

Automação em Tempo Real

Módulo 1: Introdução aos Sistemas de Automação

*A problem well stated is a problem half solved
(Sir Isaac Newton)*

Professor: Constantino Seixas Filho

terça-feira, 28 de março de 2006

Livro



- Seixas, Constantino e Szuster, Marcelo, Programação Concorrente em ambiente WNT: Uma visão de automação, Editora da UFMG, 2003

Filosofia Geral – Modelo Hierárquico



9290-102-24 (rev.:00)

3



Programação Concorrente

- Programação concorrente x Programação Seqüencial



Programação seqüencial [Gary Larson, The Far side Gallery]

9290-102-24 (rev.:00)

4



Monotarefa X Multitarefas



- Hoje tudo na vida é multitasking (multitarefas)
 - Trabalho, esporte, lazer, cuidar de filhos, etc.
- Só faz sentido quando não estamos utilizando o mesmo recurso da máquina

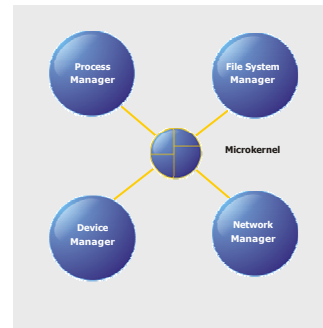
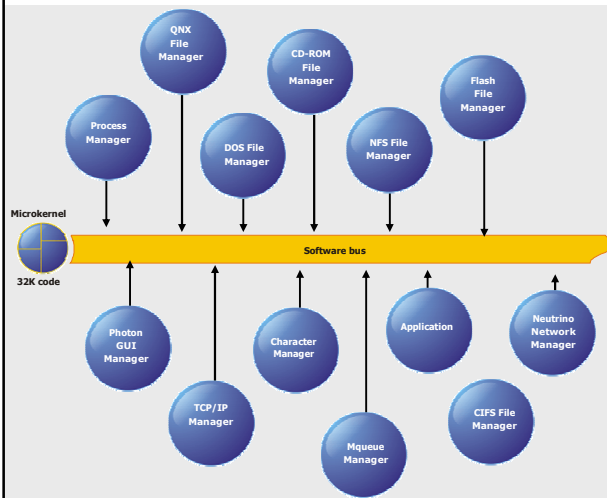


Por que programação concorrente ?



- O computador é mais rápido que seus usuários
Exemplo: digitação
 - Bom digitador: 60 palavras /minuto
 - Tempo de processamento de 1 caracter = 10^{-4} s
 - Computador realiza 10 s de trabalho a cada 2.3 dias
- Em geral várias atividades não concorrem pelos mesmos recursos: digitar, imprimir, ouvir mp3, baixar arquivos, compilar, etc.
- Melhor modelamento das atividades
- O próprio sistema operacional é formado por dezenas de processos concorrentes: arquitetura microkernel

Arquitetura Microkernel



9290-102-24-002 (Rev.:00)

9290-102-24 (rev.:00)

7

Natureza das Aplicações

- Cíclicas (Evento de tempo)
- Em resposta a eventos de processo ou de operação
- Sincronizadas ao mundo externo
- De tempo real

9290-102-24 (rev.:00)

8

Definições Básicas

- Escalonamento
- Scheduler = Escalonador
- Time-sharing
- Time-slice = Quantum
- Tasks: implementadas como processos ou threads
- Colaborativo x Preemptivo

Preempção





- A programação concorrente é mais ineficiente que a programação seqüencial. Por que ?
- Só faz sentido quando não estamos utilizando o mesmo recurso da máquina
- Processor bound (intensivo) x I/O bound

Instruções Atômicas

X := X + 1;

CPU CISC	CPU RISC
inc <X>	ld AX, <X> inc <AX> ld <X>, AX

- Instruções atômicas são instruções que o processador executa como uma operação indivisível, isto é, sem interrupção

CISC = Complete Instruction Set Computer

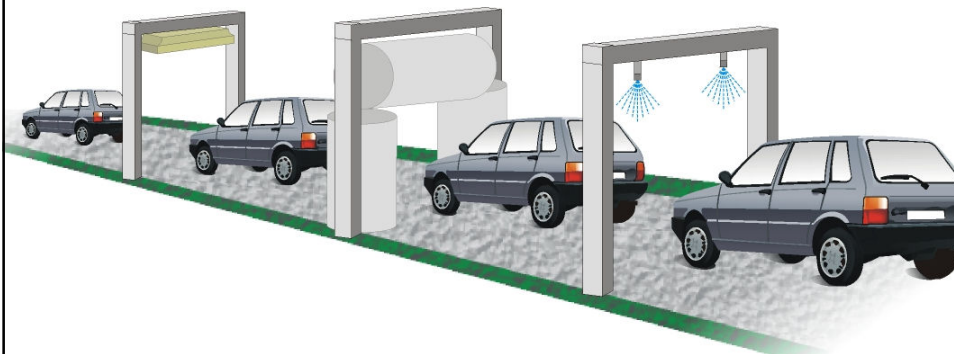
- Instruction Set com grande repertório de instruções
- Instruções gastam vários clocks para completar
- Processador possui poucos registradores internos
- Instruções envolvem memória externa
- ALU para inteiros apenas
- Operações floating point executadas em co-processador externo
- Instruções utilizam microcódigo
- Exemplo: Intel 80386

RISC = Reduced Instruction Set Computer

- Instruction Set com pequeno repertório de instruções
- Instruções executam em apenas 1 ou menos ciclos de clock
- Processador possui muitos registradores internos
- Instruções buscam dados externos para registrador interno e operam com estes registradores
- Operações com inteiros e floating points
- Instruções não utilizam microcódigo
- Arquitetura pipeline e superescalar
- Exemplo: CPU Sparc SUN

Pipeline

- Várias operações simultâneas



9290-102-24-003 (Rev.:00)

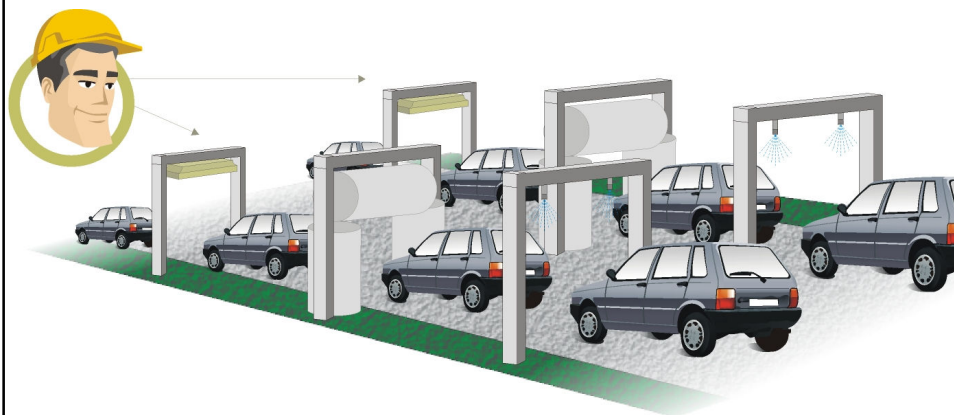
- A CPU trabalha como uma linha de montagem

9290-102-24 (rev.:00)

15

Superescalar

- Duplicando a linha de montagem



9290-102-24-003 (Rev.:00)

9290-102-24 (rev.:00)

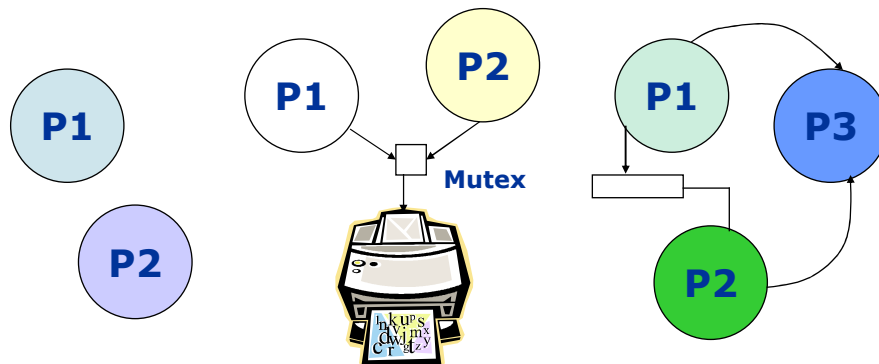
16

Mais Conceitos

- S.O. Multitasking → Multiprogramação (multitasking)
- S.O. Multiusuário
- S.O. Multiprocessado → Programação paralela
- Multiprocessamento Simétrico (SMP)
- Programação Distribuída

Interrelação entre Processos

- Não dependentes
- Contenção
- Cooperação



O Problema Fundamental da Programação Concorrente



Processo A	Processo B
<pre>int ConfirmaReserva(int PosicaoDesejada) { A1: if (MapaOcupacao[PosicaoDesejada] == LIVRE) A2: MapaOcupacao[PosicaoDesejada] = OCUPADO; else return(FALHA); return(SUCESSO); }</pre>	<pre>int ConfirmaReserva(int PosicaoDesejada) { B1: if (MapaOcupacao[PosicaoDesejada] == LIVRE) B2: MapaOcupacao[PosicaoDesejada] = OCUPADO; else return(FALHA); return(SUCESSO); }</pre>

L	L	L	O	L	L	L	L
O	L	L	O	O	L	O	O



9290-102-24 (rev.:00)

19

O Problema Fundamental da Programação Concorrente



Caso	Intercalação	Resultado
1	A1, A2, B1, B2	Programa A retorna SUCESSO e o Programa B retorna FALHA
2	B1, B2, A1, A2	Programa A retorna FALHA e o Programa B retorna SUCESSO
3	A1, B1, A2, B2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
4	B1, A1, A2, B2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
5	A1, B1, B2, A2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
6	B1, A1, B2, A2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO

- Conclusão:**
 Na programação concorrente não basta que cada programa seqüencial esteja logicamente correto. É necessário que produza um resultado correto para qualquer ordem de intercalação de suas instruções com uma outra instância do mesmo programa ou com qualquer outro programa em execução. Isto não exclui o próprio sistema operacional

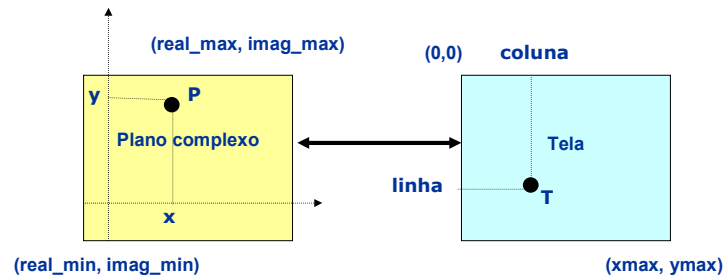
9290-102-24 (rev.:00)

20

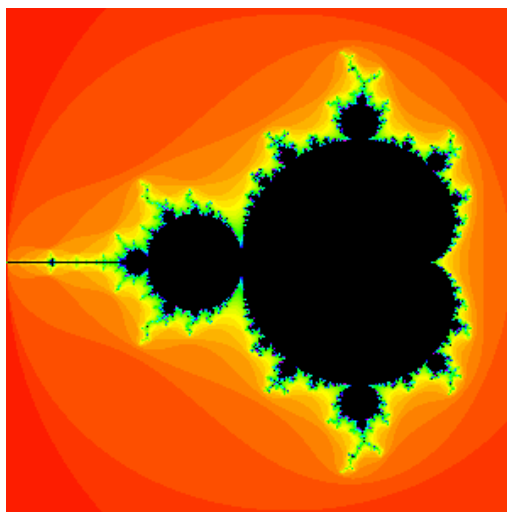
Exercícios: Mandelbrot Set

$$z_{n+1} = z_n^2 + p$$

$$p = x + jy$$



Exercícios: Mandelbrot Set



- Por que este problema é bom para ser dividido entre diversos processadores ?

Exercícios

- Como desenhar um círculo na tela do computador de forma eficiente e dividindo a tarefa entre pelo menos duas tarefas ?

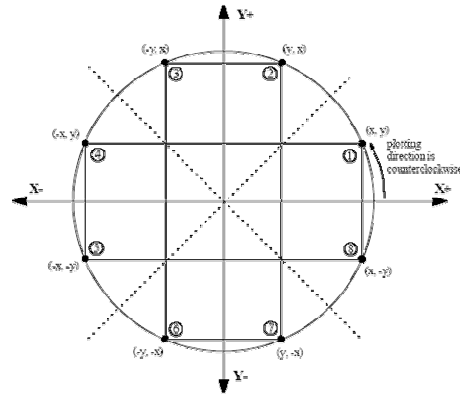
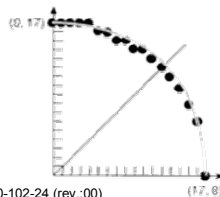
$$x^2 + y^2 = r^2$$

para $0 < x < r$

$$x = r \cos \theta$$

$$y = r \sin \theta$$

para $0 < \theta < 45^\circ$



9290-102-24 (rev.:00)

23

Exercícios

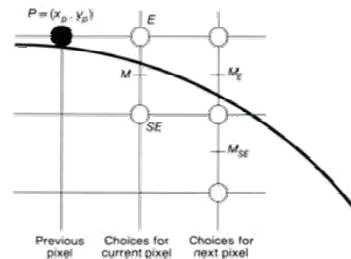
void MidPointCircle(int raio, int valor) // Fonte: Foley van Dam

// Assume centro do círculo na origem

// Algoritmo usando apenas aritmética de inteiros

```
{
    int x = 0;
    int y = raio;
    int d = 1 - raio;

    CirclePoints(x, y, valor);
    while (y > x) {
        if (d < 0) // Seleciona E
            d += 2 * x + 3;
        else { // Seleciona SE
            d += 2 * (x - y) + 5;
            y--;
        }
        x++;
        CirclePoints(x, y, valor);
    } // while
} // MidpointCircle
```



Princípio:

// Compare dist de M ao centro com R²

if M é interno ao círculo

escolha E

else escolha SE

9290-102-24 (rev.:00)

24

Exercícios

```
void CirclePoints(int X, int Y, int valor)
```

```
begin
```

```
    PutPixel(CX + X, CY + Y, valor); // ponto no octante 1
```

```
    PutPixel(CX - X, CY + Y, valor); // ponto no octante 4
```

```
    PutPixel(CX - X, CY - Y, valor); // ponto no octante 5
```

```
    PutPixel(CX + X, CY - Y, valor); // ponto no octante 8
```

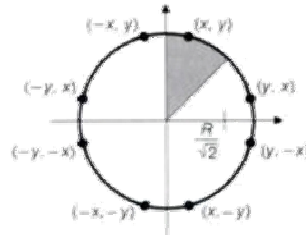
```
    PutPixel(CX + Y, CY + X, valor); // ponto no octante 2
```

```
    PutPixel(CX - Y, CY + X, valor); // ponto no octante 3
```

```
    PutPixel(CX - Y, CY - X, valor); // ponto no octante 6
```

```
    PutPixel(CX + Y, CY - X, valor) // ponto no octante 7
```

```
end; // Circle Points
```



Exercícios - Sort

- Colocar os números abaixo em ordem crescente usando um algoritmo que permita dividir a tarefa entre dois ou mais processos. Qual o algoritmo clássico mais aderente a este problema ?
- $N = \{ 737, 4, 4354, 22, 54, 742, 672, 223, 42, 3, 77, 122, 4554, 675, 1, 564, 12, 343 \}$

Exercícios - Sort



- Colocar os números abaixo em ordem crescente usando um algoritmo que permita dividir a tarefa entre dois ou mais processos. Qual o algoritmo clássico mais aderente a este problema ?
- $N = \{ 737, 4, 4354, 22, 54, 742, 672, 223, 42, 3, 77, 122, 4554, 675, 1, 564, 12, 343 \}$

Exercícios - Escalonamento



- Um casal deve realizar três tarefas: cortar a grama, passar o aspirador no chão e dar banho e trocar o bebê. Todas as três tarefas levam trinta minutos para ser realizadas e podem ser executadas por uma pessoa individualmente. Como conseguir a melhor alocação destas tarefas no tempo de tal forma a terminar o mais rápido possível ?

Muito Obrigado

UFMG

Perguntas?

Constantino Seixas Filho

constantino.seixas@task.com.br

