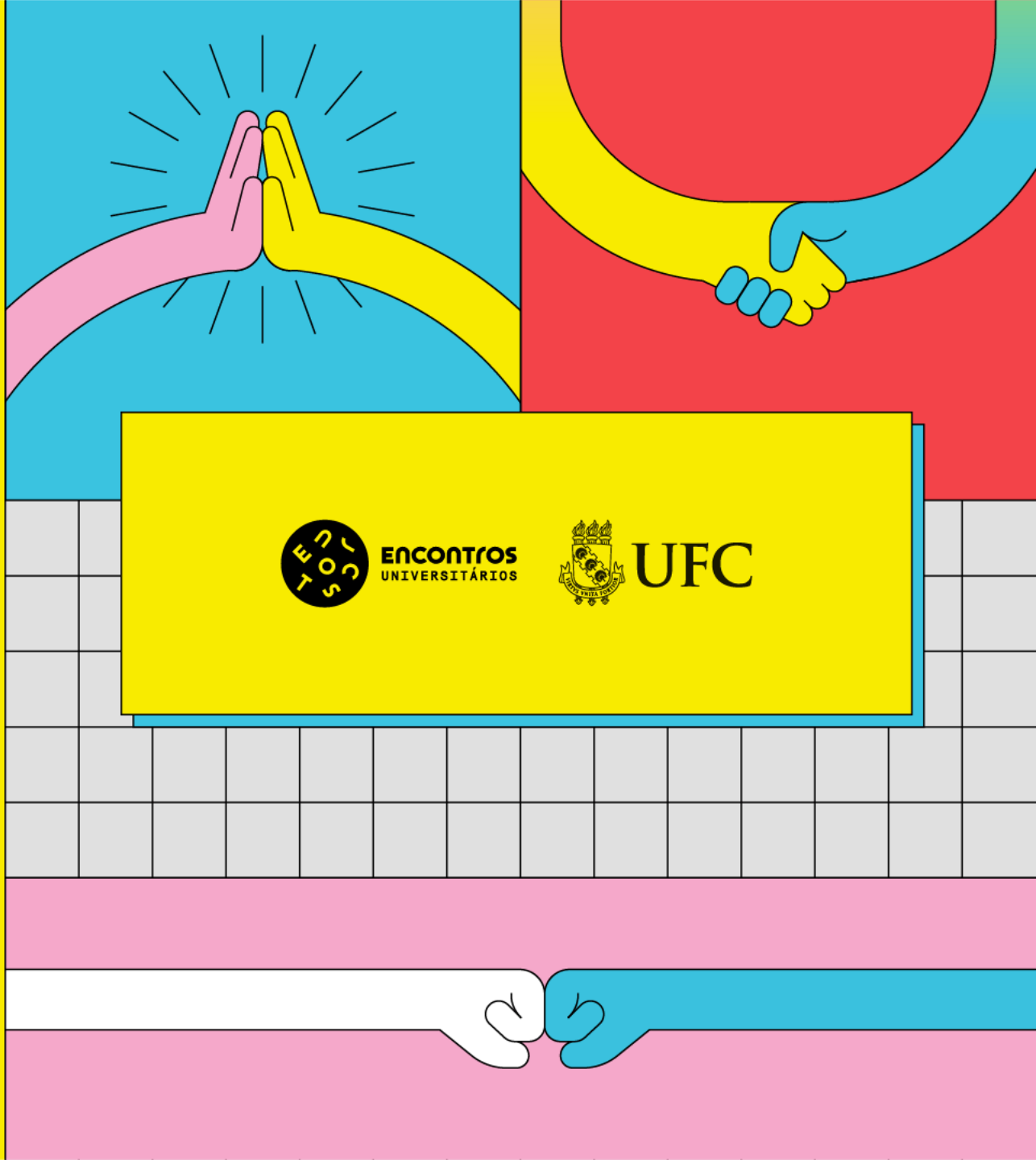
CIÊNCIA & SOCIEDADE SE ENCONTRAM NA UNIVERSIDADE



Universidade Federal do Ceará
Campus de Quixadá

O ensino de *Assembly* na construção de um pensamento crítico em programação de computadores

Bolsista: Pedro Henrique Magalhães Botelho
Orientador: Roberto Cabral Rabêlo Filho

Programa de Iniciação à Docência

Engenharia de Computação – 6° Semestre
Encontros Universitários - 2021

Introdução

- Computadores → Dia-a-dia.
- Para operar o computador → Saber seu funcionamento intrínseco.
  - Programar → Como o programa vai se comportar?
  - Aprender como o computador funciona → Capacidades do processador.
- Instruções → Operações que o computador consegue realizar.
  - Linguagem *Assembly*.
  - Entendimento do computador.
- Inspiração → Programar em C após aprender Assembly.

Objetivos

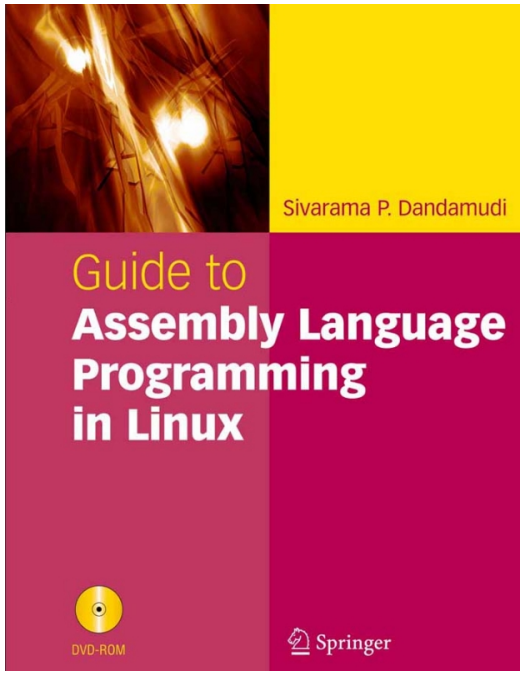
- Objetivos Principais:
  - Estabelecer o Assembly como uma **ferramenta de ensino** da arquitetura.
- Objetivos Específicos:
  - Importância de entender o **funcionamento da plataforma**.
  - Propor um **modelo de ensino** de *Assembly*, baseado em pesquisas.

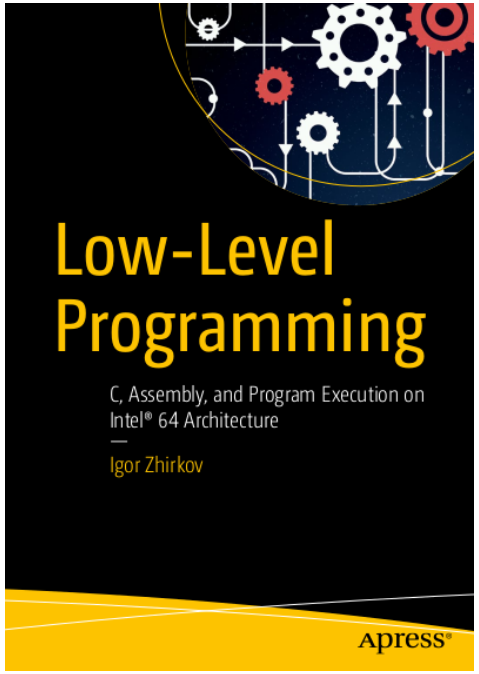
Trabalhos Relacionados

- [de Andrade 2014] → Primeira linguagem da 2ª geração.
- [Vidal 2017] → Aplicações na Engenharia de Computação.
- [Jorgensen 2020] → Vantagens na programação em geral.
- [Silva 2021] e [Faquer 2017] → Vantagens em áreas específicas:
  - Engenharia reversa, exploração de binários e otimização de código.
  - Segurança da informação.

Trabalhos Relacionados

- [Dandamudi 2005] → Conjunto de Instruções x86-32.
  - Abordagem voltada para a Arquitetura x86 em si.
- [Zhirkov 2017] → Conjunto de Instruções x86-64.
  - Abordagem crítica da programação C e *Assembly*.





Fundamentação Teórica

- Saber usar os recursos da plataforma → Entender seu funcionamento.
- **Organização de Computadores** → Componentes e interconexões.
- **Arquitetura de Computadores** → Elementos que impactam a execução.
  - **Arquitetura de Processadores** → Linha de processadores com características semelhantes.
- Computador → **Sistema Digital.**
  - Instruções para operar.
  - Cada arquitetura → Conjunto de Instruções.
  - Níveis de Abstração → Alto e baixo.

Fundamentação Teórica

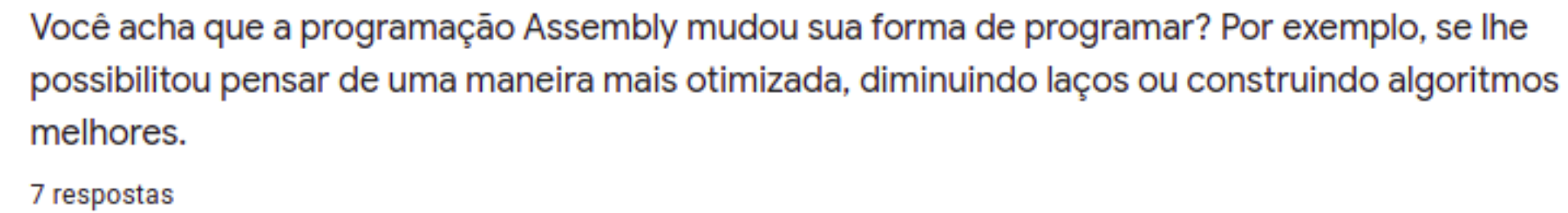
- Linguagem *Assembly* → Instruções em formato mnemônico.
  - Descreve as características de uma arquitetura.
  - Todo programa executável → Binário.
- ***Assembler*** → Transforma código Assembly em um programa executável.
- **Compilador** → Transforma código em alto nível em *Assembly*.
  - Relação intrínseca do C com o *Assembly*.
- Estudar *Assembly* → Aprender a arquitetura.

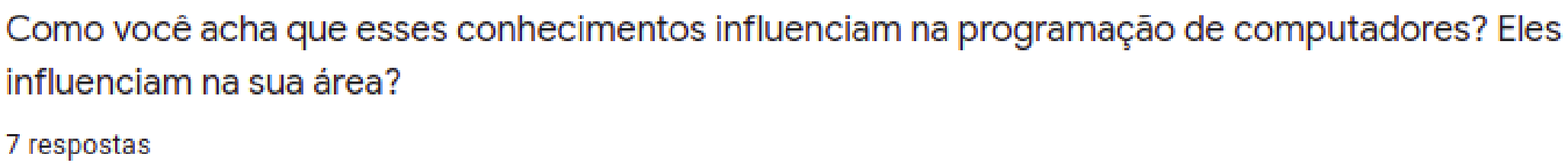
Metodologia

1) **Arquitetura de Computadores:**
   - Linguagem de Montagem específica.
   - Pensamento crítico.
   - "Programar melhor mesmo sem escrever uma linha de código em *Assembly*".
2) **Engenharia de Computação**:
   - Projeto de Sistemas Computacionais.
   - Relação *Hardware – Firmware.*
   - Entender o funcionamento e as capacidades do microprocessador.
3) **Modelo de Ensino:**
   - Conteúdos mais importantes da arquitetura em questão.
   - Levar o programador ou engenheiro a um grau de compreensão elevado.

Resultados

- Dados da pesquisa com os alunos do 3° semestre de Engenharia de Computação sobre a utilidade de *Assembly*:
  - A programação conjunta em C e *Assembly* para SE e desktop é viável.
  - Visão mais ampla e otimizada de programação.
  - Especificar melhor arquitetura da plataforma.





Resultados

- Para construir um programa em *Assembly* → Pensar fora da caixa.
- Modelo de Ensino → Curso de *Assembly* para a Arquitetura x86.
  - Estrutura Generalizada.
  - Curso dividido em três partes.
  - Usar C para melhorar o entendimento → Forte relação entre C e *Assembly*.

Modelo de Curso de Assembly para a Arquitetura x86

- **Introdução** → Motivação do estudo.
  - Definição da linguagem Assembly e sua história.
  - Funcionamento do compilador e montador.
- **Noções de Organização de Computadores** → Elementos do computador.
  - Organização de um sistema computacional.
  - Funcionamento de um sistema digital.
- **Arquitetura x86** → As bases da arquitetura em estudo.
  - Conjunto de instruções e o *Assembly* x86.
  - Pipeline de execução.
  - Banco de registradores.
  - Modos de processamento.
  - Organização de memória.
  - Primeiro programa em *Assembly*.

Modelo de Curso de Assembly para a Arquitetura x86

- **Movimentação de Dados** → Como operar a memória da arquitetura.
  - Instruções que operam a memória.
  - Endereçamento e Segmentação de memória.
- **Aritmética e *Flags*** → Entender o funcionamento da ULA.
  - *Flags* do processador.
  - Operações aritméticas e *bitwise*.
- **Controle de Fluxo** → Como estruturas condicionais operam no baixo nível.
  - Instruções de Comparação.
  - Instruções de Saltos Condicionais.
- **Procedimentos** → Funções em *Assembly*.
  - Pilha de execução.
  - Escopo de variáveis.
  - Bibliotecas externas.

Modelo de Curso de Assembly para a Arquitetura x86

- **Interface com Linguagem C** → Programar em C em conjunto com *Assembly*.
  - Incorporar o Assembly ao C.
- **Interrupções** → Entender e usar o mecanismo de entrada e saída.
  - Interrupções de *software*, *hardware* e exceções.
  - Interface de entrada e saída.
- **Conjuntos de Instruções Estendidos** → Para valores em ponto flutuante e vetoriais.
  - Instruções SSE e AVX.
- **Programação *Bare Metal*** → Aplicar os conhecimentos construindo um *firmware*.
  - Programar diretamente sobre a *BIOS*.
  - Utilização de interrupções para interação com dispositivos.

Conclusão:

- Detalhes de baixo nível e obter uma visão crítica!
- Esse trabalho ressaltou as vantagens no aprendizado do Assembly.
  - Desenvolvimento $\longleftrightarrow$ Ferramenta de Aprendizado.
  - Aplicações além do aprendizado $\rightarrow$ Assembly oferece maior controle.
- Dúvida popular: O Assembly está morto?
  - Todo código em alto nível irá passar pelo Assembly.
  - Crescimento do estudo de computadores $\rightarrow$ Devemos conhecê-los.
- Portanto, todo programador deve ter noções em Assembly.

Conclusão:

"*Conhecer o campo de batalha é antecipar o movimento do inimigo. Conhecer o inimigo é antecipar a vitória*", Sun Tzu, "Arte da Guerra".

Obrigado a todos pela atenção!

Referências Bibliográficas:

de Andrade, E. (2014). História da computação: Um pouco de assembly.

Faquer, C. (2017). Importância da linguagem de programação de baixo ní vel.

Jorgensen, E. (2020). x86-64 Assembly Language Programming with Ubuntu.

Silva, L. F. (2021). Aprendendo Assembly.

Vidal, V. (2017). Vale a pena aprender o bom e velho assembly?

Dandamudi, S. P. (2005) Guide to Assembly Language Programming in Linux, Springer.

Zhirkov, I. (2017) Low-Level Programming C, Assembly and Program Execution on Intel 64 Architecture, Apress.