

Aula 7 Organização de Computadores Outros exemplos de organização de pipeline, previsão de desvios

Profa. Débora Matos

Previsão de desvio - outras soluções

- Atualmente os microcomputadores apresentam vários estágios de pipeline, chegando a 10 ou mais estágios.
- O pipeline funciona melhor com código linear, no entanto, a grande maioria dos programas estão repletos de instruções de desvio;
- A maioria dos computadores considera que a previsão seguinte ao desvio é tomada antes de saber o resultado do teste, como no MIPS.
- O Core i7 não utiliza essa técnica e para tal utiliza um controle bem mais complexo;

Previsão de desvio - outras soluções

- Outra técnica bastante utilizada por vários processadores é prever que os desvios condicionais para trás serão sempre tomados e os para frente não;
- Os desvios para trás muitas vezes se referem a laços, onde o teste é feito muitas vezes no final;
- Como os desvios para frente são muitas vezes decorrentes de erros e como estes são raros, é uma boa aposta;

Previsão de desvio - outras soluções

- Para desvios errados, há algumas alternativas:
 - Descartar as instruções:
 - ✓ Garantindo que as instruções que devem ser descartadas não sobrescrevam posições de memória e registradores.
 - Em vez de sobrescrever um registrador, o valor é escrito em um registrador transitório sendo somente copiado para o efetivo quando tiver a confirmação da previsão;
 - ✓ Outra opção é registrar o valor em qualquer registrador que esteja pronto para ser sobrescrito.
 - Qualquer uma destas solução é complexa e exige um controle mais elaborado;

Previsão dinâmica de desvio – outras soluções

- A estratégia de uso de uma tabela histórica na qual registra os desvios condicionais à medida que eles ocorrem é também uma <u>previsão dinâmica de</u> <u>desvios</u>;
- O objetivo é armazenar o menor número de informações possíveis. Que informações precisam ser mantidas na tabela?
 - Pode ser descartado os últimos 2 bits do endereço;
 - Um bit de validade também é requerido (como ocorre em caches);

Previsão estática de desvio

- A previsão dinâmica requer HW caro, aumentando muito a complexidade do chip, por isso, outras estratégias são levadas em consideração;
- Exemplo?
- Atribuir a complexidade ao compilador (SW) ao invés de utilizar uma solução por HW;
- O compilador pode identificar sempre que corresponder a um laço (for, while) e tomar uma decisão. A taxa de erros será maior mas a solução é mais simples;

- Atualmente a maioria das CPUS possuem pipeline e são superescalares;
- Nos processadores superescalares, há uma unidade de busca que retira instruções da memória antes que elas sejam enviadas para a unidade de decodificação;
- Posteriormente, as instruções decodificadas são emitidas para as unidades adequadas para execução;
- Em alguns casos, as instruções são desmembradas em microperações antes de serem emitidas para as unidades funcionais;

- A execução em ordem nem sempre resulta em desempenho ideal devido as várias dependências;
- Uma solução muito comum é saltar instruções dependentes para a execução de instruções independentes – fora de ordem – porém sem alterar o resultado final;
- A utilização de um registrador é monitorada através de um dispositivo denominado tabela de pontuação;
- A tabela tem um pequeno contador para cada registrador informando quantas vezes um determinado registrador está sendo usado como fonte nas instruções;

					Registers being read Registers being writter														en	
Су	#	Decoded	Iss	Ret	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	R3=R0*R1	1		1	1										1				
	2	R4=R0+R2	2		2	1	1		2112			2				1	1			
2	3	R5=R0+R1	3		3	2	1									1	1	1		
	4	R6=R1+R4	-		3	2	1	_		L	L		L	Ш	L	1	1	1	L	L
3					3	2	1					2				1	1	1		
4				1	2	1	1										1	1		
				2	1	1												1		
200	0.000			3			-	-		-	_		_		-	-		-		
5	_	DT D4 D0	4			1	١.		1										1	l.
	5	R7=R1*R2	5		-	2	1	-	1	-	-		-	H	\vdash		-	H	1	1
6	6	R1=R0-R2	-		_	2	1	_	1		_				_	_	_		1	1
7		<u> </u>		4		1	1	_		_	╙				┡		_		L	1
8				5		L	<u> </u>	_		L	L				L			L	L	L
9	100000		6		1		1	200						1						
	7	R3=R3*R1	-		1	<u> </u>	1	_	_	_	┡	_		1	┡	_	_	_	┡	L
10		with a state of the state of			1		1	_				0		1			_	_		L
11				6			L				L				L			L	L	L
12			7	A DISTRICT		1	1	1								1				
	8	R1=R4+R4	-			1	L	1		L					┖	1	L		L	L
13		guernaria antes				1		1				-				1	_			
14						1		1								1				
15			11.1911.2	7																
16			8						2					1						
17							Г		2					1						
18				8							\vdash									

Су	#	Decoded	Iss	Ret	Registers being read								Registers being written							
					0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	R3=R0*R1	1		1	1										1				
	2	R4=R0+R2	2		2	1	1									1	1			
2	3	R5=R0+R1	3		3	2	1									1	1	1		П
	4	R6=R1+R4	-		3	2	1							Lan		1	1	1		
3	5	R7=R1*R2	5		3	3	2									1	1	1		1
	6	S1=R0-R2	6		4	3	3									1	1	1		1
				2	3	3	2							4-0		1		1		1
4			4		3	4	2		1							1		1	1	1
	7	R3=R3*S1	337		3	4	2		1							1		1	1	1
	8	S2=R4+R4	8		3	4	2		3							1		1	1	1
				1	2	3	2		3									1	1	1
				3	1	2	2		3										1	1
5				6		2	1		3					1					1	1
6			7			2	1	1	3					1		1			1	1
				4		1	1	1	2					1		1				1
				5				1	2					1		1				
				8				1								1				
7								1					600			1				
8								1								1				
9				7																

Execução especulativa

- Aqui permite-se a reordenação na tentativa de preencher todas as posições de emissão.
- Os maiores ganhos ocorrem quando uma operação potencialmente lenta pode ser passada para cima para que seja iniciada mais cedo;
- A transferência de um código para cima através de um desvio é denominada elevação;
- A execução do código antes mesmo de saber se ele será necessário é denominada <u>execução</u> <u>especulativa</u>.

Execução especulativa

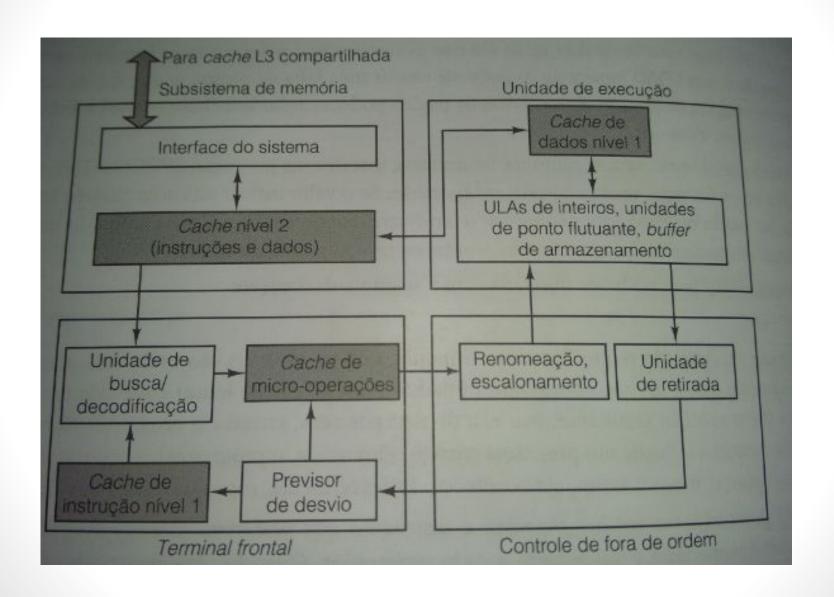
- Usar essa técnica requer suporte do <u>compilador</u> e do <u>HW</u> e ainda <u>algumas extensões</u> na arquitetura;
- Em sequências de códigos mais complicadas, para evitar que o código especulativo sobrescreva registradores antes de saber se isso é desejado, todos os registradores de destino usados pelo código especulativo são renomeados (temporários)
- Desse modo, apenas registradores temporários são modificados;
- Se as modificações deverem ser efetuadas, ao final, é só copiar os valores dos registradores temporários para os efetivos;

Execução especulativa

- Existem alguns problemas que podem ocorrer utilizando especulação, como por exemplo, a ocorrência de uma exceção (um LOAD que não está na cache, por exemplo);
- Muitas vezes algumas otimizações podem fazer a CPU ficar mais lenta do que se ela não existisse;
- Exemplo de problema na execução especulativa:

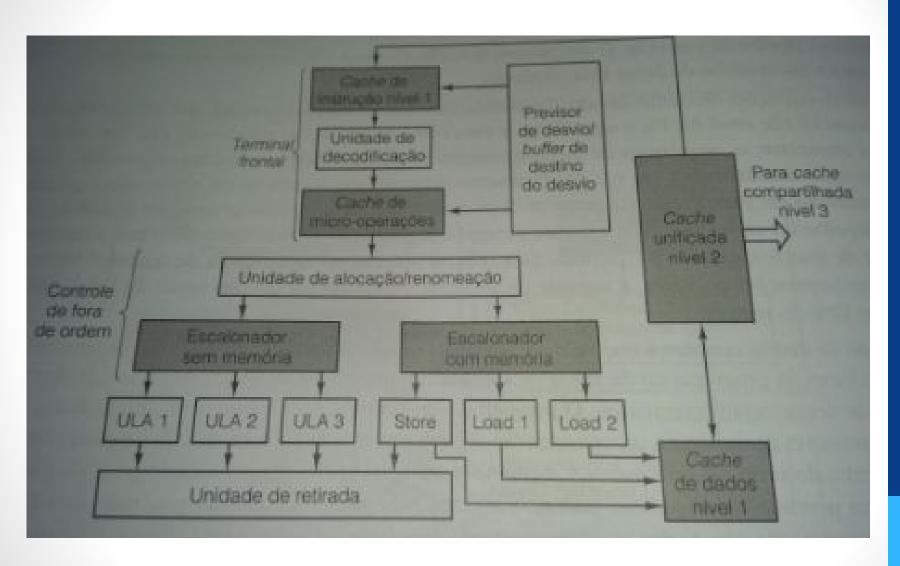
if
$$(x>0) z = y/x$$
;

Considerando que x, y e z são variáveis em ponto flutuante, uma divisão com ponto flutuante é uma operação lenta. Se x for zero (0), uma divisão por zero encerra o programa.

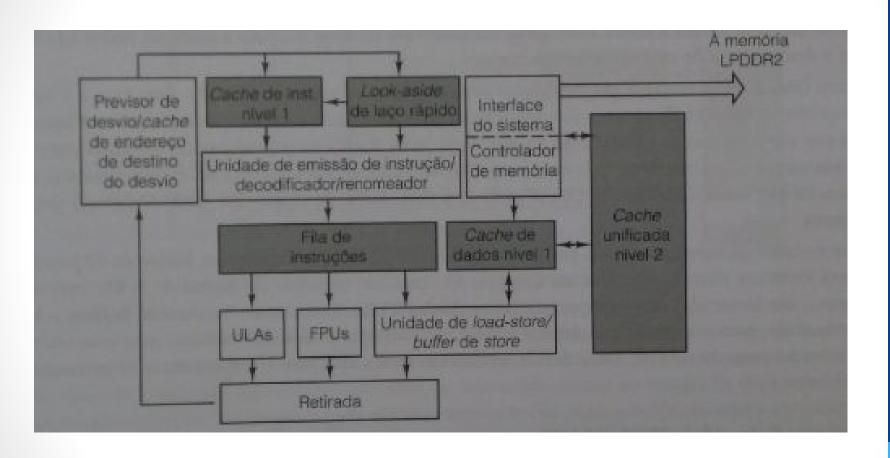


- O core i7 apresenta um grande conjunto de instruções que suporta operações com inteiros de 8, 16 e 32 bits, bem como operações de ponto flutuante de 32 bits e 64 bits;
- Os comprimentos de instruções variam de 1 a 17 bytes;
- Internamente contém um núcleo RISC moderno com elevado grau de pipelining;
- Impressionantemente, é uma arquitetura antiga com vários conceitos modernos de um processador de última geração;

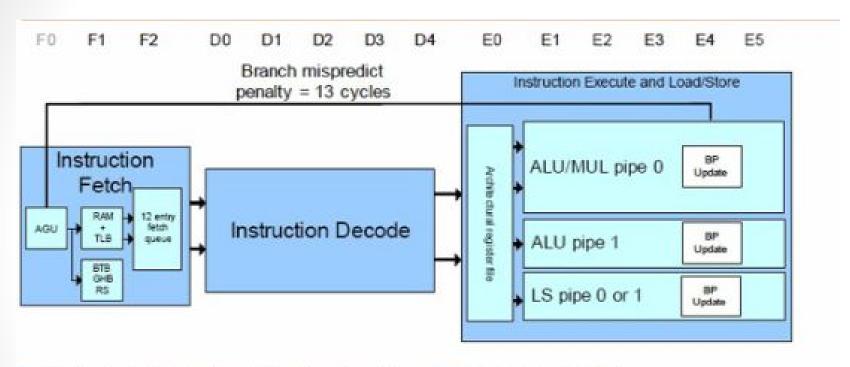
- A previsão de desvio é realizada pelo terminal frontal;
- O projeto do previsor de desvio não é aberto, já que geralmente é o componente mais crítico da velocidade geral do projeto;
- Instruções são alimentadas da cache de micro-operações para o escalonador fora de ordem;
- Quando é encontrada uma micro-operação que não pode ser executada, o escalonador a retém mas continua processando o fluxo de instruções para emitir instruções subsequentes para os quais todos os recursos estejam disponíveis;



CPU OMAP4430



CPU OMAP4430 - pipeline



- Dynamic branch predictor components
 - 512-entry 2-way branch target buffer (BTB)
 - 4K-entry GHB indexed by branch history and PC
 - 8-entry return stack

- Branch resolution
 - All branches are resolved in single stage
 - Maintains speculative and nonspeculative versions of branch history and return stack