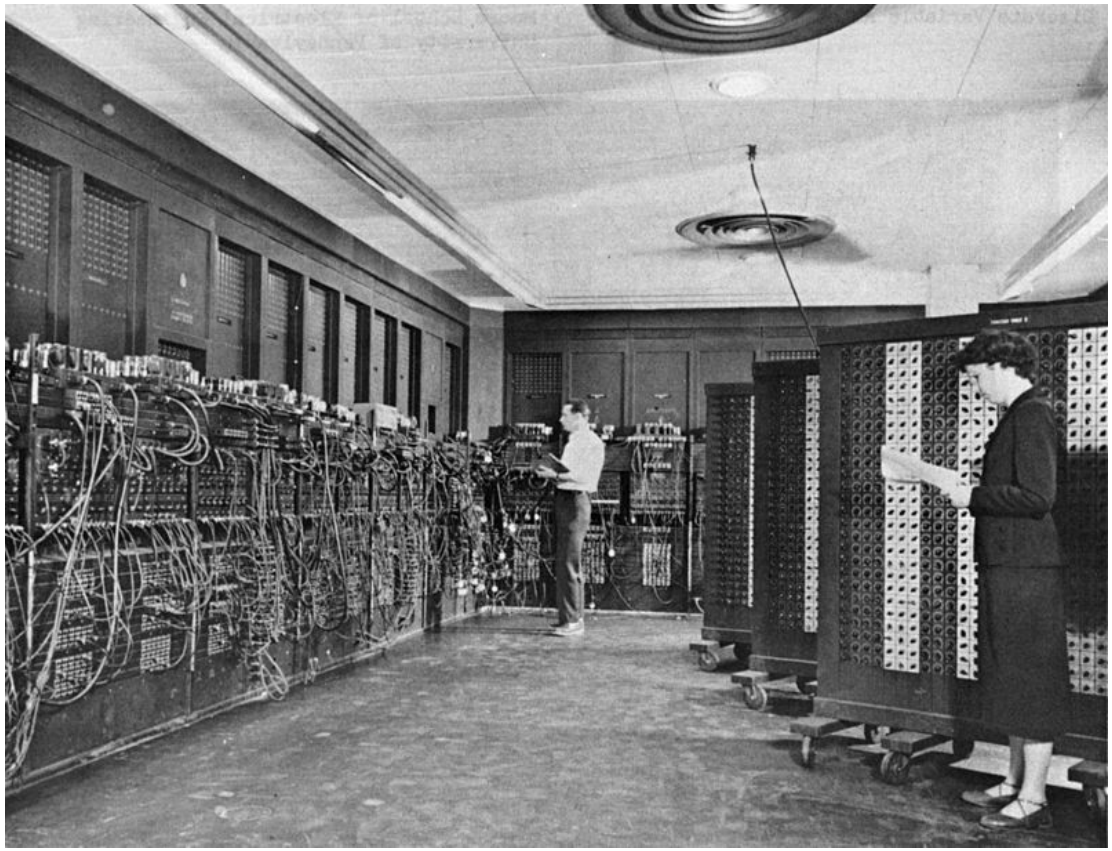


Figura: Programando o computador, fonte: `http://en.wikipedia.org/wiki/ENIAC`

Atualidade ...

Smartphones

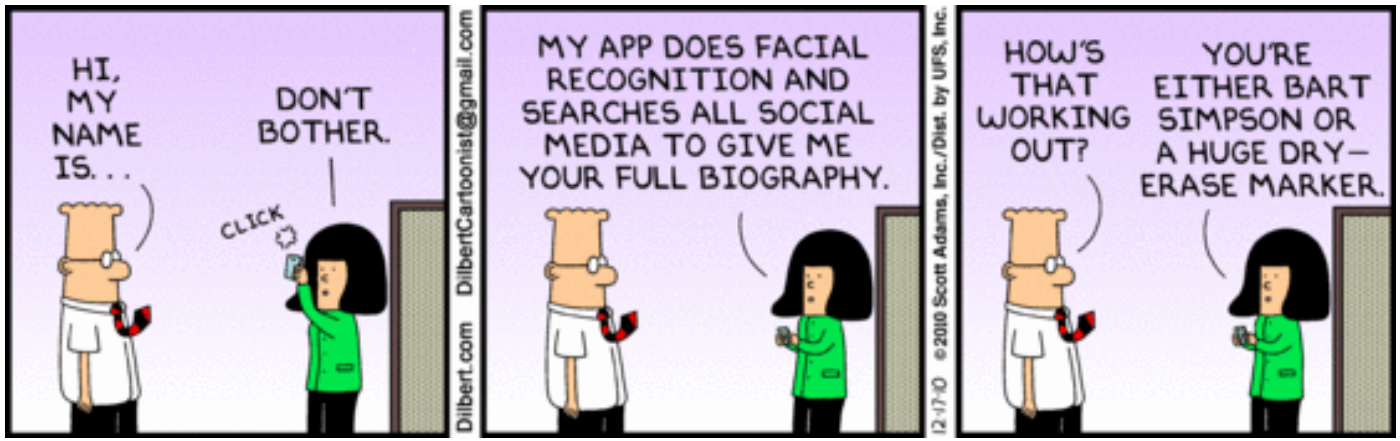


Figura: Dilbert strip, fonte: `http://www.dilbert.com/fast/2010-12-17/`

Sistemas Embarcados

Definição básica

Sistemas de computação com integração de hardware e software fortemente acoplados, projetados para executar uma função específica

- O termo "embarcado" reflete o fato do sistema ser parte integrante de um sistema maior, conhecido como sistema embarcado
- É possível que múltiplos sistemas coexistam num mesmo sistema embarcado

Discussão: atualmente a questão de se executar uma função específica encontra-se em "revisão", ou seja, os sistemas estão tendendo a ser mais "genéricos" e flexíveis...

Exemplos

- Computadores de bordo (automotivos, aeronáuticos, espaciais)
- Centrais telefônicas
- Eletrodomésticos
- Urna eletrônica de votação

Principais componentes

- Unidade de processamento (uC, uP)
- Memória de programa (ROM, EEPROM, FLASH)
- Memória de dados (RAM, SDRAM)
- Elementos de entrada (sensores, botões, ...)
- Elementos de saída (atuadores, leds, displays, ...)

Dispositivos computacionais

- uP
- uC
- DSP
- FPGA
- ASIC
- SoC
- MPSoC
- NoC

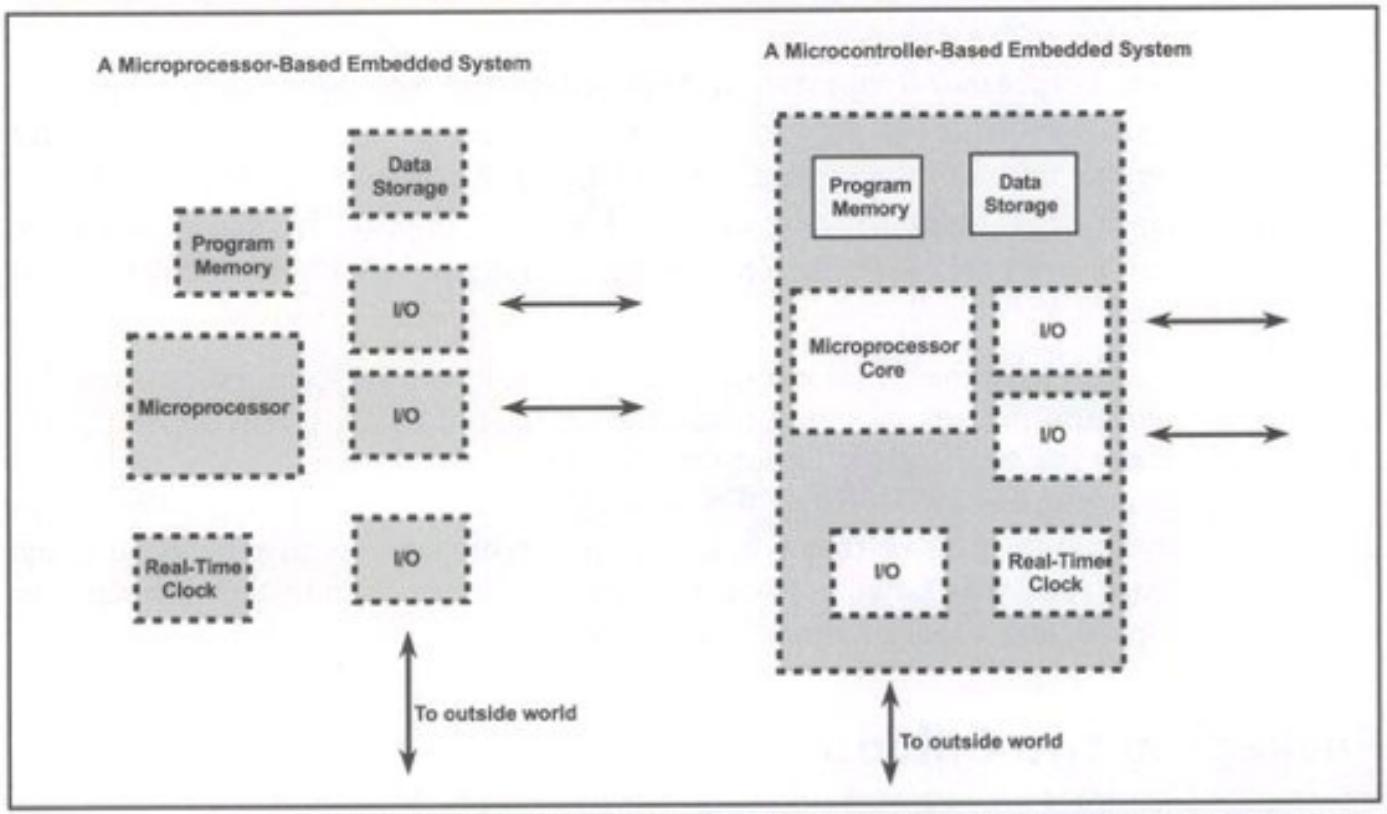
Microprocessador (uP)

- CPU num chip
- Inventados na década de 1970
- Intel 4004 foi um dos mais populares (anunciado em 15 de novembro de 1971 na revista *Electronics News*)
- uP revolucionou o controle
  - Sistemas complexos para controle realimentado
  - Carburador -> injeção eletrônica
  - Automóveis com 70+ uP dedicados
  - Computador de comando de voo de aviões (F-16)

Microcontrolador (uC)

- uP, memórias e periféricos de E/S num mesmo chip
- O Texas Instruments TMS 1000 foi o primeiro modelo, lançado em 1974
- Em 1977 a Intel lançou o 8048
- Exemplos:
  - PIC
  - ARM
  - 8051

uP e uC



In a microprocessor-based system, the CPU and the various I/O functions are packaged as separate ICs. In a microcontroller-based system many, if not all, of the I/O functions are integrated into the same package with the CPU.

Figura: Sistemas com uP e uC

Digital signal processor (DSP)

- Basicamente, processador de função específica e otimizado para execução de algoritmos de processamento de sinais
  - Filtros
  - Convoluções
  - Transformadas
- Várias unidades de execução de instrução
- Eficiente para operações com matrizes

Field-Programmable Gate Array (FPGA)

- Hardware reconfigurável
- Circuitos que podem ser alterados dinamicamente
- Modelo conceitual:
  - Gates de propósito geral (AND, OR, NOT, XOR)
  - Matriz de elementos de interconexão programável
  - Memória e registradores de propósito geral
  - Memória de configuração, quando programada, conecta os dispositivos nos blocos de circuito desejado
- Principais fabricantes:
  - Altera
  - Xilinx

Application Specific Integrated Circuit (ASIC)

- Circuitos integrados (CI) projetados para uma utilização particular
- Geralmente, projetos otimizados para performance de execução e consumo de energia
- Exemplo: CI para rodar um gravador digital de voz, acelerador gráfico

System-on-Chip (SoC)

- CI contendo todas as partes do sistema (unidade de processamento, memórias, periféricos digitais e analógicos, FPGA, ...)

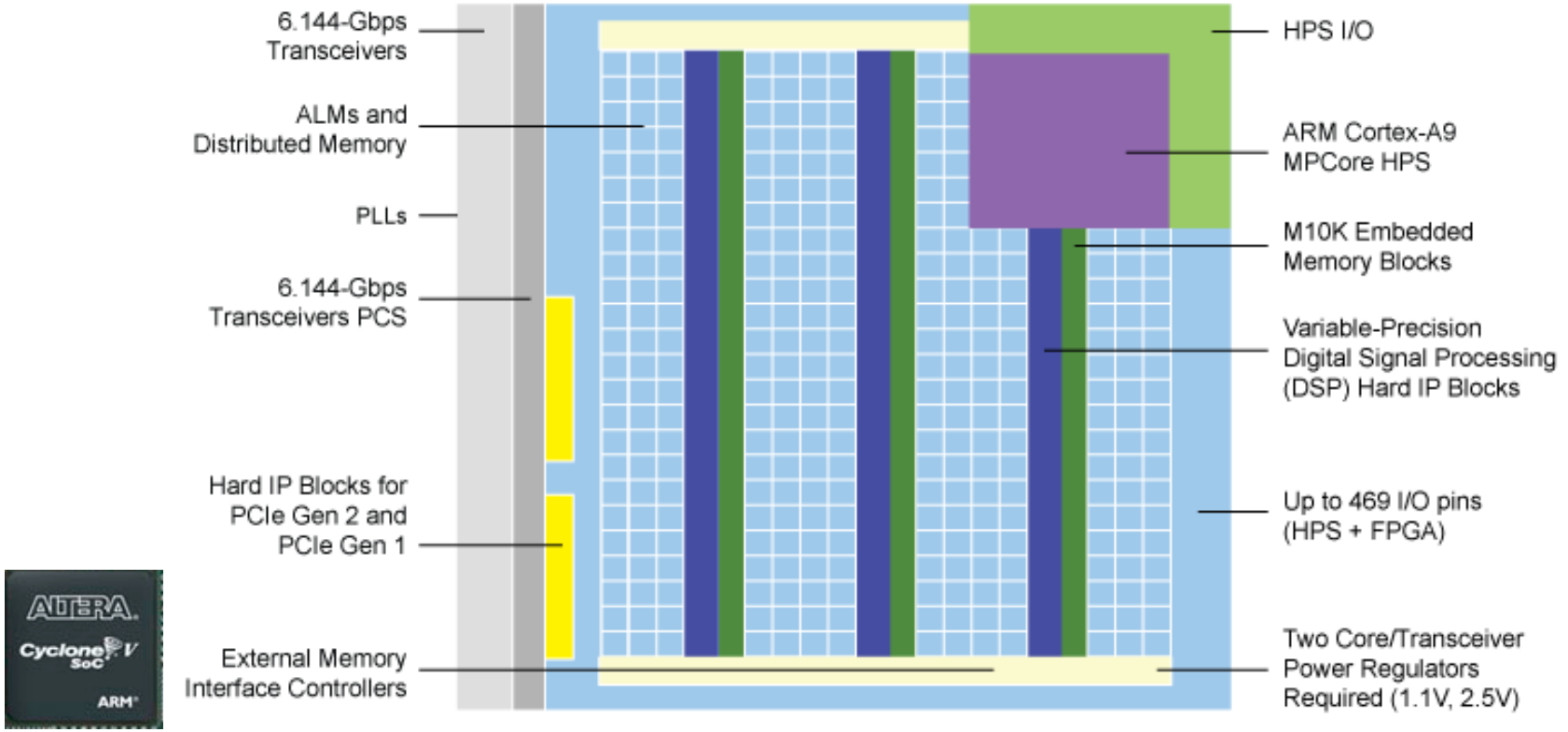


Figura: Cyclone V SoC, fonte: `http://www.altera.com`

Multiprocessor System-on-Chip (MPSoC)

- SoC com vários uP
- uP podem ser homogêneos ou heterogêneos
- Exemplo:
  ACROSS MPSoC
  ref: `http://dx.doi.org/10.1109/DSD.2012.126`

Network-on-Chip (NoC)

- Basicamente, NoC é um SoC com infraestrutura de comunicação intra-chip otimizada

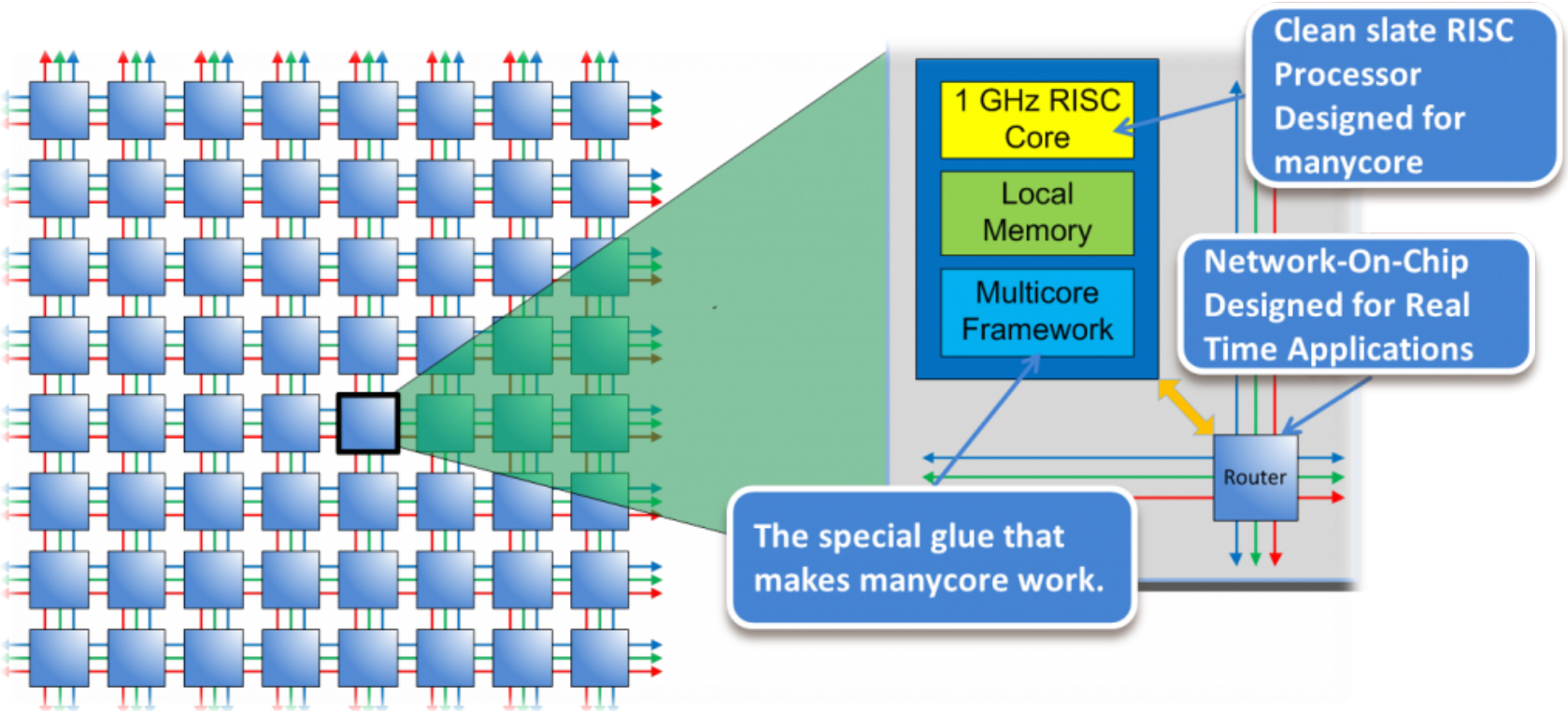


Figura: Epiphany Multicore, fonte: `http://www.adapteva.com`

Ordernação de bits - *Endianness*

- Geralmente, fonte de problema na integração de sistemas embarcados
- Exemplo: apresentação de um dado de 8bits: `0x88 (hexadecimal)`

Alinhamento a direita (*Little endian*)

1	0	0	0	1	0	0	0
7	6	5	4	3	2	1	0

MSB ... LSB

Alinhado a esquerda (*Big endian*)

1	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7

MSB ... LSB

Principais linguagens de programação utilizadas

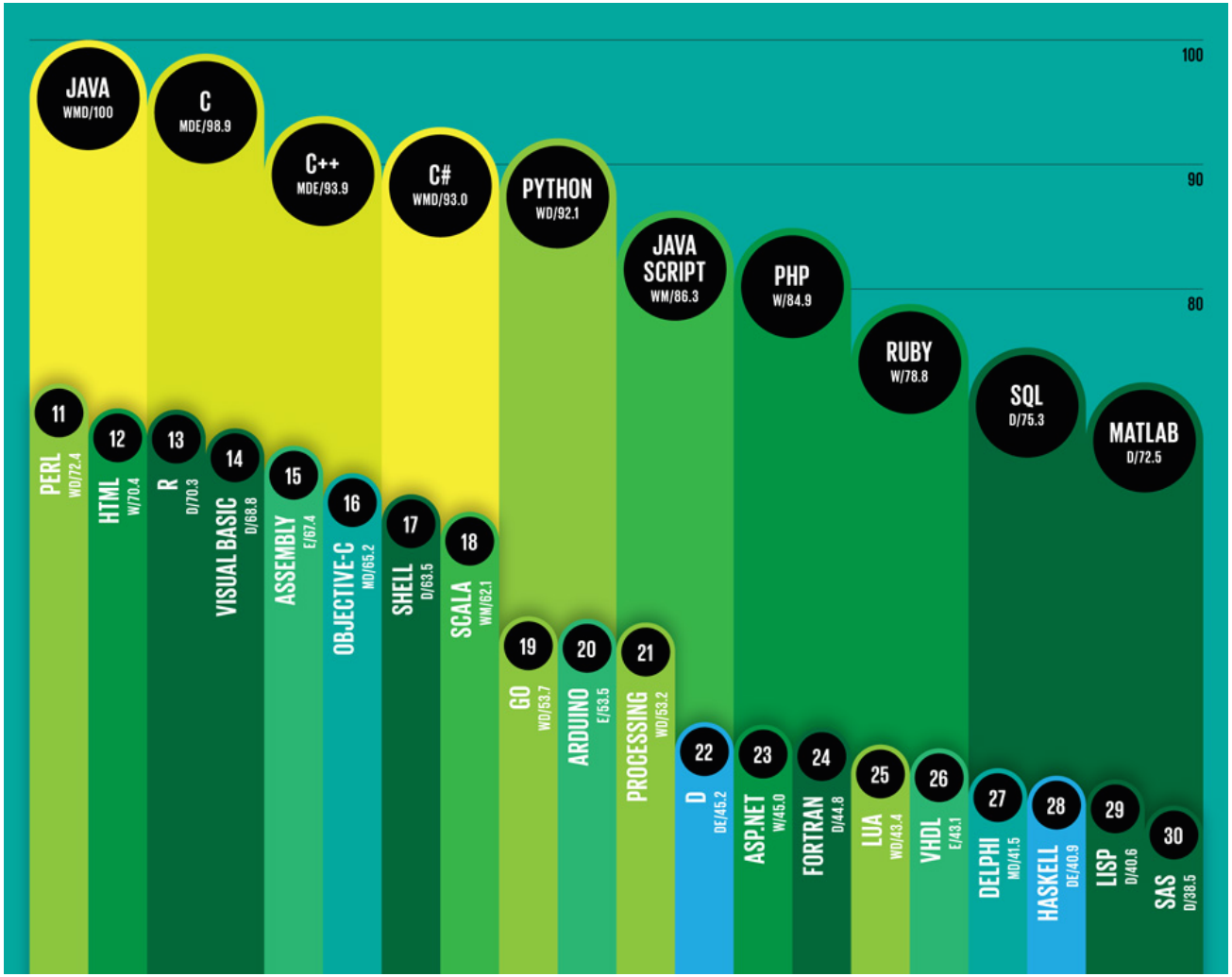


Figura: Top Programming Languages – Spectrum's 2014 Ranking,
fonte: `http://spectrum.ieee.org/computing/software/top-10-programming-languages`

As 10 primeiras, segundo rank da IEEE spectrum

Plataformas listadas em ordem: *web, mobile, desktop, embedded*

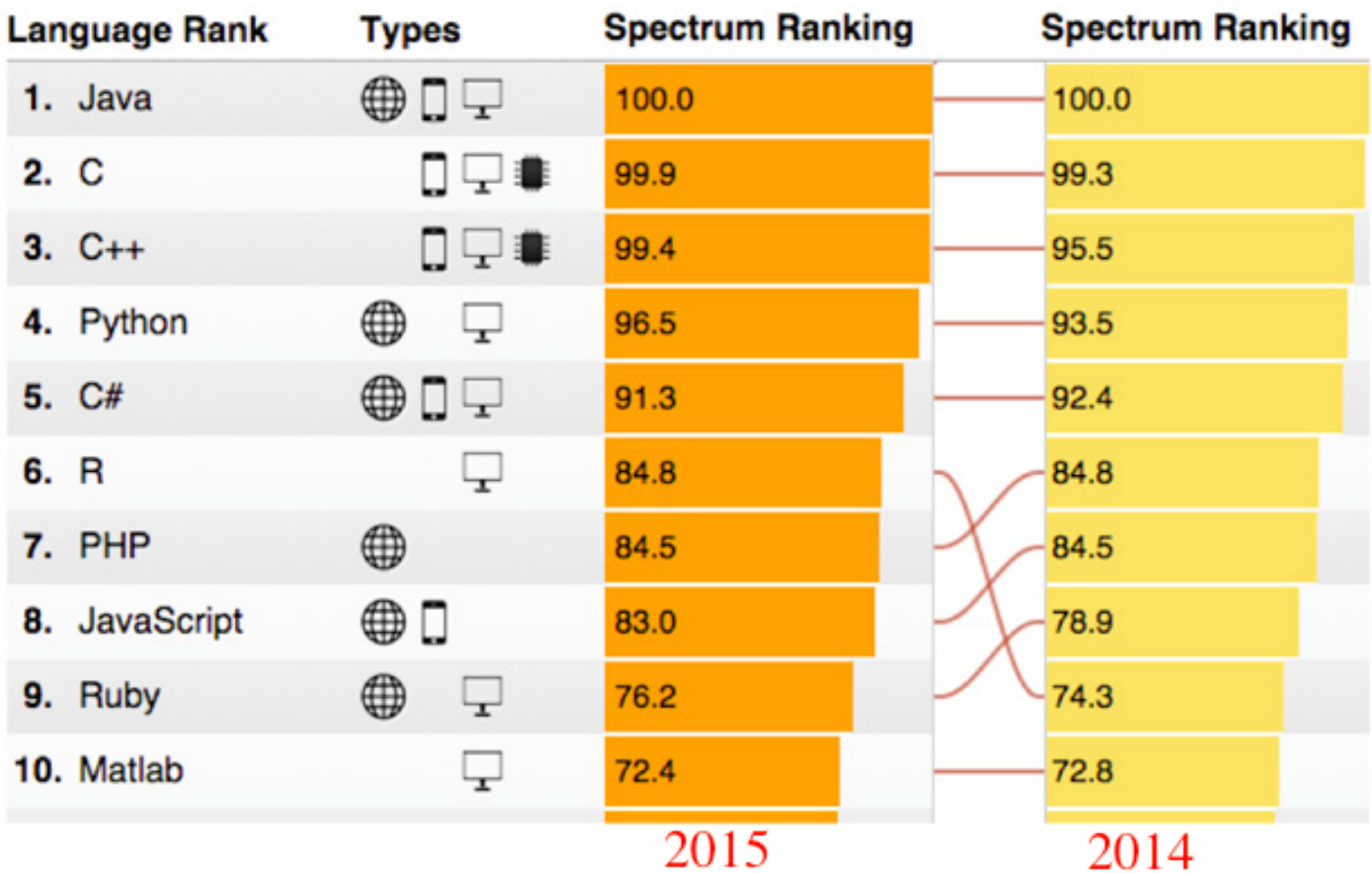


Figura: Top 10 Programming Languages – Spectrum's 2015/2014 Ranking,
fonte: `http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages`

Outras linguagens

Além das linguagens C e C++, outras linguagens figuram no desenvolvimento de sistemas embarcados

Assembly

- linguagem de programação de baixo nível
- específica para cada plataforma de hardware
- código em *assembly* é convertido em código objeto através do *assembler*

Ada

- linguagem de programação de alto nível
- orientada a objetos
- utilizada mais comumente em desenvolvimentos militares
- homenagem à *Ada Lovelace*, creditada como primeira mulher a escrever um programa de computador

Sistemas de tempo real

Definição básica

Sistemas de computação que devem responder dentro de restrições de tempo muito bem definidas. Portanto, o comportamento de tais sistemas depende igualmente da corretude de sua lógica de computação e do tempo em que o resultado é produzido

Tempo: principal aspecto do sistema; relacionado estritamente com o ambiente em que o sistema opera

Real: indica que a reação do sistema a eventos internos/externos, devem ocorrer durante sua execução

Tempo real não significa tempo de resposta rápido!

Classificação de sistemas de tempo real

Soft real time

- Restrições de tempo são mais flexíveis
- Em caso de falha não oferecem grandes riscos
- Ex: falha de uma operadora de telefone

Hard real time

- Restrições de tempo rígidas
- Em caso de falha oferecem grandes riscos financeiros ou até vidas humanas
- Resultados antecipados ou atrasados são igualmente perigosos
- Ex: computador de comando de voo

Exemplos

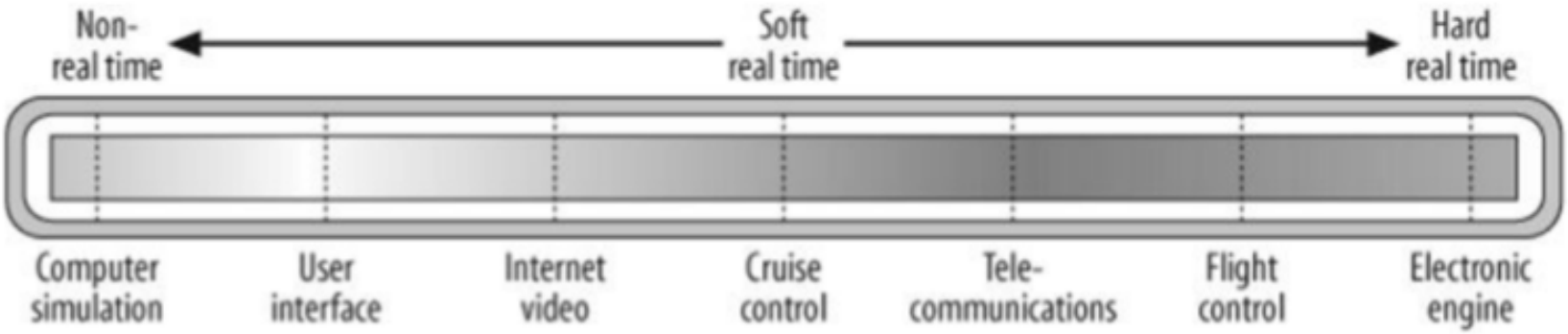


Figura: Classificação de sistemas de tempo real, fonte: BARR, M.; MASSA, A. Programming Embedded Systems: with C and GNU Development Tools. 2nd. ed. Sebastopol: O'Reilly Media, 2006. 301 p.

Ilustração

Sistemas embarcados ∩ Sistemas de tempo real

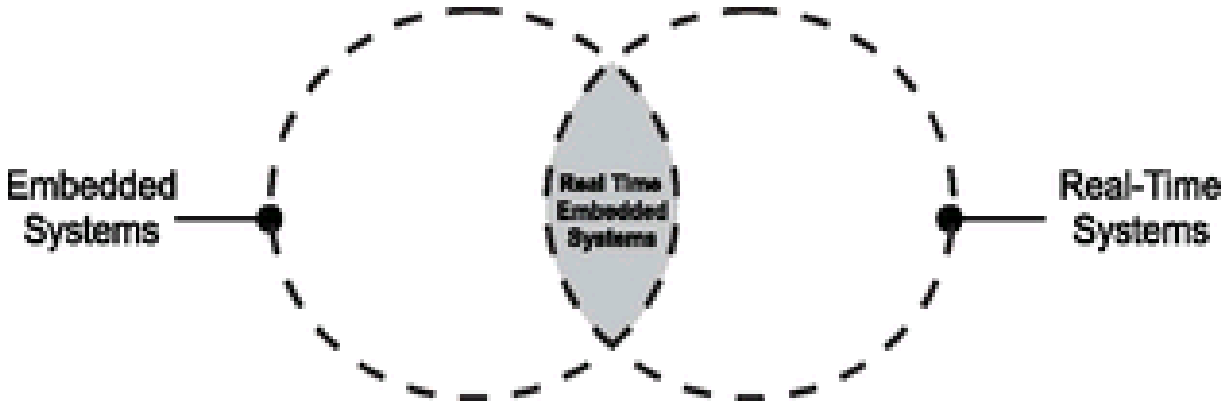


Figura: Ilustração de sistemas embarcados de tempo real

Desenvolvimento de sistemas embarcados

O projeto de sistemas embarcados deve considerar o desenvolvimento de hardware e software em paralelo

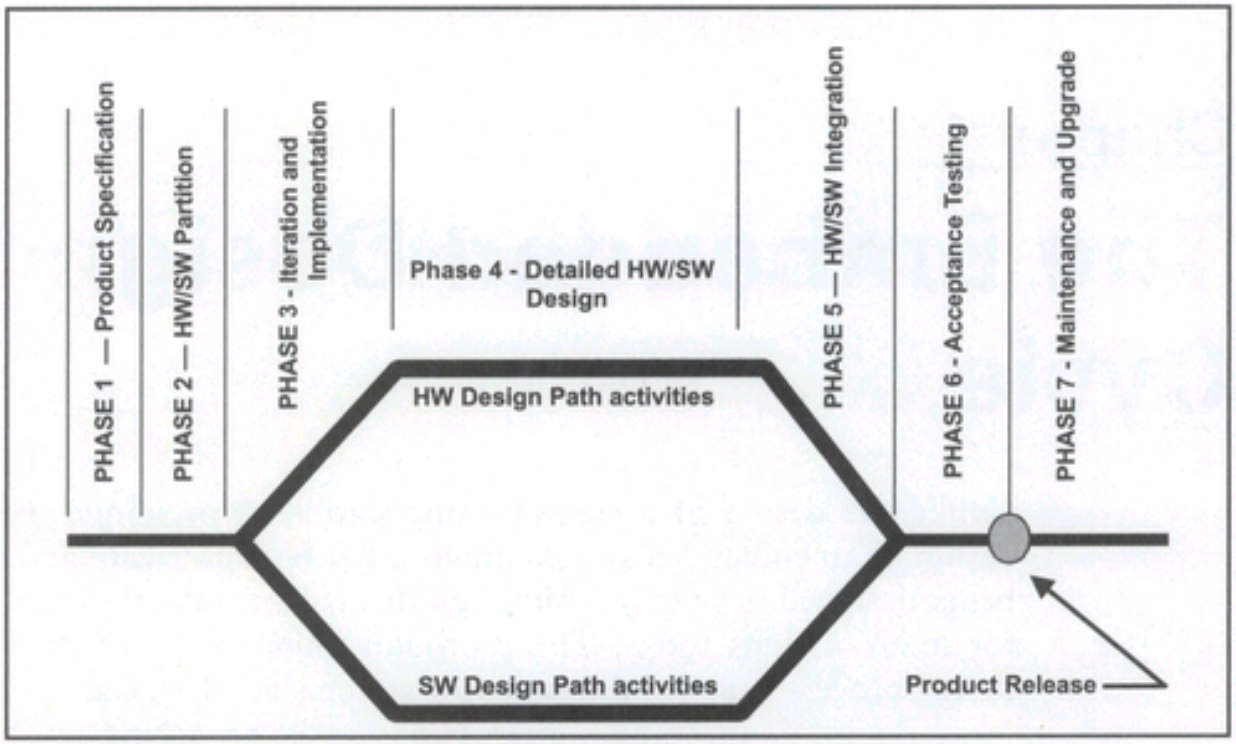


Figura: Exemplo ciclo de desenvolvimento

Fases do projeto

1. Especificação do produto
2. Particionamento do projeto entre HW/SW
3. Iterações e refinamento do particionamento
4. Desenvolvimento HW/SW em paralelo
5. Integração HW/SW
6. Testes
7. Manutenção e atualizações

Ambiente de desenvolvimento

Principais questões

- Sistemas embarcados rodam em hardware único/específico
- Linguagem de máquina nunca é independente de máquina

Com base nisto, parte do esforço de desenvolvimento é focado em ferramentas para transformação dos códigos fonte para rodar na plataforma de hardware escolhida

Para sistemas embarcados, o ambiente de desenvolvimento é do tipo *cross-platform*

Cross-platform

O software para o sistema embarcado é desenvolvido em uma plataforma (ex: x86 com win/linux/mac) para ser carregado e executado em outra plataforma (ex: ARM/Linux RT, PIC/uC-OS II, ARM/FreeRTOS)

O conceito de plataforma engloba hardware, sistema operacional e ferramentas de desenvolvimento de software usadas para desenvolvimento

Host é o sistema no qual o software embarcado é desenvolvido

Target é o sistema embarcado em desenvolvimento

Cross-platform

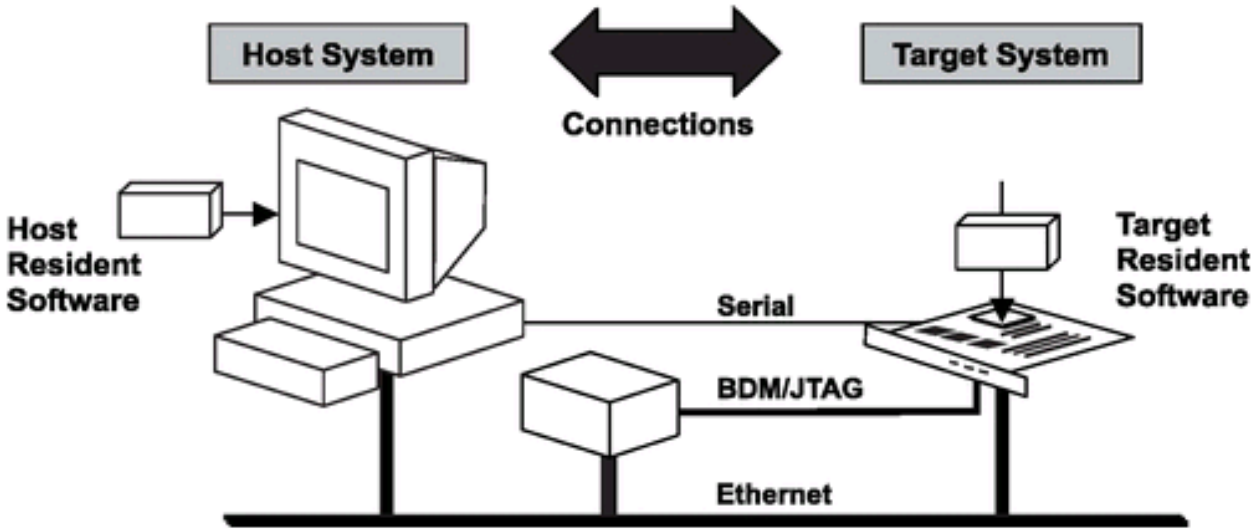


Figura: Típico ambiente de desenvolvimento *cross-platform*

Ferramentas típicas para o desenvolvimento de software embarcado

Também chamado de *tool-chain*

- *Cross compiler*
- *Assembler*
- *Linker*
- *Debugger*

Cross compiler

Compilador que roda numa arquitetura de processador mas produz código objeto (a partir do código fonte em linguagem C/C++) para outra arquitetura de processador

É utilizado porque o geralmente o *target* não é concebido para hospedar um compilador

Assembler

Gera código objeto a partir de código fonte em linguagem *Assembly*

Código objeto contém código de máquina em binário e dados do programa

Linker

Gera uma "imagem executável" para rodar no *target* a partir dos códigos objetos produzidos pelo *cross compiler* ou *assembler*

Opcionalmente, pode ainda gerar outro código objeto para ser posteriormente utilizado em outras fases do *linking* (para ser "linkado" com outros código objeto)

Neste passo decide-se como os código objeto serão combinados e onde (FLASH, RAM) será colocado o código binário e os dados no *target*

Debugger

Ferramenta que é executada no *host* e que permite rodar o software embarcado a partir do *target*, visando observar seu comportamento em tempo de execução

O software pode ser rodado observando-se o código fonte (linguagem C/C++/*Assembly*), endereços de memória, conteúdo de variáveis, entre outros

Muito útil para se testar lógica de computação

Exige cuidado para se depurar sistemas de tempo real, pois como o *debugger* interrompe a execução do software, os requisitos de tempo real **não** são mais satisfeitos...

A figura a seguir ilustra a visão geral do processo de **geração do código executável** através da utilização do *tool-chain*

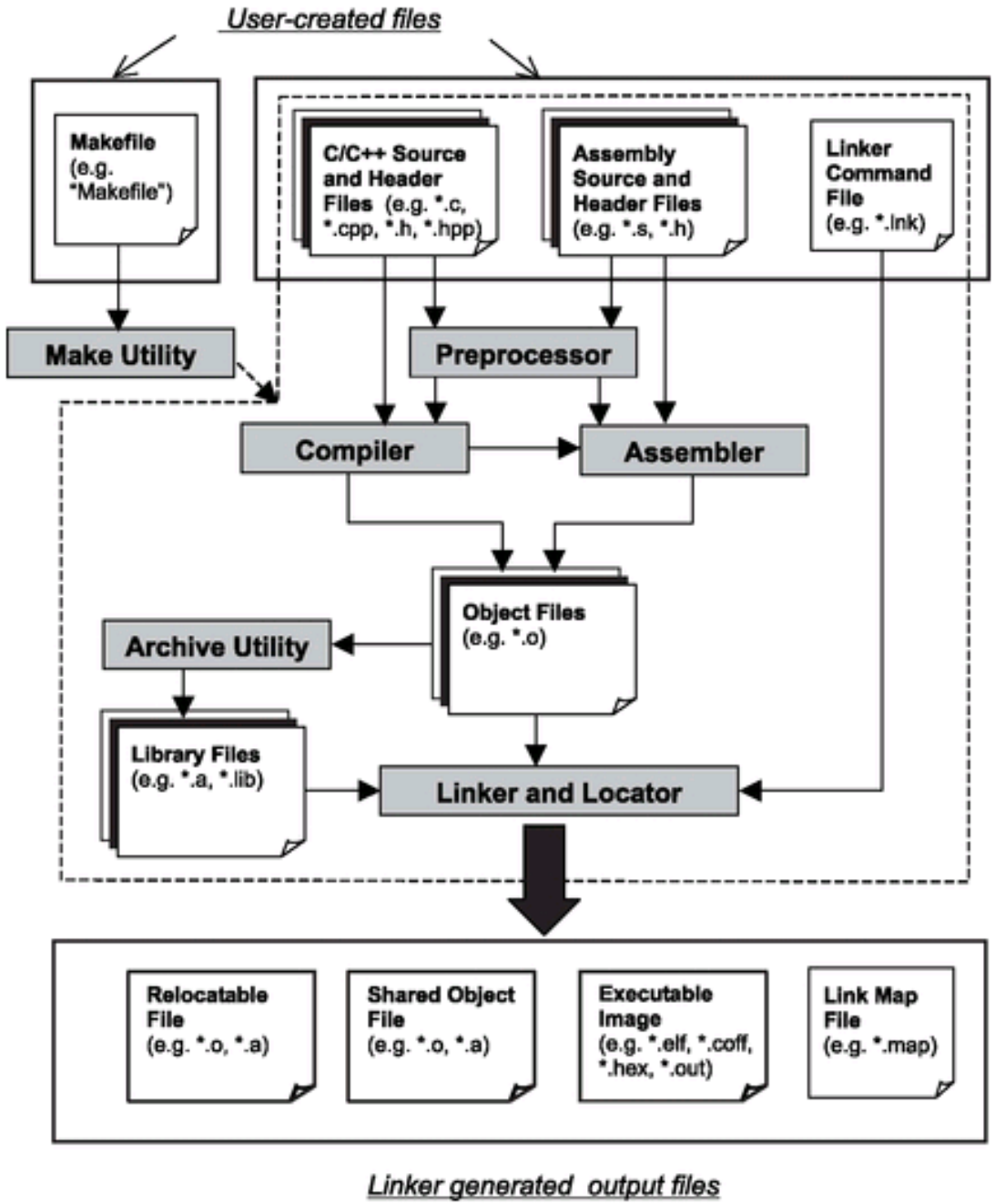

User-created files
Makefile (e.g. "Makefile")
C/C++ Source and Header Files (e.g. *.c, *.cpp, *.h, *.hpp)
Assembly Source and Header Files (e.g. *.s, *.h)
Linker Command File (e.g. *.lnk)
Make Utility
Preprocessor
Compiler
Assembler
Object Files (e.g. *.o)
Archive Utility
Library Files (e.g. *.a, *.lib)
Linker and Locator
Relocatable File (e.g. *.o, *.a)
Shared Object File (e.g. *.o, *.a)
Executable Image (e.g. *.elf, *.coff, *.hex, *.out)
Link Map File (e.g. *.map)
Linker generated output files

Ilustração: imagem executável

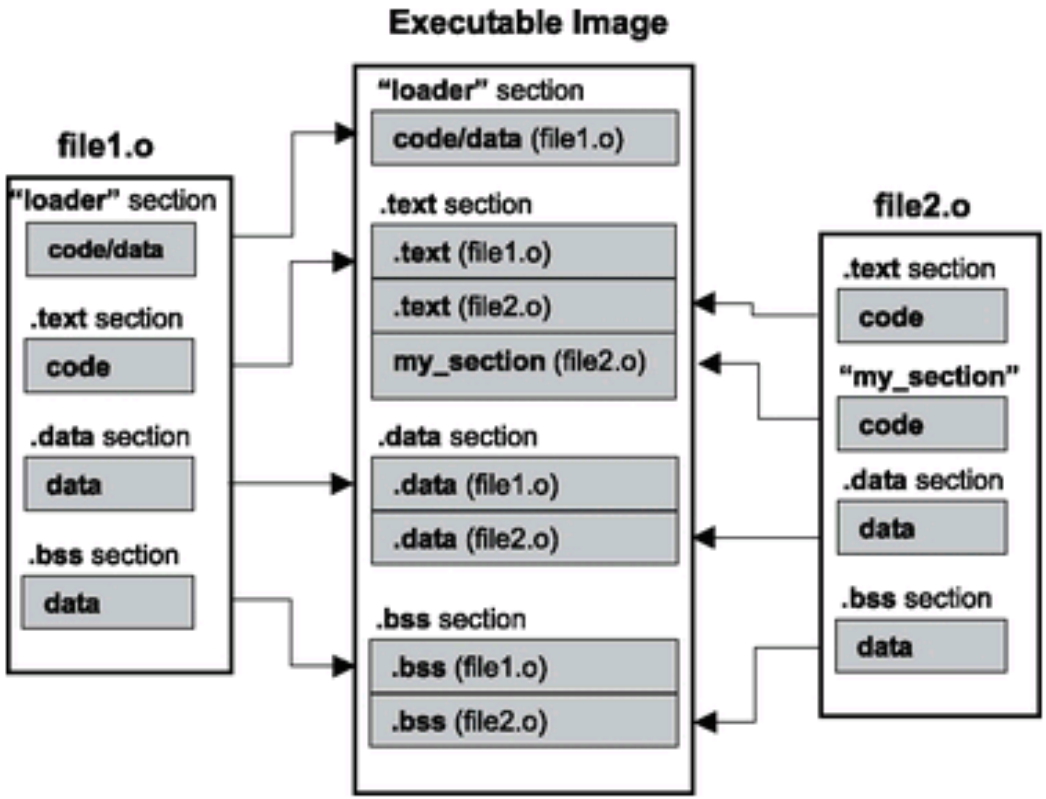


Figura: Exemplo de uma imagem executável e "seus" códigos objeto correspondentes

.text - código de programa e constantes, seção *read-only*
.data - dados (variáveis inicializadas)
.bss - dados (variáveis não inicializadas)

A figura a seguir ilustra a visão geral do **processo de inicialização** de um sistema embarcado

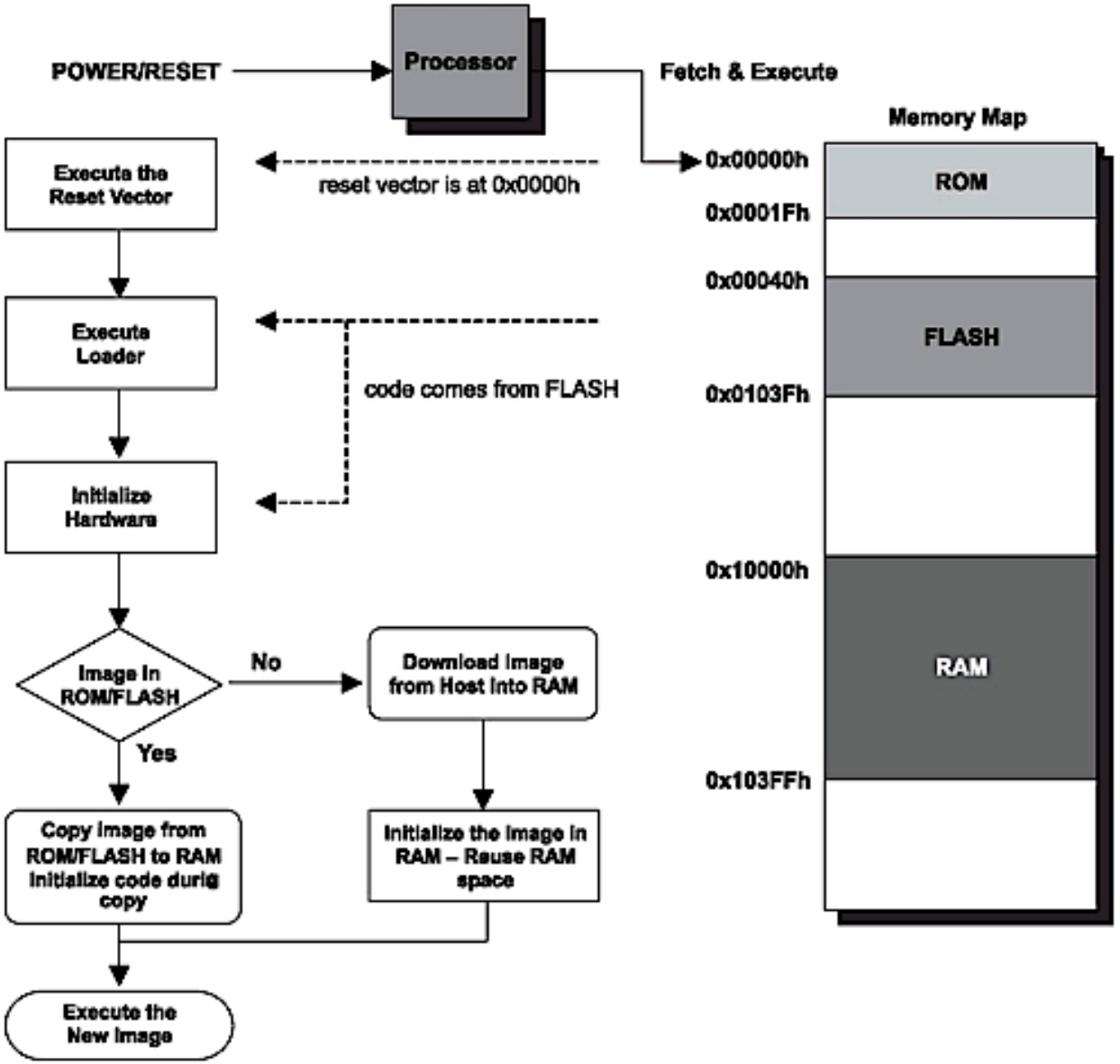


Figura: Visão geral do processo de inicialização do sistema no *target*

Algumas tendências que já se tornaram realidade

Os tópicos abaixo, representam algumas tendências que já tomaram força e se tornaram realidade além do mundo acadêmico

- *Model-based design* - MBD / *model driven development* - MDD
- Geração automática de código fonte
- Computação reconfigurável
- *Internet of Things* - IoT
- *Wearables*

Informação ao leitor

Notas de aula baseadas nos seguintes textos:


Q. Li and C. Yao, *Real-Time Concepts for Embedded Systems*.
CMP books, 2003.


A. S. Berger, *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques*.
CMP Books, 2002.


B. P. Douglass, *Real Time UML: Advances in the UML for Real-time Systems*.
Object Technology, Addison-Wesley, 3rd ed., 2004.


M. Samek, *Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems*.
Newnes (Elsevier), 2nd ed., 2009.


D. Pilone and N. Pitman, *UML 2.0 in a Nutshell*.
O'Reilly Media Inc, 2009.


L. Otávio, "Notas de Aula da ES670." 2013.