Date . . . No.

* Nguyên tắc cục bộ hoá.

4.1

Let ~~b~~ p is the probability of a cache hit
⇒ 1 - p would be probability for cache miss.

⇒ $t_a = p.t_h + (1-p).t_m$

⇒ $E = \frac{t_h}{t_a} = \frac{t_h}{p(t_h + ~~p~~(1-p)t_m}$

$= \frac{1}{p + (1-p)\frac{t_m}{t_h}}$

4.2

a) Using the rule x mod 8 for memory address x,
⇒ {2, 10, 18, 26} contend for location 2 in cache

b) With 4-way set ~~set~~ associativity, there are just two sets in 8 cache blocks
Call 0 is ~~bo~~ blocks contain 0, 1, 2, 3
Call 1 ——————— 4, 5, 6, 7.

⇒ Memory element 31 may therefore go to {4, 5, 6, 7}

c) We have:

order	1	2	3	4	5	6	7	8
Block refer	0	15	18	5	1	13	15	26
cache block	0	7	2	5	1	$\bar{5}$	7	2

HẢI TIẾN

For 4 - way set associative cache :

order	1	2	3	4	5	6	7	8
Block refer	0	15	18	5	1	13	15	26
Cache block	0	7	2	5	x			

4.4

+) Write - allocate :

• When a write (store) miss occurs, the missed cac line is brought into the first level cache before actua writing takes place.

+) No write - allocate

• When a write miss occurs the memory update bypass-es the cache ~~on~~ and updates the next level memo hierarchy where there is a hit

4.5

Imagine a 4-word cache with an access pattern of 0,1,2,3,4,0,1,2,3,4. The directed mapped cach will have a 30% hit rate while the LRU fully associative cache will have a 0% hit rate

4.6

When a word is loaded from main memory, adj words are loaded into the cache line. Spatial locality says that these adjacent bytes are likely to be used A common example is iterating through elements in an array

4.7

Advantages :

1) Improved Performance : A Physically - Tagged

Date . . . No.

L1 cache performs memory accesses based on physical addresses. When using logical indexing, it can perform accesses based on logical addresses, which can help impore performance in certain cases

2) Reduced Tag complexity : In a physically - indexed and physically - tagged L1 cache, synchronizing access operations across different threads can require more complex management. logical indexing can reduce this complexity.

4.8

A case where LRU would be expected to outperform random replacement is a randomly - accessed tree structure that is too large to fit in the cache. The LRU policy will do a better job of keeping the root of the tree in the cache and those nodes of the tree are accessed more often.

4.9

One pathological case for LRU would be an application that makes multiple sequential passes through a data set that is slightly too large to fit in the cache. The LRU policy will result in a 0% cache hit rate, while under random replacement the hit rate will be high

4.10

a) Miss
b) HIT
c) Miss

4.11

We have .

EAT = 0,9 . 10 + ~~0,8~~ . 10000
+ 0,8 . 100 .

EAT = cache hit . T_{cache} ~~+ cache miss . (~~

+ cache miss . (memory hit . (T_{cache} + T_{memory})

+ + memory miss . (T_{cache} + T_{memory}

= 0,9 . 10 + 0,1 (0,8 (10 + 100) + 0,2 (10 + 10
+ 1000

= 220 ns .

4.12 .

Memory size	:	M .	(bytes)
Cache size	:	C .	(bytes)
Block size	:	B .	(bytes)

we have .

M = 256 * C .

M = 8 . 1024 . 1024 . 1024 .

Number block in cache . N = 256 K block
= 256 . 1024 block

$C = M/256 = \dfrac{8 . 1024 . 1024 . 1024}{256}$.

$\Rightarrow$ Block size $= \dfrac{C}{N} = \dfrac{8 . 1024 \ 1024 . 1024}{256 . 1024 . 256}$

= 128 bytes .

4.13 .

4.14 .

Time for 16 blocks without cache is
$16 \times 25ns = 400ns$.

In case having cache.

$\Rightarrow 16 \times 2 + M16 \times 25 < 400ns$

$\Rightarrow M16 < \frac{400-32}{25}$.

$M16 < 14,72$

$\Rightarrow$ Cache hit rate is more than $\frac{16 - 14,72}{16} = 8\%$.

5.2 .

4.3 .

a) Here a main memory access is a memory store operation. So we will consider both cases i.e all stores are L1 miss all stores are not L1 miss.

All stores are not L1 miss $= (L1 \text{ miss rate}) \times (L2 \text{ miss rate})$
$= 0,17 . 0,12 = 2,04\%$

All stores are L1 miss =
$= (\%$ data references that read ops)
$\times (L2 \text{ miss rate}) + (\%$ data ref that are writes)
$\times (L1 \text{ miss rate}) \times (L2 \text{ miss rate})$.
$= 0,5 . 0,12 + 0,5 \times 0,17 . 0.12$
$= 0,06 + 0,042 = 0,102 = 10,2\%$

b) Data $= 8 \text{ Kbytes} / 8 = 1024 \text{ byte} = 2^{10} = 10$ bits.
Instruction $= 4 \text{ Kbytes} / 8 = 512 \text{ bytes} = 2^{9} = 9$ bits.
$L2 = 2M / 32 = 64 \text{ Kbytes} = 32 \text{K sets} = 2^{15} = 15$ bits.

c) Longest possible memory access will be when L1 miss + L2 miss + write back to main memory

HẢI TIẾN

CamScanner

Date . . . . No.

So, total cycles $= 1 + 10 + 2 \times 101 = 213$ cycles

d) If you did not treat all stores as L_1 miss then

(Avg memory access time)total $= \left(\frac{1}{1.3}\right)$ avg mem access

$+ \frac{0.5}{1.3}$

avg mem access time $=$ (L_1 hit time) + (L_1 miss rate)
$\times$ [(L_2 hit time + L_2 miss rate $\times$ mem

inst $= 1 + 0{,}02 (10 + 0{,}10 . 1{,}5 . 101] = 1{,}503$
data $= 1 + 0{,}17 (10 + 0{,}10 . 1.5 . 101) = 5{,}276$
total $= \frac{1}{1.3} . 1{,}503 + \frac{0{,}5}{1.3} . 5{,}276 = 3{,}8$

6.3

T = Transfer time
r : rotation speed
N = Number byte on track

$T = \frac{b}{r.N}$
$b = 1MB = 2^{20}$ bytes

$r = 15000$ (rotation per min)
$N = 512$ bytes per sec $\times$ 400 sec per track
$= 204800$ bytes per track

$r = \frac{15000 \text{ (rotation per min)}}{60000 \text{ (ms per min)}}$ *
$= 0{,}25$ (rotation / ms)

Date . . . No.

$$\Rightarrow T = \frac{2^{20} \text{ (bytes)}}{0{,}25 \text{ (rota/ms)} \cdot 204800 \text{ (bytes/track)}}$$

$= 20{,}48$ ms

b)

$$T_a = T_s + \frac{1}{2r} + T$$

$= 4 + 2 + 20{,}48 = 26{,}508$ ms.

c) Rotation delay $= \frac{1}{2r} = 2$ms. (average 180degree vòng)

d) ~~Tot~~ Total time to read 1 sector (512Bytes)
= seek time + rotational delay + transfer time

$= 4ms + 2ms + \frac{512}{0{,}25 \cdot 204800} = 6{,}01$ms

e) 10 ms.

6.4.

512 bytes × 1024 sectors = 0,5MB / track
Multiplying by 2048 tracks / platter give 1 GB / platter

Block refer.	0	15	18	5	1	13	15	26
Block cache.	0	~~1~~7	2	5	1	5	7	~~1~~2
Block 4-way	0							

5.2.

a) The throughout put of each bank is 80.10^6 ope/s

4 banks ⇒ The peak throughput for the memory system is

$4 \cdot 80.10^6 = 320.10^6$ operation/s

⇒ Peak data rate $= 1{,}28$ byte/s

Date . . . No.

5.3 .

a) Each bank can handle one operation / cycle, so the peak throughput is 2 operation / cycle

b) At the start each cycle, both banks are ready to accept requests. Therefore, the process will always be able to execute at least one mem request per cycle. On average., the second m memory request will target the same memory b as first request half the time, and will have to wait for the next cycle. The other 50% of the tim the two requests target different banks and can be executed simultaneously. Therefore the processor is able to execute on average 1,5 operations / cycle

c) Each memory bank can execute 1 operation / cy every 10ns (100×10^6 ope/s).

$\Rightarrow$ The average data rate is $1,5 \times 800 \times 10^6$

$= 1,2 \cdot 10^9$ bytes/s

5.4 .

a) SRAM is better choice

b) DRAM is better choice