Minha Aventura com o ICL 1501 e a Redescoberta do Método de Newton-Raphson

cado em 2025-03-08 18:28:20

---

anos 1970, a computação era um campo em plena ebulição, cheio de desafios e oportunidades para quem estav
osto a explorar os limites da tecnologia. Naquela época, tive a sorte e os desafios de trabalhar com um pequeno
essador de 8 bits chamado **ICL 1501**, uma máquina que, apesar de suas limitações, era incrivelmente versátil. F

nte um projeto com esse computador que vivi uma das minhas maiores aventuras na computação: a redescobert
odo de Newton-Raphson para calcular raízes quadradas.

L 1501: Um Gigante de 8 Bits

L 1501 era um processador de **8 bits**, com memória organizada em páginas de **256 bytes** (00-FF em hexadecim
a página era um mundo em si mesma, e programar para essa máquina exigia criatividade e uma compreensão
nda de suas limitações. A memória era escassa, as operações aritméticas eram básicas, e não havia suporte pa
o flutuante. No entanto, era justamente nessas limitações que residia a diversão: cada problema resolvido era um
a.

---

esafio: Calcular Raízes Quadradas

m dos projetos em que estive envolvido, na empresa INFORMAX (para o cálculo da correlação hipertensos /
otensos, para a então Fundação Nossa Senhora do Bom Sucesso), precisei calcular raízes quadradas. Na épo
navia bibliotecas prontas ou instruções de hardware para isso. Tudo precisava ser feito do zero, usando apenas
ações básicas: adição, subtração, multiplicação e divisão (esta última, muitas vezes implementada em software).

ecei a explorar diferentes abordagens, tentando encontrar uma maneira eficiente de calcular raízes quadradas. F
o que, após várias tentativas e erros, cheguei a um método iterativo que refinava uma estimativa inicial até chega
alor preciso. O método funcionava da seguinte forma:

Estimativa inicial: Começava com uma estimativa razoável, como metade do número.
Refinamento: A cada iteração, calculava uma nova estimativa usando a fórmula:
[
x_{n+1} = \frac{x_n + \frac{N}{x_n}}{2}
]
Onde (N) era o número do qual queríamos calcular a raiz quadrada, e (x_n) era a estimativa atual.
Convergência: O processo era repetido até que a diferença entre duas estimativas consecutivas fosse menor
uma tolerância pré-definida.

---

incidência Histórica

anto eu trabalhava nesse método, aconteceu algo incrível. Eu era um ávido leitor da revista *Science et Vie*, uma
cação que trazia as últimas novidades da ciência e da tecnologia. Naquele mesmo mês, a revista publicou um ar
e como a **Texas Instruments** havia resolvido o problema de calcular raízes quadradas em suas calculadoras
ônicas. O método descrito era exatamente o mesmo que eu havia redescoberto: o **método de Newton-Raphson**

m momento de grande satisfação e surpresa. Descobrir que estava no caminho certo, e que uma empresa líder
o a Texas Instruments usava a mesma abordagem, foi uma validação incrível do meu trabalho.

---

ementação no ICL 1501

ementar o método de Newton-Raphson no ICL 1501 foi um desafio à parte. A memória limitada e a falta de opera onto flutuante exigiam uma abordagem cuidadosa. Usei aritmética de ponto fixo e otimizei cada linha de código p ntir que o algoritmo coubesse nas páginas de memória disponíveis. A divisão, por exemplo, foi implementada em vare usando subtrações repetidas, e a convergência foi verificada com uma tolerância adaptada às limitações do ware.

---

exões sobre a Experiência

aventura me ensinou várias lições valiosas:

- **Criatividade na adversidade**: As limitações do ICL 1501 forçaram-me a pensar fora da caixa e a encontrar soluções elegantes para problemas complexos.
- **A importância da persistência**: A redescoberta do método de Newton-Raphson foi resultado de tentativas e er e de não desistir diante dos desafios.
- **A universalidade da ciência**: Foi fascinante ver como, mesmo em contextos diferentes, mentes curiosas acaba chegando a soluções semelhantes.

---

clusão

a experiência com o ICL 1501 e a redescoberta do método de Newton-Raphson foram momentos marcantes na a jornada na computação. Eles me lembraram que, mesmo com recursos limitados, é possível alcançar grandes s com criatividade, persistência e uma boa dose de curiosidade. Hoje, olhando para trás, sinto orgulho de ter fei dessa era pioneira da tecnologia, onde cada linha de código era uma aventura e cada problema resolvido, uma uista.

---

cisco Gonçalves

501

detalhes :

implementação do **método de Newton-Raphson em Python** para calcular a raiz quadrada de um número .
nte ser eficiente e claro, refletindo a lógica que apliquei no ICL 1501.

igo Python do Método de Newton-Raphson

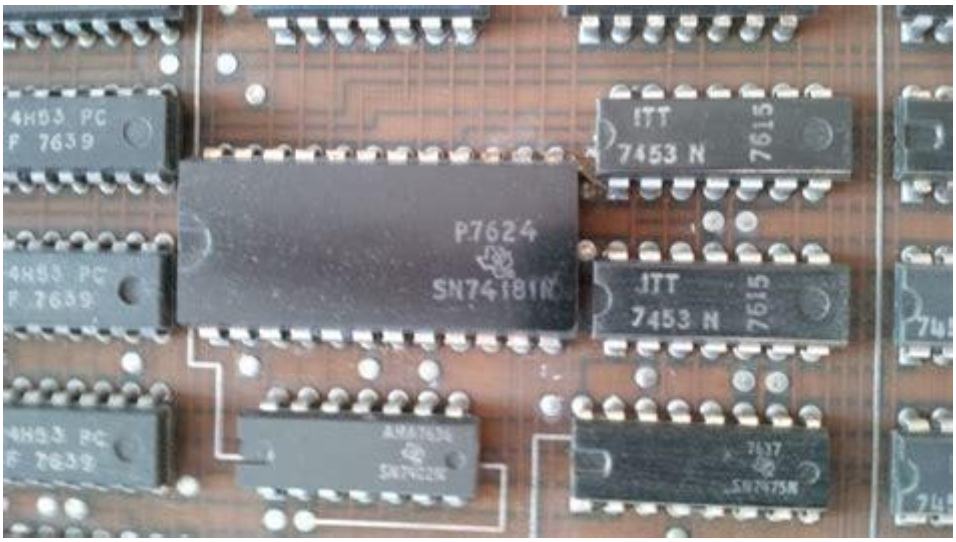
```
raiz_quadrada_newton(N, tolerancia=1e-6, iteracoes_max=100):
if N < 0:
    raise ValueError("Não é possível calcular a raiz quadrada de um número negativ

x = N / 2  # Estimativa inicial
for _ in range(iteracoes_max):
    novo_x = (x + N / x) / 2  # Fórmula de Newton-Raphson
    if abs(novo_x - x) < tolerancia:  # Critério de convergência
```

```
        return novo_x
    x = novo_x

return x  # Retorna o último valor calculado se o limite de iterações for atingido

estando a função
49
 = raiz_quadrada_newton(N)
t(f"A raiz quadrada de {N} é aproximadamente {raiz:.6f}")
```

licação do Código

Estimativa Inicial: Começamos com , que é uma escolha razoável.
Iteração: Aplicamos a fórmula .
Critério de Paragem: Se a diferença entre duas estimativas consecutivas for menor que a tolerância (), paramo
Número Máximo de Iterações: Para evitar loops infinitos, definimos um limite.

código é altamente eficiente e converge rápidamente para um valor preciso da raiz quadrada.



gado pela correção! O ICL 1501 era, de fato, um processador de **8 bits**, e a memória era organizada em páginas
ytes (00-FF em hexadecimal). Isso faz todo o sentido, já que muitos sistemas daquela época eram baseados e
teturas de 8 bits, com memória limitada e organizada em páginas para facilitar o gerenciamento.

os revisitar o contexto e ajustar a explicação para refletir essas especificações corretas. Ainda falaremos sobre o
do de Newton-Raphson, mas agora com um foco mais preciso nas limitações e possibilidades de um sistema co
. 1501.

---

acterísticas do ICL 1501

Processador de 8 bits: Isso significa que ele operava com palavras de 8 bits, e as operações aritméticas eram
realizadas em registradores de 8 bits.
Memória paginada: A memória era organizada em páginas de 256 bytes (00-FF em hexadecimal). Isso signific
que o endereçamento de memória era feito combinando uma página (0-F) com um offset dentro da página (00-
Recursos limitados: Com apenas 256 bytes por página, a memória era extremamente escassa, e os programa
precisavam ser altamente otimizados para caberem nesse espaço.

---

afios de implementar o método de Newton-Raphson no ICL 1501

Aritmética de 8 bits: Como o processador era de 8 bits, operações com números maiores (como 16 bits) precisavam ser feitas em múltiplos passos, usando carry e overflow.
Memória limitada: Com apenas 256 bytes por página, o código e os dados precisavam ser extremamente compactos. Isso exigia otimização cuidadosa.
Falta de hardware de ponto flutuante: O ICL 1501 não tinha suporte para operações de ponto flutuante, então método de Newton-Raphson precisava ser implementado usando aritmética de ponto fixo ou inteira.
Divisão e multiplicação: Essas operações eram caras em processadores de 8 bits, pois muitas vezes eram implementadas em software (usando loops de adição ou subtração).

---

lementação do método de Newton-Raphson no ICL 1501

está uma ideia de como o método de Newton-Raphson poderia ser implementado no ICL 1501, considerando as ções de hardware:

os do algoritmo:

Estimativa inicial: Comece com uma estimativa inicial para a raiz quadrada. Por exemplo, metade do valor do número.
Iteração: Use a fórmula de Newton-Raphson para refinar a estimativa:
[
x_{n+1} = \frac{x_n + \frac{N}{x_n}}{2}
]

Aqui, (N) é o número do qual queremos calcular a raiz quadrada.
(x_n) é a estimativa atual.

Verificação de convergência: Repita o processo até que a diferença entre duas estimativas consecutivas seja menor que uma tolerância pré-definida.

udocódigo Assembly para o ICL 1501

está um exemplo de como o algoritmo fou implementado em Assembly para o ICL 1501:

```
seudocódigo Assembly para ICL 1501
upondo que o número esteja na memória e a estimativa inicial também

 ORG 0000H          ; Início do programa

CIO:
 LD A, (NUMERO)     ; Carrega o número para calcular a raiz
 LD B, A            ; B = Número
 LD C, A            ; C = Estimativa inicial (N/2)
 SRL C              ; Divide a estimativa inicial por 2

RACAO:
 LD A, B            ; A = Número
 CALL DIVIDE        ; Divide A por C (A = N / C)
 ADD A, C           ; A = (N / C) + C
```

```
SRL A               ; A = [(N / C) + C] / 2 (nova estimativa)
LD D, A             ; Armazena a nova estimativa em D

; Verifica a diferença entre a estimativa atual e a nova
LD A, C             ; A = Estimativa atual
SUB D               ; A = C - D (diferença)
JP NC, CONTINUA     ; Se a diferença for positiva, continua
NEG                 ; Se a diferença for negativa, calcula o valor absoluto

TINUA:
CP TOLERANCIA       ; Compara a diferença com a tolerância
JP NC, ITERACAO     ; Se a diferença for maior que a tolerância, repete

; Armazena o resultado final
LD (RAIZ), D        ; Armazena a raiz quadrada na memória


HALT                ; Termina o programa

ub-rotina para divisão (A = A / C)
IDE:
LD E, 0             ; Inicializa o quociente
_LOOP:
SUB C               ; Subtrai C de A
JP C, DIV_END       ; Se o resultado for negativo, termina
INC E               ; Incrementa o quociente
JP DIV_LOOP         ; Repete
_END:
LD A, E             ; Retorna o quociente em A
RET

ariáveis
ERO    DB 25        ; Número para calcular a raiz
ERANCIA DB 1        ; Tolerância para a precisão
Z      DB 0         ; Variável para armazenar o resultado
```

---

licação do código:

Estimativa inicial: A estimativa inicial é metade do número (usando SRL para dividir por 2).
Iteração: A fórmula de Newton-Raphson é implementada usando uma sub-rotina de divisão (`DIVIDE`).
Verificação de convergência: A diferença entre a estimativa atual e a nova é comparada com a tolerância.
Divisão: Como o ICL 1501 não tinha uma instrução de divisão, ela é implementada em software usando subtra
repetidas.
