UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Segundo Trabalho Prático
ORGANIZACÃO DE COMPUTADORES B 2009/2

Germano de Mello Andersson
137719

Porto Alegre, 29 de novembro de 2009.

1. Investigar a influência do tipo de mapeamento empregado (direto, associativo por conjunto e totalmente associativo) e da política de reposição no desempenho da cache.

a) Dentre os experimentos realizados, qual das políticas de reposição apresenta o melhor resultado para cada uma das caches em sua opinião? Por que?

Para cache de dados a política LRU teve performance destacada em relação as outras pelo fator reutilização, característica forte de dados. Todo programa possui muitas rotinas que trabalham com as mesmas varíaveis (por consequência mesmo endereço de memória) e, além disso, estas rotinas acontecem em um espaco de tempo contínuo. Estes princípios fazem com que o conceito de localidade temporal, empregado em caches, seja decisivo no ganho de performance do programa.

Quanto a cache de instruções, a diferença de performance entre as políticas de reposição não foi muito grande. De qualquer forma, na média, a melhor performance se deu na política randômica. Acredito que a política LRU, a com piores resultados, não tenha tido muito sucesso pelo fato de que instruções são reutilizadas apenas em laços que, apesar de representarem boa parte de um programa, não assumem sua totalidade. Quanto a performance da política randômica, conforme discutido em aula, por termos uma grande probabilidade de acertarmos com a escolha de reposição, no caso 1/32, isso dá a ela boas condições de ser uma boa escolha.

b) A primeira linha da tabela acima representa que tipo de cache em termos de mapeamento? Considerando essa mesma linha citada, explique o porquê dos resultados obtidos para as três políticas aplicadas.

Representa o mapeamento direto. O resultado para este mapeamento foi exatamente o mesmo para todas políticas de reposição. Isto acontece porquê o fato de executarmos o mesmo programa e não termos mais de uma alternativa para alocação de cada bloco de dados|instruções faz com que a dinâmica de utilização da cache seja exatamente a mesma.

c) Considerando-se que as caches de dados e instruções estão separadas, qual a melhor combinação (em termos da menor taxa de misses obtida) entre cache de instruções e de dados considerando-se qualquer possibilidade de configuração para ambas (em termos de associatividade, no de conjuntos e política de reposição)?

Cache Instrucoes:
- Num Conj: 8
- Tam Blocos: 32
- Associatividade: 4
- Política: randômica

Cache Dados:
- Num Conj: 8
- Tam Blocos: 32
- Associatividade: 4
- Política: LRU

d) Qual o comportamento das duas caches quando do aumento da associatividade (e conseqüente diminuição do número de conjuntos)?

As duas caches possuem melhor performance no momento em que há certo equilíbrio entre o número de conjuntos e associatividade. Nos extremos, consequentemente no aumento da associatividade, a performance começa a diminuir em relação ao momento de equilíbrio.

2. Investigar a influência da variação do tamanho do bloco no desempenho da cache

a) Qual o comportamento observado para as duas caches em termos de percentual de erro no seu acesso?

Para a cache de instruções, conforme aumentamos o tamanho do bloco, diminuímos o percentual de erro. Já a cache de dados, conforme aumentamos o tamanho do bloco, aumentamos o percentual de erro.

b) Como você explicaria os comportamentos observados para as caches de instruções e de dados?

Por padrão, um programa costuma ter a próxima instrução a ser executada em posição de memória sequente a atual. Esta característica de localidade espacial é auxiliada quando aumentamos o tamanho do bloco utilizado na cache, pois ao trazermos determinada instrução para a cache, suas subsequentes são trazidas juntamente.

Ao contrário, os dados de um programa estão espalhados pela memória. Ao utilizarmos um tamanho de bloco elevado, temos uma penalidade de miss muito maior, pois utilizamos apenas parte do bloco trazido, onerando a utilizacão da cache. Por consequência do espalhamento dos dados pela memória, quando aumentamos o tamanho dos blocos, diminuímos o tamanho de linhas disponíveis, fazendo com que o algoritmo de reposição seja acionado mais vezes. Isto, obviamente, onera a performance do programa.

3. Investigar a influência do tamanho total da cache e do tamanho de bloco no desempenho da cache

Comparando linha a linha os resultados obtidos, considerando a mesma associatividade e mesmo tamanho de cache, mas com tamanho de bloco distintos, no caso 16 e 32, a vantagem das caches com blocos de 32bytes é considerável, com redução próxima de 45% a 50% na taxa de misses na cache de instrucões e de 20% a 25% na taxa de misses na cache de dados. Isto mostra que a localidade temporal tem grande influencia em uma boa performance da cache.

Analisando a performance adquirida com o aumento do tamanho da cache para as associatividades 1,2 e 4 percebemos que em todas associatividades, o aumento do tamanho da cache diminui a taxa de misses na cache(óbvio!). O interessante é que existe uma diferenca significativa da associatividade 1 para as associatividades 2 e 4 com relacão a cache de dados. Enquanto a cache de instrucões tem uma diminuicão em torno de 40% na taxa de misses, a cache de dados apresenta uma diminuicão de aproximadamente 97%

4. Se um cache 2-way associativa possui atraso 20% maior que a de mapeamento direto, qual das duas deveria ser escolhida para compor um processador, assumindo que a CPU pode executar tão rápido quanto se queira, e a penalidade de um miss é de 20 vezes o atraso da cache de mapeamento direto.

Analisando a primeira execucao no simulador do benchmark AMP, temos aproximadamente 3% mais misses no mapeamento direto em relacão ao 2-way na cache de dados. Na cache de instrucão a diferenca é um pouco menor. Baseado nos dados gerados por esta simulacao (arquivos txt gerados no benchmark) e nos dados fornecidos no problema, temos que:

Com 4000000 de instrucoes, no mapeamento direto:

Misses no cache de instrucoes: 9548670

Misses no cache de dados: 874849

Já no 2-way:

Misses no cache de instrucoes: 9699786

Misses no cache de dados: 605980

Então:

Atraso na cache com mapeamento direto:

Atraso do cache(HITs): 1*(40000000-9548670) + 1*(40000000-874849) + = 69.576.481

Penalidade Misses: 20*(9548670) + 20*(874849) = 208.470.380

Total: 278.046.861

Atraso na cache com mapeamento 2-way:

Atraso do cache(HITs): 1.2*(40000000-9699786)+1.2*(40000000-605980) = 83.633.080

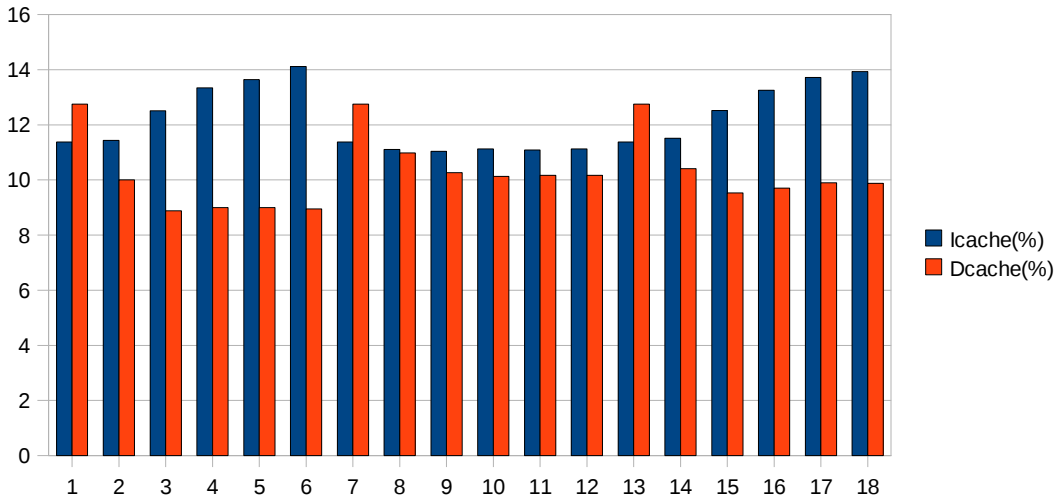
Penalidade Misses: 20*(9699786)+20*(605980)= 206.115.320

Total: 289.748.400

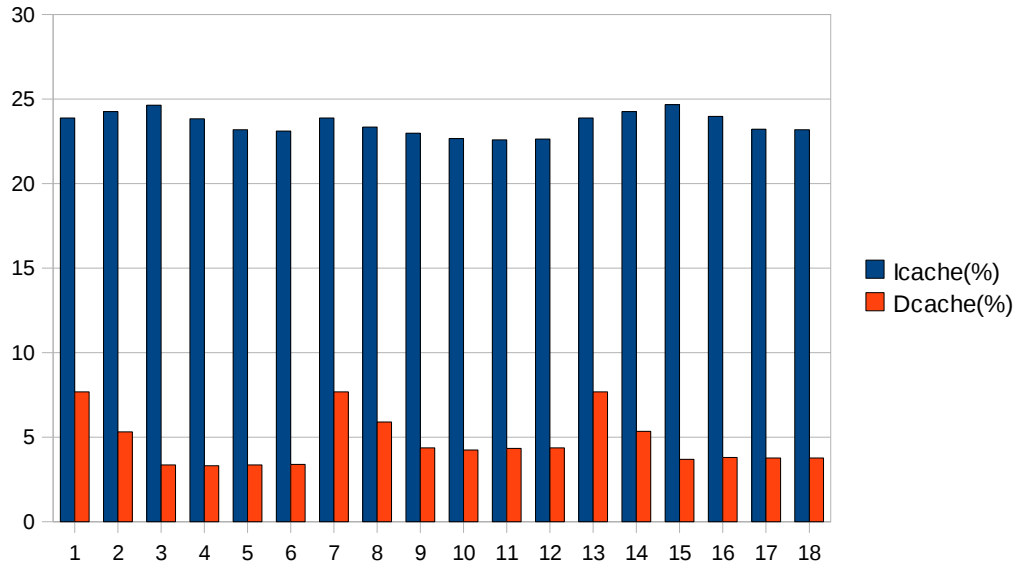
Estes cálculos ajudam a mostrar o quão próximos de performance são as duas escolhas. De qualquer forma, baseado nos cálculos desta simulacão, o mapeamento direto ainda tem melhor performance do que a escolha pelo 2-way nesta aplicacão. É bem possível que gerando outros benchmarks possamos ter resultados diferentes dos encontrados.

Dados da simulacão1:

AMP					
N_Conj	Tam_Bloco	Assoc	Politica	Icache(%)	Dcache(%)
32	32	1	l	11.38	12.75
16	32	2	l	11.43	10
8	32	4	l	12.51	8.88
4	32	8	l	13.34	9
2	32	16	l	13.64	9
1	32	32	l	14.11	8.95
32	32	1	r	11.38	12.75
16	32	2	r	11.11	10.98
8	32	4	r	11.04	10.26
4	32	8	r	11.12	10.13
2	32	16	r	11.09	10.17
1	32	32	r	11.12	10.17
32	32	1	f	11.38	12.75
16	32	2	f	11.51	10.41
8	32	4	f	12.52	9.53
4	32	8	f	13.25	9.7
2	32	16	f	13.72	9.9
1	32	32	f	13.93	9.88
Basicmath					
N_Conj	Tam_Bloco	Assoc	Politica	Icache(%)	Dcache(%)
32	32	1	l	23.87	7.68
16	32	2	l	24.25	5.32
8	32	4	l	24.63	3.35
4	32	8	l	23.83	3.31
2	32	16	l	23.19	3.35
1	32	32	l	23.11	3.39
32	32	1	r	23.87	7.68
16	32	2	r	23.33	5.89
8	32	4	r	22.97	4.37
4	32	8	r	22.66	4.23
2	32	16	r	22.58	4.33
1	32	32	r	22.63	4.36
32	32	1	f	23.87	7.68
16	32	2	f	24.25	5.34
8	32	4	f	24.66	3.69
4	32	8	f	23.97	3.8
2	32	16	f	23.22	3.76
1	32	32	f	23.19	3.76



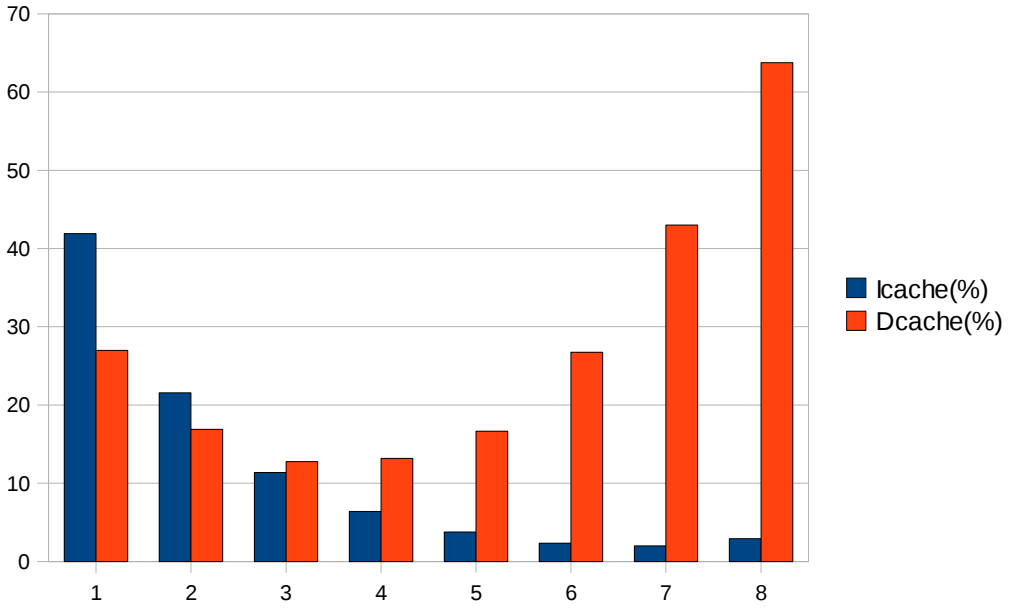
AMP



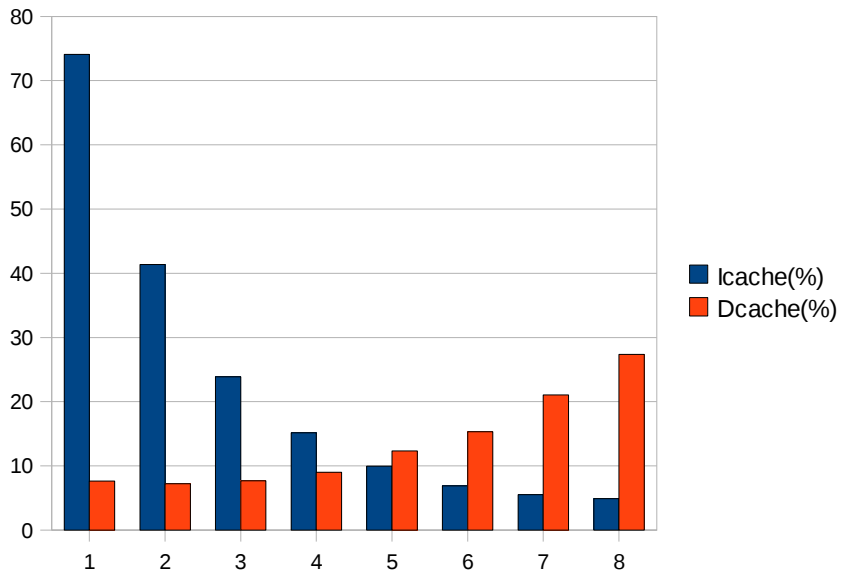
BASICMATH

Dados da simulacão2:

AMP					
N_Conj	Tam_Bloco	Assoc	Politica	Icache(%)	Dcache(%)
128	8	1	I	41.9	26.98
64	16	1	I	21.56	16.88
32	32	1	I	11.38	12.75
16	64	1	I	6.37	13.18
8	128	1	I	3.76	16.64
4	256	1	I	2.31	26.72
2	512	1	I	2	43
1	1024	1	I	2.89	63.76
BasicMath					
N_Conj	Tam_Bloco	Assoc	Politica	Icache(%)	Dcache(%)
128	8	1	I	74.06	7.64
64	16	1	I	41.37	7.23
32	32	1	I	23.87	7.68
16	64	1	I	15.16	9.02
8	128	1	I	9.99	12.33
4	256	1	I	6.9	15.33
2	512	1	I	5.54	21.04
1	1024	1	I	4.89	27.37



AMP



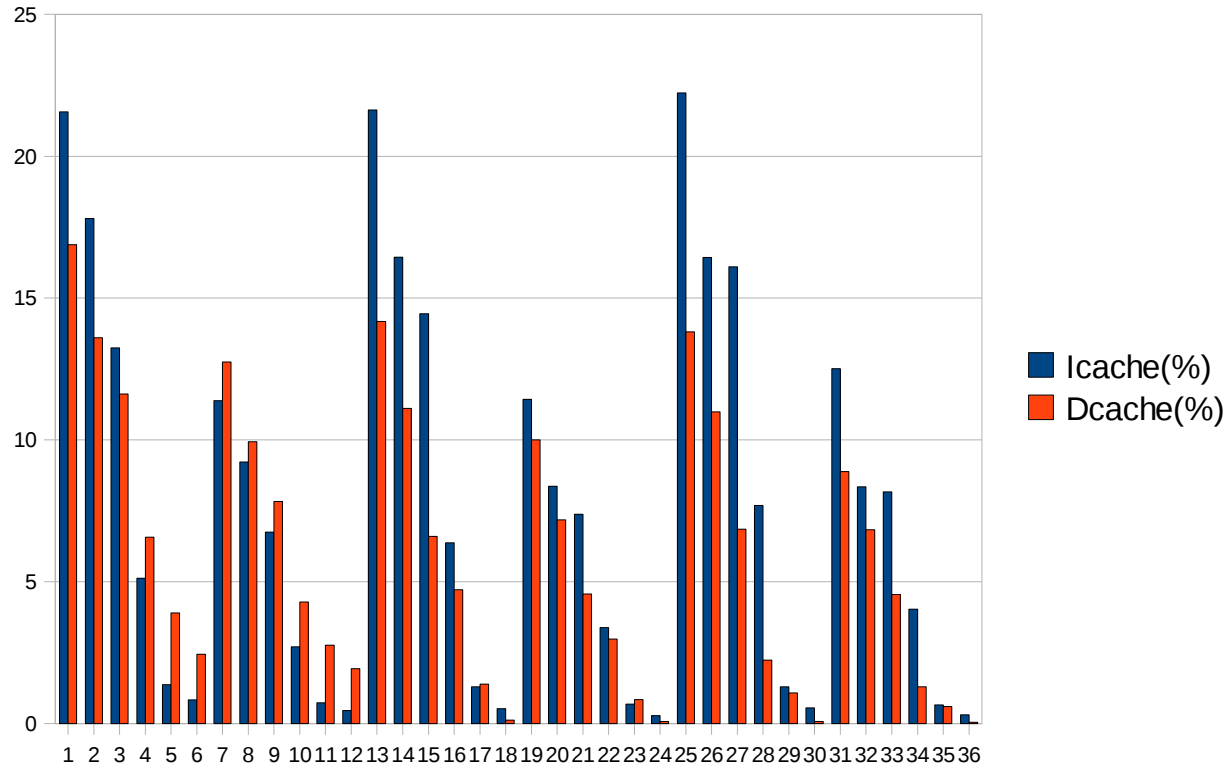
BASICMATH

Dados da simulacão3:

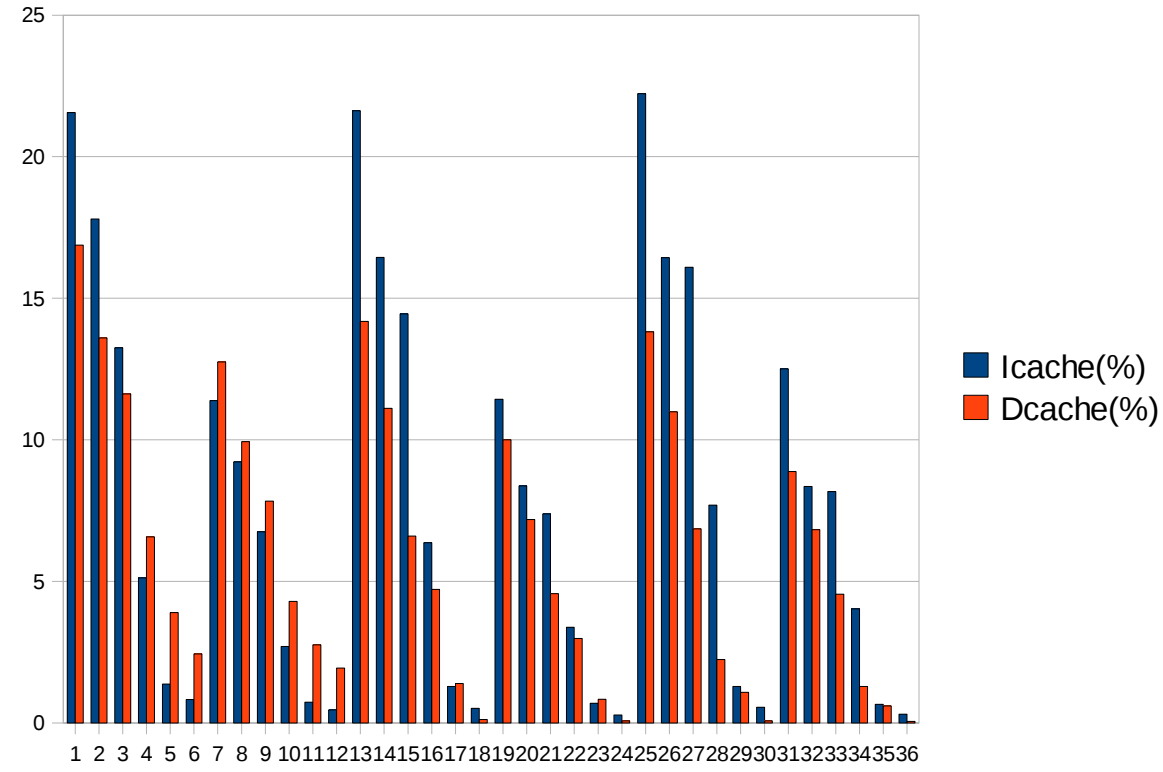
N_Conj	Tam_Bloco	Assoc	AMP Politica	Icache(%)	Dcache(%)
64	16	1	l	21.56	16.88
128	16	1	l	17.8	13.6
256	16	1	l	13.25	11.62
512	16	1	l	5.12	6.57
1024	16	1	l	1.37	3.9
2048	16	1	l	**0.83**	**2.44**
32	32	1	l	11.38	12.75
64	32	1	l	9.22	9.94
128	32	1	l	6.75	7.83
256	32	1	l	2.7	4.29
512	32	1	l	0.73	2.76
1024	32	1	l	**0.46**	**1.94**
32	16	2	l	21.63	14.18
64	16	2	l	16.44	11.11
128	16	2	l	14.45	6.6
256	16	2	l	6.37	4.72
512	16	2	l	1.29	1.39
1024	16	2	l	**0.52**	**0.12**
16	32	2	l	11.43	10
32	32	2	l	8.37	7.18
64	32	2	l	7.38	4.57
128	32	2	l	3.38	2.98
256	32	2	l	0.69	0.84
512	32	2	l	**0.28**	**0.07**
16	16	4	l	22.23	13.81
32	16	4	l	16.43	10.99
64	16	4	l	16.1	6.85
128	16	4	l	7.69	2.24
256	16	4	l	1.29	1.08
512	16	4	l	**0.55**	**0.07**
8	32	4	l	12.51	8.88
16	32	4	l	8.35	6.83
32	32	4	l	8.17	4.55
64	32	4	l	4.03	1.29
128	32	4	l	0.66	0.6
256	32	4	l	**0.31**	**0.05**

BasicMath

N_Conj	Tam_Bloco	Assoc	Politica	Icache(%)	Dcache(%)
64	16	1	I	41.37	7.23
128	16	1	I	34.23	4.87
256	16	1	I	27.98	2.97
512	16	1	I	22.39	2.56
1024	16	1	I	15.99	0
2048	16	1	I	6.39	0
32	32	1	I	23.87	7.68
64	32	1	I	20.01	4.77
128	32	1	I	16.49	3.22
256	32	1	I	13.19	2.64
512	32	1	I	9.69	0
1024	32	1	I	4.03	0
32	16	2	I	40.54	4.49
64	16	2	I	33.58	1.87
128	16	2	I	25.37	0.37
256	16	2	I	17.9	0.14
512	16	2	I	11.72	0
1024	16	2	I	4.26	0
16	32	2	I	24.25	5.32
32	32	2	I	19.82	1.89
64	32	2	I	15.09	0.53
128	32	2	I	10.43	0.14
256	32	2	I	7.22	0
512	32	2	I	2.84	0
16	16	4	I	39.4	3.89
32	16	4	I	31.98	1.28
64	16	4	I	25.49	0.18
128	16	4	I	16.99	0.01
256	16	4	I	11.92	0
512	16	4	I	2.24	0
8	32	4	I	24.63	3.35
16	32	4	I	18.74	1.39
32	32	4	I	14.93	0.49
64	32	4	I	9.72	0.01
128	32	4	I	7.38	0
256	32	4	I	1.89	0



AMP



BASICMATH

Anexo1

Conforme dica2 da definicão do trabalho, segue o script e os arquivos com informacões utilizados para coletar os dados utilizados neste trabalho:

Trabalho2.sh:

```
#!/bin/bash
HOJEPRECISO=`date "+%d%m%y%H%M%S"`

mkdir bench/$HOJEPRECISO

DIRTRAB=bench/$HOJEPRECISO

TABELA=$DIRTRAB/trab2.csv



while IFS=, read bench conj tam assoc repl

do

    if [ "$bench" == "^" ];

    then

        simulacao=$conj-$tam-$assoc-$repl

        ./sim-cache -max:inst 40000000 -redir:sim $DIRTRAB/$simulacao.txt \

        -cache:il1 il1:$conj:$tam:$assoc:$repl -cache:il2 none \

        -cache:dl1 dl1:$conj:$tam:$assoc:$repl -cache:dl2 none \

        bench/$benchmark.ss -funroll-loops -fforce-mem -fce-follow-jumps -O bench/cccp.i -o bench/cccpi.s

        Icache=`cat $DIRTRAB/$simulacao.txt |grep il1.miss_rate |tr -s " *" " " |cut -d' ' -f2`

        Dcache=`cat $DIRTRAB/$simulacao.txt |grep dl1.miss_rate |tr -s " *" " " |cut -d' ' -f2`

        RateIcache=`echo "$Icache * 100" |bc -l`

        RateDcache=`echo "$Dcache * 100" |bc -l`

        echo ";$conj;$tam;$assoc;$repl;$RateIcache;$RateDcache" >> $TABELA

    else

        echo "" $TABELA

        echo "$bench;N_Conj;Tam_Bloco;Assoc;Politica;Icache(%);Dcache(%)" >> $TABELA

        benchmark=$bench

    fi
```

done <dados3-trab2.txt

#done <dados2-trab2.txt

#done <dados1-trab2.txt

dados1-trab2.txt:

amp,,,

^,32,32,1,l

^,16,32,2,l

^,8,32,4,l

^,4,32,8,l

^,2,32,16,l

^,1,32,32,l

^,32,32,1,r

^,16,32,2,r

^,8,32,4,r

^,4,32,8,r

^,2,32,16,r

^,1,32,32,r

^,32,32,1,f

^,16,32,2,f

^,8,32,4,f

^,4,32,8,f

^,2,32,16,f

^,1,32,32,f

basicmath,,,

^,32,32,1,l

^,16,32,2,l

^,8,32,4,l

^,4,32,8,l

^,2,32,16,l

^,1,32,32,l

^,32,32,1,r

^,16,32,2,r

^,8,32,4,r

^,4,32,8,r

^,2,32,16,r

^,1,32,32,r

^,32,32,1,f

^,16,32,2,f

^,8,32,4,f

^,4,32,8,f

^,2,32,16,f

^,1,32,32,f

dados2-trab2.txt:
amp,,,

^,128,8,1,l

^,64,16,1,l

^,32,32,1,l

^,16,64,1,l

^,8,128,1,l

^,4,256,1,l

^,2,512,1,l

^,1,1024,1,l

basicmath,,,

^,128,8,1,l

^,64,16,1,l

^,32,32,1,l

^,16,64,1,l

^,8,128,1,l

^,4,256,1,l

^,2,512,1,l

^,1,1024,1,l

dados2-trab2.txt:

amp,,,

^,64,16,1,l

^,128,16,1,l

^,256,16,1,l

^,512,16,1,l

^,1024,16,1,l

^,2048,16,1,l

^,32,32,1,l

^,64,32,1,l

^,128,32,1,l

^,256,32,1,l

^,512,32,1,l

^,1024,32,1,l

^,32,16,2,l

^,64,16,2,l

^,128,16,2,l

^,256,16,2,l

^,512,16,2,l

^,1024,16,2,l

^,16,32,2,l

^,32,32,2,l

^,64,32,2,l

^,128,32,2,l

^,256,32,2,l

^,512,32,2,l

^,16,16,4,l

^,32,16,4,l

^,64,16,4,l

^,128,16,4,l

^,256,16,4,l

^,512,16,4,l

^,8,32,4,l

^,16,32,4,l

^,32,32,4,l

^,64,32,4,l

^,128,32,4,l

^,256,32,4,l

basicmath,,,

^,64,16,1,l

^,128,16,1,l

^,256,16,1,l

^,512,16,1,l

^,1024,16,1,l

^,2048,16,1,l

^,32,32,1,l

^,64,32,1,l

^,128,32,1,l

^,256,32,1,l

^,512,32,1,l

^,1024,32,1,l

^,32,16,2,l

^,64,16,2,l

^,128,16,2,l

^,256,16,2,l

^,512,16,2,l

^,1024,16,2,l

^,16,32,2,l

^,32,32,2,l

^,64,32,2,l

^,128,32,2,l

^,256,32,2,l

^,512,32,2,l

^,16,16,4,l

^,32,16,4,l

^,64,16,4,l

^,128,16,4,l

^,256,16,4,l

^,512,16,4,l

^,8,32,4,l

^,16,32,4,l

^,32,32,4,l

^,64,32,4,l

^,128,32,4,l

^,256,32,4,l