Introdução aos Sistemas de Operação: evolução histórica e conceitos fundamentais

José A. Cardoso e Cunha

DI-FCT-UNL

Hoje em dia...

Temos SO em diferentes tipos de máquinas:

Hoje em dia...

Temos SO em diferentes tipos de máquinas:

- ***Mainframes, minicomputadores, servidores***

Hoje em dia...

Temos SO em diferentes tipos de máquinas:

- ***Mainframes, minicomputadores, servidores***
- ***Desktops* e computadores pessoais (PC)**

Hoje em dia...

Temos SO em diferentes tipos de máquinas:

- ***Mainframes, minicomputadores, servidores***
- ***Desktops* e computadores pessoais (PC)**
- **Supercomputadores e multiprocessadores**

Hoje em dia...

Temos SO em diferentes tipos de máquinas:

- ***Mainframes, minicomputadores, servidores***
- ***Desktops* e computadores pessoais (PC)**
- **Supercomputadores e multiprocessadores**
- **Sistemas embutidos -- *Embedded* (telemóveis, PDA, câmaras digitais, etc.)**

Mas no início...?

- **Como começou? Na década de 1940...**
- **Como apareceram os SO?**

- **As arquitecturas dos computadores e os SO evoluiram em conjunto**

Operação do computador

1. **Máquina dedicada a um programa**

Operação do computador

1. **Máquina dedicada a um programa**
2. **Submissão sequencial de trabalhos**
   - ***Em lotes -- batch* (sem ou com SPOOL)**

Operação do computador

1. **Máquina dedicada a um programa**
2. **Submissão sequencial de trabalhos**
   - ***Em lotes -- batch*** **(sem ou com SPOOL)**
3. **Operação interactiva com:**
   - **Múltiplos programas**
   - **Múltiplos utilizadores**

Operação do computador

1. **Máquina dedicada a um programa**
2. **Submissão sequencial de trabalhos**
   - ***Em lotes -- batch*** **(sem ou com SPOOL)**
3. **Operação interactiva com:**
   - **Múltiplos programas**
   - **Múltiplos utilizadores**
4. **Operação em rede de computadores**

Operação do computador

Hoje podemos ter uma máquina (PC) dedicada a cada utilizador. Até ao fim dos anos 70 não era assim...

Regime de máquina dedicada

- **Hardware, sem ´´programas de sistema´´...**
  - **1 utilizador → 1 programa de cada vez**

Regime de máquina dedicada

- **Hardware, sem ´´programas de sistema´´...**
  - **1 utilizador → 1 programa de cada vez**
  - **Cada utilizador agendava um período de utilização, pago, durante o qual era o ´´dono´´ e ´´senhor´´´´da máquina**

Regime de máquina dedicada

- **Hardware, sem ´´programas de sistema´´...**
  - **1 utilizador → 1 programa de cada vez**
  - **Cada utilizador agendava um período de utilização, pago, durante o qual era o ´´dono´´ e ´´senhor´´´´da máquina**
- **Não havia quaisquer programas de apoio:**
  - **Carregamento manual dos programas binários, pelo utilizador, bit a bit**

Regime de máquina dedicada

- **Hardware, sem ´´programas de sistema´´...**
  - **1 utilizador → 1 programa de cada vez**
  - **Cada utilizador agendava um período de utilização, pago, durante o qual era o ´´dono´´ e ´´senhor´´´´da máquina**
- **Não havia quaisquer programas de apoio:**
  - **Carregamento manual dos programas binários, pelo utilizador, bit a bit**
  - **O utilizador era responsável pela programação de todas as subrotinas de entrada e saída e de carregamento em memória**

Programas carregadores:

Carregar programas e dados em memória ...

Exemplo – IBM 1460 (1963)

Exemplos de terminais antigos

Teletype 33-ASR
1965

Melhoria: Programas carregadores:

Carregar programas e dados em memória, a partir de fitas perfuradas de papel ou de cartões perfurados de papel (em vez de bit a bit manualmente...)

Melhoria: Programas carregadores:

- Carregar programas e dados em memória, a partir de fitas de papel perfuradas ou de cartões de papel perfurados (em vez de bit a bit manualmente...)
- Os programas carregadores tinham de ser carregados, eles próprios, em memória:

Melhoria: Programas carregadores:

- Carregar programas e dados em memória, a partir de fitas de papel perfuradas ou de cartões de papel perfurados (em vez de bit a bit manualmente...)
- Os programas carregadores tinham de ser carregados, eles próprios, em memória:
- Surgiu o conceito de ´´bootstrap loader´´, pequeno programa carregador: no início, é automaticamente activado e, a partir de um periférico pré-definido, carrega em memória os programas do SO

Assemblers, compiladores

- **Primeiras linguagens a níveis acima do da linguagem máquina: Assembly, FORTRAN**
- **Melhoraram a produtividade do desenvolvimento dos programas**

Bibliotecas de subrotinas para I/O

- **libertaram o utilizador da programação dos pormenores das entradas e saídas**

Bibliotecas de subrotinas para I/O

- **libertaram o utilizador da programação dos pormenores das entradas e saídas**
- **surgiram as funções genéricas de ´´read´´ e ´´write´´ sobre periféricos**

Bibliotecas de subrotinas para I/O

- libertaram o utilizador da programação dos pormenores das entradas e saídas
- surgiam as funções genéricas de ´´read´´ e ´´write´´ sobre periféricos
- bibliotecas de rotinas de controlo dos periféricos (*device drivers*)
  - Todos os programadores as usam e podem estar sempre carregadas em memória
  - começava a perceber-se a utilidade de ter ´´reads´´ e ´´writes´´ que pudessem receber, como parâmetro, a identificação do periférico específico sobre o qual operavam

Rendimento de utilização

- **Como os computadores eram muito caros, havia que optimizar o rendimento da sua utilização**
- **Optimizar:**

Rendimento de utilização

- **Como os computadores eram muito caros, havia que optimizar o rendimento da sua utilização**
- **Optimizar:**
  - **O tempo de utilização do CPU**
  -

Rendimento de utilização

- **Como os computadores eram muito caros, havia que optimizar o rendimento da sua utilização**
- **Optimizar:**
  - **O tempo de utilização do CPU**
  - **O espaço ocupado de memória**

Rendimento de utilização

- **Como os computadores eram muito caros, havia que optimizar o rendimento da sua utilização**
- **Optimizar:**
  - **O tempo de utilização do CPU**
  - **O espaço ocupado de memória**
  - **O tempo da operações de I/0**

Parâmetros de eficiência

- **Taxa de utilização dos recursos**

  Tempo utilização efectiva / tempo total

  – **De cada periférico e, principalmente, do CPU** *(o ideal seria 100%)*

Parâmetros de eficiência

- **Taxa de utilização dos recursos**

  Tempo utilização / tempo total

  – **De cada periférico e, principalmente, do CPU** *(o ideal seria 100%)*

- **Débito de trabalhos**

  Número de trabalhos executados / unid. tempo

  (quanto mais melhor)

Parâmetros de eficiência

- **Taxa de utilização dos recursos**

  Tempo utilização / tempo total

  – **De cada periférico e, principalmente, do CPU** *(o ideal seria 100%)*

- **Débito de trabalhos**

  Trabalhos executados / unid. tempo

  (quanto mais melhor)

- **Tempo de resposta para o utilizador**

  – **Tempo que decorre desde a submissão do trabalho até obter os resultados**

  (segundos? minutos? horas? dias???)

Problemas

1. **Muitos procedimentos manuais para inicialização dos periféricos e carregamento dos programas**

Problemas

1. **Muitos procedimentos manuais para inicialização dos periféricos e carregamento dos programas**
2. **Periféricos muito lentos *versus* CPU**

Problemas

1. **Muitos procedimentos manuais para inicialização dos periféricos e carregamento dos programas**
2. **Periféricos muito lentos *versus* CPU**
3. **Regime de monoprogramação: 1 só Programa Utilizador presente:**

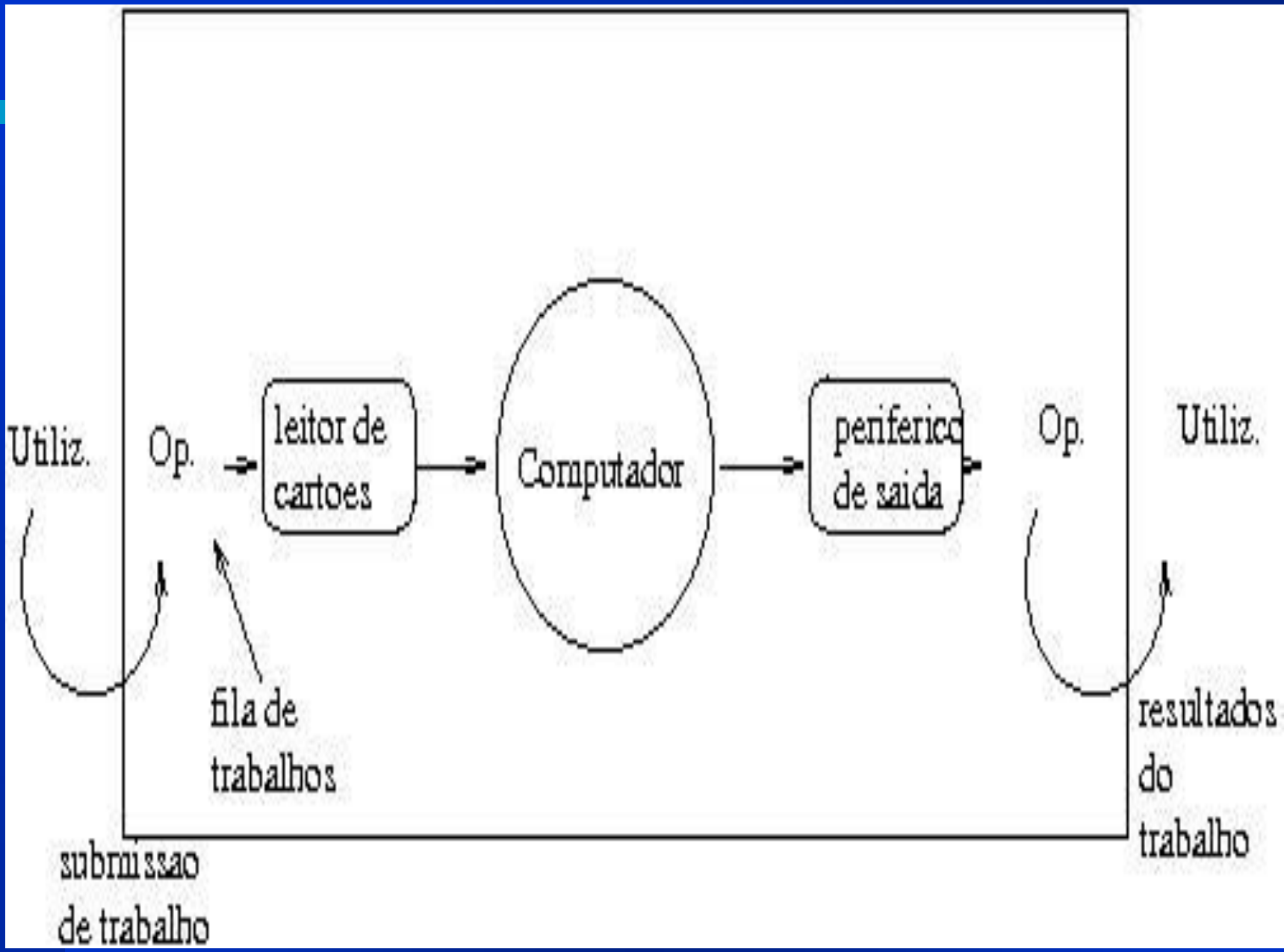
   - Qualquer operação de I/O com tempos de espera pelos dados, deixava o CPU sem ter instruções úteis para executar, durante esse tempo de espera

Processamento em lotes *(batch)*

- **Para melhorar o rendimento:**
  - **o Utilizador deixou de ter acesso directo à Máquina**

Processamento em lotes *(batch)*

- **Para melhorar o rendimento:**
  - **o Utilizador deixou de ter acesso directo à Máquina**
  - **aparece um Operador do Sistema, especialmente treinado para operar o computador e seus periféricos**


Utiliz.
Op.
leitor de cartoes
Computador
periferico de saida
Op.
Utiliz.
fila de trabalhos
resultados do trabalho
submissao de trabalho

Processamento em lotes *(batch)*

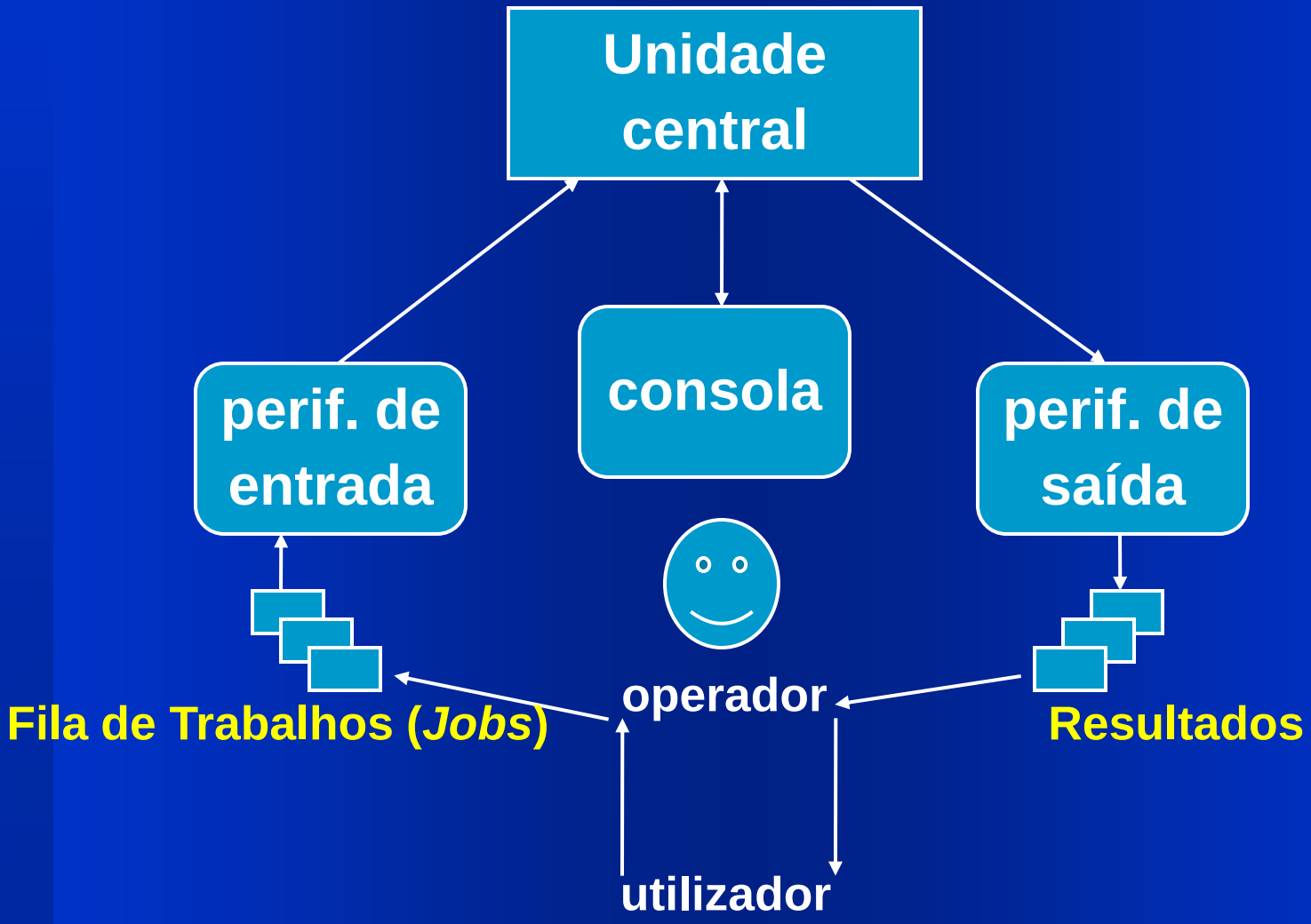
- **Para melhorar o rendimento:**
  - **o Utilizador deixou de ter acesso directo à Máquina**
  - **aparece o Operador do Sistema, especialmente treinado para operar o computador e seus periféricos**
  - **define-se a noção de ´´Trabalho´´ (*Job*): uma sequência de passos:**

    Carregar, Compilar, Ligar, Executar

    pedida por um utilizador autenticado

- **O Utilizador trabalha *off-line*, em modo não-interactivo, submetendo pedidos de trabalhos (programas e dados)**
- **Cada trabalho é descrito numa linguagem (*Job Description / Job Control Language*) reconhecida por um interpretador de comandos**

Processamento em *Batch*



- **O Operador define a ordem de execução dos trabalhos (estabelece um escalonamento) que considere justo e rentabilize a utilização da máquina**

- O Operador define a ordem de execução dos trabalhos (estabelece um escalonamento) que considere justo e rentabilize a utilização da máquina
- **Uma vez decidida essa ordem: coloca o lote de trabalhos no periférico de entrada, para serem lidos e processados, sequencialmente**

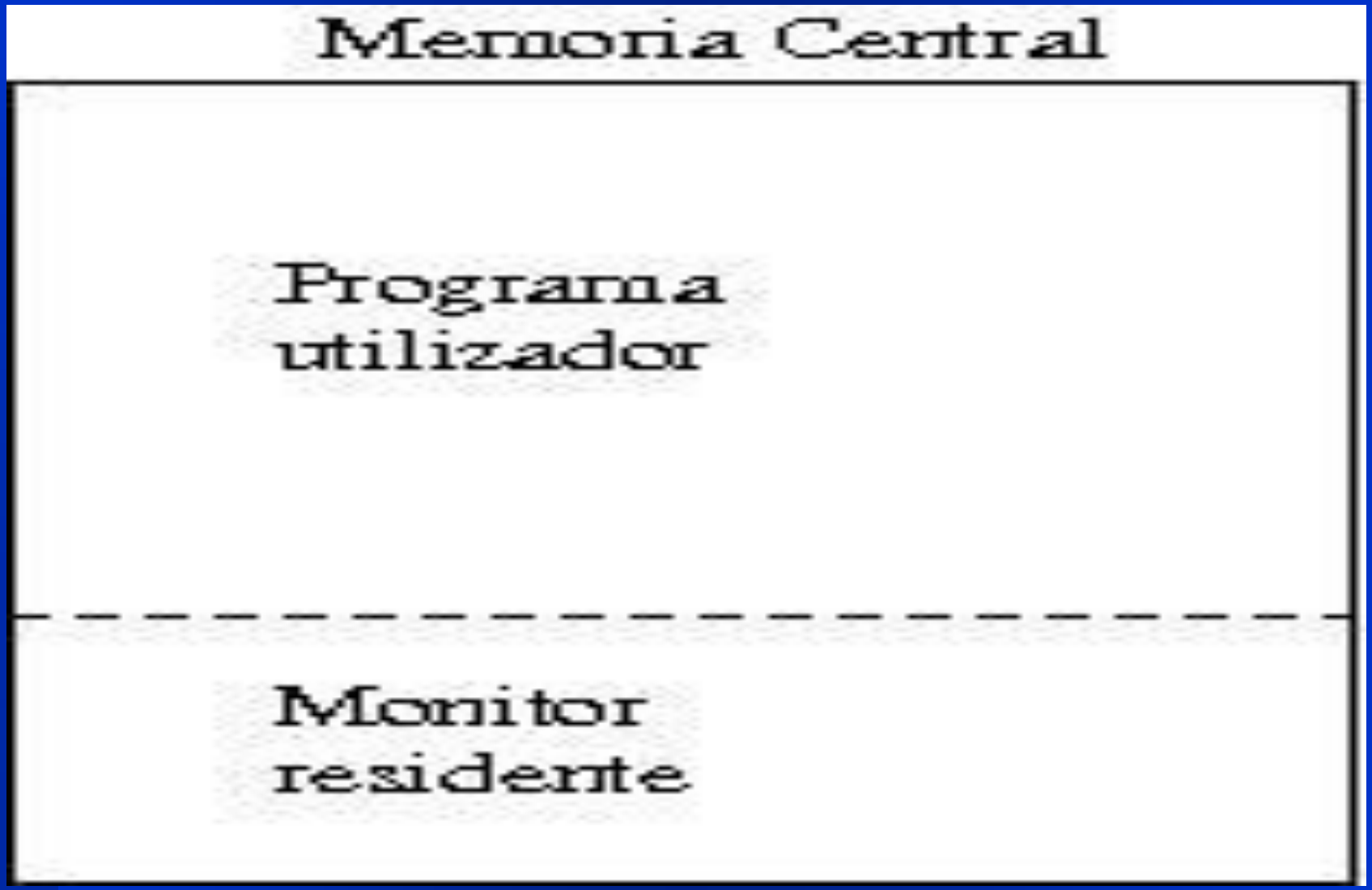
Génese dos SO

- **Para auxiliar o operador**
  - **Existe um programa (*Monitor Control Program*) que, no fim de cada trabalho, carrega o próximo trabalho e o põe em execução**

Génese dos SO

- **Para auxiliar o operador**
  - **Existe um programa (Monitor Control Program) que, no fim de cada trabalho, carrega o próximo trabalho e o põe em execução**
    - **O monitor é um <u>interpretador da linguagem de descrição de trabalhos</u>: encadeia os passos dos trabalhos em sequências automáticas de operações (compilar programa, definir os seus dados de entrada, pôr em execução, ..., passar ao próximo trabalho)**

O Monitor e o Programa…



Problemas de protecção

- **O programa utilizador tinha completo controlo da máquina:**

Problemas de protecção

- **O programa utilizador tinha completo controlo da máquina:**

1. **Por erro, podia gerar endereços fora da sua zona de trabalho em memória**

Problemas de protecção

- **O programa utilizador tinha completo controlo da máquina:**

1. **Por erro, podia gerar endereços fora da sua zona de trabalho em memória**
2. **Por erro, podia começar a ler outros trabalhos da fila de entrada**

Problemas de protecção

- **O programa utilizador tinha completo controlo da máquina:**

1. **Por erro, podia gerar endereços fora da sua zona de trabalho em memória**
2. **Por erro, podia começar a ler outros trabalhos da fila de entrada**
3. **Por erro, podia envolver-se num ciclo eterno de execução**

Ciclos infinitos dos programas

- **O problema da Terminação dos programas (´´Halting problem´´, A. Turing, 1936) → Teoria da Computação**

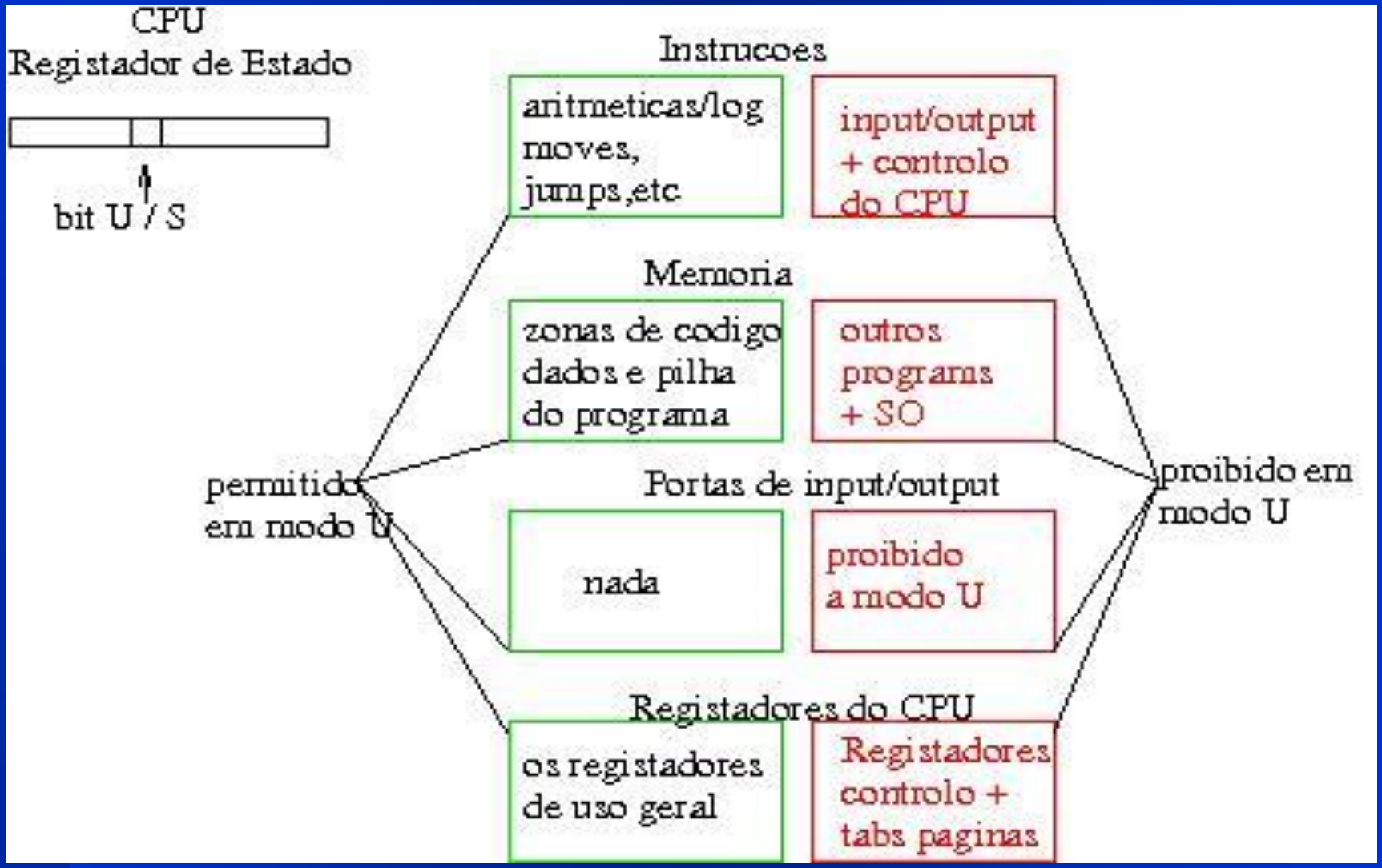
Protecção por hardware

- **Necessidade de protecção**
- **Aparecem dois modos de execução dos programas, reconhecidos pelo CPU:**
  - **O modo utilizador**
    - **Execução dos programas em modo ´´utilizador´´**
    - **Estes não podem aceder às interfaces dos periféricos nem à memória do monitor**

Protecção por hardware

- **Necessidade de protecção**
- **Aparecem dois modos de execução dos programas, reconhecidos pelo CPU:**
  - **O modo utilizador**
    - **Execução dos programas em modo ´´utilizador´´**
    - **Estes não podem aceder às interfaces dos periféricos nem à memória do monitor**
  - **Modo supervisor (ou protegido)**
    - **Para execução do monitor e das rotinas de I/O**

Modos Utilizador/Supervisor



Pedido de serviços ao SO?

- **Mas:**
  - **Se o programa utilizador deixou de ter acesso às interfaces hardware de input/output, como é que efectua as operações de entrada e saída?**

Pedido de serviços ao SO?

- **Mas:**
  - **Se o programa utilizador deixou de ter acesso às interfaces hardware de input/output, como é que efectua as operações de entrada e saída?**
- **Resposta:**
  - **Pede-as ao SO, como um cliente que pede um serviço**

Chamadas ao supervisor

- **Precisamos de uma ´´instrução máquina´´ e um mecanismo tal que:**

Chamadas ao supervisor

- **Precisamos de uma ´´instrução máquina´´ e um mecanismo tal que:**
  - **O controlo da execução do programa salte para um ponto do código do SO onde está uma subrotina**

Chamadas ao supervisor

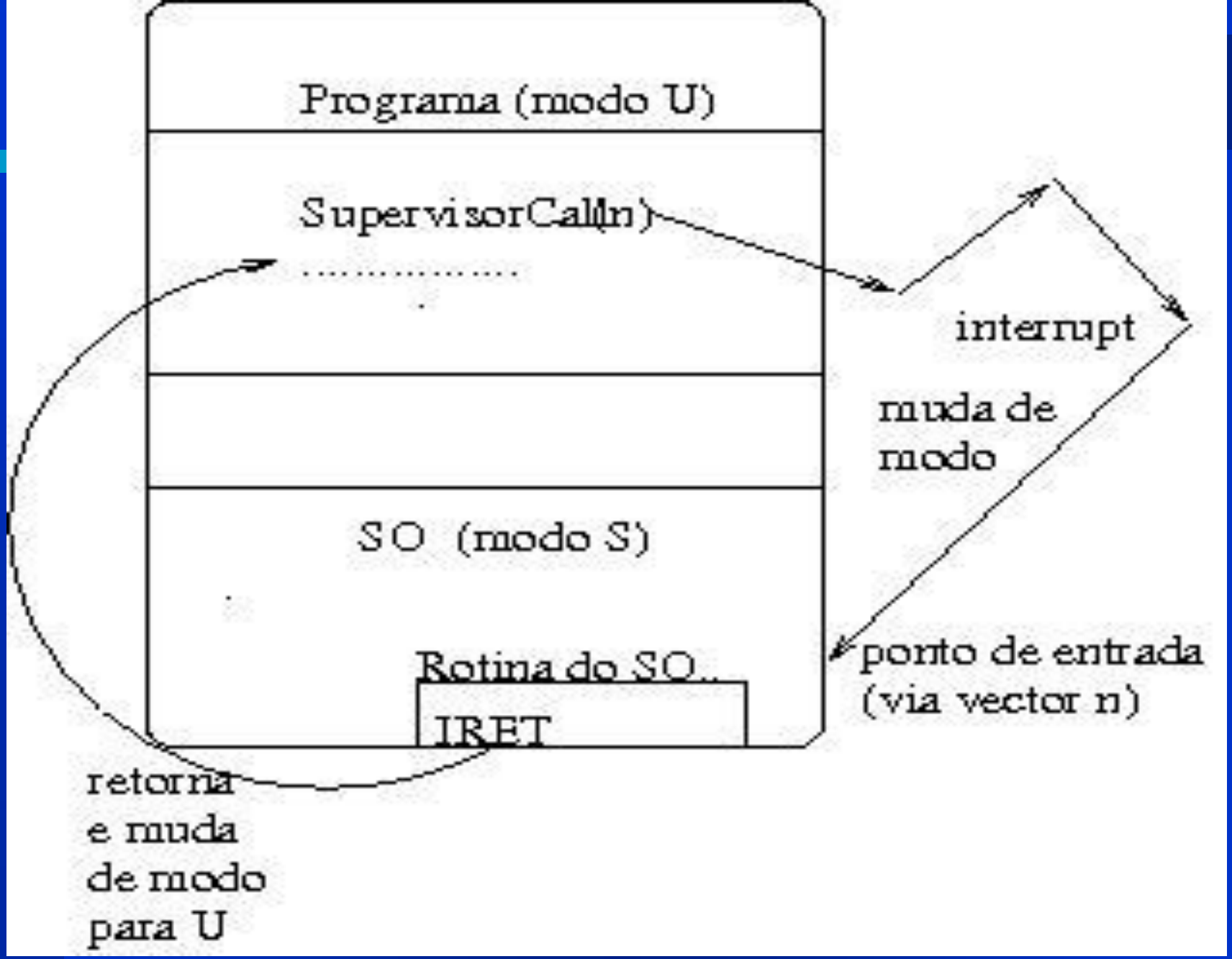
- **Precisamos de uma ´´instrução máquina´´ e um mecanismo tal que:**
  - **O controlo da execução do programa salte para um ponto do código do SO onde está uma subrotina**
  - **Não convém que o programa saiba qual o endereço da subrotina em memória**

Chamadas ao supervisor

- **Precisamos de uma ´´instrução máquina´´ e um mecanismo tal que:**
  - **O controlo da execução do programa salte para um ponto do código do SO onde está uma subrotina**
  - **Não convém que o programa saiba qual o endereço da subrotina em memória**
  - **Na chamada, altere o modo de operação do CPU para Supervisor, para dar privilégios à execução da subrotina pedida**

Chamadas ao supervisor

- **Precisamos de uma ´´instrução máquina´´ e um mecanismo tal que:**
  - **O controlo da execução do programa salte para (afectando o PC) um ponto do código do SO onde está uma subrotina**
  - **Não convém que o programa saiba qual o endereço da subrotina em memória**
  - **Na chamada, altere o modo de operação do CPU para Supervisor, para dar privilégios à execução da subrotina pedida**
  - **Ao retornar ao programa utilizador, repor o modo do CPU em Utilizador**


Programa (modo U)
SupervisorCall(n)
.....
interrupt
muda de
modo
SO (modo S)
Rotina do SO
IRET
ponto de entrada
(via vector n)
retorna
e muda
de modo
para U

Ciclos infinitos dos programas

- O problema da Terminação dos programas (´´Halting problem´´, A. Turing, 1936)
- Soluções de recurso a temporizadores: impondo ´´time-out´´ e interrompendo a execução do programa, passando controlo a uma rotina de serviço do monitor

Eficiência de utilização

- **Como os computadores eram muito caros, havia que optimizar o rendimento da sua utilização**
- **Optimizar:**
  - **O tempo de utilização do CPU**
  - **O espaço ocupado de memória**
  - **O tempo da operações de I/0**

Parâmetros de eficiência

- **Taxa de utilização dos recursos**

  tempo utilização / tempo total

  – **De cada periférico e, principalmente, do CPU** *(o ideal seria 100%)*

- **Débito de trabalhos**

  trabalhos executados / unid. tempo

  (quanto mais melhor)

- **Tempo de resposta para o utilizador**

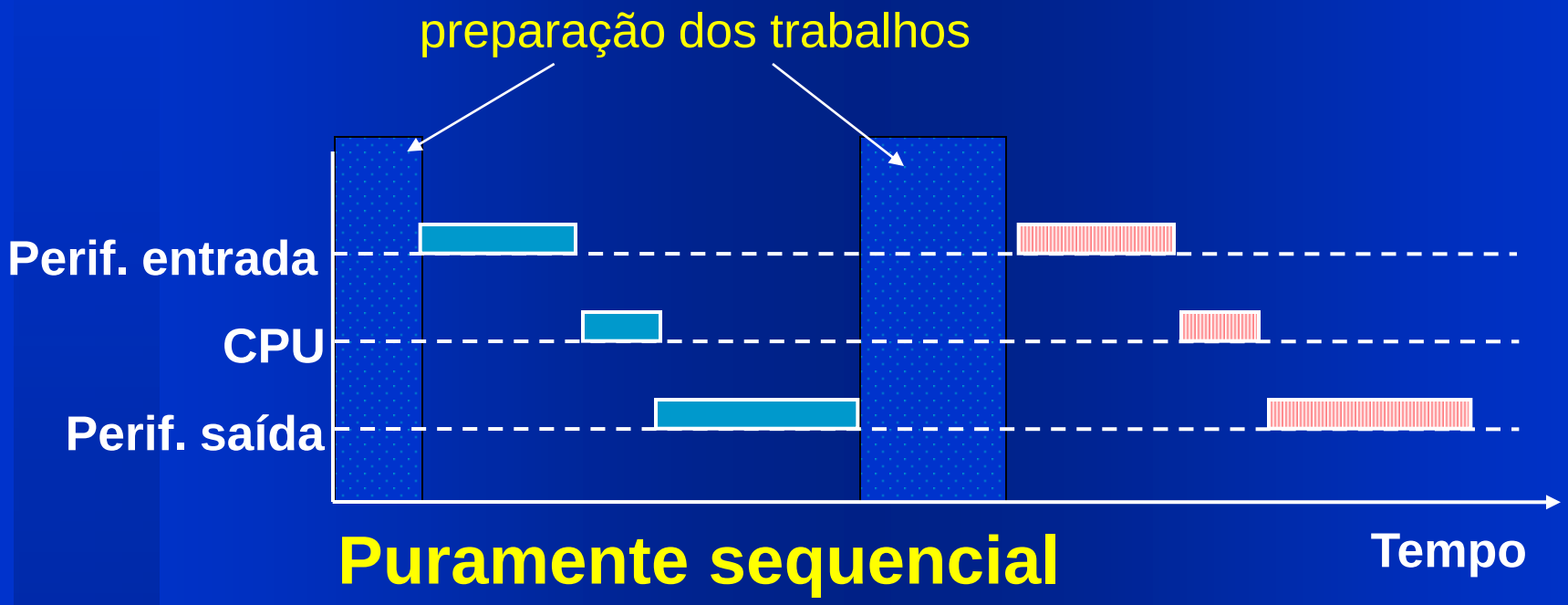
  – **Tempo que decorre desde a submissão do trabalho até obter os resultados**

  (minutos? horas? dias???)

Exemplo – IBM 1460 (1963)


Operador
Computador
ler instrucoes
inicializar perifericos
controlar switches
do painel da consola
activar execucao
Executar
terminou normal?
desmontar unidades
perifericas
seleccionar proximos
trabalhos
activar execucao
Executar
TEMPO

Processamento sequencial



Puramente sequencial

Interrupções e *buffers* de I/O

- **Permitir concorrência entre acções de I/O e a computação executada pelo CPU:**
  - **O programa escreve num *buffer* em memória**

Interrupções e *buffers* de I/O

- **Permite-se concorrência entre acções de I/O e a computação executada pelo CPU:**
  - **O programa escreve num *buffer* em memória**
  - **A rotina de atendimento da impressora vai lendo do *buffer* e desencadeia a acção de impressão**

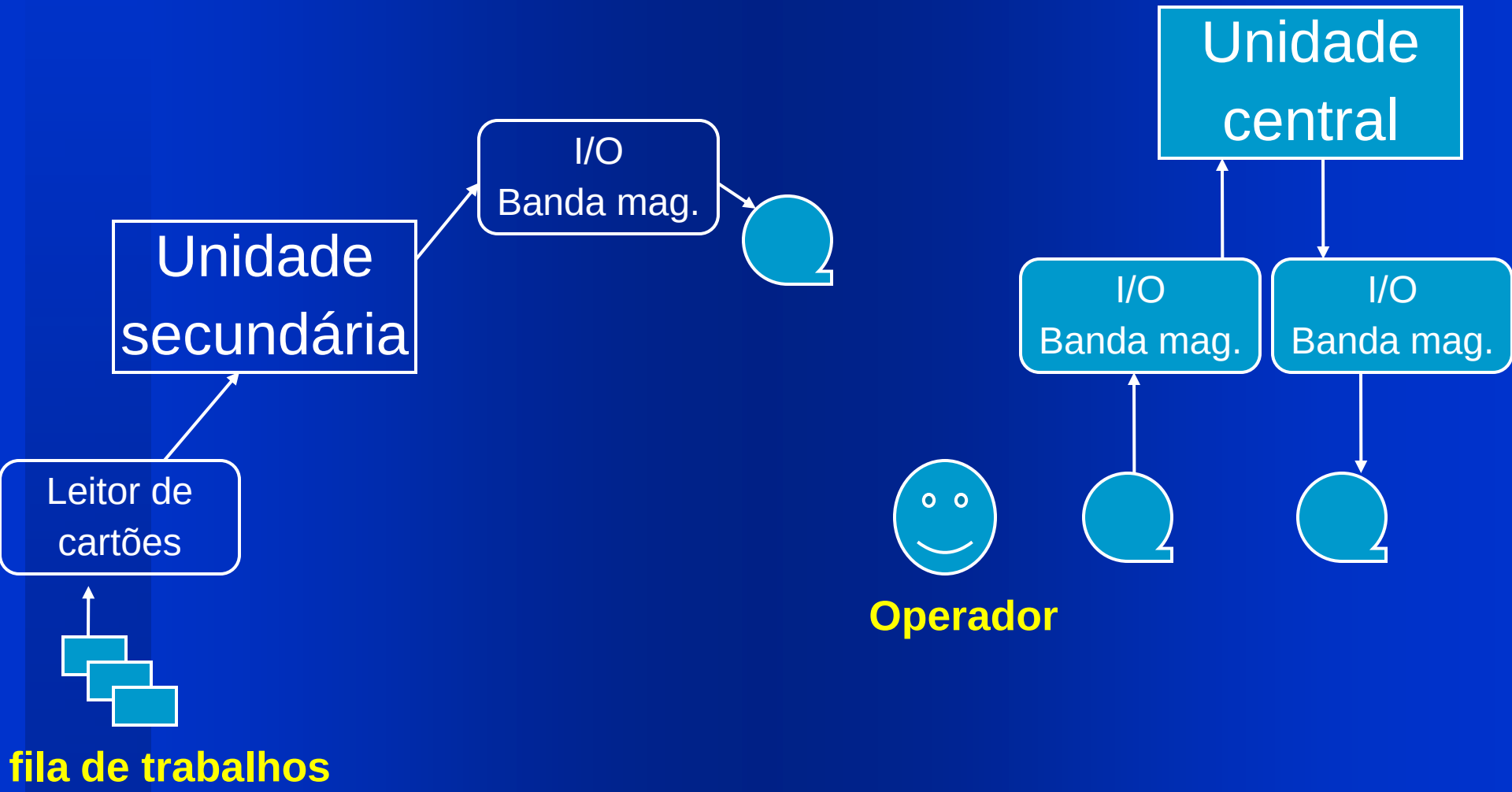
Interrupções e *buffers* de I/O

- **Permite-se concorrência entre acções de I/O e a computação executada pelo CPU:**
  - **O programa escreve num *buffer* em memória**
  - **A rotina de atendimento da impressora vai lendo do *buffer* e desencadeia a acção de impressão**
  - **Quanto mais autónomo o controlador da impressora, menos trabalho para a rotina de atendimento → menos sobrecarga para CPU**
- **Mas os *buffers* têm tamanho limitado e as operações físicas de I/O continuam lentas...**

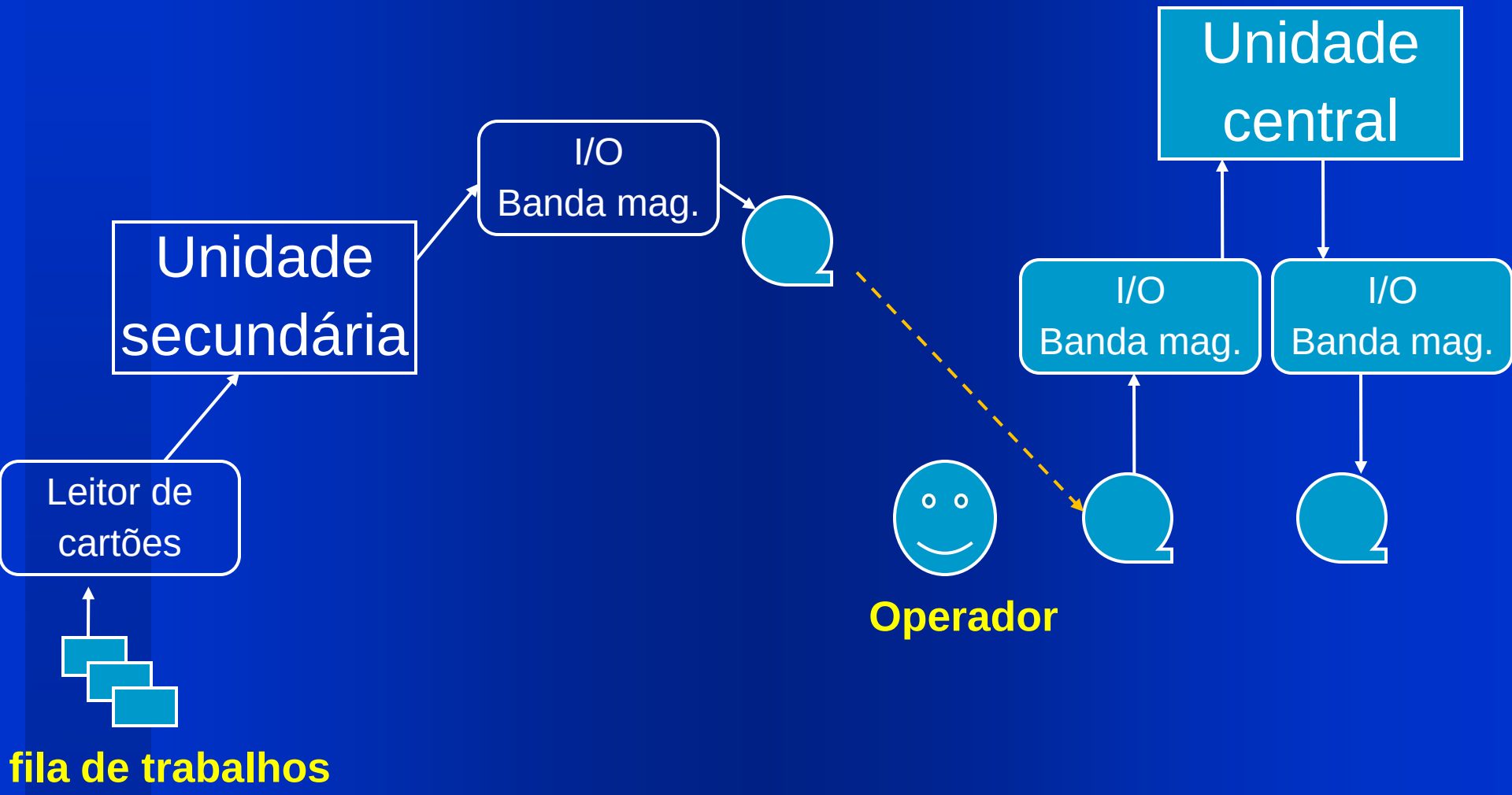
Uma tentativa...

- **Utilizar 2 processadores!**
- **Foi uma solução adoptada pela IBM e muito utilizada nas primeiras instalações comerciais de computadores -- década de 1960...**

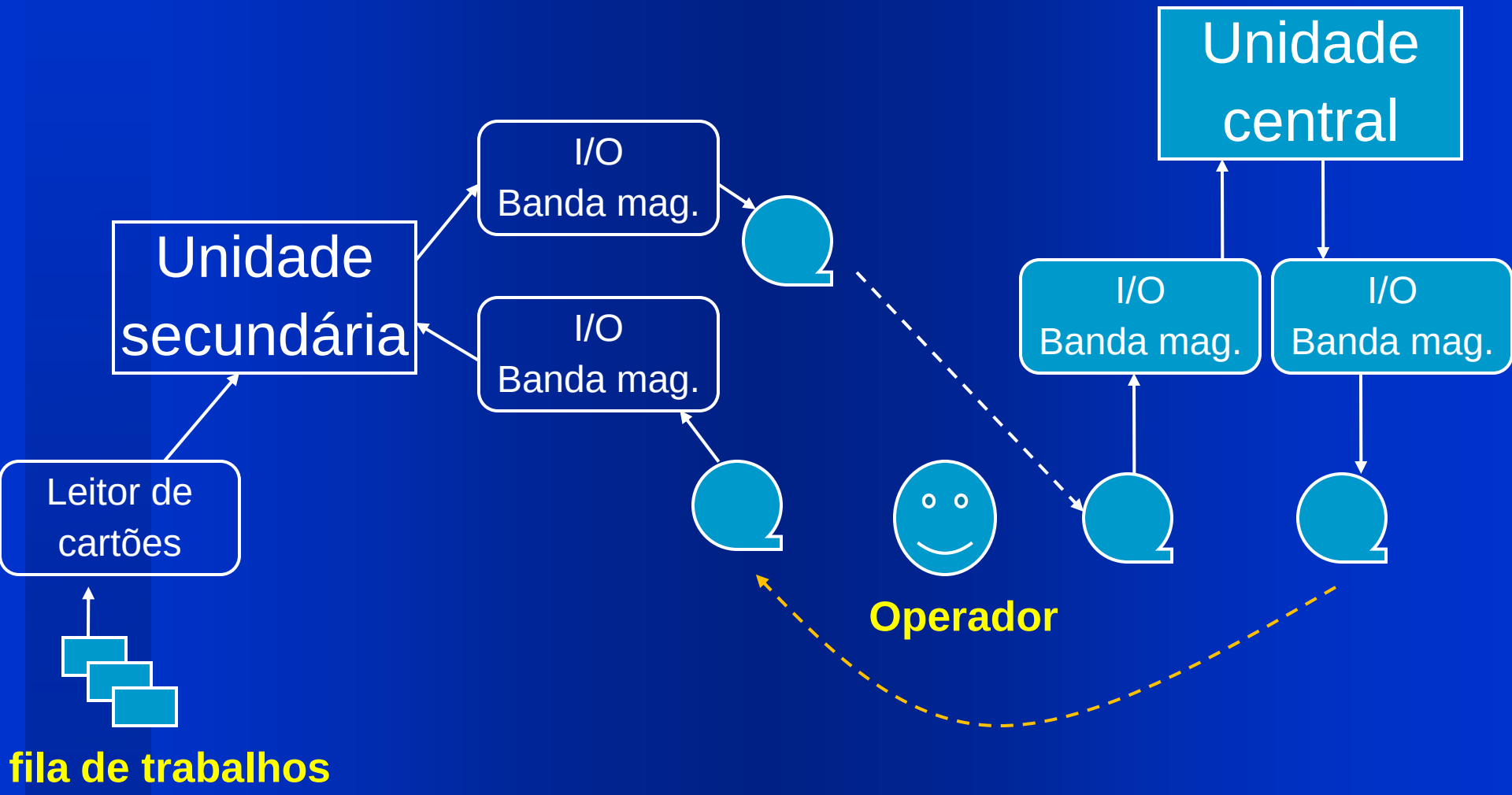
Processamento do I/O em off-line



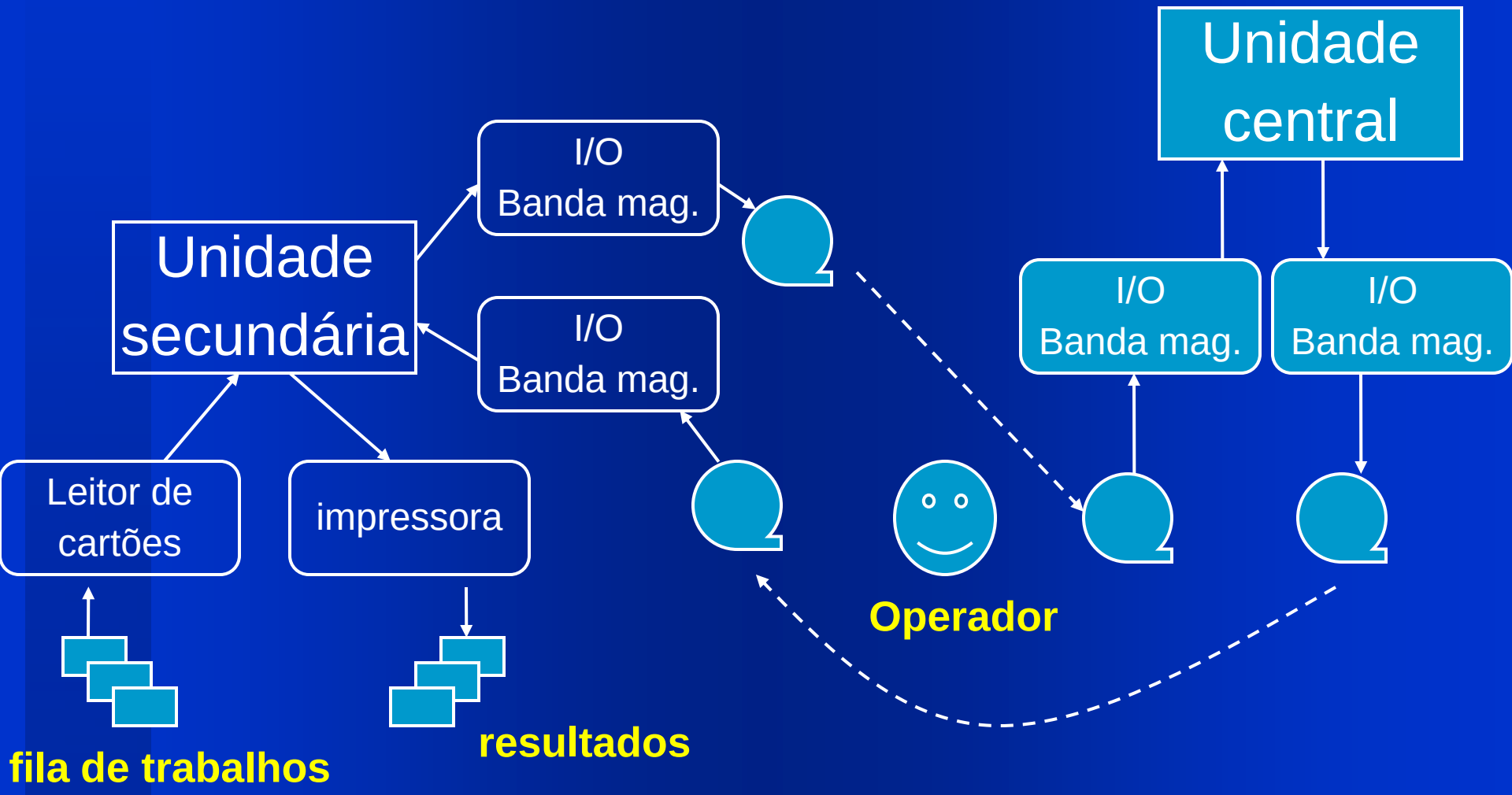
Processamento do I/O em off-line

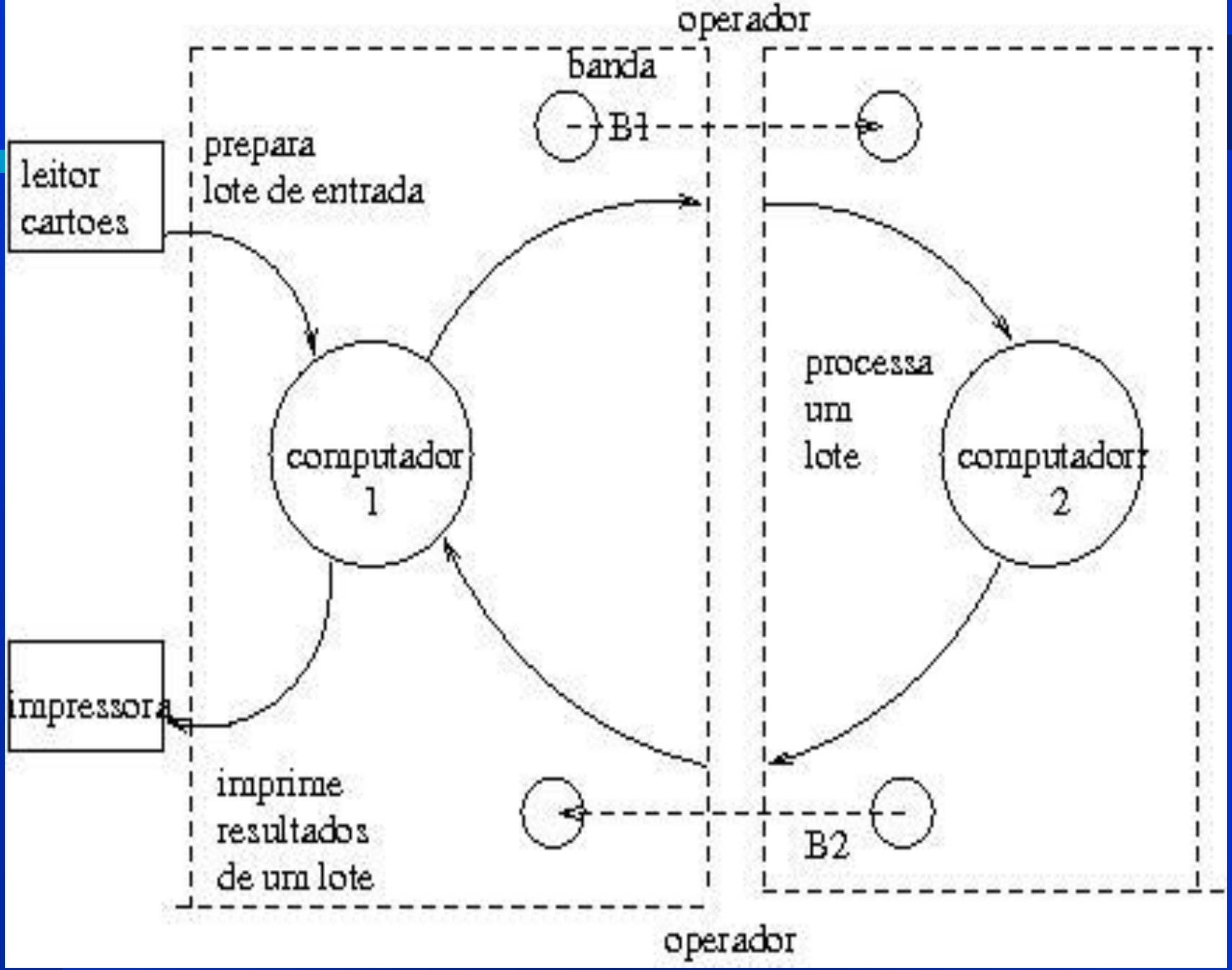


Processamento do I/O em off-line



Processamento do I/O em off-line




operador
banda
B1
leitor
cartoes
prepara
lote de entrada
processa
um
lote
computador
1
computador
2
impressora
imprime
resultados
de um lote
B2
operador

Lotes (batch) de trabalhos

- **Periféricos mais lentos operam *off-line***
- **Lotes de trabalhos em Bandas Magnéticas (mais rápidas do que Leitor de Cartões)**
- **Acções de I/O (lentas), em paralelo com a execução do programa**
- **Um segundo computador dedicado ao I/O (mais simples e barato do que o principal)**

Limitações deste regime

- **Há que montar / desmontar as bandas: → juntar vários trabalhos para fazer lotes, cujo tamanho compense:**
  - **Se tempo montar/desmontar = 10 min.**

Limitações

- **Há que montar/desmontar as bandas: → juntar vários trabalhos para fazer lotes, cujo tamanho compense:**
  - **Se tempo montar/desmontar = 10 min.**

    lote = 10 trabalhos → sobrecarga= 1 min/trab.

Limitações

- **Há que montar/desmontar as bandas: → juntar vários trabalhos para fazer lotes, cujo tamanho compense:**
  - **Se tempo montar/desmontar = 10 min. = 600 seg.**

    lote = 10 trabalhos → sobrecarga= 1 min/trab.

    lote = 100 trabalhos →sobrecarga= 6 seg/trab.

Limitações

- **Há que montar/desmontar as bandas: → juntar vários trabalhos para fazer lotes, cujo tamanho compense:**
  - **Se tempo montar/desmontar = 10 min.**

    lote = 10 trabalhos → sobrecarga= 1 min/trab.

    lote = 100 trabalhos →sobrecarga= 6 seg/trab.
- **O processamento continua sequencial**
  - **Uma fila de trabalhos nos lotes nas bandas magnéticas, mas regime de monoprogramação**

  → *O CPU é mal utilizado durante a espera pelo fim das operações de I/O*

Acesso sequencial à banda magnética

- **A ordem de execução ficava determinada pela ordem dos trabalhos na banda:**

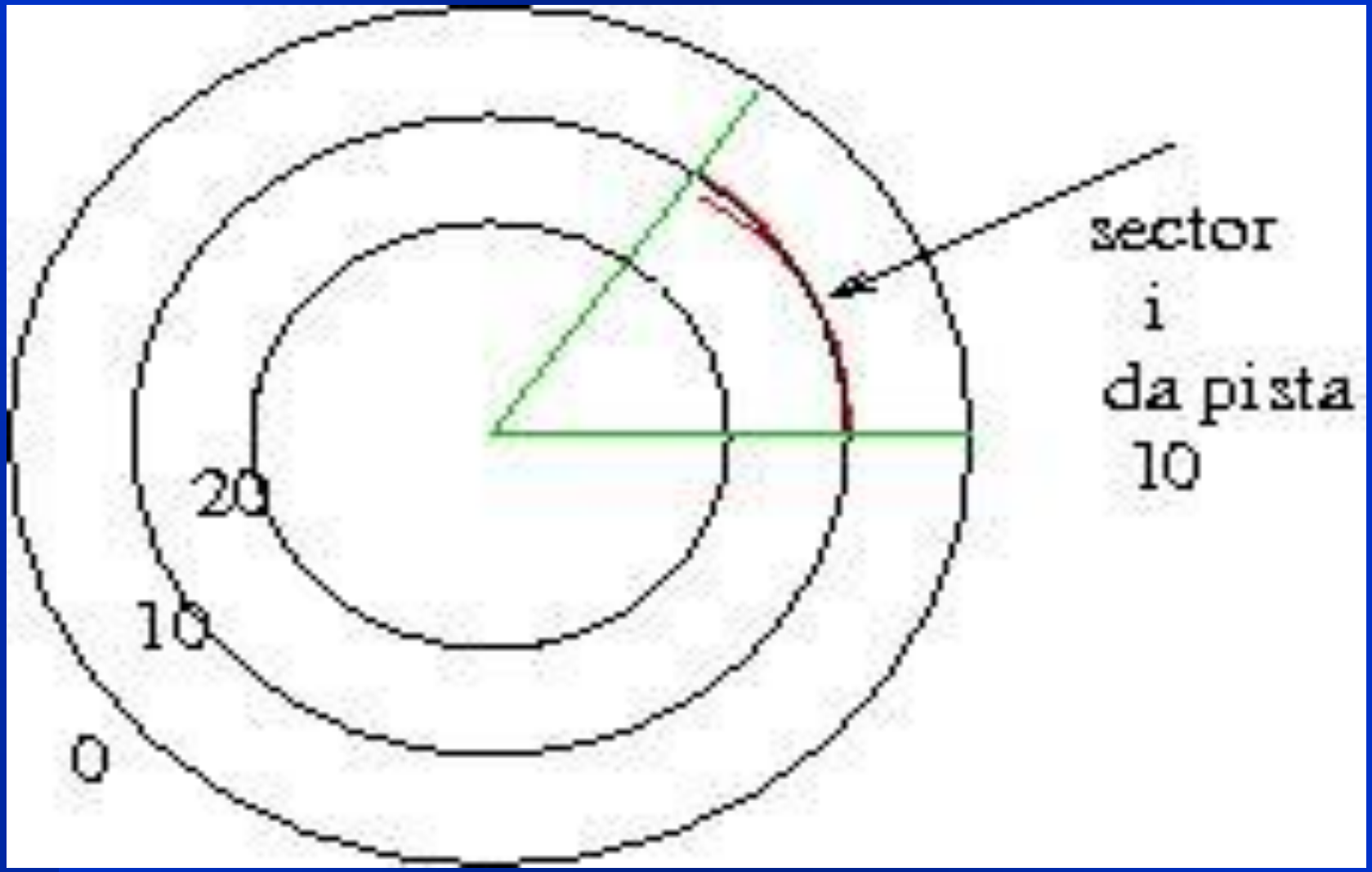
Situações de ´´injustiça´´

- *um trabalho ´´muito rápido´´ que fica atrás de um trabalho ´´muito demorado´´...*
- *só no fim de todo o lote é que a banda, com os resultados, é carregada, para impressão pelo processador de I/O*
- **o operador, quando muito, podia reordenar os trabalhos que coloca no leitor de cartões, para construir um lote mais "justo“, segundo prioridades**

Dispositivos de acesso directo

- **Os discos**
  - **Permitem o armazenamento dos trabalhos**
  - **Mais rápidos do que as bandas**
- **São melhores para manter a fila de pedidos**
  - **Permitem o acesso directo a qualquer pista onde estivessem os dados de cada trabalho**

Acesso directo ao disco



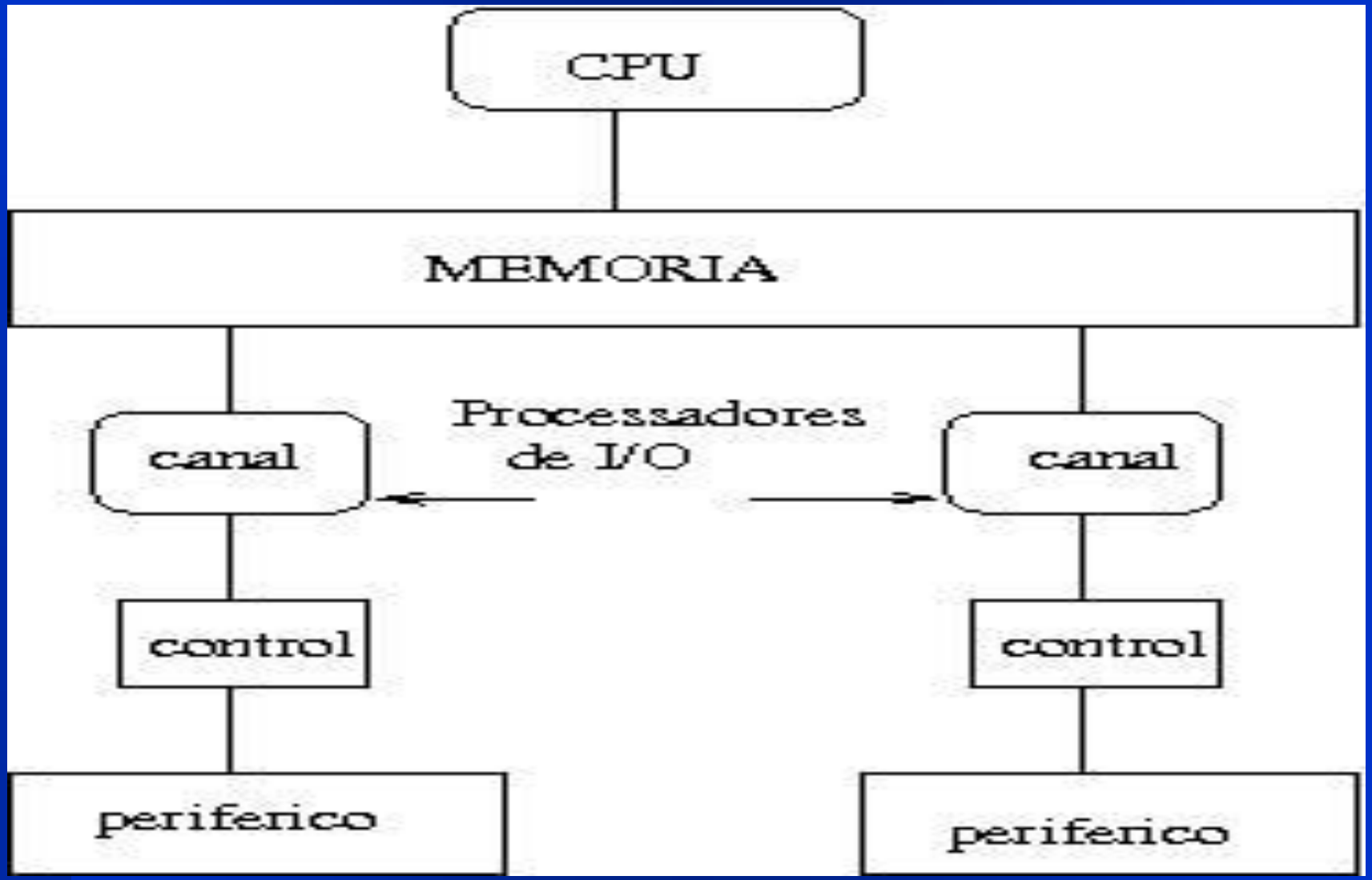
Operação *on-line*

E com Direct Memory Access (DMA) ...

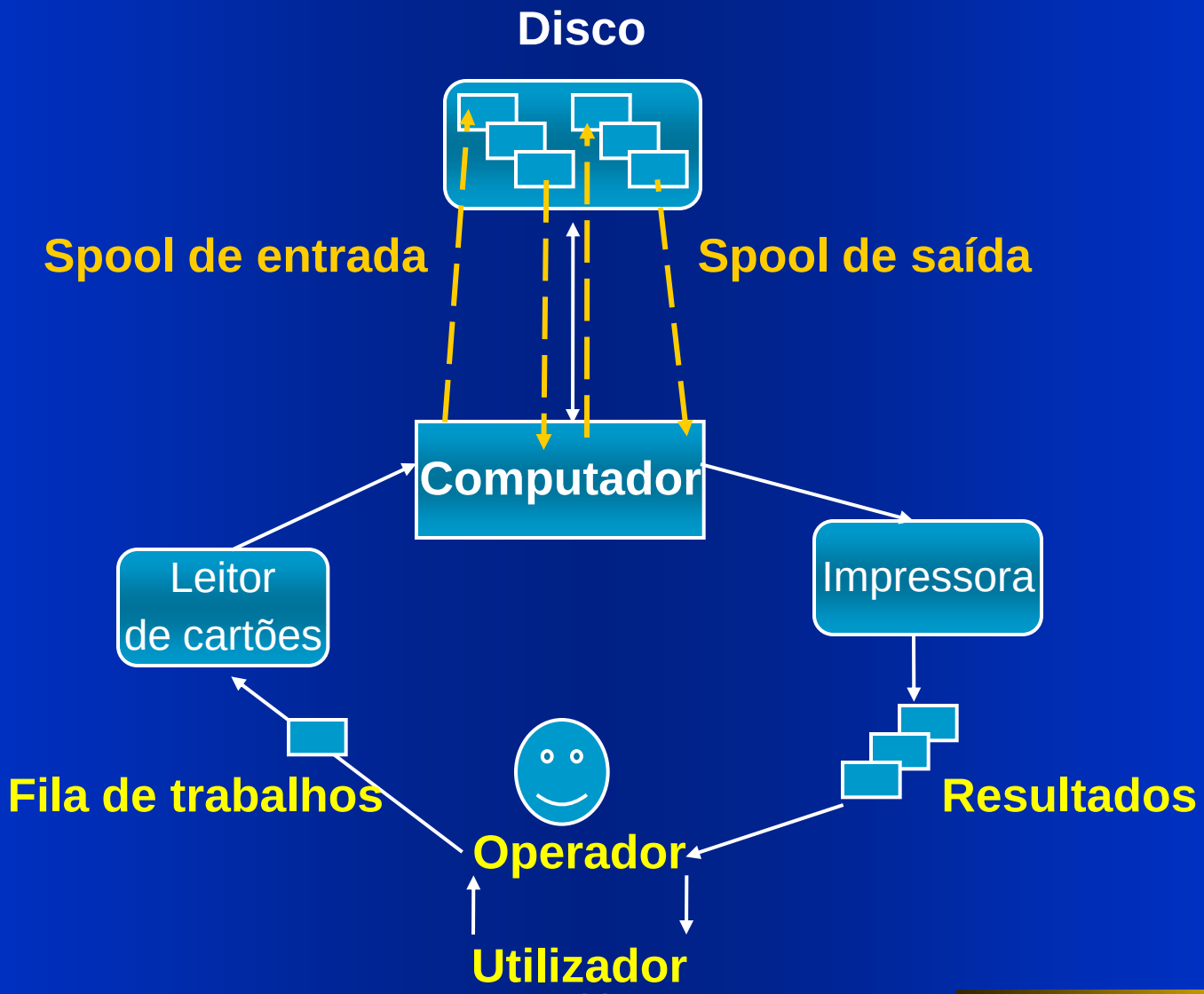
Permite-se operação on-line (o disco está continuamente ligado ao computador, não sendo preciso perder tempo na montagem / desmontagem manual)

- **Não há necessidade de esperar pelo fim de todos os trabalhos dos lotes**

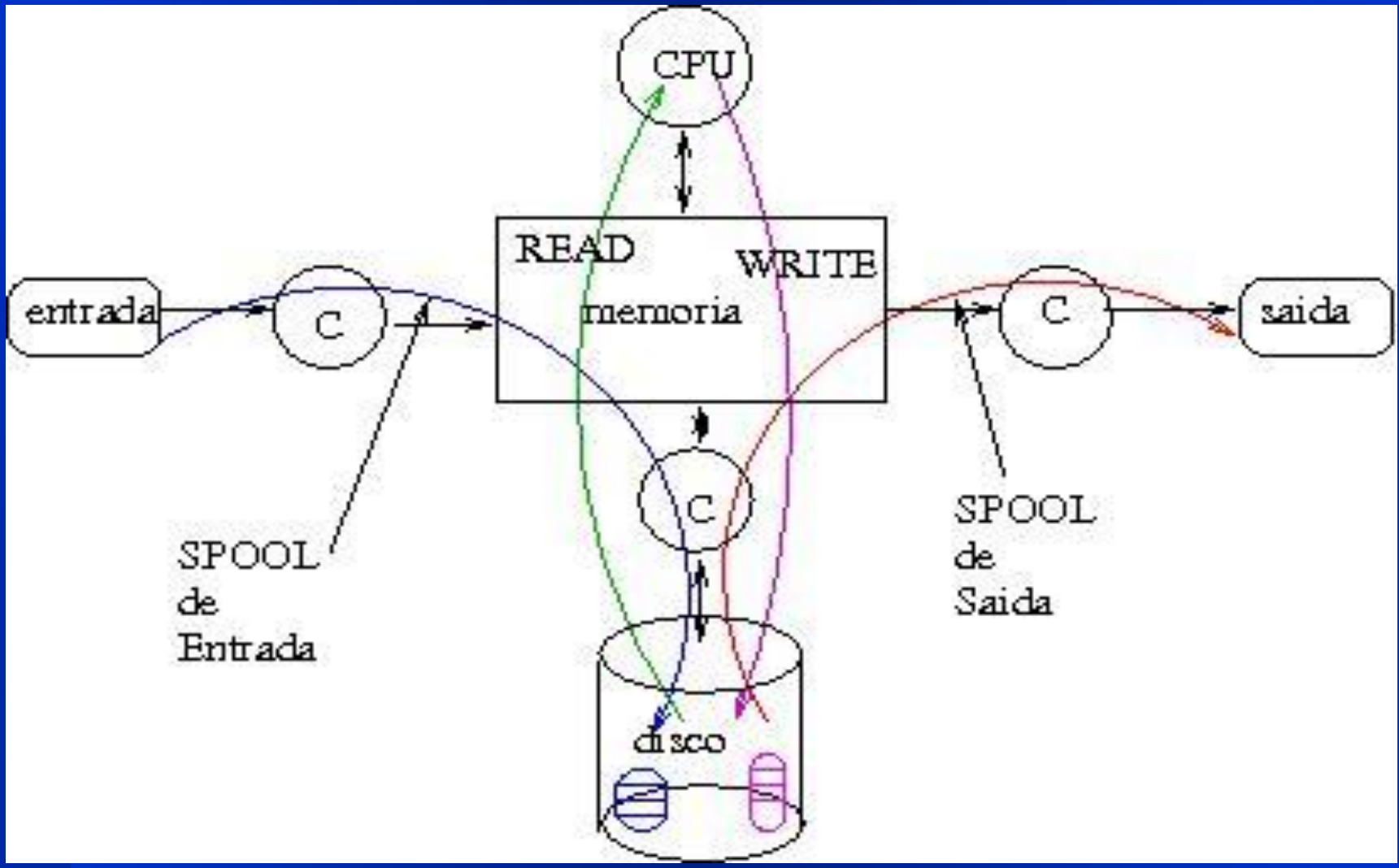
Processadores de I/O: DMA+++



Processamento com SPOOL



Operações simultâneas I/O -- CPU



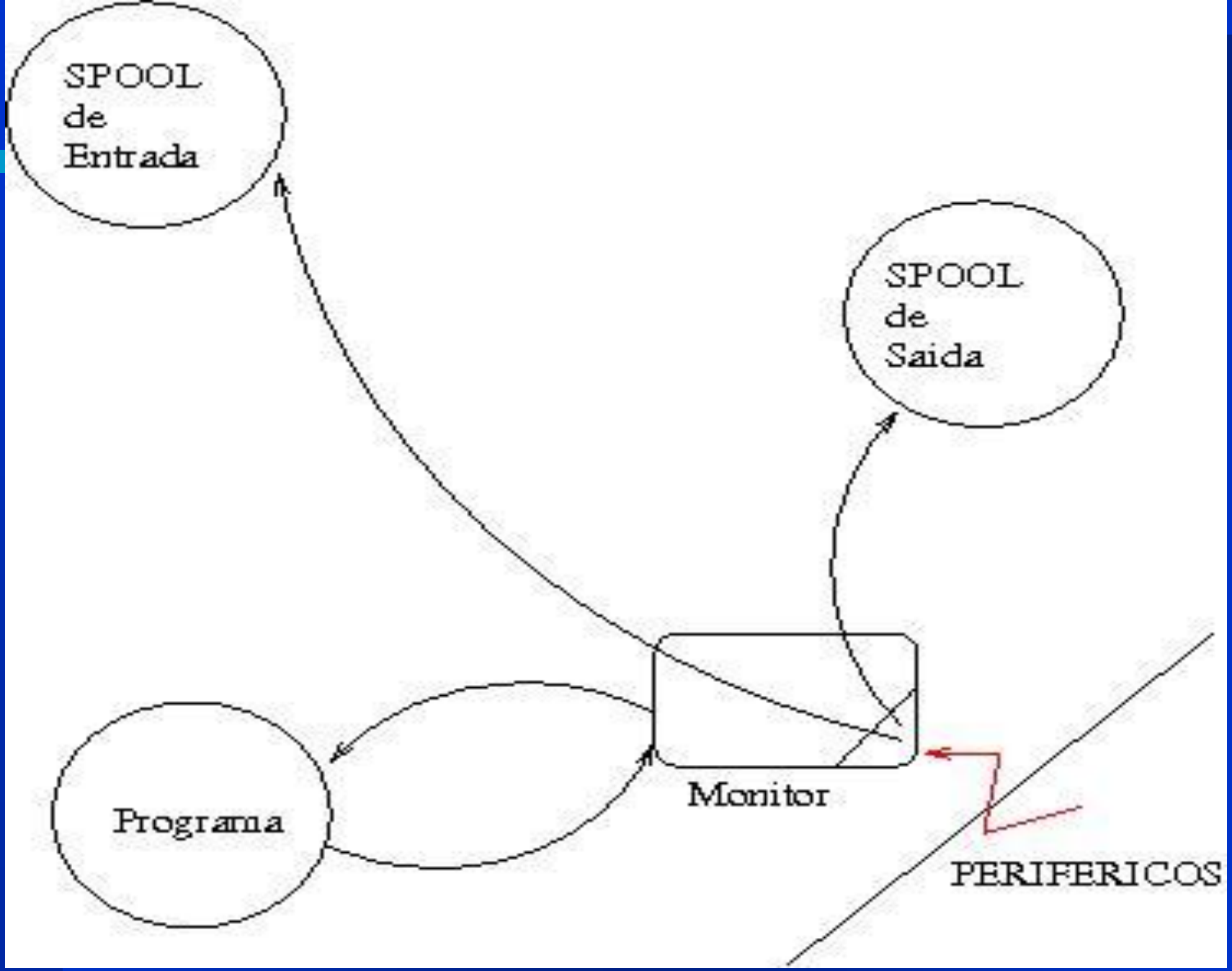
Simultaneous
Peripheral
Operation
On-
Line

Simultaneous Peripheral Operation On-Line

- "periféricos on-line"
  - Os periféricos ligados ao computador através de controladores que libertam o CPU (p.ex. c/ DMA)

Simultaneous Peripheral Operation On-Line

- "periféricos on-line"
  - Os periféricos ligados ao computador através de controladores que libertam o CPU (p.ex. c/ DMA)
- "operações em simultâneo":
  - Executar um programa
  - Ler trabalhos para uma fila em disco
  - Imprimir resultados a partir de uma fila em disco


SPOOL
de
Entrada
SPOOL
de
Saida
Programa
Monitor
PERIFERICOS

SPOOL (cont.)

- **O disco (usado como memória secundária):**
  - **Fila de trabalhos (programas e seus dados)**
  - **Fila de resultados**

SPOOL (cont.)

- O disco (usado como memória secundária):
  - Fila de trabalhos (programas e seus dados)
  - Fila de resultados
- O SO contém as rotinas para controlo:
  - Leitura dos periféricos (→ para o disco)
  - Escrita nos periféricos (→ a partir do disco)
  - Monitor / controlador da sequência de trabalhos

SPOOL (cont.)

- O disco (usado como memória secundária):
  - Fila de trabalhos (programas e seus dados)
  - Fila de resultados
- O SO contém as rotinas para controlo:
  - Leitura dos periféricos (→ para o disco)
  - Escrita nos periféricos (→ a partir do disco)
  - Monitor / controlador da sequência de trabalhos
- O SO gere as entradas e as saídas em curso, em concorrência com a execução do programa corrente
  - Os programas não podem aceder directamente aos periféricos

SPOOL (cont.)

- **É necessário suporte do hardware:**

SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**

SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**
  - **Quando necessário, o controlador gera uma interrupção, suspendendo a execução do Programa e passando o CPU a executar uma Rotina do SO**

SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**
  - **Quando necessário, o controlador gera uma interrupção, suspendendo a execução do Programa e passando o CPU a executar uma Rotina do SO**
    - **Para receber os próximos dados;**

SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**
  - **Quando necessário, o controlador gera uma interrupção, suspendendo a execução do Programa e passando o CPU a executar uma Rotina do SO**
    - **Para receber os próximos dados;**
    - **Para enviar os próximos dados;**

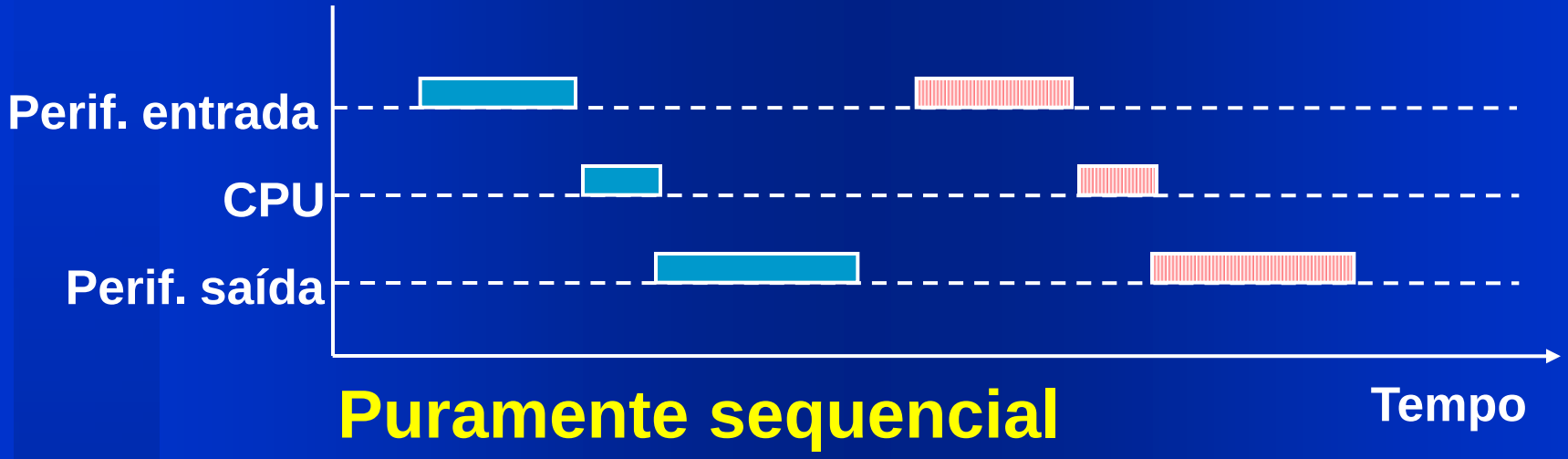
SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**
  - **Quando necessário, o controlador gera uma interrupção, suspendendo a execução do Programa e passando o CPU a executar uma Rotina do SO**
    - **Para receber os próximos dados;**
    - **Para enviar os próximos dados;**
    - **Para parar o I/O;**

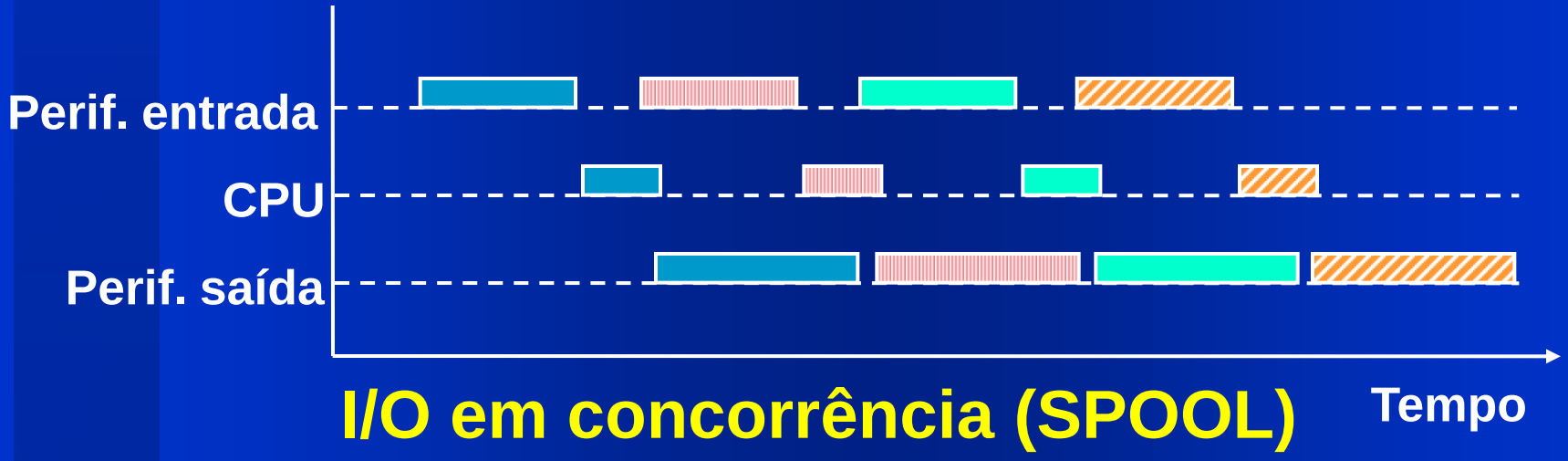
SPOOL (cont.)

- **É necessário suporte do hardware:**
  - **Cada periférico dispõe de um controlador com alguma autonomia**
  - **Quando necessário, o controlador gera uma interrupção, suspendendo a execução do Programa e passando o CPU a executar uma Rotina do SO**
    - **Para receber os próximos dados;**
    - **Para enviar os próximos dados;**
    - **Para parar o I/O;**
    - **Etc...**
- **Concorrência entre:**
  - **leitor, escritor e o programa do utilizador**
  - **controlada pelo monitor**

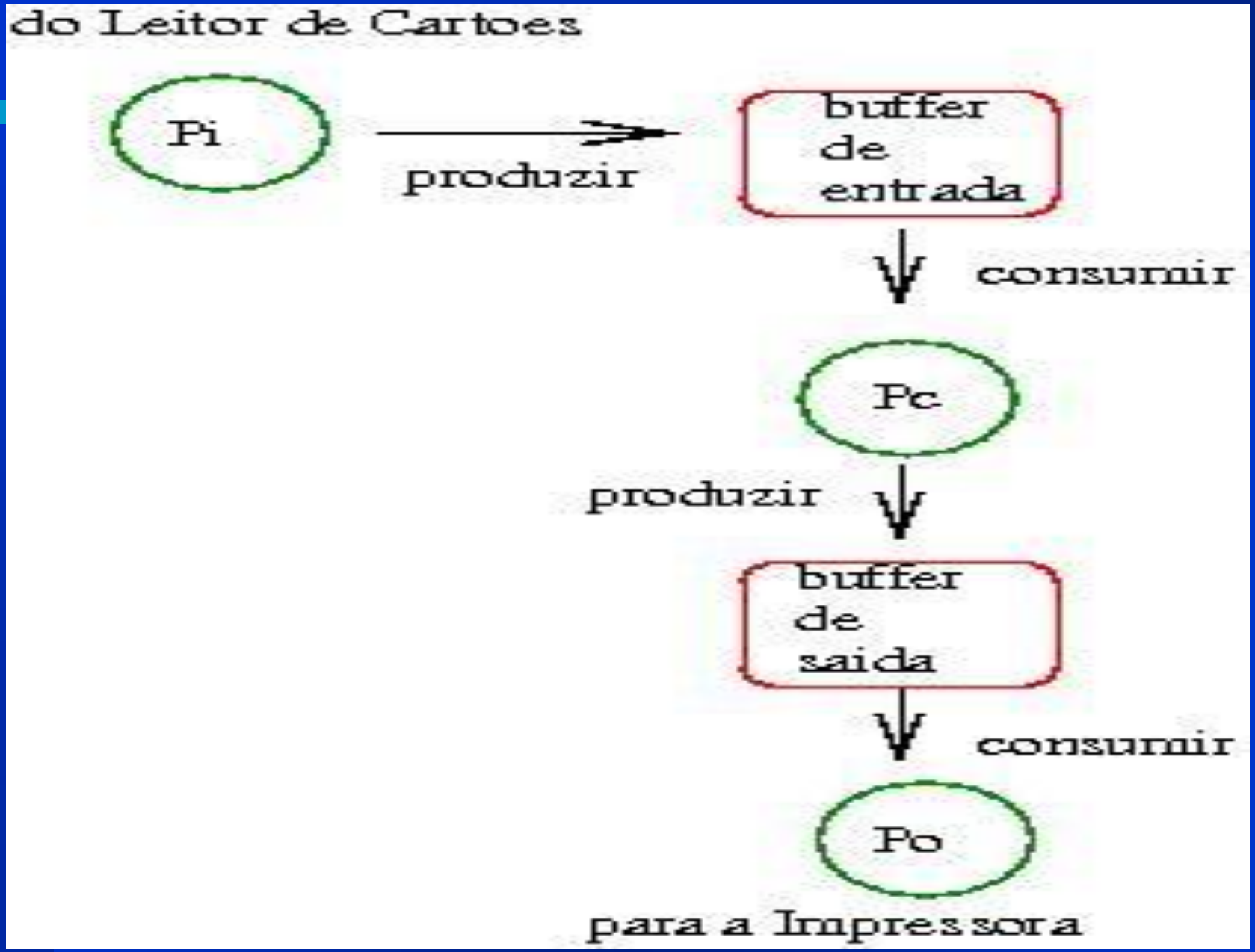
Utilização do CPU e periféricos



Puramente sequencial



I/O em concorrência (SPOOL)


do Leitor de Cartoes
Pi
produzir
buffer de entrada
consumir
Pc
produzir
buffer de saida
consumir
Po
para a Impressora

Limitações do SPOOL

- **regime de Monoprogramação: continua a haver tempos em que o CPU não é bem utilizado, porque só há um programa presente no computador**

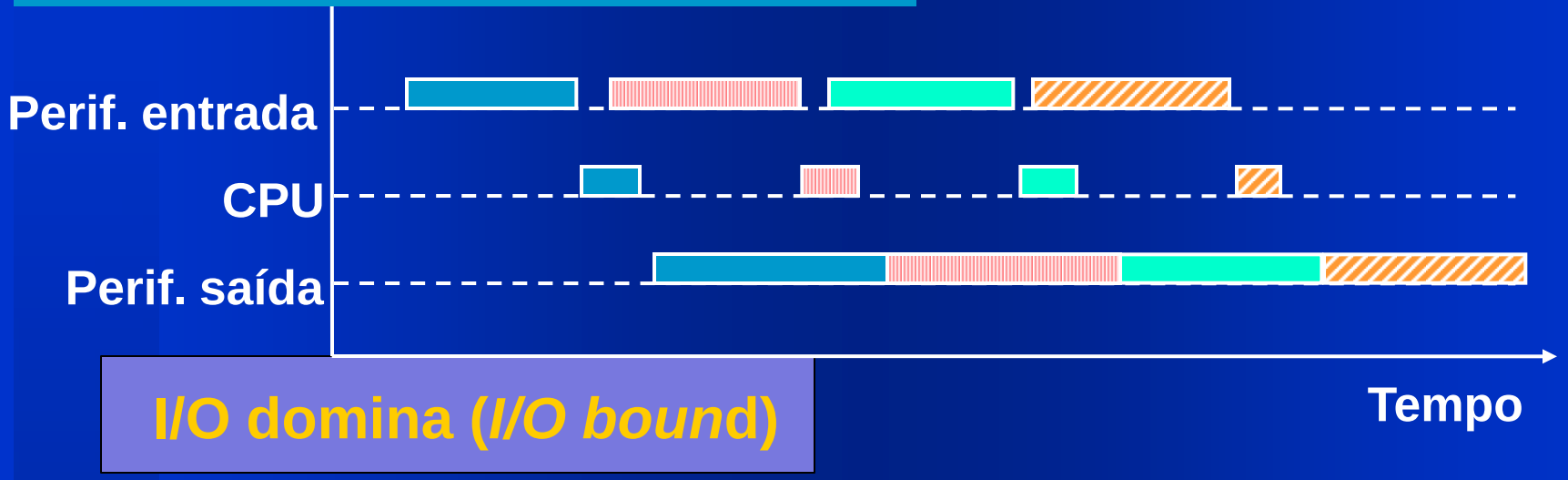
Limitações do SPOOL

- **regime de Monoprogramação: continua a haver tempos em que o CPU não é bem utilizado, porque só há um programa presente no computador**
- **Programa *I/O bound*: os tempos das operações de I/O dominam o tempo total**

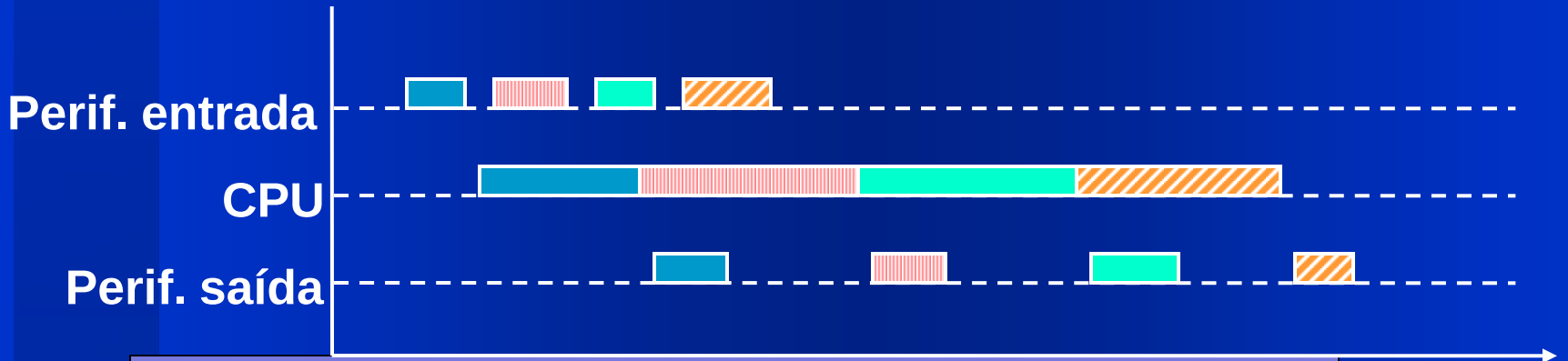
Limitações do SPOOL

- **regime de monoprogramação: continua a haver tempos em que o CPU não é bem utilizado, porque só há um programa presente no computador**
- **Programa *I/O bound*: os tempos das operações de I/O dominam o tempo total**
- **Programa *CPU bound*: os tempos da execução de instruções pelo CPU dominam o tempo total de execução**

Perfis dos programas



I/O domina (*I/O bound*)



Processamento CPU domina (*CPU bound*)

Problema

- **Em geral não se pode, *a priori*, prever o perfil de comportamento de um programa**
  - **O perfil pode variar entre execuções e durante a mesma execução, dependendo dos dados submetidos...**

Problema

- **Em geral não se pode, *a priori*, prever o perfil de comportamento de um programa**
  - **O perfil pode variar entre execuções e durante a mesma execução, dependendo dos dados submetidos...**
- **Os programas tipicamente alternam fases dominadas por operações de I/O e fases dominadas pela execução de instruções no CPU**
- **Programas interactivos *e* programas de simulação...**

Quais os Objectivos?

- **Boa utilização dos periféricos?**
  - **Um fluxo regular de I/O, independente do programa corrente**

- **Boa utilização dos periféricos?**
  - **Um fluxo regular de I/O, independente do programa corrente**
  - **Baseado em concorrência, gerida por meio de interrupções/DMA e usando *buffers* de entrada e de saída**

Quais os Objectivos?

- **Boa utilização do CPU?**
  - **Libertar o CPU, o mais possível, do controlo do I/O: dando autonomia ao controlo das interfaces (interrupções, DMA, processadores dedicados de I/O)**

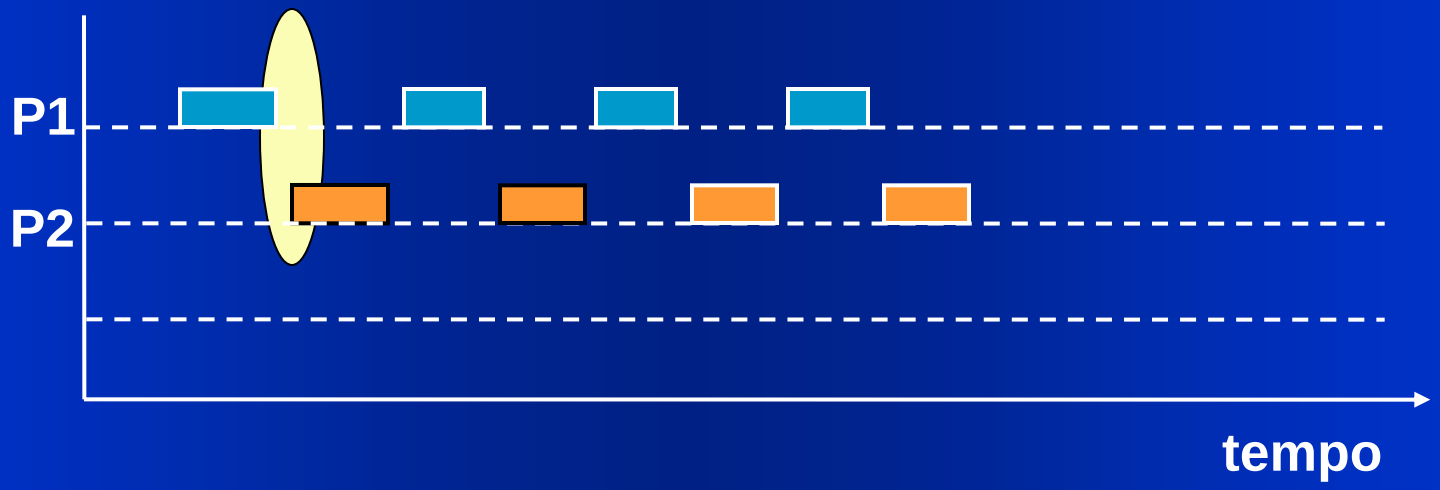
- **Boa utilização do CPU?**
  - **Libertar o CPU, o mais possível, do controlo do I/O: dando autonomia ao controlo das interfaces (interrupções, DMA, processadores dedicados de I/O)**
  - **Mas, se só há um programa presente no sistema de cada vez, torna-se impossível evitar tempos de espera, em que o CPU não é utilizado…**

Solução: múltiplos programas

- **Manter vários Programas, independentes, em Memória, prontos a executar**

Solução: múltiplos programas

- **Manter vários programas, independentes, em memória, prontos a executar**
- **Quando um programa pede I/O e tem de esperar → o SO escolhe outro programa para execução**



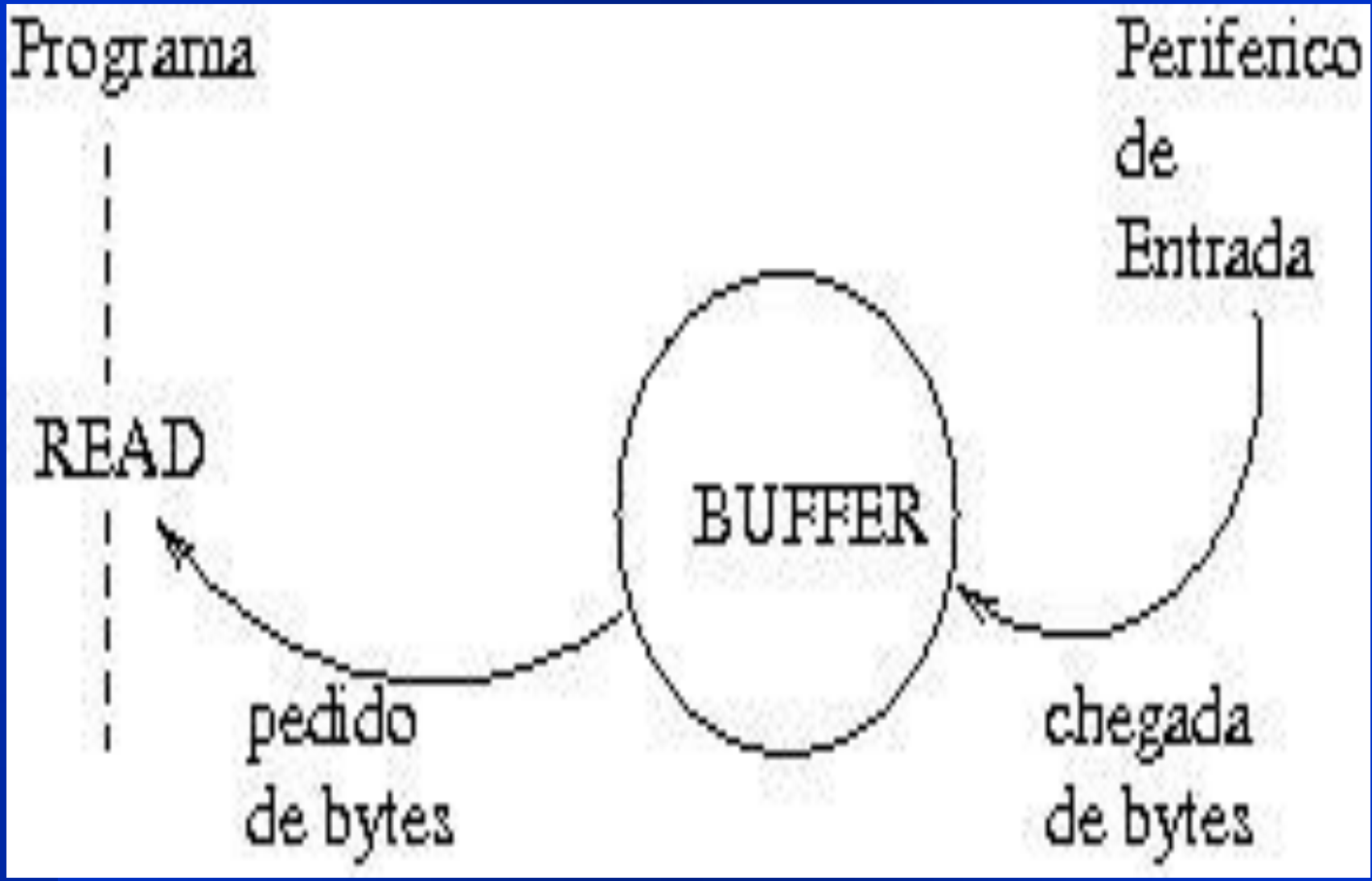
Solução: múltiplos programas

- **Manter vários programas, independentes, em memória, prontos a executar**
- **Quando um programa pede I/O e tem de esperar → o SO escolhe outro programa para execução**

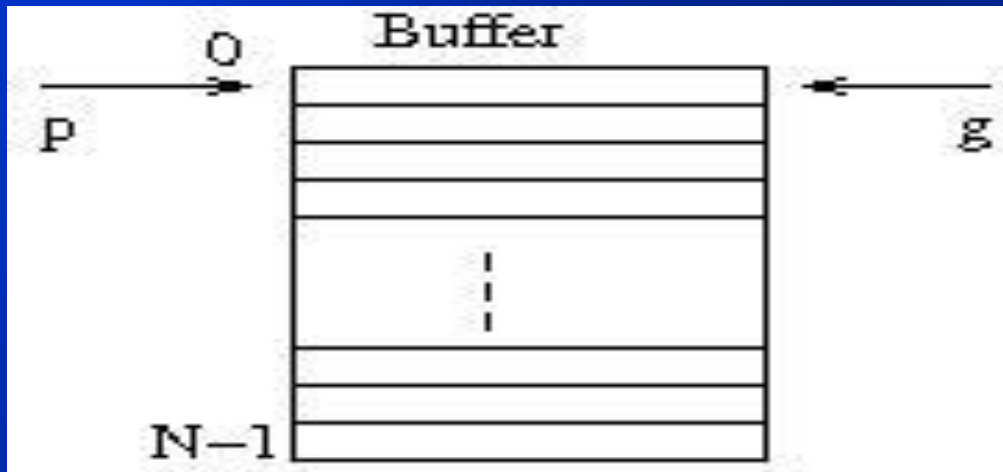
Vários programas em Concorrência (Multiprogramação)

(Não são executados em simultâneo: se só há 1 CPU!)

Leitura de dados através de *buffers*



Uso de *buffers* circulares




```
put: M[p] := x;
     p := (p+1) mod N;
     cont := cont + 1;

get: x := M[g];
     g := (g+1) mod N;
     cont := cont – 1;
```

```
p, g: (0 .. N–1)

inicio:
     p = g = 0
     cont = 0

antes de put:
  if cont = N then
        —— buffer chei
  else
     put(x);

antes de get:
  if cont = 0 then
        —— buffer vazio
  else
     x := get();
```

Leitura, com espera activa!!!

READ:

```
while (cont = 0) do;
```

ciclo de espera activa

```
x := get();
```

retorna ao programa

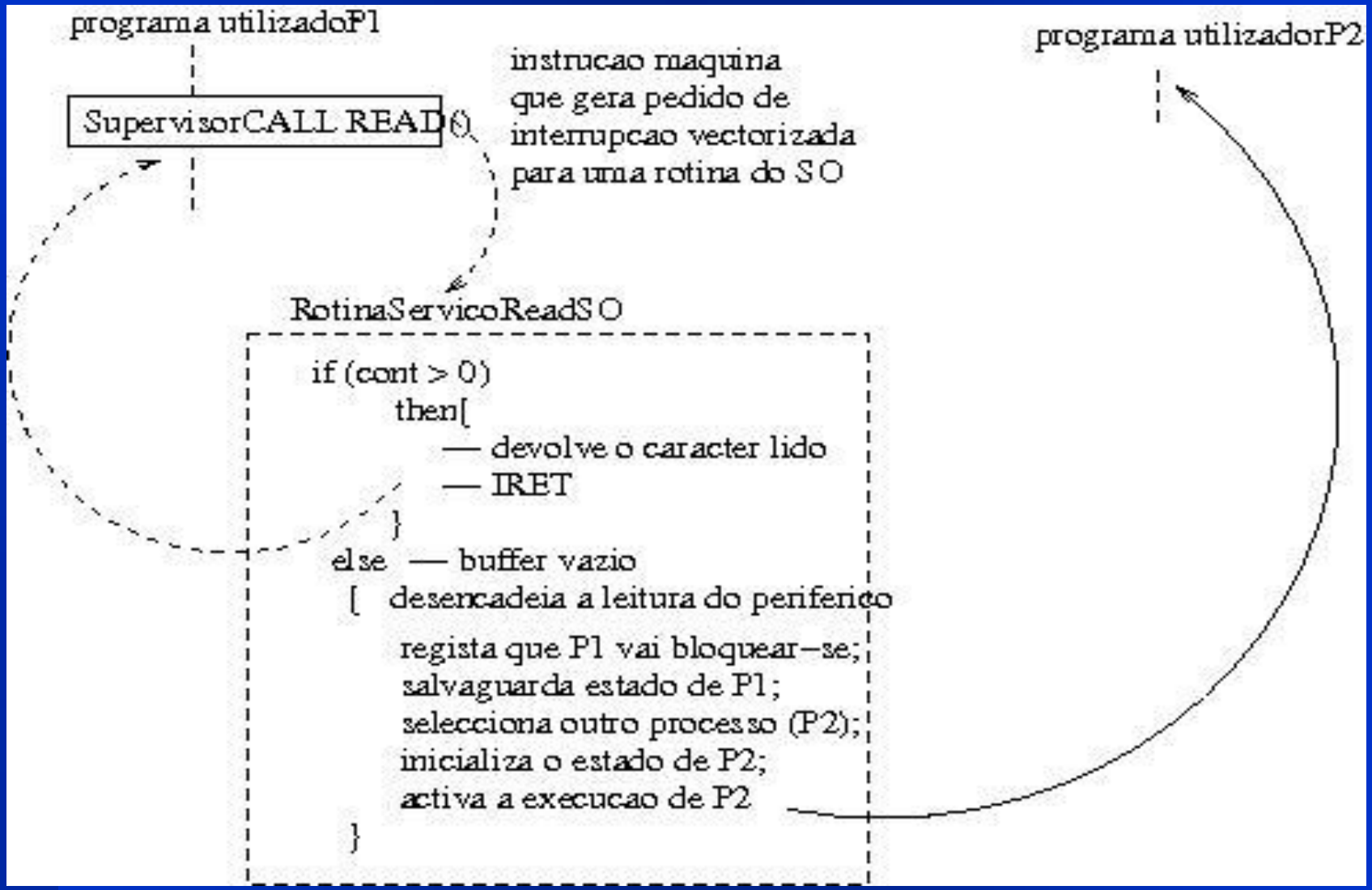
Do lado da Rotina de Serviço

ROTINA SERVICO INTERRUPCOES:

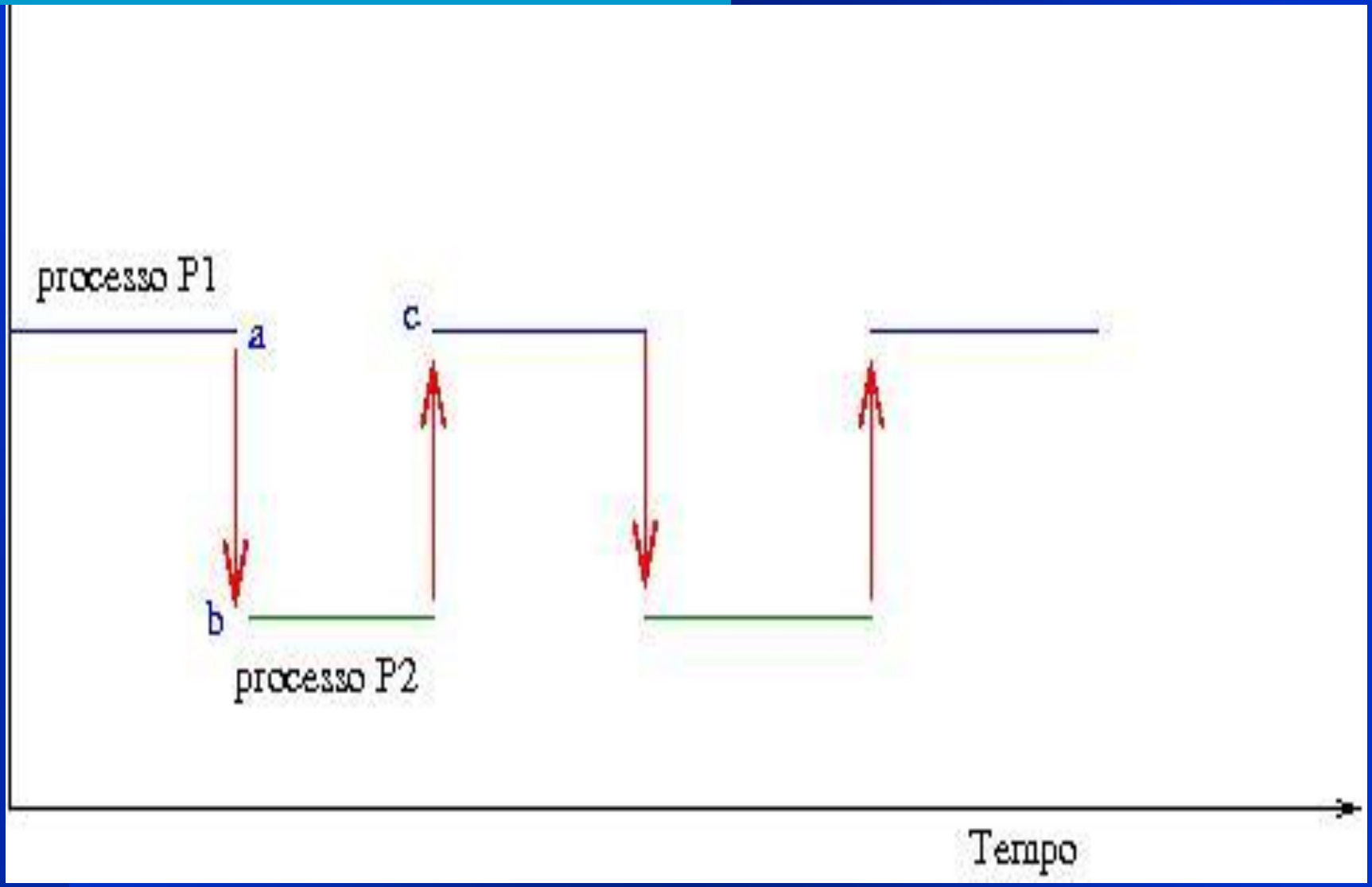
```
 RegCPU  := PortaDadosTecla;
if (cont = N)
     then
          -- buffer cheio
     else
          put(RegCPU);

 IRET;
```

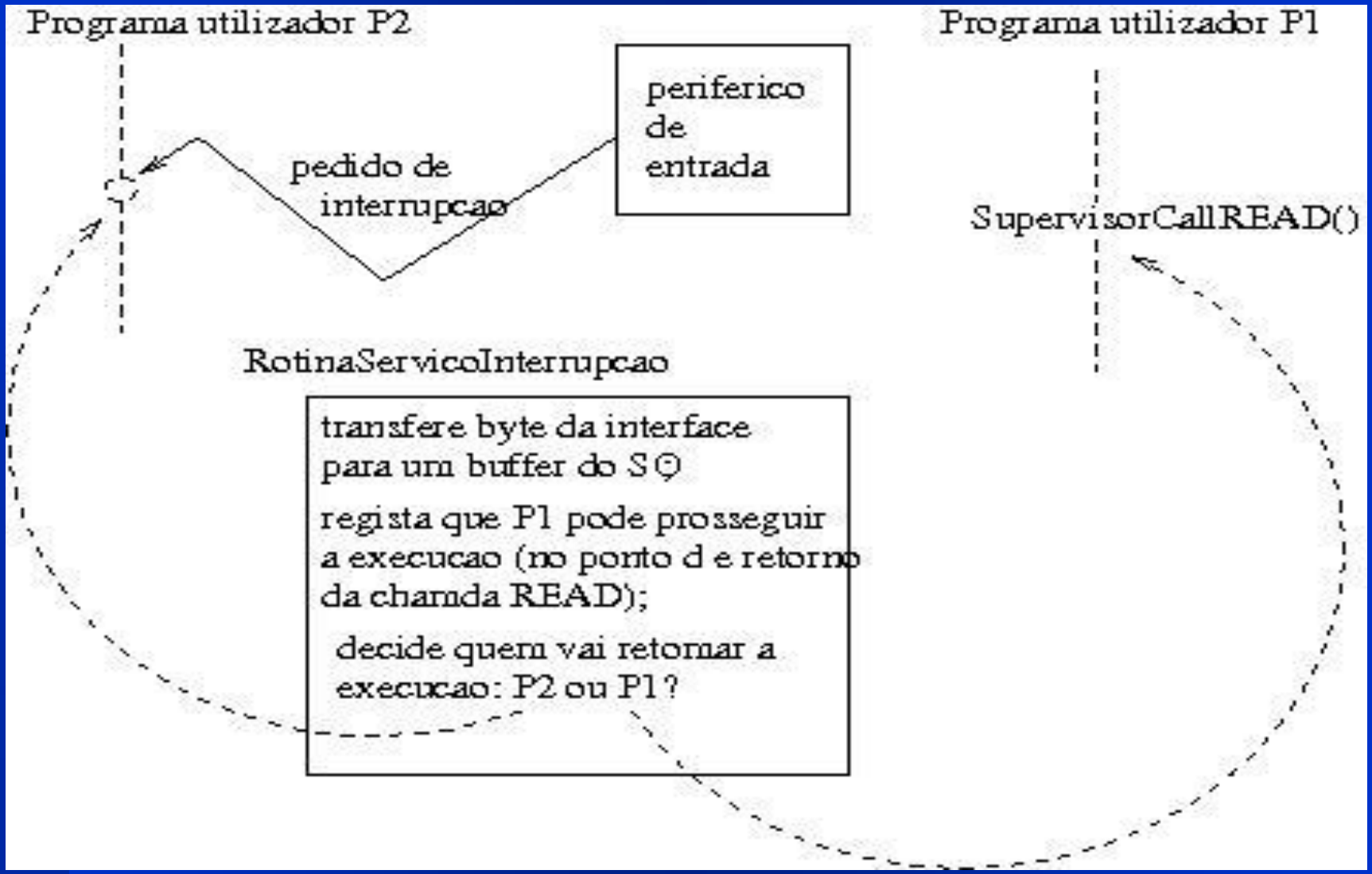
Leitura com bloqueio de P1



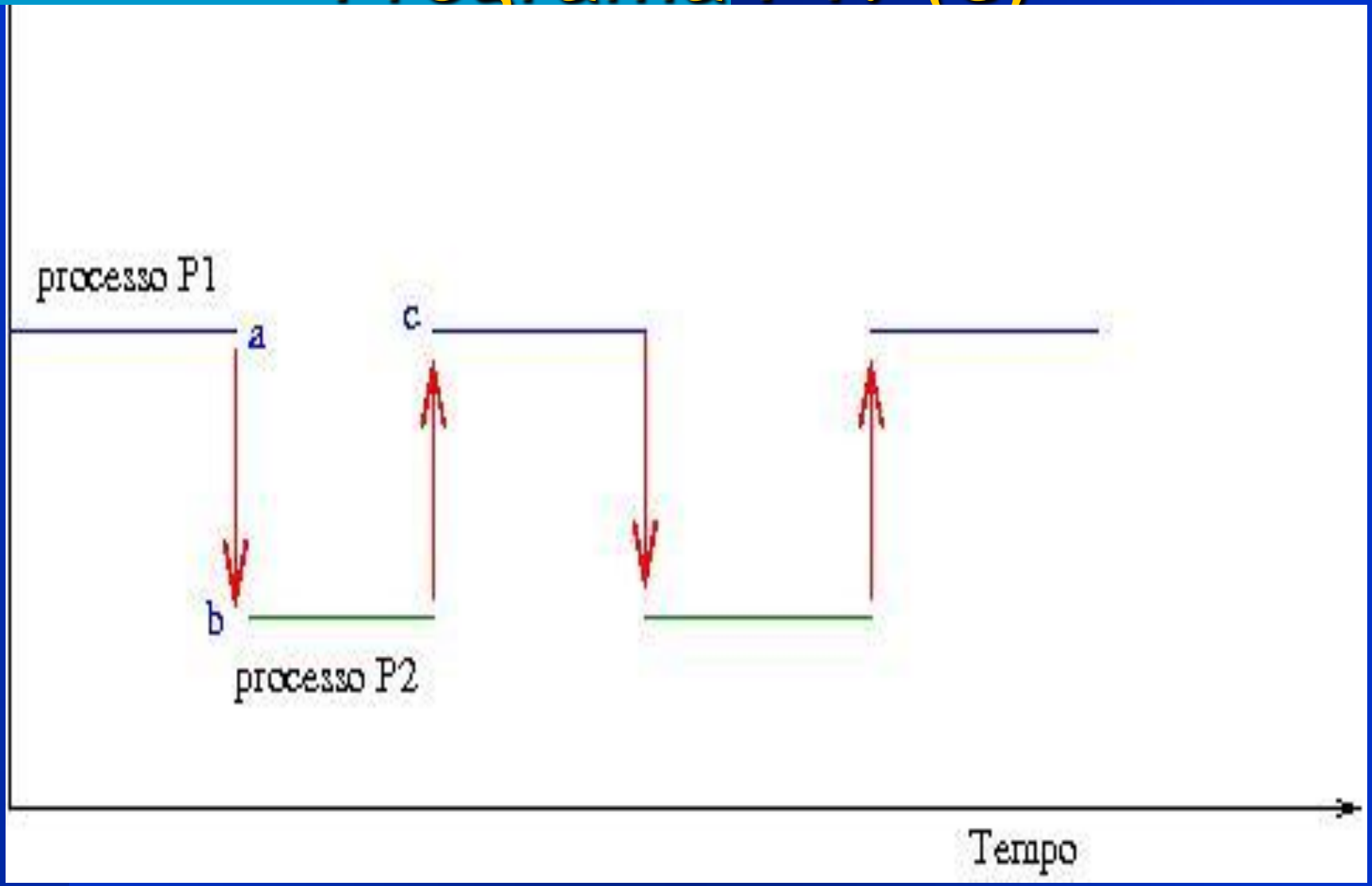
Mudança para Programa P2: (b)



Quando os dados de P1 chegam…



Mudança de regresso ao Programa P1: (c)



Multiprogramação

- **Cada programa executa num Contexto (Processo), definido por:**
  - **Imagem carregada em memória (Código, Dados e Pilha)**
  - **Estado do CPU (valores dos Registadores)**
  - **Estado das interfaces I/O + Ficheiros abertos**

Multiprogramação

- Cada programa executa num Contexto (Processo), definido por:
  - Imagem carregada em memória (Código, Dados e Pilha)
  - Estado do CPU (valores dos Registadores)
  - Estado das interfaces I/O + Ficheiros abertos
- O SO muda de contexto quando necessário
  - Suporta um ambiente de execução virtual para cada processo