ELEC3441 Hw2_gp- 32 Bit version

Bao Xinyang 3035952989

Jiang Feiyu 3035770800

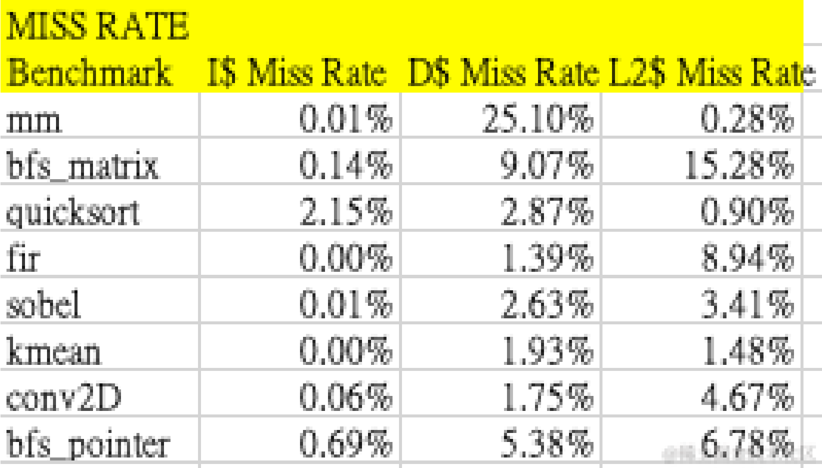
This is the riscV-32 version, which all the program is run by modifying makefile:
`CC = riscv32-unknown-elf-gcc`
And then `make clean`, `make`

```
spike --ic=128:1:32 --dc=256:1:32 --l2=1024:2:128 pk {folder}
```

B1.4

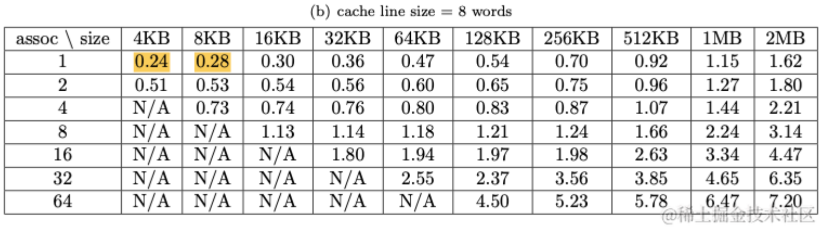
Tabels generated with `./src/auto.py`, the raw data can refer to `./src/result.csv` and `./src/result_32.xlsx`.

1. `kmean` has the best performance with `L1 I$`, `fir` has the best performance with `L1 D$`, `mm` has the best performance with `L2$`, but the worst performance with `L1 D$`. `quicksort` has the worst performance with `L1 I$`, `bfs` matrix has the worst performance with `L2$`.



MISS RATE			
Benchmark	I$ Miss Rate	D$ Miss Rate	L2$ Miss Rate
mm	0.01%	25.10%	0.28%
bfs_matrix	0.14%	9.07%	15.28%
quicksort	2.15%	2.87%	0.90%
fir	0.00%	1.39%	8.94%
sobel	0.01%	2.63%	3.41%
kmean	0.00%	1.93%	1.48%
conv2D	0.06%	1.75%	4.67%
bfs_pointer	0.69%	5.38%	6.78%

2. `L1 I$` has a line size of $32bytes/4 = 8words$, and capasity of $128 \times 1 \times 32 = 4KiB$. The cache access time is 240ps.
   `L1 D$` has a line size $32bytes/4 = 8words$, and capasity of $256 \times 1 \times 32 = 8KiB$. The cache access time is 280ps.
   `L2$` has a line size of $128bytes/4 = 32words$, this configuration is not included in table `B.1`.

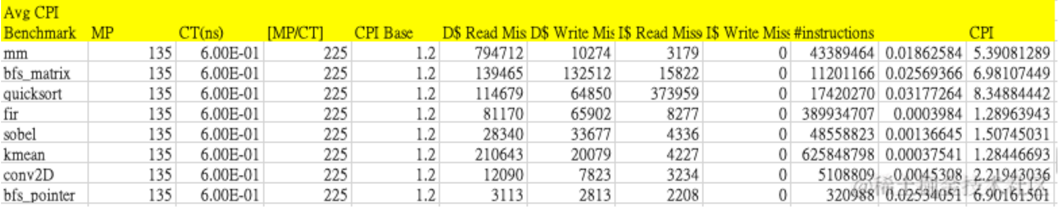


(b) cache line size = 8 words

assoc \ size	4KB	8KB	16KB	32KB	64KB	128KB	256KB	512KB	1MB	2MB
1	0.24	0.28	0.30	0.36	0.47	0.54	0.70	0.92	1.15	1.62
2	0.51	0.53	0.54	0.56	0.60	0.65	0.75	0.96	1.27	1.80
4	N/A	0.73	0.74	0.76	0.80	0.83	0.87	1.07	1.44	2.21
8	N/A	N/A	1.13	1.14	1.18	1.21	1.24	1.66	2.24	3.14
16	N/A	N/A	N/A	1.80	1.94	1.97	1.98	2.63	3.34	4.47
32	N/A	N/A	N/A	N/A	2.55	2.37	3.56	3.85	4.65	6.35
64	N/A	N/A	N/A	N/A	N/A	4.50	5.23	5.78	6.47	7.20

3. For the processor without `L2`, the cache access time is less than 600ps, so the cycle time is 600ps. For the processor with `L2`, the cache access time of `L2` is more than 600ps, so the cycle time is the cache access time of `L2`.

4. For MP, we assume it's $100 + 7 \times 5 = 135$, as there are 8 words in one line; For CT, we assume it's 600ps, discussed in B.1.4.3. The general formula is: $CPI = 1.2 + \frac{D\ read\ miss + D\ write\ miss + I\ read\ miss}{I\ read\ accesses\ No.} \times \frac{135}{0.6}$

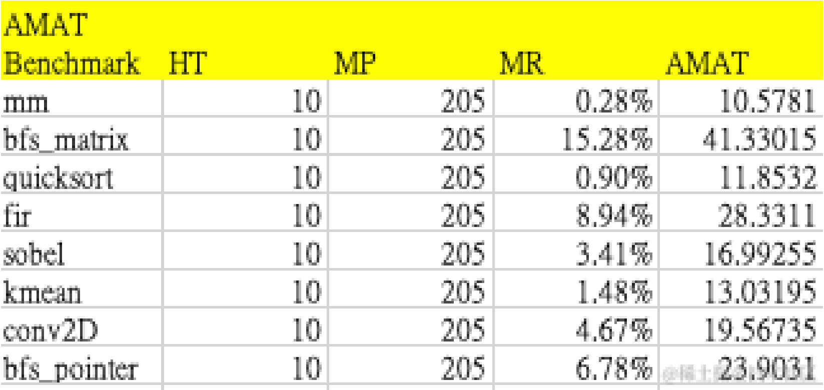


Avg CPI											
Benchmark	MP	CT(ns)	[MP/CT]	CPI Base	D$ Read Mis	D$ Write Mis	I$ Read Miss	I$ Write Miss	#instructions		CPI
mm	135	6.00E-01	225	1.2	794712	10274	3179	0	43389464	0.01862584	5.39081289
bfs_matrix	135	6.00E-01	225	1.2	139465	132512	15822	0	11201166	0.02569366	6.98107449
quicksort	135	6.00E-01	225	1.2	114679	64850	373959	0	17420270	0.03177264	8.34884442
fir	135	6.00E-01	225	1.2	81170	65902	8277	0	389934707	0.0003984	1.28963943
sobel	135	6.00E-01	225	1.2	28340	33677	4336	0	48558823	0.00136645	1.50745031
kmean	135	6.00E-01	225	1.2	210643	20079	4227	0	625848798	0.00037541	1.28446693
conv2D	135	6.00E-01	225	1.2	12090	7823	3234	0	5108809	0.0045308	2.21943036
bfs_pointer	135	6.00E-01	225	1.2	3113	2813	2208	0	320988	0.02534051	6.90161501

Benchmark	CPI
mm	5.39081289
bfs_matrix	6.981074488
quicksort	8.348844421
fir	1.28963943
sobel	1.507450306
kmean	1.284466927
conv2D	2.21943036
bfs_pointer	6.901615014

The average CPI is 4.24041673.

5. According to the formula, $AMAT_{L2} = HT_{L2} + MR_{L2} \times MP_{L2}$, we assume the $MP_{L12} = 100 + 7 * 15 = 205, HT_{L2} = 10$, thus:

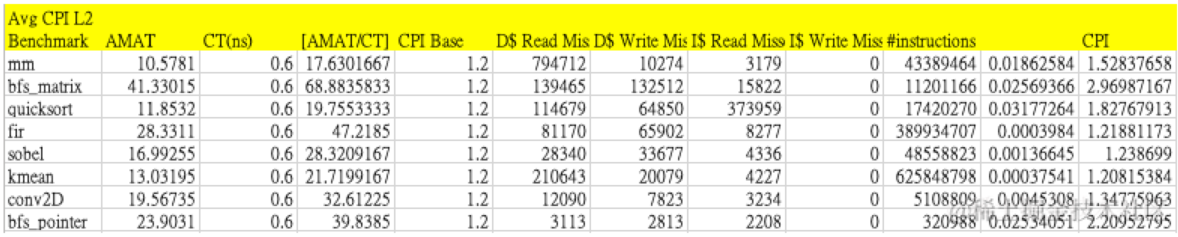


AMAT				
Benchmark	HT	MP	MR	AMAT
mm	10	205	0.28%	10.5781
bfs_matrix	10	205	15.28%	41.33015
quicksort	10	205	0.90%	11.8532
fir	10	205	8.94%	28.3311
sobel	10	205	3.41%	16.99255
kmean	10	205	1.48%	13.03195
conv2D	10	205	4.67%	19.56735
bfs_pointer	10	205	6.78%	23.9031

6. Calculate the average CPI for the benchmarks with L2 cache：

$$CPI = 1.2 + \frac{D\ read\ miss + D\ write\ miss + I\ read\ miss}{I\ read\ accesses\ No.} \times \frac{AMAT_{L2}}{0.6}$$

Since there is no Cache access time table for L2, we assume the CT still equals to 600ps.



Avg CPI L2											
Benchmark	AMAT	CT(ns)	[AMAT/CT]	CPI Base	D$ Read Mis	D$ Write Mis	I$ Read Miss	I$ Write Miss	#instructions		CPI
mm	10.5781	0.6	17.6301667	1.2	794712	10274	3179	0	43389464	0.01862584	1.52837658
bfs_matrix	41.33015	0.6	68.8835833	1.2	139465	132512	15822	0	11201166	0.02569366	2.96987167
quicksort	11.8532	0.6	19.7553333	1.2	114679	64850	373959	0	17420270	0.03177264	1.82767913
fir	28.3311	0.6	47.2185	1.2	81170	65902	8277	0	389934707	0.0003984	1.21881173
sobel	16.99255	0.6	28.3209167	1.2	28340	33677	4336	0	48558823	0.00136645	1.238699
kmean	13.03195	0.6	21.7199167	1.2	210643	20079	4227	0	625848798	0.00037541	1.20815384
conv2D	19.56735	0.6	32.61225	1.2	12090	7823	3234	0	5108809	0.0045308	1.34775963
bfs_pointer	23.9031	0.6	39.8385	1.2	3113	2813	2208	0	320988	0.02534051	2.20952795

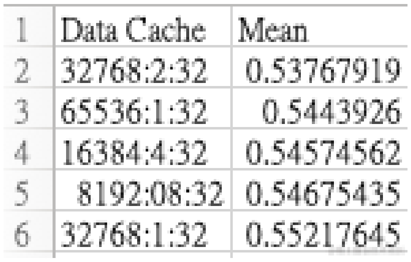
Benchmark	CPI
mm	1.528376576
bfs_matrix	2.969871672
quicksort	1.827679131
fir	1.218811731
sobel	1.238698998
kmean	1.208153843
conv2D	1.347759635
bfs_pointer	2.209527954

The average CPI is 1.693609943.

7. `L2$` contributes positively to system performance by improving cache hit rate, reducing miss penalty, and increasing data and instruction availability. Therefore, it can be said that `L2$` helps enhance performance.

B1.5

Full code avaliable at `./src/5.sh` and `./src/analysis.ipynb`, full data avaliable at `./src/dc.csv` and `./src_32/ranking.xlsx`.



1	Data Cache	Mean
2	32768:2:32	0.53767919
3	65536:1:32	0.5443926
4	16384:4:32	0.54574562
5	8192:08:32	0.54675435
6	32768:1:32	0.55217645

The top configurations is:

Cache Configuration	Mean Miss Rate
32768:2:32	0.537679192%

which is one of the is the optimal cache configuration for the giving benchmarks.