Aula 5

Professores:

Lúcia M. A. Drummond

Simone de Lima Martins

Conteúdo:

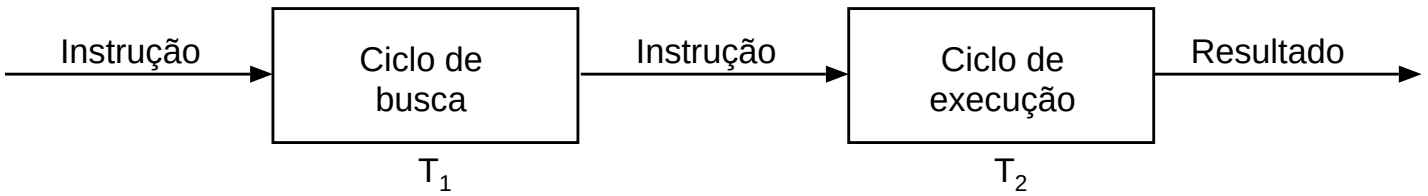
Unidade Central de Processamento 3

- Linha de montagem ou pipelining
- Barramentos
- Implementação de controle

Fundação CECIERJ
Consórcio cederj

Linha de montagem ou pipelining

- Etapas do ciclo de instrução são executadas de forma seqüencial
- Tempo total de execução da instrução é igual à soma dos tempos gastos em cada etapa



T = tempo de execução da instrução = $T_1 + T_2$

Ciclo de busca = leitura da instrução, incremento do CI

Ciclo de execução = decodificação, busca do operando, execução da operação

(Fig. 6.35 do livro texto)

Linha de montagem ou pipelining

- Analogia com fabricação de um automóvel
  - Montagem da carroceria (T1)
  - Montagem do motor (T2)
  - Montagem das rodas (T3)
  - Instalação elétrica (T4)
  - Instalação de bancos e espelhos (T5)
  - Acabamento final (T6)
  - Tempo total T = T1 + T2 + T3 + T4 + T5 + T6
  - Em um turno de X horas, um máximo de X/T carros podem ser montados

Linha de montagem ou pipelining

- Metodologia inventada por Henry Ford no início do século XX
  - Dividir processo de fabricação em estágios independentes
  - Denominada linha de montagem (pipeline) pois um item pode iniciar sua fabricação antes que um item anterior termine sua montagem
  - Objetivo é possibilitar a fabricação de mais unidades por unidade de tempo
  - Não diminui o tempo de fabricação de uma unidade
  - Esta técnica é utilizada na arquitetura de unidades centrais de processadores

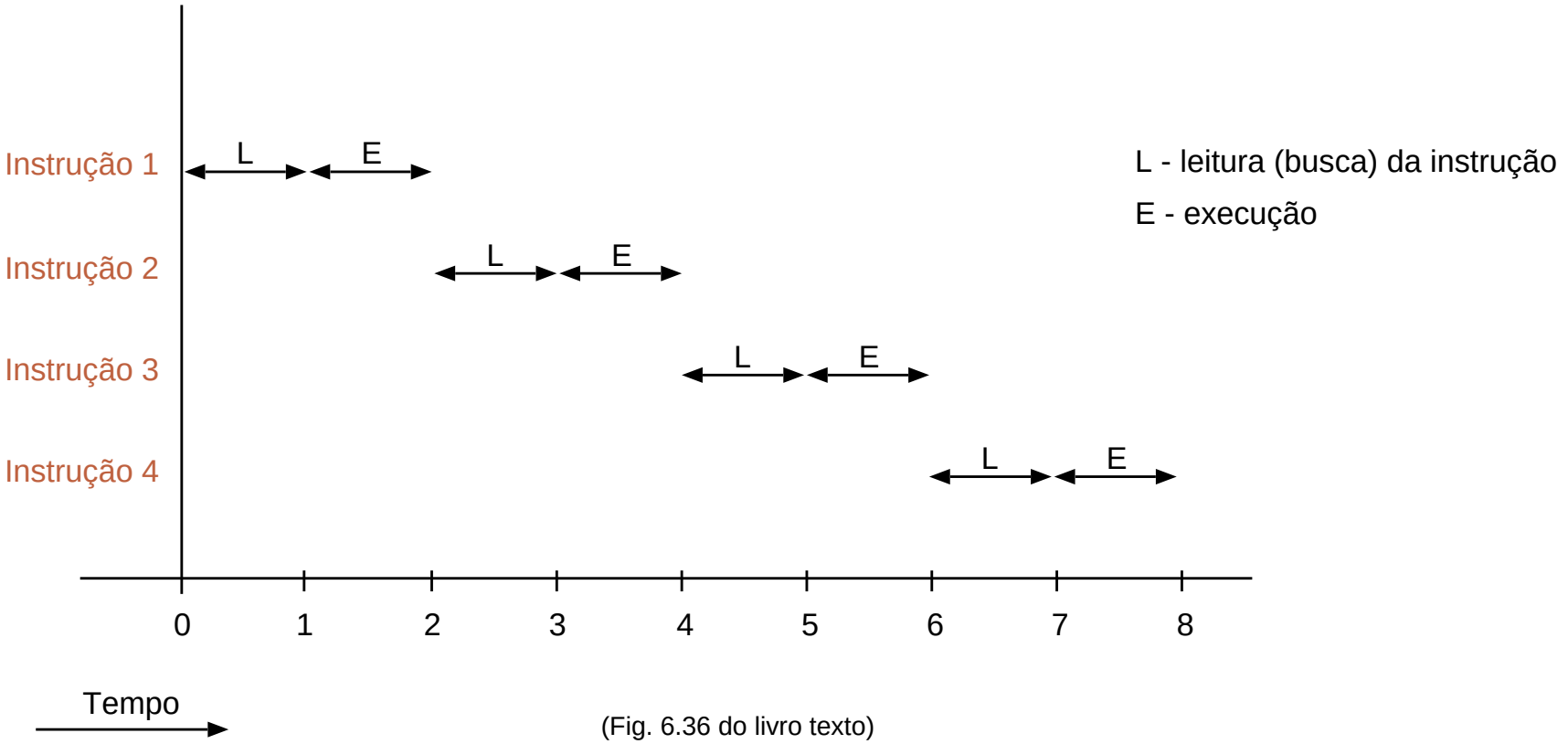
Linha de montagem ou pipelining

- Baseada em duas premissas básicas
  - O processo (fabricação de um automóvel ou execução de uma instrução) tem que ser dividido em estágios de realização independentes um do outro
  - Um novo produto (carro ou instrução) inicia seu processo de fabricação ou execução antes do anterior concluir seu processo

Linha de montagem ou pipelining

- Implementação de pipelining com divisão do ciclo de instrução em dois estágios
  - Busca da instrução
    - Acesso à memória
  - Execução da instrução
    - Nem sempre é necessário acessar a memória (decodificação ou execução da instrução)
  - Quando não há atividade de acesso à memória no estágio de execução, pode-se ativar o estágio de busca da instrução

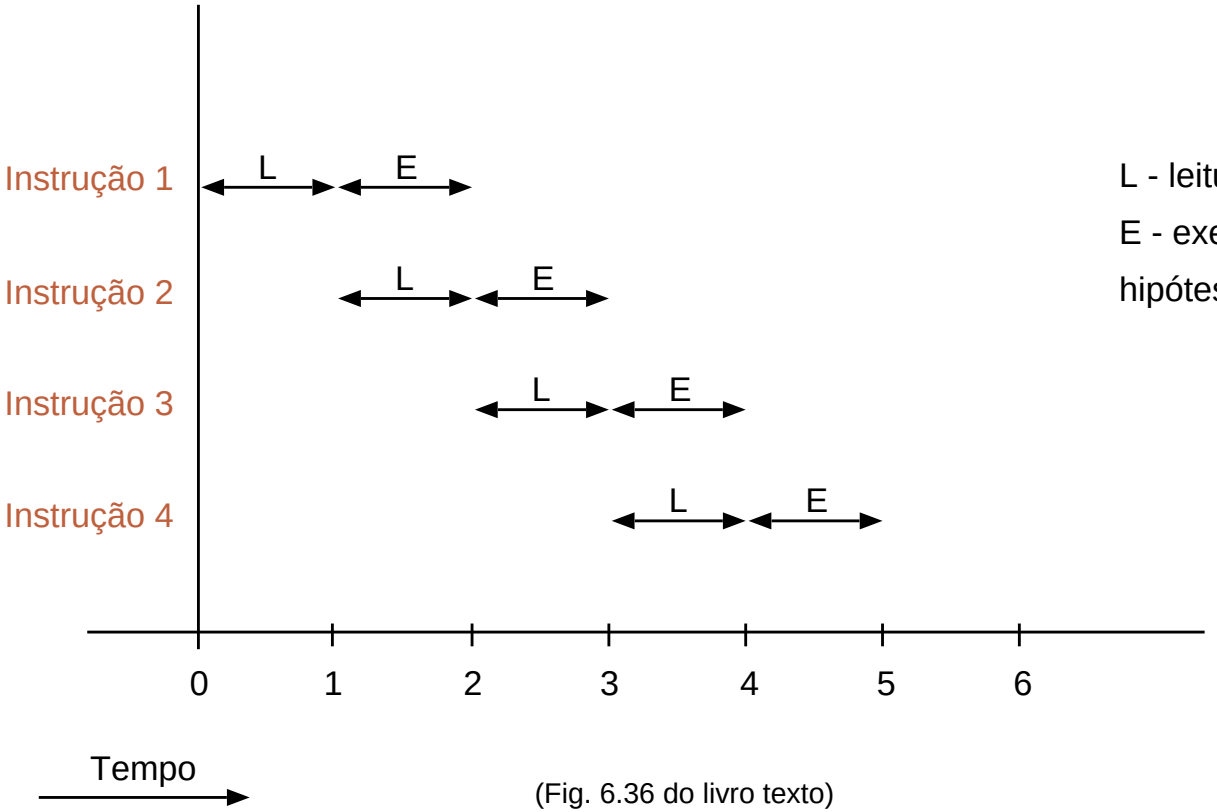
Linha de montagem ou pipelining



(Fig. 6.36 do livro texto)

- Supondo que cada estágio gaste o mesmo período de tempo T
  - Para executar as 4 instruções de forma seqüencial seriam gastos 8T

Linha de montagem ou pipelining



L - leitura (busca) da instrução

E - execução

hipótese: tempo de L = tempo de E

(Fig. 6.36 do livro texto)

- Supondo que cada estágio gaste o mesmo período de tempo T
  - Para executar as 4 instruções de forma seqüencial seriam gastos 8T
  - Na estrutura em pipelining se gasta 5T
  - Mas cada instrução gasta 2T

Linha de montagem ou pipelining

- Razões para não haver aumento significativo de produtividade em pipelining com 2 estágios
  - Tempo de realização do estágio de execução em geral é maior do que o tempo de busca (acesso à memória e execução de operações). Então o estágio de busca pode ter de esperar algum tempo para poder enviar a instrução para o estágio de execução
  - A ocorrência de instruções de desvio condicional faz com que o endereço da próxima instrução a ser buscada seja desconhecido. Então estágio de busca tem que esperar a finalização do estágio de execução

Linha de montagem ou pipelining

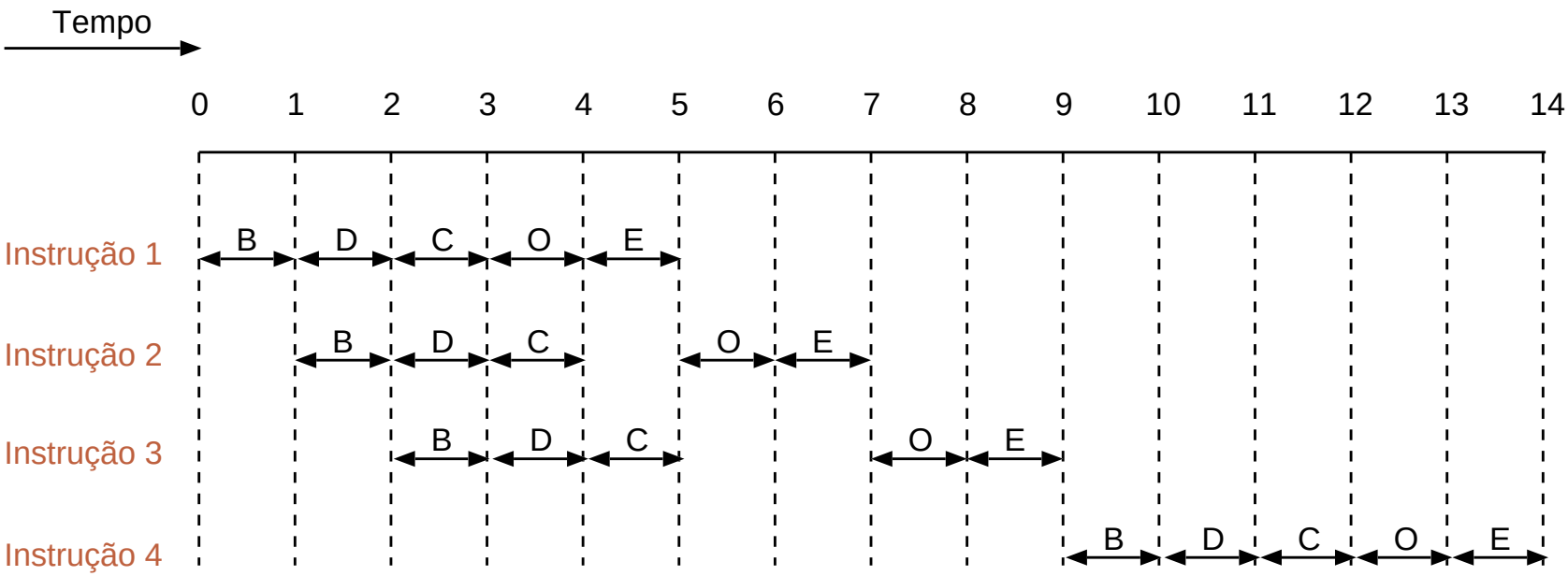
- Para se obter maior produtividade o pipelining deve ser dividido em mais estágios, pois teremos uma maior probabilidade de obter tempos iguais em cada estágio

- Divisão de cada instrução em 5 estágios com duração igual
  - Busca da instrução (B)
  - Decodificação da instrução (D)
  - Cálculo do endereço de operandos (C)
  - Busca dos operandos (O)
  - Execução da operação (E)

Linha de montagem ou pipelining

- Considerações sobre a arquitetura que implementa o pipelining
  - Somente um acesso à memória pode ser realizado de cada vez, ou seja, durante a execução do estágio de busca de instrução (B), o estágio de busca de operandos (O) não pode ser realizado
  - Sempre existe acesso à memória no estágio de execução
  - Todos os estágios são realizados em um período de tempo igual

Linha de montagem ou pipelining



(Fig. 6.37 do livro texto)

- Tempo de execução seqüencial: 20 unidades de tempo
- Tempo de execução com pipelining: 14 unidades de tempo

Linha de montagem ou pipelining

- Instrução de desvio condicional

  Exemplo:

  - JP OP
  - Se ACC > 0, então CI ← OP

- Não há meios de se conhecer o endereço da próxima instrução antes do teste da condição definida na instrução
  - Se condição verdadeira, endereço definido na instrução
  - Se condição falsa, endereço é o seguinte

Linha de montagem ou pipelining

- Soluções para diminuir o atraso no fluxo de processamento de instruções, devido à instrução de desvio condicional
  - Estágio de busca obtém na memória a instrução armazenada após a instrução de desvio. Caso não haja desvio, nenhum tempo foi perdido. Caso contrário, descarta-se a instrução buscada e busca-se a nova instrução
  - Busca da instrução armazenada após a instrução de desvio e da instrução armazenada no endereço do desvio

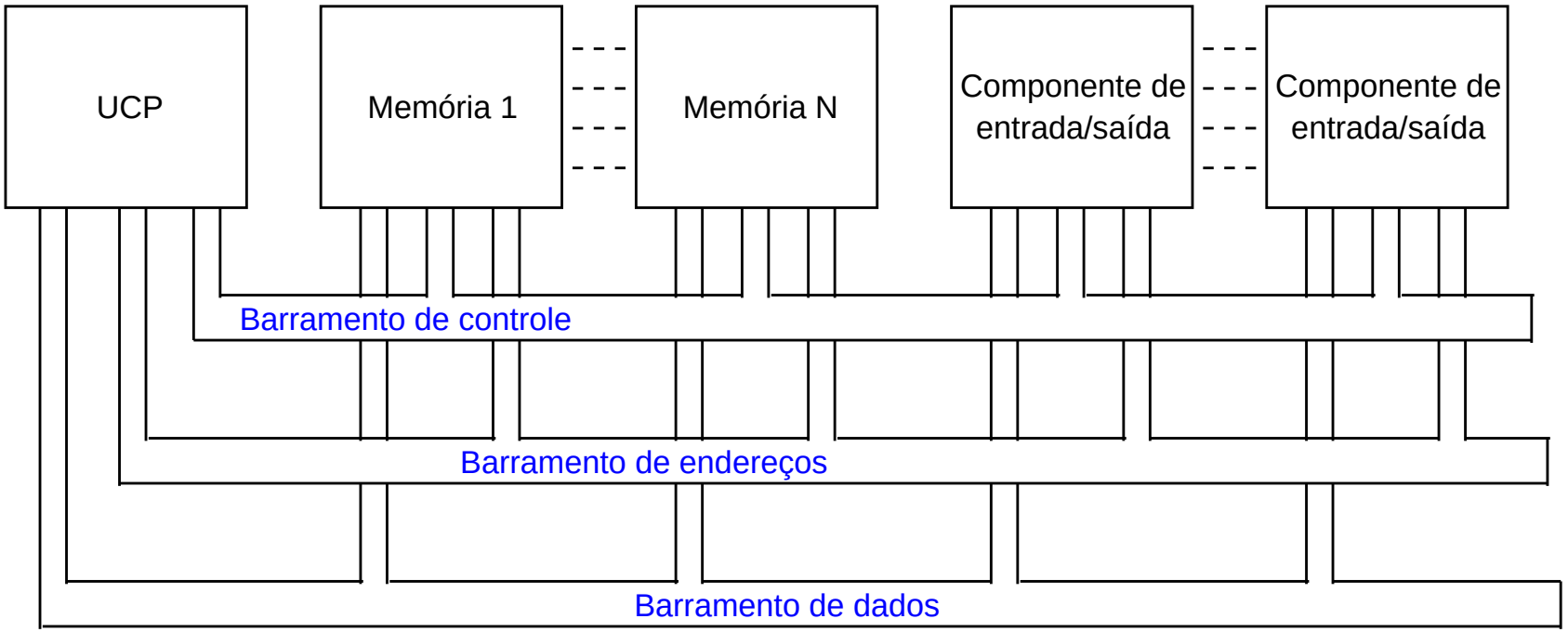
Linha de montagem ou pipelining

- Soluções para diminuir o atraso no fluxo de processamento de instruções, devido à instrução de desvio condicional
  - Sistema tenta prever se desvio vai ocorrer ou não
    - Prever que o desvio nunca será tomado
    - Prever que o desvio sempre será tomado
    - Prever se o desvio será tomado baseado no código de operação
    - Prever o desvio com base em uma tabela de histórico de desvios para cada instrução de desvio

Barramento

- O barramento de um sistema de computação é o elemento responsável pela interligação dos demais componentes, conduzindo de modo sincronizado o fluxo de informações (dados, endereços e sinais de controle) entre eles de acordo com uma programação de atividades previamente definida pela unidade de controle

Barramento



(Fig. 6.38 do livro texto)

Barramento

- Barramento de dados
  - Múltiplas linhas condutoras sendo que cada uma permite a passagem de um bit de informação
  - Diferentes quantidades de bits: 8, 16, 32, 64 e 128
- Barramento de endereços
  - Utilizado pelo processador para indicar de onde um dado deve ser lido e onde um dado deve ser escrito
  - Quantidade de bits especifica a quantidade máxima de memória principal
  - O valor binário colocado neste barramento pode indicar um endereço de dispositivo de entrada e saída

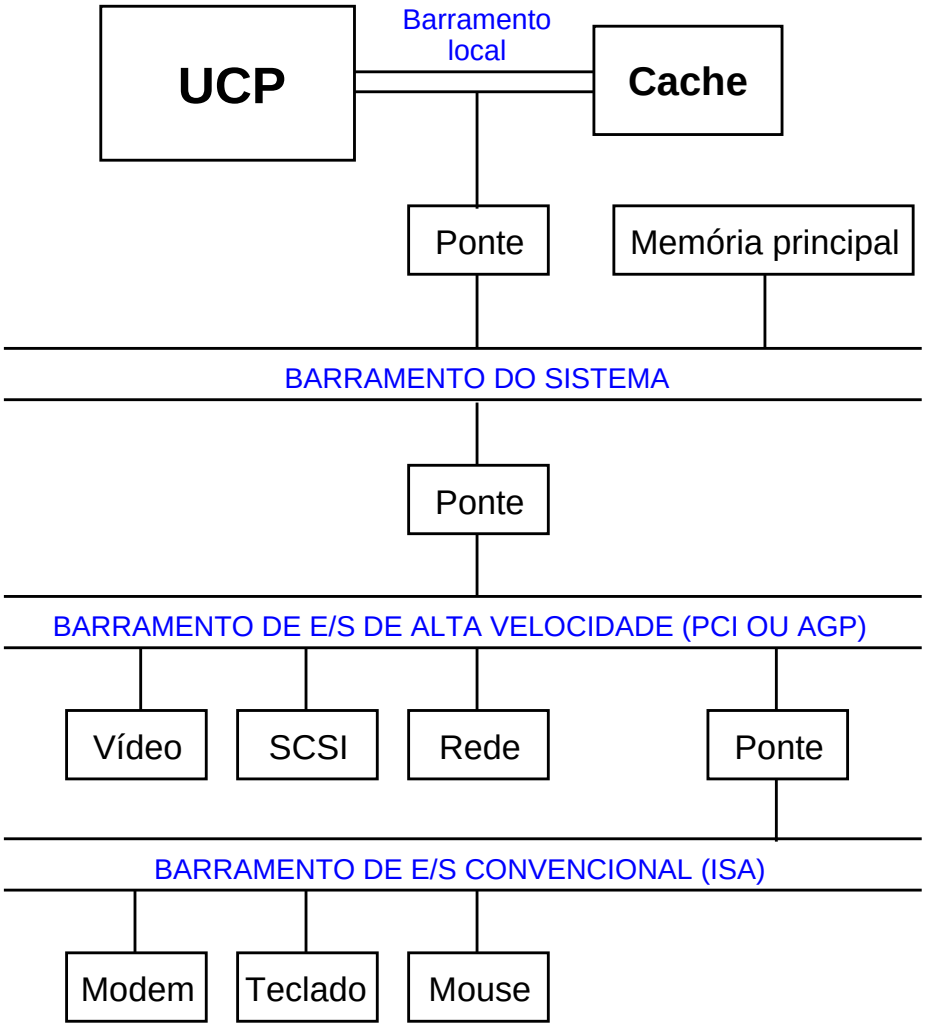
Barramento

- Barramento de controle
  - Constituído de linhas por onde fluem sinais específicos da programação do sistema
    - Leitura de dados (memory read)
    - Escrita de dados (memory write)
    - Leitura de E/S (I/O read)
    - Escrita de E/S (I/O write)
    - Certificação de transferência de dados (ACK)
    - Pedido de interrupção (interrupt request)
    - Relógio (clock)

Barramento

- Arquiteturas modernas não utilizam barramentos compartilhados por todos os componentes
  - Características diferentes entre os diversos componentes levaram à criação de diversos tipos de barramento
  - Cada barramento possui taxas de transferência de bits diferentes e apropriadas às velocidades dos componentes conectados, sendo organizados de forma hierárquica

Barramento



(Fig. 6.40(b) do livro texto)

Barramento

- Largura do barramento
  - Caracteriza a quantidade de informação (número de bits, em geral) que pode fluir simultaneamente no barramento
- Ciclo de tempo do barramento
  - Tempo requerido para mover um grupo de bits (cujo tamanho é definido pela largura do barramento) ao longo do barramento

Barramento

- Barramentos são compartilhados pelos componentes
- Compartilhamento implica na necessidade de definição de regras explícitas de acesso ao barramento
  - Quando acessar
  - Como acessar
- Compartilhamento implica na necessidade de definição de regras explícitas de comunicação
  - Como interrogar um componente destinatário
  - Qual resposta deve ser enviada
  - Quanto dura a comunicação

Barramento

- Protocolos de barramento foram criados para definir estas regras de acesso ao barramento
  - ISA - Industry Standard Adapter
    - Adotado para periféricos de baixa velocidade
  - PCI - Peripheral Component Interconnect
    - Permite várias larguras de barramento e velocidades de transmissão
    - Quase um padrão no mercado
  - USB - Universal Serial Bus
    - Pode-se conectar muitos dispositivos simultaneamente
  - AGP - Accelerated Graphics Port

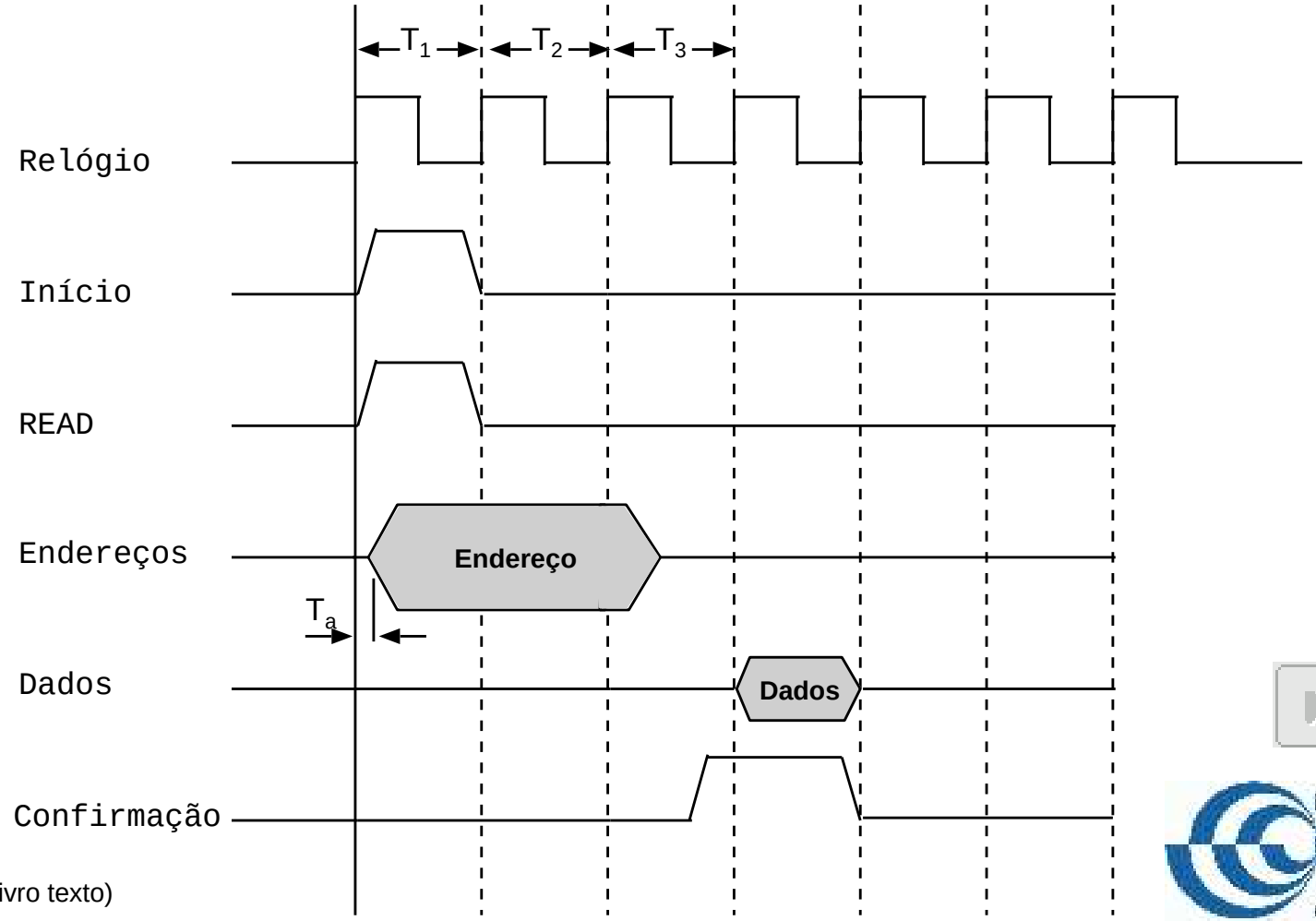
Barramento

- Projeto do barramento
  - Método de controle do acesso ao barramento
    - Mestre/escravo
      - Um mestre é o único que pode acessar o barramento
      - Toda comunicação é realizada via mestre, que pode causar um gargalo no sistema
    - Acesso direto
      - Todos os componentes podem acessar o barramento
      - Requer maior complexidade nos circuitos de controle do barramento

Barramento

- Projeto do barramento
  - Tipo de sincronização

    Operação síncrona

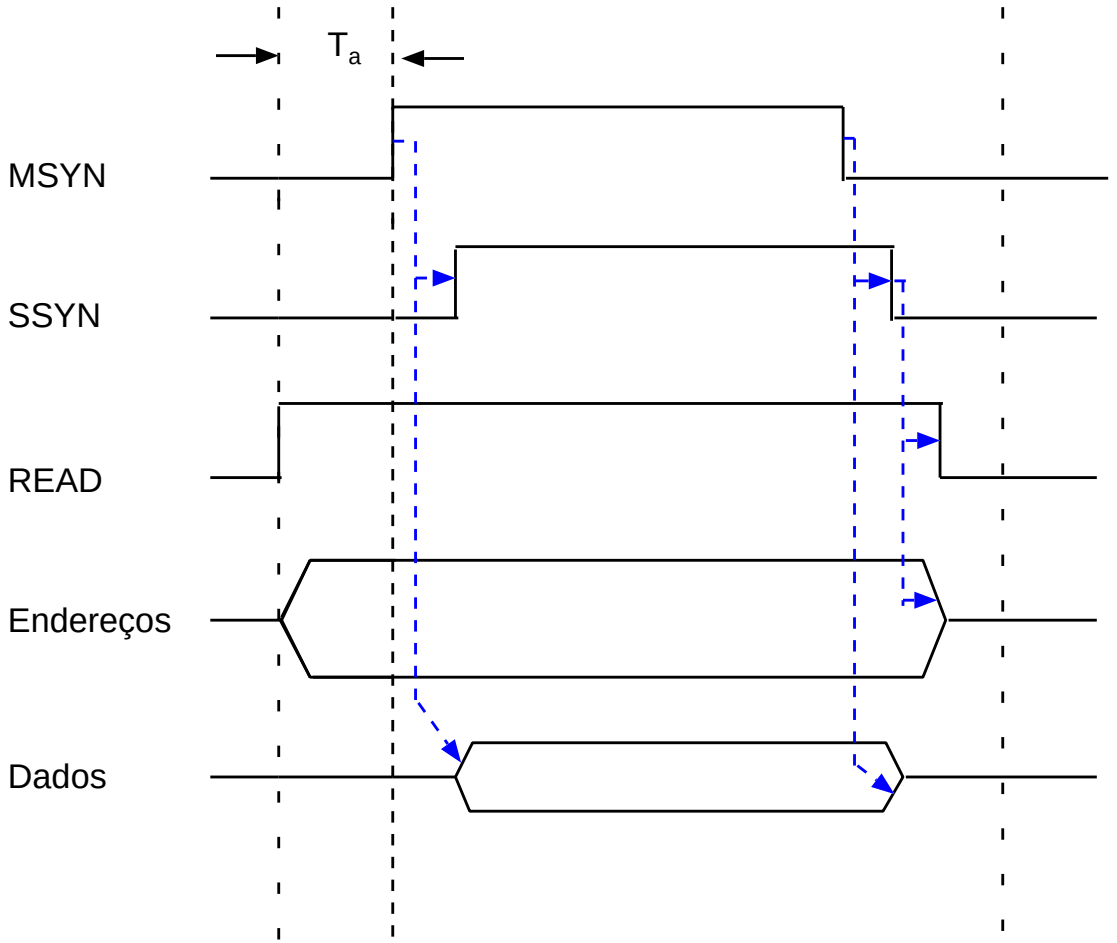


(Fig. 6.41 do livro texto)

Barramento

- Projeto do barramento
  - Tipo de sincronização

    Operação assíncrona

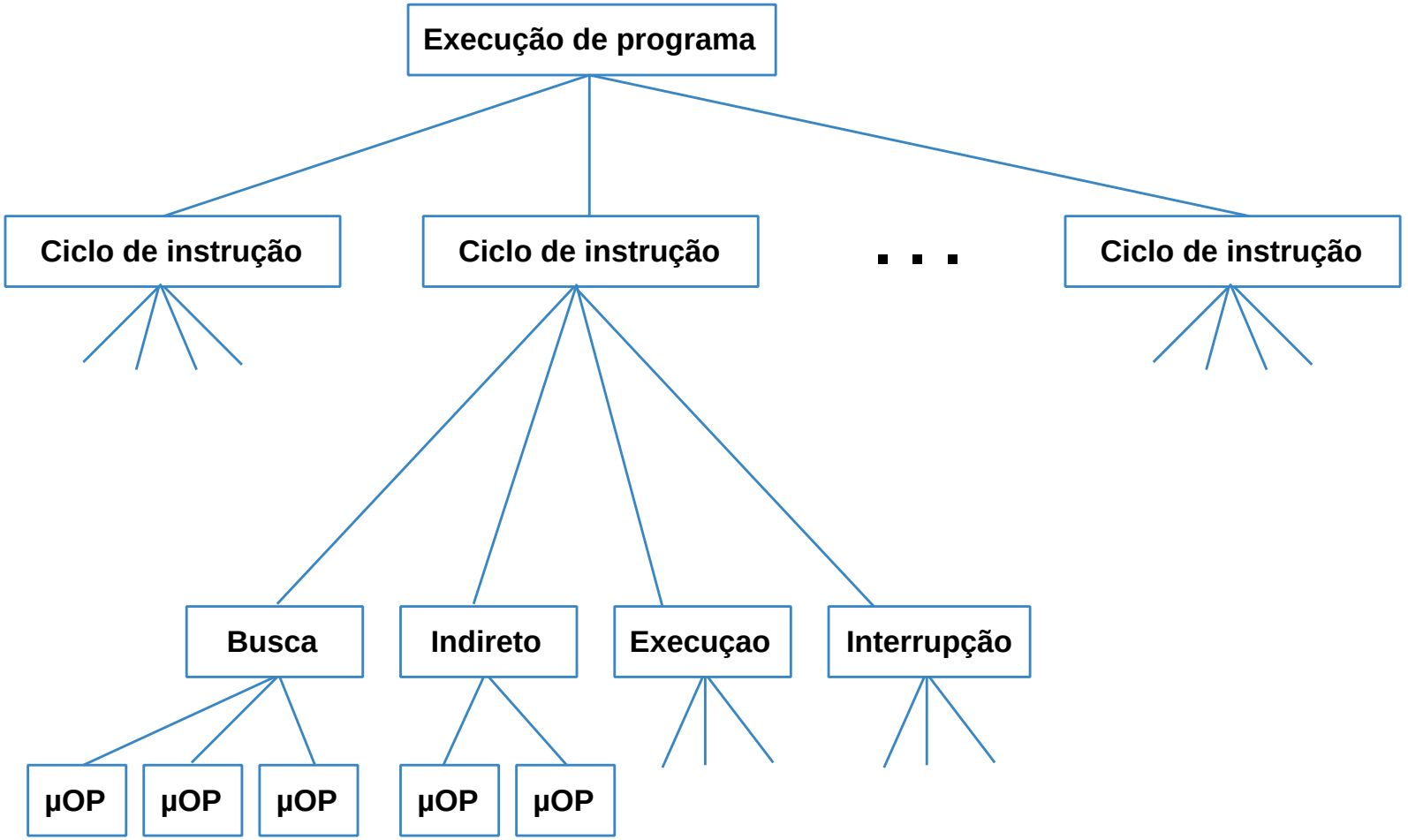


(Fig. 6.42 do livro texto)

Barramento

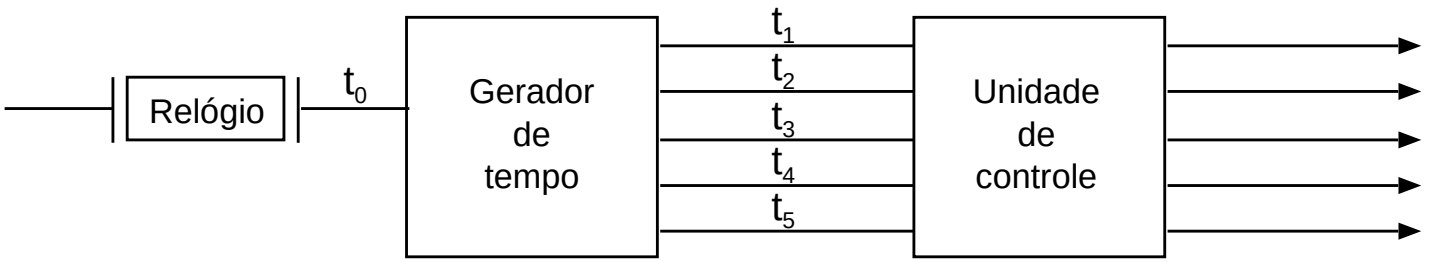
- Projeto do barramento
  - Sincronização síncrona X assíncrona
    - Barramento síncrono é fácil de implementar e testar
    - Barramento síncrono pode ter problemas ao trabalhar com dispositivos com tempos de transferência diferentes
    - Barramento assíncrono pode interligar componentes de diversas velocidades e tecnologias

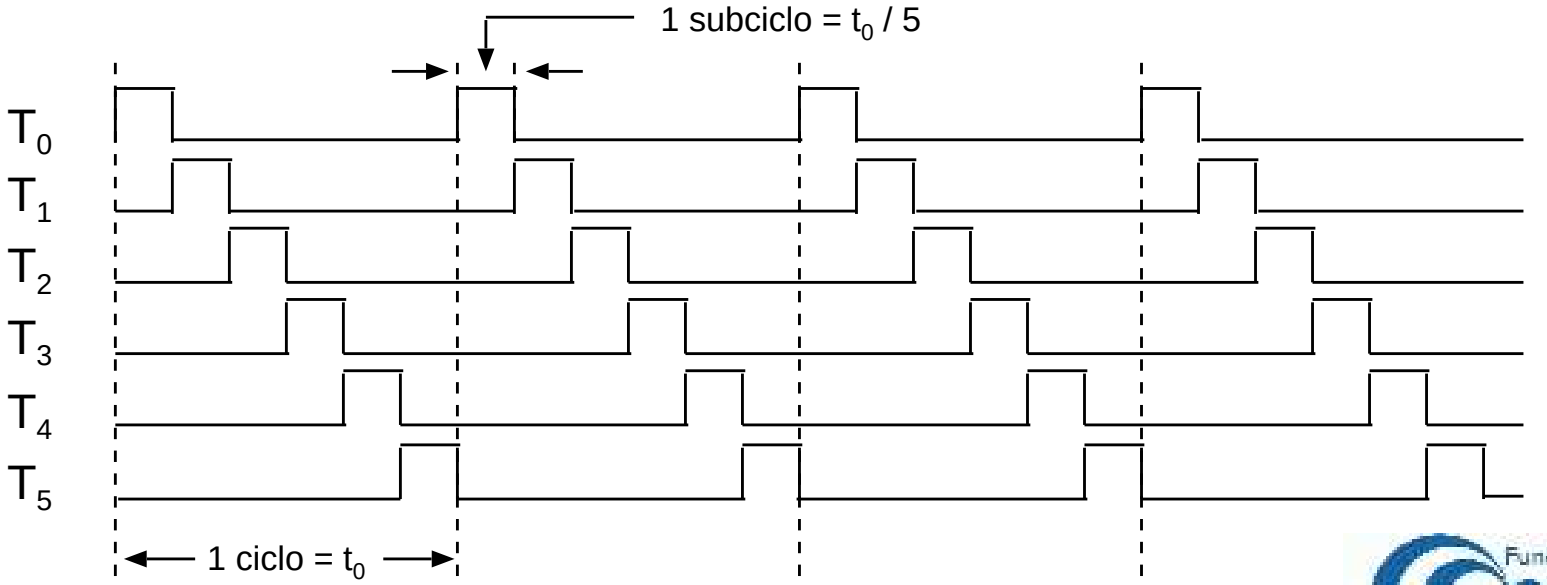
Implementação de controle



Implementação de controle

- O conjunto de µoperações de cada subciclo do ciclo de instrução (busca, indireto, execução, interrupção) deve ser executado em um ciclo de relógio





(Fig. 6.15 do livro texto)

Implementação de controle

- Ciclo de busca

t1: REM ← (CI)
t2: RDM ← Memória
CI ← CI+1
t3: RI ← (RDM)

Implementação de controle

- Ciclo de execução
  - ADD X (Soma do conteúdo do endereço X com o conteúdo do acumulador)

t1: REM ← (RI(Endereço))
t2: RDM ← Memória
t3: ACC ← (ACC) + (RDM)

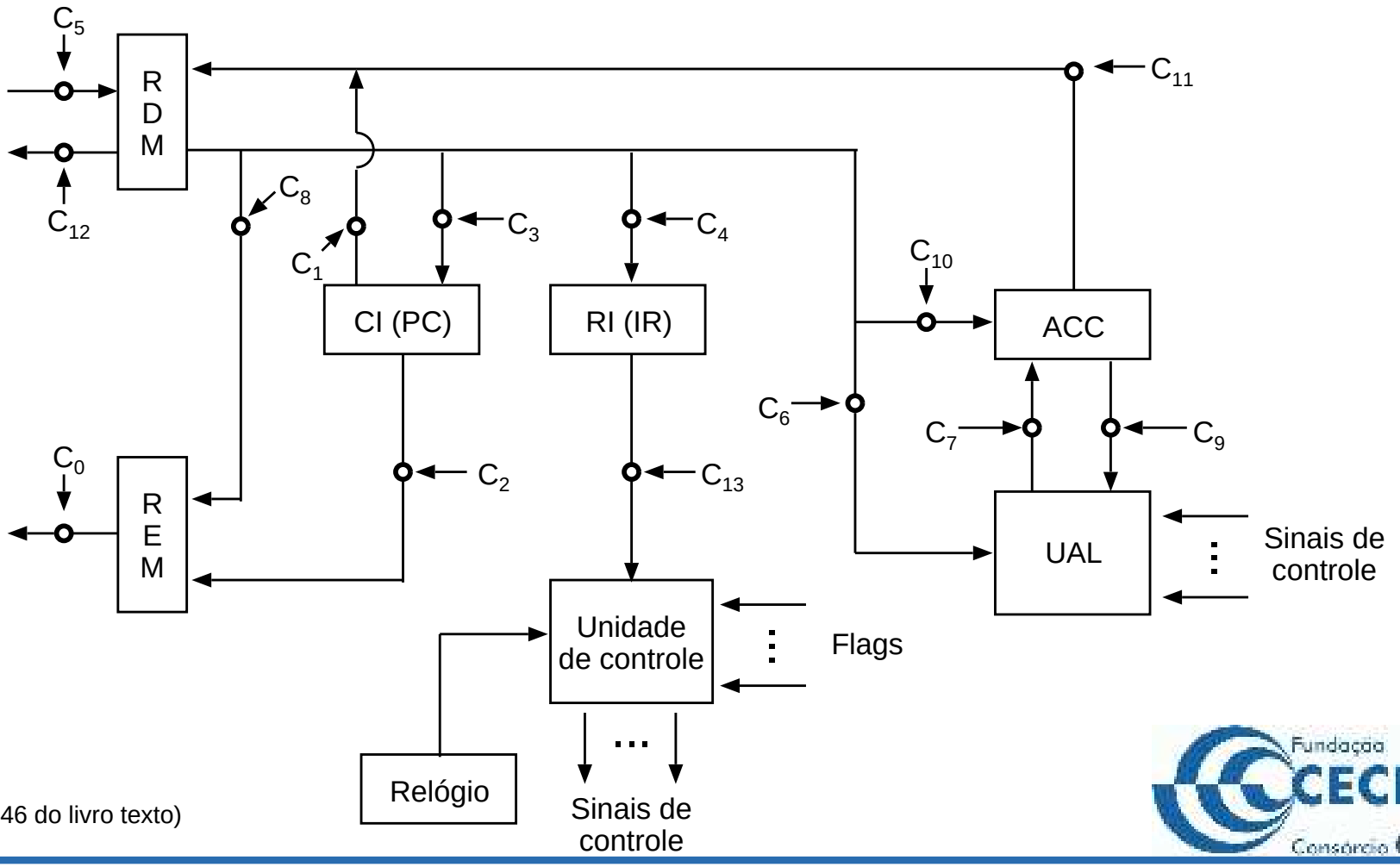
Implementação de controle

- Unidade de controle deve ser construída contendo a programação da geração dos sinais de controle internos e externos à UCP conforme a instrução que será decodificada

- Duas maneiras utilizadas no projeto de uma UC
  - Controle programado diretamente no hardware (hardwired control)
  - Controle por microprogramação

Implementação de controle

- Controle programado diretamente no hardware (hardwired control)
  - Unidade de controle é construída como um conjunto de circuitos logicamente combinados
  - Sinais de controle

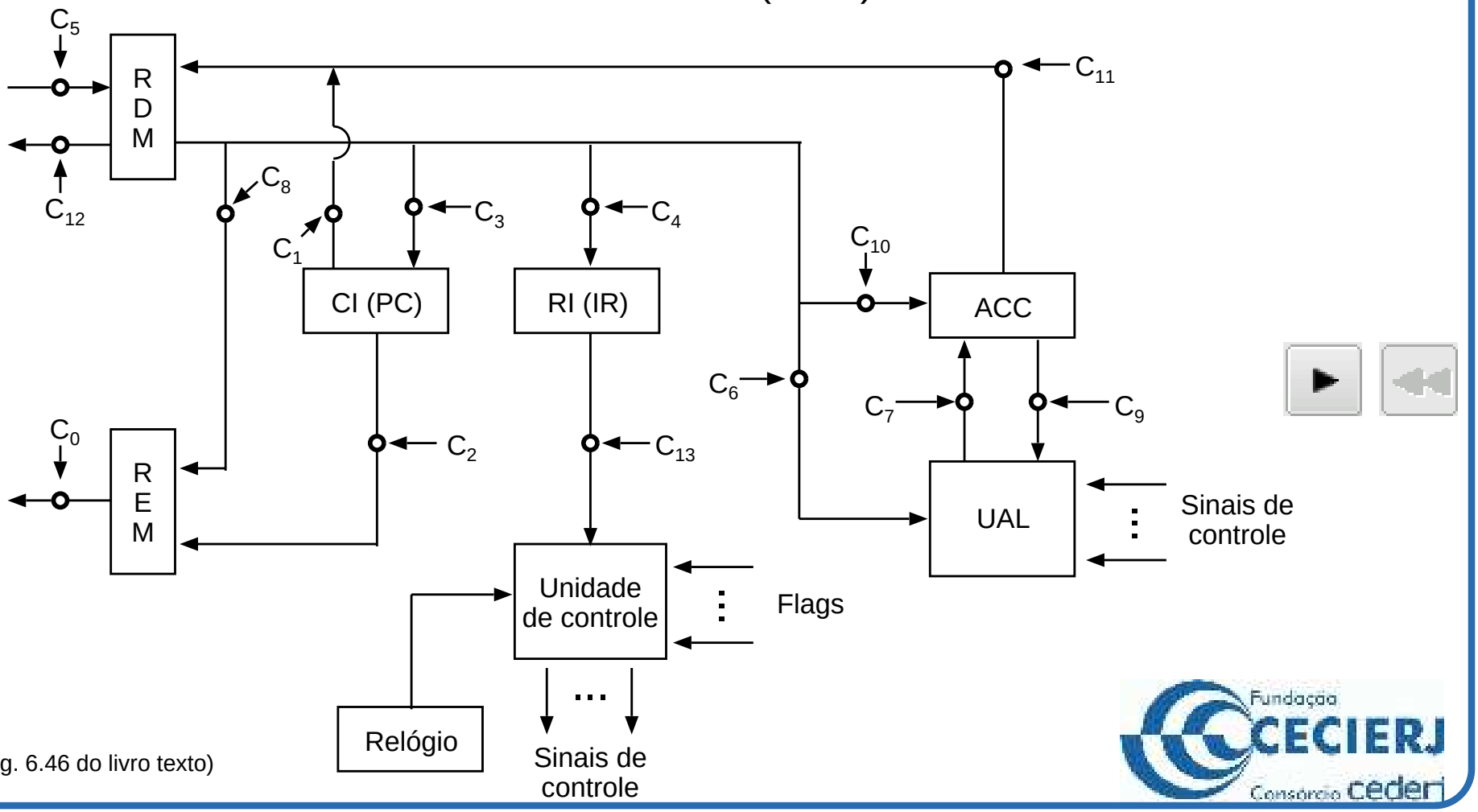


(Fig. 6.46 do livro texto)

Implementação de controle

- Sinais de controle
  - Ciclo de busca

t1: REM ← (CI)
t2: RDM ← Memória
CI ← CI+1
t3: RI ← (RDM)

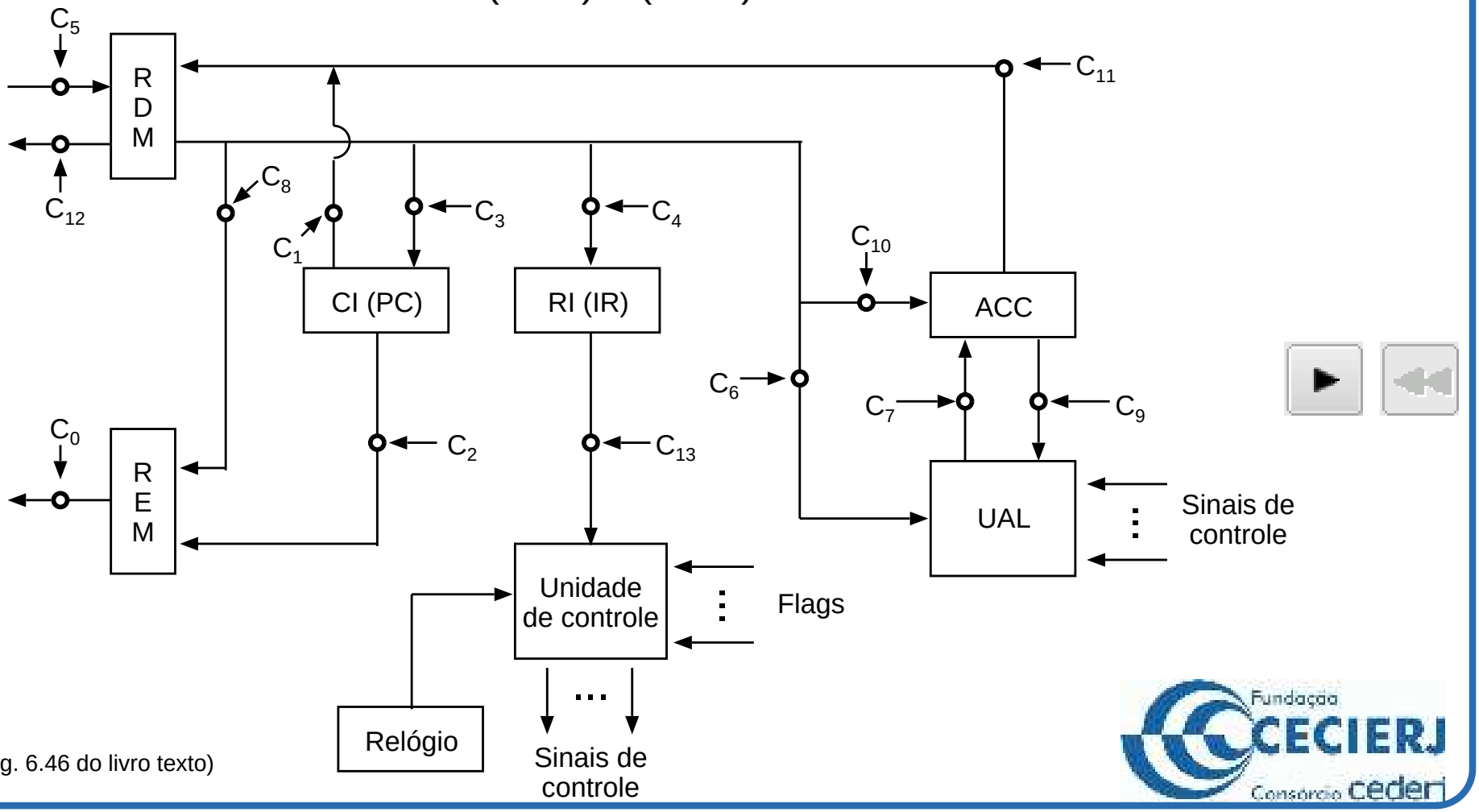


(Fig. 6.46 do livro texto)

Implementação de controle

- Sinais de controle
  - Ciclo de execução (instrução ADD X)

t1: REM ← (RI(Endereço))
t2: RDM ← Memória
t3: ACC ← (ACC) + (RDM)



(Fig. 6.46 do livro texto)

Implementação de controle

Subciclo	Temporização	Sinais de controle ativos
Busca:	t1: REM ← (CI) t2: RDM ← Memória CI ← CI+1 t3: RI ← (RDM)	C2 C0, C5 e READ C4
Execução:	t1: REM ← (RI(endereço)) t2: RDM ← Memória t3: ACC ← (ACC) + (RDM)	C4, C8 C0, C5 e READ C6, C7, C9

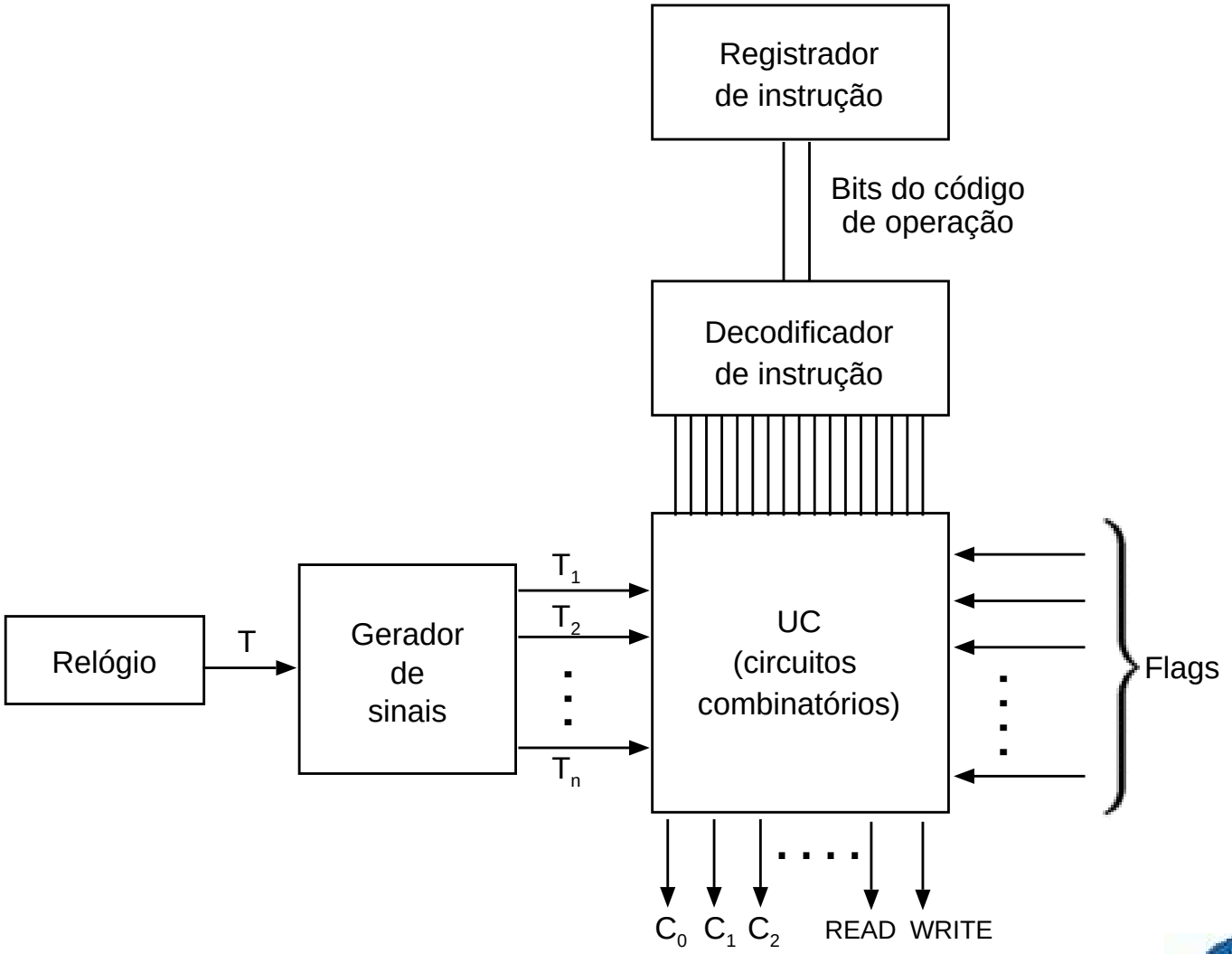
Sinais PQ = 00 Subciclo de busca
PQ = 01 Subciclo indireto
PQ = 10 Subciclo de execução
PQ = 11 Subciclo de interrupção

Sinal C5: ativo em Busca e Execução de ADD, em t2

$$C5 = \overline{P} \bullet \overline{Q} \bullet T2 + P \bullet \overline{Q} \bullet (LDA + ADD) \bullet T2$$

Implementação de controle

- Esquema de um sistema de controle programado diretamente no hardware



(Fig. 6.45 do livro texto)

Implementação de controle

- Controle programado diretamente no hardware
  - Número de equações booleanas requeridas pode ser muito grande
  - Implementar um circuito combinatório que satisfaça estas equações pode se tornar extremamente difícil
  - Abordagem utilizada para simplificar o projeto da unidade de controle: microprogramação

Implementação de controle

- Controle por microprogramação
  - Conceito desenvolvido por Wilkes em 1951

Subciclo	Temporização	Sinais de controle ativos
Busca:	t1: REM ← (CI) t2: RDM ← Memória CI ← CI+1 t3: RI ← (RDM)	C2 C0, C5 e READ C4
Execução:	t1: REM ← (RI(endereço)) t2: RDM ← Memória t3: ACC ← (ACC) + (RDM)	C4, C8 C0, C5 e READ C6, C7, C9

  - Cada microoperação é descrita em notação simbólica
  - Esta notação é conhecida como linguagem de microprogramação

Implementação de controle

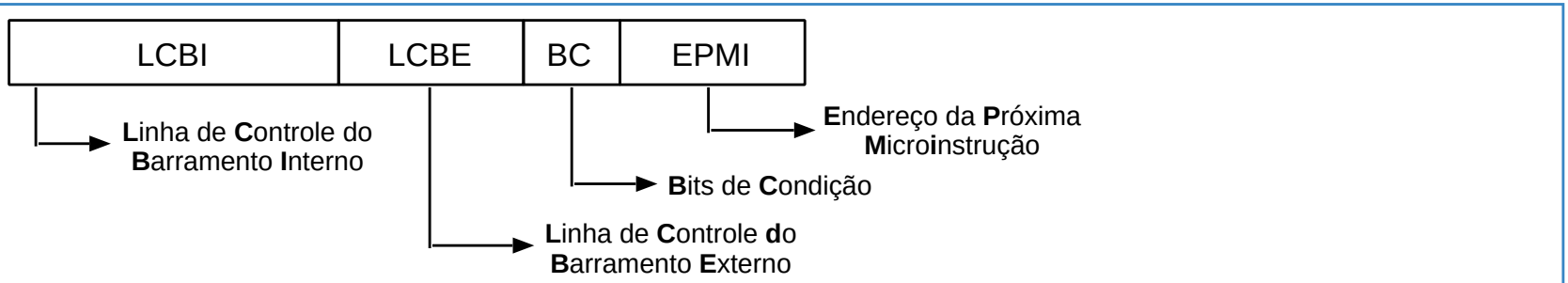
- Controle por microprogramação
  - Cada linha descreve um conjunto de microoperações que ocorrem ao mesmo tempo e é conhecida como microinstrução
  - Uma seqüência de microinstruções é um microprograma ou firmware

Implementação de controle

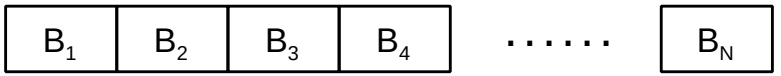
- Como o conceito de microprogramação pode ser utilizado para implementar a unidade de controle ?
  - Para cada microoperação, a unidade de controle ativa ou desativa um conjunto de sinais
  - Cada linha de controle é representada por um dígito binário
  - Uma palavra de controle é um conjunto de bits onde cada bit representa uma linha de controle
  - Cada microoperação pode ser representada por um padrão de 0s e 1s na palavra de controle

Implementação de controle

- Microinstruções horizontais



(a) Exemplo de um formato de microinstrução horizontal



B_1 - porta lógica CI → REM
B_2 - porta lógica ROM → RI
B_3 - sinal relógio para CI
Bit B_1, B_2 ... B_N { 1 - ativa a linha; 0 - desativa a linha

(b) Exemplo de campo LCBI de uma microinstrução

B_1 - não desvia
B_2 - desvio incondicional
B_3 - desviar se reg. = 0
B_4 - desviar se OUL

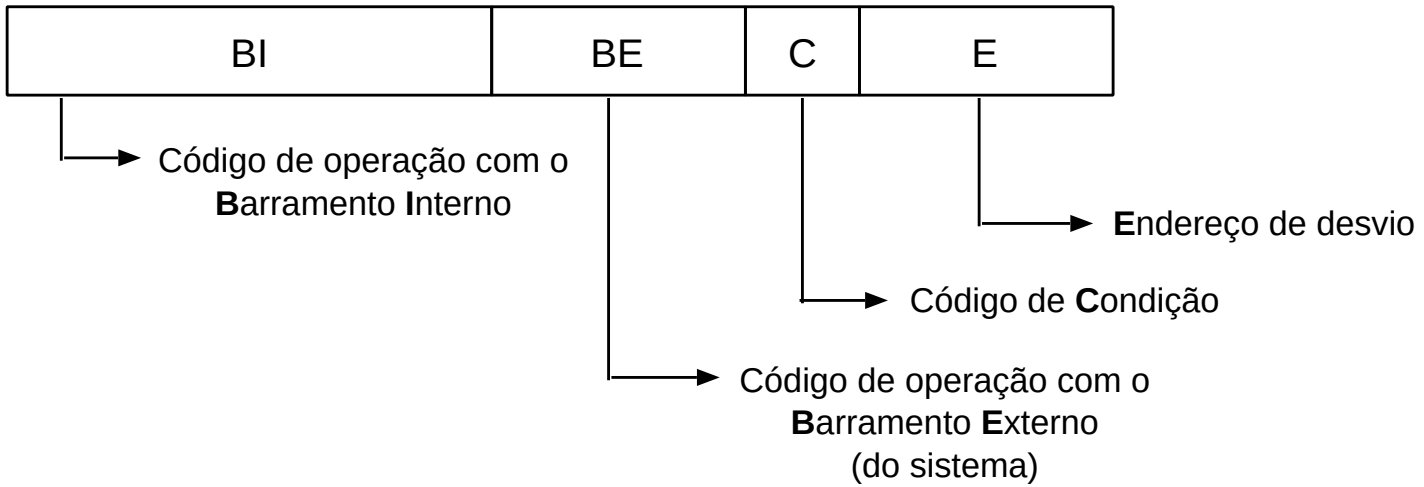
(c) Exemplo de campo BC de uma microinstrução

(Fig. 6.49 do livro texto)

- Linhas de controle ativam sinais de controle que fazem com que uma ou mais microoperações sejam executadas
- Se condição especificada no campo de condição for falsa, executa a próxima microinstrução da seqüência
- Se condição verdadeira, o endereço da próxima microinstrução será dado pelo campo Endereço da Próxima MicroInstrução

Implementação de controle

- Microinstruções verticais

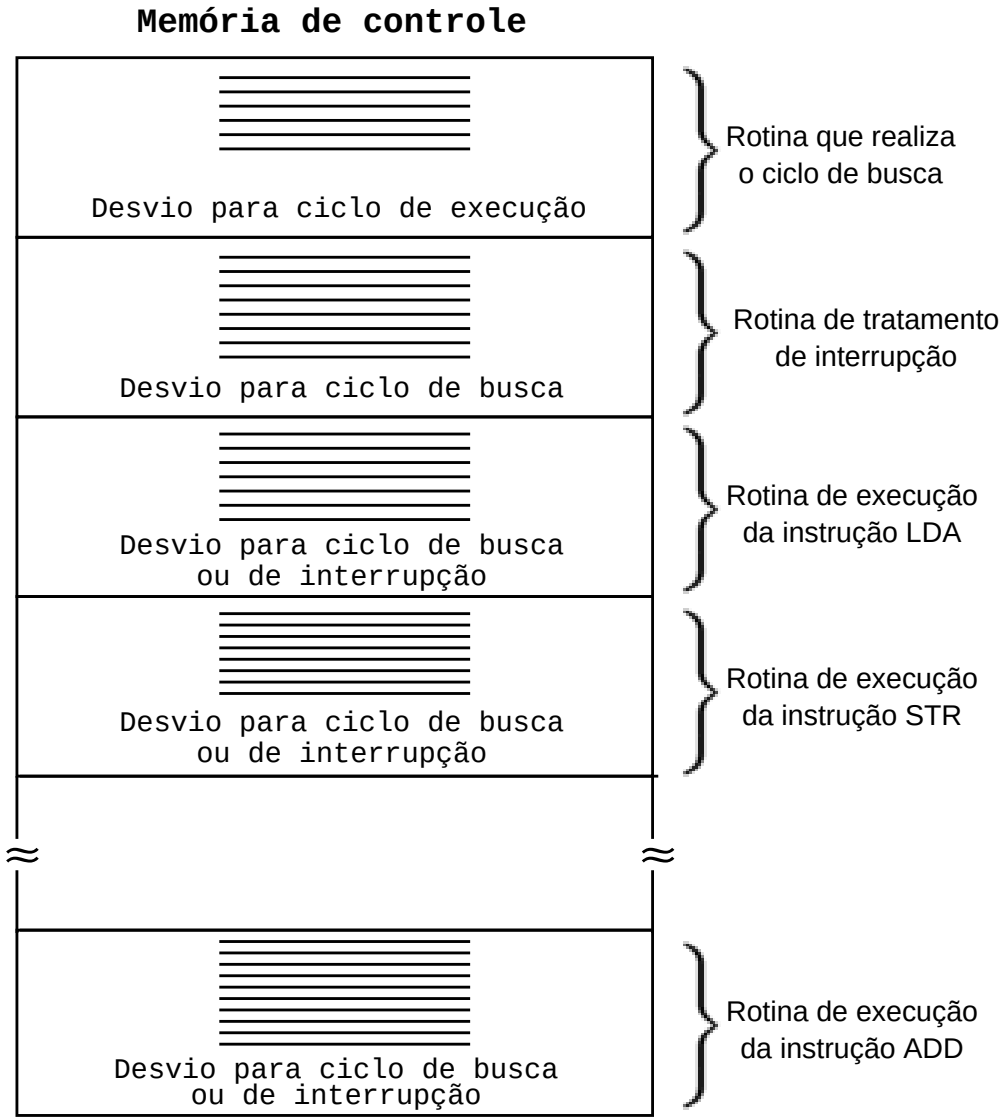


(Fig. 6.50 do livro texto)

- Os bits das linhas de controle não acessam diretamente os sinais de controle
- O campo BI contém um código de um grupo de ações que deve ser decodificado para gerar os sinais de controle de forma correta
- Utilizado para diminuir o número de bits das microinstruções

Implementação de controle

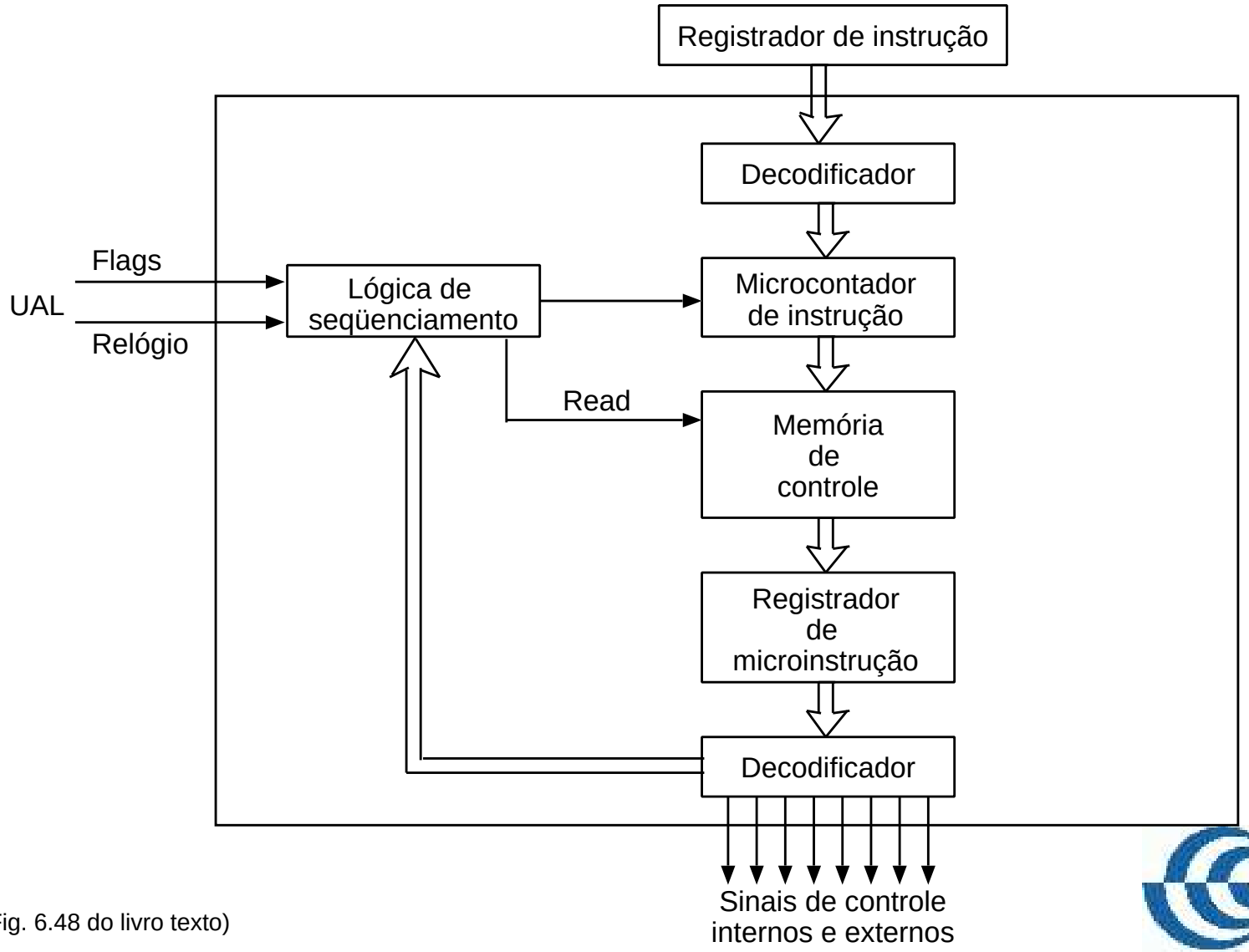
- Organização da memória de controle



(Fig. 6.51 do livro texto)

Implementação de controle

- Projeto de uma unidade de controle microprogramada



(Fig. 6.48 do livro texto)

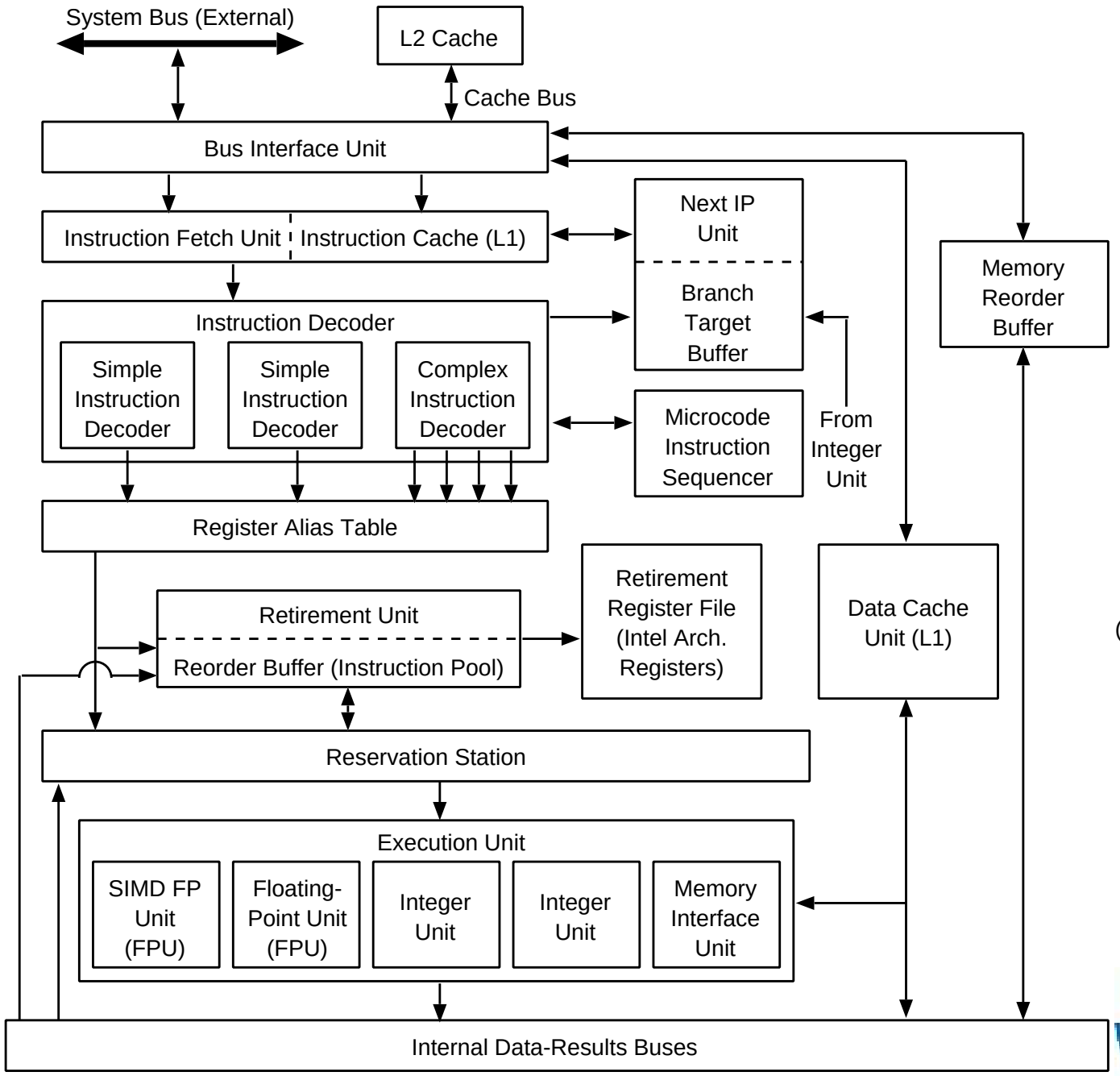
Implementação de controle

- Vantagens de uma unidade de controle microprogramada
  - Simplifica o projeto da unidade de controle
  - Implementação mais barata e menos sujeita a erro
  - Lógica de seqüenciamento e dos decodificadores possuem uma lógica mais simples que a lógica exigida na implementação por hardware

Implementação de controle

- Desvantagens de uma unidade de controle microprogramada
  - Mais lenta que a unidade implementada por hardware
  - Unidades de controle de máquinas CISC implementadas por microprogramação, devido à facilidade de implementação
  - Os processadores RISC possuem formato de instrução mais simples e as unidades de controle são tipicamente implementadas por hardware

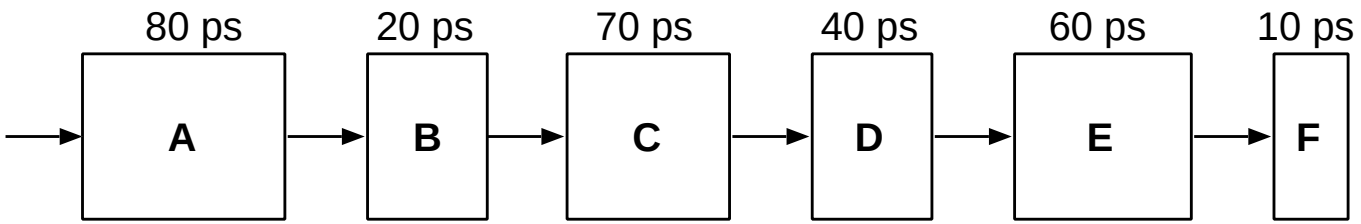
Diagrama em blocos dos processadores da família P6



(Fig. 6.61 do livro texto)

Exercício

- Analisando-se uma unidade central de processamento, verificou-se que suas instruções podem ser separadas em uma seqüência de 6 blocos (A-F), que apresentam os seguintes tempos de execução: 80, 20, 70, 40, 60 e 10 ps, onde p=10^{-12}, como mostrado na figura abaixo:



Exercício

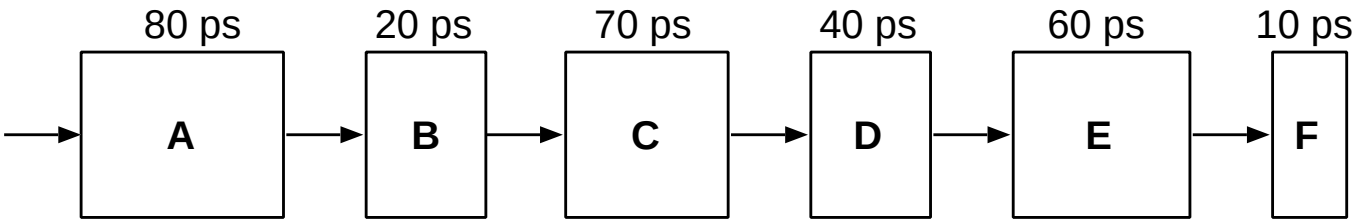
- Existem várias maneiras de se criar uma arquitetura utilizando-se pipeline para esta máquina. Indique como você implementaria esta arquitetura para cada um dos casos abaixo, indicando como as operações seriam divididas em estágios de pipeline. Para cada caso indique também o tempo de execução de uma seqüência de 10 instruções.

a) Pipeline em 2 estágios
b) Pipeline em 3 estágios
c) Pipeline em 4 estágios
d) Pipeline em 5 estágios

Exercício

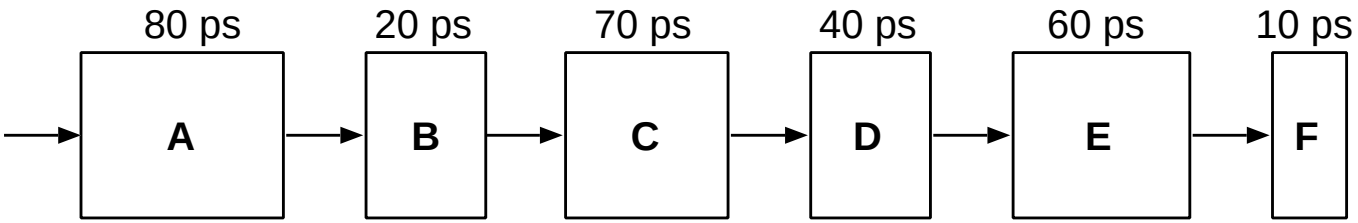
- Para cada caso, deve-se agrupar os blocos em cada estágio, de modo que os estágios possuam tempos de execução parecidos.

  - Pipeline em dois estágios
  - Os blocos A, B e C devem ficar no primeiro estágio com um tempo de execução de 170 ps e os blocos D, E e F no segundo estágio com tempo de execução de 110 ps. Logo a máquina deverá ser projetada de modo que cada estágio possua o maior dos tempos de execução dentre os estágios, que no caso é 170 ps. O tempo de execução de 10 instruções será : $340 + 9 \times 170 = 1870$ ps



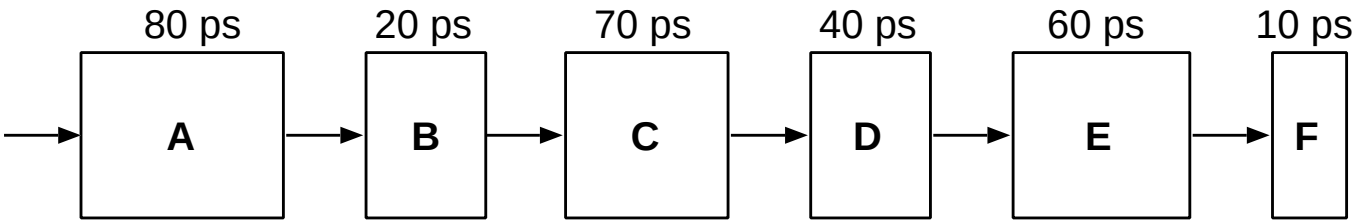
Exercício

- Pipeline em três estágios
- Os blocos A e B devem ficar no primeiro estágio com um tempo de execução de 100 ps, os blocos C e D no segundo com tempo de execução de 110 ps e os blocos E e F no terceiro com tempo de execução de 70 ps. Logo a máquina deverá ser projetada de modo que cada estágio possua o maior dos tempos de execução dentre os estágios, que no caso é 110 ps. O tempo de execução de 10 instruções será : $330 + 9 \times 110 = 1320$ ps



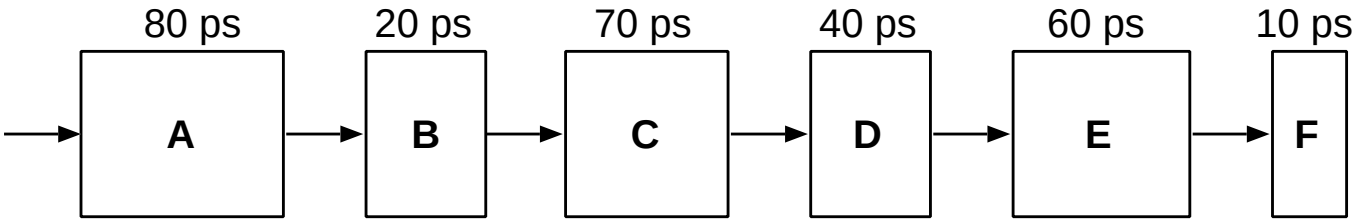
Exercício

- Pipeline em quatro estágios
- O bloco A deve ficar no primeiro estágio com um tempo de execução de 80 ps, os blocos B e C no segundo com tempo de execução de 90 ps, o bloco D no terceiro estágio com duração de 40 ps e os blocos E e F no quarto com tempo de execução de 70 ps. Logo a máquina deverá ser projetada de modo que cada estágio possua o maior dos tempos de execução dentre os estágios, que no caso é 90 ps. O tempo de execução de 10 instruções será : $360 + 9 \times 90 = 1170$ ps



Exercício

- Pipeline em cinco estágios
- O bloco A deve ficar no primeiro estágio com um tempo de execução de 80 ps, o bloco B no segundo com duração igual a 20 ps, C no terceiro com tempo de execução de 70 ps, o bloco D no quarto estágio com duração de 40 ps e os blocos E e F no quinto com tempo de execução de 70 ps. Logo a máquina deverá ser projetada de modo que cada estágio possua o maior dos tempos de execução dentre os estágios, que no caso é 80 ps. O tempo de execução de 10 instruções será : $400 + 9 \times 80 = 1120$ ps



Exercícios

- Capítulo 6 do livro texto
  - 23, 24 e 25