Ch5

5.3

按照惯例，cache以它包含的数据量来进行命名（例如，4KiB cache可以容纳4KiB的数据），但是cache还需要SRAM来存储元数据，如标签和有效位等。本题研究cache的配置如何影响实现它所需的SRAM总量以及cache的性能。对所有的部分，都假设cache是字节可寻址的，并且地址和字都是64位的。

5.3.1

Each word is 8 bytes; each block contains 2 words; thus, each block contains $16 = 2^4 bytes$.

The cache contains $32KiB = 2^{15}$ bytes of data. Thus, it has $2^{15}/2^4 = 2^{11}$ lines of data.

Each 64-bit address is divided into:

(1) a 3-bit word offset,

(2) a 1-bit block offset,

(3) an 11-bit index (because there are 2^11 lines),

and (4) a 49-bit tag (64 − 3 − 1− 11 = 49).

The cache is composed of: $2^{15} \times 8$ bits of data + $2^{11} \times 49$ bits of tag + $2^{11} \times 1$ valid bits = 364,544 bits = 45,568 bytes.

5.3.2

Each word is 8 bytes; each block contains 16 words; thus, each block contains 128 = 2^7 bytes.

The cache contains 64KiB = 2^{16} bytes of data. Thus, it has $2^{16}/2^7 = 2^9$ lines of data.

Each 64-bit address is divided into:

(1) a 3-bit word off set,

(2) a 4-bit block off set,

(3) a 9-bit index (because there are 2^9 lines),

and (4) a 48-bit tag (64 − 3 − 4 − 9= 48).

The cache is composed of: $2^{16} \times 8$ bits of data + $2^9 \times 48$ bits of tag + $2^9 \times 1$ valid bits = 549,376 bits

5.3.3

The larger block size may require an increased hit time and an increased miss penalty than the original cache. The fewer number of blocks may cause a higher conflict miss rate than the original cache.

5.3.4

Associative caches are designed to reduce the rate of conflict misses. As such, a sequence of read requests with the same 12-bit index field but a different tag field will generate many misses. For the cache described above, the sequence 0, 32768, 0, 32768, 0, 32768, …, would miss on every access, while a two-way set associate cache with LRU replacement, even one with a significantly smaller overall capacity, would hit on every access after the first two.

5.5

对一个64位地址的直接映射cache的设计，地址的以下位用于访问cache。

标签	索引	偏移
63~10	9~5	4~0

5.5.1

Each cache block consists of four 8-byte words. The total offset is 5 bits.

Three of those 5 bits is the word offset (the offset into an 8-byte word). The remaining two bits are the block offset. Two bits allows us to enumerate 2^2 = 4
words.

5.5.2

There are five index bits. This tells us there are $2^5 = 32$ lines in the cache.

5.5.3

The ratio is 1.21.

The cache stores a total of
$32 lines \times 4 words/block \times 8 bytes/word = 1024 bytes = 8192 bits.$

In addition to the data, each line contains 54 tag bits and 1 valid bit. Thus, the total bits required

= 8192 + 54*32 + 1*32 = 9952 bits.

5.5.4

下表记录了从上电开始cache访问的字节地址。

十六进制	00	04	10	84	E8	A0	400	1E	8C	C1C	B4	884
十进制	0	4	16	132	232	160	1024	30	140	3100	180	2180

Answer:

BYTES ADDRESS	BINARY ADDRESS	TAG	INDEX	OFFSET	HIT/MISS	BYTE REPLACED
0x00	0000 0000 0000	0x0	0x00	0x00	M	
0x04	0000 0000 0100	0x0	0x00	0x04	H	
0x10	0000 0001 0000	0x0	0x00	0x10	H	
0x84	0000 1000 0100	0x0	0x04	0x04	M	
0xE8	0000 1110 1000	0x0	0x07	0x08	M	
0xA0	0000 1010 0000	0x0	0x05	0x00	M	
0x400	0100 0000 0000	0x1	0x00	0x00	M	0x00-0x1F
0x1E	0000 0001 1110	0x0	0x00	0x1E	M	0x400-0x41F

BYTES ADDRESS	BINARY ADDRESS	TAG	INDEX	OFFSET	HIT/MISS	BYTE REPLACED
0x8C	0000 1000 1100	0x0	0x04	0x0C	H	
0xC1C	1100 0001 1100	0x3	0x00	0x1C	M	0x00-0x1F
0xB4	0000 1011 0100	0x0	0x05	0x14	H	
0x884	1000 1000 0100	0x2	0x04	0x04	M	0x80-0x9F

5.5.5

4/12 = 33%.

5.5.6

<index, tag, data>

<0, 3, Mem[0xC00]-Mem[0xC1F]>

<4, 2, Mem[0x880]-Mem[0x89f]>

<5, 0, Mem[0x0A0]-Mem[0x0Bf]>

<7, 0, Mem[0x0e0]-Mem[0x0ff]>

5.6

回想一下，我们有2种写策略和2种写分配策略，它们的组合可以在L1和L2 cache中实现。假设L1 和 L2 cache有以下的选择：

L1	L2
写直达，写不分配	写返回，写分配

5.6.1

在不同级别的存储器层次结构之间使用缓冲器以减少访问延迟。对于这个给定的配置，列出L1和L2 cache之间以及L2 cache和内存之间可能需要的缓冲器。

Answer:

The L1 cache has a low write miss penalty while L2 cache has a high write miss penalty. A write buffer between the L1 and L2 cache would hide the write miss latency of the L2 cache. The L2 cache would benefit from write buffers when replacing a dirty block, since the new block would be read in before the dirty block is physically written to memory.

5.6.2

On an L1 write miss, the word is written directly to L2 without bringing its block into the L1 cache. If this results in an L2 miss, its block must be brought into the L2 cache, possibly replacing a dirty block, which must first be written to memory.

5.6.3

After an L1 write miss, the block will reside in L2 but not in L1. A subsequent read miss on the same block will require that the block in L2 be written back to
memory, transferred to L1, and invalidated in L2.

5.10

本题研究不同cache容量对整体性能的影响。

	L1大小	L1失效率	L1命中时间
P1	2KiB	8.0%	0.66ns
P2	4KiB	6.0%	0.90ns

5.10.1

P1	1.515GHZ
P2	1.11GHZ

5.10.2

P1	6.31NS	9.56 CYCLES
P2	5.11NS	5.68 CYCLES

5.10.3

P1	12.64 CPI	8.34 NS
P2	7.36 CPI	6.63 NS

For P1, every instruction requires at least one cycle. In addition, 8% of all instructions miss in the instruction cache and incur a 107-cycle delay.

Furthermore, 36% of the instructions are data accesses. 8% of these 36% are cache misses, which adds an additional 107 cycles.

With a clock cycle of 0.66 ps, each instruction requires 8.34 ns.

Using the same logic, we can see that P2 has a CPI of 7.36 and an average of only 6.63 ns/instruction.

5.10.4

$1 + 0.08 \times (9 + 0.95 \times 107) = 9.85$

worse.

5.10.5

$9.85 + 0.36 \times 8.85 = 13.04.$

5.10.6

Because the clock cycle time and percentage of memory instructions is the same for both versions of P1, it is sufficient to focus on AMAT. We want AMAT with L2 < AMAT with L1 only

$1 + 0.08 \times (9 + m \times 107) < 9.56$,

this happens when $m < 0.916$.

5.10.7

We want P1's average time per instruction to be less than 6.63ns, this means that we want

$CPI(P1) \times 0.66 < 6.63$. Thus, we need $CPI(P1) < 10.05$.

$CPI(P1) = AMAT(P1) + 0.36 \times (AMAT(P1)) - 1$

Thus, AMAT(P1) < 7.65.

$1 + 0.08 \times (9 + m \times 107) < 7.65$

and find that

$m < 0.693$

This means rate can be at most 69.3%.

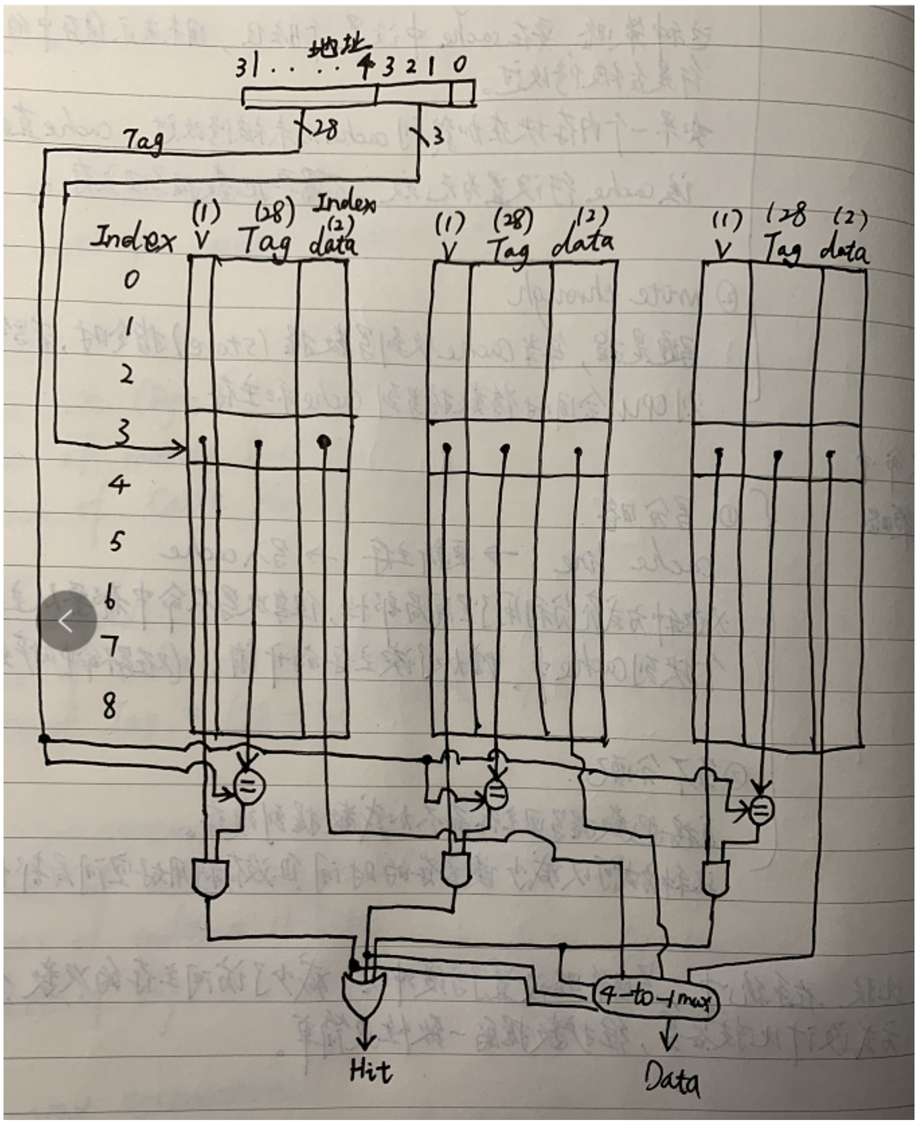
5.11

本题研究不同cache设计的效果，特别是组相联cache与直接映射cache进行比较。有关这些练习，请参阅下面显示的字地址序列：

0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, 0xce.

5.11.1

Each line in the cache will have a total of six blocks(two in each of three ways). There will be a total of 48/6 = 8 lines.



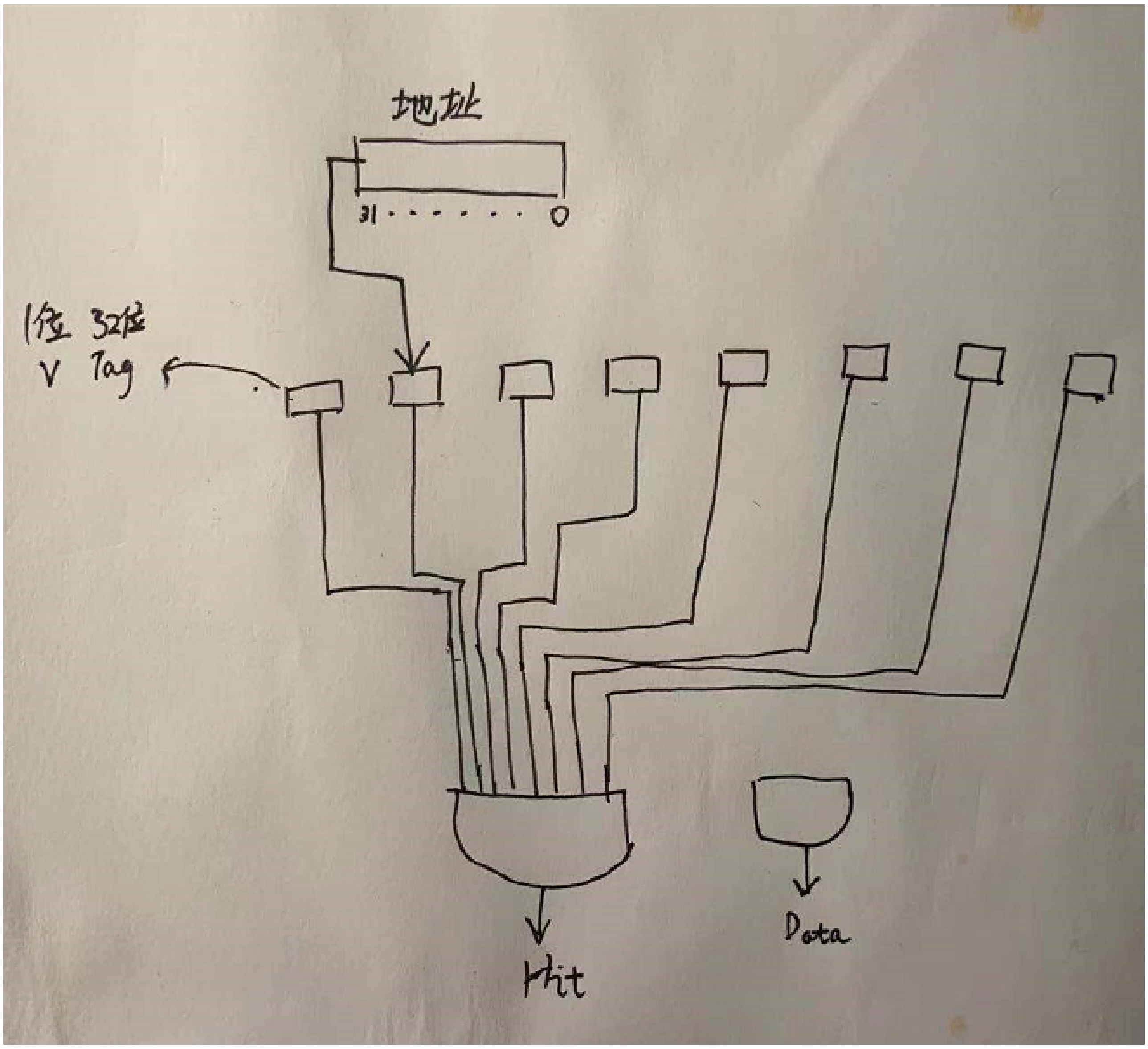
5.11.2

WORD ADDRESS	BINARY ADDRESS	TAG	INDEX	OFFSET	HIT MISS	WAY 0	WAY 1	WAY 2
0x03	0000 0011	0x0	1	1	M	T(1)=0		
0xb4	1011 0100	0xb	2	0	M	T(1)=0 T(2)=b		
0x2b	0010 1011	0x2	5	1	M	T(1)=0 T(2)=b T(5)=2		
0x02	0000 0010	0x0	1	0	H	T(1)=0 T(2)=b T(5)=2		

WORD ADDRESS	BINARY ADDRESS	TAG	INDEX	OFFSET	HIT MISS	WAY 0	WAY 1	WAY 2
0xbe	1011 1110	0xb	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b		
0x58	0101 1000	0x5	4	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0x0e	0000 1110	0x0	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	
0x1f	0001 1111	0x1	7	1	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xb5	1011 0101	0xb	2	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1

WORD ADDRESS	BINARY ADDRESS	TAG	INDEX	OFFSET	HIT MISS	WAY 0	WAY 1	WAY 2
0xba	1011 1010	0xb	5	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0 T(5)=b	T(7)=1
0x2e	0010 1110	0x2	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0xce	1100 1110	0xc	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=c

5.11.3

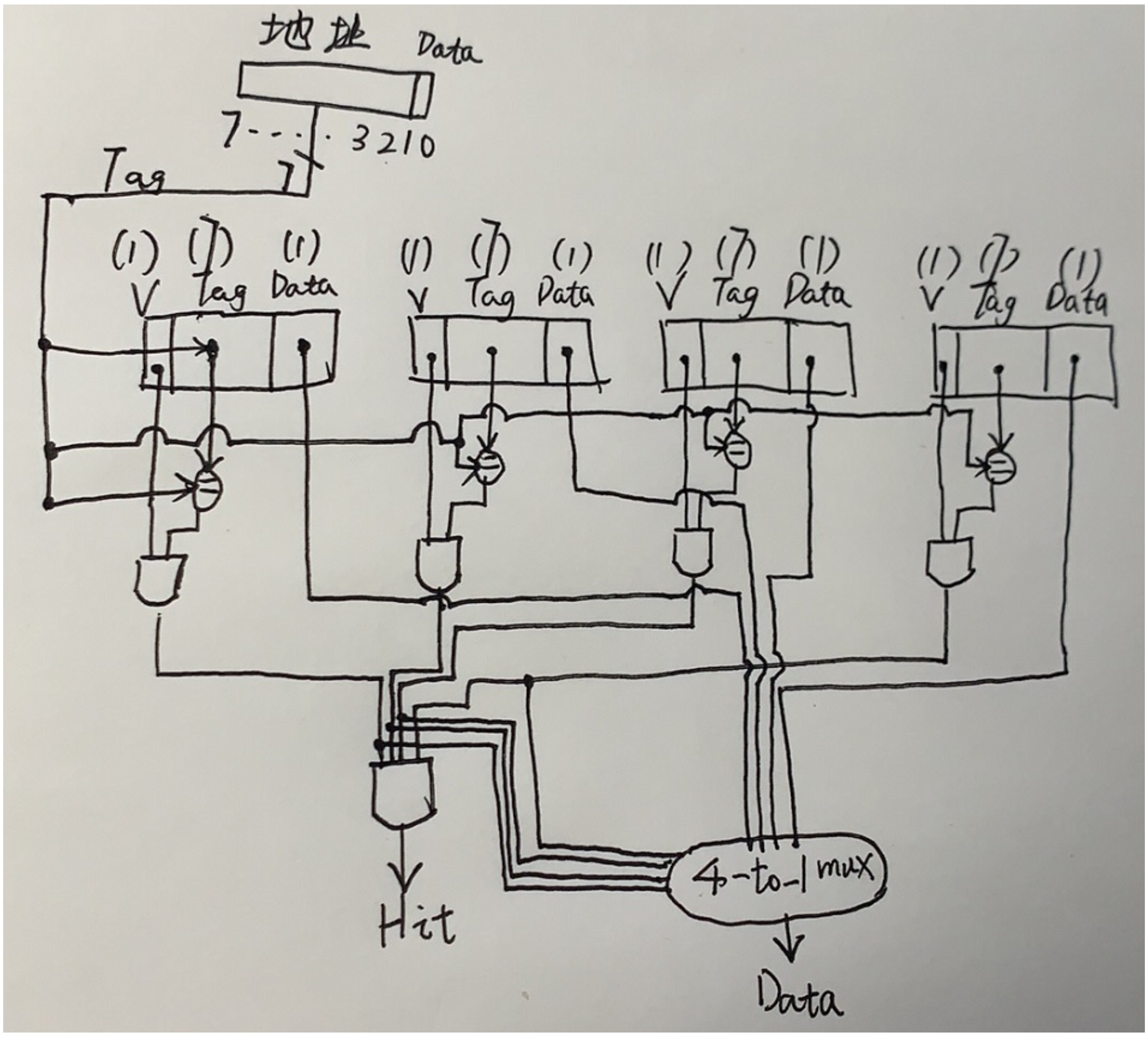


5.11.4

WORD ADDRESS	BINARY ADDRESS	TAG	HIS/ MISS	CONTENTS
0x03	0000 0011	0x03	M	3
0xb4	1011 0100	0xb4	M	3, b4
0x2b	0010 1011	0x2b	M	3, b4, 2b
0x02	0000 0010	0x02	M	3, b4, 2b, 2
0xbe	1011 1110	0xbe	M	3, b4, 2b, 2, be
0x58	0101 1000	0x58	M	3, b4, 2b, 2, be, 58
0xbf	1011 1111	0xbf	M	3, b4, 2b, 2, be, 58, bf
0x0e	0000 1110	0x0e	M	3, b4, 2b, 2, be, 58, bf, e
0x1f	0001 1111	0x1f	M	b4, 2b, 2, be, 58, bf, e, 1f

WORD ADDRESS	BINARY ADDRESS	TAG	HIS/ MISS	CONTENTS
0xb5	1011 0101	0xb5	M	2b, 2, be, 58, bf, e, 1f, b5
0xbf	1011 1111	0xbf	H	2b, 2, be, 58, e, 1f, b5, bf
0xba	1011 1010	0xba	M	2, be, 58, e, 1f, b5, bf, ba
0x2e	0010 1110	0x2e	M	be, 58, e, 1f, b5, bf, ba, 2e
0xce	1100 1110	0xce	M	58, e, 1f, b5, bf, ba, 2e, ce

5.11.5



5.11.6

Because this cache is fully associative, there is no index. (Contents shown in the order the data were accessed. Order does not simply physical location.)

WORD ADDRESS	BINARY ADDRESS	TAG	OFFSET	HIT/ MISS	CONTENTS
0x03	0000 0011	0x01	1	M	[2, 3]
0xb4	1011 0100	0x5a	0	M	[2, 3], [b4, b5]
0x2b	0010 1011	0x15	1	M	[2, 3], [b4, b5], [2a, 2b]
0x02	0000 0010	0x01	0	H	[b4, b5], [2a, 2b], [2, 3]
0xbe	1011 1110	0x5f	0	M	[b4, b5], [2a, 2b], [2, 3], [be, bf]
0x58	0101 1000	0x2c	0	M	[2a, 2b], [2, 3], [be, bf], [58, 59]
0xbf	1011 1111	0x5f	1	H	[2a, 2b], [2, 3], [58, 59], [be, bf]
0x0e	0000 1110	0x07	0	M	[2, 3], [58, 59], [be, bf], [e, f]
0x1f	0001 1111	0x0f	1	M	[58, 59], [be, bf], [e, f], [1e, 1f]
0xb5	1011 0101	0x5a	1	M	[be, bf], [e, f], [1e, 1f], [b4, b5]
0xbf	1011 1111	0x5f	1	H	[e, f], [1e, 1f], [b4, b5], [be, bf]
0xba	1011 1010	0x5d	0	M	[1e, 1f], [b4, b5], [be, bf], [ba, bb]
0x2e	0010 1110	0x17	0	M	[b4, b5], [be, bf], [ba, bb], [2e, 2f]
0xce	1100 1110	0x67	0	M	[be, bf], [ba, bb], [2e, 2f], [ce, cf]

5.11.7

WORD ADDRESS	BINARY ADDRESS	TAG	OFFSET	HIT/ MISS	CONTENTS
0x03	0000 0011	0x01	1	M	[2, 3]
0xb4	1011 0100	0x5a	0	M	[2, 3], [b4, b5]
0x2b	0010 1011	0x15	1	M	[2, 3], [b4, b5], [2a, 2b]
0x02	0000 0010	0x01	0	H	[b4, b5], [2a, 2b], [2, 3]
0xbe	1011 1110	0x5f	0	M	[b4, b5], [2a, 2b], [2, 3], [be, bf]
0x58	0101 1000	0x2c	0	M	[b4, b5], [2a, 2b], [2, 3], [58, 59]
0xbf	1011 1111	0x5f	1	M	[b4, b5], [2a, 2b], [2, 3], [be, bf]
0x0e	0000 1110	0x07	0	M	[b4, b5], [2a, 2b], [2, 3], [e, f]

WORD ADDRESS	BINARY ADDRESS	TAG	OFFSET	HIT/ MISS	CONTENTS
0x1f	0001 1111	0x0f	1	M	[b4, b5], [2a, 2b], [2, 3], [1e, 1f]
0xb5	1011 0101	0x5a	1	H	[2a, 2b], [2, 3], [1e, 1f], [b4, b5]
0xbf	1011 1111	0x5f	1	M	[2a, 2b], [2, 3], [1e, 1f], [be, bf]
0xba	1011 1010	0x5d	0	M	[2a, 2b], [2, 3], [1e, 1f], [ba, bb]
0x2e	0010 1110	0x17	0	M	[2a, 2b], [2, 3], [1e, 1f], [2e, 2f]
0xce	1100 1110	0x67	0	M	[2a, 2b], [2, 3], [1e, 1f], [ce, cf]

5.11.8

WORD ADDRESS	BINARY ADDRESS	TAG	OFFSET	HIT/ MISS	CONTENTS
0x03	0000 0011	0x01	1	M	[2, 3]
0xb4	1011 0100	0x5a	0	M	[2, 3], [b4, b5]
0x2b	0010 1011	0x15	1	M	[2, 3], [b4, b5], [2a, 2b]
0x02	0000 0010	0x01	0	H	[2, 3], [b4, b5], [2a, 2b]
0xbe	1011 1110	0x5f	0	M	[2, 3], [b4, b5], [2a, 2b], [be, bf]
0x58	0101 1000	0x2c	0	M	[58, 59], [b4, b5], [2a, 2b], [be, bf]
0xbf	1011 1111	0x5f	1	H	[58, 59], [b4, b5], [2a, 2b], [be, bf]
0x0e	0000 1110	0x07	0	M	[e, f], [b4, b5], [2a, 2b], [be, bf]
0x1f	0001 1111	0x0f	1	M	[1e, 1f], [b4, b5], [2a, 2b], [be, bf]
0xb5	1011 0101	0x5a	1	H	[1e, 1f], [b4, b5], [2a, 2b], [be, bf]
0xbf	1011 1111	0x5f	1	H	[1e, 1f], [b4, b5], [2a, 2b], [be, bf]
0xba	1011 1010	0x5d	0	M	[1e, 1f], [b4, b5], [ba, bb], [be, bf]
0x2e	0010 1110	0x17	0	M	[1e, 1f], [b4, b5], [2e, 2f], [be, bf]
0xce	1100 1110	0x67	0	M	[1e, 1f], [b4, b5], [ce, cf], [be, bf]

5.16

虚拟内存使用页表来追踪虚拟地址到物理地址的映射。本题显示了在访问地址时必须如何更新页表。以下数据构成了在系统上看到的虚拟字节地址流。假设有4KiB页，一个4表项全相联的TLB，适用严格的LRU替换策略。如果必须从磁盘中取回页，请增加下一次能取得最大页码：

5.16.1

ADDRESS	VIRTUAL PAGE	TLB	VALID	TAG	PHYSICAL PAGE
4669	1	TLB miss	1	b	12
0x123d		PT hit	1	3	6
		PF	1	7	4
			1	1	13
2227	0	TLB miss	1	3	6
0x08b3		PT hit	1	7	4
			1	1	13
			1	0	5
13916	3	TLB hit	1	7	4
0x365c			1	1	13
			1	0	5
			1	3	6
34587	8	TLB miss	1	1	13
0x871b		PT hit	1	0	5
		PF	1	3	6
			1	8	14
48870	b	TLB miss	1	0	5
0xbee6		PT hit	1	3	6
			1	8	14
			1	b	12
12608	3	TLB hit	1	0	5
0x3140			1	8	14
			1	b	12
			1	3	6
49225	c	TLB miss	1	8	14
0xc049		PT miss	1	b	12
			1	3	6
			1	c	15

5.16.2

ADDRESS	VIRTUAL PAGE	TLB	VALID	TAG	PHYSICAL PAGE

ADDRESS	VIRTUAL PAGE	TLB	VALID	TAG	PHYSICAL PAGE
4669	0	TLB miss	1	b	12
0x123d		PT hit	1	3	6
			1	7	4
			1	0	5
2227	0	TLB hit	1	b	12
0x08b3			1	3	6
			1	7	4
			1	0	5
13916	0	TLB hit	1	b	12
0x365c			1	3	6
			1	7	4
			1	0	5
34587	2	TLB miss	1	3	6
0x871b		PT hit	1	7	4
		PF	1	0	5
			1	2	13
48870	2	TLB hit	1	3	6
0xbee6			1	7	4
			1	0	5
			1	2	13
12608	0	TLB hit	1	3	6
0x3140			1	7	4
			1	2	13
			1	0	5
49225	3	TLB hit	1	7	4
0xc049			1	2	13
			1	0	5
			1	3	6

A larger page size reduces the TLB miss rate but can lead to higher fragmentation and lower utilization of the physical memory.

5.16.3

ADDRESS	VIRTUAL PAGE	TAG	INDEX	TLB H/M	VALID	TAG	PHYSICAL PAGE	INDEX

ADDRESS	VIRTUAL PAGE	TAG	INDEX	TLB H/M	VALID	TAG	PHYSICAL PAGE	INDEX
0x123d	1	0	1	TLB miss	1	b	12	0
				PT hit	1	7	4	1
				PF	1	3	6	0
					1	0	13	1
0x08b3	0	0	0	TLB miss	1	0	5	0
				PT hit	1	7	4	1
					1	3	6	0
					1	0	13	1
0x365c	3	1	1	TLB miss	1	0	5	0
				PT hit	1	1	6	1
					1	3	6	0
					1	0	13	1
0x871b	8	4	0	TLB miss	1	0	5	0
				PT hit	1	1	6	1
				PF	1	4	14	0
					1	0	13	1
0xbee6	b	5	1	TLB miss	1	0	5	0
				PT hit	1	1	6	1
					1	4	14	0
					1	5	12	1
0x3140	3	1	1	TLB hit	1	0	5	0
					1	1	6	1
					1	4	14	0
					1	5	12	1
0xc049	c	6	0	TLB miss	1	6	15	0
				PT miss	1	1	6	1
				PF	1	4	14	0
					1	5	12	1

5.16.4

ADDRESS	VIRTUAL PAGE	TAG	INDEX	TLB H/M	VALID	TAG	PHYSICAL PAGE	INDEX

ADDRESS	VIRTUAL PAGE	TAG	INDEX	TLB H/M	VALID	TAG	PHYSICAL PAGE	INDEX
0x123d	1	0	1	TLB miss	1	b	12	0
				PT hit	1	0	13	1
				PF	1	3	6	2
					0	4	9	3
0x08b3	0	0	0	TLB miss	1	0	5	0
				PT hit	1	0	13	1
					1	3	6	2
					1	4	9	3
0x365c	3	0	3	TLB miss	1	0	5	0
				PT hit	1	0	13	1
					1	3	6	2
					1	0	6	3
0x871b	8	2	0	TLB miss	1	2	14	0
				PT hit	1	0	13	1
				PF	1	3	6	2
					1	0	6	3
0xbee6	b	2	3	TLB miss	1	2	14	0
				PT hit	1	0	13	1
					1	3	6	2
					1	2	12	3
0x3140	3	0	3	TLB miss	1	2	14	0
				PT hit	1	0	13	1
					1	3	6	2
					1	0	6	3
0xc049	c	3	0	TLB miss	1	3	15	0
				PT miss	1	0	13	1
				PF	1	3	6	2
					1	0	6	3

5.16.5

Without a TLB, almost every access would require two accesses to RAM: An access to the page table, followed by an access to the request data.

5.17

有几个参数会影响页表的整体大小。下面列出的是关键的页表参数：

虚拟地址大小	页大小	页表项大小
32位	8 KiB	4字节

5.17.1

tag size = 32 - log2(8192) = 32-13 = 19 bits. All five page tables would require $5 \times (2^{19} \times 4)\ bytes = 10MB$.

5.17.2

In the two-level approach, the 2^{19} page table entries are divided into 256 segments that are allocated on demand. Each of the second-level tables contains $2^{(19-8)} = 2048$ entries, requiring $2048 \times 4 = 8\ kB$ each and covering $2048 \times 8KB = 16MB(2^{24})$ of the address space.

If we assume that "half the memory" means $2^{31}\ bytes$, then the minimum amount of memory required for the second-level tables would be $5 \times (2^{31}/2^{24}) \times 8KB = 5MB$.

The first-level tables would require an additional $5 \times 256 \times 6\ bytes = 7680\ bytes$.

The maximum amount would be if all 1st-level segments were activated, requiring the use of all 256 segments in each application. This would require $5 \times 256 \times 8KB = 10MB$ for the second-level tables and 7680 bytes for the first-level tables.

5.17.3

The page index is 13 bits (address bits 12 down to 0).

A 16 KB direct-mapped cache with two 64-bit words per block would have 16-byte blocks and thus 16 KB/16 bytes = 1024 blocks. Thus, it would have 10 index bits and 4 offset bits and the index would extend outside of the page index.

The designer could increase the cache’s associativity. This would reduce the number of index bits so that the cache’s index fits completely inside the page index.

5.20

5.20.1

There is no hits.

5.20.2

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
M	M	M	M	M	M	M	M	H	H	M	M	M	M	H	H	M

5.20.3

There are various answer for this, it can be thought as if each replace is done randomly.

5.20.4

MRU is an optimal policy.

5.20.5

The best block to evict is the one that will cause the fewest misses in the future.

Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.

5.20.6

If you knew that an address had limited temporal locality and would conflict with another block in the cache, choosing not to cache it could improve the miss rate.

On the other hand, you could worsen the miss rate by choosing poorly which addresses to cache.