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

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

Nos anos 1970, a computação era um campo em plena ebulição, cheio de desafios e oportunidades para quem estava disposto a explorar os limites da tecnologia.

vivi uma das minhas maiores aventuras na computação: a redescoberta do **método de Newton-Raphson** para calcular raízes quadradas.

O ICL 1501: Um Gigante de 8 Bits

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

O Desafio: Calcular Raízes Quadradas

Em um dos projetos em que estive envolvido, na empresa INFORMAX (para o cálculo da correlação hipertensos / normotensos, para a então Fundação Nossa Senhora do Bom Sucesso), precisei calcular raízes quadradas. Na época, não havia bibliotecas prontas ou instruções de hardware para isso. Tudo precisava ser feito do zero, usando apenas as operações básicas: adição, subtração,

encontrar uma maneira eficiente de calcular raízes quadradas. Foi então que, após várias tentativas e erros, cheguei a um método iterativo que refinava uma estimativa inicial até chegar a um valor preciso. O método funcionava da seguinte forma:

1. **Estimativa inicial**: Começava com uma estimativa razoável, como metade do número.
2. **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.
3. **Convergência**: O processo era repetido até que a diferença entre duas estimativas consecutivas fosse menor que uma tolerância pré-definida.

A Coincidência Histórica

Enquanto eu trabalhava nesse método, aconteceu algo incrível. Eu era um ávido leitor da revista Science et Vie, uma publicação que trazia as últimas novidades da ciência e da tecnologia. Naquele mesmo mês, a revista publicou um artigo sobre como a **Texas Instruments** havia resolvido o problema de calcular raízes quadradas em suas

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

Implementação no ICL 1501

Implementar o método de Newton-Raphson no ICL 1501 foi um desafio à parte. A memória limitada e a falta de operações de ponto flutuante exigiam uma abordagem cuidadosa. Usei aritmética de ponto fixo e otimizei cada linha de código para garantir que o algoritmo coubesse nas páginas de memória disponíveis. A divisão, por exemplo,

Reflexões sobre a Experiência

Essa aventura me ensinou várias lições valiosas:

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

Conclusão

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

Mais detalhes :

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

Código Python do Método de Newton-Raphson

```
def raiz_quadrada_newton(N, tolerancia=1e-6, iteracoes_max=10
    if N < 0:
        raise ValueError("Não é possível calcular a raiz quad

    x = N / 2  # Estimativa inicial
```

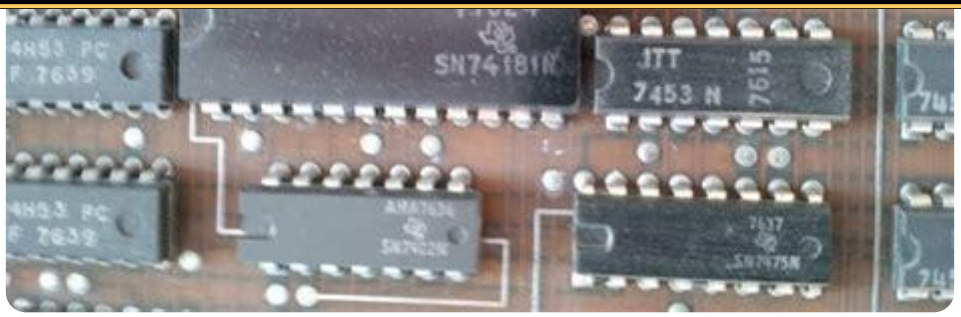
Raphson
e conve

```
        x = novo_x

    return x  # Retorna o último valor calculado se o limite

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

Explicação do Código

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

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



Obrigado pela correção! O ICL 1501 era, de fato, um processador de **8 bits**, e a memória era organizada em páginas de **256 bytes** (00-FF em hexadecimal). Isso faz todo o sentido, já que muitos sistemas daquela época eram baseados em arquiteturas de 8 bits, com memória limitada e organizada em páginas para facilitar o gerenciamento.

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

Características 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 significa que o endereçamento de memória era feito

programas precisavam ser altamente otimizados para caberem nesse espaço.

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

1. **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.
2. **Memória limitada**: Com apenas 256 bytes por página, o código e os dados precisavam ser extremamente compactos. Isso exigia otimização cuidadosa.
3. **Falta de hardware de ponto flutuante**: O ICL 1501 não tinha suporte para operações de ponto flutuante, então o método de Newton-Raphson precisava ser implementado usando aritmética de ponto fixo ou inteira.
4. **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).

Raphson poderia ser implementado no ICL 1501, considerando as limitações de hardware:

Passos do algoritmo:

1. **Estimativa inicial**: Comece com uma estimativa inicial para a raiz quadrada. Por exemplo, metade do valor do número.
2. **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.

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

Pseudocódigo Assembly para o ICL 1501

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

```
; Pseudocódigo Assembly para ICL 1501
; Supondo que o número esteja na memória e a estimativa inici
```

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

ITERACAO:
    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 estimati
    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, continu
    NEG                 ; Se a diferença for negativa, calcula

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

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

FIM:
    HALT                ; Termina o programa
```

```
    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
DIV_END:
    LD A, E             ; Retorna o quociente em A
    RET

; Variáveis
NUMERO    DB 25         ; Número para calcular a raiz
TOLERANCIA DB 1         ; Tolerância para a precisão
RAIZ      DB 0          ; Variável para armazenar o resultado
```

Explicação do código:

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