1.)

a.) 512k / 32 = 16k or 6.25% miss rate || It is proportional to the block size not the cache size || they are compulsory misses

b.) 512k / 16 = 32k or 12.5% miss rate || 512k / 64 = 8k or 3.125% || 512k / 128 = 4k or 1.56% || Spatial locality

c.) Almost zero percent because it will have to go through a buffer before checking if it is In the cache, so it allows for a less chance to miss.

2.)

a.)Avg memory access time is = (time for a hit) + (miss rate) * (latency)

Block Size	Miss Rate	Latency	Avg Access Time
8 Bytes	4%	160 cycles	(1)+(0.04)*(160) = 7.4
16 Bytes	3%	320 cycles	(1)+(0.03)*(320) = 10.6
32 Bytes	2%	640 cycles	(1)+(0.02)*(640) = 13.8
64 Bytes	1.5%	1280 cycles	(1)+(0.015)*(1280) = 20.2
128 Bytes	1%	2560 cycles	(1)+(0.01)*(2560) = 26.6

Best one will be 8 bytes for block size because it has the lowest Access time of 7.4

b.)Avg memory access time is = (time for a hit) + (miss rate) * (latency)

Block Size	Miss Rate	Latency	Avg Access Time
8 Bytes	4%	32 cycles	(1)+(0.04)*(32) = 2.28
16 Bytes	3%	40 cycles	(1)+(0.03)*(40) = 2.2
32 Bytes	2%	56 cycles	(1)+(0.02)*(56) = 2.12
64 Bytes	1.5%	88 cycles	(1)+(0.015)*(88) = 2.32
128 Bytes	1%	152 cycles	(1)+(0.01)*(152) = 2.52

Best one will be 32 bytes for block size because it has the lowest Access time of 2.12

c.)You always chose the cones with the less miss rate so the best one would be 128 bytes

3.)

a.) 1 / L1 hit time || P1 Clock Rate = 1 / 0.66 = 1.515 GHz || P2 Clock Rate = 1 / 0.9 = 1.11 GHz

b.) Average Memory Access Time = L1 Hit time + (L1 miss rate * Memory Access Time)
P1 Access Time = 0.66 + (0.08 * 70) = 6.26 ns
P2 Access Time = 0.9 + (0.06 * 70) = 5.1 ns

c.) Total CPI = Base CPI + (Memory Access Time * L1/L2 Miss Rate) / L1/L2 Hit Time) * # of Memory
P1 CPI = 1 + (70 * 0.08) / 0.66 * 0.36 = 4.055
P2 CPI = 1 + (70 * 0.06) / 0.9 * 0.36 = 2.68
P2 < P1 so P2 is faster

d.) Average Memory Access Time for P1 including L2 Cache =
(L1 hit time) + (L1 miss rate) * (L2 hit time + L2 miss rate * Memory Access Time) =
= 0.66 + 0.08 * (5.62 + 0.95 * 70) = 6.43 ns
Since P1 including L2 Cache > P1 && P2, this addition of L2 Cache makes it worse for the Average Memory Access Time.

e.) Total CPI = Base CPI + # of Memory Instructions * (L1 miss rate * (L2 Hit in Cycles + L2 Memory Miss in Cycles)) =
Base CPI + # of Memory Instructions * (L1 miss rate * ((L2 Hit in Cycles) +
(L2 Miss Rate * Memory Access Time) / L1 P1 Hit Time) =
= 1 + 0.36 * (0.08 * (5.62 + 0.95 * 70) / 0.66)) = 4.15

f.) P2 is faster if P2 is faster, it just stays the same so we don't have to find anything
If P1 is faster first solve P2 time:
P2 Time ~> Total CPI of P2 * L2 Hit time = 2.68 * 0.9 = 2.412 ns / instruction
Then we do:
L1 Hit Time * (Base CPI + Memory Access Time * L1 Miss Rate * 106)
= 0.66 * (1 + 0.36 * ? * 106) = 0.069 = 6.9%

4.)

a.)

Word Address	Binary Address	Offset	Tag	Index	Hit(H)/Miss(M)
3	0000 0011	1	00000	01	M
180	1011 0100	0	10110	10	M
43	0010 1011	1	00101	01	M
2	0000 0010	0	00000	01	H
191	1011 1111	1	10111	11	M

88	0101 1000	0	01011	00	M
190	1011 1110	0	10111	11	H
14	0000 1110	0	00001	11	M
181	1011 0101	1	10110	10	H
44	0010 1100	0	00101	10	M
186	1011 1010	0	10111	01	M
253	1111 1101	1	11111	10	M

b.)

Word Address	Binary Address	Offset N/A Fully Associative Doesn't have Offset	Tag	Index N/A Fully Associative Doesn't have Index	Hit(H)/Miss(M)
3	0000 0011		0000 0011		M
180	1011 0100		1011 0100		M
43	0010 1011		0010 1011		M
2	0000 0010		0000 0010		M
191	1011 1111		1011 1111		M
88	0101 1000		0101 1000		M
190	1011 1110		1011 1110		M
14	0000 1110		0000 1110		M
181	1011 0101		1011 0101		M
44	0010 1100		0010 1100		M
186	1011 1010		1011 1010		M
253	1111 1101		1111 1101		M

c.)

5.)

a.)

Miss Penalty = Main Memory Access / (1 / Processor Speed) = 100 / (1 / 2) = 200

1.) First level CPI = Base CPI + (L1 Miss Rate * Miss Penalty)
First Level CPI = 1.5 + (0.07 * 200) = 15.5
X2 Main Memory Access = 1.5 + (0.07 * 400) = 29.5
X0.5 Main Memory Access = 1.5 + (0.07 * 100) = 8.5

2.) Second Level 2-Way CPI = Base CPI + (L1 Miss Rate * L2 Map Speed) + (L2 2-Way Miss Rate * Global 2-Way Miss Penalty)
Second Level 2-Way CPI = 1.5 + (0.07 * 12) + (0.035 * 212) = 9.76
X2 Main Memory Access = 1.5 + (0.07 * 12) + (0.035 * 412) = 16.76
X0.5 Main Memory Access = 1.5 + (0.07 * 12) + (0.035 * 112) = 6.26

3.) Second Level 8-Way CPI = Base CPI + (L1 Miss Rate * L2 8-Way Map Speed) + (L2 8-Way Miss Rate * Global 8-Way Miss Penalty)
Second Level 8-Way CPI = 1.5 + (0.07 * 28) + (0.015 * 228) = 6.88
X2 Main Memory Access = 1.5 + (0.07 * 28) + (0.015 * 428) = 9.88
X0.5 Main Memory Access = 1.5 + (0.07 * 28) + (0.015 * 128) = 5.38

b.)

Yes, we can have even greater cache hierarchy.
Miss Penalty = Main Memory Access / (1 / Processor Speed) = 100 / (1 / 2) = 200
Third Level CPI = Base CPI + (L1 Miss Rate * L2 Map Speed) + (L2 2-Way Miss Rate * L3 Map Speed) + (L3 Miss Rate * Reduced Global Miss Rate)
CPI = 1.5 + (0.07 * 12) + (0.035 * 50) + (0.013 * 200) = 6.69

c.)

2nd Level Direct Mapped Cache:
CPI = Base CPI + (L1 Miss Rate * L2 Map Speed) + (L2 2-Way Miss Rate * Global 2-Way Miss Penalty)
1.5 + (0.07 * 12) + (0.035 * 212) = 9.76
Cache Needed:
1.5 + (0.07 * 12) + ((0.04 - 0.007n) * 212) = Approx 3 so 3 Times more cache

Second Level 8-Way CPI = Base CPI + (L1 Miss Rate * L2 8-Way Map Speed) + (L2 8-Way Miss Rate * Global 8-Way Miss Penalty)
Second Level 8-Way CPI = 1.5 + (0.07 * 28) + (0.015 * 228) = 6.88
Cache Needed:
1.5 + (0.07 * 28) + ((0.04 - 0.007) * 228) = Approx 5 so we need 5 times more cache