Programação Genética

Síntese Automática de Programas Assembly para Microcontroladores Digitais

Douglas Mota Dias
ICA – Núcleo de Inteligência Computacional Aplicada/DEE
PUC-Rio

Sumário

- Motivação
- Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- Programação Genética
- Sistema de Síntese Automática de Programas em Linguagem de Montagem
- Estudos de Caso
- Conclusões

Motivação

- Uso de sistemas digitais para controle de processos é vantajoso
  - Dificuldades com implementação analógica podem ser contornadas:
    1. Não há problemas com precisão ou alterações dos componentes
    2. É mais simples ter cálculos sofisticados na lei de controle e incluir funções lógicas e não-lineares
- Micro Controladores (MCs) são amplamente utilizados em sistemas *embedded*
  - Sistemas onde não é possível ou não se justifica o uso de computadores

Motivação

Automação do projeto de sistemas de controle ótimo por MCs

- A PG é capaz de contornar duas dificuldades do projeto tradicional:
  1. Obtenção matemática tradicional de uma solução de controle a partir do modelo da planta
  2. Programação manual do MC em função da solução encontrada na etapa anterior
- Controladores PID, apesar de flexíveis, não fornecem uma solução de controle ótimo na maioria das vezes

Sumário

- ✓ Motivação
- ☐ Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- ☐ Programação Genética
- ☐ Sistema de Síntese Automática de Programas em Linguagem de Montagem
- ☐ Estudos de Caso
- ☐ Conclusões

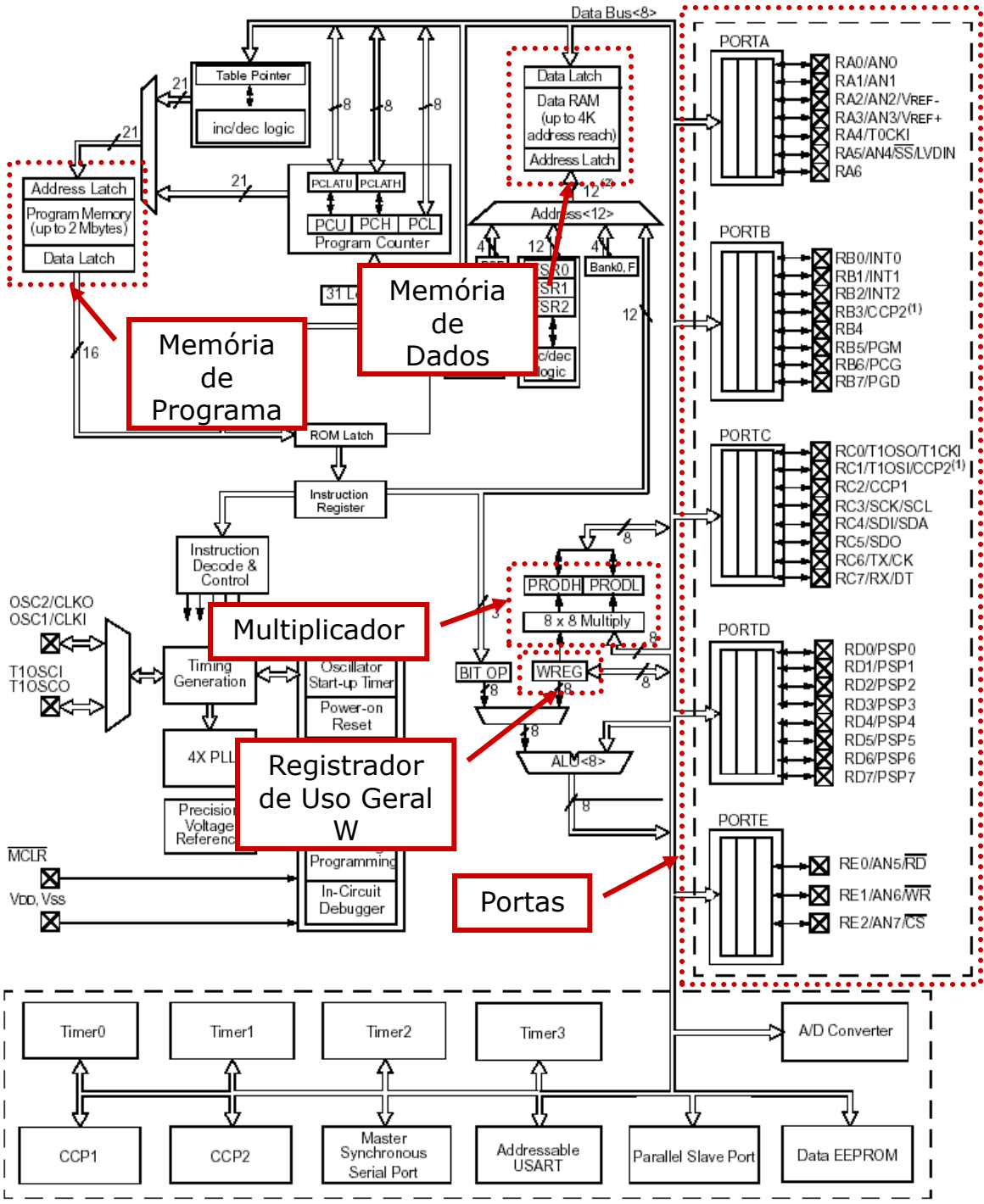
Controle de Processos por Meio de MCs

Microcontrolador

- Dispositivo que incorpora CPU, memórias e periféricos em um único CI
  - Redução do custo final do controlador
  - Redução do tamanho da placa de circuito impresso
  - Aumento da confiabilidade
- Produto dos avanços da microeletrônica
- A CPU dedicada à tarefa de controle pode ser simples
- Os microcontroladores são simples, baratos e eficientes

Controle de ...

- Arquitetura do PIC18F452



Controle de Processos por Meio de MCs

Instruções do PIC18F452

- Possui um conjunto de 75 instruções
- Exemplos:

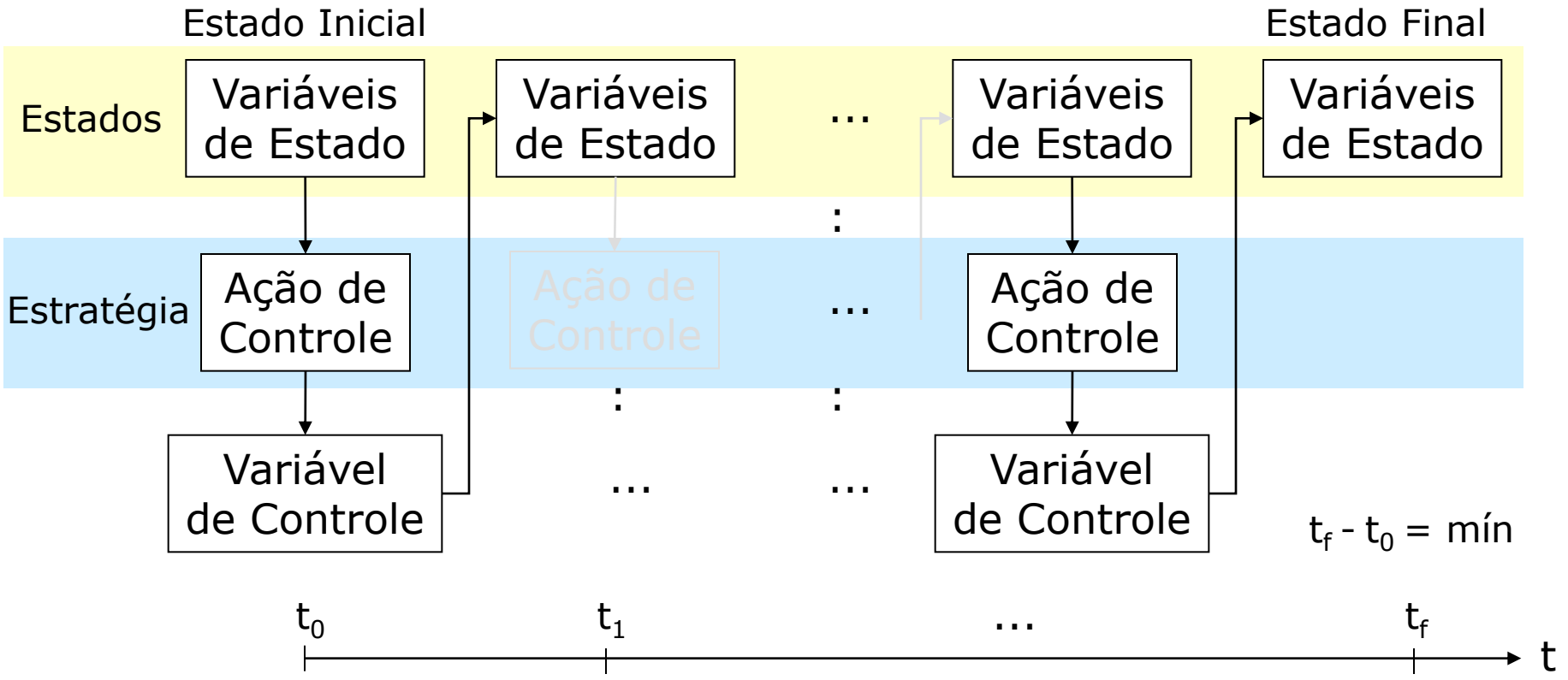
Instrução	Funcionamento
`ADDWF f, d`	Soma W e f
`CLRF f`	Atribui o valor zero a f
`CPFSEQ f`	Compara f com W, salta próx. inst. se f = W
`DECFSZ f, d`	Decrementa f, salta próx. inst. se f = 0
`MOVFF` f_s, f_d	Copia o conteúdo de f_s em f_d
`MULWF f`	Multiplica W por f
`BTFSC f, b`	Testa o bit b de f, salta próx. inst. se b = 0

- f : designa um dos registradores de uso geral da RAM
- W: acumulador na saída da ALU
- d : designa o registrador de destino do resultado (0=W, 1=f)

Controle de Processos por Meio de MCs

Controle Ótimo e Sistemas Não-Lineares

- Problema de controle ótimo:



Controle de Processos por Meio de MCs

Controle Ótimo e Sistemas Não-Lineares

- Problemas:
  - Soluções de controle normalmente implicam em funções altamente não-lineares: impossibilita uma solução matemática exata
  - Geralmente os sistemas a serem controlados são lineares somente dentro de certos limites de operação: aproximação dos modelos

Controle de Processos por Meio de MCs

Controle Ótimo e Sistemas Não-Lineares

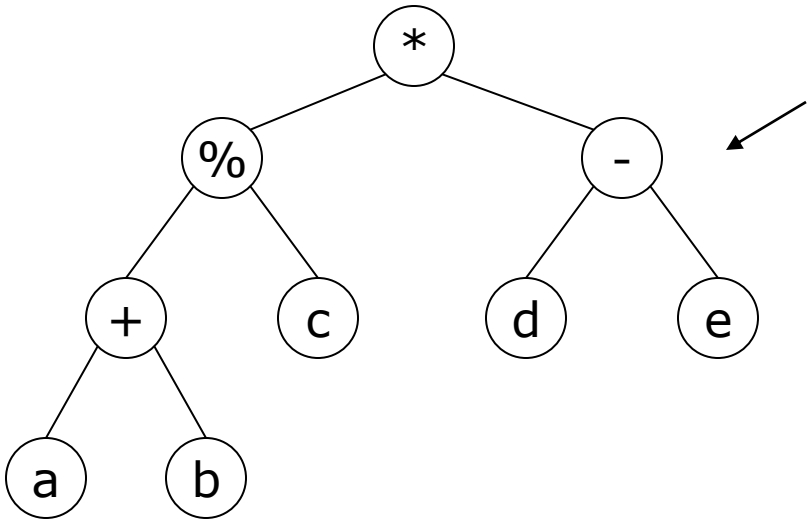
- Contribuições da PG
  - Sintetizar programas que implementem estratégias de controle ótimo, a partir apenas da modelagem por equações dinâmicas: contorna dificuldades matemáticas
  - Sintetizá-los especificamente para uma plataforma limitada em recursos - um MC: contorna dificuldades de programação

Sumário

- ✓ Motivação
- ✓ Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- ☐ Programação Genética
- ☐ Sistema de Síntese Automática de Programas em Linguagem de Montagem
- ☐ Estudos de Caso
- ☐ Conclusões

Programação Genética Tradicional

- Formalizada por John Koza (1990)
- Representação dos indivíduos por árvores
- Evolui programas apenas em linguagens funcionais (p.ex. LISP)

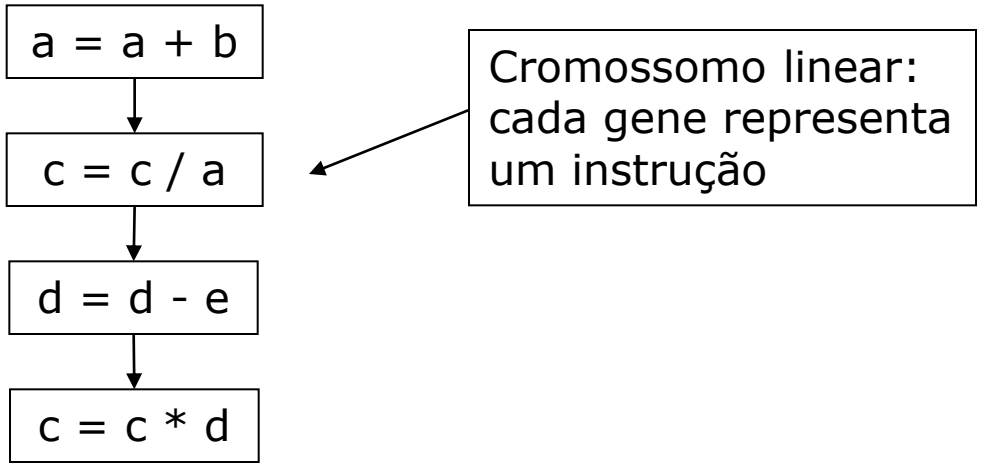


Cromossomo
do programa:

((a + b) % c) * (d - e)

Programação Genética Linear

- Representa indivíduos por uma lista linear de instruções
- Evolui programas em linguagens imperativas (p.ex. ling. assembly e C)



Programação Genética

Evolução de Programas em Linguagem Assembly

- Vantajosa quando necessita-se de soluções muito eficientes
  - Plataformas com fortes restrições de tempo de execução e de uso de memória
- Otimização mais eficiente do código
  - Nível mais baixo para se otimizar um programa
  - Onde os maiores ganhos são possíveis
- Ferramentas de alto nível podem não existir
- Linguagem de montagem é considerada difícil de se trabalhar
  - Pode ser mais fácil deixar o computador evoluir os programas

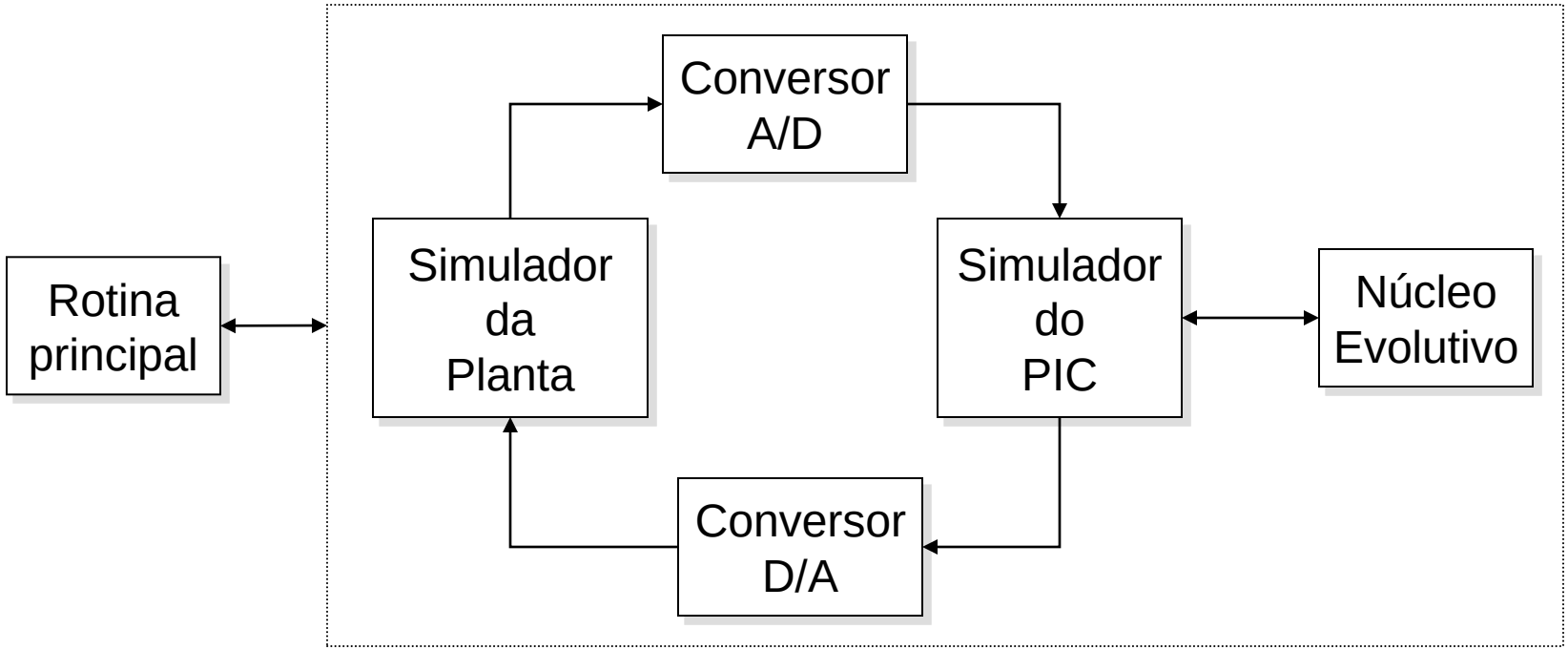
Sumário

- ✓ Motivação
- ✓ Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- ✓ Programação Genética
- ☐ Sistema de Síntese Automática de Programas em Linguagem de Montagem
- ☐ Estudos de Caso
- ☐ Conclusões

Sistema de Síntese Automática de Programas

- Sintetiza programas de controle em linguagem de montagem, a partir de um modelo matemático, por PG
- Desenvolvido para utilização nos estudos de caso
- Desenvolvido em C – melhor desempenho computacional

Diagrama em blocos do sistema:

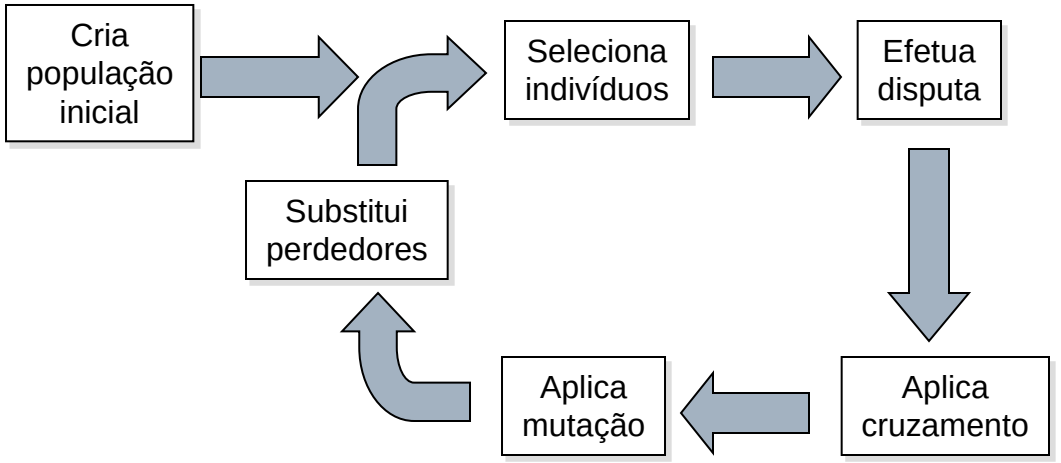


Sistema de Síntese Automática de Programas

Núcleo Evolutivo

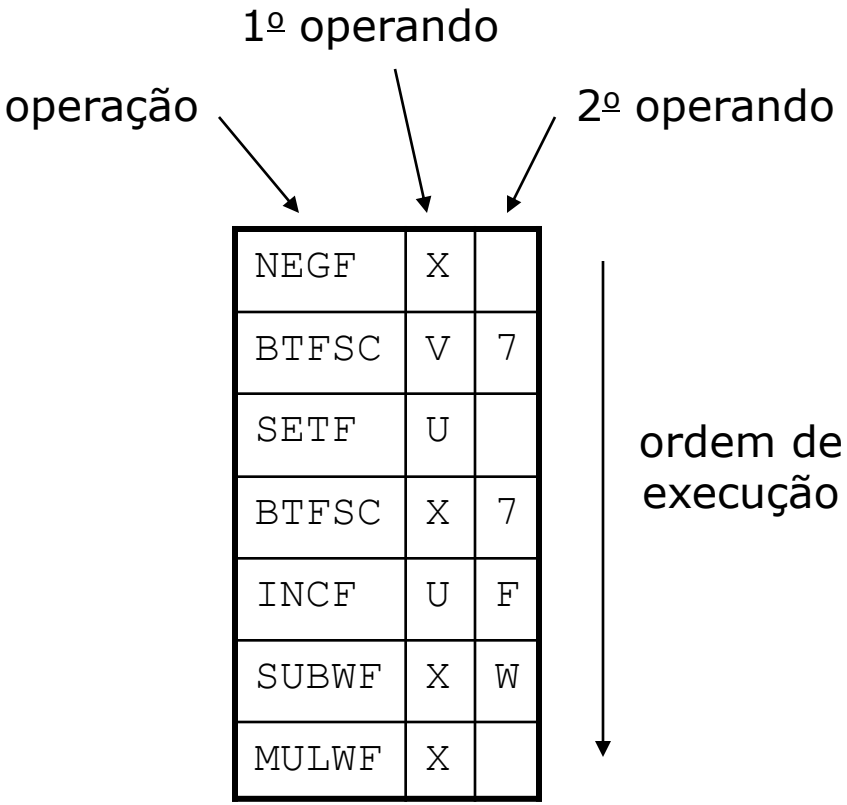
- Não existem ferramentas de PG linear
- Implementa o algoritmo de PG e os operadores genéticos
- Cria a população inicial, seleciona indivíduos e aplica os operadores genéticos

Funcionamento do núcleo evolutivo:



Núcleo Evolutivo

- Representação dos indivíduos



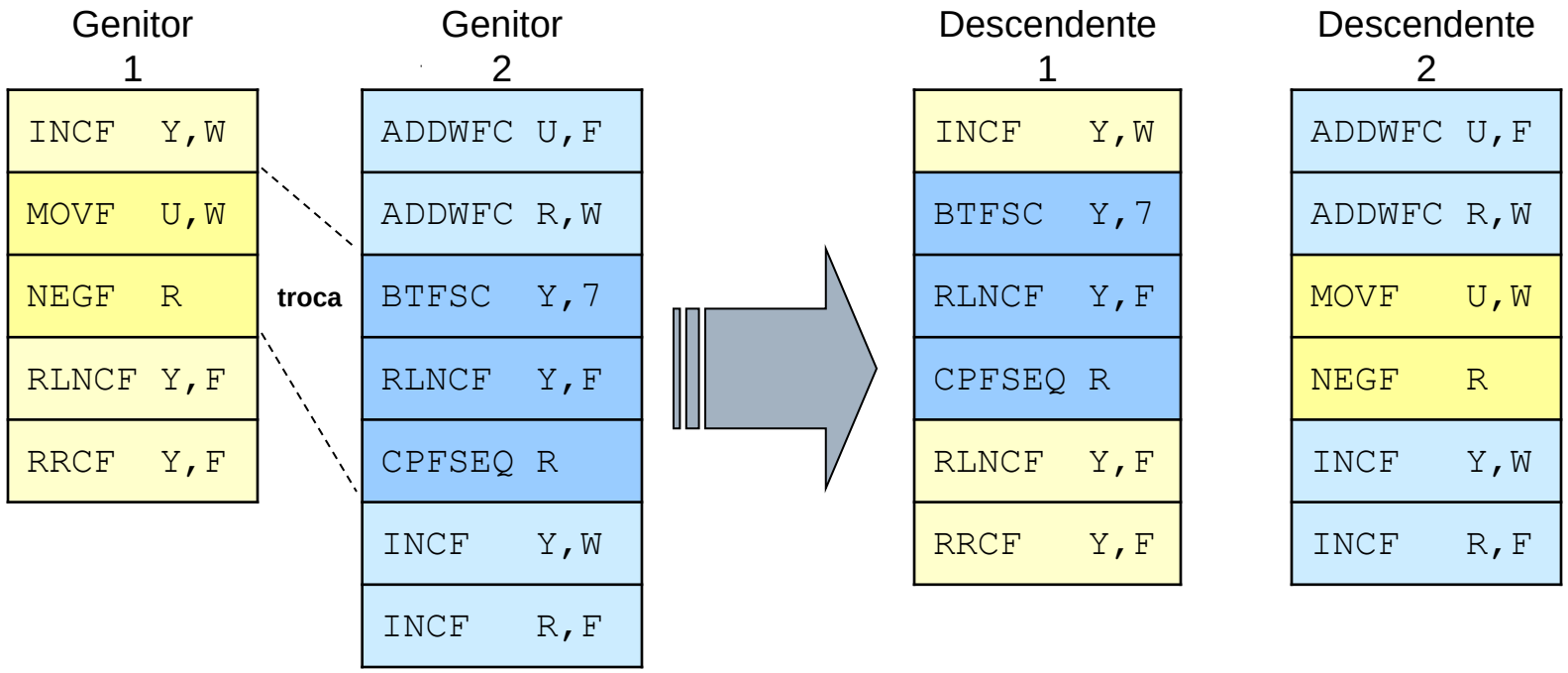
Núcleo Evolutivo

- Iniciação da população
  - Escolhe-se aleatoriamente o comprimento inicial de um indivíduo
    - comp_ini_mín < comp < comp_ini_max
  - Todos os genes são preenchidos com instruções geradas aleatoriamente
  - Processo repete-se para todos os indivíduos da população

Sistema de Síntese Automática de Programas

Núcleo Evolutivo

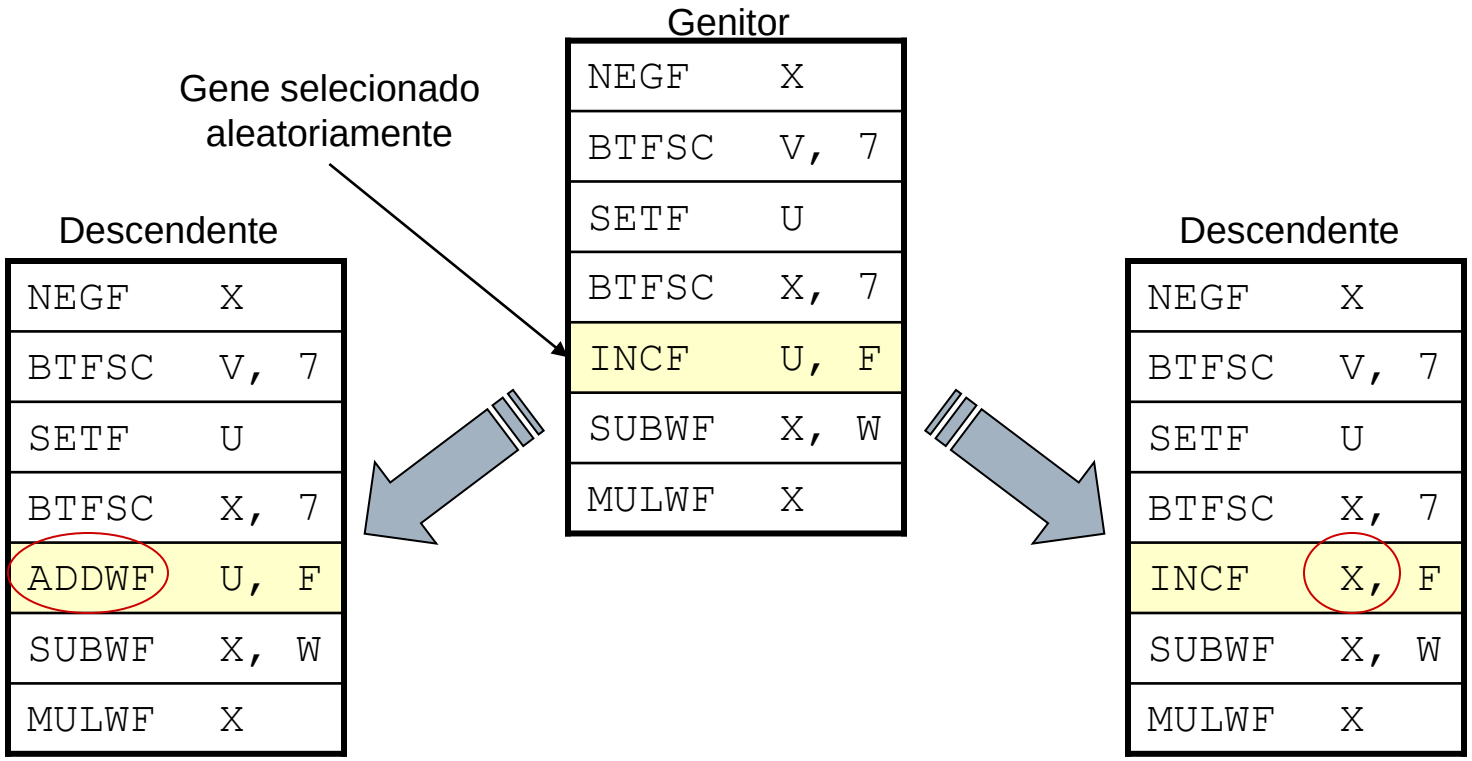
- Cruzamento
  - Comprimento dos segmentos limitado a 4 ameniza o efeito destrutivo do cruzamento



Sistema de Síntese Automática de Programas

Núcleo Evolutivo

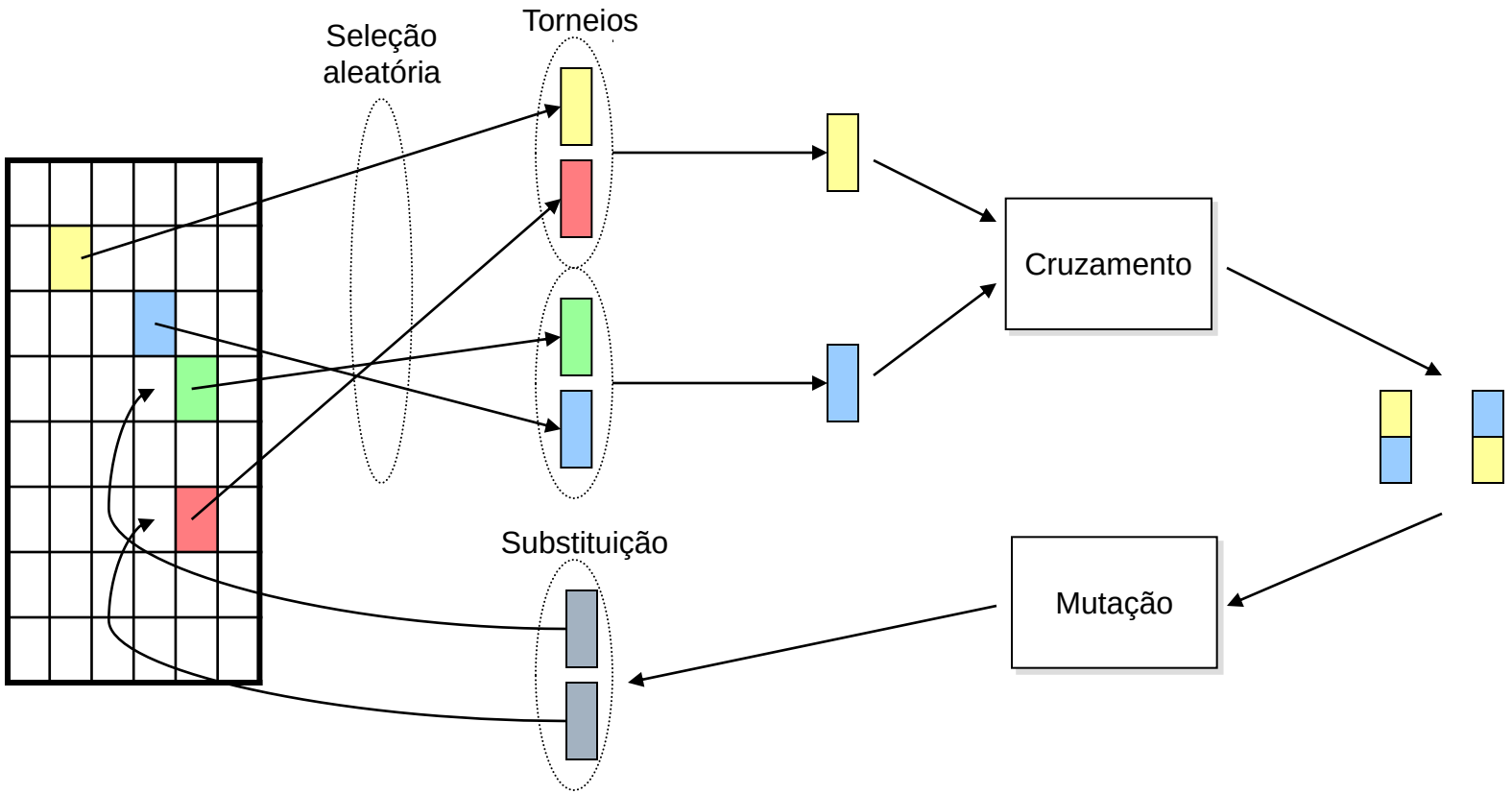
- Mutação
  - Aplicada no operador ou em um dos 2 operandos: micromutação



Sistema de Síntese Automática de Programas

Núcleo Evolutivo

- Seleção e evolução



Sistema de Síntese Automática de Programas

Núcleo Evolutivo

- Seleção
  - Algoritmo de seleção por torneios
  - Baseia-se na competição entre indivíduos de um subconjunto da população
  - Prevalecimento das características dos melhores indivíduos sobre aquelas dos piores
  - Não requer comparação centralizada de aptidões entre todos os indivíduos de uma população
    - Acelera evolução
    - Elimina *overhead* entre avaliações existente na seleção proporcional à aptidão
  - Este método vem sendo muito utilizado em PG

Sistema de Síntese Automática de Programas

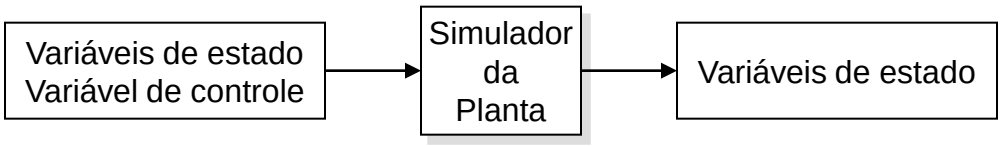
Núcleo Evolutivo

- Evolução
  - Algoritmo chamado *steady-state*
    - Baseia-se no modelo de seleção por torneio
  - Não há intervalos fixos definindo gerações
    - Há um fluxo contínuo de indivíduos encontrando-se, unindo-se e produzindo descendentes
  - Descendentes substituem indivíduos existentes na mesma população
  - Método simples de se implementar
  - Possui alguns benefícios de eficiência
  - Bibliografia relata bons resultados de convergência
  - Vem ganhando terreno entre pesquisadores

Sistema de Síntese Automática de Programas

Simulador da Planta

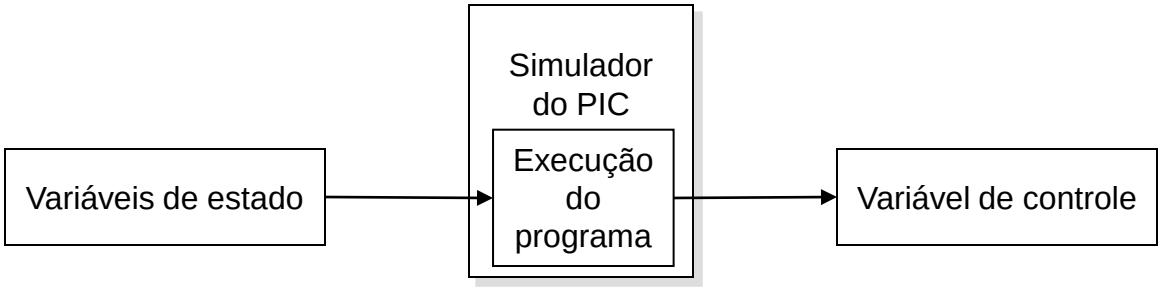
- Três, um para cada estudo de caso
- Implementado por uma função
  - Entradas
    - Variáveis de estado da planta (por referência)
    - Variável de controle da planta (por valor)
  - Saídas
    - Variáveis de estado da planta atualizadas
- Núcleo composto pelas equações dinâmicas que modelam a planta $t+1$



Sistema de Síntese Automática de Programas

Simulador de Microcontrolador PIC

- Implementado por uma função
  - Entradas
    - Variáveis de estado da planta (por valor)
  - Saída
    - Variável de controle



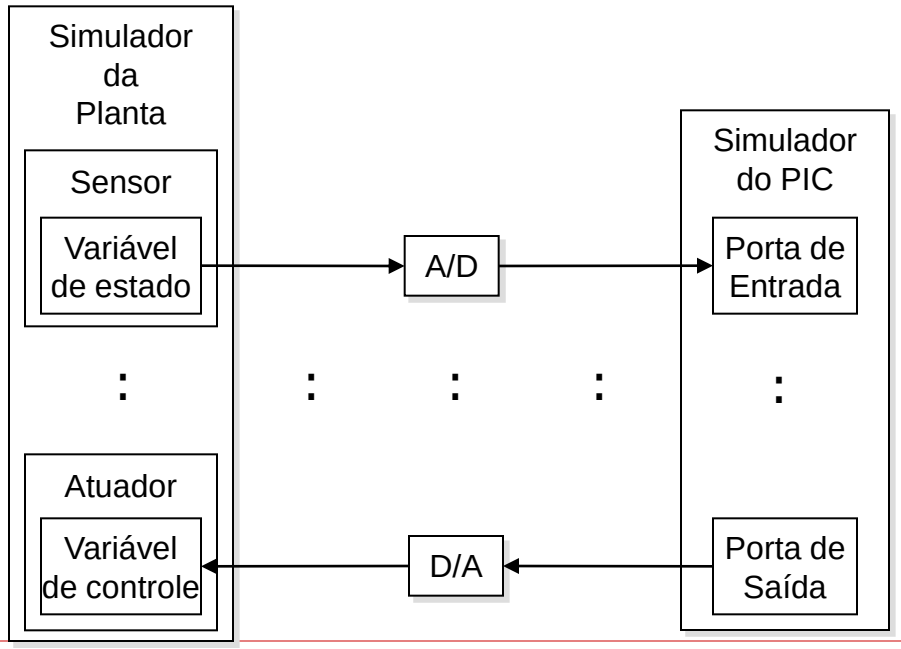
Sistema de Síntese Automática de Programas

Mnemônico e Operandos	Descrição
ADDWF f, d	Soma W e f
ADDWFC f, d	Soma W e o bit de *Carry* a f
CLRF f	Atribui o valor zero a f
CPFSEQ f	Compara f com W, salta próxima instrução se f = W
CPFSGT f	Compara f com W, salta próxima instrução se f > W
CPFSLT f	Compara f com W, salta próxima instrução se f < W
DECF f, d	Decrementa f
DECFSZ f, d	Decrementa f, salta próxima instrução se f = 0
INCF f, d	Incrementa f
MOVF f, d	Copia o conteúdo de f
MOVFF f_s, f_d	Copia o conteúdo de f_s em f_d
MOVWF f	Copia o conteúdo de W em f
MULWF f	Multiplica W por f
NEGF f	Complementa f a dois
RLCF f, d	Desloca f para a esquerda através do bit de *Carry*
RLNCF f, d	Desloca f para a esquerda sem o bit de *Carry*
RRCF f, d	Desloca f para a direita através do bit de *Carry*
RRNCF f, d	Desloca f para a direita sem o bit de *Carry*
SETF f	Atribui o valor um a todos os bits de f
SUBWF f, d	Subtrai W de f
BTFSC f, b	Testa o bit b de f, salta próxima instrução se b = 0
BTFSS f, b	Testa o bit b de f, salta próxima instrução se b = 1

Sistema de Síntese Automática de Programas

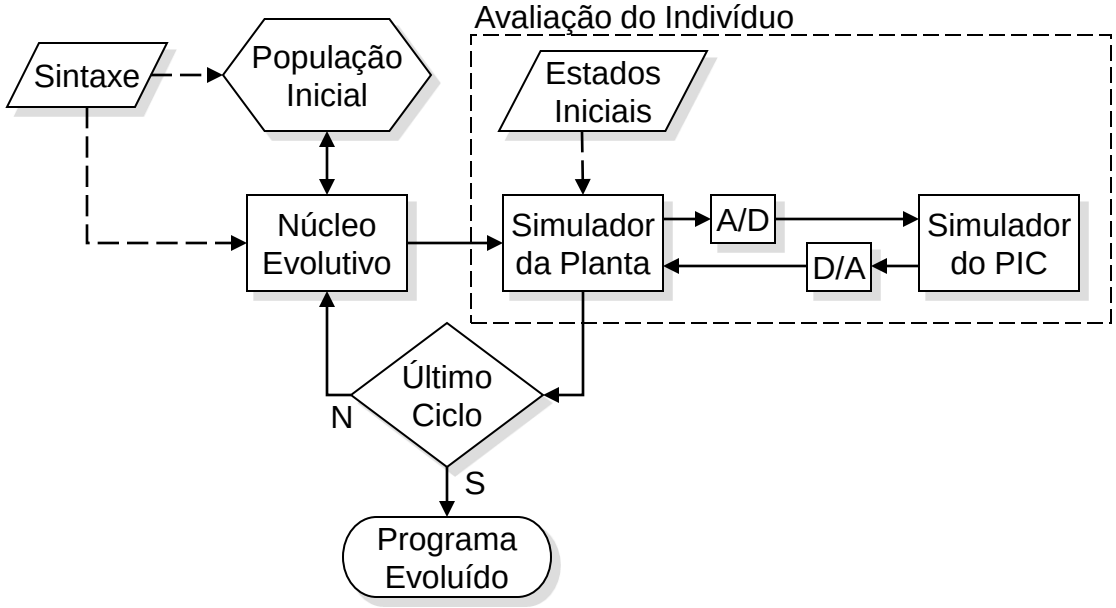
Conversores A/D e D/A

- Implementados por funções
- Estabelecem interfaces entre os valores analógicos das grandezas da planta e seus equivalentes digitais no PIC.



Sistema de Síntese Automática de Programas

Visão Geral



Sumário

- ✓ Motivação
- ✓ Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- ✓ Programação Genética
- ✓ Sistema de Síntese Automática de Programas em Linguagem de Montagem
- ☐ Estudos de Caso
- ☐ Conclusões

Estudos de Caso

1. Sistema de aquecimento de água (Waterbath)
2. Estacionamento do carro (Cart centering problem)
3. Pêndulo invertido

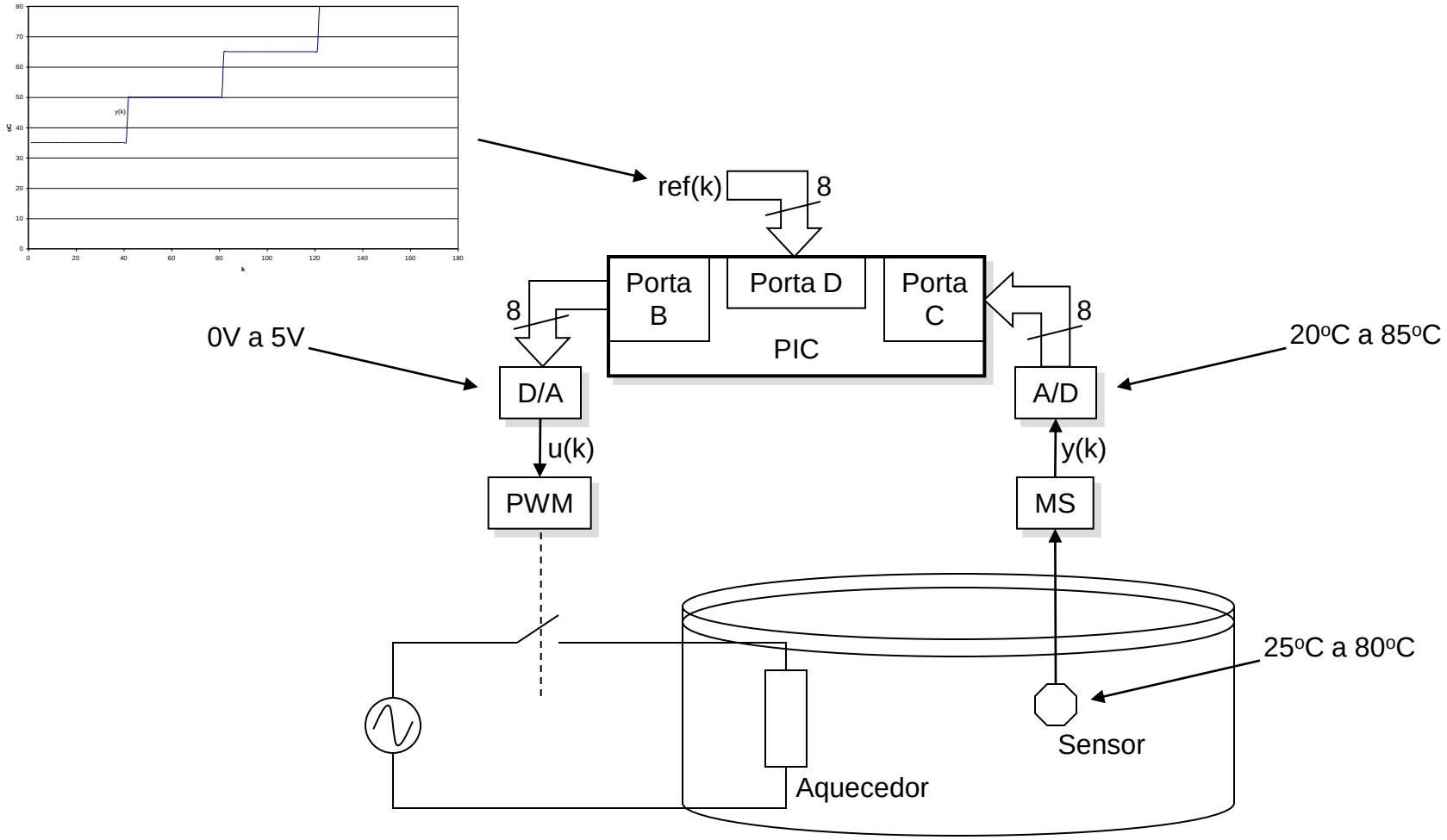
- Experimentos realizados em um PC Pentium IV de 2,8 GHz

Estudos de Caso

1. Waterbath
   - Sistema de aquecimento de água baseado em uma planta real
   - Aplicação real em processos de destilação
   - Utilizado em diversos trabalhos de inteligência computacional
   - Objetivos principais
     - Avaliação preliminar da abordagem de síntese desenvolvida
     - Auxiliar na definição inicial dos valores dos parâmetros evolutivos
       - Tempo pequeno de processamento

Estudos de Caso

1. Waterbath



Estudos de Caso

1. Waterbath

- Modelo Matemático

$$y(k+1) = a(T_s)y(k) + \frac{b(T_S)}{1+e^{0,5y(k)-\gamma}}u(k) + [1-a(T_s)]Y_0$$

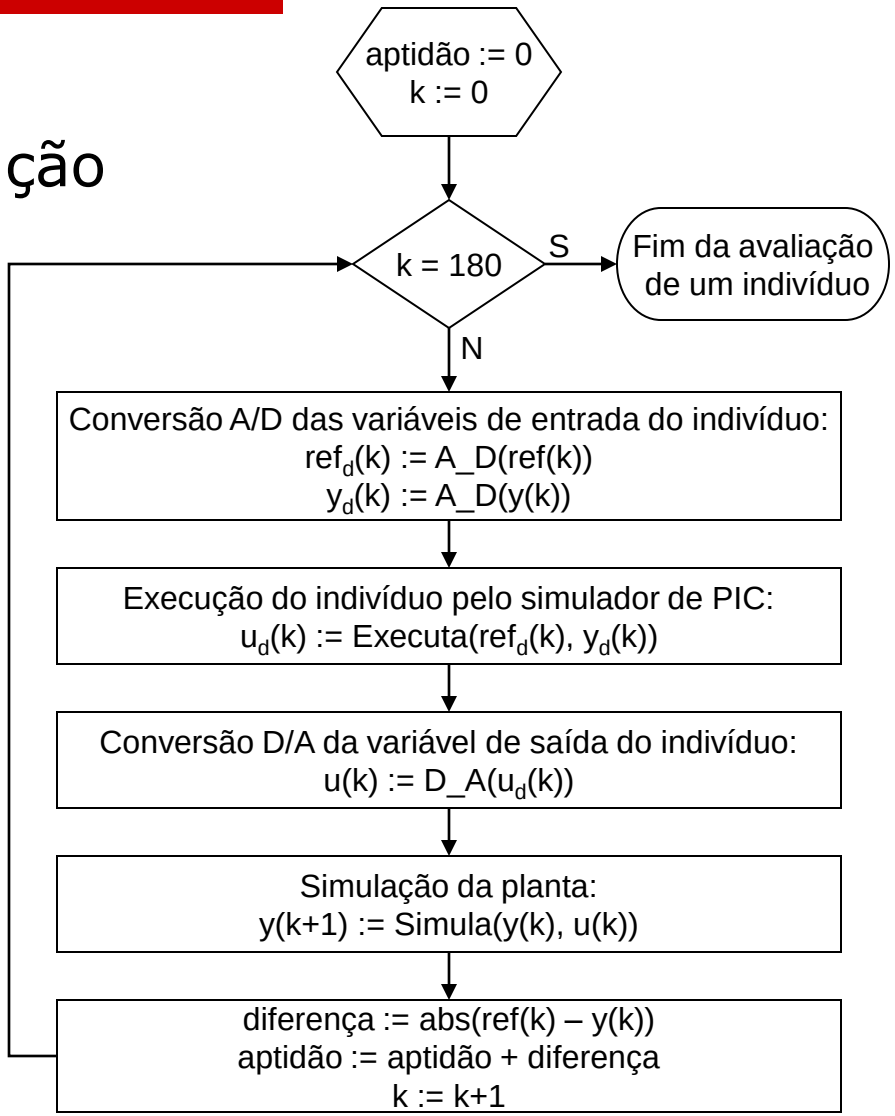
$$a(T_s) = e^{-\alpha T_s} \qquad b(T_s) = \frac{\beta}{\alpha}(1-e^{-\alpha T_s})$$

Y_0 : temperatura ambiente (25 ºC)
k : índice do tempo discretizado
u(k) : entrada do sistema (V)
y(k) : saída do sistema (ºC)
T_s : período de amostragem (30 s)

Constantes:
α = 1,00151E-4, β = 8,67973E-3, γ = 40,
dependentes de:
C : capacidade térmica do sistema
R : resistência térmica entre as fronteiras do sistema e o ambiente

Estudos de Caso

1. Waterbath
   - Função de Avaliação



Estudos de Caso

1. Waterbath

Resumo do Experimento

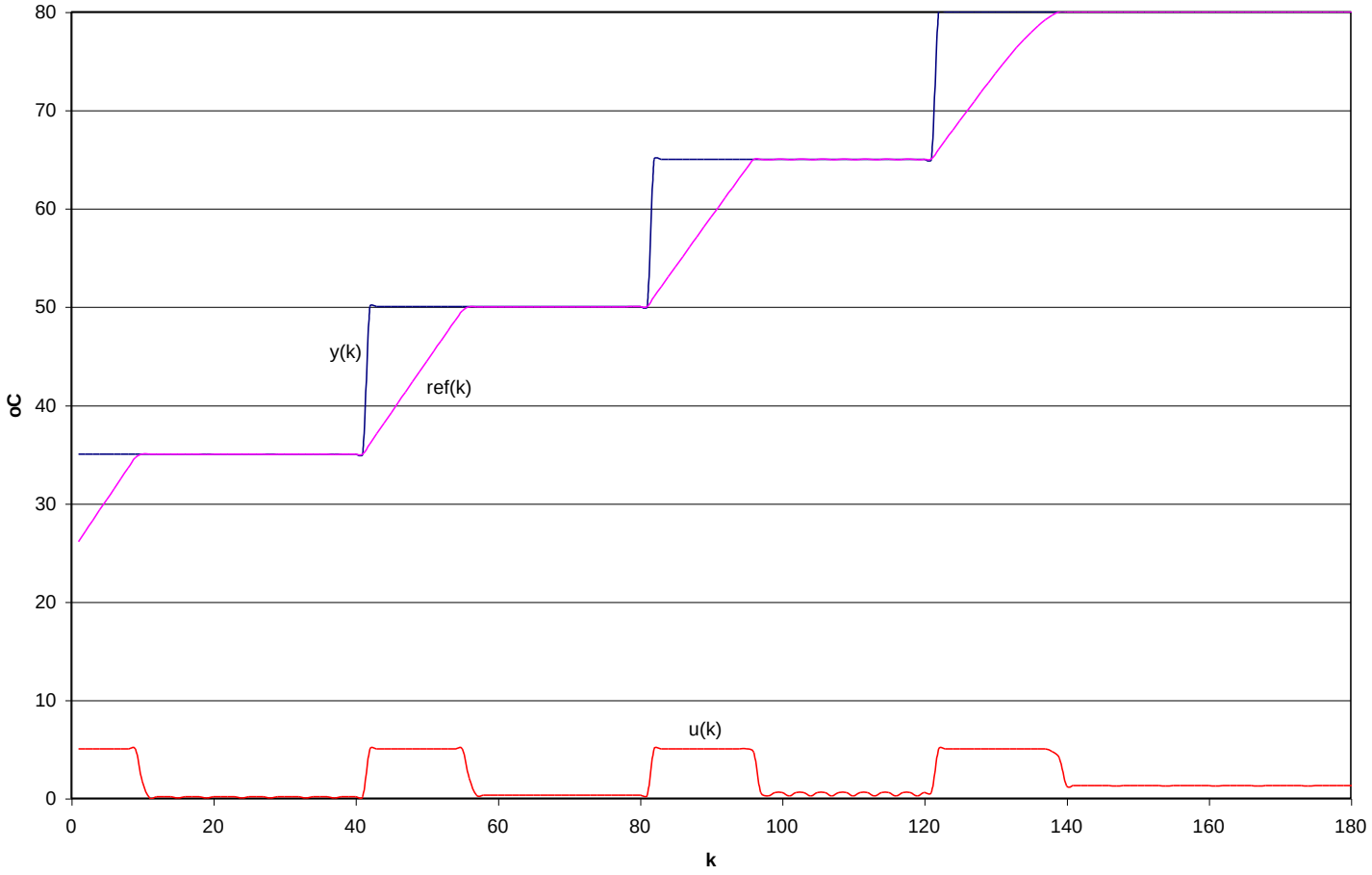
Objetivo:	Sintetizar um programa em linguagem de montagem que possibilite a um MC PIC18F452 controlar uma planta *water bath*, de forma a acompanhar o mais proximamente possível um dado perfil de temperatura de referência.
Conjunto de terminais:	As variáveis: R (referência), Y (temperatura) e U (controle); A1 e A2 (auxiliares); PRODH e PRODL (registradores do *hardware* multiplicador); STATUS (registrador de *flags*).
Conjunto de funções:	As 22 instruções implementadas no simulador do PIC.
Casos de treinamento:	Os 180 pontos (*k*,*ref(k)*) de amostras do perfil de referência.
Aptidão:	$\sum_k \lvert ref(k) - y(k) \rvert$
Avaliação:	A menor aptidão e, como critério de desempate, o menor comprimento do indivíduo.
Parâmetros principais:	M = 500. *Steady state*. Número de indivíduos processados: 2.000.000, o que equivale a G = 4.000. p_c = 0,8, p_m = 0,4.
Predicado de sucesso:	Nenhum

Estudos de Caso

1. Waterbath

- Resultados

Desempenho de controle do programa evoluído

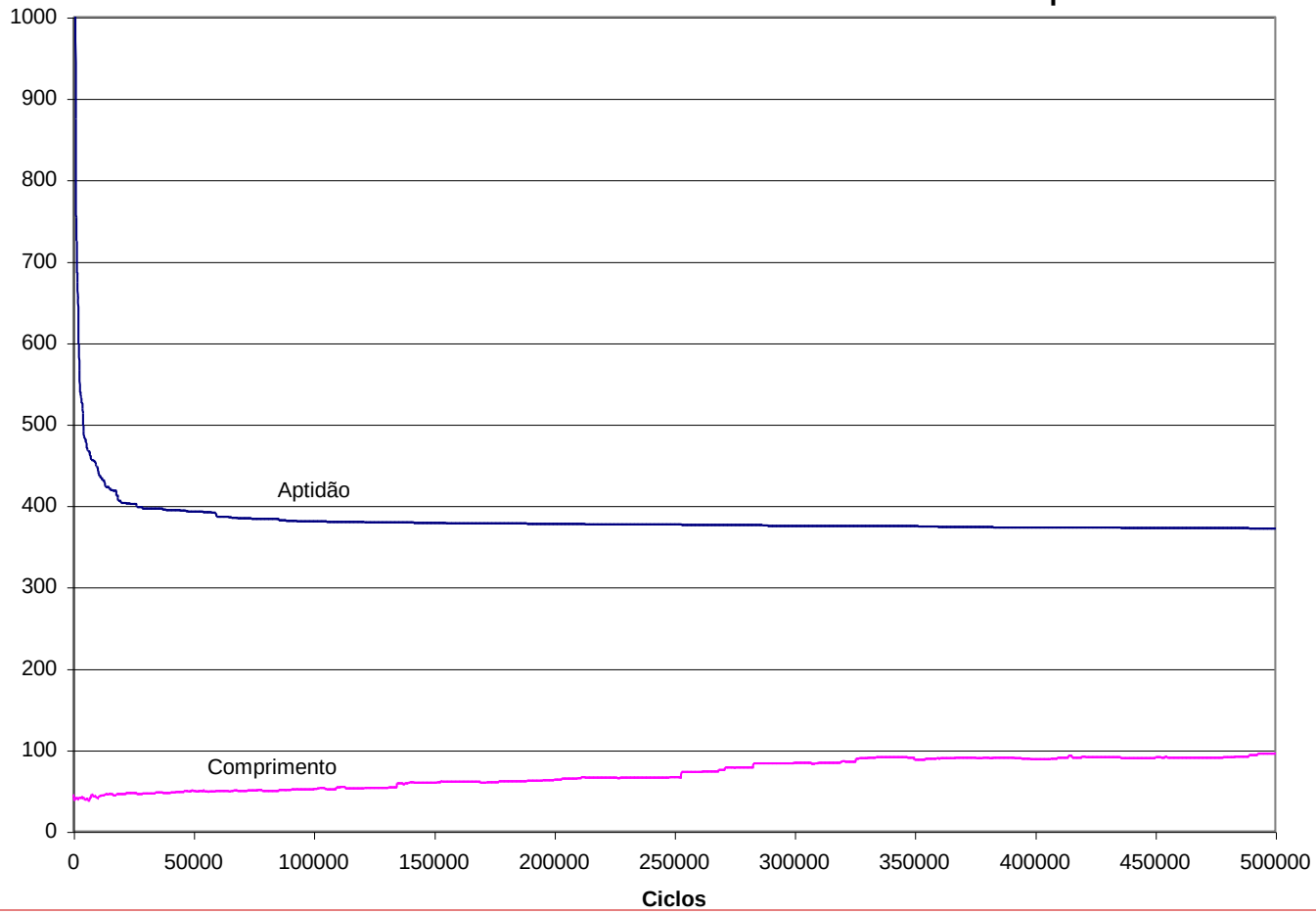


Estudos de Caso

1. Waterbath

- Resultados

Evolução sem controle sobre o comprimento

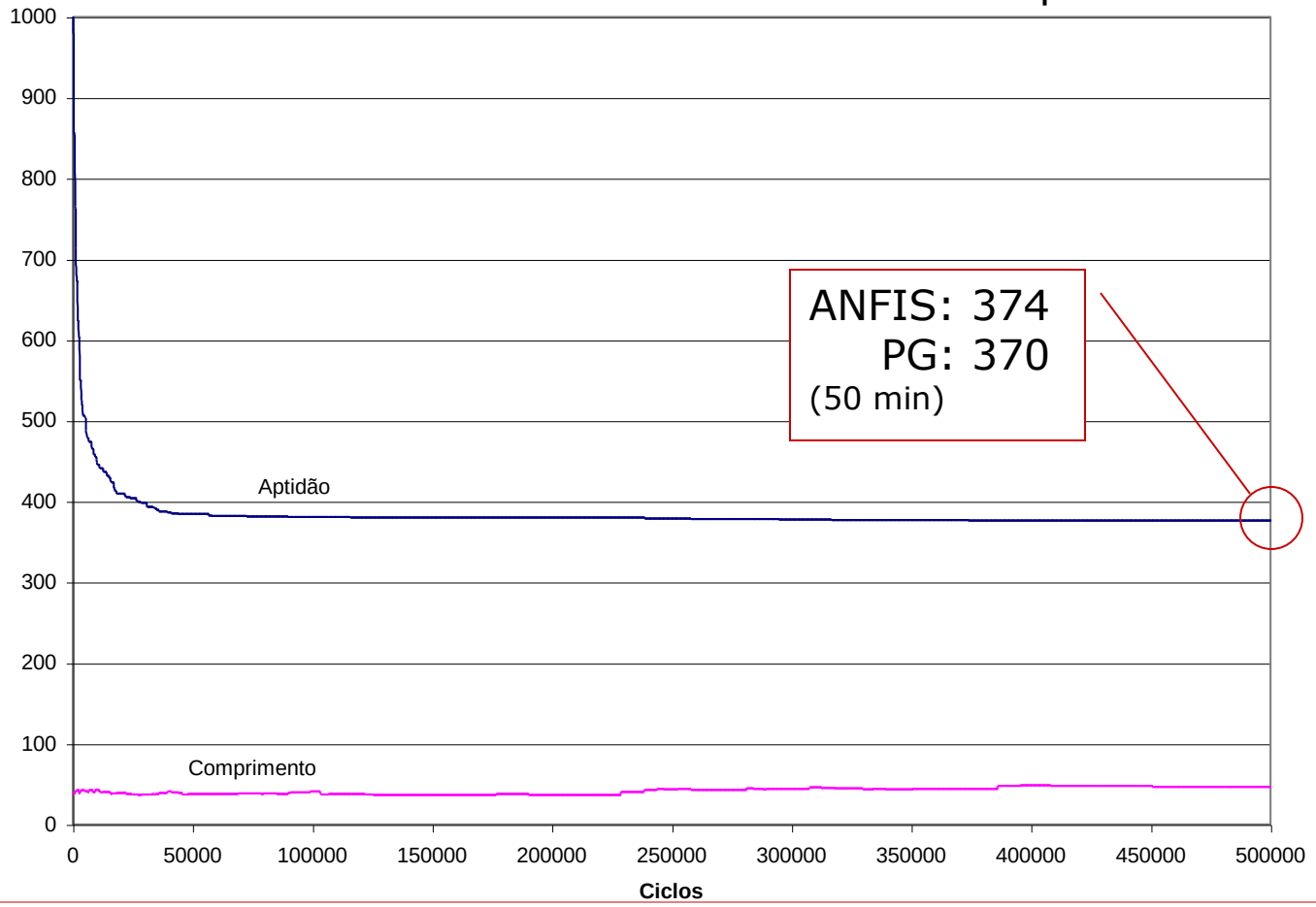


Estudos de Caso

1. Waterbath

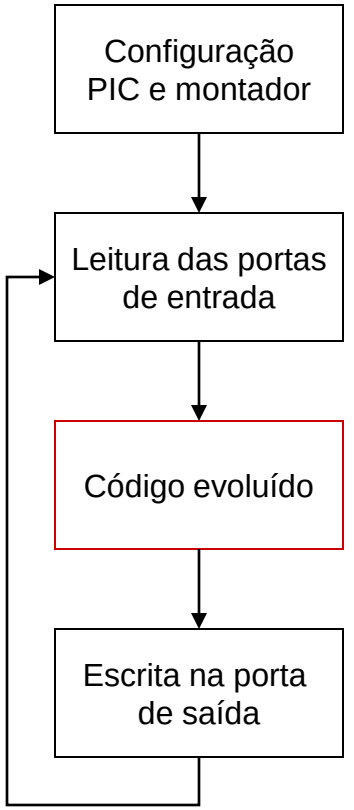
- Resultados

Evolução com controle sobre o comprimento



Estudos de Caso

1. Waterbath
   - Resultados
     - Programa resultante
       - Código evoluído com 33 instruções



Estudos de Caso

1. Waterbath

- Resultados

Alocação de memória

Configuração das portas

Leitura das portas de entrada

```
#INCLUDE <P18F452.INC>

          CBLOCK 0x08          ;Início da memória de usuário
                 R             ;Referência
                 Y             ;Temperatura
                 U             ;Controle
                 A2            ;Auxiliar
          ENDC                 ;Fim do bloco de memória

          ORG 0x00             ;Vetor de reset
          GOTO BEGIN           ;Endereço inicial de execução
BEGIN
          SETF      TRISD      ;TRISD:=11111111b
                               ;Define porta D como entrada
          SETF      TRISC      ;TRISC:=11111111b
                               ;Define porta C como entrada
          CLRF      TRISB      ;TRISB:=00000000b
                               ;Define porta B como saída
LOOP
          CLRF      W          ;W:=0.     Inicia reg.W
          CLRF      PRODH      ;PRODH:=0. Inicia reg.PRODH
          CLRF      PRODL      ;PRODL:=0. Inicia reg.PRODL
          MOVFF     PORTD, R   ;R:=PORTD
                               ;Lê porta D (referência)
          MOVFF     PORTC, Y   ;Y:=PORTC
                               ;Lê porta C (temperatura)
```

Estudos de Caso

1. Waterbath

- Resultados
  - Programa resultante

Código evoluído

```
INCF      Y, W
MOVF      U, W
NEGF      R
RLNCF     Y, F
RRCF      Y, F
ADDWFC    PRODL, F
ADDWFC    R, W
BTFSC     Y, 7
RLNCF     Y, F
CPFSEQ    R
INCF      Y, W
INCF      R, F
SUBWF     U, F
INCF      U, F
BTFSC     PRODL, 7
RLCF      R, F
BTFSC     PRODL, 7
CPFSGT    R
BTFSC     Y, 7
RLCF      R, F
SUBWF     PRODH, F
INCF      A2, F
CPFSEQ    Y
BTFSC     Y, 7
MOVWF     U
MULWF     U
INCF      U, F
RLCF      R, W
SUBWF     PRODH, F
BTFSC     R, 7
SETF      U
CPFSGT    R
ADDWFC    U, F
```

Escrita na porta de saída

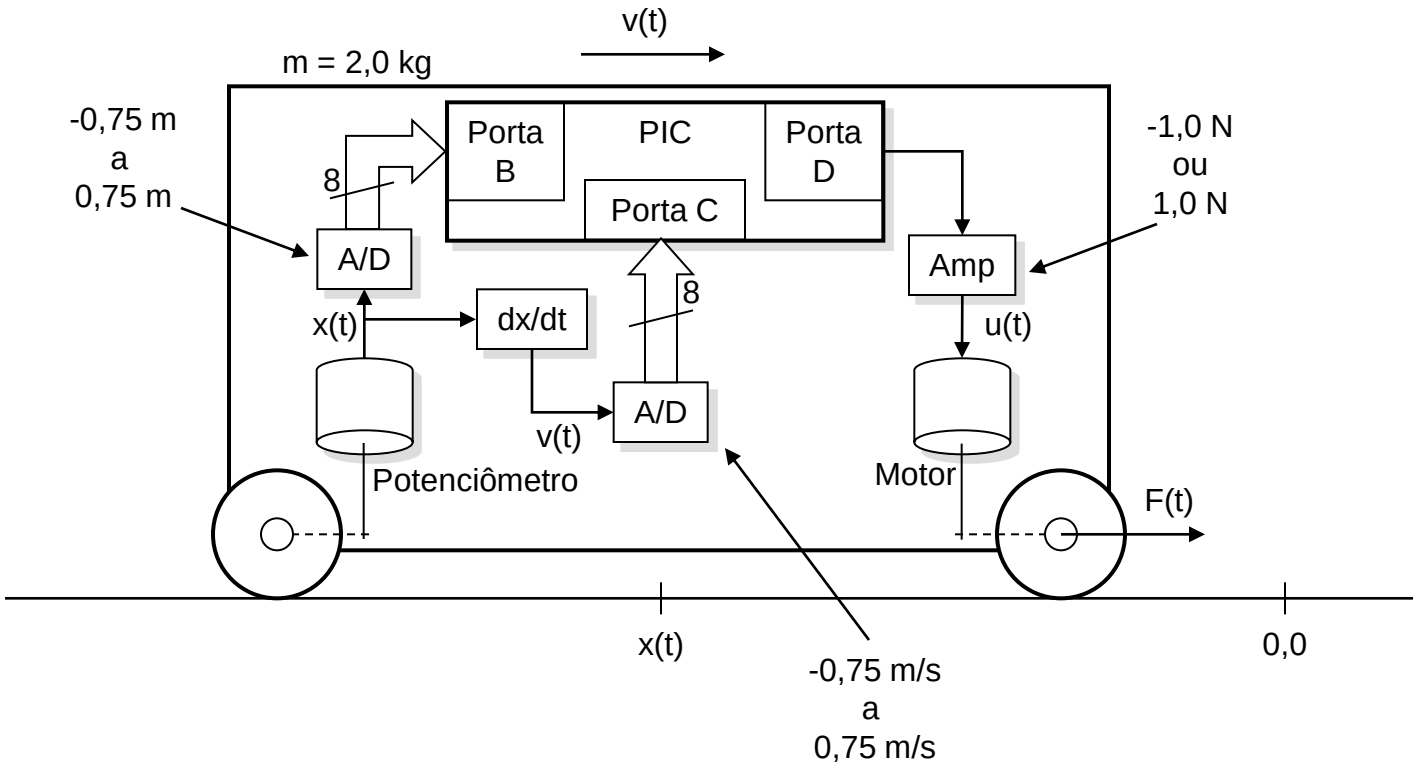
```
MOVFF U, PORTB      ;PORTB:=U
                    ;Escreve na porta B
GOTO  LOOP          ;Volta ao início do laço
END
```

Estudos de Caso

2. Cart Centering
   - Estacionar um carro, sobre um trilho, no menor tempo possível, aplicando-se força *bang-bang*
   - Problema bem conhecido na literatura introdutória sobre controle ótimo
   - Muito utilizado para testar novos conceitos em PG e em outras abordagens de inteligência computacional
   - Problema de tempo ótimo
   - Solução conhecida
   - Objetivo principal
     - Verificar a capacidade do sistema em evoluir um programa que implementasse a estratégia de controle ótimo

Estudos de Caso

2. Cart Centering



Estudos de Caso

2. Cart Centering

- Modelo Matemático

$$a(t) = \frac{F(t)}{m}$$

$$v(t+1) = v(t) + \tau a(t)$$

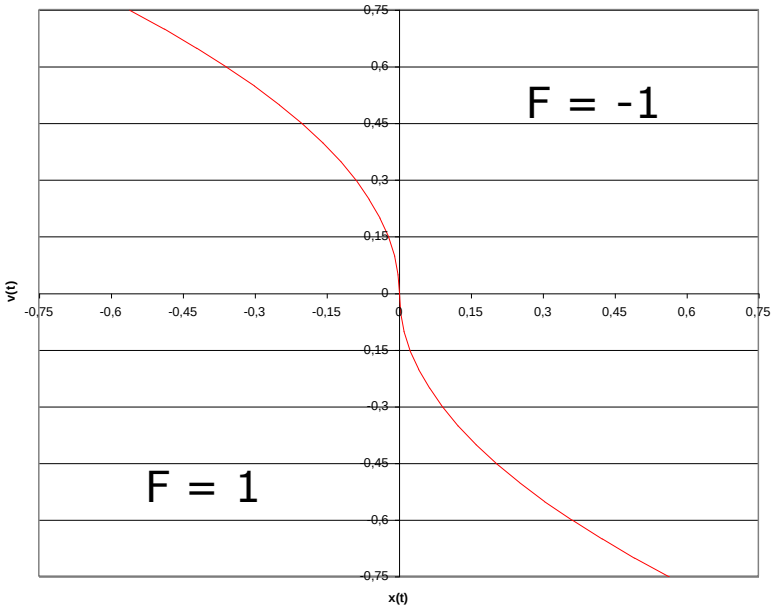
$$x(t+1) = x(t) + \tau v(t)$$

a(t) : aceleração do carro (m/s^2)
F(t) : força aplicada ao carro em t (1,0 N ou –1,0 N)
m : massa do carro (2,0 kg)
v(t) : velocidade do carro (m/s)
x(t) : posição do carro (m)
τ : período de amostragem (0,02 s)

Estudos de Caso

2. Cart Centering

- Estratégia de Controle Ótimo



F(t) é positiva se

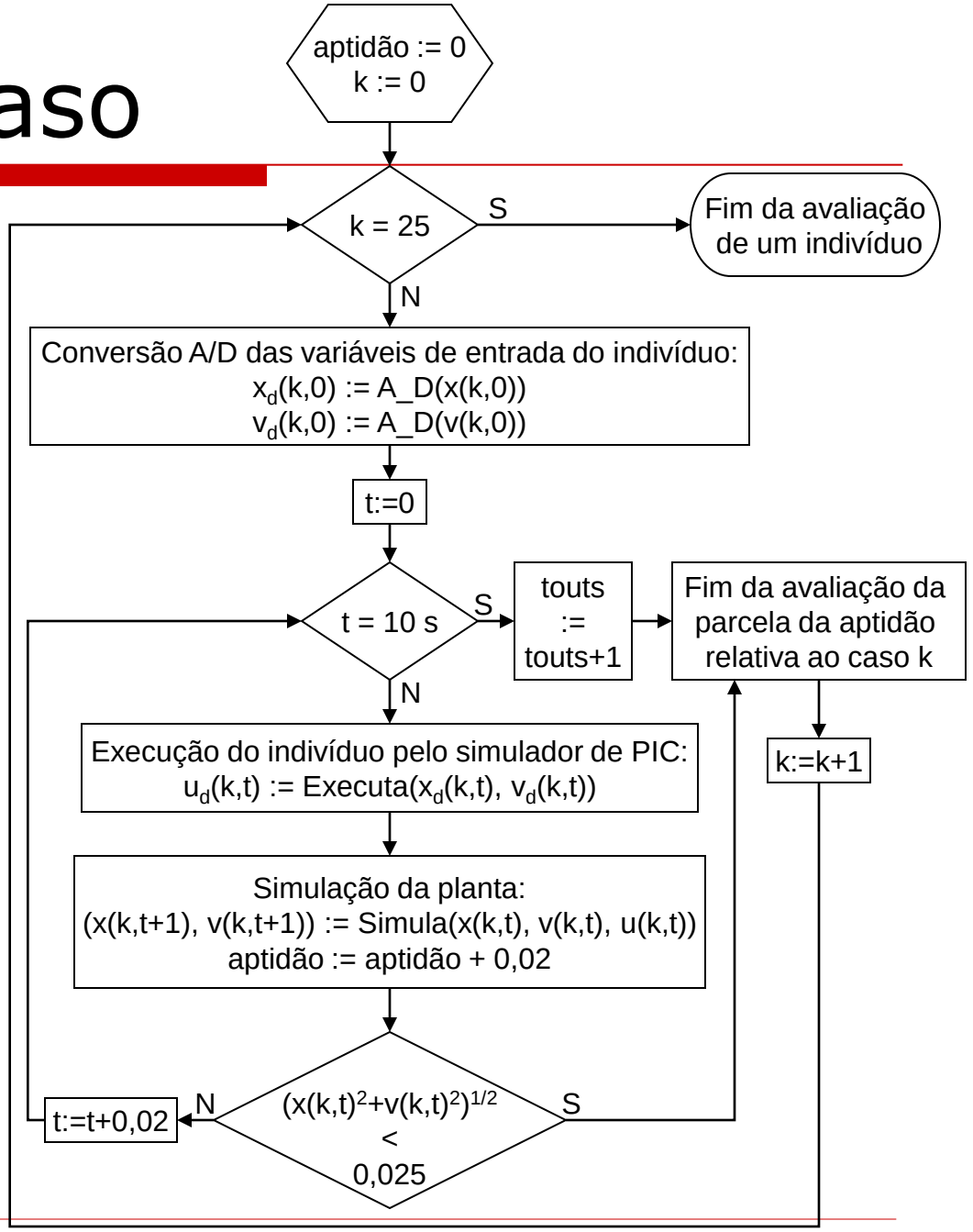
$$-x(t) > \frac{v(t)^2 \, Sign\, v(t)}{2|F|/m}$$

Estudos de Caso

2. Cart Centering

- Função de avaliação

Valores iniciais:
x = -0,75; -0,375; 0,0;
0,375; 0,75
v = -0,75; -0,375; 0,0;
0,375; 0,75
(todas as 25 combinações)



Estudos de Caso

2. Cart Centering

- Resumo do Experimento

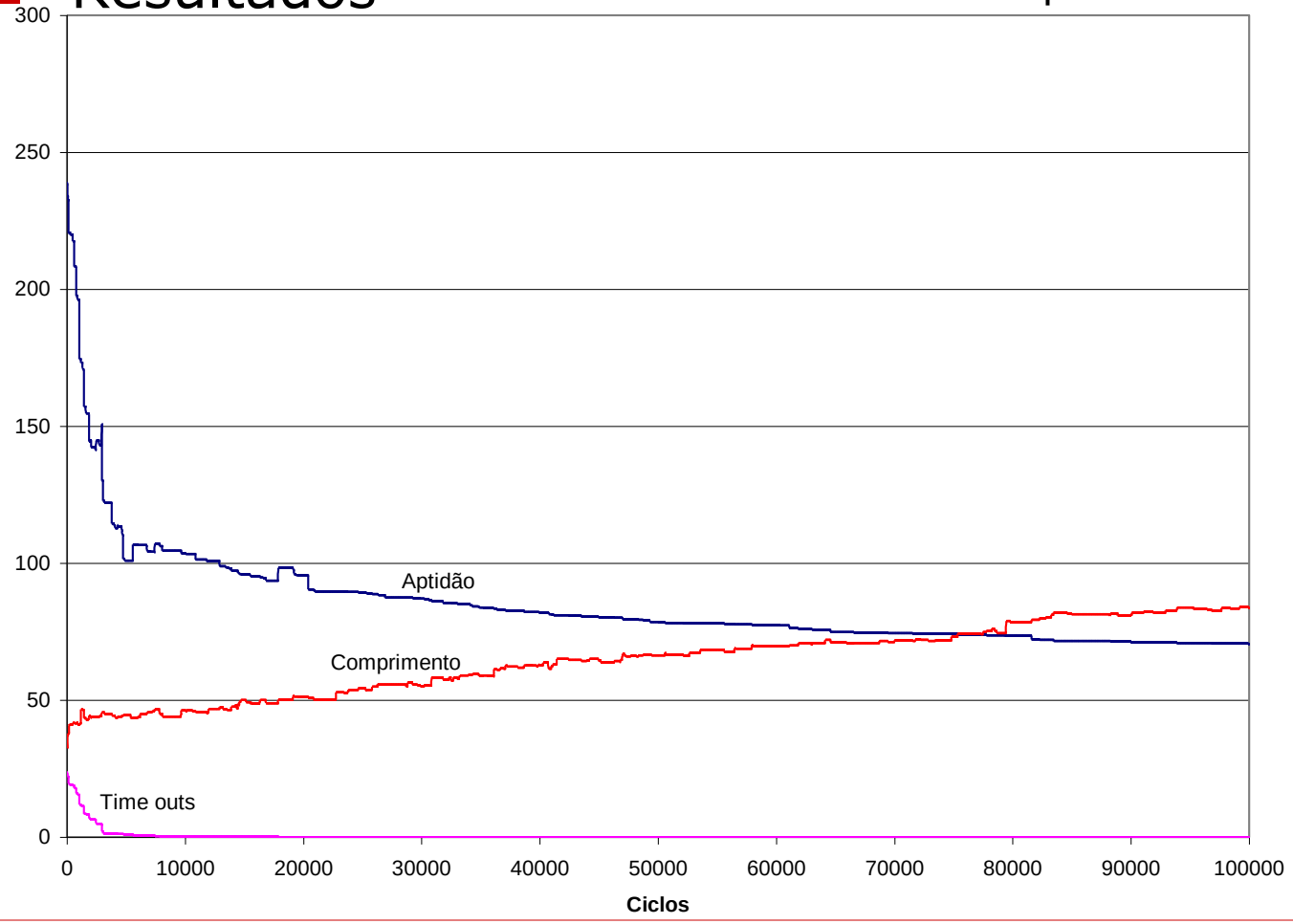
Objetivo:	Sintetizar um programa em linguagem de montagem que possibilite a um MC PIC18F452 controlar um carro em um trilho sem atrito, de forma a estacioná-lo em uma posição central final.
Conjunto de terminais:	X (posição), V (velocidade) e U (controle); A1 e A2 (auxiliares); PRODH e PRODL (registradores do *hardware* multiplicador); STATUS (registrador de *flags*).
Conjunto de funções:	As 22 instruções implementadas no simulador do PIC
Casos de treinamento:	25 pontos de condições iniciais (x,v), escolhidos uniformemente espaçados, entre –0,75 e 0,75. 5 valores para x e 5 valores para v (-0,750, -0,375; 0; 0,375, 0,750).
Acertos:	Número de casos de treinamento que não causaram *timeout*
Aptidão:	Soma do tempo, sobre os 25 casos de treinamento, levado para estacionar o carro. Quando um caso de treinamento causa um *timeout*, a contribuição é de 10 segundos.
Avaliação:	O menor número de *timeouts*. Como primeiro critério de desempate, a menor aptidão. Como segundo critério de desempate, o menor comprimento do indivíduo.
Parâmetros principais:	M = 500. *Steady state*. Número de indivíduos processados: 400.000, o que equivale a G = 800. p_c = 0,8, p_m = 0,4.
Predicado de sucesso:	Nenhum.

Estudos de Caso

2. Cart Centering

- Resultados

Evolução sem controle sobre o comprimento

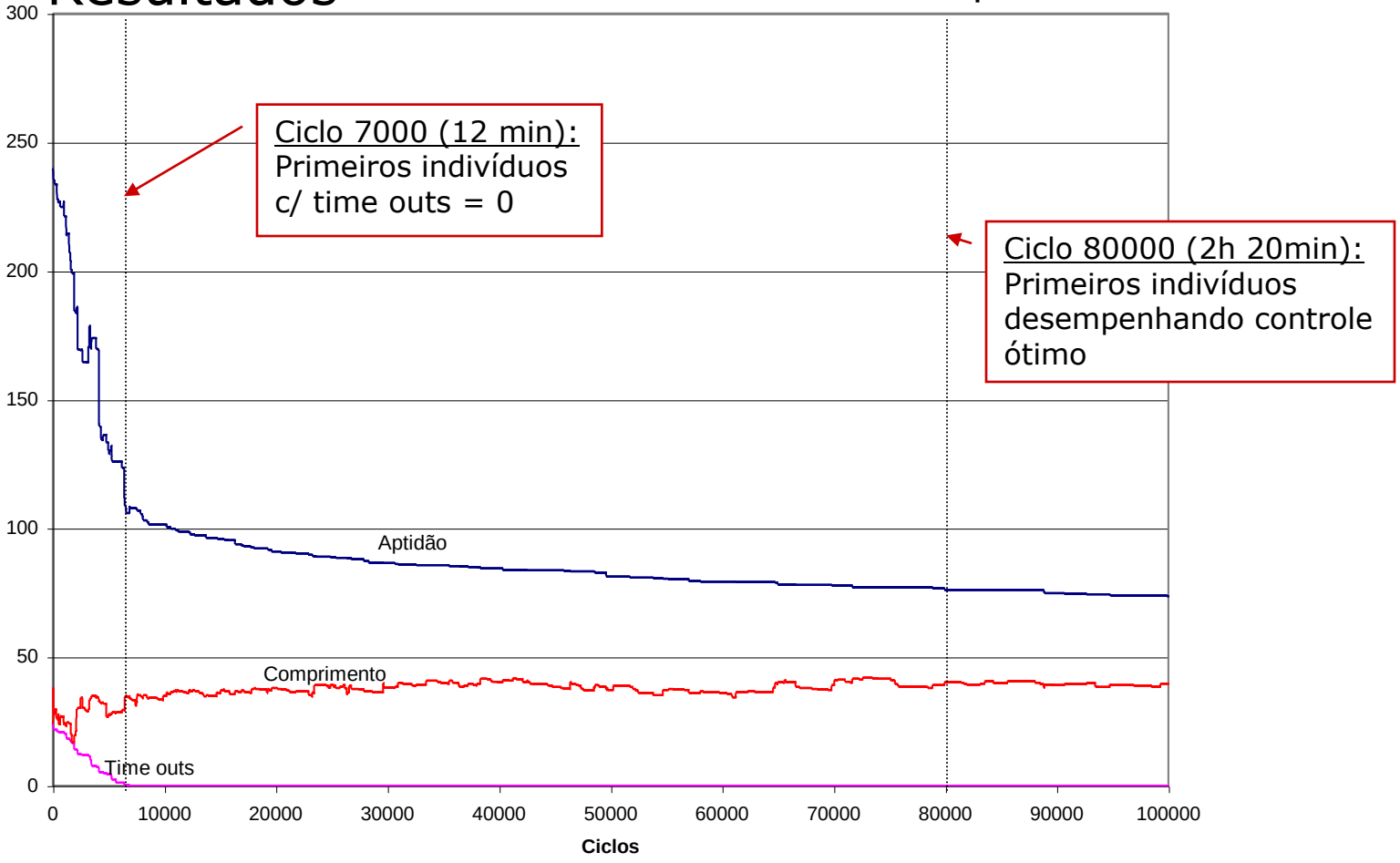


Estudos de Caso

2. Cart Centering

- Resultados

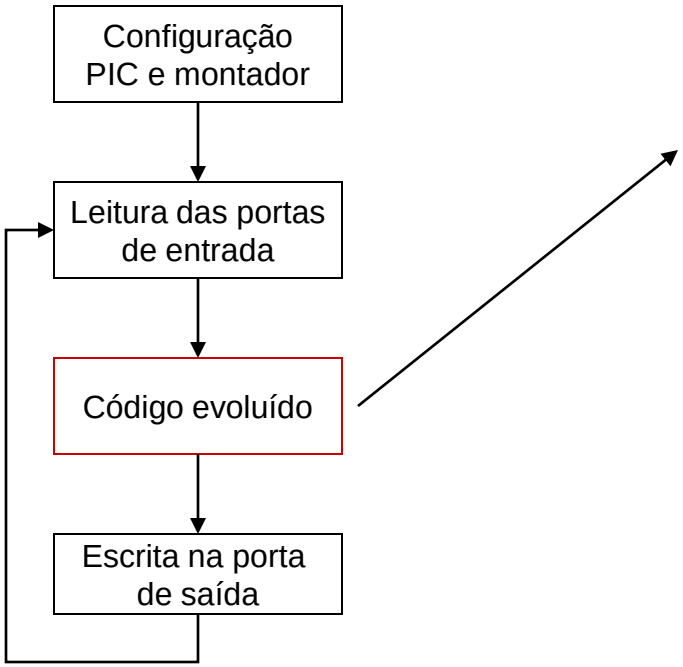
Evolução com controle sobre o comprimento



Estudos de Caso

2. Cart Centering

- Resultados
  - Programa resultante
    - Código evoluído com 25 instruções



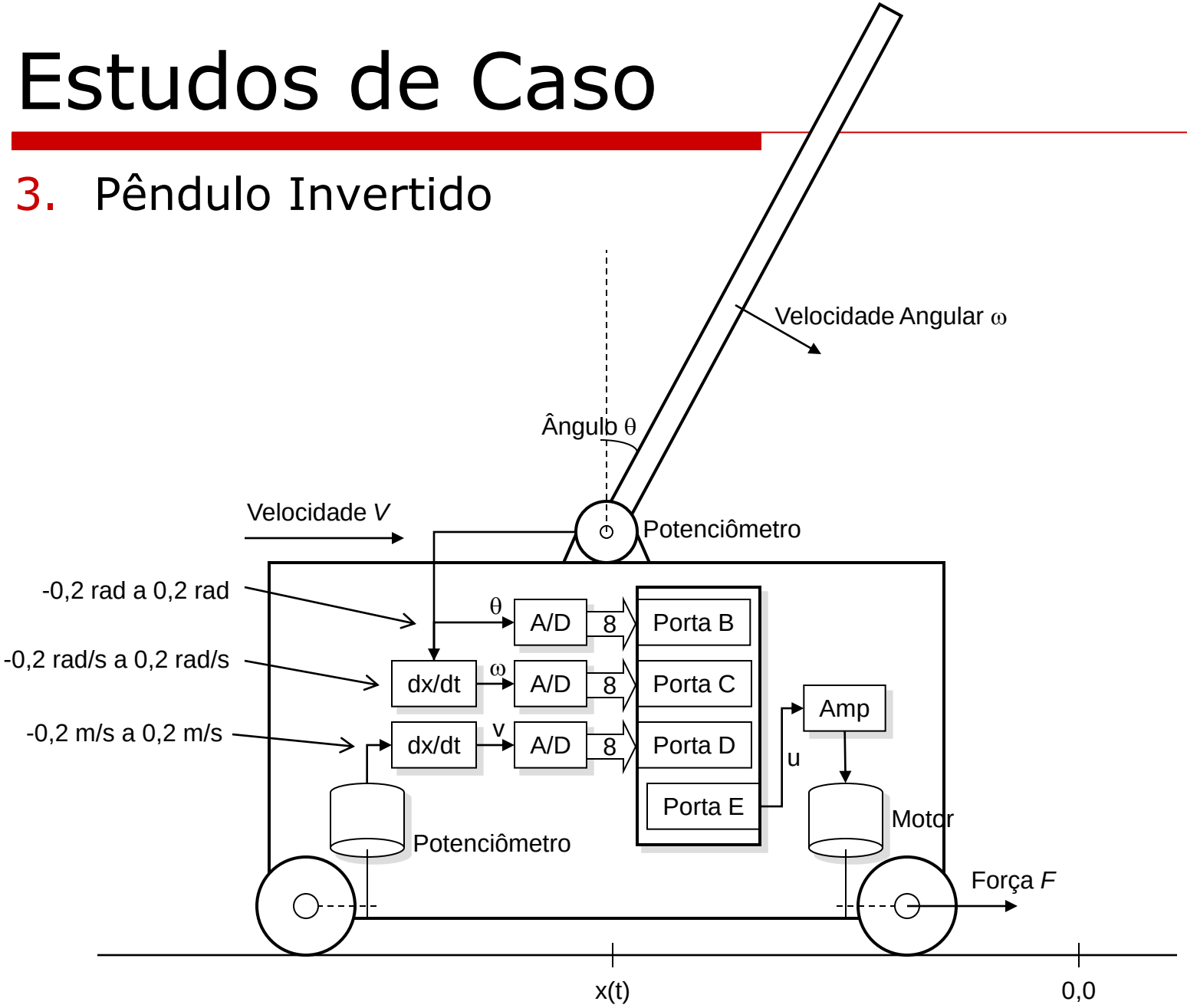

```
BTFSC    X,7
SUBWF    V,W
BTFSC    V,7
SUBWF    V,F
MULWF    X
INCF     V,W
ADDWF    X,F
ADDWF    X,F
MULWF    PRODH,W
SUBWF    PRODH,W
MULWF    X
ADDWF    X,F
ADDWF    X,F
SUBWF    PRODH,W
BTFSC    V,7
ADDWF    X,F
BTFSC    X,7
SETF     U
MULWF    PRODH
SUBWF    PRODH,W
ADDWF    X,F
MULWF    X
MULWF    PRODH
BTFSC    PRODH,7
SETF     U
```

Estudos de Caso

3. Pêndulo Invertido
   - Equilibrar um pêndulo sobre um carro no menor tempo possível
   - Problema bem conhecido em teoria de controle
   - Sistema mecânico inerentemente instável e de dinâmica não linear
   - Solução desconhecida
   - Controladores lineares obtiveram sucesso limitado
   - Controladores não-lineares foram desenvolvidos por diversas abordagens de inteligência computacional
   - Objetivos principais
     - Verificar o desempenho do sistema ao sintetizar uma solução para uma planta mais complexa
     - Comparar o resultado com a solução obtida por Koza (1992)

Estudos de Caso

3. Pêndulo Invertido



Estudos de Caso

3. Pêndulo Invertido

- Modelo Matemático

$$\Phi(t) = \frac{g \operatorname{sen}\theta + \cos\theta \dfrac{-F - m_p \lambda \omega \theta^2 \operatorname{sen}\theta}{m_c + m_p}}{\lambda\left(\dfrac{4}{3} - \dfrac{m_p \cos^2\theta}{m_c + m_p}\right)}$$

Φ : aceleração angular do pêndulo (rad/s^2)
ω : velocidade angular do pêndulo (rad/s)
θ : ângulo de inclinação do pêndulo (rad)
F : força aplicada ao carro (1,0 N ou –1,0 N)
$\mathbf{m_c}$: massa do carro (0,9 kg)
$\mathbf{m_p}$: massa do pêndulo (0,1 kg)
λ : altura do centro de gravidade do pêndulo (0,8106 m)
g : gravidade (1,0 m/s)

Estudos de Caso

3. Pêndulo Invertido

- Modelo Matemático

$$\omega(t+1) = \omega(t) + \tau\Phi(t)$$

$$\theta(t+1) = \theta(t) + \tau\omega(t)$$

$$a(t) = \frac{F + m_p\lambda\left(\theta^2 \operatorname{sen}\theta - \omega\cos\theta\right)}{m_c + m_p}$$

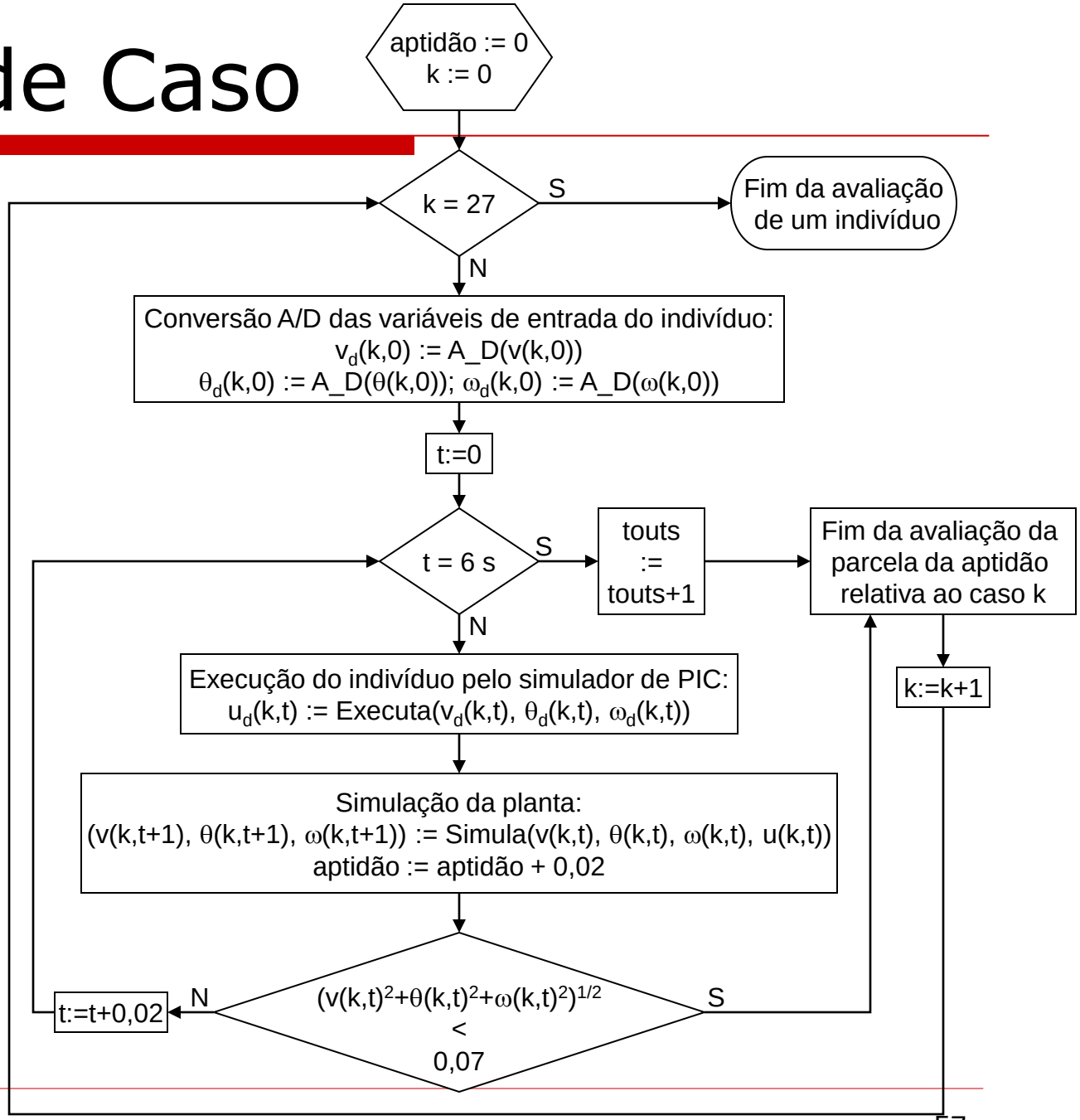
$$v(t+1) = v(t) + \tau a(t)$$

a : aceleração do carro (m/s^2)
τ : período de amostragem (0,02 s)

Estudos de Caso

3. Pêndulo Invertido
   - Função de Avaliação

Valores iniciais:
v = -0,2; 0,0; 0,2
θ = -0,2; 0,0; 0,2
ω = -0,2; 0,0; 0,2
(todas as 27 combinações)



Estudos de Caso

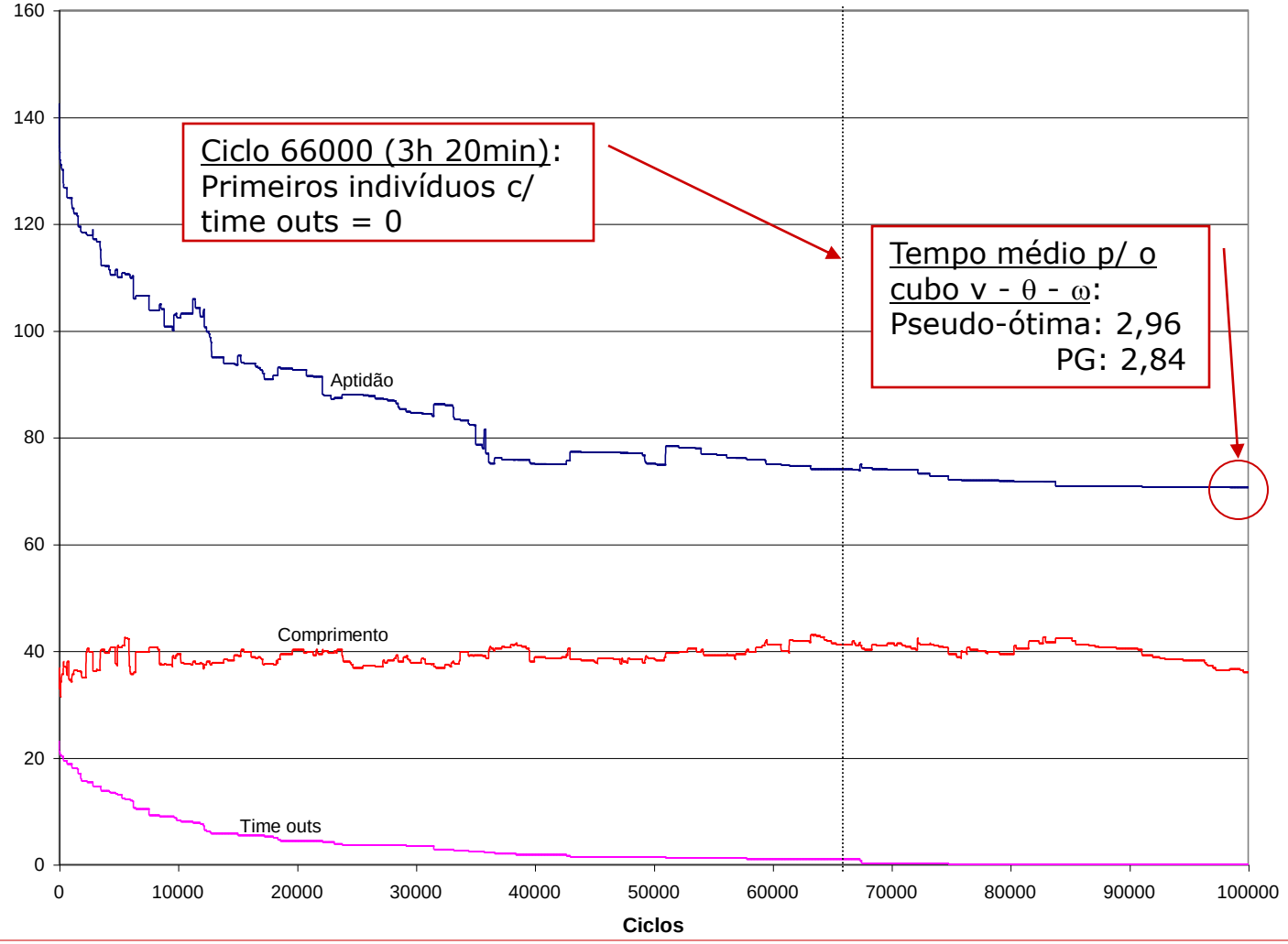
3. Pêndulo Invertido

- Resumo do Experimento

Objetivo:	Sintetizar um programa em linguagem de montagem que possibilite a um MC PIC18F452 controlar um carro em um trilho, de forma a conduzir um pêndulo invertido ao equilíbrio.
Conjunto de terminais:	V (velocidade do carro), O (ângulo), w (velocidade angular) e U (controle); A1 e A2 (auxiliares); PRODH e PRODL (registradores do *hardware* multiplicador); STATUS (registrador de *flags*).
Conjunto de funções:	As 22 instruções implementadas no simulador do PIC
Casos de treinamento:	27 pontos de condições iniciais (v, θ, ω), escolhidos uniformemente espaçados, entre –0,2 e 0,2. São 3 valores para cada variável: -0,2; 0 e 0,2.
Acertos:	Número de casos de treinamento que não causaram *timeout*
Aptidão:	Soma do tempo, sobre os 27 casos de treinamento, levado para estacionar o carro. Quando um caso de treinamento causa um *timeout*, a contribuição é de 6 segundos.
Avaliação:	O menor número de *timeouts*. Como primeiro critério de desempate, a menor aptidão. Como segundo critério de desempate, o menor comprimento do indivíduo.
Parâmetros principais:	M = 1000. *Steady state*. Número de indivíduos processados: 400.000, o que equivale a G = 400. p_c = 0,8, p_m = 0,4.

Estudos de Caso

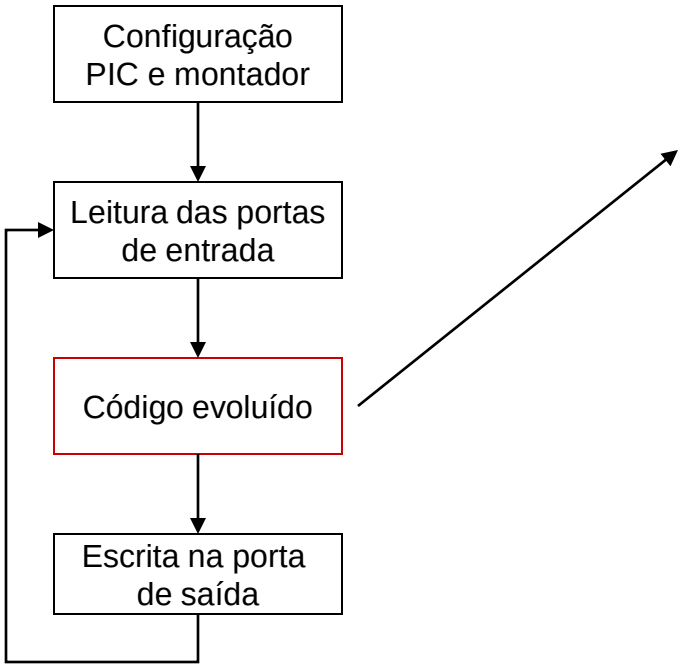
3. Pêndulo Invertido / Resultados



Estudos de Caso

3. Pêndulo Invertido

- Resultados
  - Programa resultante
    - Código evoluído com 25 instruções




```
ADDWFC   PRODH, F
BTFSC    STATUS, 0
RLCF     PRODL, F
RLCF     w, F
RLCF     O, W
ADDWF    w, F
ADDWFC   PRODH, F
ADDWF    w, W
ADDWFC   V, W
SUBWF    PRODL, F
MOVFF    w, W
RLCF     U, F
SUBWF    PRODH, F
RLCF     PRODL, F
ADDWF    V, W
ADDWFC   V, W
RLCF     O, W
ADDWFC   PRODL, F
CPFSGT   V
ADDWFC   PRODL, F
CPFSEQ   PRODL
CPFSGT   V
SUBWF    PRODH, W
BTFSS    U, 7
RLCF     U, F
```

Sumário

- ✓ Motivação
- ✓ Controle de Processos por Meio de Microcontroladores: Caracterização e Aplicação
- ✓ Programação Genética
- ✓ Sistema de Síntese Automática de Programas em Linguagem de Montagem
- ✓ Estudos de Caso
- ☐ Conclusões

Conclusões

- Evolução direta em assembly mostrou-se imprescindível
  - Evolução de programas com otimização do código – plataforma limitada
  - A compilação, efetuada a cada avaliação, tomaria muito tempo e inviabilizaria o uso prático do sistema
- Sistema mostrou desempenho satisfatório nos estudos de caso
  - Sintetizou programas com desempenho comparável a outras abordagens

Conclusões

- Uso prático do sistema mostrou-se vantajoso
  - Programas obtidos diretamente do modelo matemático da planta
    - Elimina busca matemática por uma solução de controle ótimo ou sub-ótimo
  - Soluções no formato final da plataforma de implementação
    - Elimina programação manual da solução encontrada
    - Elimina possíveis etapas intermediárias entre a modelagem e a implementação

Conclusões

- Tempo de síntese competitivo com um ser humano
  - Forneceu soluções em poucas horas
  - Síntese tradicional pode levar alguns dias
    - Complexidade dos sistemas típicos representados pelos estudos de caso
    - Programação de um dispositivo limitado em recursos como o MC
- É importante diminuir o tempo total de evolução
  - Mesmo sendo competitivo, ainda é relativamente longo

Conclusões

- Possíveis direções de trabalhos futuros
  - Desenvolver algoritmo que identifique trechos não efetivos de código (*introns*)
    - *Introns* não seriam executados pelo simulador do MC
      - Economia de tempo
    - Mutação aplicada apenas no código efetivo
      - Maior rendimento do operador
  - Investigar o uso de nichos (*demes*)
    - Retardar convergência final
    - Processamento paralelo

Conclusões

- Possíveis investigações
  - Investigar as possíveis contribuições de se evoluir laços
    - Estudar o tratamento de laços infinitos
  - Desenvolver o sistema para MCs baseados na plataforma Intel x86
    - Aumento da velocidade do sistema
      - Programas executados pela CPU do PC
      - Estabelecimento de uma plataforma de evolução intrínseca