



INSTITUTO SUPERIOR TÉCNICO

MEEC

2º SEMESTRE 2014/2015

ARQUITECTURAS AVANÇADAS DE COMPUTADORES

2º Projecto

Descrição do processador μ RISC a funcionar
em pipeline

João Baúto N° 72856

João Severino N° 73608

Docente: Prof. Leonel Sousa

10 de Maio de 2015

Conteúdo

1 Decisões de Projecto	2
2 Conflitos de Dados	3
3 Conflitos de Controlo	5
3.1 Formato I	5
3.2 Formato II e III	5
3.3 BTB	6
3.4 BPB	6
4 Estatísticas e Resultados	7

1. Decisões de Projecto

Numa primeira abordagem ao projecto, foi planeada a implementação de forwarding de dados permitindo resolver conflitos de dados e técnicas de predição dinâmica de saltos com base em Branch Prediction Buffers(BPB) e Branch Target Buffers(BTB).

A primeira versão implementada consistia em forwarding de dados com BTB contudo esta arquitectura implicava um aumento do caminho crítico do processador que a nosso ver em termos de performance não trazia benefícios ao nível do tempo de execução (Código relativo à BTB está no ficheiro BTB.vhd).

Na segunda e última versão, transitou-se para uma arquitectura com uma BPB no andar de ID&RF permitindo minimizar alguns problemas dos associados à BTB como o elevado caminho crítico e saltos incondicionais são previstos com uma certeza de 100% uma vez que já foi feito o decode da instrução. Adicionalmente foi considerado a existência de delayed branches, ou seja, é executada a instrução que se segue ao salto quer este seja *taken* ou *not-taken*. Assim no caso de a predição do salto ser incorrecta apenas é necessário introduzir um stall no pipeline (caso BPB no andar de ID com delay slot).

Esta última consideração transfere a responsabilidade da utilização do delay slot do programador para o compilador sendo este capaz ou não de encontrar uma instrução válida para colocar no delay slot.

Na secção Estatísticas e Resultados apresentam-se os resultados dos testes realizados no laboratório e um outro teste adicional.

2. Conflitos de Dados

Um dos principais problemas na execução de programas em processadores com desenvolvimento em *pipeline* é a existência de dependências de dados (registos) entre instruções sequenciais. Uma instrução que necessite dos resultados produzidos pela instrução no andar à sua frente terá que ser "atrasada", com recurso a stalls (instrução *nop*), até os resultados estejam disponíveis para leitura.

Para resolver estes problemas de dados é utilizada a técnica de hardware de *forwarding* de dados. Esta técnica consistem em passar os dados produzidos pela instrução actualmente em execução para o andar de pipeline anterior caso exista um Conflito de Dados (*Data Hazard*).

Os diferentes tipos de conflitos encontram-se descritos na tabela 2.1.

Conflito	Descrição	Solução
Read-After-Write	Tentativa de leitura de dados que estão a ser produzidos pela instrução no andar de pipeline à frente	Forward de Dados
Write-After-Write	Tentativa de escrita no mesmo registo de destino por duas instruções sequenciais	Introdução de Stall
Write-After-Read	Tentativa de escrita num registo antes de a instrução à sua frente ter feito a leitura	Alteração do registo destino

Tabela 2.1: Tipos de Conflitos.

Aplicado à arquitectura do μ RISC apenas é possível obter conflitos do tipo Read-After-Write (RAW) uma vez que a esta arquitectura está associada execução em ordem ao contrário de nas arquitecturas com Tomasulo onde é feita execução fora de ordem após o *dispatch* das instruções permitindo assim eliminar conflitos Write-After-Read (WAR) quando é feito o *issue* e *dispatch* de instruções para as estações de reserva e Write-After-Write (WAW), também aplicados ao Tomasulo, quando é realizada a escrita no *Common Data Bus* (CDB).

Na arquitectura desenvolvida foi utilizada a técnica de *forwarding* de dados "apanhando" dos sinais de saídas das unidades funcionais e direcionando-as para o início do andar anterior com recurso a *Multiplexers*.

O controlo dos sinais de selecção destes muxes é feito no bloco *DataHazardUnit.vhd* onde é feita a seguinte verificação:

1. Comparação entre os registos RA e RB (registos de leitura) e com o registo de destino (registo de escrita)
2. Se o *id* dos registos forem iguais é um sinal prematuro de que será necessário fazer forwarding
3. De seguida é feita a verificação da operação. Se for uma instrução válida é a confirmação de que forward é necessário. A verificação da operação deve-se à possibilidade de colisão entre o *OPCODE* de uma instrução com o valor de uma constante.
4. É então "enviado" o sinal de selecção para a mux correcta permitindo assim executar a instrução presente no andar de ID&RF sem ser preciso introduzir um stall no *pipeline*.

Na figura 2.1 está a arquitectura com o forward de dados. Este forward consistem apanhar os sinais da Unidade Lógico-Aritmética, Unidade de Constantes e saída da Memória para o andar de pipeline anterior.

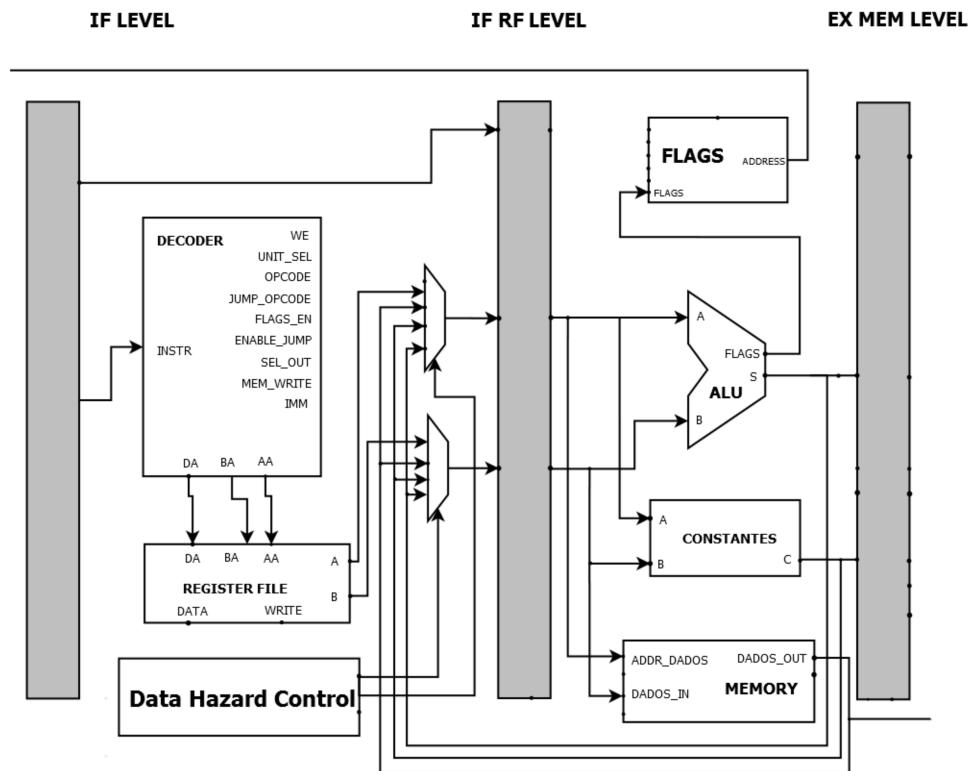


Figura 2.1: Forward de Dados.

Nota: Para manter a simplicidade do esquema foram eliminados sinais presentes no pipeline.

3. Conflitos de Controlo

A unidade de Conflitos de Controlo analisa as instruções que chegam ao andar de *Instruction Decoding/Register Fetch* e em caso de se tratar de uma instrução de transferência de controlo decide o próximo endereço a colocar no registo do *Program Counter*.

As instruções de transferência de controlo dividem-se em 3 formatos:

3.1 Formato I

15	14	13	12	11	8	7	0
0	0	0	0	Cond		Destino	
0	0	0	1	Cond		Destino	

Para este formato é usada uma BTB onde são armazenados os dados para prever o resultado. Para executar a predição dos saltos são usados 2 bits que codificam 4 estados, onde o bit mais significativo é usado para decidir entre saltar e não saltar.

- 00 strong not taken
- 01 weak not taken
- 10 weak taken
- 11 strong taken

Esta predição é depois atualizada com os resultados da unidade de flags e com a utilização de uma BPB e armazenada esta nova informação na BTB.

3.2 Formato II e III

No caso dos Formatos II e III não existe a necessidade de fazer predição, pois tratam-se de saltos incondicionais, ou seja, ocorrem sempre.

No caso do Formato II é um salto com um offset e é usado um somador na própria unidade de Conflitos de Controlo.

15	14	13	12	11		0
0	0	1	0		Destino	

No caso do Formato III é um salto para um valor armazenado num registo RB, valor este que é verificado pela unidade de Conflito de Dados se o valor presente no Register File é o mais atualizado ou se há a necessidade de fazer forwarding.

15	14	13	12	11	10		3	2	0
0	0	1	1	R			RB		

3.3 BTB

A BTB consiste numa pequena memória que funciona como *cache* que armazena os resultados de ocorrências passadas da mesma instrução e qual o seu desfecho. A BTB tem 128 posições endereçadas pelos 7 bits menos significativos do PC da instrução de controlo e os restantes bits são armazenados como uma TAG que é usada para verificar se a instrução em memória corresponde à mesma transferência controlo.

É também armazenados os bits de predição que ajudam a determinar se o salto vai ser *taken* ou *not taken*.

3.4 BPB

A BPB consiste numa unidade que recebe os bits de predição antigos e o verdadeiro resultado da instrução proveniente da unidade de flags e calcula os bits atualizados.

Old bits	Taken	New bits
0 0	0	0 0
0 0	1	0 1
0 1	0	0 0
0 1	1	1 0
1 0	0	0 1
1 0	1	1 1
1 1	0	1 0
1 1	1	1 1

Estes novos bits são armazenados na BTB para serem usados em próximas ocorrências da instrução.

4. Estatísticas e Resultados

É importante referir uma grande consideração feita na elaboração destas estatísticas que foi o facto de todos os delay slots serem utilizados uniformemente, ou seja, frequência absoluta uniforme (na prática um delay slot pode ter uma frequência absoluta muito superior a outro).

# Teste	Delay Slot Usage (%)	# Operations	# Cycles	# Nop	# Jumps	# Override	Jump Miss Rate (%)	CPI	Execution Time (ns)	Speed Up	Frequency (MHz)
1	0	59	65	6	7	0	0	1,10	0,256	1,000	253,89
	16,67	59	64	5				1,08	0,252	1,016	
	33,3	59	63	4				1,07	0,248	1,032	
	50	59	62	3				1,05	0,244	1,048	
	66,6	59	61	2				1,03	0,240	1,066	
	83,3	59	60	1				1,02	0,236	1,083	
	100	59	59	0				1	0,232	1,102	
2	0	2343	2914	481	523	90	17,21	1,24	11,477	1,000	253,89
	11,11	2343	2771	428				1,18	10,914	1,052	
	22,22	2343	2718	375				1,16	10,705	1,072	
	33,33	2343	2664	321				1,14	10,493	1,094	
	44,44	2343	2611	268				1,11	10,284	1,116	
	55,56	2343	2557	214				1,09	10,071	1,140	
	66,67	2343	2504	161				1,07	9,863	1,164	
	77,78	2343	2450	107				1,05	9,650	1,189	
	88,89	2343	2397	54				1,02	9,441	1,216	
	100	2343	2343	0				1,00	9,228	1,244	
3	0	1047	1426	311	279	68	24,37	1,36	5,617	1,000	253,89
	16,67	1047	1307	260				1,25	5,148	1,091	
	33,3	1047	1255	208				1,20	4,943	1,136	
	50	1047	1203	156				1,15	4,738	1,185	
	66,6	1047	1151	104				1,10	4,533	1,239	
	83,3	1047	1099	52				1,05	4,329	1,298	
	100	1047	1047	0				1,00	4,124	1,362	
4	0	4014670	6613297	2447628	2656355	150999	5,684	1,65	26047,883	1,000	253,89
	11,11	4014670	6190340	2175670				1,54	24381,976	1,068	
	22,22	4014670	5918381	1903711				1,47	23310,808	1,117	
	33,33	4014670	5646422	1631752				1,41	22239,639	1,171	
	44,44	4014670	5374464	1359794				1,34	21168,475	1,231	
	55,56	4014670	5102505	1087835				1,27	20097,306	1,296	
	66,67	4014670	4830546	815876				1,20	19026,137	1,369	
	77,78	4014670	4558588	543918				1,14	17954,973	1,451	
	88,89	4014670	4286629	271959				1,07	16883,804	1,543	
	100	4014670	4014670	0				1,00	15812,635	1,647	

Figura 4.1: Estatísticas do Pipeline para os Testes realizados.

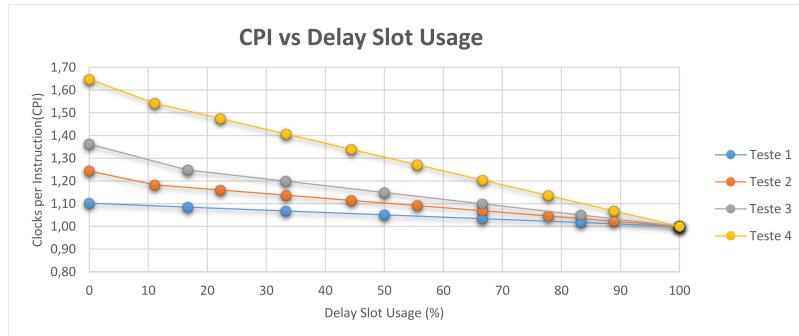


Figura 4.2: Relação CPI - Utilização do Delay Slot para os Testes 1,2,3 e 4.

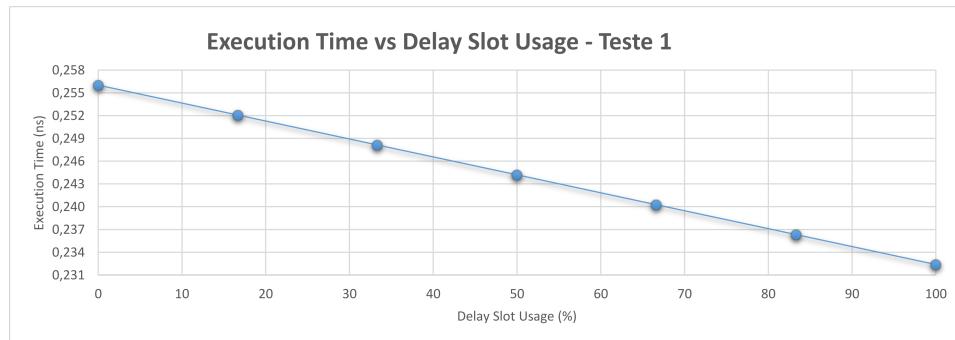


Figura 4.3: Relação Tempo de execução - Utilização do Delay Slot para Teste 1

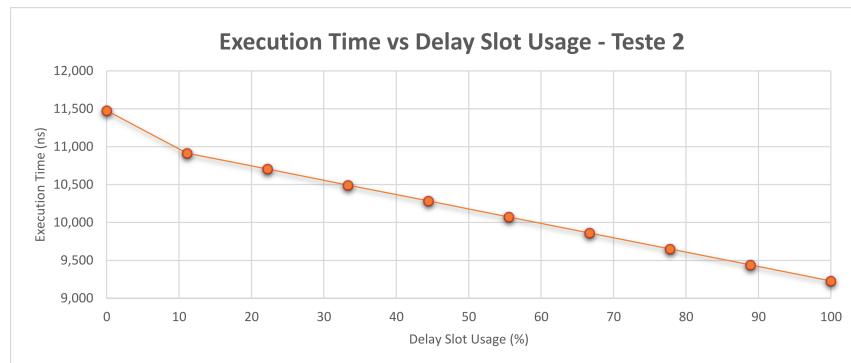


Figura 4.4: Relação Tempo de execução - Utilização do Delay Slot para Teste 2.

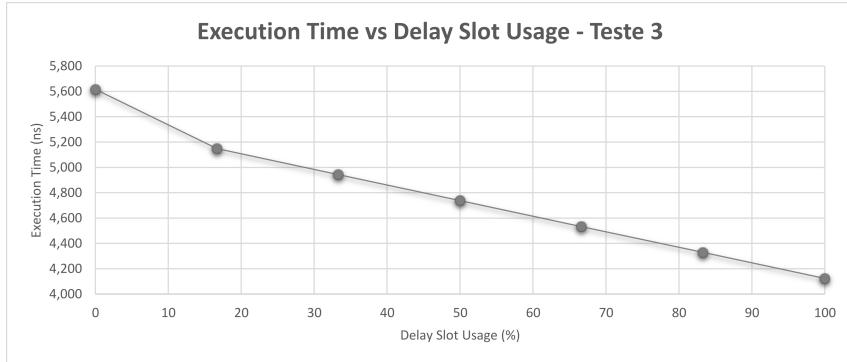


Figura 4.5: Relação Tempo de execução - Utilização do Delay Slot para Teste 3.

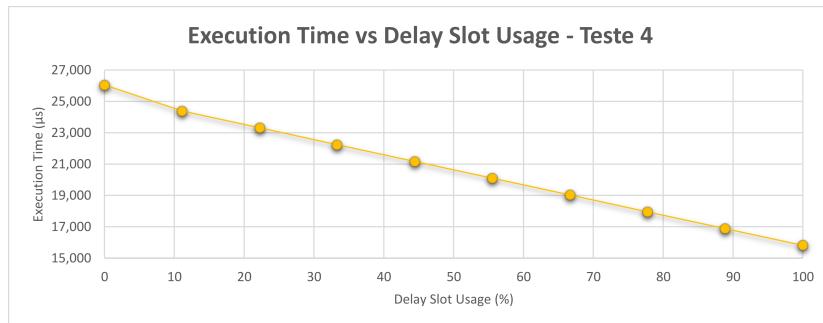


Figura 4.6: Relação Tempo de execução - Utilização do Delay Slot para Teste 4.