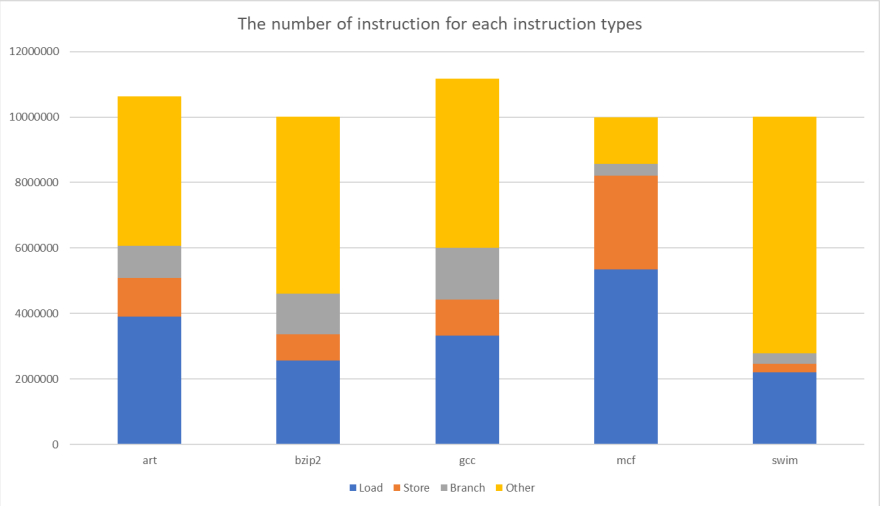
Project1 Report

20190439 이규빈

Instruction의 종류

	art	bzip2	gcc	mcf	swim
Load	3912504	2563875	3317963	5354259	2204476
Store	1165329	804499	1116048	2855249	251924
Branch	993637	1240759	1577732	358908	337248
Other	4554841	5408083	5170460	1432866	7223356
Total	10626311	10017216	11182203	10001282	10017004

[표1] Benchmark에 따른 instruction 개수



[그림1] Benchmark에 따른 instruction 개수

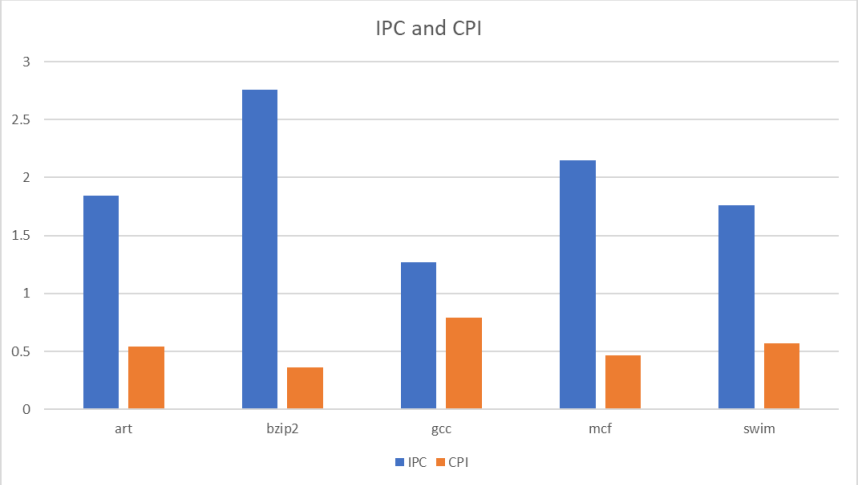
각 benchmark의 load, store, branch, 그리고 그 외 타입의 instruction이 몇 회 실행되었는지 기록하고 그래프로 나타내 보았다. 이를 통해 아래와 같은 사실들을 알 수 있었다.

- 총 instruction의 개수는 거의 비슷하나 gcc가 다른 benchmark에 비해 조금 많다.
- mcf의 경우 전체 instruction에서 load/store instruction이 차지하는 비율이 상대적으로 높다.
- swim의 경우 전체 instruction에서 load/store instruction이 차지하는 비율이 상대적으로 낮은데, 특히 store instruction의 수가 상당히 작다.
- art, bzip2, gcc에 비해 mcf와 swim은 branch instruction이 차지하는 비율이 낮다. 이는 한 branch당 instruction의 수가 mcf와 swim이 더 크다는 것을 의미한다.

IPC와 CPI

	art	bzip2	gcc	mcf	swim
IPC	1.8419	2.7567	1.2697	2.1458	1.7582
CPI	0.5429	0.3628	0.7876	0.466	0.5688

[표2] 각 benchmark들의 IPC와 CPI



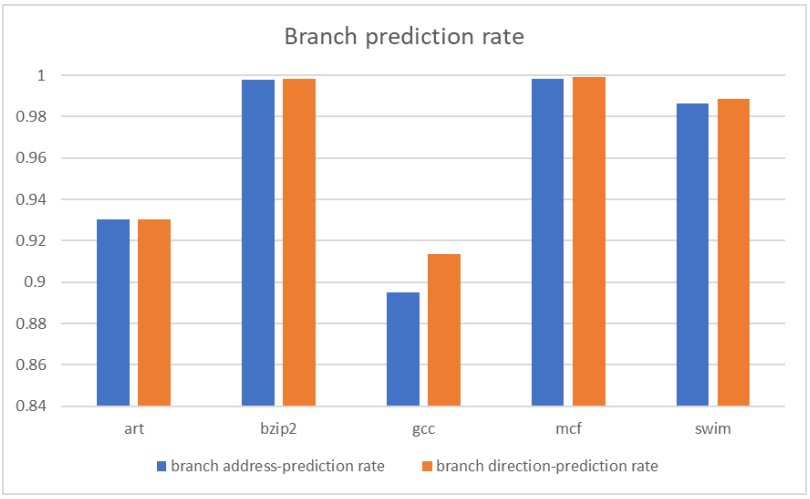
[그림2] 각 benchmark들의 IPC와 CPI

각 benchmark들의 IPC와 CPI를 기록하고 그래프로 나타내 보았다. 기본적으로 IPC와 CPI는 역수 관계이기 때문에 IPC가 높으면 CPI는 낮게 된다. 모든 benchmark에 대해 IPC가 1보다 크기 때문에 모두 pipelining같은 parallel한 instruction의 실행이 일어남을 알 수 있다. IPC가 큰 순서대로 나열하면 bzip2 > mcf > art > swim > gcc인 것으로 나타났다. IPC가 클 경우 (clock cycle time이 모두 같을 때) 한 cycle안에 실행될 수 있는 instruction의 수가 더 많기 때문에 bzip2 > mcf > art > swim > gcc순으로 처리 속도가 빠르다고 할 수 있다.

Branch Prediction Rate

	art	bzip2	gcc	mcf	swim
branch address-prediction rate	0.9302	0.9978	0.8949	0.9984	0.9863
branch direction-prediction rate	0.9304	0.998	0.9136	0.9991	0.9883

[표3] 각 benchmark들의 branch prediction rate



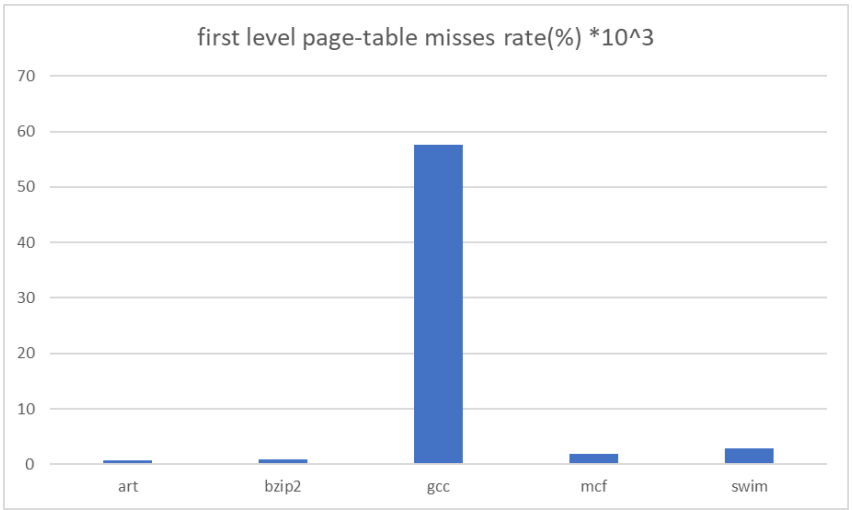
[그림3] 각 benchmark들의 branch prediction rate

각 benchmark들의 branch prediction rate를 기록하고 그래프로 나타내 보았다. bzip2, mcf, swim의 경우 branch prediction rate가 0.99 수준으로 매우 높은 확률로 branch prediction에 성공하였지만 art와 gcc의 경우 branch prediction rate가 상대적으로 낮았다. 잘못된 branch prediction은 stall이 발생하는 원인이 되기 때문에 art와 gcc의 경우 낮은 branch prediction rate에 따른 성능 저하가 우려된다.

Page-table

	art	bzip2	gcc	mcf	swim
first level page-table misses rate(%) *10^3	0.645023	0.830634	57.62427	1.959488	2.89244709

[표4] 각 benchmark들의 fist level page-table misses rate(결과값에 10^3을 곱함)



[그림4] 각 benchmark들의 fist level page-table misses rate(결과값에 10^3을 곱함)

각 benchmark들의 branch prediction rate를 기록하고 그래프로 나타내 보았다. (결과값이 매우 작

았기 때문에 misses rate를 퍼센트로 나타낸 후 얻어진 값에 10^3을 곱하였다) 그 결과 gcc의 경우 다른 benchmark에 비해 misses rate가 상당히 높음을 알 수 있었다. Page table의 misses rate가 높을 경우 메모리에 접근하는 instruction(load/store 같은)이 많아지게 될수록 원하는 메모리 주소가 속한 적합한 page table을 찾는데 쓰는 시간이 많아지게 되기 때문에 성능이 저하될 수 있다.