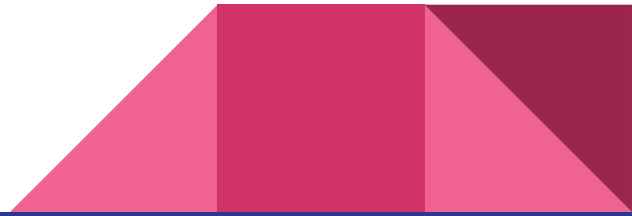
Projeto 2: Desempenho do Processador

Membros do grupo:

- Pedro Elias Lucas Ramos Meireles RA: 148914
- Andre Tsuyoshi Sakiyama RA: 150547
- Caio Vinicius Piologo Véras Fernandes RA: 145574
- Luiz Rodolfo Felet Sekijima RA: 117842

Principais tópicos

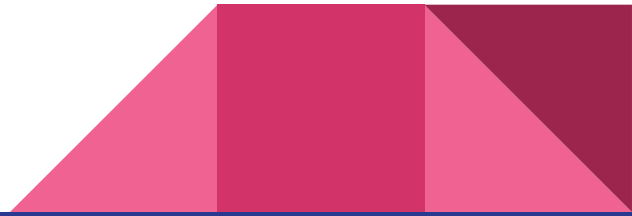
- Objetivos
- Hazards
- Branch Prediction
- Caches



Objetivos

Objetivos

- Avaliar o desempenho de máquinas em diferentes condições
- Aplicar diferentes métodos de processamento
- Estudar o impacto deles na execução de um programa



Hazards

Tipos de hazards

- Hazard Estrutural
  - Um mesmo componente de hardware é requisitado por diferentes estágios do pipeline ao mesmo tempo.
- Hazard de Controle
  - Ocorre quando a próxima instrução não é conhecida pelo processador.
  - Também ocorre devido a mudanças no fluxo de controle (quando se altera o registrador PC)
- Hazard de Dados
  - Um input de instrução não está disponível para uso no ciclo em que o dado é necessário.
- Hazard de Memória
  - Quando é necessário buscar informação na memória.
  - Esse tipo de operação é muito custosa em relação ao tempo de execução do programa

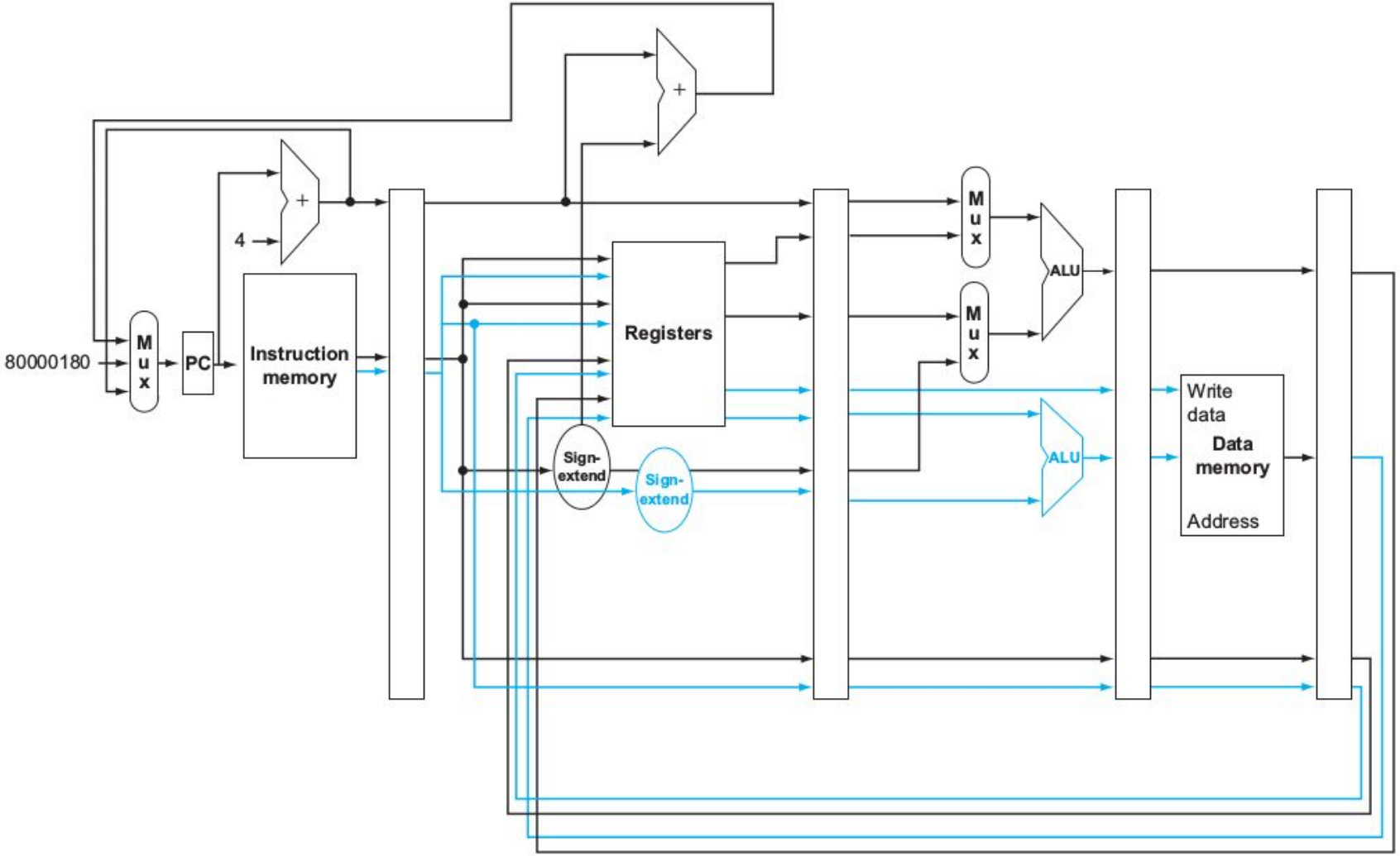
Hazard de dados

Hazard de dados - Tipos

- Read After Write (RAW): j tenta ler um dado antes que seja escrito em i
- Write After Read (WAR): instrução j tenta escrever antes que i leia
- Write After Write (WAW): j tenta escrever antes que i escreva
- Apenas hazards do tipo RAW acontecem nos pipelines apresentados
    - Resolvidos por fowarding e stalls
    - Pipeline escalar: load seguido por instrução que depende de dados do load gera um bubble

Hazard de dados - Pipeline Superescalar

- Usa de paralelismo para rodar mais instruções por ciclo
- Aloca duas instruções por ciclo de modo estático
  - Operações aritiméticas (ALU) e branches em um pipeline
  - Loads e stores em outro pipeline
- Na teoria: ciclos caem pela metade
- Na prática: novos hazards surgem
  - Instrução de add que depende de load em mesmo estágio: 2 stalls
  - Instrução de load que depende de add em mesmo estágio: 1 stall

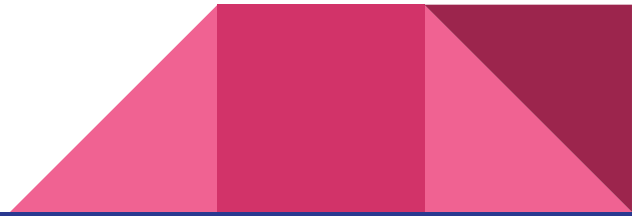

80000180
M
u
x
PC
Instruction
memory
4
+
+
Registers
Sign-
extend
Sign-
extend
M
u
x
M
u
x
ALU
ALU
Write
data
Data
memory
Address

Resultados: Pipeline Escalar

BenchMark	Ciclos	Data Stalls
Bitcount	511004870	8170
Dijkstra	181060823	30155599
Rijndael (encode)	420492338	21916061
Rijndael (decode)	441097079	29800097
JPEG (encode)	102672414	5610850
JPEG (decode)	32161693	2892153

Resultados: Pipeline Superescalar

BenchMark	Ciclos	Data Stalls
Bitcount	482864335	15778283
Dijkstra	129641625	70925848
Rijndael (encode)	342272216	75048164
Rijndael (decode)	111294148	105984811
JPEG (encode)	73682627	34104664
JPEG (decode)	21784657	14492166



Branch Prediction

Branch Estático

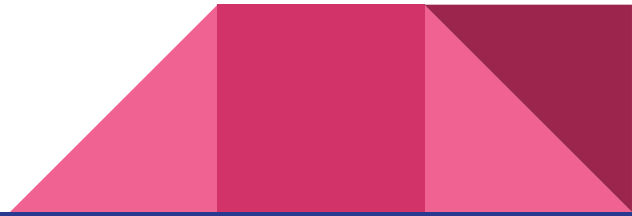
- Assume que a posição de troca para o próximo PC estará sempre correta, e carrega desta forma, no pipeline, as posições referentes a este PC.
- Se realmente o salto estava correto: não há alteração no fluxo de execução
- Senão.... Ele cria bolhas para corrigir o erro, para todas as instruções com o PC errado.
    - Bolhas nada mais são do que instruções que não deveriam ter sido executadas naquele momento.

Resultados: Branch Estático

BenchMark	Instruções	Branches	Branch Misprediction	Misses Percentage
Bitcount	536894266	117341166	28130876	24.0%
Dijkstra	223690619	51329314	18271877	35.6%
Rijndael (encode)	453556705	15040294	10336543	68.7%
Rijndael (decode)	483868969	16654333	9924102	59.6%
JPEG (encode)	111543858	13528023	6910492	51.1%
JPEG (decode)	35847190	1816631	222811	12.3%

Branch Dinâmico

- Leva em consideração a existência de muito loops, onde há varios branchs seguidos para o mesmo endereço
- Utiliza bits para armazenar se recentemente houve ou não um branch.
- Os mais empregados são as máquinas de estado de 1 e 2 bits
- Mais acertos do que o branch prediction estático



Branch Dinâmico

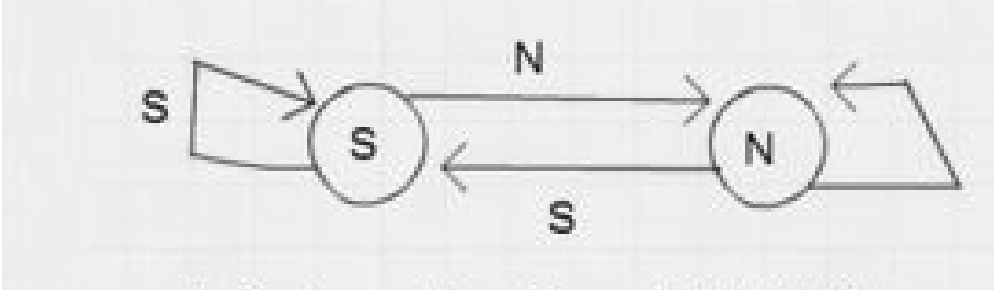


Imagem 2: Branch prediction dinâmico de 1 bit.

Resultados: Branch Dinâmico

BenchMark	Instruções	Branches	Branch Misprediction	Misses Percentage
Bitcount	536894310	117341166	6750164	5.8%
Dijkstra	223691867	51329314	153864	0.3%
Rijndael (encode)	453563104	15040294	412354	2.7%
Rijndael (decode)	483868969	16654333	405998	2.4%
JPEG (encode)	111543858	13528023	287842	2.1%
JPEG (decode)	35848023	1816631	58818	3.2%

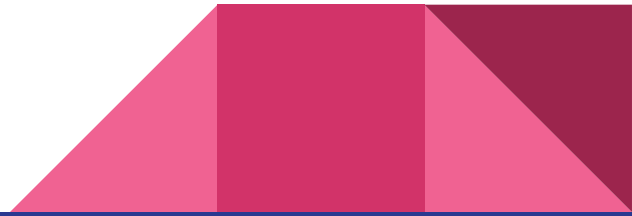
Conclusão - Branches

- Como esperado, o Branch Dinâmico foi muito mais eficiente do que o Branch Estático, visto que ele possui muito menos mispreditcions.
- Enquanto que o Branch Estático possui uma média de predição de erros entre 10% a 70%....
- O Branch Dinâmico possui uma média de predição de erros entre 0% ~10%.

Caches

Etapas para avaliação da cache

1. Produção dos traces executáveis do mips para Dinero IV
    a. Através de modificações do código mips_isa.cpp
2. Execução dos traces gerados no Dinero IV
    a. Para cada benchmark e Configuração de cache L1 e L2.



Configurações de cache compostas no projeto

	Tamanho L1 data	Tamanho L1 instrução	Tamanho L2 Unificada	Bloco L1 data	Bloco L1 Instrução	Bloco L2 Unificada
Configuração 1:	**8K**	**8K**	**128K**	**64**	**64**	**64**
Configuração 2:	**64K**	**64K**	**128K**	**512**	**512**	**64**
Configuração 3:	**8K**	**8K**	**256K**	**64**	**64**	**512**
Configuração 4:	**64K**	**64K**	**256K**	**512**	**512**	**512**

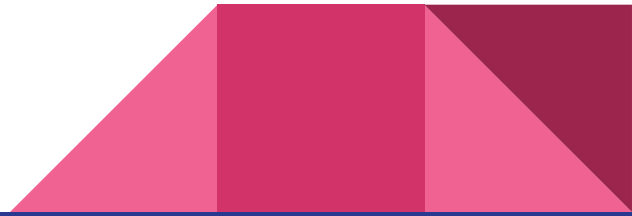
Programa	Dijkstra					
Tipo de Cache	Cache L1 data		Cache L1 Instruction		Cache L2 Unificated	
Configuração//Parâmetro	Demand Fetches (Total)	Demand miss rate (total)	Demand Fetches (Total)	Demand miss rate (total)	Demand Fetches (Total)	Demand miss rate (total)
Configuração 1	63336187	2,92%	223690592	0,20%	2718646	0,80%
Configuração 2	63336187	0,03%	223690592	0,00%	216648	17,46%
Configuração 3	63336187	2,92%	223690592	0,20%	2718646	0,11%
Configuração 4	63336187	0,03%	223690592	0,00%	27081	9,36%

Tabela: exemplo dos dados do benchmark Dijkstra

Conclusão

Configuração 1

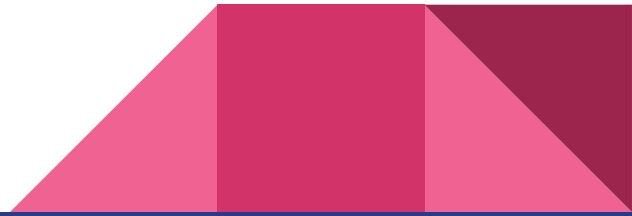
- Pipeline de 5 estágios
- Processador escalar
- Clock: 7.5 ns (133 MHz)
- Tamanho L1 dados: 8KB
- Tamanho L1 instrução: 8KB
- Tamanho L2 unificada: 128KB
- Bloco L1 dados: 64B
- Bloco L1 instruções: 64B
- Bloco L2 unificada: 64B
- Sem branch prediction



Configuração 1	Bitcount	Dijkstra	Rijndael (encode)	Rijndael (decode)	JPEG(encode)	JPEG (decode)
Total instruções	536894239	223690592	453556707	483862590	111543017	35847192
Miss rate L1 dados	0.00%	2.92%	23.91%	22.35%	2.23%	2.33%
Miss rate L1 instruções	0.00%	0.20%	3.88%	4.14%	0.04%	0.09%
Miss rate L2	17.30%	0.80%	17.97%	13.89%	19.10%	3.04%
Stalls por branch	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Stalls por dados	0.03%	13.48%	16.94%	20.88%	15.21%	22.19%
Total ciclos	771588925	361294184	682386397	717429446	47800555	43171316
Tempo de execução	5.79 s	2.71 s	5.12 s	5.38 s	1.11 s	0.32 s

Configuração 6

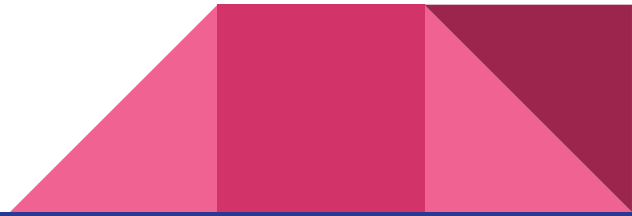
- Pipeline de 5 estágios
- Processador escalar
- Clock: 7.5 ns (133MHz)
- Tamanho L1 dados: 64KB
- Tamanho L1 instrução: 64KB
- Tamanho L2 unificada: 256KB
- Bloco L1 dados: 512B
- Bloco L1 instruções: 512B
- Bloco L2 unificada: 512B
- Branch prediction dinâmico



Configuração 6	Bitcount	Dijkstra	Rijndael(encode)	Rijndael (decode)	JPEG(encode)	JPEG (decode)
Total instruções	536894310	223691867	453563104	483868969	111543858	35848023
Miss rate L1 dados	0.00%	0.03%	18.12%	16.32%	0.75%	0.38%
Miss rate L1 instruções	0.00%	0.00%	0.36%	0.00%	0.00%	0.01%
Miss rate L2	61.60%	9.36%	56.74%	47.72%	57.98%	36.99%
Stalls por branch	5.8%	0.3%	2.7%	2.4%	2.1%	3.2%
Stalls por dados	0.03%	13.48%	16.94%	20.88%	15.21%	22.19%
Total ciclos	550404352	254216016	624734764	694992963	119886193	39142780
Tempo de execução	4.13 s	1.91 s	4.69 s	5.21 s	0.90 s	0.29 s

Configuração 7

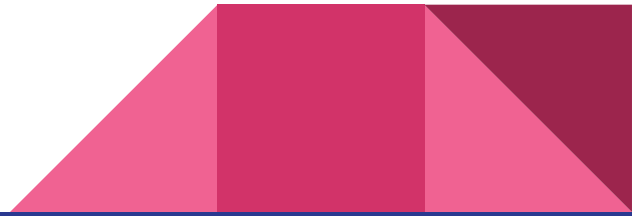
- Pipeline de 5 estágios
- Processador superescalar
- Clock: 5 ns (200MHz)
- Tamanho L1 dados: 64KB
- Tamanho L1 instrução: 64KB
- Tamanho L2 unificada: 256KB
- Bloco L1 dados: 512B
- Bloco L1 instruções: 512B
- Bloco L2 unificada: 512B
- Branch prediction dinâmico



Configuração 7	Bitcount	Dijkstra	Rijndael(encode)	Rijndael (decode)	JPEG(encode)	JPEG (decode)
Total instruções	536894310	223691867	453563104	483868969	111543858	35848023
Miss rate L1 dados	0.00%	0.03%	18.12%	16.32%	0.75%	0.38%
Miss rate L1 instruções	0.00%	0.00%	0.36%	0.00%	0.00%	0.01%
Miss rate L2	61.60%	9.36%	56.74%	47.72%	57.98%	36.99%
Stalls por branch	5.8%	0.3%	2.7%	2.4%	2.1%	3.2%
Stalls por dados	0.03%	13.69%	17.81%	22.55%	16.95%	28.53%
Total ciclos	566174465	294986265	677866867	771177677	148380007	50742793
Tempo de execução	2.83 s	1.47s	3.39s	3.86s	0.74s	0.25s

Comparação

- Configurações 1 e 6
  - Aumento de bloco e tamanho da cache ⇒ Menor Miss Rate (L1 e L2)
  - Branch Predictor Dinâmico ⇒ Redução em stalls de branch
  - Melhor performance ⇒ Configuração 6
- Configurações 6 e 7
  - Maior quantidade de ciclos ⇒ por volta de 10%
  - Redução no clock ⇒ 33%
  - Melhor performance ⇒ Superescalar (Configuração 7)



Obrigado!