PROVA 1 - ARQUITETURA DE COMPUTADORES

Aluno: Daniel Sant' Anna Andrade
Matrícula: 20200036904

3) Pois assim, é possível utilizar em um mesmo ciclo, o banco de registradores em duas instituições diferentes, uma escrevendo (1º 1ns) e a outra lendo (2º 1ns).

4) O tempo de execução de uma instrução no pipeline precisa ser igual para todas as partes da instrução, ou seja: busca, decodificação, execução, acesso a memória e escrita nos registradores necessita ter a mesma duração. Assim, cada instrução precisa ter o mesmo tempo de duração. Apesar de lw ser a de maior duração levando 8ns, a sw ter 7ns, as de tipo r levarem 6ns e as de beqz levarem 5ns, todos os tipos de instruções precisam passar pelas mesmas etapas do pipeline mesmo que elas não sejam utilizadas. Assim, o tempo de cada instrução individualmente será sempre a mesma coisa, o que irá ser reduzido é o tempo total para executar todo o conjunto de instruções.

5) No MIPS cada instrução leva 9ns para ser executada, sendo 2ns para busca da instrução, 2ns para leitura dos registradores (apesar de só utilizar 1ns, tem um espaço de 1ns antes da leitura para não gerar conflito tanto com a escrita nos registradores como com o resto das outras instruções do pipeline), 2ns para a execução da instrução, 2ns para o acesso a memória (que apesar de passar por esse processo, apenas as instruções de load e store realmente a utilizam) e 1ns para realizar a escrita nos resultados. Mesmo que uma instrução não precise utilizar todos os passos do ciclo de instrução, ela precisa passar por todos os passos para não atrapalhar a execução do pipeline, por isso, o tempo total gasto será de 9 x o tempo do estágio no mínimo, em caso de conflito é necessário possuir mais estágios pois precisa se inserir bolhas.

6) Pois a alocação de informação na memória cache ocorre em qualquer espaço que estiver vazio, assim ela preenche por completo a cache. Porém, para alocar espaço para novas instruções é necessário a utilização de algoritmos de substituição, que podem ser: FIFO, LRU ou Aleatório.

8) Devemos utilizar algoritmos de substituição de bloco quando utilizamos mapeamento completamente associativo. Quando a cache estiver completamente preenchida, é necessário substituir algum dos dados da mesma, sendo mais indicado o LRU, que junto de um hardware contador, irá retirar o bloco que não é utilizado a mais tempo. Os outros métodos de mapeamento também substituem blocos, mas a substituição é mais direta pois cada mapeamento tem uma forma de organizar os blocos em um espaço mais preciso.

10)

	T1	T2	T3	T4	T5	T6	T7	T8	T9
LW R3 30($8)	BI	REG	EXE	MEM	[redacted]				
OR R4 R2 R5		BI	REG	EXE	MEM	[redacted]			
AND R7 R6 R4			BI	REG	EXE	MEM	REG		
LW R6 10($9)				BI	[redacted]	EXE	MEM	REG	
SUB R7 R5 R8					BI	[redacted]	EXE	MEM	REG

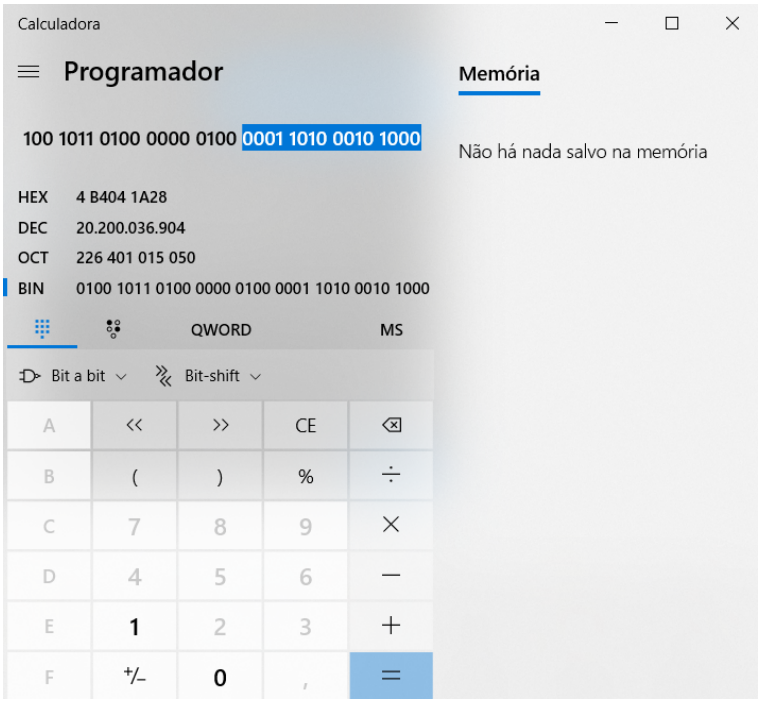
Como a máquina faz adiantamento de dados, ela irá utilizar 9 ciclos para realizar o conjunto das instruções. Mesmo tendo dependência de dados, da segunda para a terceira

instrução, o adiantamento de dados (bypassing) envia a informação direto do final do ciclo de execução da segunda instrução, para o início do ciclo da execução da terceira instrução, como indicado pela seta no desenho. E apesar de no t5 e no t6 duas instruções utilizarem ao mesmo tempo os registradores, a que está escrevendo utiliza o 1º ns para utilizar o registrador e a que está lendo utiliza o 2º ns para utilizar o registrador.

11) A instrução de load pode gerar dependências no pipeline, necessitando utilizar bolhas para atrasar algumas instruçõs, pois a instrução de load necessita de todos os 5 ciclos para terminar a sua execução. É bom lembrar que ela não cria dependência em instruções que estão utilizando o banco de registradores ao mesmo tempo que ela, pois ela irá escrever nos registradores primeiro e a outra instrução irá ler dos registradores só depois que ela terminar de escrever. Reordenar as instruções, quando possível, também auxilia a esquivar de dependências.

12) A localidade espacial determina que ao utilizar uma instrução, a probabilidade de precisar das próximas isntruções é alta também. Isso ocorre pois as instruções são escritas na memória uma após a outra, de forma sequencial. Sendo assim, ao se guardar a informação na cache, se guarda em blocos de 32 bits, de modo que as instruções estejam todas uma após a outra, facilitando o acesso utilizando ao máximo a cache.

13)



Os 16 primeiros bits da minha matrícula é: num1 = 1 1010 0010 1000. Somando esse valor a 0.101 (0,625 ou $2^{-1}.1 + 2^{-2}.0 + 2^{-3}.1$), temos num1 = 1101000101000.101.

Esse resultado no padrão IEEE é $1101000101000.101 \cdot 2^{0}$.
Movendo a vírgula para a direita, fica: $1.101000101000101 \cdot 2^{12}$.

Na tabela, é necessário primeiramente descobrir o expoente, somando o 12 com 127.

1	1	1	1	1				
128	64	32	16	8	4	2	1	
	1	1	1	1	1	1	1	127
				1	1	0	0	12
1	0	0	0	1	0	1	1	139

Agora é só alocar os valores na tabela de 32 bits, bit mais significativo (31) é o bit que indica o sinal, caso seja positivo o bit é 0, caso seja negativo o bit é 1.

Do 30 ao 23, se coloca o expoente em binário (a soma do expoente achado com 127, no meu caso 139).

O restante será a mantissa, o valor depois da vírgula após encontrar o expoente no caso 101000101000101. Os espaços que sobrarem se completa com 0.

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	1	0	0	0	1	0	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
4				5				D				1				4				5				0				0			