UNIVERSITY OF FLORIDA

Assignment 6

Digital Computer Architecture

James Giandelone

HW Problems(blue book)

6.3.1 Calculate the average time to read or write a 1024-byte sector for each disk listed in the table.

a)average rotational latency = 1/(7200rpm*2/60) = 4.1666ms
disk transfer rate = 1024/(34*2^20) = .0287 ms
disk controller rate = 1024/(480/8*2^20) = .016 ms
average time = 11+4.1666+.0287+.016 = 15.2113 ms
b) average rotational latency = 4.1666ms
disk transfer rate = 1024/(30*2^20) = .03255 ms
disk controller rate = 1024/(500/8*2^20) = .0156 ms
average time = 9+4.1666+.03255+.0156 = 13.2 ms

6.3.2 Calculate the minimum time to read or write a 2048 byte sector for each disk listed in the table.

The minimum time to read will be when the head of the hard drive is already over the correct sector. Consequently the seek time and rotational latency will be 0. Thus the minimum time will only be the sum of the disk transfer rate, and the controller transfer rate/

a)disk transfer rate = 2048/(34*2^20) = .0577 ms
disk controller rate = 2048/(480/8*2^20) = .03255 ms
minimum time = .0577 + .03255 = .09025 ms
b)disk transfer rate = 2048/(30*2^20) = .0651 ms
disk controller rate = 2048/(500/8*2620) = .03125 ms
minimum time = .0651+.03125 = .09635 ms

6.15.1 Calculate the new parity P' for RAID 3.
a)FEFE XOR A387 XOR F345 XOR FF00 = 513C
b)AB9C XOR 0098 XOR 00FF XOR 2FFF = 8404

6.15.2 Calculate the new parity P' for RAID 4.
note: the tables initial parity value is incorrect which is why the final result of this problem does not match the parity from previous problem
a)FEFE XOR 00FF XOR 4582 = BB83
b)AB9C XOR F457 XOR A387 = FC4C

6.15.3 Is RAID 3 or RAID 4 more efficient? Are there reasons why RAID 3 would be preferable to RAID 4?
RAID 4 is more efficient since there are only 2 xor operations compared to RAID 3's 4 xor operations. To calculate parity for raid 3, you must make 4 disk accesses, while RAID 4 requires 3. RAID 3 has no advantages

6.15.4 RAID 4 and RAID 5 use roughly the same mechanism to calculate and store parity for data blocks. How does RAID 5 differ from RAID 4 and for what applications would RAID 5 be more efficient?

RAID 4 has a dedicated parity drive while RAID 5 has alternating drives holding parity. Consequently, there is no parity drive bottleneck in RAID 5 like there is in RAID 4. RAID 5 will be more efficient in all applications.

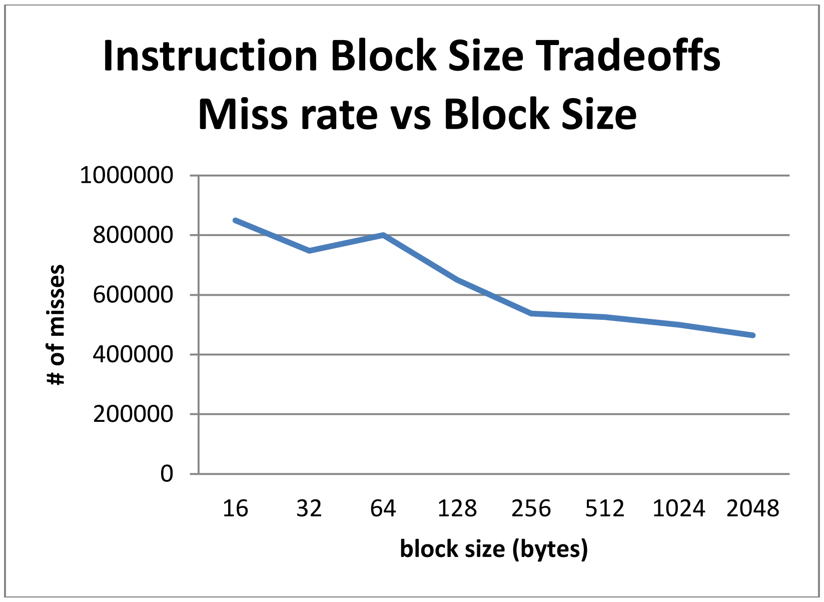
6.18.1 Calculate annual failure rate(AFR) for disks in the table.
a)8760*1000/1000000 = 8.76
b)10512*1000/1500000 = 7.008

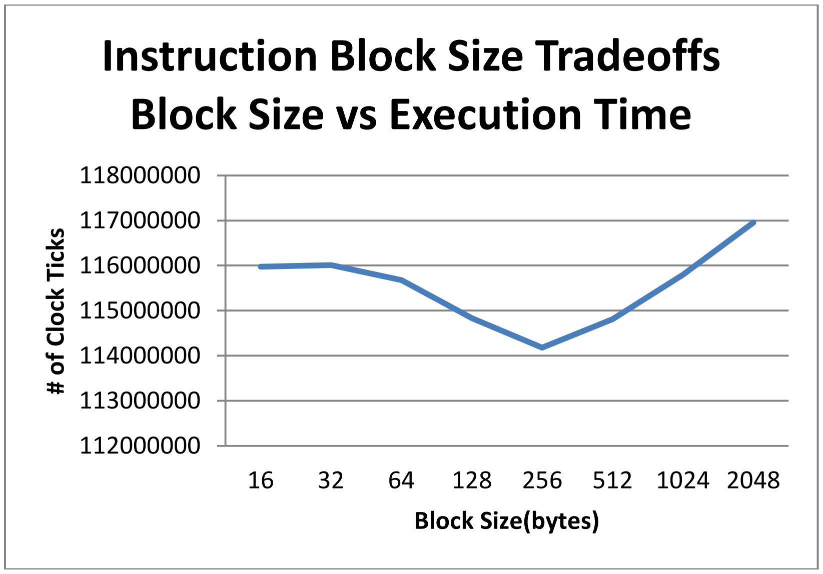
6.18.2
a)7 years = 8.76*3/12 + 8.76*11/12 + 8.76*3 + 8.76*2 + 8.76*4 = 142.35
 10 years = 142.35 + 8.76*8+8.76*16 = 459.905
b) 7 years = 7.008 * 3/12 + 7.008 * 11/12 + 7.008*2+7.008*4 = 50.224
 10 years = 50.224+ 7.008*8 + 7.008 * 16 = 218.416
--this is assuming failed drives are being replaced

Instruction Cache Block size Tradeoffs

Question: Plot two graphs: L1 I-cache miss rate (il1_miss_rate in the SESC simulation output) versus block size, and total execution time (sim_cycle) versus block size. Discuss your results with meaningful discussion (i.e. do not simply state what can be read off of the graphs themselves).

Instruction Block Size Tradeoffs					
Block Size	# of Hits	# of Misses	Clock Ticks	Access Time(ns)	Hit latency (cycles)
16	19673709	849376	115976785	.756	3
32	19532470	747424	116009735	.727	3
64	19468904	799960	115678369	.688	3
128	19759259	649771	114832219	.665	3
256	19947205	536890	114179783	.721	3
512	19981596	525871	114813257	.923	4
1024	20010611	499337	115797356	1.53	5
2048	20069377	464166	116953194	1.54	6





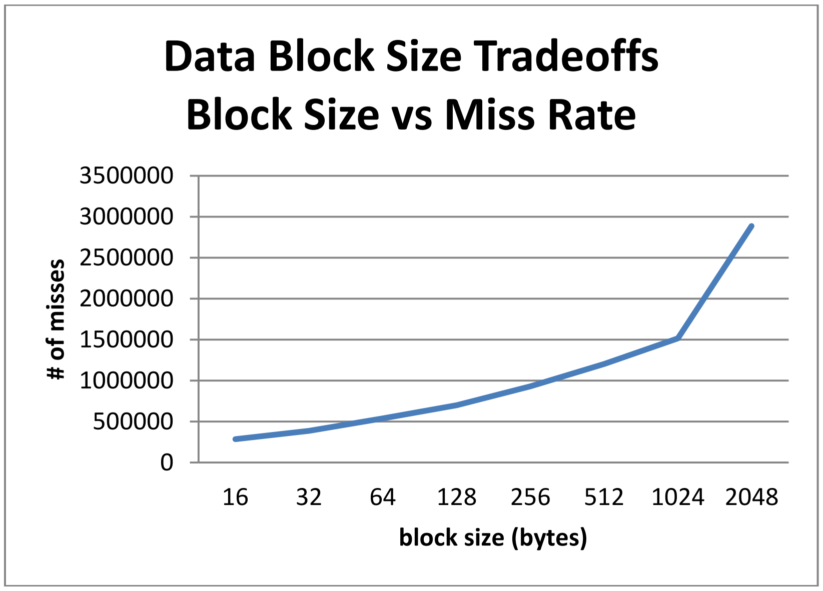
The miss rate graph shows that an increase in block size will decrease the number of misses. The block size determines the degree to which spatial locality is exploited. Spatial locality refers to the idea that if a memory location is accessed, there is a high likelihood that the memory around it will also be accessed. Thus, the increase in block size will cause more spatially close data to be stored at each cache block - decreasing the miss rate. On the other hand, if the block size were to be increased to far we would see an increase in the number of misses due to garbage data being loaded into cache and thus having wasted cache space.

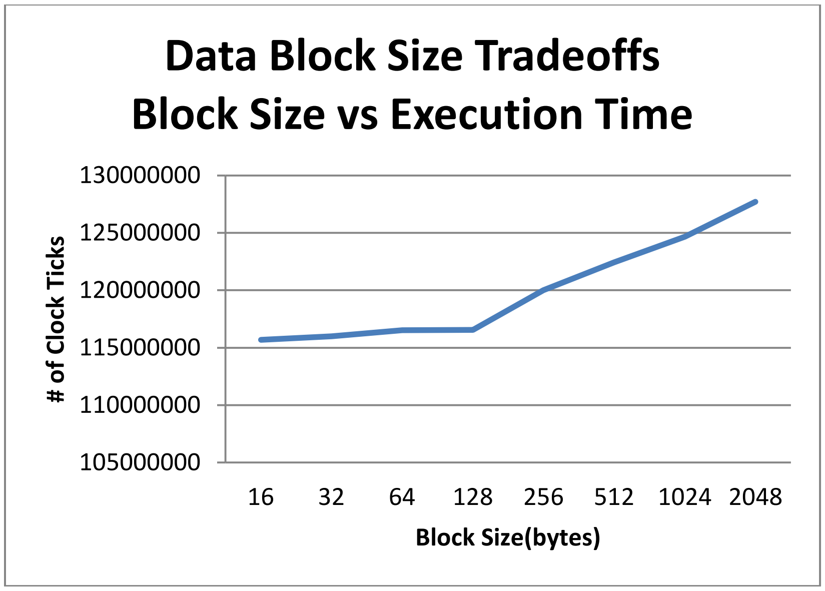
Read misses are highly detrimental to a processors performance. Consequently, a decrease in the number of read misses will cause an increase in the CPU performance - which can be seen in the execution time graph. However, we see that past a block size of 256 the execution time begins to increase. This is because the time required to perform a read from cache has increased by 1 cycle. Thus every successful read takes a longer period of time - which evidently has a much larger impact on performance than reducing cache misses.

Data Cache Block Size Tradeoffs

Question: Repeat part a), but now changing the block size of the L1 data cache. Plot the same two graphs. Discuss your results, comparing to the results obtained in a).

Data Block Size Tradeoffs					
Block Size	# of Hits	# of Misses	Clock Ticks	Access Time(ns)	Hit latency (cycles)
16	20009291	288258	115690561	.756	3
32	19841257	387301	116009735	.727	3
64	19603032	540267	116521899	.688	3
128	19369449	702012	116564691	.665	3
256	19017778	930365	120015775	.721	3
512	18553982	1201849	122445305	.923	4
1024	18090119	1516359	124673022	1.53	5
2048	15560865	2884645	127708197	1.54	6





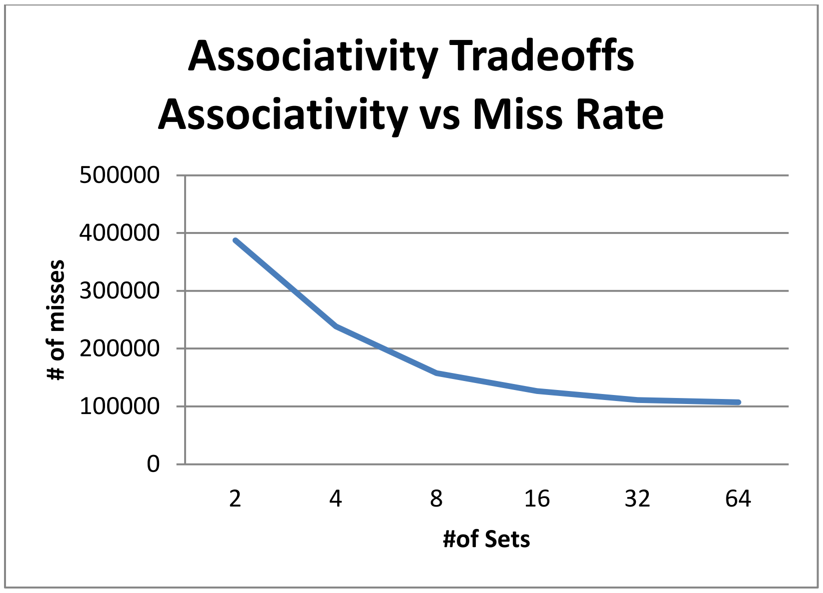
As opposed to instruction memory which makes large use of spatial locality, data memory needs to be more heavily dependent on temporal locality. Temporal locality is maximized by having a larger number of addressable cache blocks. Consequently, as you increase the block size (increases your use of spatial locality) you decrease the number of addressable cache blocks (decreasing your use of temporal locality). This is why the graph comparing miss rate to block size shows the miss rate increasing as the block size increases.

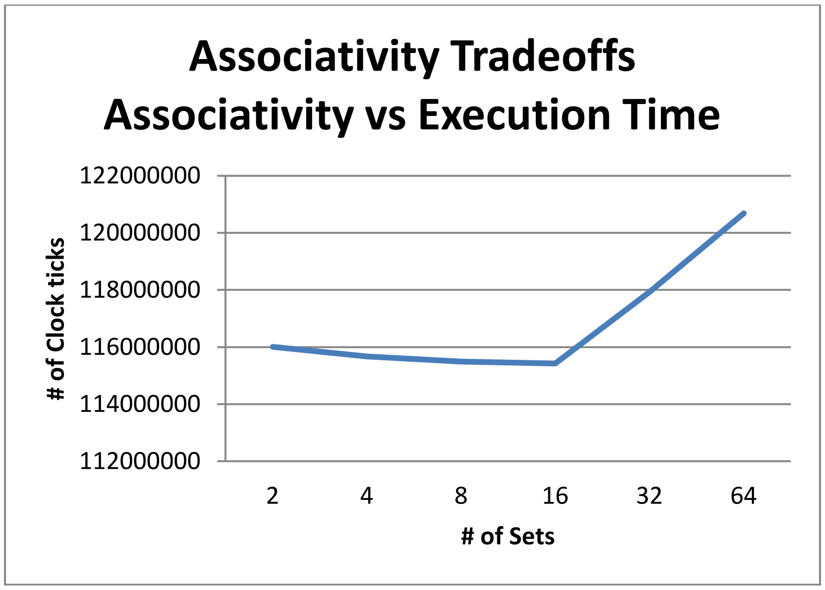
The execution time graph shows that the increased number of misses causes the execution time to increase. However, like before, the increased hit time has a larger effect on the overall performance.

Associativity Tradeoffs

Question: Plot two graphs: L1 D-cache miss rate versus associativity, and total execution time (clock cycles) versus cache associativity. Discuss your results.

Associativity Tradeoffs					
Associativity	# of Hits	# of Misses	Clock Ticks	Access Time(ns)	Hit latency (cycles)
2	19841257	387301	116009735	.728	3
4	20091047	238549	115673322	.715	3
8	20217131	157793	115488889	.728	3
16	20266011	126633	115425190	.833	3
32	20304887	110920	117929968	1.15	4
64	20326646	107341	120689438	1.53	5





Sets increase the number of locations that any particular address can be stored at. Consequently, as you increase the number of sets you will also decrease the miss rate - which can be seen in the Associativity vs. Miss rate graph. That said, increasing the set associativity will have diminishing returns as you approach full associativity because the number of contending memory locations in any program is fixed. Thus increasing associativity beyond that limit will have no effect.

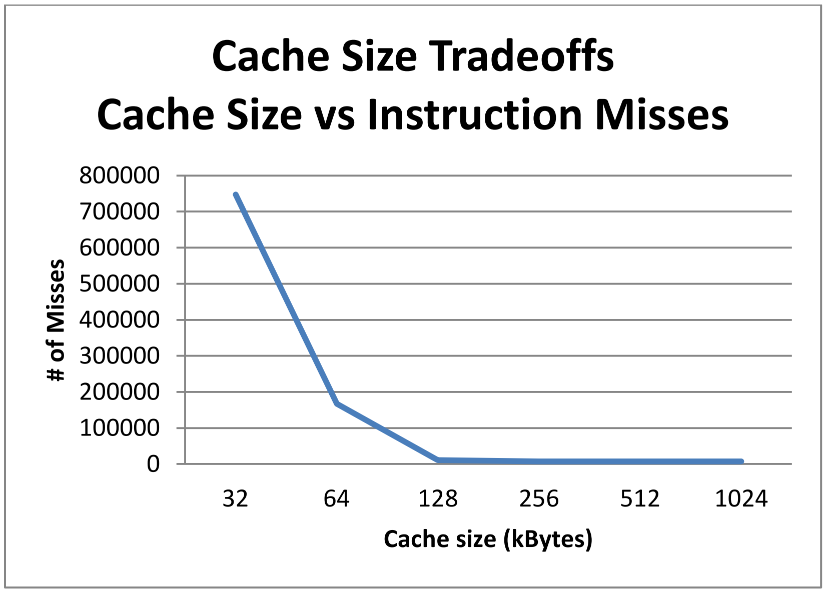
The execution time graph shows us that decreasing the miss rate will decrease the execution time (as before) so long as the hit time does not increase. When the hit time increases after 16 you see a sharp increase in execution time.

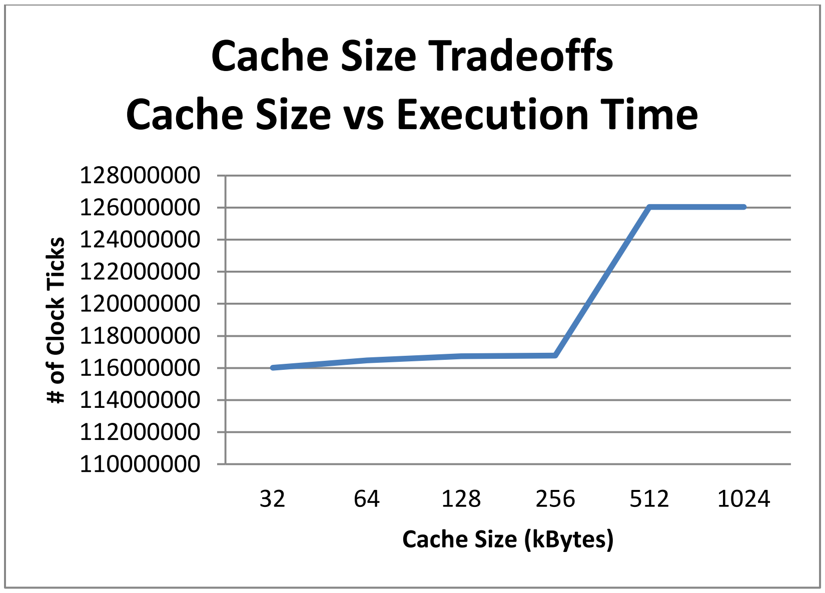
Cache Size Tradeoffs

Question: Plot two graphs: L1 I-cache miss rate versus associativity, and total execution time (clock cycles) versus cache size. Discuss your results.

Cache Size Tradeoffs

Cache Size (kByes)	# of Hits	# of Misses	Clock Ticks	Access Time(ns)	Hit latency (cycles)
32	19532470	747424	116009735	.727	3
64	20488381	167052	116485553	.813	3
128	20748775	10661	116743604	1.06	4
256	20753358	7792	116771227	1.18	4
512	20753476	7722	126034904	2.06	7
1024	20753482	7718	126044796	2.42	8





Increasing the cache size will decrease the miss rate with diminishing returns. Larger caches allow for more data to be stored in them, reducing the amount of replacement necessary. However, once the cache size is larger than the program's total memory usage, increase in cache will have no effect on performance. The cache size tradeoff shows this relation of diminishing returns.

The execution time graph shows that once the miss rate is not decreasing but the access time is, the execution time jumps greatly - which makes sense since the access time almost doubles at 512 however the miss rate does not fall at all.

Overall cache performance experiment

Options	Instruction Cache Block Size (bytes)	Data Cache Block Size (bytes)	Data Cache Associativity	Instruction cache Size (kBytes)
1	256	16	16	64
2	256	16	16	32
3	256	32	16	64

Options	# of Hits	# of Misses	Clock Ticks	Instruction Access Time(ns)	Data Cache Access Time(ns)	Hit latency for all caches (cycles)
1	20282617	102699	112554396	.749	.798	3
2	20297105	102678	113361659	.721	.798	3
3	20259332	126622	113582964	.749	.833	3

Option 1:

- Instruction Cache Block size: I chose 256 because it yielded the best performance when tested alone.
- Data Cache Block Size: I chose 16 because it yielded the best performance when tested alone.
- Associativity: I chose 16 because it yielded the best performance when tested alone.
- Instruction cache size: I ignored the best performance for this value, and instead chose the value with the fewest number of misses but kept hit latency of 3.

Option 2

- Instruction Cache Size: I changed this value from 64 to 32 because the graph for Instruction Cache Size shows that 32 has a slightly faster execution time.

Option 3

- Data Cache Block Size: I originally set this value at 128, but was forced to continuously lower it because the hit latency was too high. The idea behind increasing this, was to see if I a slight increase in spatial locality without affecting access time would increase performance. As expected though, it did not.

Comments On Choices

For each option, my highest priority was to keep the hit latency to 3 cycles. I did this because the previous simulations had made it very clear that an increase in hit latency would overpower any other advantage the system gained. For option 1, I chose what I felt to be the surefire correct answer - which turned out to be correct. I considered options 2 and 3 to be long shots, but they were still worth testing.

Best Option: Option 1