CPU - Estrutura e funcionamento

Organização e Arquitetura de Computadores.

Professora Danielle Morais de Andrade.

Grupo: João Victor Silva Jucá, Luis Felipe Melo, Paulo Domingos.

Organização do Processador

Busca da instrução;

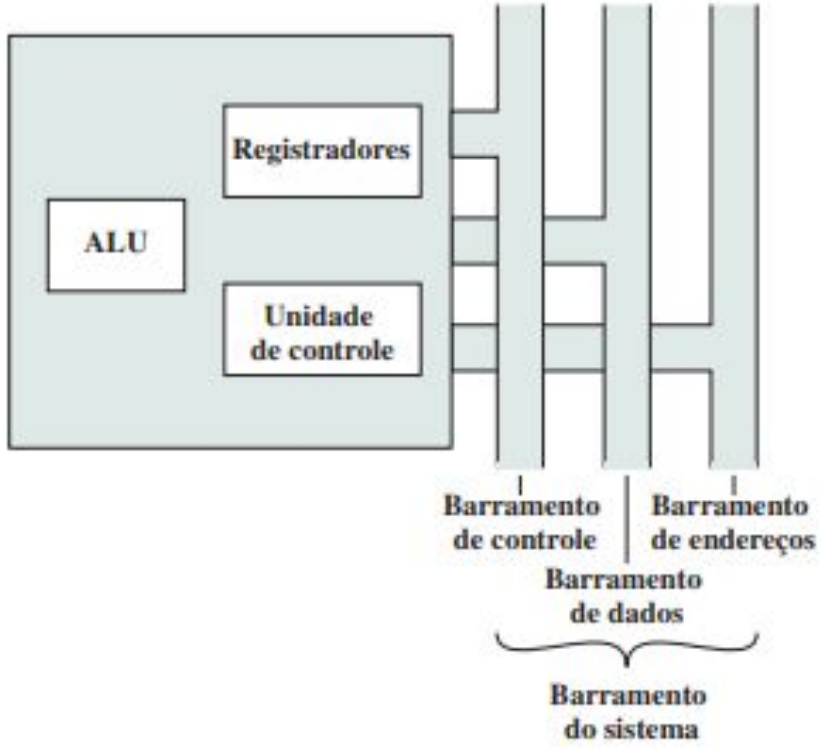
Interpretação da instrução

Busca dos dados;

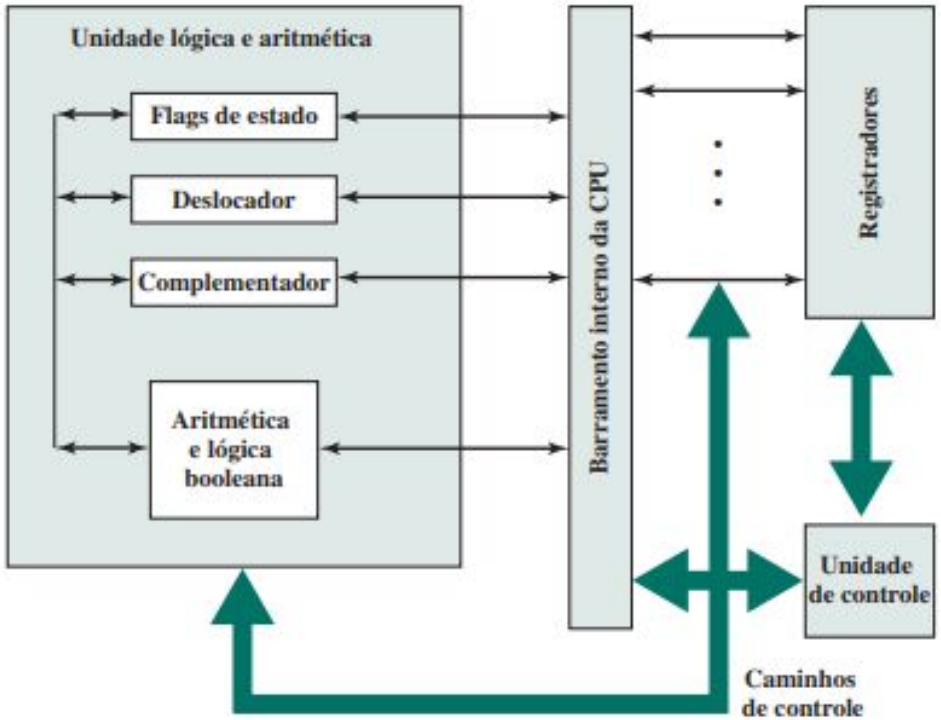
Processamento dos dados;

Escrita dos dados.

Organização do Processador



Organização do Processador



Organização dos registradores

Registradores visíveis ao usuário:

Registradores de uso geral;

Registradores de endereços:

Ponteiros de segmento;

Registradores de índice;

Ponteiros de pilha.

Registradores de códigos condicionais

Organização dos registradores

Registradores de controle e de estado:

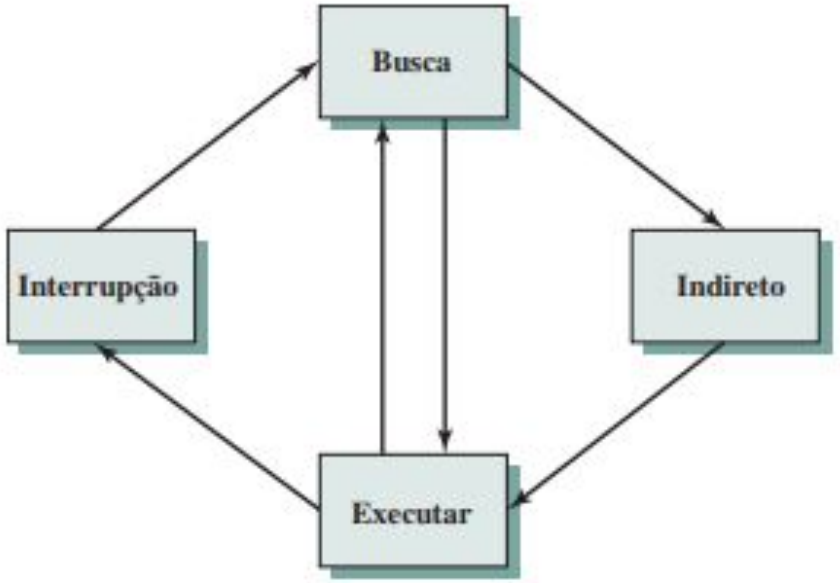
Contador de programas (PC):

Registrador da instrução

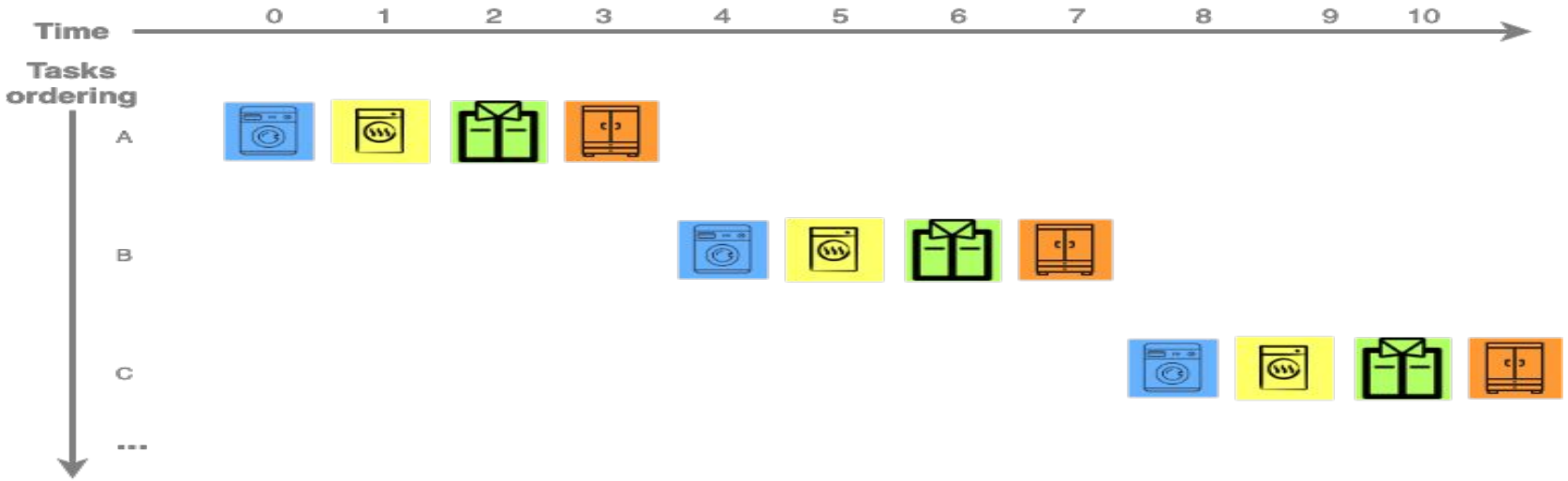
Registrador de endereço de memória (MAR):

Registrador de buffer de memória (MBR):

Ciclo da Instrução



Pipeline de Instrução



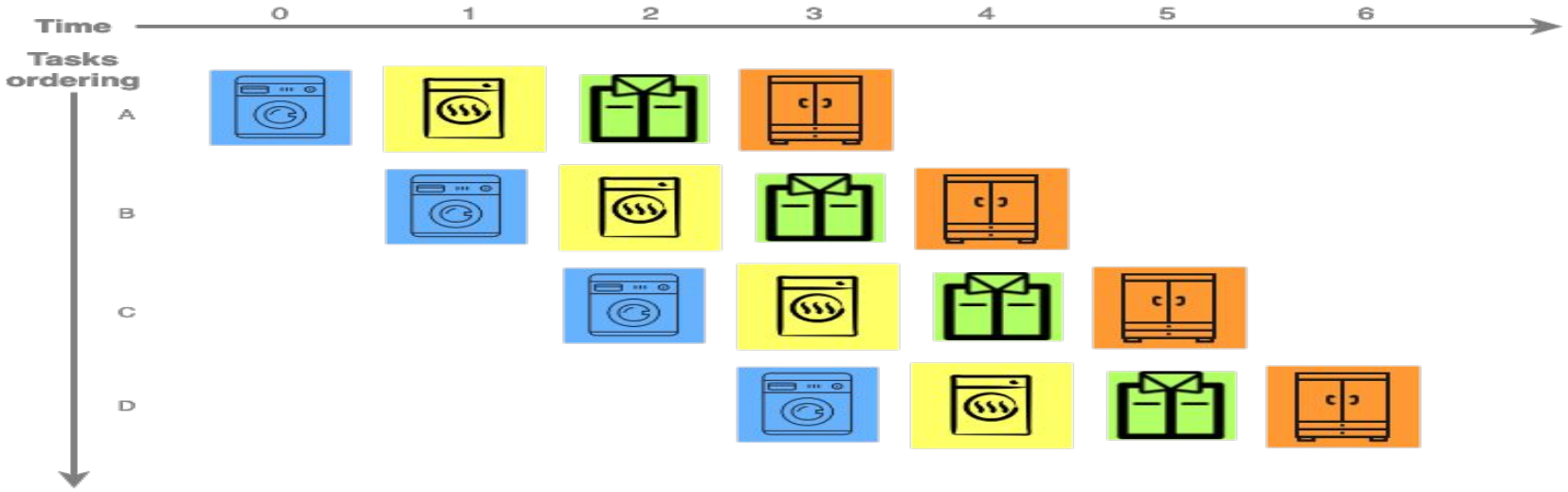
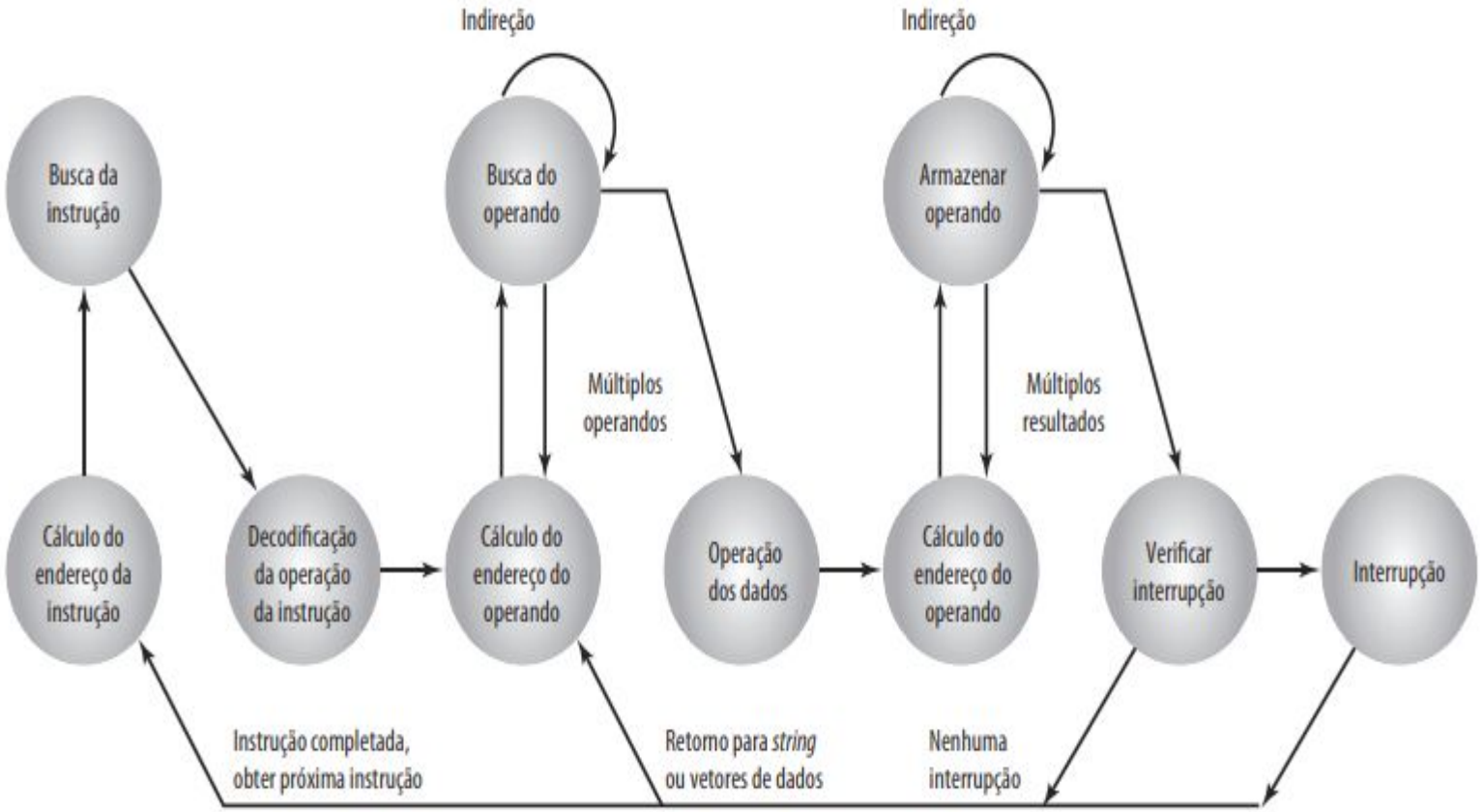
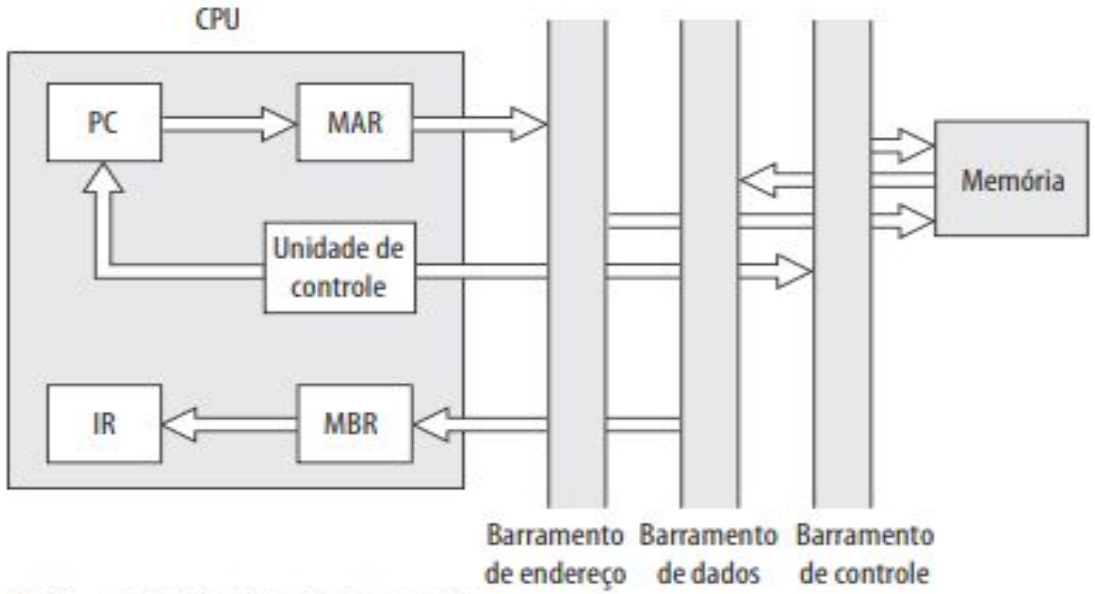


Diagrama de Estado do Ciclo de Instrução

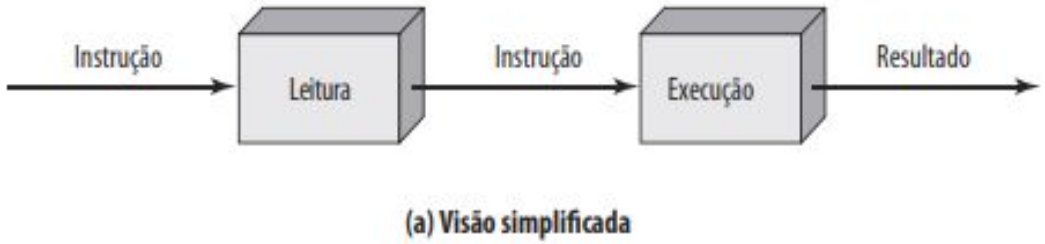


Fluxo de Dados do Ciclo de busca

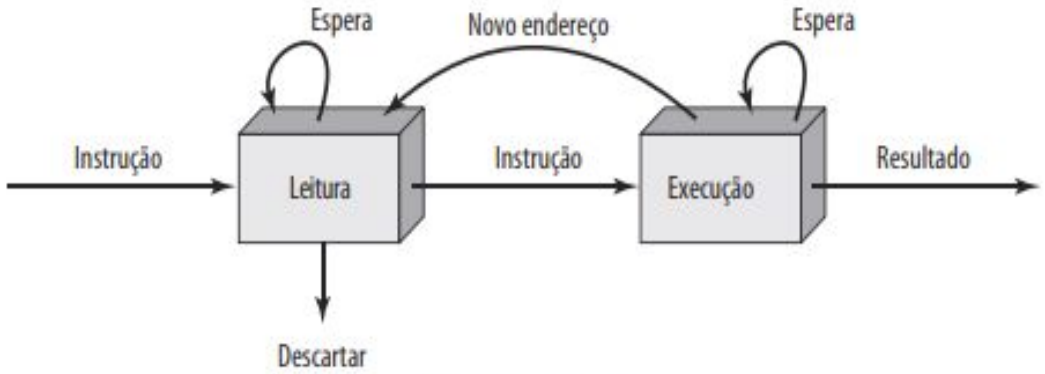


MBR = registrador de buffer de memória
MAR = registrador de endereço de memória
IR = registrador da instrução
PC = contador de programa

Pipeline de Instrucao de dois estágios



(a) Visão simplificada



(b) Visão expandida

Diagrama do tempo

Diagrama de tempo para operação do pipeline da instrução

Tempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instrução 1	FI	DI	CO	FO	EI	WO								
Instrução 2		FI	DI	CO	FO	EI	WO							
Instrução 3			FI	DI	CO	FO	EI	WO						
Instrução 4				FI	DI	CO	FO	EI	WO					
Instrução 5					FI	DI	CO	FO	EI	WO				
Instrução 6						FI	DI	CO	FO	EI	WO			
Instrução 7							FI	DI	CO	FO	EI	WO		
Instrução 8								FI	DI	CO	FO	EI	WO	
Instrução 9									FI	DI	CO	FO	EI	WO

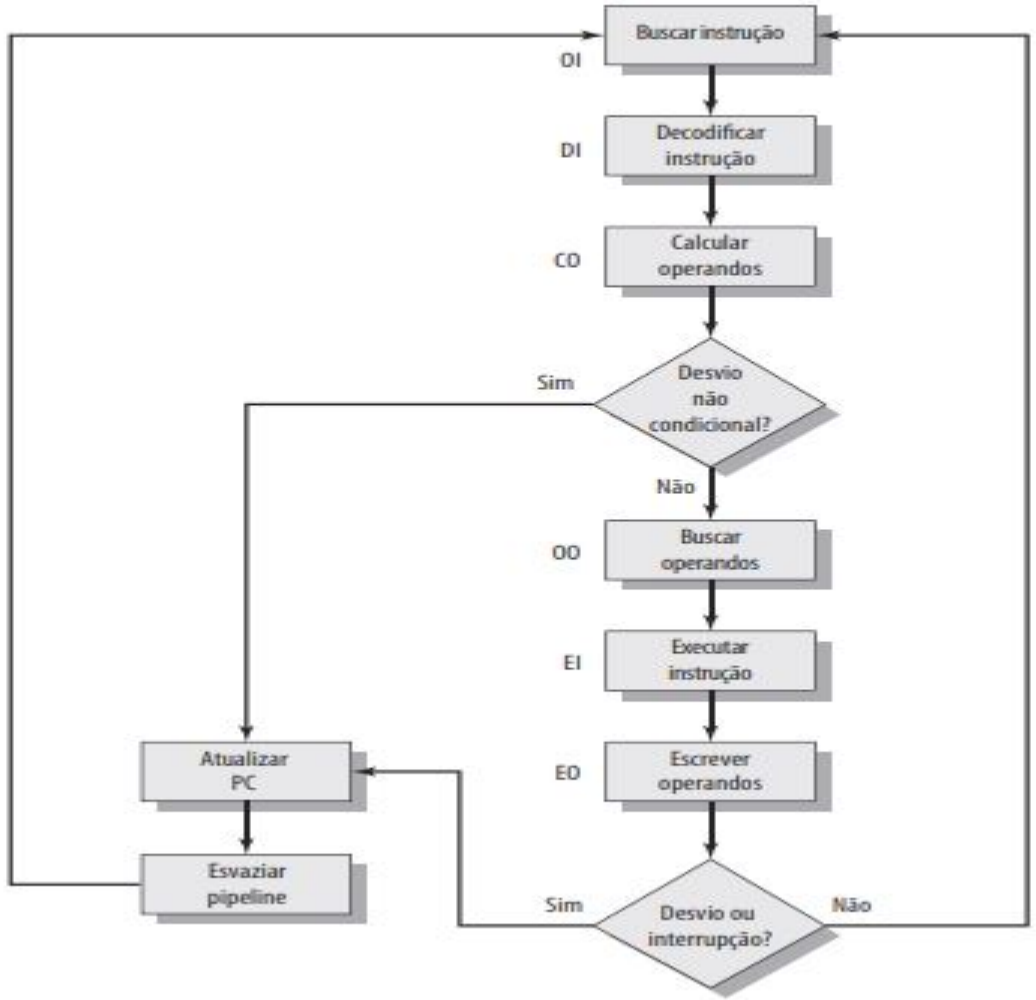
Diagrama do tempo com Desvio

O efeito de um desvio condicional na operação do pipeline da instrução

	Tempo →							Penalidade por desvio ←						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instrução 1	FI	DI	CO	FO	EI	WO								
Instrução 2		FI	DI	CO	FO	EI	WO							
Instrução 3			FI	DI	CO	FO	EI	WO						
Instrução 4				FI	DI	CO	FO							
Instrução 5					FI	DI	CO							
Instrução 6						FI	DI							
Instrução 7							FI							
Instrução 15								FI	DI	CO	FO	EI	WO	
Instrução 16									FI	DI	CO	FO	EI	WO

Pipeline de Instrução 6 estágios

Pipeline de instrução de uma CPU de seis estágios



Desempenho do Pipeline

. O tempo de ciclo t de uma instrução do
pipeline é o tempo necessário para que a instrução avance um estágio dentro do pipeline

$$\tau = max[\tau_i] + d = \tau_m + d \quad 1 \leq i \leq k ,$$

onde

τ_i = tempo de demora de resposta do circuito no estágio *i* do pipeline.

τ_m = tempo de demora máximo do estágio (demora do estágio que apresenta o maior tempo de demora de resposta).

k = número de estágios na instrução do pipeline.

d = tempo de resposta de um ciclo necessário para avançar sinais e dados de um estágio para o próximo.

Desempenho do Pipeline

Descrição alternativa de um pipeline

Tempo

	FI	DI	CO	FO	EI	WO
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	
6	I6	I5	I4	I3	I2	I1
7	I7	I6	I5	I4	I3	I2
8	I8	I7	I6	I5	I4	I3
9	I9	I8	I7	I6	I5	I4
10		I9	I8	I7	I6	I5
11			I9	I8	I7	I6
12				I9	I8	I7
13					I9	I8
14						I9

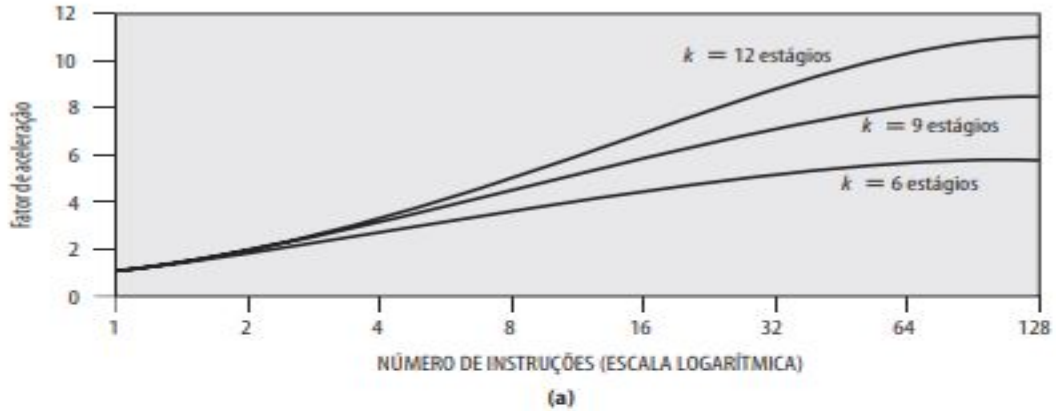
(a) Sem desvios

	FI	DI	CO	FO	EI	WO
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	
6	I6	I5	I4	I3	I2	I1
7	I7	I6	I5	I4	I3	I2
8	I15					I3
9	I16	I15				
10		I16	I15			
11			I16	I15		
12				I16	I15	
13					I16	I15
14						I16

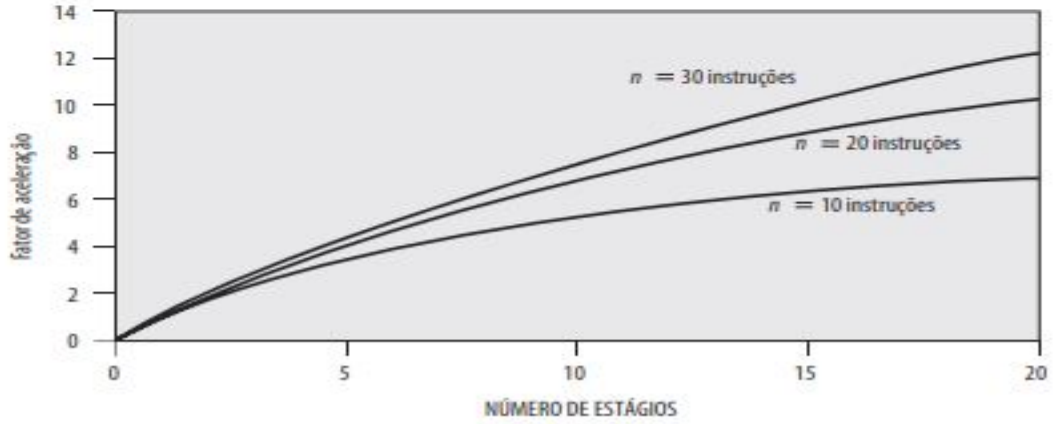
(b) Com desvios condicionais

Fatores de Aceleração

Fatores de aceleração com pipeline da instrução



(a)

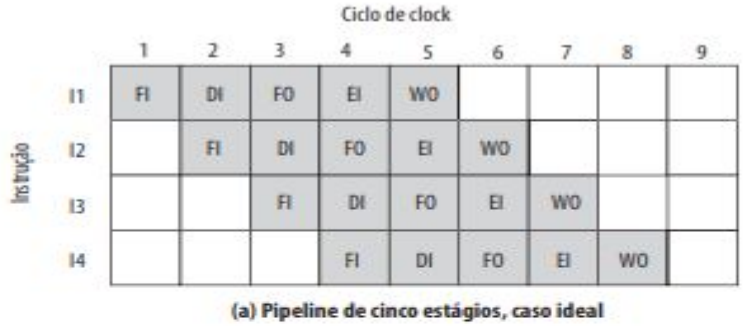


(b)

Hazards de Recursos

HAZARDS DE RECURSOS Um hazard de recursos ocorre quando duas (ou mais) instruções que já estão no pipeline precisam do mesmo recurso. O resultado é que as instruções precisam ser executadas em série em vez de em
paralelo para uma parte do pipeline. Um hazard de recursos às vezes é chamado de hazard estrutural

Exemplo de hazard de recursos



(a) Pipeline de cinco estágios, caso ideal

(b) Operando de origem de I1 na memória

Hazards de dados

HAZARDS DE DADOS

Um hazard de dados ocorre quando há um conflito no acesso de uma posição de operando. De um modo geral, podemos definir o hazard da seguinte forma: duas instruções em um programa estão para ser executadas na sequência e ambas acessam um determinado operando de memória ou registrador. No entanto,se as instruções são executadas em um pipeline, então é possível que o valor do operando seja atualizado de tal forma que produza um resultado diferente do que seria com uma execução estritamente sequencial. Em outras palavras, o programa produz um resultado incorreto por causa do uso do pipelining

Hazards de Dados

HAZARDS DE DADOS

Um hazard de dados ocorre quando há um conflito no acesso de uma posição de operando. De um modo geral, podemos definir o hazard da seguinte forma: duas instruções em um programa estão para ser executadas na sequência e ambas acessam um determinado operando de memória ou registrador. No entanto,se as instruções são executadas em um pipeline, então é possível que o valor do operando seja atualizado de tal forma que produza um resultado diferente do que seria com uma execução estritamente sequencial. Em outras palavras, o programa produz um resultado incorreto por causa do uso do pipelining

Existem três tipos de hazards de dados:

- Leitura após escrita ou dependência verdadeira: uma instrução modifica um registrador ou uma posição de memória e uma instrução subsequente lê os dados dessa posição de memória ou registrador. O hazard ocorre quando a operação de leitura acontece antes da escrita ter sido completada.

- escrita após leitura ou antidependência: uma instrução lê um registrador ou uma posição de memória e uma instrução subsequente escreve nessa posição. O hazard ocorre se a operação de escrita é completada antes da operação de leitura.

- escrita após escrita ou dependência de saída: duas instruções escrevem na mesma posição. O perigo ocorre se as operações de escrita acontecerem na sequência inversa da esperada.

Hazards de Controle

HAZARDS DE CONTROLE

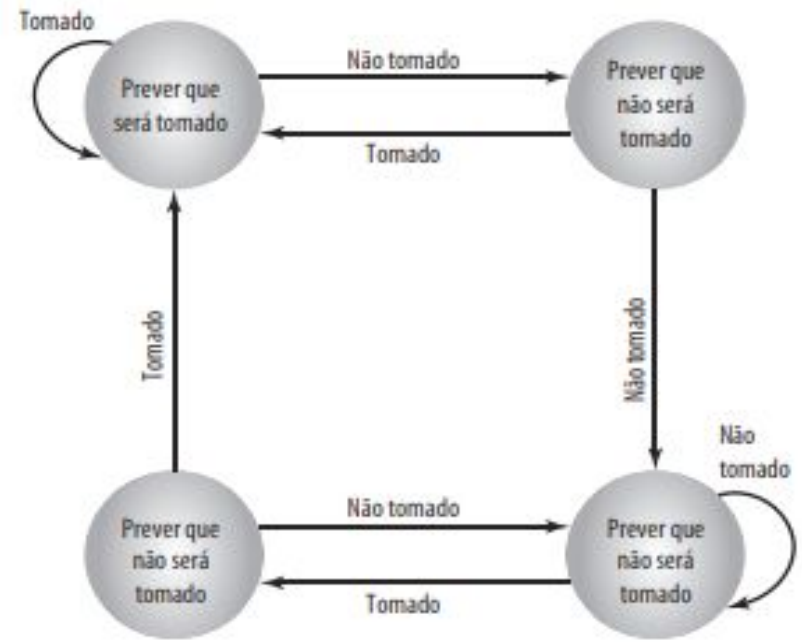
Um hazard de controle, também conhecido como hazard de desvio, acontece quando o pipeline toma decisão errada ao prever um desvio e assim acaba trazendo instruções dentro do pipeline que precisam ser descartadas logo em seguida.

Uma série de abordagens foram implementadas para lidar com desvios condicionais:

- Múltiplos fluxos.
- Busca antecipada do alvo do desvio.
- Buffer de laço de repetição.
- Previsão de desvio (branch prediction).
- Desvio atrasado.

Diagrama de Estados de Previsao

Diagrama de estados de previsão de desvio



Pipeline 80486

Um exemplo instrutivo de um pipeline de instruções é o de Intel 80486. Ele implementa um pipeline de cinco Estágios:

 Leitura: instruções são obtidas a partir da cache ou da memória externa e são colocadas em um de dois buffers de busca antecipada de 16 bits. O objetivo do estágio de leitura é preencher os buffers de busca antecipada com dados novos assim que os dados antigos tenham sido consumidos pelo decodificador da instrução. Como as instruções têm tamanhos variáveis (de 1 a 11 bytes sem contar prefixos), o estado do estágio da busca antecipada em relação a outros estágios varia de instrução para instrução. Em média, em torno de cinco instruções são obtidas com cada carga de 16 bytes (CRAWFORD, 1990n). O estágio de leitura opera independentemente de outros estágios para manter os buffers de busca antecipada cheios.

 Estágio de decodificação 1: toda a informação de opcode e modo de endereçamento é decodifi cada no estágio D1. A informação requerida, assim como a informação sobre o tamanho da instrução, é incluída em, no máximo, nos 3 primeiros bytes da instrução. Por isso, os 3 bytes são passados para o estágio D1 a partir dos buff ers de busca antecipada. O decodificador D1 pode então direcionar o estágio D2 para pegar o restante da instrução (dados imediatos e de deslocamento), a qual não está envolvida na decodificação em D1.

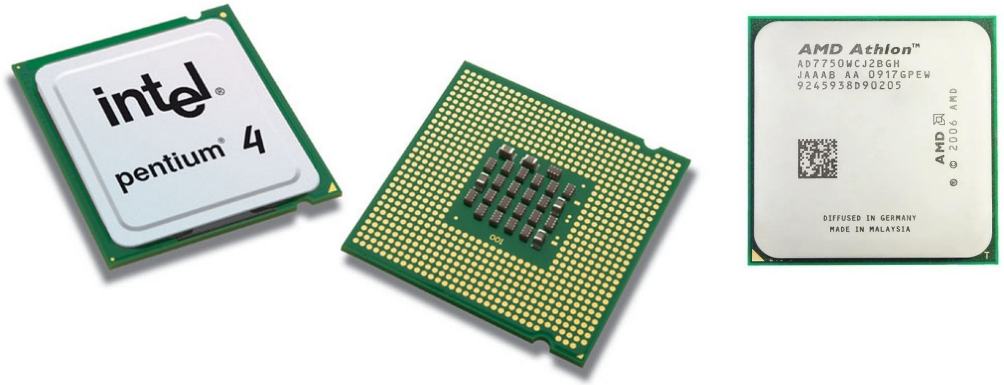
 Estágio de decodificação 2: o estágio D2 traduz cada opcode em sinais de controle para ALU. Ele também controla o cálculo de modos de endereçamento mais complexos.

 Execução: este estágio inclui operações de ALU, acesso a cache e atualização de registradores.

 Escrita: este estágio, se necessário, atualiza registradores e fl ags de estado modificados durante o processo da execução anterior. Se a instrução corrente atualiza a memória, o valor computado é enviado para a cache e a interface de barramento escreve nos buffers ao mesmo tempo

Família de Processadores x86

A organização x86 evoluiu consideravelmente ao longo dos anos. Analisaremos alguns dos detalhes das mais recentes organizações dos processadores, focando em elementos comuns em processadores únicos. Abaixo temos imagens do processador Pentium 4 (Intel) e Athlon XP (AMD).



Organização dos registradores

A organização dos registradores inclui os seguintes tipos de registradores:

(a) Unidade de inteiros no modo 32-bits

Tipo	Número	Extensão (bits)	Propósito
Geral	8	32	Registradores de uso geral
Segmento	6	16	Contém seletores de segmento
EFLAGS	1	32	Bits de estado e controle
Ponteiro de instrução	1	32	Ponteiro de instrução

(b) Unidade de inteiros no modo 64-bits

Tipo	Número	Extensão (bits)	Propósito
Geral	16	32	Registradores de uso geral
Segmento	6	16	Contém seletores de segmento
RFLAGS	1	64	Bits de estado e controle
Ponteiro de instrução	1	64	Ponteiro de instrução

Unidade de Ponto Flutuante

Existem também registradores dedicados especialmente para unidade de ponto flutuante:

(c) Unidade de ponto flutuante

Tipo	Número	Extensão (bits)	Propósito
Numérico	8	80	Armazena números de ponto flutuante
Controle	1	16	Bits de controle
Estado	1	16	Bits de estado
Palavra de rótulo	1	16	Especifica o conteúdo de registradores numéricos
Ponteiro de instrução	1	48	Aponta para instrução interrompida pela exceção
Ponteiro de dados	1	48	Aponta para operando quando interrompido pela exceção

REGISTRADOR EFLAGS

O registrador EFLAGS indica a condição do processador e ajuda a controlar suas operações.

Registrador EFLAGS do x86.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

X ID = flag de identificação
X VIP = interrupção virtual pendente
X VIF = flag de interrupção virtual
X AC = verificação de alinhamento
X VM = modo virtual do 8086
X RF = flag de reinício
X NT = flag de tarefa aninhada
X IOPL = nível de privilégio de E/S
S OF = flag de *overflow*

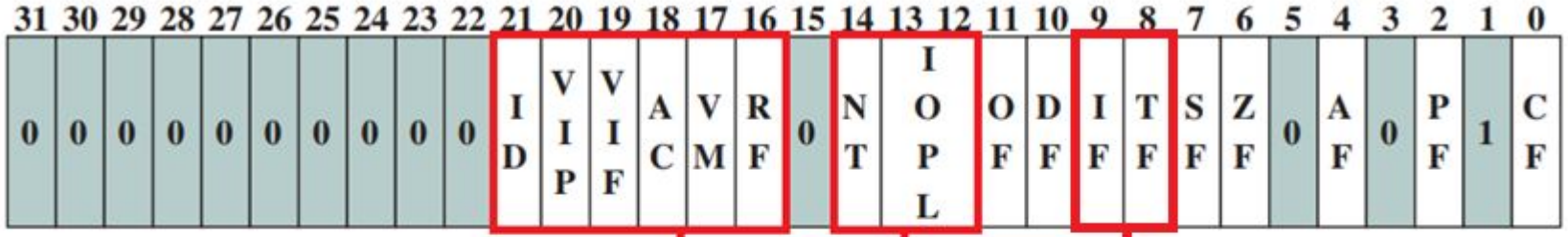
C DF = flag direcional
X IF = flag para habilitar interrupção
X TF = flag de *trap*
S SF = flag de sinal
S ZF = flag zero
S AF = flag de *carry* auxiliar
S PF = flag de paridade
S CF = flag de *carry*

S indica flag de estado.
C indica uma flag de controle.
X indica uma flag de sistema.
Os bits sombreados são reservados.

REGISTRADOR EFLAGS

O registrador EFLAGS indica a condição do processador e ajuda a controlar suas operações.

Registrador EFLAGS do x86.



X ID = flag de identificação
X VIP = interrupção virtual pendente
X VIF = flag de interrupção virtual
X AC = verificação de alinhamento
X VM = modo virtual do 8086
X RF = flag de reinício
X NT = flag de tarefa aninhada
X IOPL = nível de privilégio de E/S
S OF = flag de *overflow*

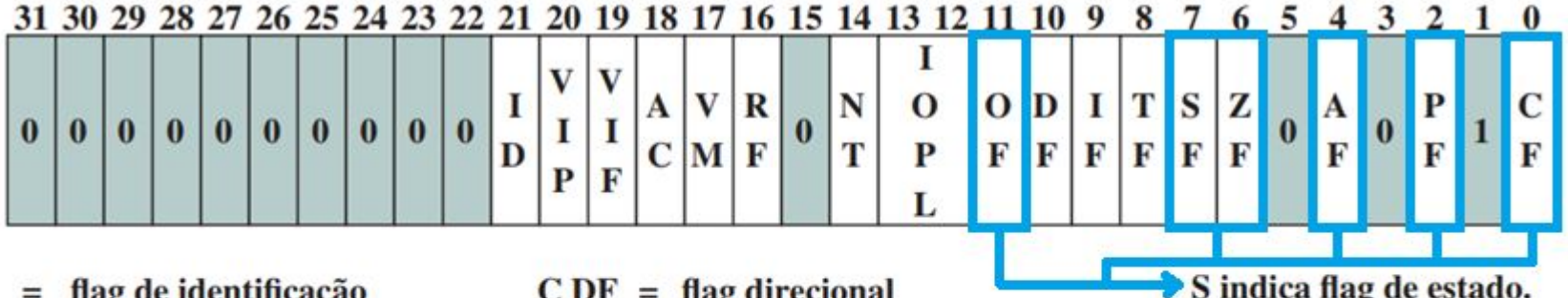
C DF = flag direcional
X IF = flag para habilitar interrupção
X TF = flag de *trap*
S SF = flag de sinal
S ZF = flag zero
S AF = flag de *carry* auxiliar
S PF = flag de paridade
S CF = flag de *carry*

S indica flag de estado.
C indica uma flag de controle.
X indica uma flag de sistema.
Os bits sombreados são reservados.

REGISTRADOR EFLAGS

O registrador EFLAGS indica a condição do processador e ajuda a controlar suas operações.

Registrador EFLAGS do x86.



X ID = flag de identificação
X VIP = interrupção virtual pendente
X VIF = flag de interrupção virtual
X AC = verificação de alinhamento
X VM = modo virtual do 8086
X RF = flag de reinício
X NT = flag de tarefa aninhada
X IOPL = nível de privilégio de E/S
S OF = flag de *overflow*

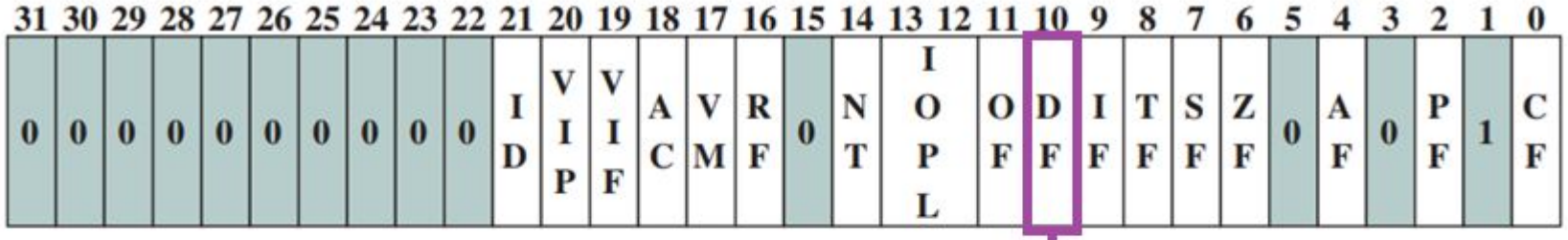
C DF = flag direcional
X IF = flag para habilitar interrupção
X TF = flag de *trap*
S SF = flag de sinal
S ZF = flag zero
S AF = flag de *carry* auxiliar
S PF = flag de paridade
S CF = flag de *carry*

S indica flag de estado.
C indica uma flag de controle.
X indica uma flag de sistema.
Os bits sombreados são reservados.

REGISTRADOR EFLAGS

O registrador EFLAGS indica a condição do processador e ajuda a controlar suas operações.

Registrador EFLAGS do x86.



X ID = flag de identificação
X VIP = interrupção virtual pendente
X VIF = flag de interrupção virtual
X AC = verificação de alinhamento
X VM = modo virtual do 8086
X RF = flag de reinício
X NT = flag de tarefa aninhada
X IOPL = nível de privilégio de E/S
S OF = flag de *overflow*

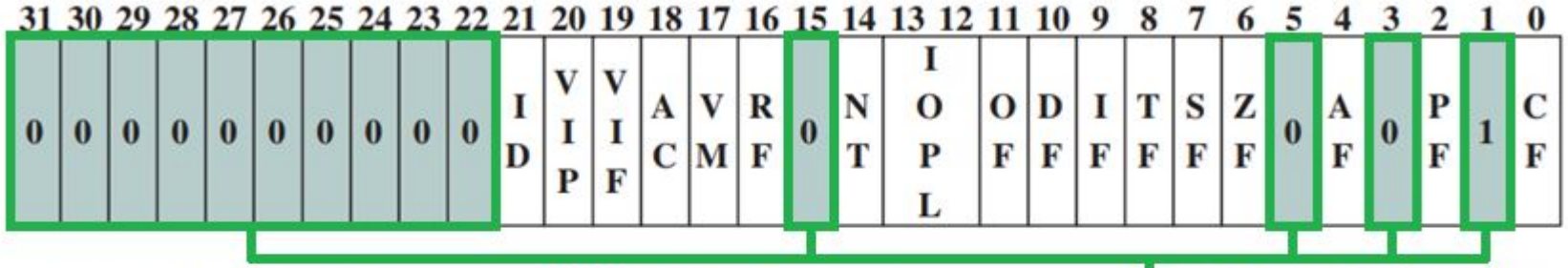
C DF = flag direcional
X IF = flag para habilitar interrupção
X TF = flag de *trap*
S SF = flag de sinal
S ZF = flag zero
S AF = flag de *carry* auxiliar
S PF = flag de paridade
S CF = flag de *carry*

S indica flag de estado.
C indica uma flag de controle.
X indica uma flag de sistema.
Os bits sombreados são reservados.

REGISTRADOR EFLAGS

O registrador EFLAGS indica a condição do processador e ajuda a controlar suas operações.

Registrador EFLAGS do x86.



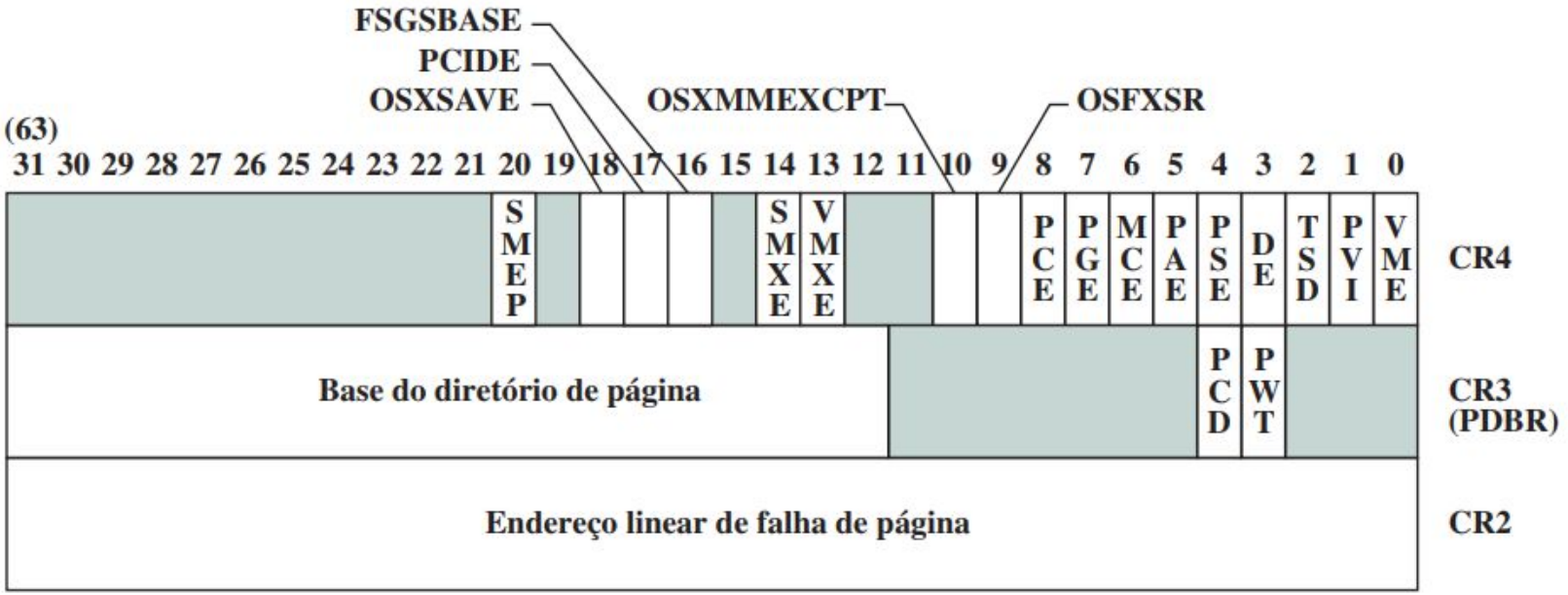
X ID = flag de identificação
X VIP = interrupção virtual pendente
X VIF = flag de interrupção virtual
X AC = verificação de alinhamento
X VM = modo virtual do 8086
X RF = flag de reinício
X NT = flag de tarefa aninhada
X IOPL = nível de privilégio de E/S
S OF = flag de *overflow*

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

S indica flag de estado.
C indica uma flag de controle.
X indica uma flag de sistema.
Os bits sombreados são reservados.

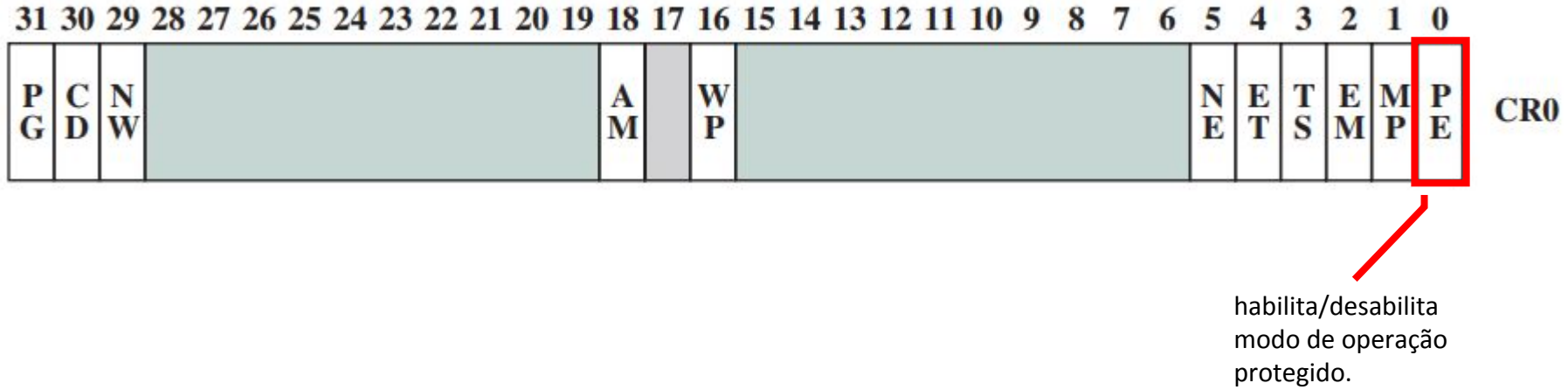
REGISTRADORES DE CONTROLE

O x86 usa quatro registradores de controle (o registrador CR1 não é usado) para controlar os vários aspectos da operação do processador. Todos os registradores, exceto CR0, têm o tamanho de 32 ou 64 bits, de acordo com a implementação: se ela suporta arquitetura x86 de 64 bits ou não.



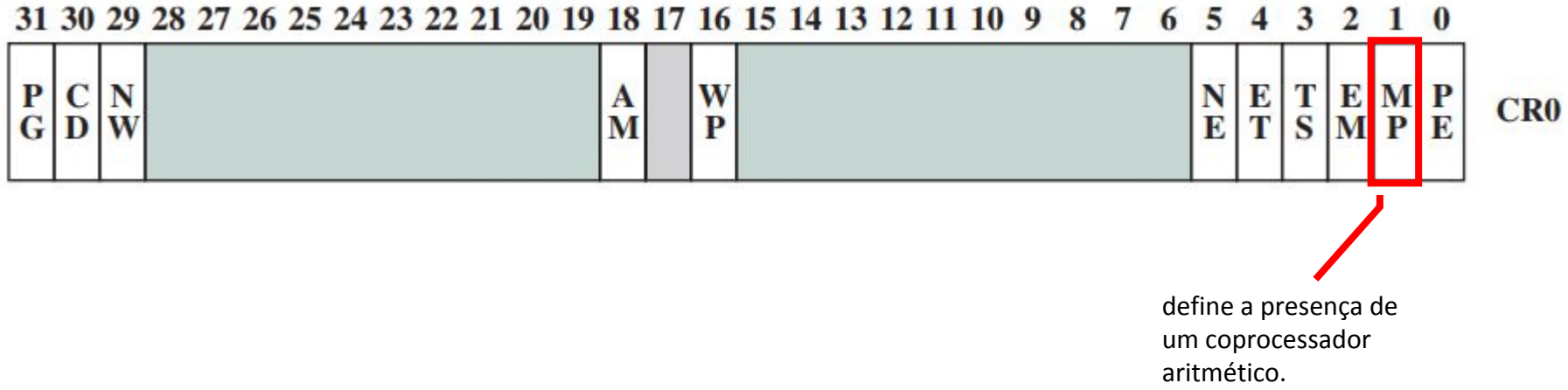
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



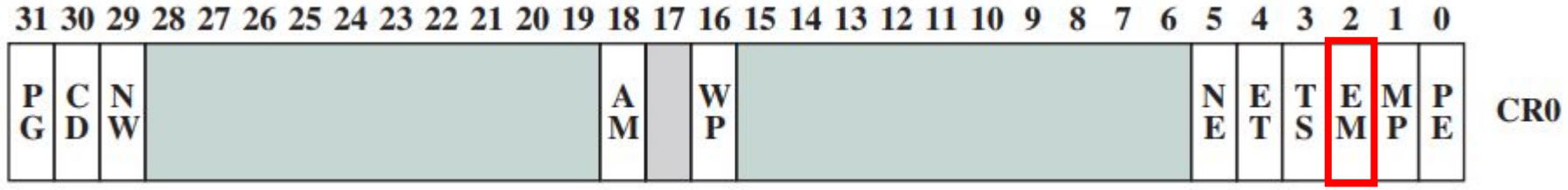
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



REGISTRADORES DE CONTROLE

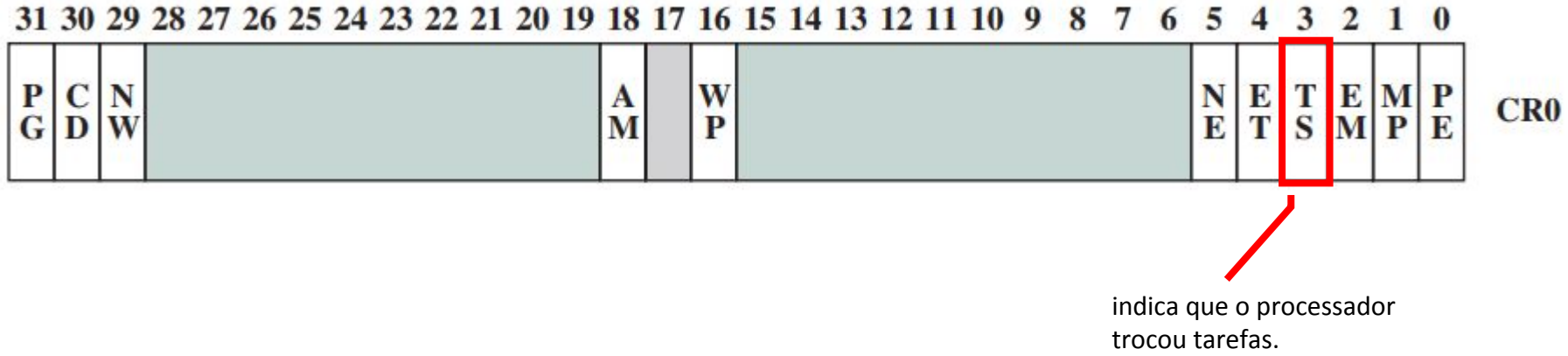
O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



quando o processador não possui unidade de ponto flutuante, causa interrupção quando uma tentativa é feita para executar instruções de ponto flutuante.

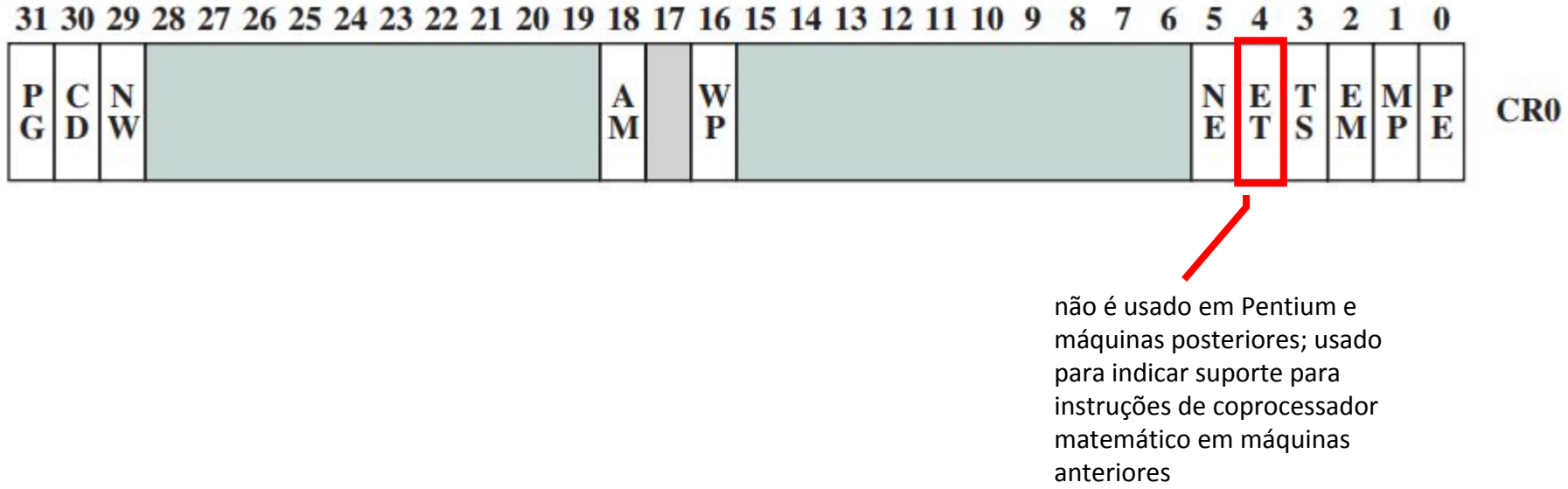
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



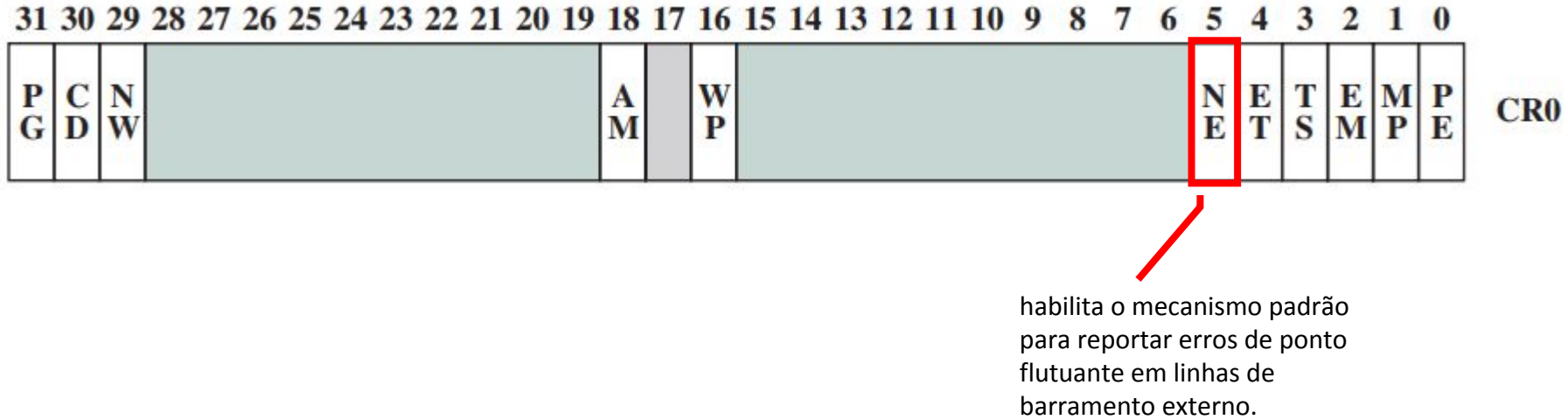
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



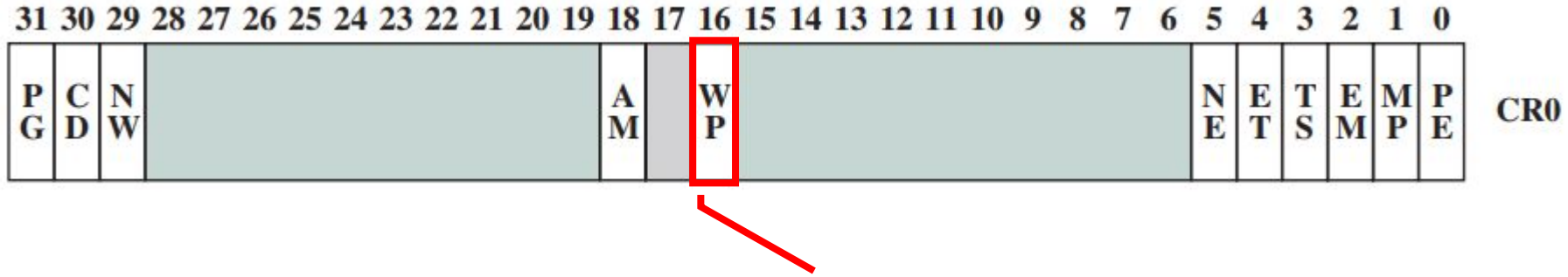
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



REGISTRADORES DE CONTROLE

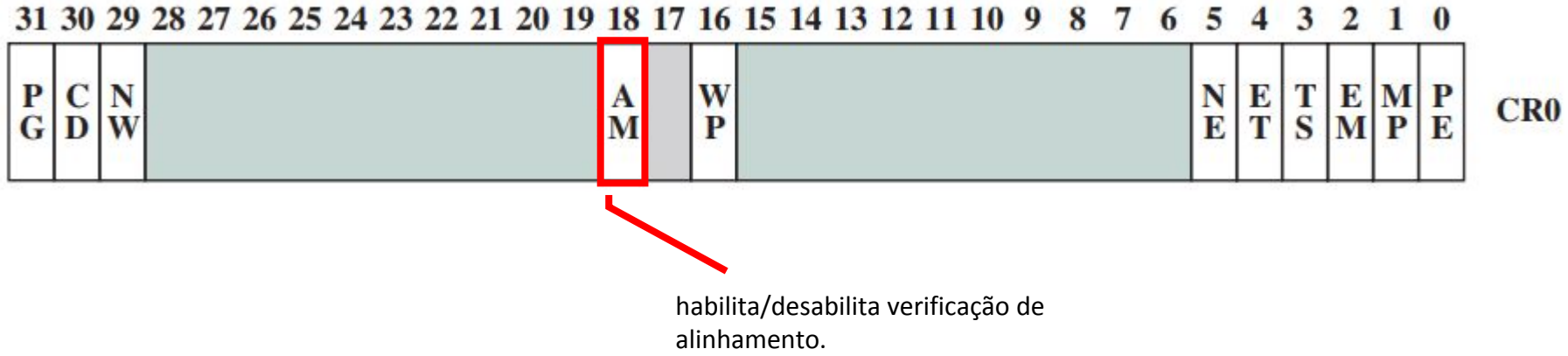
O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



quando este bit é igual a zero, páginas de usuário com permissão de somente leitura podem ser escritas por um processo supervisor.

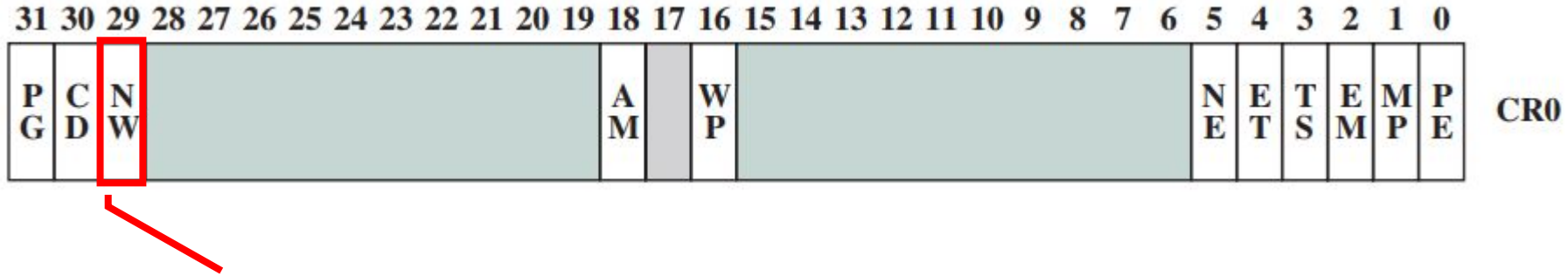
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



REGISTRADORES DE CONTROLE

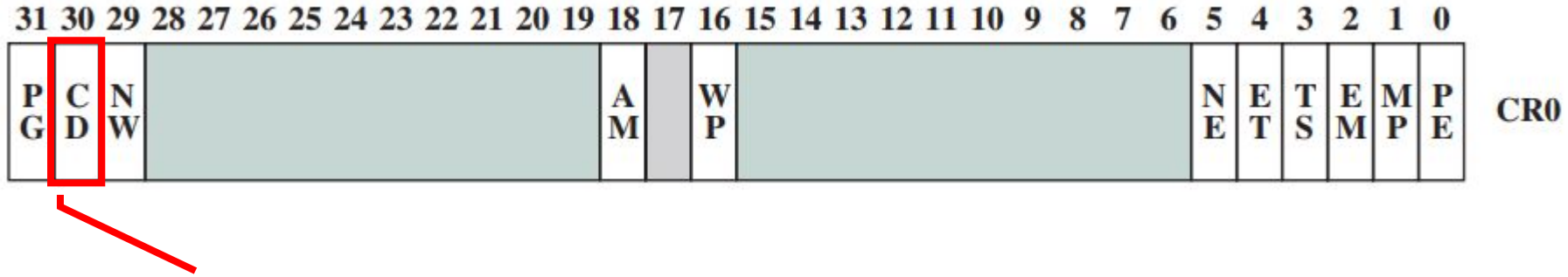
O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



seleciona o modo de operação de cache de dados. Quando esse bit é um, a cache de dados é inibida a partir das operações de cache *write through.*

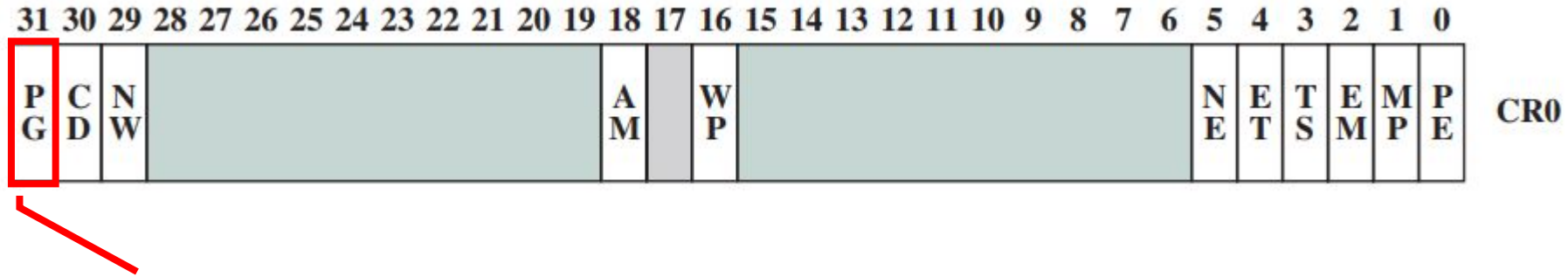
REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



REGISTRADORES DE CONTROLE

O registrador CR0 contém flags de controle do sistema que controlam o modo ou indicam estados que se aplicam normalmente ao processador em vez da execução de uma determinada tarefa. Os flags são os seguintes:



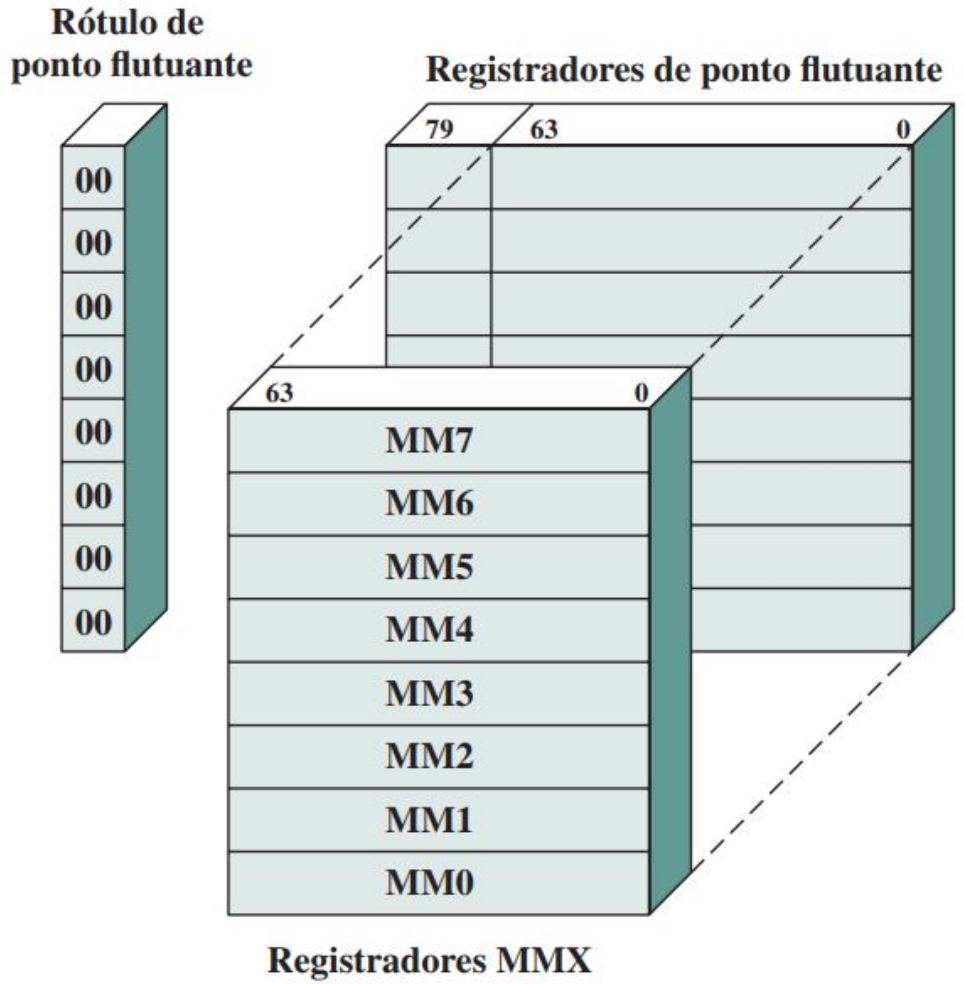
habilita/desabilita paginação.

REGISTRADORES MMX

As instruções MMX fazem uso de campos de endereço de registrador de 3 bits para que oito registradores MMX sejam suportados. Na verdade, o processador não inclui registradores MMX específicos. Em vez disso, o processador usa uma técnica de mapeamento. Os registradores de ponto flutuante existentes são usados para armazenar valores MMX. Especificamente, 64 bits de baixa ordem de cada registrador de ponto flutuante são usados para formar oito registradores MMX. Dessa forma, a antiga arquitetura x86 de 32 bits é facilmente estendida para suportar a capacidade MMX.

REGISTRADORES MMX

Mapeamento de registradores MMX para registradores de ponto flutuante.



Processamento de interrupções

O processamento de interrupções dentro de um processador é uma facilidade oferecida para suportar o sistema operacional. Isso permite que um programa aplicativo seja suspenso para que uma variedade de condições de interrupções possa ser atendida e depois seja reiniciado.

INTERRUPÇÕES E EXCEÇÕES: Duas classes de eventos fazem com que o x86 suspenda o fluxo de execução da instrução corrente e responda ao evento: interrupções e exceções. Em ambos os casos, o processador salva o contexto do processo atual e transfere para uma rotina predefinida para atender a condição.

Processamento de interrupções

Existem duas origens das interrupções e duas origens das exceções:

1. Interrupções

Interrupções mascaráveis: recebidas no pino INTR do processador. O processador não reconhece uma interrupção mascarável se o flag de habilitar interrupção (IF) não estiver definido.

Interrupções não mascaráveis: recebidas no pino NMI do processador. O reconhecimento de tais interrupções não pode ser evitado.

2. Exceções

Exceções detectadas pelo processador: resultam quando o processador encontra um erro enquanto tenta executar uma instrução.

Exceções programadas: essas são as instruções que geram uma exceção (por exemplo, INTO, INT3, INT e BOUND)

Tabela de Vetores e Interrupções

O processamento de interrupção em x86 usa uma tabela de vetores de interrupções. Cada tipo de interrupção possui um número vinculado e esse número é usado para indexar a tabela de vetores de interrupções. Essa tabela contém 256 vetores de interrupção de 32 bits, que representa o endereço (segmento e offset) da rotina para atender a interrupção para esse determinado número de interrupção.

mostra a atribuição de números na tabela de vetores de interrupções; os campos sombreados representam interrupções, enquanto os não sombreados são exceções. A interrupção NMI de hardware é do tipo 2. Às interrupções INTR de hardware são atribuídos os números do intervalo de 32 até 255; quando uma interrupção INTR é gerada, ela precisa ser acompanhada dentro do barramento pelo número do vetor de interrupção para essa interrupção. Os números de vetores restantes são usados para exceções.

Tabela de Vetores e Interrupções

Número do vetor	Descrição
0	Erro de divisão; estouro de *overflow* ou divisão por zero.
1	Exceção de depuração; inclui várias falhas e *traps* relacionados à depuração.
2	Interrupção do pino NMI; sinal no pino NMI.
3	*Breakpoint*; causado pela instrução INT 3 que é uma instrução de 1 byte, útil para a depuração.
4	*Overflow* detectado em INTO; ocorre quando o processador executa INTO com o flag OF igual a um.
5	Limite em BOUND excedido; instrução BOUND compara um registrador com limites armazenados na memória e gera uma interrupção se o conteúdo do registrador está fora dos limites.
6	Opcode indefinido.
7	Dispositivo indisponível; tentativa de uso da instrução ESC ou WAIT falha por causa da demora do dispositivo externo.
8	Falha dupla; duas interrupções ocorrem durante a mesma instrução e não podem ser tratadas em série.
9	Reservado.
10	Segmento de estado de tarefa inválido; segmento que descreve a tarefa requerida não é inicializado ou não é válido.
11	Segmento ausente; segmento requerido não está presente.
12	Falha de pilha; limite do segmento da pilha excedido ou segmento da pilha ausente.
13	Proteção geral; violação da proteção que não causa outra exceção (por exemplo, escrever no segmento que é somente para leitura).
14	Falha de página.
15	Reservado.
16	Erro de ponto flutuante; gerado por uma instrução aritmética de ponto flutuante.
17	Verificação de alinhamento; acesso a uma palavra armazenada em um endereço de byte ímpar ou uma palavra dupla armazenada em um endereço não múltiplo de 4.
18	Verificação de máquina; específico para cada modelo.
19–31	Reservado.
32–255	Vetores de interrupções de usuário; fornecidos quando sinal INTR é ativado.

Sem sombra: exceções.
Com sombra: interrupções.

Tabela de Vetores e Interrupções

Se mais do que uma exceção ou interrupção está pendente, o processador as atende de maneira previsível.

A posição de números do vetor dentro da tabela não reflete a prioridade. Em vez disso, a prioridade entre exceções e interrupções é organizada em cinco classes. Em ordem descendente de prioridade aqui estão:

}} **Classe 1:** paradas (traps) na instrução anterior (vetor número 1).

}} **Classe 2:** interrupções externas (2, 32–255).

}} **Classe 3:** falhas na busca da próxima instrução (3, 14).

}} **Classe 4:** falhas na decodificação da próxima instrução (6, 7).

}} **Classe 5:** falhas na execução de uma instrução (0, 4, 5, 8, 10–14, 16, 17).

Tratamento de Instruções

Assim como acontece com a transferência da execução usando uma instrução CALL, uma transferência para uma rotina de tratamento de interrupção usa a pilha do sistema para armazenar o estado do processador. Quando uma interrupção ocorre e é reconhecida pelo processador, uma sequência de eventos acontece:

1. Se a transferência envolve uma mudança do nível de privilégio.
2. O valor atual do registrador EFLAGS é colocado na pilha.
3. Flags de interrupção (IF) e trap (TF) são definidos com valor zero.
4. O ponteiro de segmento de código corrente e o ponteiro da instrução corrente são colocados na pilha.
5. Se a interrupção é acompanhada por um código de erro, então o código de erro é colocado na pilha.
6. O conteúdo do vetor de interrupção é obtido e carregado nos registradores CS e IP ou EIP.

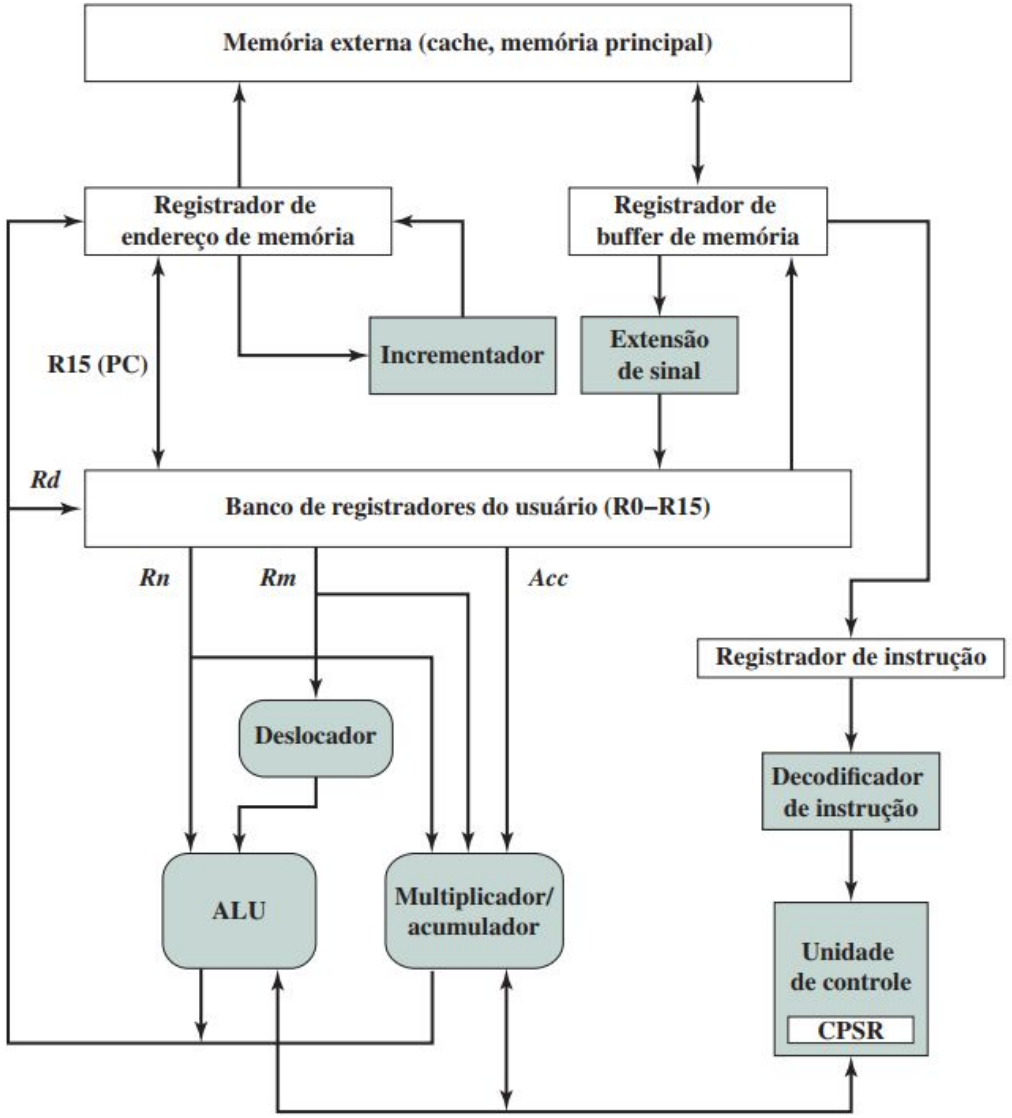
PROCESSADOR ARM

O ARM é, em primeiro lugar, um sistema RISC com as seguintes características principais:

- Um conjunto moderado de registradores uniformes, mais do que são encontrados em alguns sistemas CISC, porém menos do que encontrados em muitos sistemas RISC.
- Modelo load/store (carregar/armazenar) de processamento de dados, no qual as operações são executadas apenas com os operandos nos registradores e não diretamente na memória. Todos os dados precisam ser carregados em registradores antes que uma operação possa ser efetuada; o resultado então pode ser usado para o processamento posterior ou armazenado em memória.
- Uma instrução uniforme de tamanho fixo de 32 bits para o conjunto padrão e 16 bits para o conjunto de instruções Thumb.
- Para tornar cada instrução de processamento de dados mais flexível, um deslocamento ou uma rotação pode pré-processar um dos registradores de origem. Para suportar esse recurso eficientemente, a unidade aritmética lógica (ALU) e unidades de deslocamento são separadas.
- Um número pequeno de modos de endereçamento com todos os endereços de load/store sendo determinados a partir dos registradores e campos da instrução. Endereçamento indireto ou indexado envolvendo valores na memória não é usado.
- Modos de endereçamento com autoincremento e autodecremento são usados para melhorar a operação de loops dos programas.
- Execução condicional das instruções minimiza a necessidade das instruções de desvios condicionais, melhorando assim a eficiência do pipeline, porque o esvaziamento do pipeline é reduzido .

Organização do processador

Organização ARM simplificada.



Modos do Processador

Os modos de exceção são os seguintes:

Modo supervisor: em geral é o modo em que executa o SO. Ele é ativado quando o processador encontra uma instrução de interrupção de software. Interrupções de software são um jeito padrão para chamar os serviços do sistema operacional no ARM.

Modo aborto de acesso: ativado como resposta a falhas de memória.

Modo indefinido: ativado quando o processador tenta executar uma instrução que não é suportada nem pelo core principal nem por um dos coprocessadores.

Modo interrupção rápida: ativado sempre que o processador recebe um sinal de interrupção a partir de uma fonte designada de interrupção rápida. Uma interrupção rápida não pode ser interrompida, porém uma interrupção rápida pode interromper uma interrupção normal.

Modo interrupção: ativado sempre que o processador recebe um sinal de interrupção a partir de qualquer outra origem de interrupção (diferente da interrupção rápida). Uma interrupção apenas pode ser interrompida por uma interrupção rápida

Organização dos Registradores

Organização dos registradores do ARM.

Modos						
	Modos privilegiados					
		Modos de exceção				
Usuário	**Sistema**	**Supervisor**	**Aborto de acesso**	**Indefinido**	**Interrupção**	**Interrupção rápida**
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13(SP)	R13(SP)	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14(LR)	R14(LR)	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Sombreado indica que o registrador normal usado pelo modo usuário ou sistema foi substituído por um registrador específico para modo de exceção.

SP = ponteiro de pilha **CPSR = registrador de estado do programa corrente**

LR = registrador de ligação **SPSR = registrador de estado do programa salvo**

PC = contador de programa

Organização dos Registradores

Registradores de R0 a R7, registrador R15 e registrador de estado de programa corrente (CPSR) são visíveis e compartilhados por todos os modos.

Registradores de R8 até R12 são compartilhados por todos os modos exceto interrupção rápida, que possui seus próprios registradores dedicados de R8_fiq até R12_fiq.

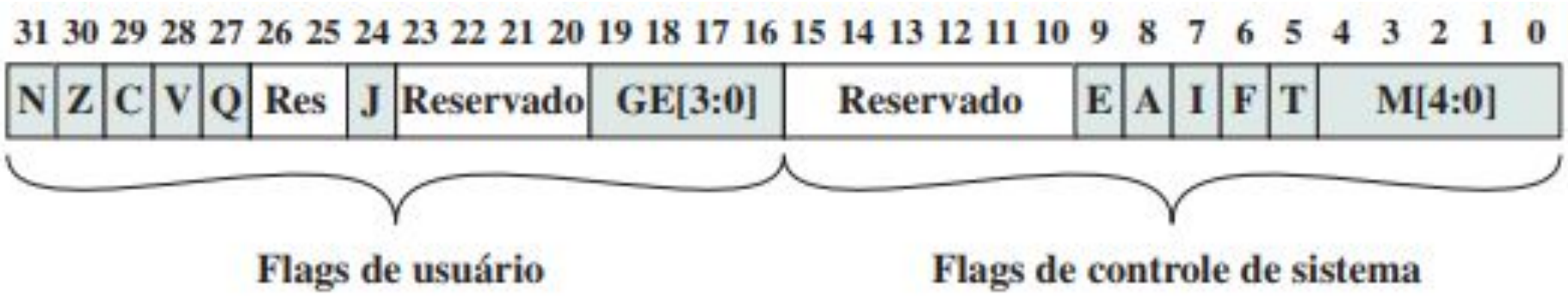
Todos os modos de exceção têm suas próprias versões de registradores R13 e R14.

Todos os modos de exceção têm um registrador próprio dedicado ao estado do programa salvo (SPSR — do inglês, Saved Program Status Register).

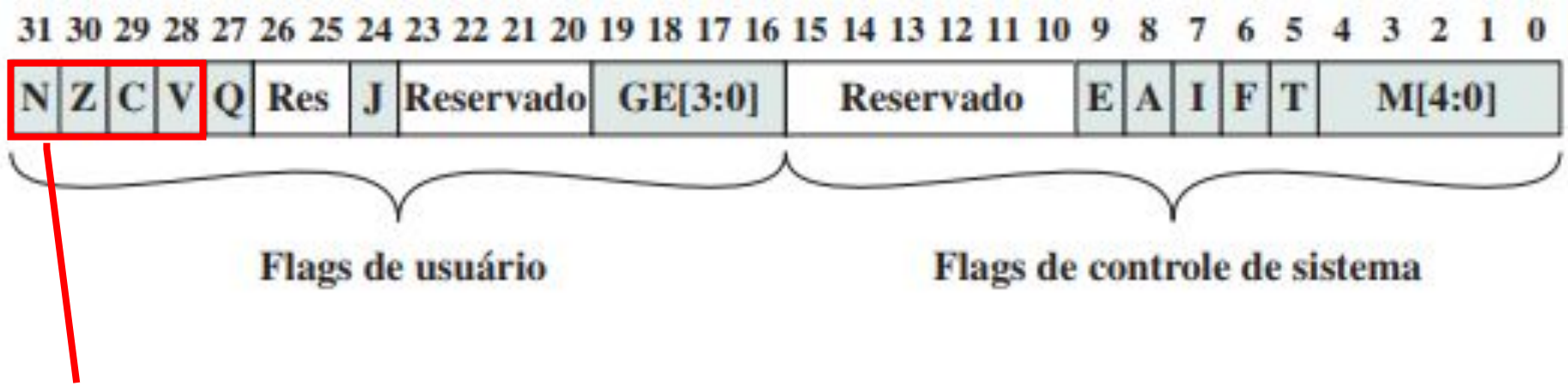
Organização dos Registradores

REGISTRADORES DE ESTADOS DO PROGRAMA: O CPSR é acessível em todos os modos do processador. Cada modo de exceção também possui um SPSR dedicado que é usado para preservar o valor de CPSR quando uma exceção associada acontece.

Os 16 bits mais significativos do CPSR contêm flags de usuário visíveis no modo usuário e que podem ser usados para afetar a operação de um programa.

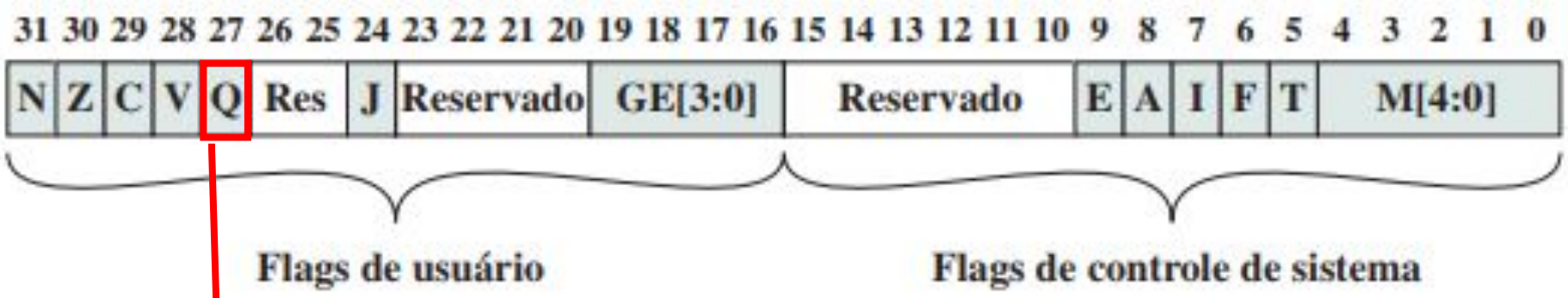


Organização dos Registradores



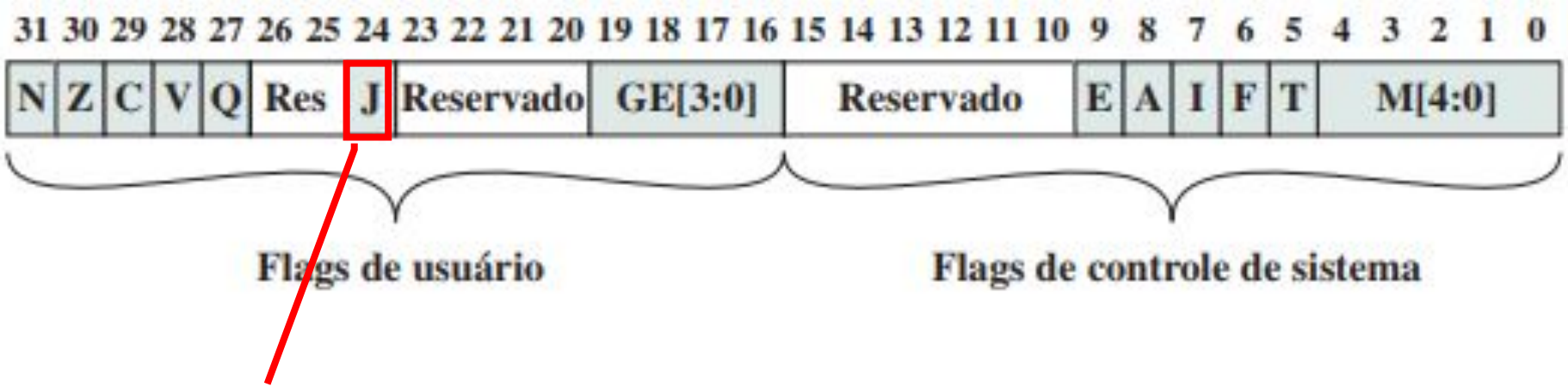
Flags de estaddo

Organização dos Registradores



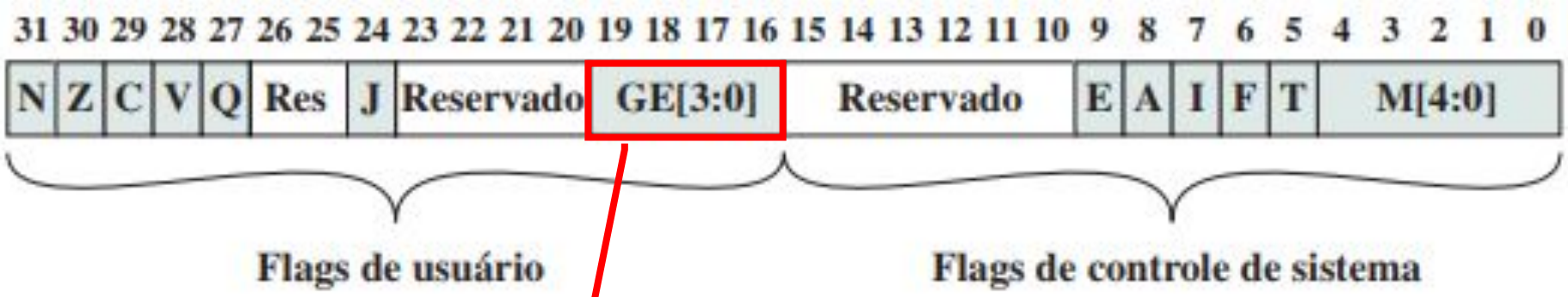
usado para indicar se um overflow e/ou saturação ocorreu em alguma instrução orientada a SIMD

Organização dos Registradores



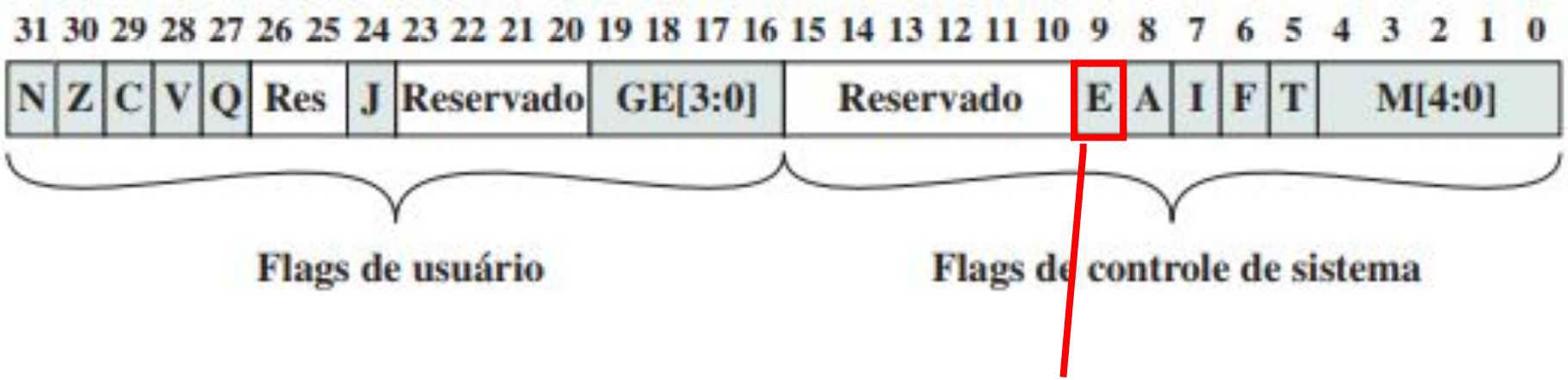
indica o uso de instruções especiais de 8 bits

Organização dos Registradores



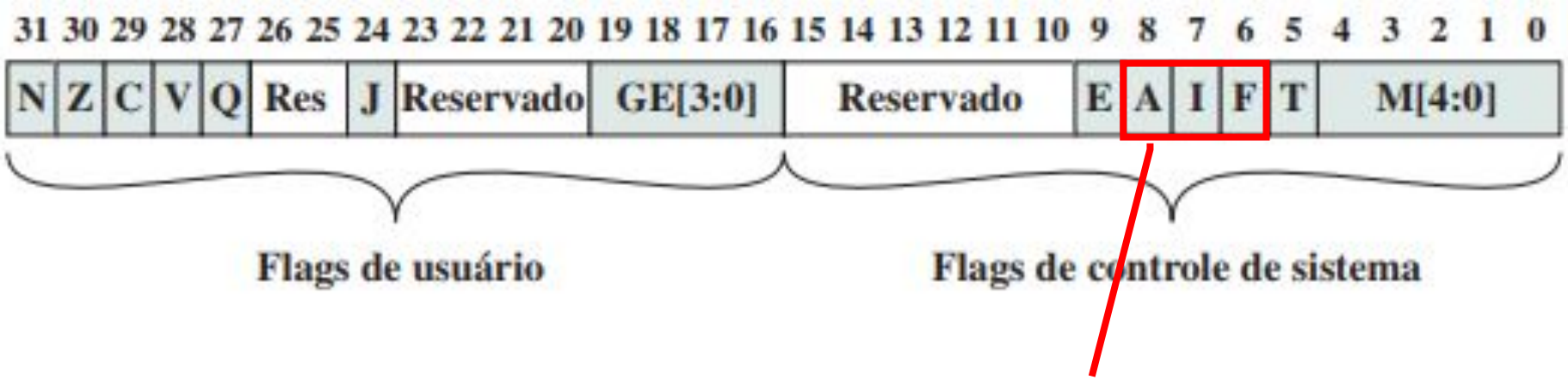
instruções SIMD usam bits[19:16] como flags maior que ou igual para bytes individuais ou meias palavras do resultado.

Organização dos Registradores

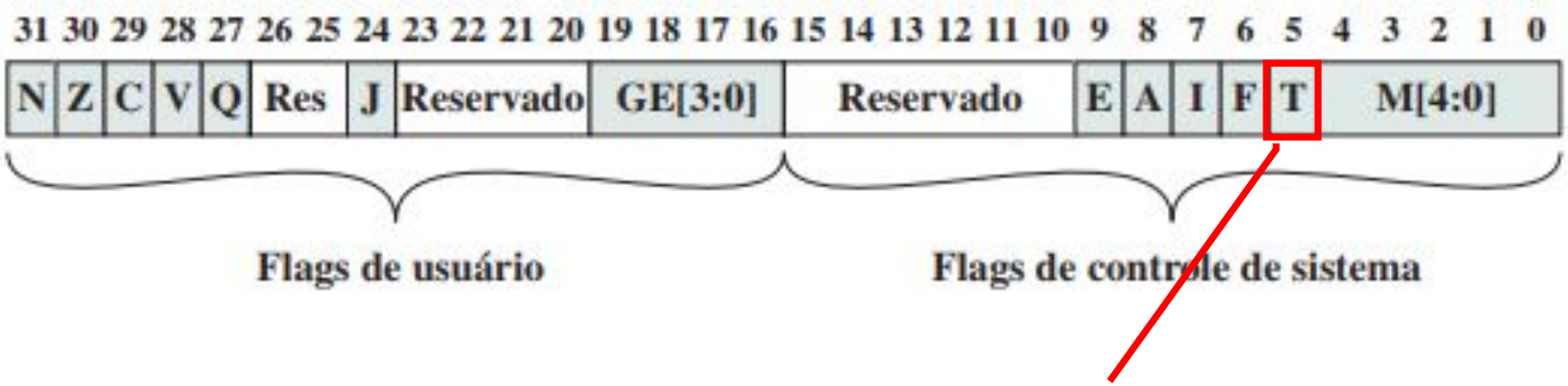


controla carga e armazenamento endianness de dados; ignorado para busca de instruções.

Organização dos Registradores

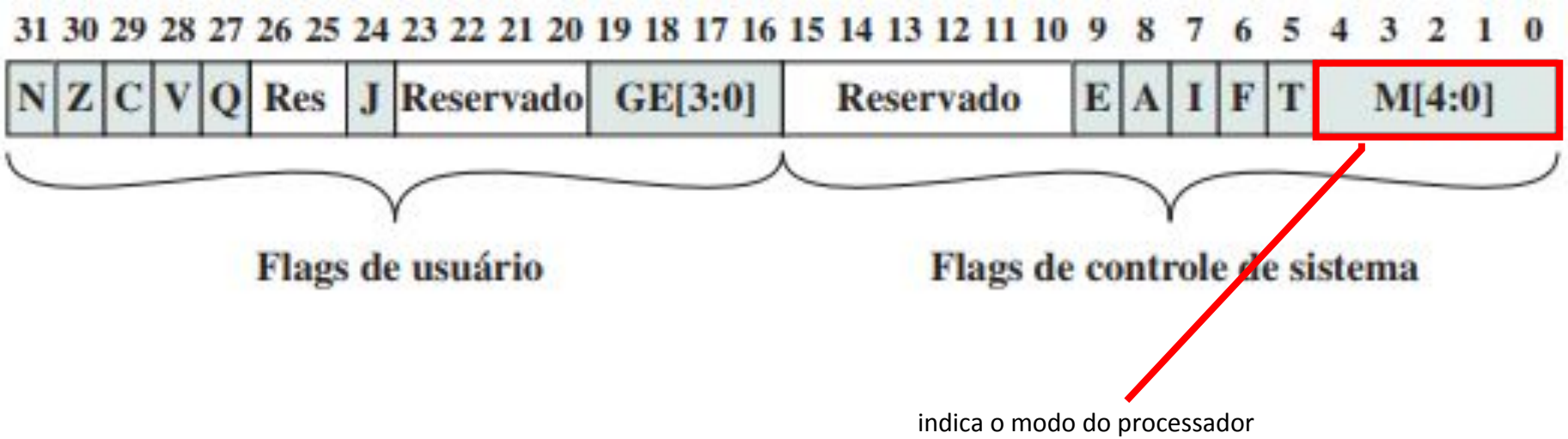


Organização dos Registradores



indica se a instrução deve ser interpretada como uma instrução ARM normal ou uma instrução Thumb.

Organização dos Registradores



Questões

1º) Quais papéis gerais são desempenhados pelos registradores do processador?

2º) O que Pipeline?

3º) Descreva o Registrador EFLAGS e qual sua função na arquitetura x86.


Obrigado