1.

Request	Cache Hit/Miss	Request on the bus	Who responds	State in Cache 1	State in Cache 2	State in Cache 3
				Inv	Inv	Inv
P1: Wr X	Miss	Wr req X	Memory	M(x)	Inv	Inv
P2: Wr X	Miss	Wr req X	P1	M(x)	M(x)	Inv
P3: Rd X	Miss	Rd req X	P2	M(x)	M(x)	S(X)
P1: Rd X	Hit	–	–	M(x)	M(X)	S(X)
P3: Wr X	Hit	Wr req X	–	Inv	Inv	M(X)
P3: Rd Y	Miss	Rd req Y	Memory	Inv	Inv	S(Y)
P2: Wr Y	Miss	Wr req Y	P3	Inv	M(Y)	Inv

Request	Cache Hit /Miss.	Messages	Directory State	State in Cache 1	State in Cache 2	State in Cache 3
			X:S: Y:S:	Inv	Inv	Inv
P1: Wr X	Miss	Wr-req to Dir in P4, Dir responds with the copy	X:M:1 Y:S:	M(X)	Inv	Inv
P2: Wr X	Miss	Wr-req to Dir in P4, Dir responds with the copy	X:M:1, 2 Y:S:	M(X)	M(X)	Inv
P3: Rd X	Miss	Rd-req to Dir in P4, Dir responds with the copy	X:S:1, 2, 3 Y:S:	S(X)	S(X)	S(X)
P1: Rd X	Hit	–	X:S:1, 2, 3 Y:S:	M(X)	M(X)	S(X)
P3: Wr X	Hit	Wir-req to Dir in P4, Dir sends invalidation to P1 and P2	X:M:3 Y:S:	Inv	Inv	M(X)
P3: Rd Y	Miss	Rd-req to Dir in P4, Dir responds with the copy	X:S: Y:S:3	Inv	Inv	S(Y)
P2: Wr Y	Miss	Wr-req to Dir in P4, Dir sends invalidation to P3	X:S Y:M:2	Inv	M(Y)	Inv

Scanned with CamScanner

3、解：(a) $1\ EB = 2^{60}$ bytes B

$8\ KB = 2^{13}$ B

$$\text{Number of rows} = \frac{2^{60}\ B}{2^{13}\ B/\text{row}} = 2^{47}\ \text{row}$$

(b) 每行DRAM每64 ms刷新一次

∴ Number of refreshes $= 2^{47}$ refreshes

4、解：(a) ①访问模式：应用模式程序A可能表现出更多的随机访问模式，而应用程序B表现出更多的流式访问模式（或其它更具局部性的访问模式）

②行缓冲利用：流式访问模式更有效地利用行缓冲，因为一旦一行被加载到行缓冲中，对该行的后续访问不需要额外的内存延迟。

③缓存命中率：应用程序B可能有更好的缓存命中率，因为它的访问模式更连续，而应用程序A的随机访问可能导致频繁的缓存未命中。

(b) ①刷新次数：应用程序A的随机访问模式可能导致更频繁的行刷新，因为行缓冲经常被新数据替换。

②功耗：频繁的行刷新会增加功耗，因为刷新操作需要能量来恢复数据。

③缓存未命中：更多的缓存未命中意味着更多的内存访问，这增加了功耗。

(c) ①资源竞争：两个应用程序同时运行时，它们竞争内存资源，如行缓冲和内存带宽。

②访问模式冲突：应用程序A的随机访问模式可能与应用程序B的流式访问模式冲突，导致更多的缓存未命中和行冲突。

③优先级和调度：如果内存调度器优先处理流式访问（如应用程序B），则应用程序A可能遭受更多的延迟。

(d) ①优点：这种策略减少了不必要的刷新，节省了能耗并提高了性能。

②缺点：如果应用模式程序的访问模式变化很快，这种策略可能导致数据丢失，因为一些行可能在刷新之前就已经过时。

③结论：这是一个好的设计，因为它优化了能耗和性能，但需要确保数据一致性。

(e) B的内存访问可能更具局部性，应用新策略会使较少的行被频繁访问和刷新，而A可能由于其内存访问模式导致更多的行被频繁访问，即使使用智能刷新策略，但对于高随机访问模式的程序A，其节能效果不如程序B。

Scanned with CamScanner

5、解：

LCC总大小：32 MB

缓存块大小：64 B

$$\text{Number of blocks} = \frac{32MB}{64B} = 2^{19}$$

$$\text{Number of sets} = \frac{2^{19}}{16} = 2^{15}$$

$$\text{Number of blocks per page} = \frac{16kB}{64B} = 2^{8}$$

每个 ~~title~~ tile 必须至少包含 ~~一个组(16个块)~~ 一个页面的缓存块数

$$\therefore \text{最大tile数量} = \frac{2^{19}}{2^{8}} = 2^{11} = 2048$$

6、解：(a) (1) FCFS策略：①严格按照请求到达的顺序处理

②如果一个请求需要的数据不在行缓冲中，必须等待行激活和数据传输，这可能导致等待时间

(2) FR-FCFS策略：①优先处理已经就绪的请求(即数据已经在行缓冲中)

②减少等待时间，提高内存访问的效率

(3) 例子：假设有两个请求R1和R2，R1请求的数据已经在行缓冲中，而R2请求的数据需要行激活。在FCFS中，R1必须等待R2的行激活完成，而在FR-FCFS中，R1可以立即处理，从而减少了总的等待时间。

(b) 在FR-FCFS策略下，线程B(随机访问模式)可以被对手用来对线程A(流式访问模式)执行Dos攻击。这是因为在FR-FCFS策略中，随机访问模式可以用来对频繁地打开新的行(Row)，从而使流式访问模式的线程难以连续访问同一行中的数据。由于线程A顺序地访问同一行中的数据，线程B可以通过随机访问不同行的数据，使得线程A难以利用已经打开的行，因为它的请求经常被线程B的请求打断。这样，线程A的性能会显著下降，因为它需要不断等待打开，而线程B则可以利用这种策略来降低线程A的性能，从而实现Dos攻击。

Scanned with CamScanner