SATOLANG.btc

LINGUAGEM DE PROGRAMAÇÃO

VIRTUAL MACHINE

LUCAS IKAWA
CIÊNCIA DA COMPUTAÇÃO - INSPER
LINGUAGENS E PARADIGMAS - 4º SEMESTRE

MOTIVAÇÃO

QUAL FOI A MOTIVAÇÃO PARA A SATOLANG

A motivação para criar a SatoLang surgiu da necessidade de uma ferramenta educacional e prática para trabalhar com conceitos de blockchain e criptomoedas. Linguagens tradicionais como Python ou JavaScript, embora poderosas, são muito genéricas e verbosas para este domínio específico. SatoLang é uma DSL (Domain-Specific Language) que permite simular de forma simples e intuitiva:

01	02	03	04
Operações de Blockchain	Mineração	Transações e Mercado	estratégias de trading

O projeto é ideal para educação, permitindo que estudantes e entusiastas aprendam conceitos complexos através de código conciso e expressivo. Além disso, serve como plataforma de testes para estratégias de trading e simulações de mercado.

CARACTERÍSTICAS

TRADING LOOPS

Trading Loops são construções de controle de fluxo especializadas da SatoLang que encapsulam estratégias comuns de trading de criptomoedas. São loops com semântica específica do domínio de trading, diferenciando-se de loops tradicionais (for, while) por incorporarem lógica de mercado e estratégias de investimento.

`buy_the_dip`

Buy the Dip: Comprar quando preço cai

`take_profit_until`

Realizar lucro ao atingir meta

`hodl_until`

HODL: segurar independente de variação

`scalp_for`

Scalping: Operações rápidas de curto prazo

SINTAXE NATURAL PARA BLOCKCHAIN

SatoLang usa operadores e comandos que mapeiam diretamente para ações do mundo real de criptomoedas, tornando o código tão legível quanto uma descrição em linguagem natural de operações blockchain.

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet alice

wallet bob

alice mine

alice mine

bob mine

alice → bob : 30

if saldo alice > 50 {

    alice mine

}

battle alice vs bob
```

```
Gênesis: Satoshi criou uma blockchain com supply de 21 milhões,

reward de 50 por bloco e preço inicial de 1000 dólares.

Crie uma wallet chamada Alice.

Crie uma wallet chamada Bob.

Alice minera um bloco.

Alice minera outro bloco.

Bob minera um bloco.

Alice transfere 30 moedas para Bob.

Se o saldo de Alice for maior que 50:

    Alice minera um bloco.

Compare Alice versus Bob em uma batalha.
```

SISTEMA DE MERCADO DINÂMICO

SatoLang possui um sistema de mercado integrado que simula a volatilidade de preços de criptomoedas em tempo real. O comando market update aplica uma variação aleatória de ±5% no preço atual, permitindo testar estratégias de trading em condições realistas de mercado. Isso torna a linguagem útil para simular e validar algoritmos de trading antes de implementá-los em ambiente real.

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet trader

trader mine

// Estratégia: minerar apenas quando o preço está baixo

market                    // Exibe preço inicial: $1000

if market < 1050 {

    trader mine           // Preço está favorável

}

market update             // Preço varia: pode ir para $950-$1050
```

```
if market > 1100 {

    trader mine           // Alta - minerar vale mais a pena

}

// Loop que aguarda o mercado subir

hodl_until 1200 {

    market update         // Atualiza até atingir $1200+

}

market                    // Exibe preço final após a alta
```

VM CUSTOMIZADA COM BLOCKCHAIN NATIVA

A SatoLang utiliza uma Virtual Machine própria com blockchain integrado nativamente na arquitetura, diferente de linguagens tradicionais que precisam de bibliotecas externas para simular blockchain. A VM é implementada em C com modelo de registradores (R0, R1, R2, PC, SP) e possui áreas de memória especializadas:

- Blockchain (armazena blocos minerados com hash, nonce, timestamps),
- Wallets (256 carteiras com saldo e identificação),
- Market (preço dinâmico com volatilidade), e Stack (1024 posições para controle de fluxo).

Cada operação blockchain (`MINE`, `TRANSFER`, `BATTLE`) é uma instrução Assembly nativa, executada diretamente pela VM sem overhead de interpretação adicional.
O compilador traduz código SatoLang em 30+ instruções Assembly específicas para blockchain, incluindo operações únicas como `MARKET_UPDATE` (simula volatilidade ±5%), `SHOW_CHAIN` (exibe toda a blockchain), e `WALLET_CREATE` (aloca carteira na memória da VM).

CURIOSIDADES

CURIOZIDADES DA LINGUAGEM SATOLANG

- Nome inspirado em Satoshi Nakamoto, o pseudônimo do criador do Bitcoin - toda blockchain SatoLang inicia com genesis satoshi

- Extensão de arquivo .btc (Bitcoin) - os programas são literalmente "código Bitcoin"

- 21 milhões é o limite padrão - referência direta ao supply máximo do Bitcoin real (21.000.000 BTC)
- Halving nativo - a VM implementa redução automática de recompensas de mineração a cada 210 blocos, simulando o evento real do Bitcoin

- Operador -> para transferências - sintaxe visual que representa o fluxo de moedas entre carteiras (alice -> bob : 50)

- "Battle" como mecanismo de consenso alternativo - ao invés de Proof-of-Work tradicional, carteiras podem "batalhar" com resultado baseado em saldo

CURIOZIDADES DA LINGUAGEM SATOLANG

- Trading loops com nomes de estratégias reais - buy_the_dip, hodl_until, take_profit_until são termos autênticos do mercado de criptomoedas

- Palavra-chave hodl - referência ao famoso typo "HODL" (Hold On for Dear Life) que virou meme na comunidade crypto

- VM sem garbage collector - modelo simplificado onde wallets e blockchain ocupam memória estática (256 wallets, 1000 blocos)

- Compilação em 3 estágios - btc_parser (valida sintaxe) → btc_compiler (gera .asm) → btc_vm (executa Assembly)

- Hash SHA-256 simulado - a VM usa hash simplificado baseado em nonce para demonstrar o conceito de mineração sem overhead criptográfico real

CURIOZIDADES DA LINGUAGEM SATOLANG

- Preço de mercado volátil por design - market update sempre varia ±5%, nunca fica estático
- Scope global obrigatório - não existe escopo local, refletindo a natureza pública e distribuída da blockchain

- Linguagem sem tipos explícitos - apenas identifiers (wallets) e numbers (valores), seguindo a filosofia "simples como Bitcoin"

- Comentários com // - único símbolo não relacionado a blockchain na sintaxe, mantendo familiaridade com C/C++

EXEMPLOS

HELLO BLOCKCHAIN

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet alice

alice mine

showchain
```

TRANSFERÊNCIA SIMPLES

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet alice
wallet bob

alice mine

alice → bob : 30
```

CONDICIONAL DE MERCADO

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet trader

if market < 1100 {
    trader mine
}
```

LOOP DE MINERAÇÃO

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet miner

for 5 {
    miner mine
}
```

TRADING LOOP (HOLD)

```
genesis satoshi supply 21000000 reward 50 start_price 1000

wallet investor

hodl_until 2000 {
    market update
}
```

BATTLE

```
genesis satoshi supply 21000000 reward 50 start_price 1000
wallet alice
wallet bob

alice mine
bob mine

battle alice vs bob
```

PROGRAMA COMPLETO

```
genesis satoshi supply 21000000 reward 50 start_price 1000
wallet alice
wallet bob
alice mine
alice mine
alice → bob : 25
if saldo bob > 20 {
    bob mine
}
market update
battle alice vs bob
```