Universidade Federal do Rio Grande do Sul
Escola de Engenharia - Instituto de Informática
Graduação em Engenharia de Computação

Ferramenta para Desenvolvimento de Sistemas Embarcados Utilizando Linguagem de Alto Nível

Francisco Roberto Peixoto Socal

Prof. Dr. Altamiro Amadeu Susin
Orientador

Introdução: Sistemas Embarcados

- Sistemas dedicados ou de aplicação específica
- Aumento da complexidade com avanços tecnológicos
- Diversos fabricantes e tecnologias
- Dificuldades de projeto
  - múltiplas plataformas
  - tempo de desenvolvimento
  - prototipação rápida
  - restrições de recursos

Introdução: Engenharia de Software

- Objetivos
  - manutenção
  - portabilidade
  - flexibilidade
- Uso de técnicas de Engenharia de Software
  - modularização
  - reutilização de código
- Solução: linguagens de programação de alto nível
  - C
  - C++
  - Java

Introdução: Linguagens de Programação

- Linguagens de mais alto nível: Scheme, Python, Lua
  - flexibilidade
  - extensibilidade
  - integração com demais linguagens e programas
  - funções como valores de primeira classe
- Linguagem Lua
  - extensão de aplicações
  - simples, portável, compacta, ...
  - software livre
- Abordagem intermediária: uso de bytecodes $\rightarrow$ recompilação de código

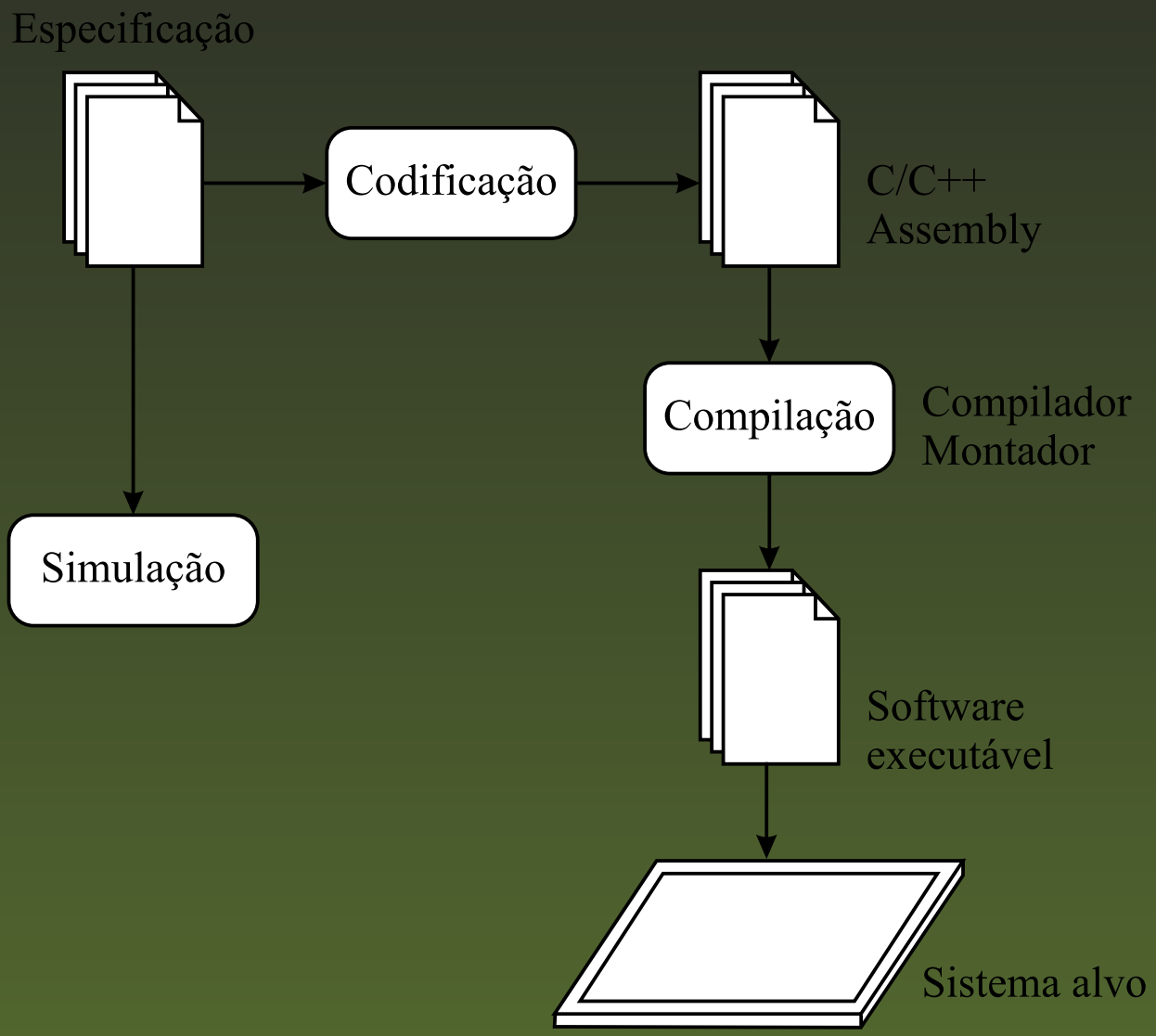
Introdução: Objetivos

- Uso da linguagem Lua em sistemas embarcados
  - já suportada em várias plataformas
  - necessidade da LVM e sistema operacional
- A ferramenta proposta
  - linguagem Lua em sistemas mínimos
  - estudo de caso: microcontroladores PIC
  - simliar ao RISC: facilidade na geração de código
  - tradução de programas em Lua $\rightarrow$ assembly PIC
  - recompilação de bytecodes
- Linguagem de alto nível para um microcontrolador
- Metologia única de desenvolvimento de software

Sistemas Embarcados

- Tarefas específicas, em sistemas maiores
  - conhecidas em tempo de projeto
  - diferentes dos sistemas de propósito geral
- Uso de microprocessadores comerciais, DSPs, ASICs, ASIPs, SOCs, ...
- Desenvolvimento de Software
  - distinção entre plataformas alvo (*target*) e de desenvolvimento (*host*)
  - desenvolvimento conjunto de hardware (co-design)
  - respeito a restrições temporais, de recursos, de projeto, ...

Sistemas Embarcados: Desenvolvimento de Software



Sistemas Embarcados: Metodologias

- Especificação como ponto de partida
  - simulação, prototipação
  - testes, codificação
- Diversas linguagens e ferramentas $\rightarrow$ aumento da complexidade envolvida
- Metodologia SASHIMI
  - descrição Java
  - síntese automática de um ASIP e seu programa
  - ferramentas padrão do Java para compilação, simulação, etc

A Linguagem Lua

- Linguagem de scripting para extensão e customização de aplicações
  - sintaxe simples, limpa e familiar
  - expressibilidade para descrição de dados
  - extensibilidade para uso em diversos domínios
- Contraponto
  - ligação de código e tipagem dinâmica
  - susceptibilidade a erros de sintaxe!

Lua: Características

- Tabelas associativas como estruturas de dados
  - solução única para implementação de arrays, listas, mapas, árvores, grafos, ...
  - containers de dados ortogonais
- tratamento de funções como valores de primeira classe
  - criação dinâmica de funções
  - migração de código $\rightarrow$ testes remotos
  - meta-suporte a orientação a objetos
- Garbage collection
- Extensão de semântica por meta-tabelas

Lua: Aplicações

- Aplicações
  - Jogos, como engines de inteligência artificial e como “lógica de cola” entre módulos
  - CGI em aplicações Web
  - Linguagem de configuração de aplicativos e equipamentos

Lua: Exemplo de Código

```
-- Criação dinâmica de funções
cache = function( f )
  local c = {}

  local ff = function( x )
     local y = c[x]
     if not y then
        y = f(x)
        c[x] = y
     end
     return y
  end

  return ff
end
```

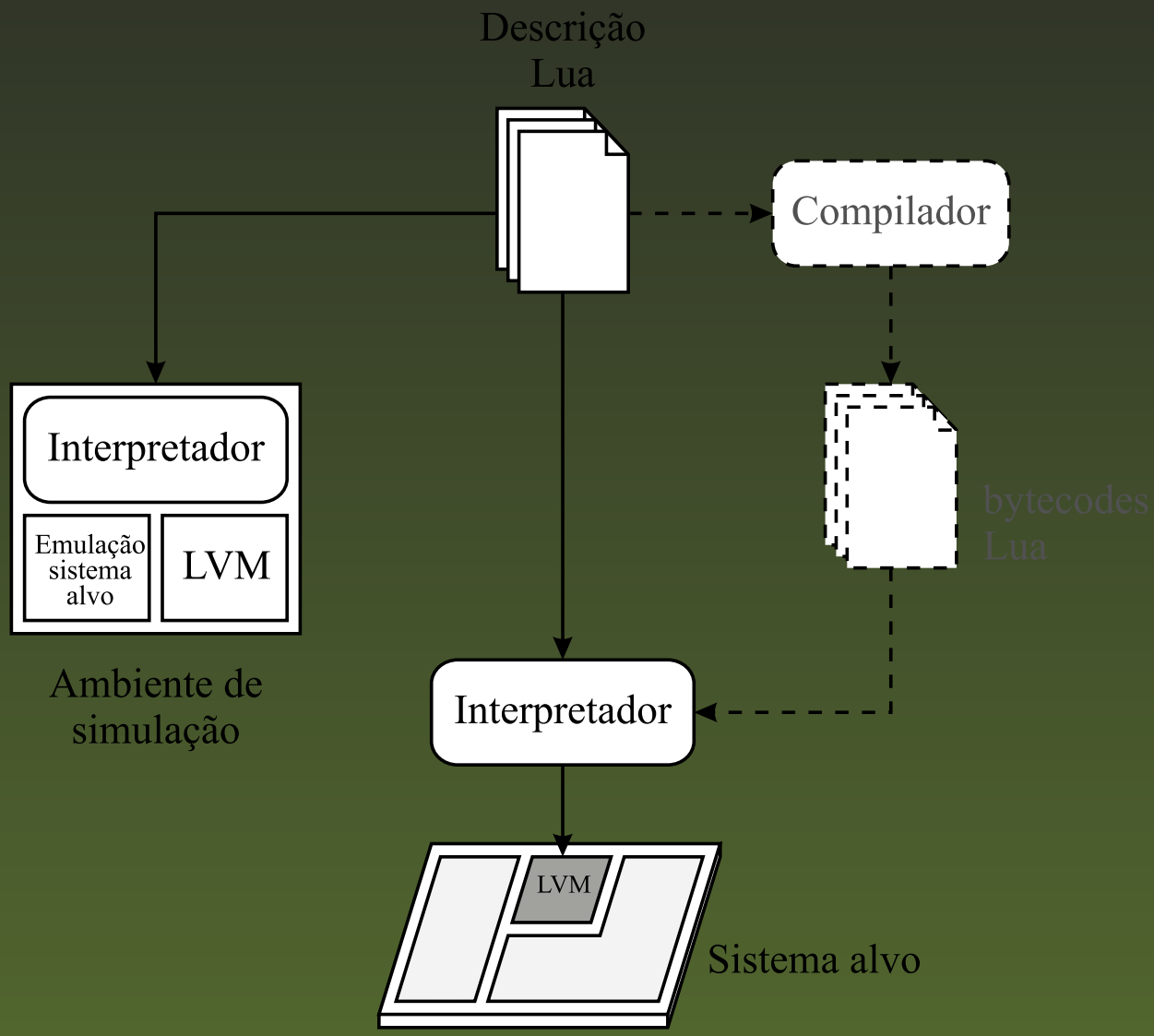
Lua: Implementação

- Disponibilizada como uma biblioteca de funções C, ligada a uma aplicação hospedeira
- Primitivas para
  - instanciar ambientes de execução Lua
  - definir e manipular dados
  - definir e chamar funções escritas em Lua ou C
  - criar primitivas em C
- Exemplo trivial: intepretador de comandos
  - lê string da entrada padrão
  - chama a biblioteca Lua para execução
  - exibe resultados retornados pela biblioteca

Lua: Máquina Virtual

- Implementação baseada na Máquina Virtual Lua (LVM)
  - estratégia intermediária entre compilação e interpretação de código
  - baseada em set de registradores (não é uma máquina de pilha)
  - correspondência com o modelo de memória do microprocessador escolhido $\rightarrow$ facilidade na geração de código

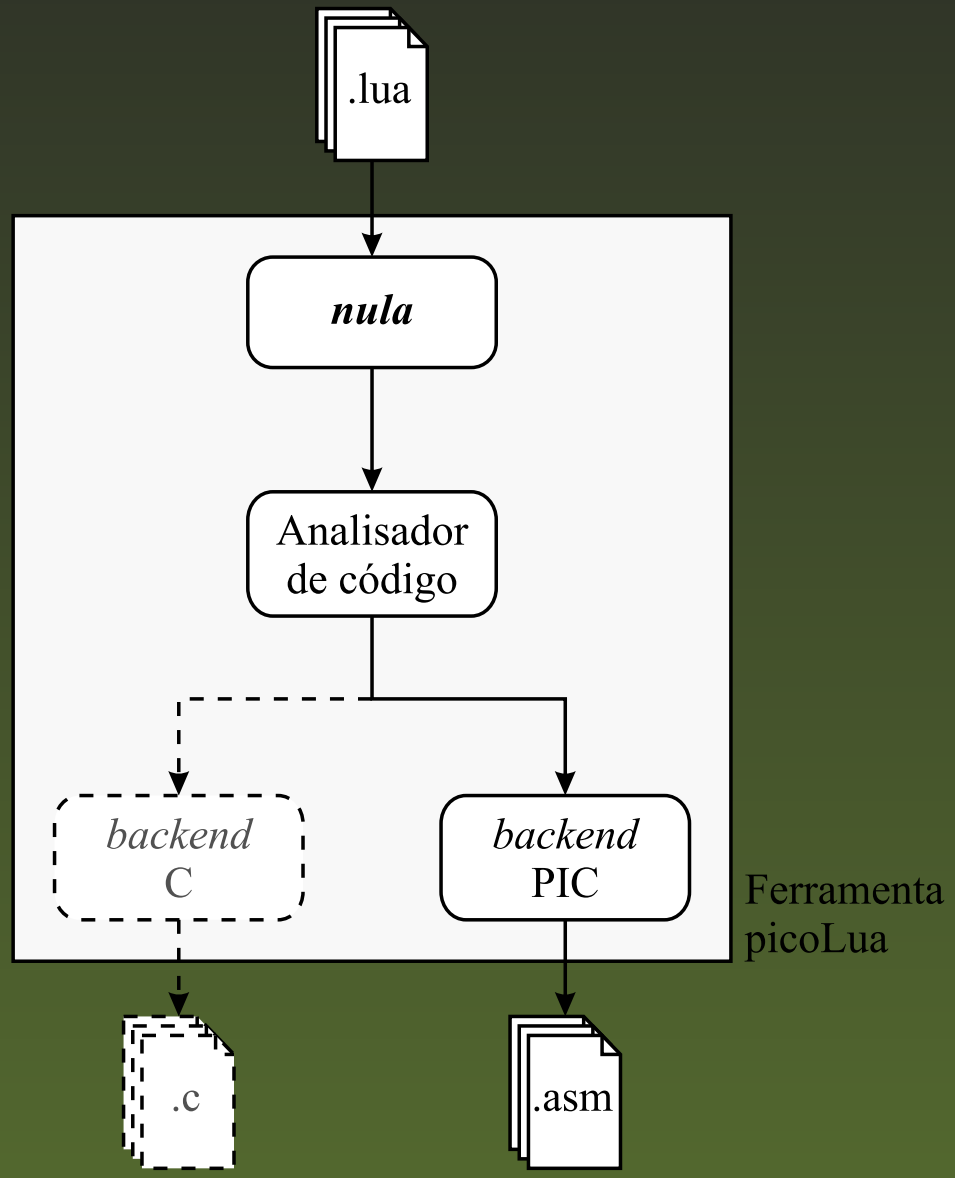
Lua: Utilização



A Ferramenta picoLua

- Recompilação de bytecodes para assembly
  - elimina análise léxica e sintática
  - dispensa o uso da Máquina Virtual e do sistema operacional
- Subconjunto da linguagem Lua suportado
  - ausência de tipagem dinâmica
  - identificação estática de chamada de funções, passagem de parâmetros, etc pela ferramenta
- Implementação modularizada, em Lua!
  - geração de código para outras plataformas

picoLua: Implementação



picoLua: Módulo *nula*

- Tradução da descrição textual para bytecodes
  - função *loadfile* da biblioteca padrão
  - resultado é uma função Lua, pronta para execução
- Tradução de função Lua para representação interpretável
  - resultado em uma tabela Lua
  - argumentos, variáveis locais, constantes, etc
  - relação de instruções: opcodes, operandos e rótulos

picoLua: Analisador de Código

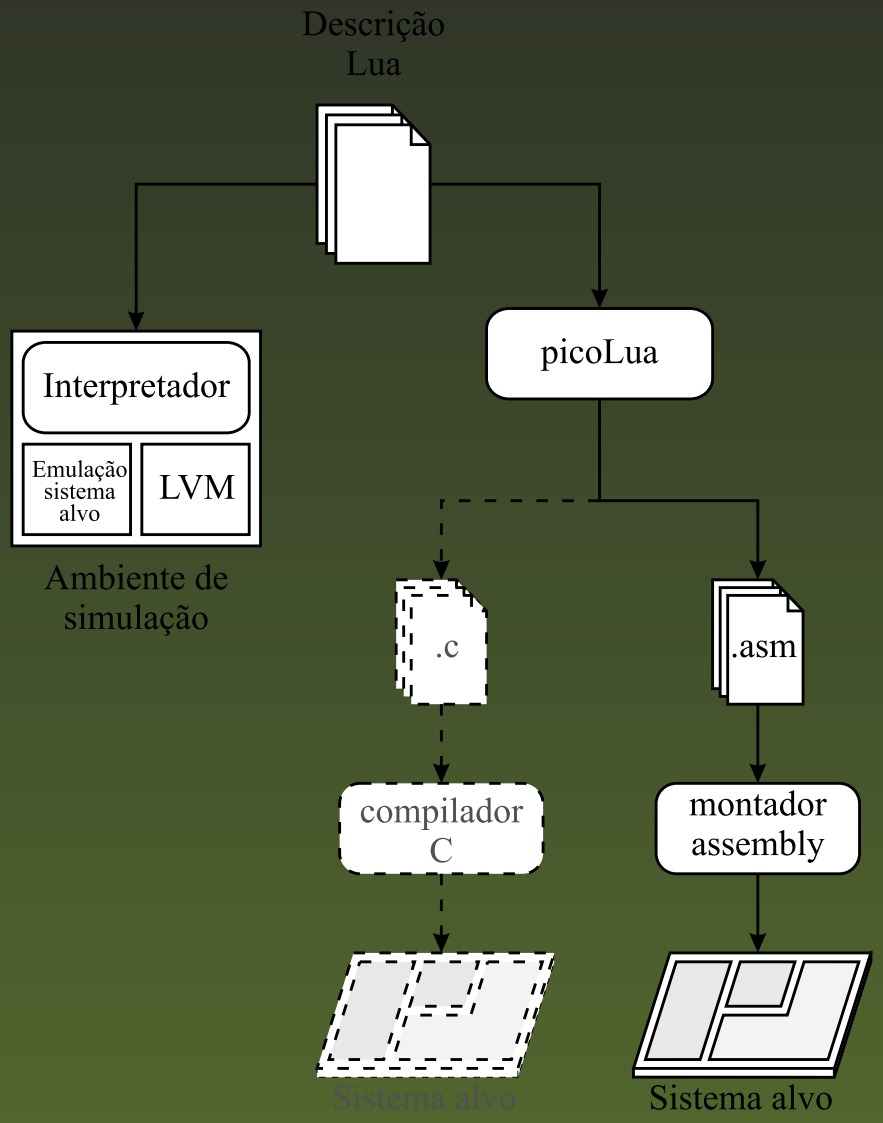
- Identificação de atribuições, operações, testes, ...
- Chamadas ao gerador de código (*backend*)

1. Tradução instrução-por-instrução
   - correspondência um-para-um com código gerado
   - MOVE, LOADK, ADD, SUB, ...
2. Análise de contexto
   - extensões da ferramenta, variáveis globais, chamadas de funções
   - criação e chamada de função
   - GETGLOBAL, CALL, CLOSURE, ...

picoLua: Backend para PIC

- Geração de código e suporte à arquitetura
  - modelo de memória
  - política de alocação de recursos
  - política de passagem de parâmetros
- Modelo de memória do PIC semelhante à LVM!
- Extensões à linguagem Lua
  - acesso a dispositivos
  - assembly inline
  - modelo do processador, configuração, ...

picoLua: Meotodologia Proposta



Estudos de Caso

- Multiplicação inteiros
  - operações aritméticas, testes e assembly inline
- Sistema de Medição de Vibração
  - acesso a dispositivos, instalação interrupções
  - modularização e reuso de código
- Modelo Aplicações Distribuídas
  - portabilidade e versatilidade da linguagem
  - computador pessoal, handheld, microcontrolador
- Testes de equipamentos: DataCom Telemática
  - depuração, prototipação
  - teste de fábrica centralizado

Desempenho

- Tamanho do código gerado, em palavras de memória
- Estimativa para otimização de código

Programa	**Assembly**	**Lua**	**Lua otimizado**
Multiplicação inteiros	17	34	29
Comunicação serial	80	179	133
Firmware AXPC	174	482	251

Conclusão

- Ligação de domínios distintos
  - sistemas embarcados, com restrições de recursos
  - linguagens de *scripting* de alto nível
- Benefícios para prototipação de sistemas
  - modularização, reuso de código
  - especificação única
  - migração de código
  - estratégias de teste

Conclusão

- Trabalhos futuros
  - otimização de código
  - melhor tratamento de erros
  - suporte a mais construções da linguagem
  - geração de código em C ou para outras arquiteturas