Arquitetura e Organização de Computadores

Unidade Central de Processamento (CPU)

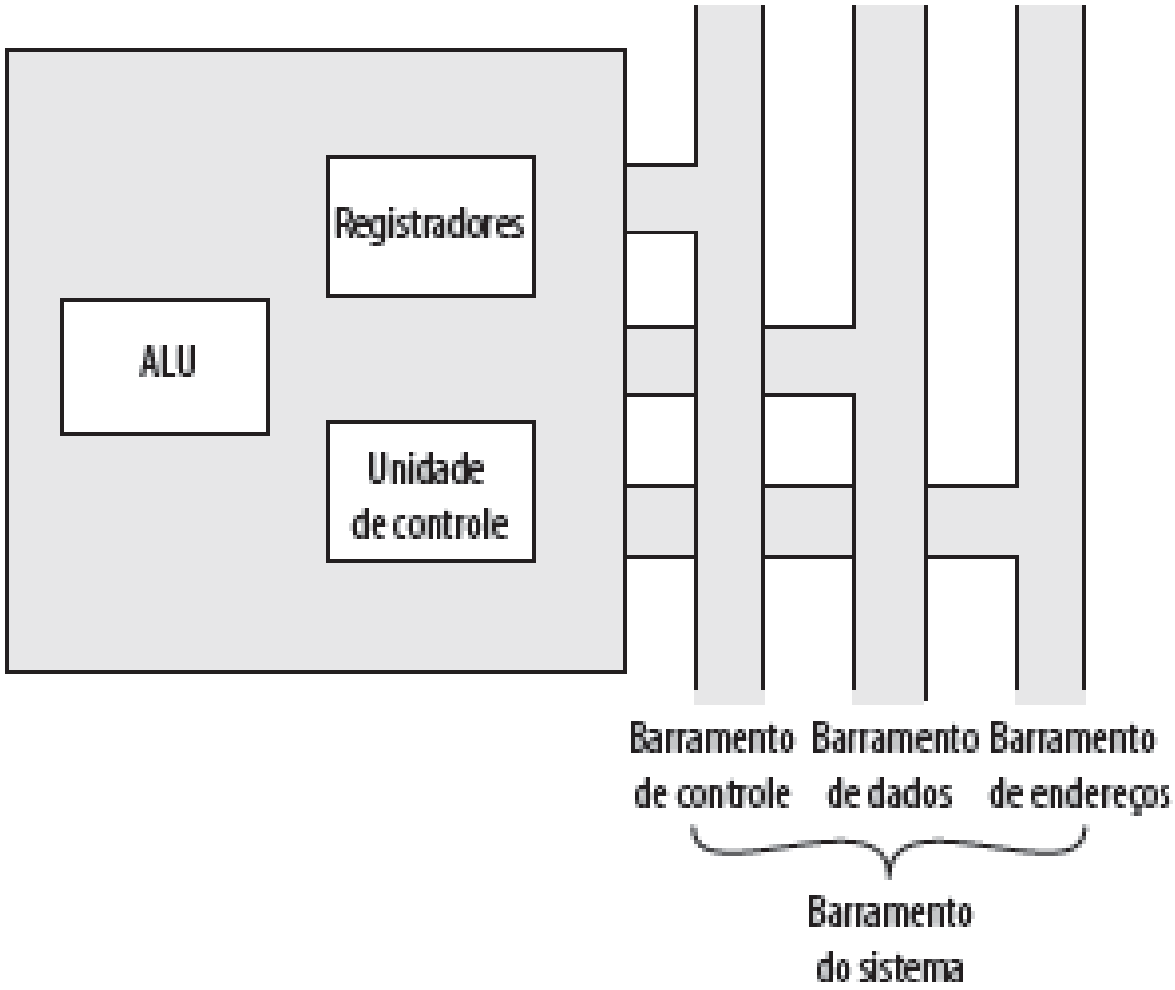
Givanaldo Rocha de Souza
http://docente.ifrn.edu.br/givanaldorocha
givanaldo.rocha@ifrn.edu.br

Baseado nos slides do capítulo 12 do livro “Arquitetura e Organização de Computadores” (William Stallings), 8ª edição.

Estrutura da CPU

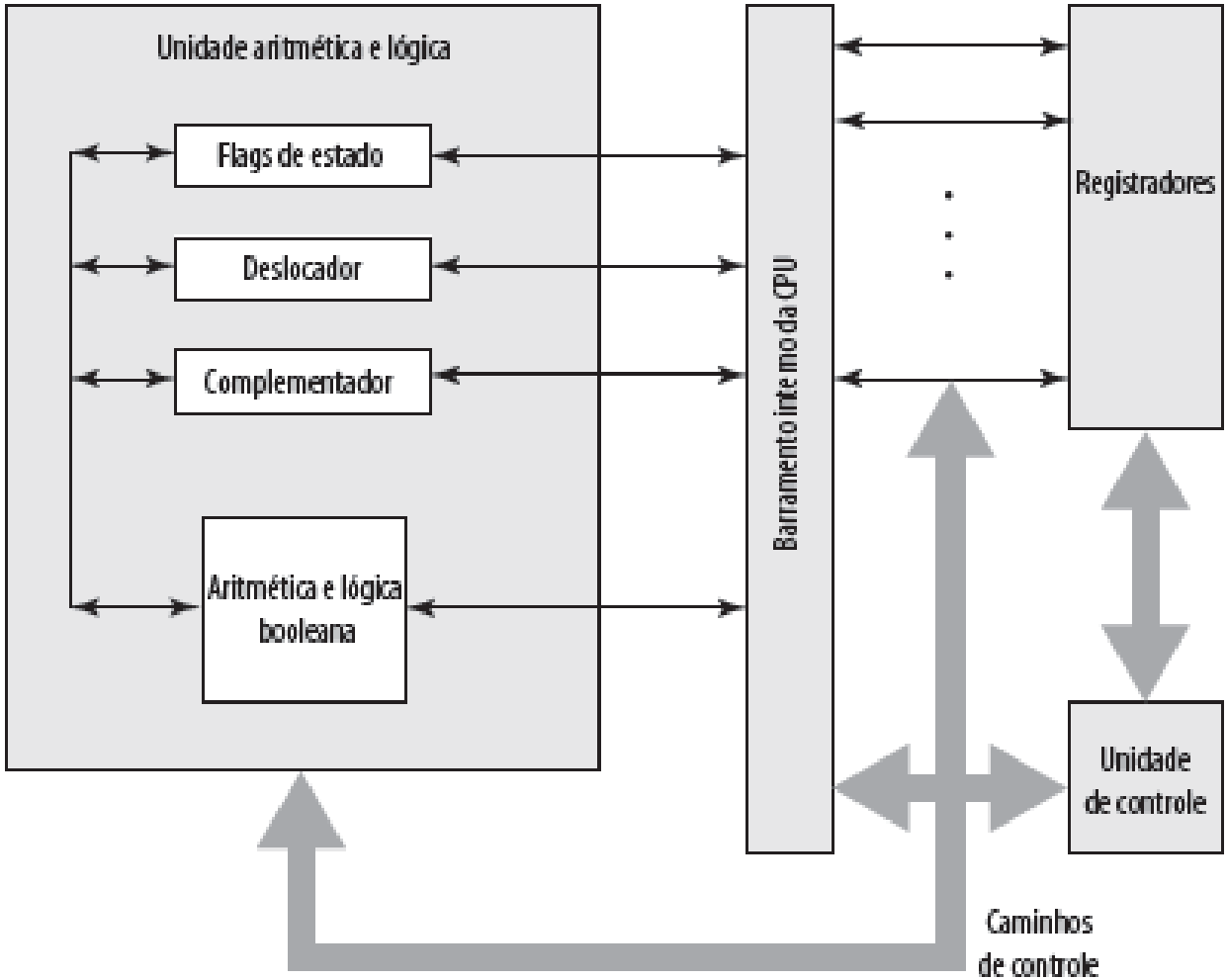
- CPU precisa:
  - Buscar instruções.
  - Interpretar instruções.
  - Obter dados.
  - Processar dados.
  - Gravar dados.

CPU com barramento de sistemas
Visão simplificada



Estrutura interna da CPU

Visão mais detalhada



Registradores

- CPU precisa ter algum espaço de trabalho (armazenamento temporário) → uso dos **registradores**.
- Número e função dos registradores variam entre projetos de processadores.
- Uma das principais decisões de projeto, pois influirá diretamente no desempenho.
- Alto nível de hierarquia de memória (acima da memória cache e da memória principal).

Registradores visíveis ao usuário

- Possibilitam que o programador de linguagem de máquina ou de montagem minimize as referências à memória principal.

- Uso geral.
- Dados.
- Endereços.
- Códigos condicionais.

Registradores de uso geral

- Podem ser de propósito geral verdadeiro.
- Podem ser restritos.
- Podem ser usados para dados ou endereçamento.
- Dados:
  - Acumulador.
- Endereçamento:
  - Segmento.

Registradores de uso geral

- Torne-os de uso geral:
  - Aumente flexibilidade e opções do programador.
  - Aumente tamanho de instrução e complexidade.

- Torne-os especializados:
  - Instruções menores (mais rápidas).
  - Menos flexibilidade.

Quantos registradores de uso geral?

- Entre 8 –32.
- Menos → mais referências à memória.
- Mais → não reduz as referências à memória e ocupa espaço no processador.

De qual tamanho?

- Grande o suficiente para manter endereço completo.
- Grande o suficiente para manter palavra completa.
- Normalmente, é possível combinar dois registradores de dados.
  - Programação C.
  - Double int a.
  - Long int a.

Registradores de código condicional

- Conjuntos de bits individuais.
  - Exemplo: resultado da última operação foi zero.
- Podem ser lidos (implicitamente) por programas.
  - Exemplo: Jump if zero.
- Não podem (normalmente) ser alterados por programas.

Registradores de controle e estado

- Contador de programa (PC).
  - Contém o endereço de uma instrução a ser lida.
- Registrador de decodificação de instrução (IR).
  - Contém a instrução lida mais recentemente.
- Registrador de endereço de memória (MAR).
  - Contém o endereço de uma posição de memória.
- Registrador de buffer de memória (MBR).
  - Contém uma palavra de dados para ser escrita na memória ou a palavra lida mais recentemente.

Palavra de estado do programa

- Um conjunto de bits.
- Inclui *flags* (códigos condicionais).
- Sinal do último resultado.
- Zero.
- *Carry.*
- Igual.
- *Overflow.*
- Habilitar/desabilitar interrupção.
- Supervisor.

Modo supervisor

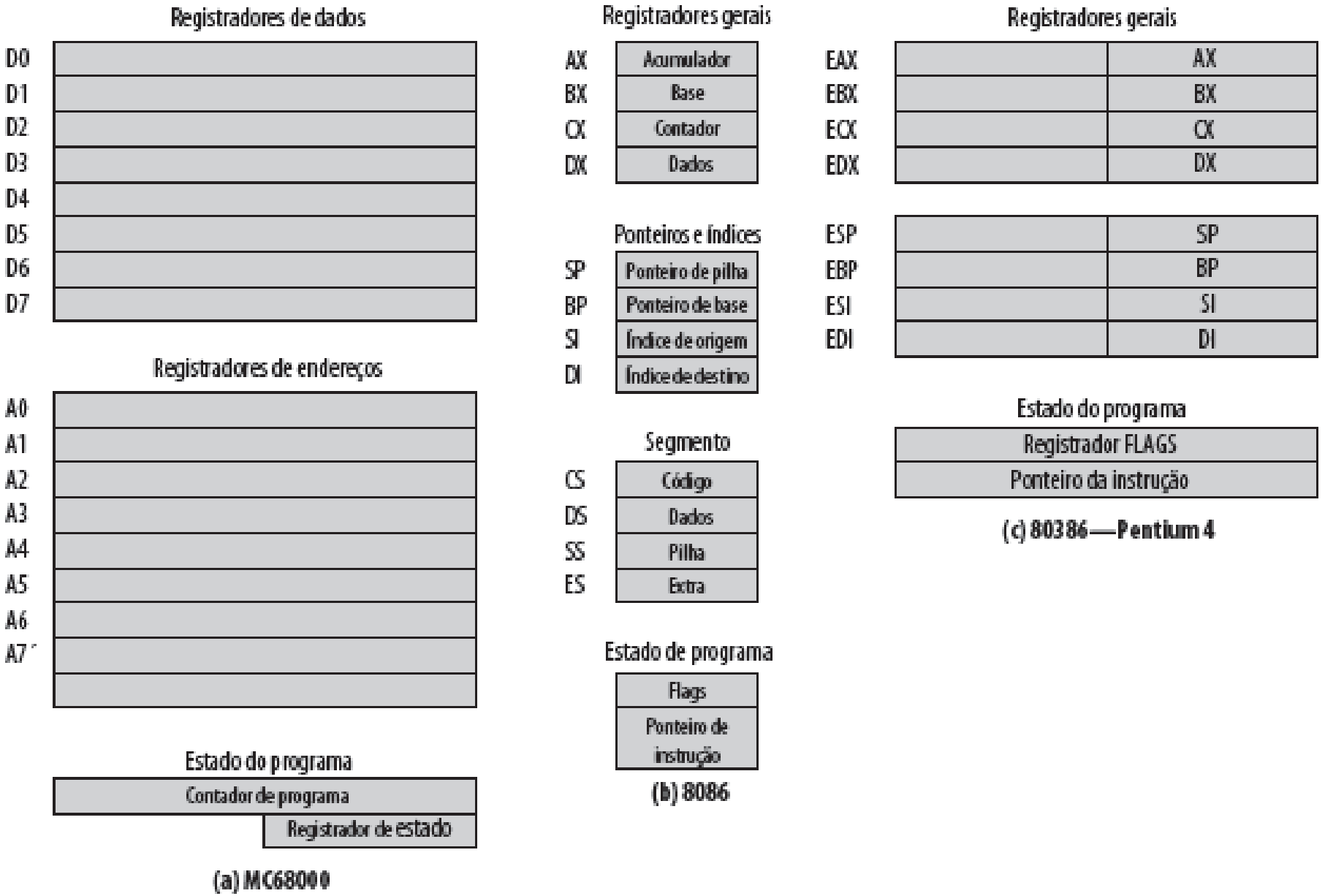
- Modo kernel.
- Permite execução de instruções privilegiadas.
- Usado pelo sistema operacional.
- Não disponível aos programas do usuário.

Outros registradores

- Podem ter registradores apontando para:
  - Blocos de controle de processo.
  - Vetores de interrupção.

- OBS.: Projeto da CPU e projeto do sistema operacional são bastante interligados.

Exemplo de organizações de registradores



Fluxo de dados (busca de instrução)

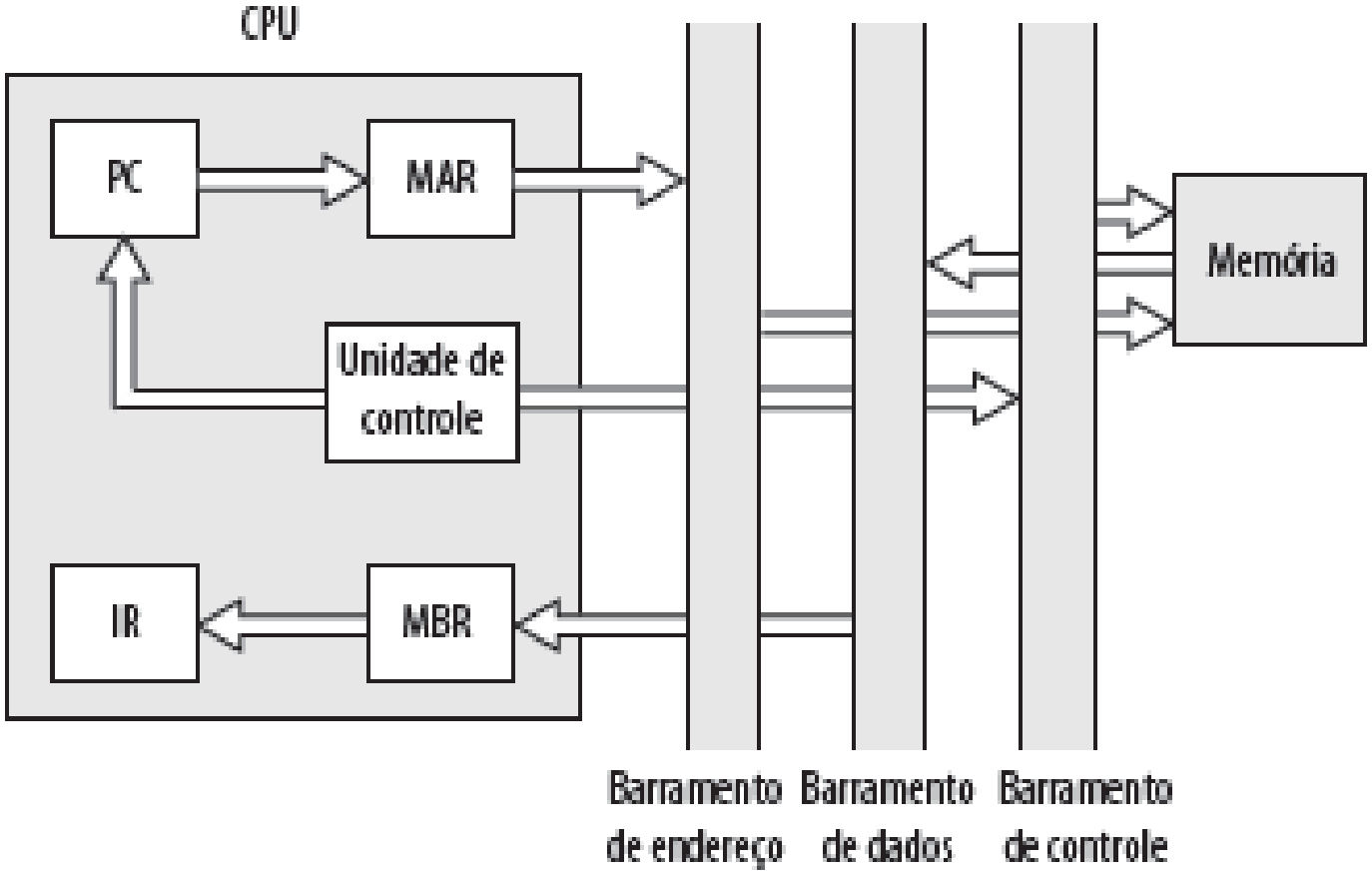
- Depende do projeto da CPU.
- Em geral:

- Busca:
  - PC contém endereço da próxima instrução.
  - Endereço movido para MAR.
  - Endereço colocado no barramento de endereço.
  - Unidade de controle solicita leitura de memória.
  - Resultado colocado no barramento de dados, copiado para MBR, depois para IR.
  - Enquanto isso, PC incrementado em 1.

Fluxo de dados (busca de dados)

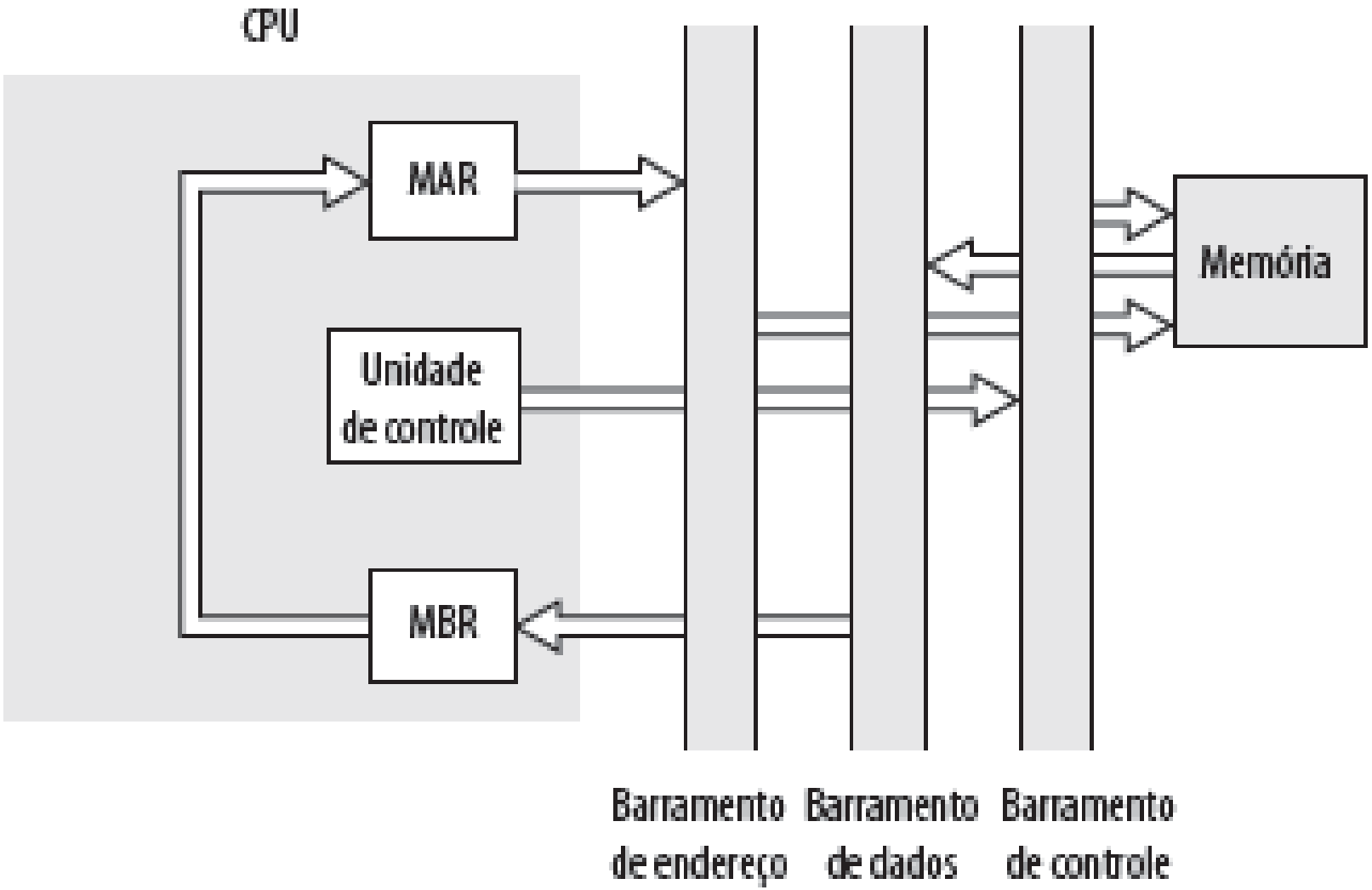
- IR é examinado.
- Se endereçamento indireto, ciclo indireto é realizado.
  - N bits da extrema direita do MBR transferidos para MAR.
  - Unidade de controle solicita leitura de memória.
  - Resultado (endereço do operando) movido para MBR.

Fluxo de dados (ciclo de busca)



MBR = registrador de buffer de memória
MAR = registrador de endereço de memória
IR = registrador da instrução
PC = contador de programa

Fluxo de dados (ciclo indireto)



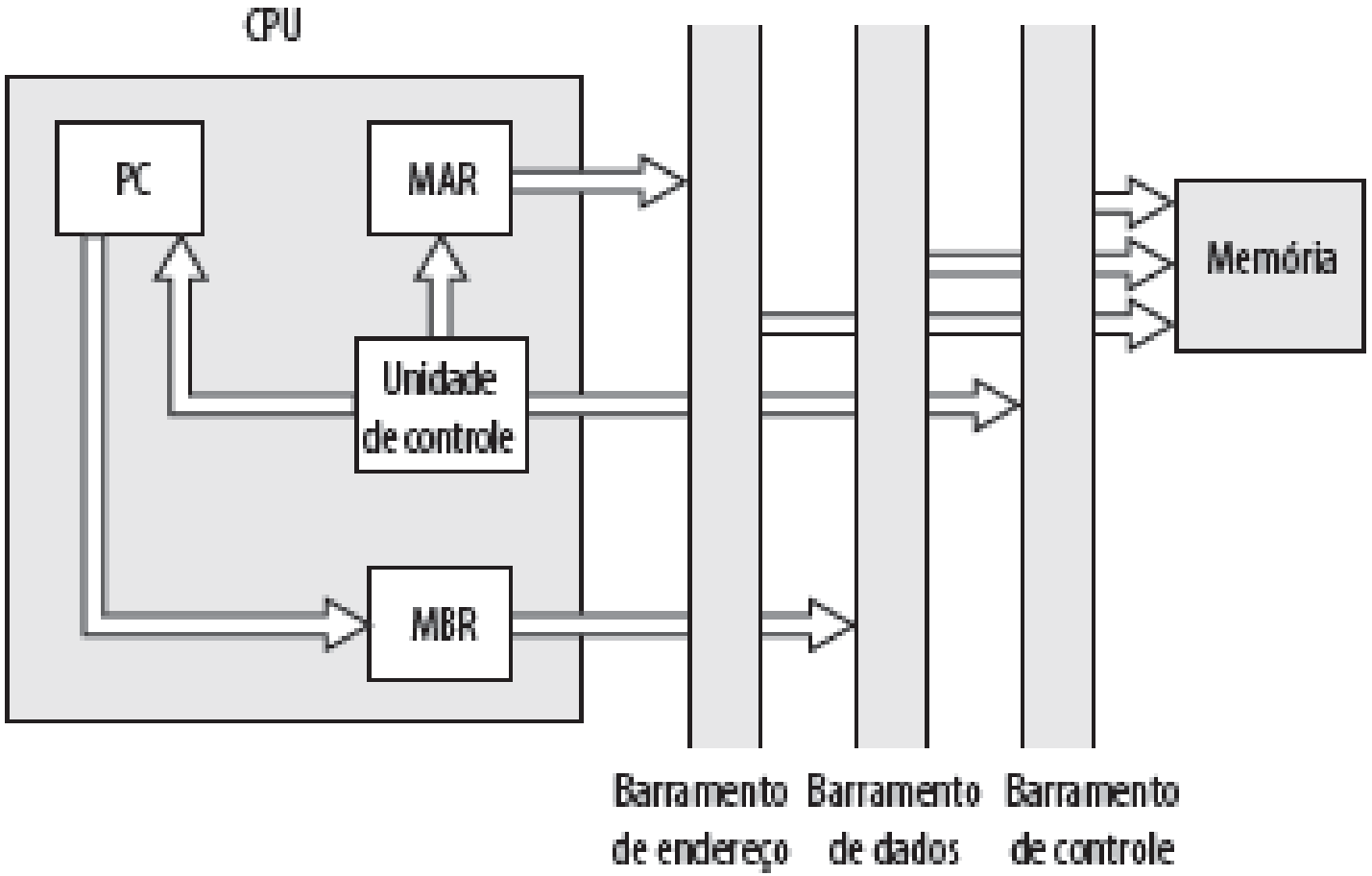
Fluxo de dados (execução)

- Pode tomar muitas formas.
- Depende da instrução sendo executada.
- Pode incluir:
  - Leitura/escrita da memória.
  - Entrada/saída.
  - Transferências de registradores.
  - Operações da ALU.

Fluxo de dados (interrupção)

- Simples.
- Previsível.
- PC atual salvo para permitir retomada após interrupção.
- Conteúdo do PC copiado para MBR.
- Local especial da memória (p.e., ponteiro de pilha) carregado no MAR.
- MBR gravado na memória.
- PC carregado com endereço da rotina de tratamento de interrupção.
- Próxima instrução (primeira do tratador de interrupção) pode ser obtida.

Fluxo de dados (ciclo de interrupção)



Pipelining

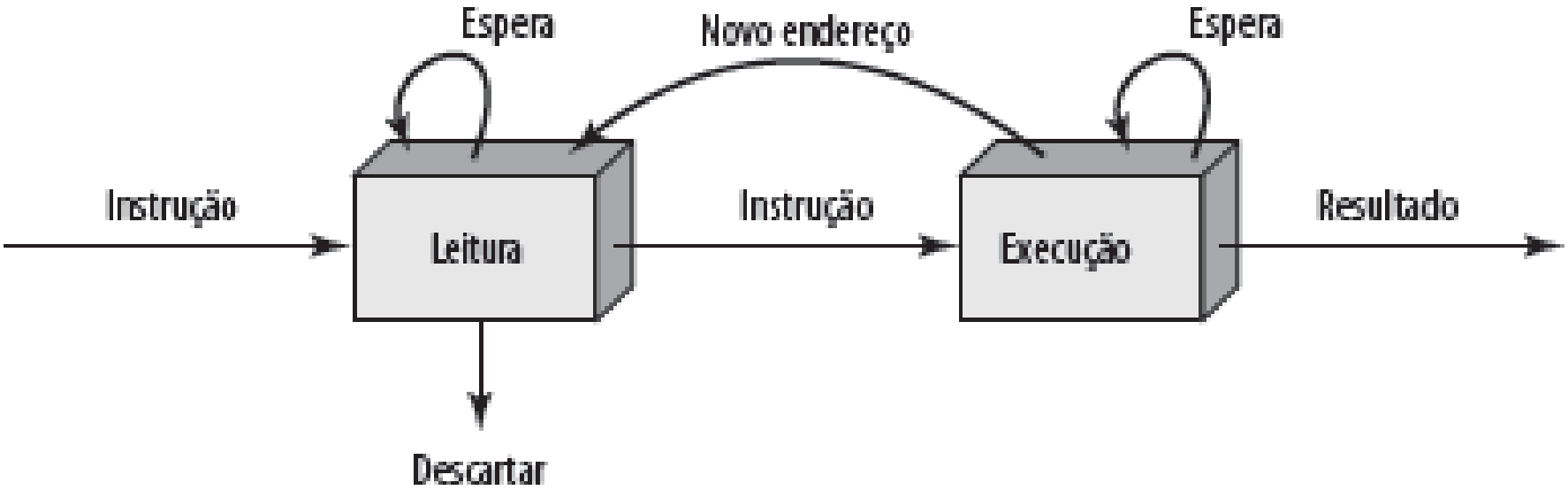
- Semelhante a uma linha de montagem industrial, onde produtos em vários estágios podem ser trabalhados simultaneamente.

- Estágio do processamento da instrução:
  - Buscar instrução (FI).
  - Decodificar instrução (DI).
  - Calcular operandos (CO).
  - Buscar operandos (FO).
  - Executar instrução (EI).
  - Escrever operando (WO).

- Sobrepor estas operações.

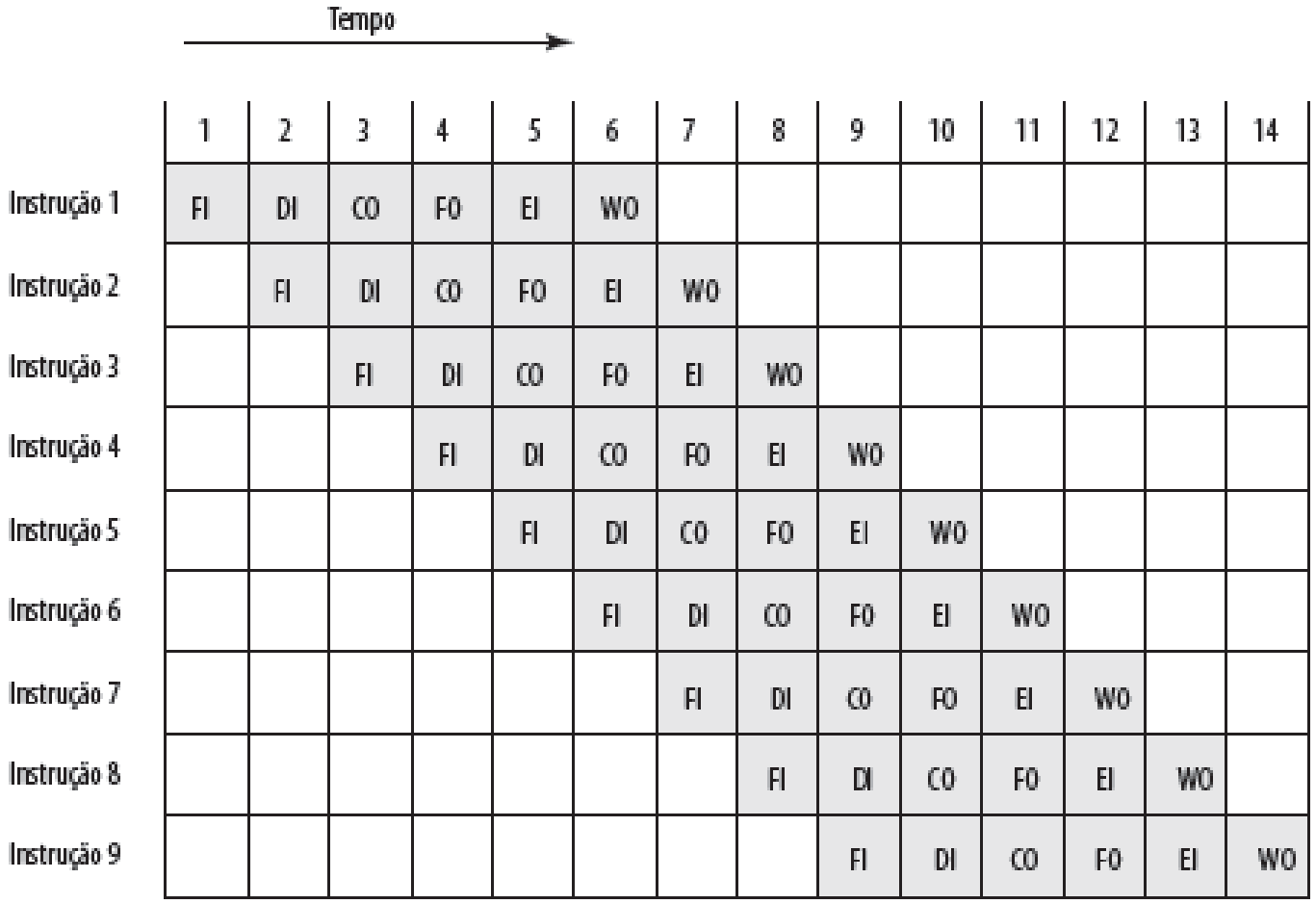
Pipeline de instrução de dois estágios

(a) Visão simplificada

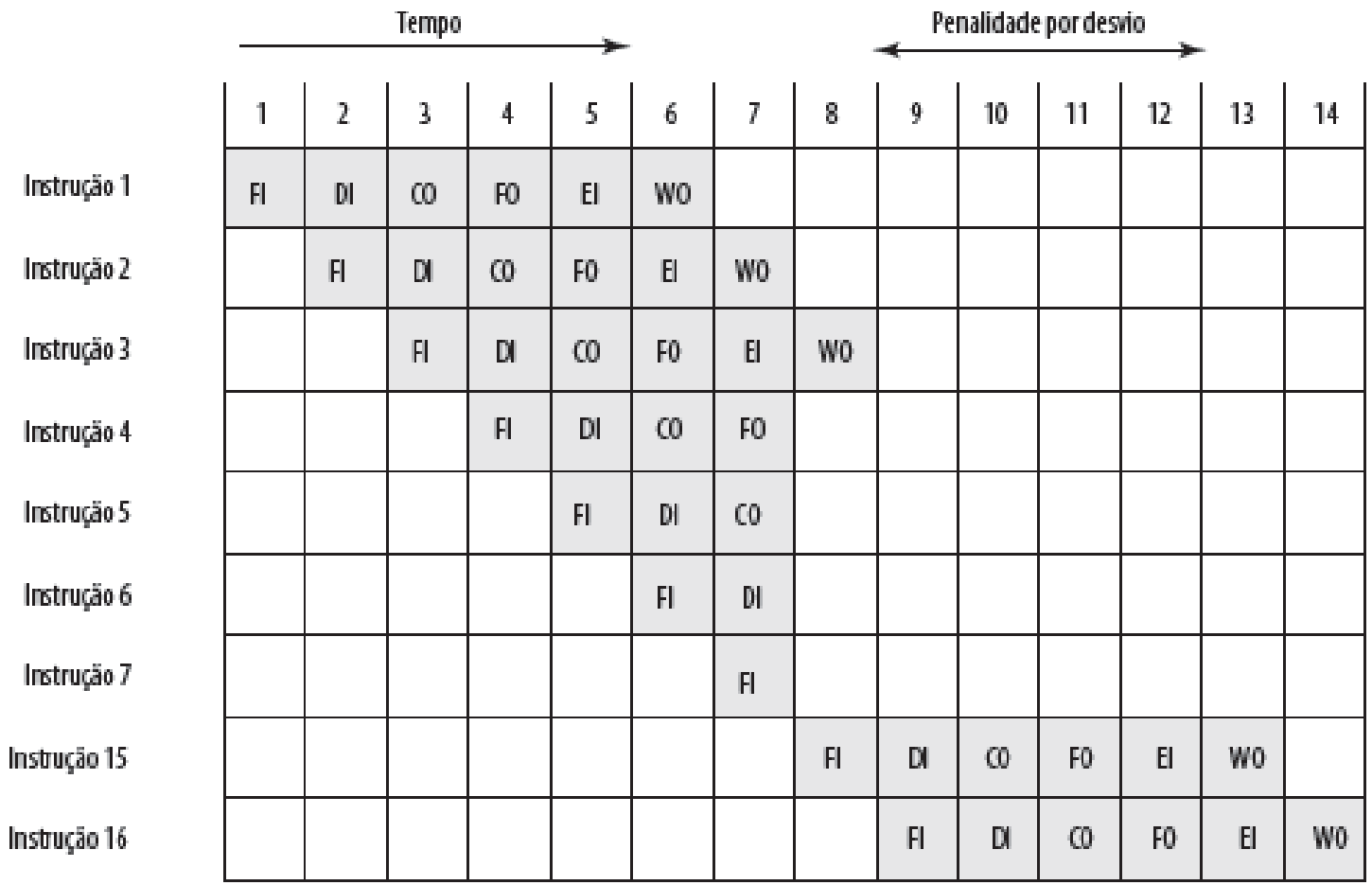


(b) Visão expandida

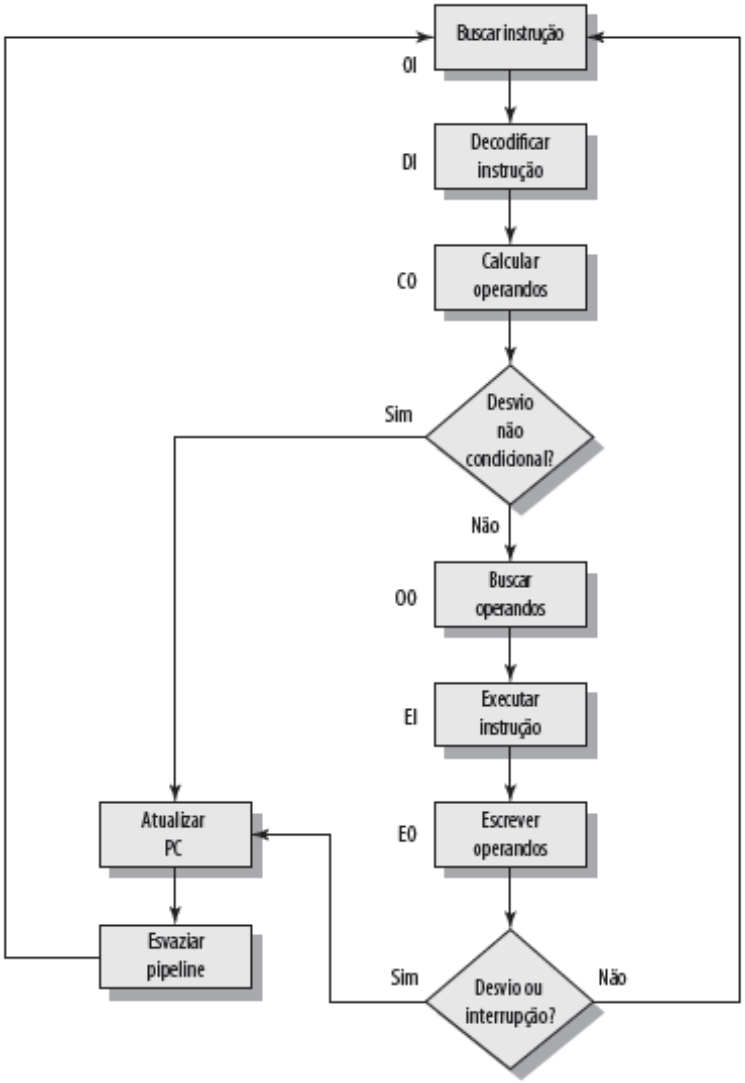
Diagrama de tempo para a operação do pipeline da instrução



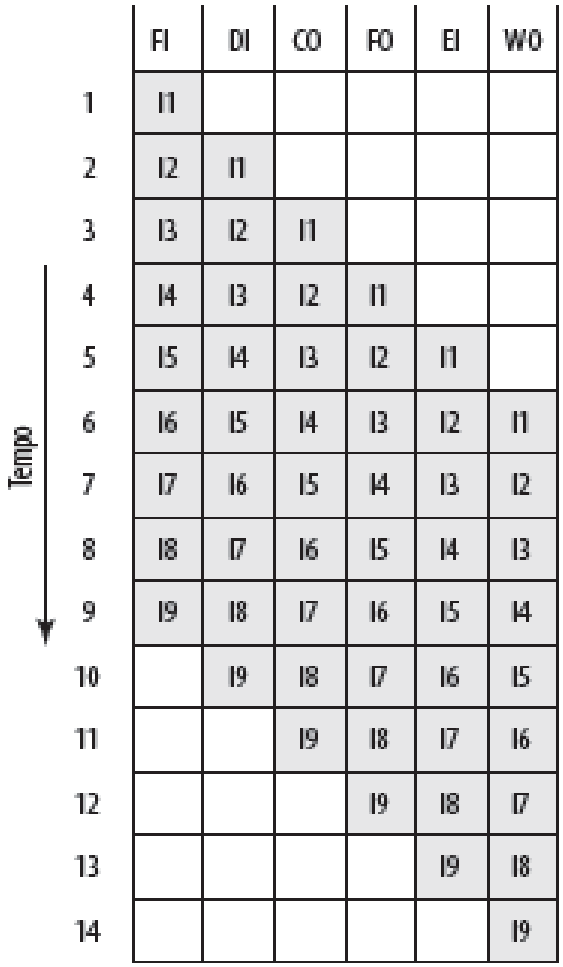
Efeito de desvio condicional na operação do pipeline na instrução



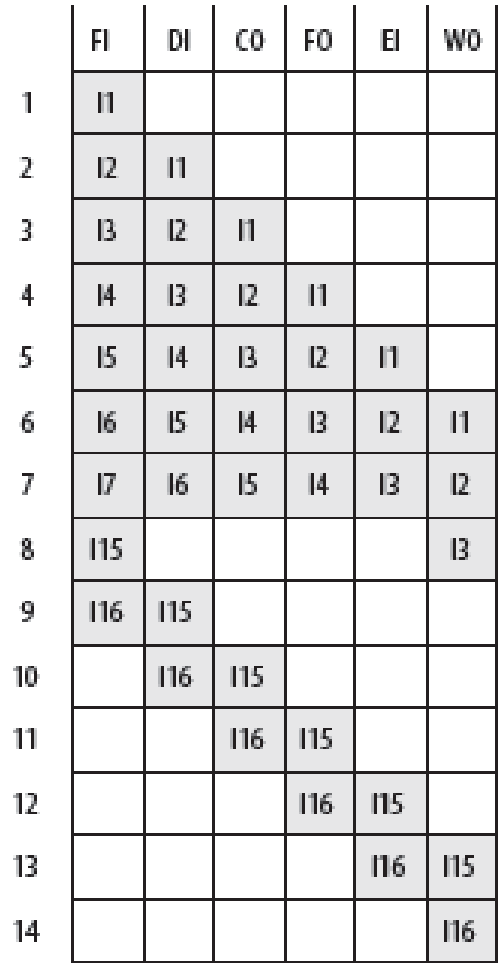
Pipeline de instrução de seis estágios



Descrição alternativa de um pipeline



(a) Sem desvios



(b) Com desvios condicionais

Hazards do pipeline

- Pipeline, ou alguma parte do pipeline, precisa parar porque as condições não permitem a execução contínua.
- Também conhecida como *bolha de pipeline.*
- Tipos de hazards:
  - Recursos.
  - Dados.
  - Controle.

Hazards de recursos

- Duas (ou mais) instruções no pipeline precisam do mesmo recurso.
- Executados em série, e não em paralelo, parâmetro parte do pipeline.
- Também chamado *hazard estrutural.*
- No caso ideal, cada nova instrução entra no pipeline a cada ciclo de clock.
- Suponha que a memória principal tenha única porta.
- Considere buscas de instrução e leituras e escritas de dados uma por vez.
- Ignore a cache.
- Leitura ou escrita de operando não podem ser realizadas em paralelo com busca de instrução.
- Estágio de busca de instrução fica ocioso por um ciclo buscando I3.
- P.e., várias instruções prontas para entrar na fase de execução de instrução.
- Única ALU.
- Uma solução: aumentar recursos disponíveis.
  - Múltiplas portas da memória principal.
  - Múltiplas ALUs.

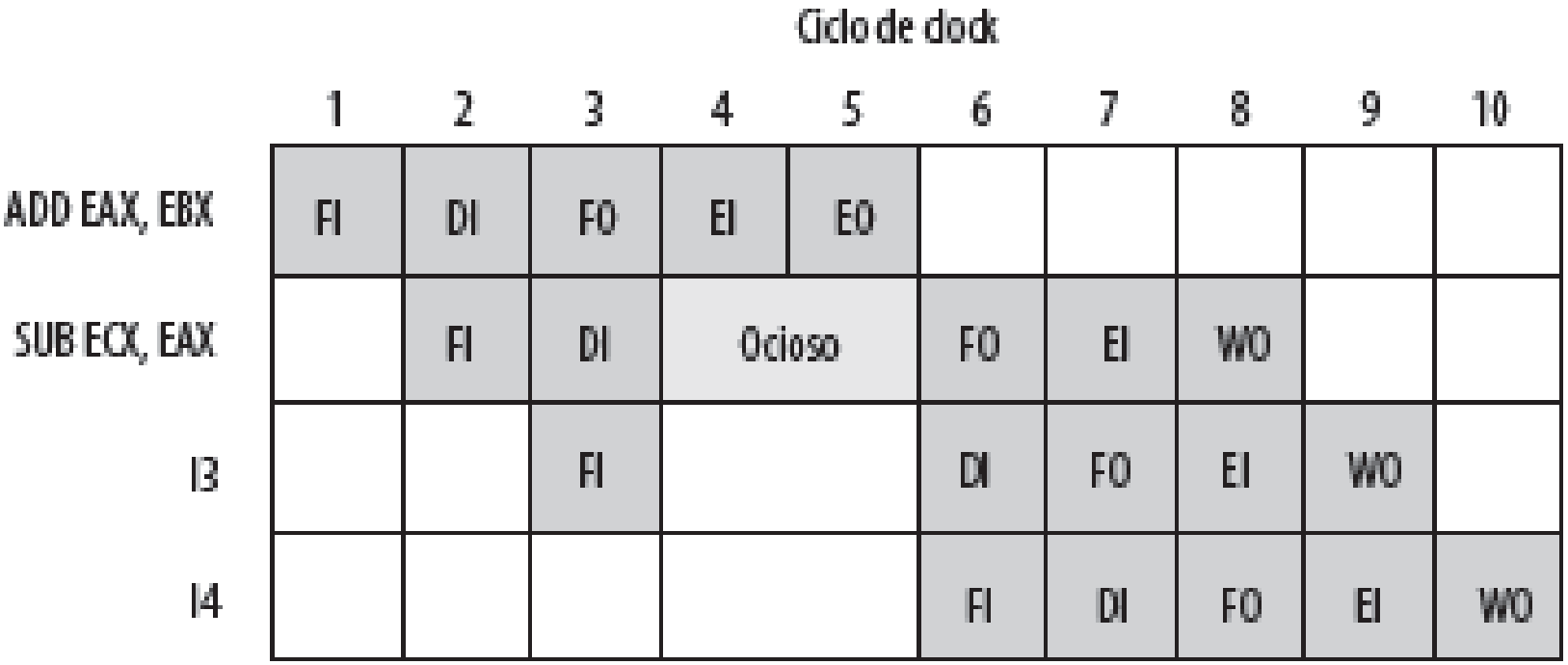
Hazards de dados

- Conflito no acesso de um local de operando.
- Duas instruções a serem executadas em sequência.
- Ambas acessam uma memória em particular ou operando do registrador.
- Se na sequência estrita, não ocorre problema.
- Se em um pipeline, valor do operando poderia ser atualizado para produzir resultado diferente da execução sequencial estrita.
- P.e., sequência de instruções de máquina do x86:

- ADD EAX, EBX /* EAX = EAX + EBX
- SUB ECX, EAX /* ECX = ECX – EAX

- Instrução ADD não atualiza EAX até o fim do estágio 5, no ciclo de clock 5.
- Instrução SUB precisa do valor no início do seu estágio 2, no ciclo de clock 4.
- Pipeline precisa parar por dois ciclos de clock.
- Sem hardware especial e algoritmos de impedimento específicos, resulta em uso ineficaz do pipeline.

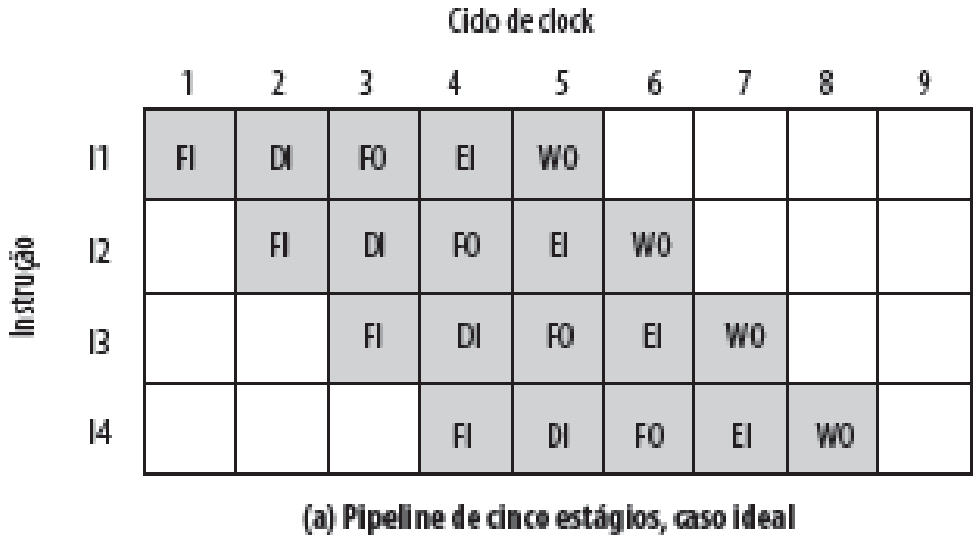
Exemplo de hazard de dados



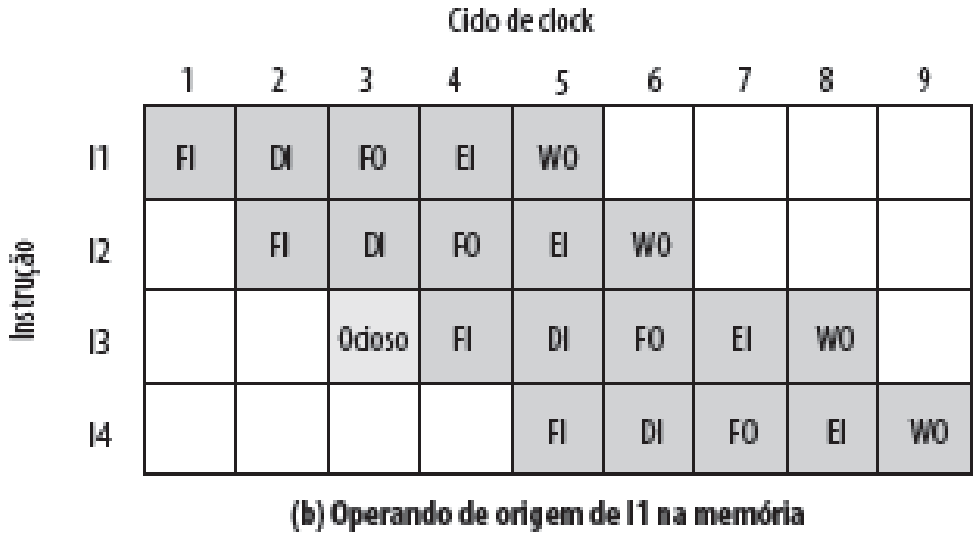
Tipos de hazard de dados

- Leitura após escrita (RAW), ou dependência verdadeira:
  - Uma instrução modifica um registrador ou local de memória.
  - Instrução seguinte lê dados nesse local.
  - Hazard se leitura ocorre antes do término da escrita.
- Escrita após leitura (WAR), ou antidependência:
  - Uma instrução lê um registrador ou local da memória.
  - Instrução seguinte escreve no local.
  - Hazard se escrita termina antes que ocorra a leitura.
- Escrita após escrita (WAW), ou dependência de saída:
  - Duas instruções escrevem no mesmo local.
  - Hazard se a escrita ocorre na ordem contrária à sequência intencionada.
- Exemplo anterior é um hazard RAW.

Exemplo de hazard de recursos



(a) Pipeline de cinco estágios, caso ideal



(b) Operando de origem de I1 na memória

Hazard de controle

- Também conhecido como *hazard de desvio.*
- Pipeline toma decisão errada sobre previsão de desvio.
- Traz para o pipeline instruções que precisam ser descartadas subsequentemente.

Pipeline de Intel 80486

- Leitura:
  - Da cache ou da memória externa.
  - Colocadas em um de 2 buffers de busca antecipada de 16 bits.
  - Enche buffer com novos dados quando antigos são consumidos.
  - Em média, 5 instruções lidas por carga.
  - Independente de outros estágios para manter buffers cheios.
- Estágio de decodificação 1:
  - *Opcode* e informação de modo de endereçamento.
  - No máximo 3 primeiros bytes da instrução.
  - Pode direcionar estágio D2 para obter restante da instrução.
- Estágio de decodificação 2:
  - Expande *opcode* para sinais de controle.
  - Cálculo de modos de endereçamento complexos.
- Execução:
  - Operações da ALU, acesso a cache, atualização de registrador.
- Escrita:
  - Atualiza registradores e flags.
  - Resultados enviados à cache e buffers de escrita da interface de barramento.

Exemplos de pipeline da instrução do 80486

(a) Nenhum atraso para carregar dados no pipeline

(b) Atraso para carregar ponteiro

(c) Temporização da instrução de desvio

Registradores do Pentium 4

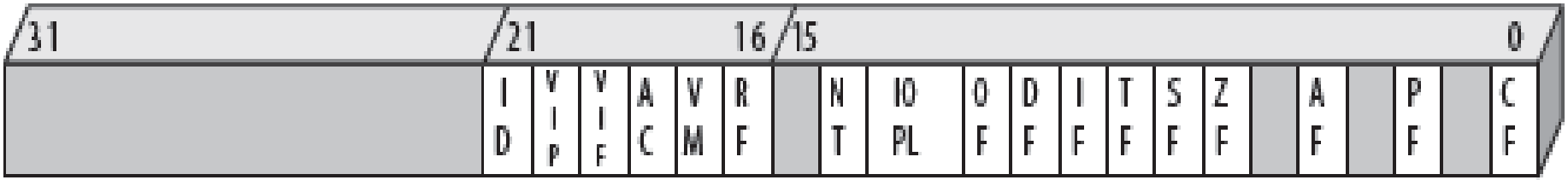
(a) Integer Unit

Type	Number	Length (bits)	Purpose
General	8	32	General-purpose user registers
Segment	6	16	Contain segment selectors
Flags	1	32	Status and control bits
Instruction Pointer	1	32	Instruction pointer

(b) Floating-Point Unit

Type	Number	Length (bits)	Purpose
Numeric	8	80	Hold floating-point numbers
Control	1	16	Control bits
Status	1	16	Status bits
Tag Word	1	16	Specifies contents of numeric registers
Instruction Pointer	1	48	Points to instruction interrupted by exception
Data Pointer	1	48	Points to operand interrupted by exception

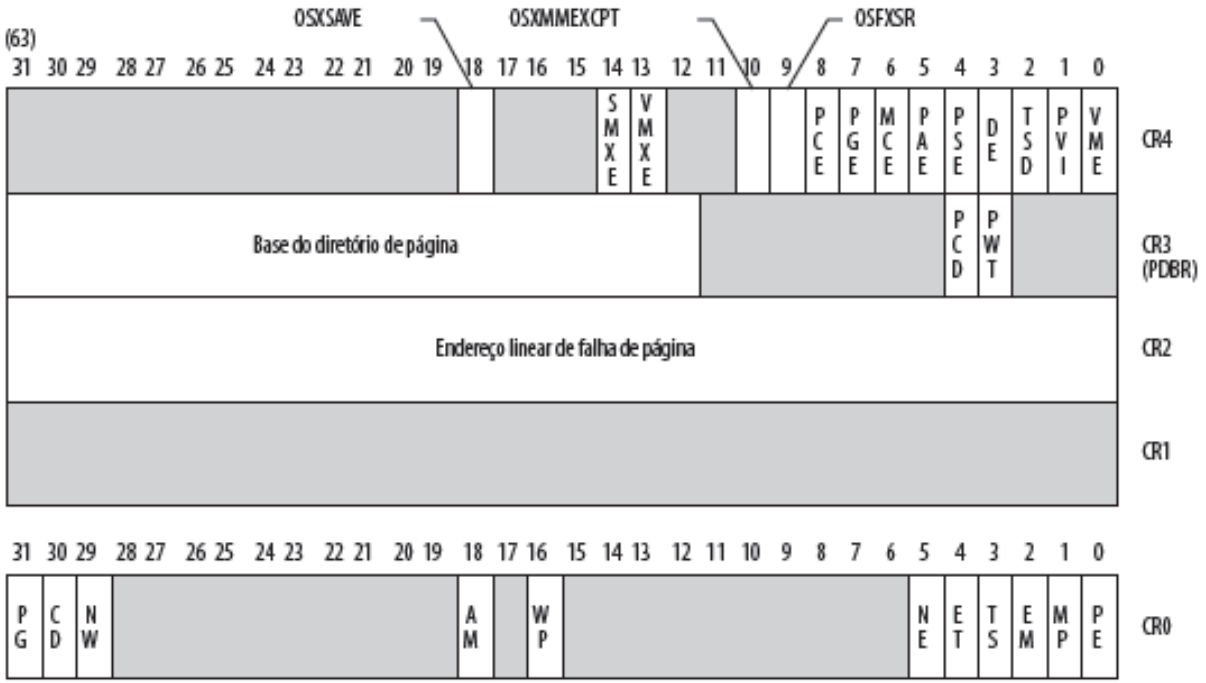
Registrador EFLAGS



ID = flag de identificação
VIP = Interrupção virtual pendente
VIF = flag de interrupção virtual
AC = Verificação de alinhamento
VM = 8086 modo virtual
RF = flag de resumo
NT = flag tarefa aninhada
IOPL = de privilégio de EIS
OF = flag de *overflow*

DF = flag direcional
IF = flag para habilitar interrupção
TF = flag de *trap*
SF = flag de *sinal*
ZF = flag Zero
AF = flag de *carry* auxiliar
PF = flag de paridade
CF = flag de *carry*

Registradores de controle de x86



Áreas sombreadas indicam bits reservados.

OSXSAVE = habilita bit XSAVE
SMXE = habilita extensões do modo de segurança
VMXE = habilita extensões de máquina virtual
OSXMMEXCPT = Suporta excessões SIMD FP não mascaradas
OSFXSR = Suporta FXSAVE, FXSTOR
PCE = habilita conector de desempenho
PGE = habilita paginação global
MCE = habilita verificação de máquina
PAE = extensão de endereço físico
PSE = extensões de tamanho de página
DE = extensões de depuração
TSD = desabilitar *time stamp*
PVI = interrupções virtuais no modo protegido
VME = modo de extensão virtual de 8086

PCD = desabilita cache de página
PWT = escrita transparente em nível de página
PG = paginação
CD = desabilita cache
NW = not write through
AM = máscara de alinhamento
WP = proteção de escrita
NE = erro numérico
ET = tipo de extensão
TS = troca de tarefa
EM = emulação
MP = monitor do coprocessador
PE = habilitação de proteção

Sugestões de pesquisa

- Verificar a composição dos processadores ARM e Intel e analisar suas diferenças de arquitetura e organização interna.
- Pesquisar mais sobre o mecanismo de pipeline, inclusive a sua influência no desempenho do processador.
- Pesquisar sobre o projeto de processadores específicos para aplicações de tempo real.