ICS Seminar Week7 Prep

李天宇 段棋怀 许珈铭

2023.10.28

Rules

remainder <- ordinal number in WeChat Group % 4

for all questions do

    if question number % 4 == remainder then

        you should work on it

    end

end

Q1

12、以下关于存储器的说法中，正确的是：

A. SRAM 和 DRAM 的融合形成 SDRAM ×

B. SRAM 单元比 DRAM 单元的晶体管数目更多 ✓

C. DRAM 是一种非易失性存储器 ×

D. 固态硬盘的读写速度基本相当 ×

Synchronous DRAM

6:1

读更快

B

Q2

11．下列有关存储器的说法中，正确的是：

A．DMA 技术是指，当需要磁盘中的数据时，比起将磁盘数据先传送到主存，可以直接将磁盘数据通过总线传送到寄存器中，以提升效率

B．对于只有一个盘片，一个读写头的旋转磁盘，在分配逻辑块时，最好让属于同一逻辑块的扇区均匀分布在不同磁道上，以减少读取一个逻辑块的时间

C．SSD 相比于旋转磁盘，不需要寻道，但更容易磨损，因此常使用平均磨损逻辑以提高 SSD 的寿命

D． SRAM 的电路可以无限期保持其状态，断电之后依旧可以保存信息，而 DRAM 需要周期性地刷新状态，断电后信息将消失

C

Q3

12．下列有关存储器的说法中，错误的是：

A．旋转磁盘对扇区的访问时间有三个主要的部分：寻道时间、旋转时间和传送时间✓

B．在旋转磁盘中，磁盘控制器和逻辑块的设计理念有利于为操作系统隐藏磁盘的复杂性，同时也有利于对损坏的扇区进行管理✓

C．DRAM 和磁盘的性能发展滞后于 CPU，现代处理器为了弥补 CPU 的访存需求和内存延迟的差距，频繁地使用高速缓存✓

D．SSD 的闪存芯片包含若干个块，一个块由若干页组成，写入页的时间和该页是否包含数据（是否全为 1）无关 x 必须全置1才能写

D

Q4

14. 以下关于存储的描述中，正确的是（　）

A) 由于基于 SRAM 的内存性能与 CPU 的性能有很大差距，因此现代计算机使用更快的基于 DRAM 的高速缓存，试图弥补 CPU 和内存间性能的差距。× 使用 cache，且SRAM>DRAM

B) SSD 相对于旋转磁盘而言具有更好的读性能，但是 SSD 写的速度通常比读的速度慢得多，而且 SSD 比旋转磁盘单位容量的价格更贵，此外 SSD 底层基于 EEPROM 的闪存会磨损。√

C) 一个有 2 个盘片、10000 个柱面、每条磁道平均有 400 个扇区，每个扇区有 512 个字节的双面磁盘的容量为 8GB。 2面 $2\times2\times10000\times400\times512=8.192\times10^9$，8.2 GB

D) 访问一个磁盘扇区的平均时间主要取决于寻道时间和旋转延迟，因此一个旋转速率为 6000RPM、平均寻道时间为 9ms 的磁盘的平均访问时间大约为 19ms。

$$\frac{1}{6000\,RPM}\times 60\times1000\,ms/min\left[\times\frac{1}{2}\right]+9\,ms=14ms$$

平均旋转时间

B

Q5

15．在高速缓存存储器中，关于全相联和直接映射结构，以下论述正确的是：

A．如果配备同样容量、技术的高速缓存，配备全相联高速缓存的计算机总是比配备直接映射高速缓存的计算机性能低

B．如果配备同样容量、技术的高速缓存，配备全相联高速缓存的计算机总是比配备直接映射高速缓存的计算机性能高

C．如果配备同样容量、技术的高速缓存，当数据在缓存中时，配备全相联高速缓存的计算机总是比配备直接映射高速缓存的计算机读数据慢

D．如果配备同样容量、技术的高速缓存，当数据在缓存中时，配备全相联高速缓存的计算机总是比配备直接映射高速缓存的计算机读数据快

在缓存中时，直接映射不需要逐块比对，只需检验一个块的 tag 即可

C

Q6

A、B、C 都没有办法找到原先hit，现在miss的情况

16、关于 cache 的 miss rate，下面那种说法是错误的。

A.保持 E 和 B 不变，增大 S，miss rate 一定不会增加

B.保持总容量和 B 不变，提高 E，miss rate 一定不会增加

C.保持总容量和 E 不变，提高 B，miss rate 一定不会增加

D.如果不采用"LRU"，使用"随机替换策略"，miss rate 可能会降低

答：（ ）

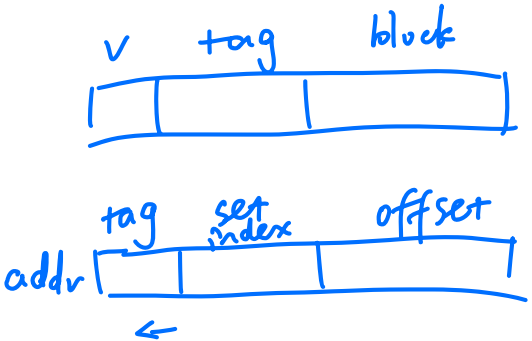
tag缩短

容量↑

S↓

S↓

可能"总"没问题



E：set的行数

B：没有块字节数

S：set数

set 0 line 1

line E

set 1

⋮

set S-1

A.考虑原先匹配set，现不匹配的一定是原先tag不匹配，原先也会miss，所以不会增加

B. set index缩短，原先匹配的set C 现也匹配，原先在某个line的现在也在，不会增加

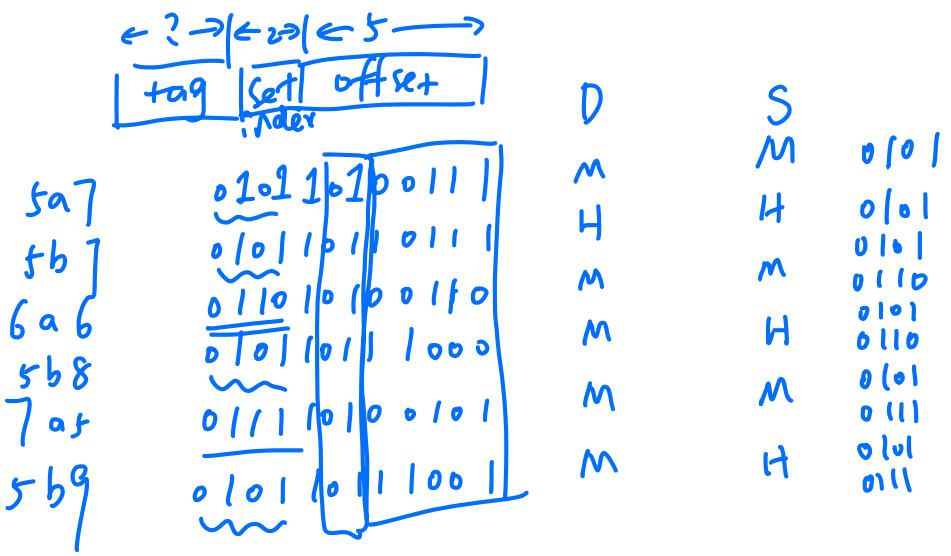
C. B↑ S↓ 原先已经缓存的 set 现也已缓存，但可能先前进来了不同 tag 的地址，可能 miss

Q7

12. 设一种缓存共包含 4 个缓存块，每个缓存块 32 字节，采用 LRU 替换算法。设缓存初始状态为空，对其进行下列 char 类型内存地址序列访存，0x5a7，0x5b7，0x6a6，0x5b8，0x7a5，0x5b9。对于直接映射缓存和 2 路组相联缓存，分别能产生几次命中？

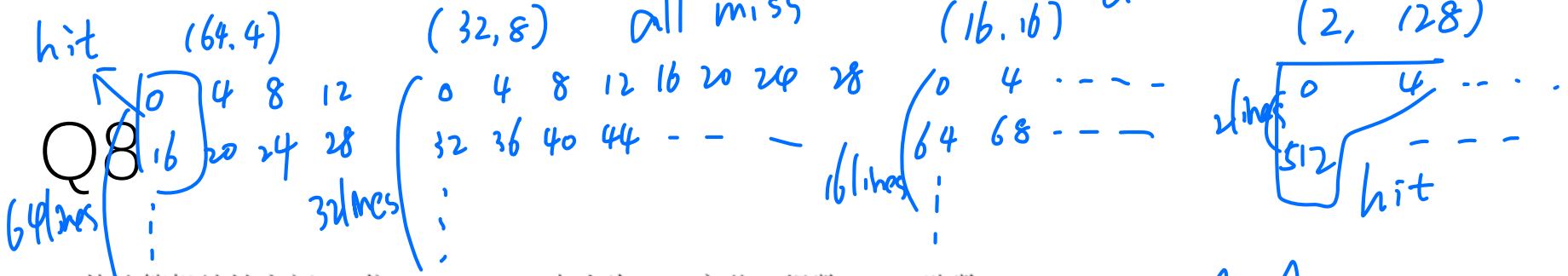
A) 1，3
B) 1，4
C) 2，3
D) 2，4

1个字节 2^5 Least Recently Used



A

all miss



15. 某计算机地址空间 12 位，L1 cache 大小为 256 字节，组数 S=4，路数 E=2，现在在地址 0x0 处开始有一个 N 行 M 列的 int 类型的数组 int A[N][M]，有如下 C 代码：

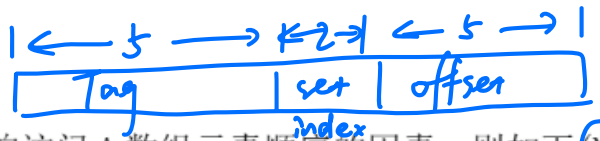
```
int ans=0;
for(int j=0;j<M;++j)
    for(int i=0;i<N;++i)
        ans+=A[i][j];
```

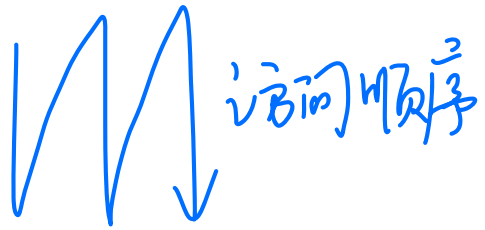
不考虑并行、编译优化等等会影响访问 A 数组元素顺序的因素，则如下(N,M)对中会导致全部 Cache Miss 的有（ ）

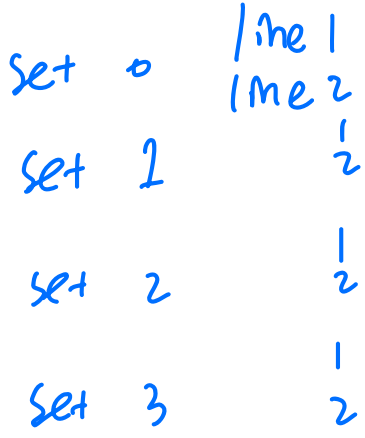
(N,M)= (64,4)，(32,8)，(16,16)，(2,128)

A) 4 个
B) 3 个
C) 2 个
D) 1 个

B=2^6÷4÷2=32=2^5







2进地址÷32再%4得到set

C

Q16

14. 假设已有声明 int i, int j, float x, int y, const int n, int a[n], int b[n], int *p, int *q int *r, int foo(int), 以下哪项程序优化编译器总是可以进行：

A	for (i = 0; i < n; i++) x = (x+ a[j]) + b[j]	for (i = 0; i < n; i++) x= x + (a[j] + b[j])
B	*p += *q *p += *r	int tmp; tmp = *q + *r *p += tmp

C	for(i = 0; i < foo(n);
	i++)
	sum += i;
		int tmp = foo(n)
		for(i = 0; i < tmp; i++)
		sum += i;

D	for (i = 0; i < n; i++) y = (y * a[j]) * b[j]	for (i = 0; i < n; i++) y = y * (a[j] * b[j])

set 0 set 1

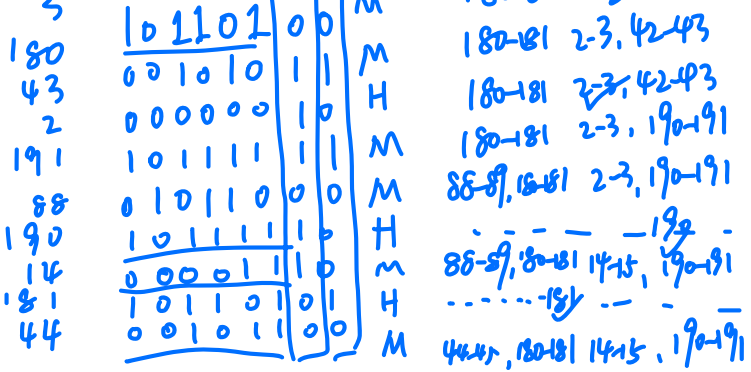
180 = 128+0+32+16+0+4+0+0

43 = 32+0+8+0+2+1

191 = 128+0+32+16+8+4+2+1

D

Q17



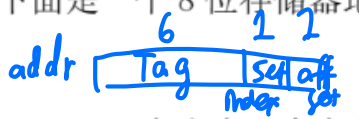
88 = 64+0+16+8+0+0+0

14 = 8+4+2+0

第五题（15 分）

Cache 为处理器提供了一个高性能的存储器层次框架。下面是一个 8 位存储器地址引用的列表（地址单位为字节，地址为 10 进制表示）：

3，180，43，2，191，88，190，14，181，44



1. 考虑如下 cache（S=2, E=2），每个 cache block 大小为 2 个字节。假设 cache 初始状态为空，替换策略为LRU。请填补下表：
(Tag 使用二进制格式；Data 使用十进制格式，例：M[6-7]表示地址 6 和 7 对应的数据)

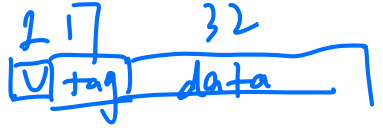
	V	Tag	Data	V	TAG	Data
SET 0	1	000101	M[44-45]	1	101101	M[180-181]
SET 1	1	000011	M[14-15]	1	101111	M[190-191]

共命中 3 次，分别访问地址 2, 190, 181 （地址用 10 进制表示）

3. 现在考虑另外一个计算机系统。在该系统中，存储器地址为 32 位，并采用如下的 cache：

Cache datasize	Cache block size	Cache mode
32 KiB	8 Bytes	直接映射

此 cache 至少要占用 41984 Bytes. (datasize + (valid bit size + tag size) * blocks)



4K = 4096 sets = 2^{12}

Q17

$\frac{18 \times 4K}{8} = 9K$

$9K + 32K = 41K$

|17|12|3|1

tag | set index | offset

1024
41
1024
4096
41984

	V	Tag	Data	V	TAG	Data
SET 0	1	101101 M[180-181]		1	001011 M[44-45]	
SET 1	1	000011 M[14-15]		1	101111 M[190-191]	

3 2, 190, 181
41984

Q18

a) C_1: 25×4+1×3=230 Cycles

C_2: 25×9+1×5=245 Cycles

C_1更好

b) $C_1=\frac{8}{10}=80\%$

$C_2=\frac{9}{10}=90\%$

第五题（15 分）

Cache 为处理器提供了一个高性能的存储器层次框架。下面是一个 8 位存储器地址引用的列表（地址单位为字节，地址为 10 进制表示）：

3，180，43，2，191，88，190，14，181，44

2. 现在有另外两种直接映射的 cache 设计方案 C1 和 C2，每种方案的 cache 总大小都为 8 个字节，C1 块大小为 2 个字节，C2 块大小为 4 个字节。假设从内存加载一次数据到 cache 的时间为 25 个周期，访问一次 C1 的时间为 3 个周期，访问一次 C2 的时间为 5 个周期。
   a) 针对第一问的地址访问序列，哪一种 cache 的设计更好？
   b) 请分别给出两种 cache 访问第一问地址序列的总时间，以及 miss rate。

3. 现在考虑另外一个计算机系统。在该系统中，存储器地址为 32 位，并采用如下的 cache：

Cache datasize	Cache block size	Cache mode
32 KiB	8 Bytes	直接映射

此 cache 至少要占用 41984 Bytes. (datasize + (valid bit size + tag size) * blocks)

Jump Q17

Q18

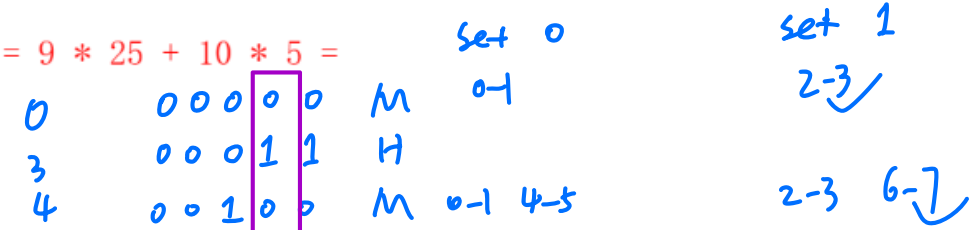
Address	Binary address	C1 hit/miss	C2 hit/Miss
3	000000 11	M	M
180	101101 00	M	M
43	001010 11	M	M
2	000000 10	M	M
191	101111 11	M	M
88	010110 00	M	M
190	101111 10	H	H
14	000011 10	M	M
181	101101 01	H	M
44	001011 00	M	M

C1 更好。（1 分）

C1: miss rate = 8/10 = 80%, （0.5 分） total cycles = 8 * 25 + 10 * 3 = 230 （1 分）

C2: miss rate = 9/10 = 90%, （0.5 分） total cycles = 9 * 25 + 10 * 5 = 275 （1 分）

41984



Q19

第五题（20 分）

现有一个能够存储 4 个 Block 的 Cache，每一个 Cache Block 的大小为 2 Byte（即 B = 2）。内存空间的大小是 32 Byte，即内存空间地址范围如下：

0_{10} (00000_2) -- 31_{10} (11111_2)

现有一程序，访问内存地址序列如下所示，单位是 Byte。

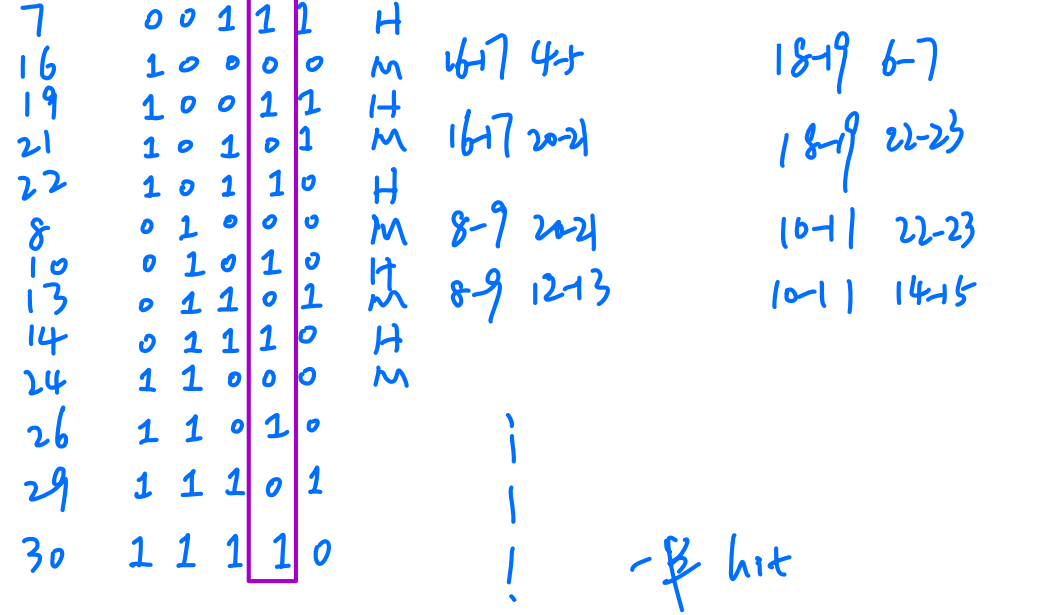
0_{10} 3_{10} 4_{10} 7_{10} 16_{10} 19_{10} 21_{10} 22_{10} 8_{10} 10_{10} 13_{10} 14_{10} 24_{10} 26_{10} 29_{10} 30_{10}

1. Cache 的结构如下图所示（S=2， E=2），初始状态为空，替换策略 LRU（Least Recently Used，最近最少使用）。请在下图空白处填入上述数据访问后 Cache 的状态。（TAG 使用二进制格式；Data Block 使用十进制格式，例：M[6-7]表示地址 6_{10}-7_{10} 对应的数据）

	V	TAG	Data Block	V	TAG	Data Block
set0	1	110	M[24-25]	1	111	M[28-29]
set1	1	110	M[26-27]	1	111	M[30-31]

上述数据访问一共产生了多少次 Hit：0

2. 在第 1 小题的基础上，现增加一条数据预取规则：每当 cache 访问出现 miss 时，被访问地址及其后续的一个 cache block 都会被放入缓存，即当 M[0-1]访问发生 miss，则把 M[0-1]和 M[2-3]都放入缓存中。那么，这 16 次数据访问一共产生了多少次 Hit：8



Q19

现有一个能够存储 4 个 Block 的 Cache，每一个 Cache Block 的大小为 2 Byte（即 B = 2）。内存空间的大小是 32 Byte，即内存空间地址范围如下：

0_{10} (00000_2) -- 31_{10} (11111_2)

现有一程序，访问内存地址序列如下所示，单位是 Byte。

0_{10} 3_{10} 4_{10} 7_{10} 16_{10} 19_{10} 21_{10} 22_{10} 8_{10} 10_{10} 13_{10} 14_{10} 24_{10} 26_{10} 29_{10} 30_{10}

1．Cache 的结构如下图所示（S=2， E=2），初始状态为空，替换策略 LRU。请在下图空白处填入上述数据访问后 Cache 的状态。(12 分)

（TAG 使用二进制格式；Data Block 使用十进制格式，例：M[6-7]表示地址 6_{10}-7_{10} 对应的数据）

	V	TAG	Data Block	V	TAG	Data Block
set0	1	110	M[24-25]	1	111	M[28-29]
set1	1	110	M[26-27]	1	111	M[30-31]

上述数据访问一共产生了多少次 Hit：__0__ (2 分)

2．在第 1 小题的基础上，现增加一条数据预取规则：每当 cache 访问出现 miss 时，被访问地址及其后续的一个 cache block 都会被放入缓存，即当 M[0-1] 访问发生 miss，则把 M[0-1]和 M[2-3]都放入缓存中。那么，这 16 次数据访问一共产生了多少次 Hit：__8__ (2 分)

Q20

第五题（20 分）

现有一个能够存储 4 个 Block 的 Cache，每一个 Cache Block 的大小为 2 Byte（即 B = 2）。内存空间的大小是 32 Byte，即内存空间地址范围如下：

0_{10} (00000_2) -- 31_{10} (11111_2)

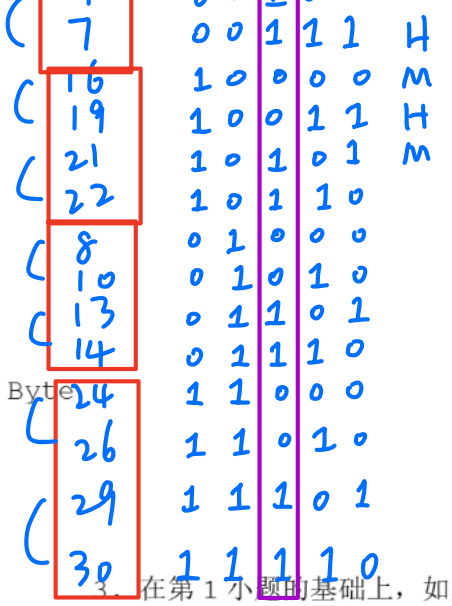
现有一程序，访问内存地址序列如下所示，单位是 Byte。

0_{10} 3_{10} 4_{10} 7_{10} 16_{10} 19_{10} 21_{10} 22_{10} 8_{10} 10_{10} 13_{10} 14_{10} 24_{10} 26_{10} 29_{10} 30_{10}

1. Cache 的结构如下图所示（S=2， E=2），初始状态为空，替换策略 LRU（Least Recently Used，最近最少使用）。请在下图空白处填入上述数据访问后 Cache 的状态。（TAG 使用二进制格式；Data Block 使用十进制格式，例：M[6-7]表示地址 6_{10}-7_{10} 对应的数据）

	V	TAG	Data Block	V	TAG	Data Block
set0						
set1						

上述数据访问一共产生了多少次 Hit：_____



可以观察发现
每2个都在同一大小为4 bytes 的 block
每4个都在同一 8 bytes 的 block

3. 在第 1 小题的基础上，如果每一个 Cache Block 的大小扩大为 4 Byte（即 B = 4，cache 大小变为原来的 2 倍），这 16 次数据访问一共产生了多少次 Hit：_____8_____

4. 在第 3 小题的基础上，考虑增加 2 中的数据预取规则，这 16 次数据访问一共产生了多少次 Hit：_____12_____

32÷8=4
16-4=12

Q20

现有一个能够存储 4 个 Block 的 Cache，每一个 Cache Block 的大小为 2 Byte（即 B = 2）。内存空间的大小是 32 Byte，即内存空间地址范围如下：

0_{10} (00000_2) -- 31_{10} (11111_2)

现有一程序，访问内存地址序列如下所示，单位是 Byte。

0_{10} 3_{10} 4_{10} 7_{10} 16_{10} 19_{10} 21_{10} 22_{10} 8_{10} 10_{10} 13_{10} 14_{10} 24_{10} 26_{10} 29_{10} 30_{10}

1．Cache 的结构如下图所示（S=2， E=2），初始状态为空，替换策略 LRU。请在下图空白处填入上述数据访问后 Cache 的状态。(12 分)
（TAG 使用二进制格式；Data Block 使用十进制格式，例：M[6-7]表示地址 6_{10}-7_{10} 对应的数据）

	V	TAG	Data Block	V	TAG	Data Block
set0	1	110	M[24-25]	1	111	M[28-29]
set1	1	110	M[26-27]	1	111	M[30-31]

上述数据访问一共产生了多少次 Hit：__0__ (2 分)

3．在第 1 小题的基础上，如果每一个 Cache Block 的大小扩大为 4 Byte（即 B = 4，cache 大小变为原来的 2 倍），这 16 次数据访问一共产生了多少次 Hit：__8___ (2 分)

4．在第 3 小题的基础上，考虑增加 2 中的数据预取规则，这 16 次数据访问一共产生了多少次 Hit：__12___ (2 分)

13. 对于 loop-unrolling 这种优化技巧，请指出下面哪一个陈述是错误的（一个）：

A. Loop-unrolling 的原理是将尽量多的循环操作去掉相关性并重组，从而提高循环操作的并行性。

B. Loop-unrolling 是一种将循环操作拆散的技术。

C. Loop-unrolling 可以利用目标处理器的并行处理能力。

D. 支持 Loop-unrolling 是有代价的，没有限制地增加并行支路数反而会降低运算速度。

B。因为简单将循环拆散并不能提高性能，必须实现更高的并行合并

B

14、下面哪些选项是错误的？答：（　　）

A. 同一个任务采用时间复杂度为 O(logN)算法一定比采用复杂度为 O(N)算法的执行时间短

B. 编译器进行程序优化时，总是可以使用算数结合律来减少计算量

C. 增大循环展开（loop unrolling）的级数，有可能降低程序的执行性能（即增加执行时间）

D. 分支预测时，“总是预测不跳转”（branch not taken）一定比“总是预测跳转”（branch taken）预测准确率高

A。错误。还有空间复杂度，局部性等带来的内存操作开销可能因为实现方式不同而不同，可能会超过算法本身的提升，在数据量比较小的情况下不一定时间更短
B。错误。浮点数不能使用结合律
C。正确。发生寄存器溢出时
D。错误。当分支高度可预测时，总是预测跳转更好

ABD

13、下面关于程序性能的说法中，哪种是正确的？

A．处理器内只要有多个功能部件空闲，就能实现指令并行，从而提高程序性能。

B．同一个任务采用时间复杂度为 O(logN)算法一定比采用复杂度为 O(N)算法的执行时间短

C．转移预测总是能带来好处，不会产生额外代价，对提高程序性能有帮助。

D．增大循环展开（loop unrolling）的级数，有可能降低程序的性能（即增加执行时间）

答：（　　）

A。错误。只有操作间没有数据相关时，才能实现指令并行
B。错误。还有空间复杂度，局部性等带来的内存操作开销可能因为实现方式不同而不同，可能会超过算法本身的提升，在数据量比较小的情况下不一定时间更短
C。当转移预测错误时，会有预测处罚
D。正确，存在寄存器溢出问题

D

14．关于局部性（locality）的描述，不正确的是：

A．循环通常具有很好的时间局部性

B．循环通常具有很好的空间局部性

C．数组通常具有很好的时间局部性

D．数组通常具有很好的空间局部性

C。数组只能体现空间局部性

C

14．关于局部性（locality）的描述，正确的是：

A．数据的时间局部性或数据空间局部性，在任何有意义的程序中都能体现

B．指令的时间局部性或数据空间局部性，在任何有意义的程序中都能体现

C．数据的时间局部性，在任何循环操作中都能体现

D．数据的空间局部性，在任何数组操作中都能体现

B。选项笔误，“ 数据空间局部性 ”->“ 指令空间局部性 ”，更改后：只要程序中连续地取出指令，并有多次执行的指令，都可以体现，而这是普遍的
A、C、D “ 任何 ”不对

B

14. 一个不含数组和指针访问的循环通常不能体现以下哪种局部性？

A. 数据的时间局部性
B. 数据的空间局部性
C. 指令的时间局部性
D. 指令的空间局部性

B。数组和指针通常能够实现对连续内存的访问，这是体现数据空间局部性的需要

B

9、下面代码中不能体现以下哪种局部性：

A．数据的时间局部性

B．数据的空间局部性

C．指令的时间局部性

D．指令的空间局部性

```
for(i=0; i<100; i++){
   a[0]=a[0]+i;
}
```

B 只访问了a[0]，没有体现数据空间局部性

B

13. 分析下面的 C 程序，以下关于局部性(locality)说法错误的是：

```
int i = 0, a = 1, b = 1;
for (;i < 100; i++)   {
    a = a + b;
    b = a + b;
}
```

A. 体现了数据的时间局部性
B. 体现了指令的时间局部性
C. 体现了指令的空间局部性
D. 以上都没有体现

A。连续使用a和b的值，体现了数据时间局部性
B。循环体会被多次执行，体现了指令的时间局部性
C。循环中的指令可以连续的取出，体现了指令的空间局部性

D

14、仅考虑以下代码，哪些程序优化总是被编译器自动进行？（假设 int i, int j, int A[N], int B[N], int m, int *p 都是局部变量，N 是一个整数型常量，int foo(int) 是一个函数）

	优化前	优化后

A.	for (j = 0 ; j < N ; j ++)
	B[i] *= A[j];
		int temp = B[i];
		for (j= 0 ; j < N ; j ++)
		temp *= A[j];
		B[i] = temp;

B.	for (j = 0 ; j < N ; j ++)
	m + = i*N*j;
		int temp = i*N;
		for (j= 0 ; j < N ; j ++)
		m + = temp * j;

C.	i = foo(N);
	j = foo(N);
	if (*p != 0)
	m = j ;
		j = foo(N);
		if (*p != 0)
		m = j ;

D.	for (j = 0 ; j < foo(N) ; j ++)
	m ++;
		int temp = foo(N);
		for (j= 0 ; j < temp ; j ++)
		m ++;

答：（ ）

A。数组A和B可能指向同一地址，内存别名使用
B。正确
C、D。函数调用可能存在副作用

B

20. 以下哪些程序优化编译器总是可以自动进行？（假设 int i, int j, int A[N], int B[N], float m 都是局部变量，N 是一个整数型常量，int foo(int) 是一个函数）

	优化前	优化后

A.	`for (j = 0 ; j < N ; j ++)`
	`  m + = i*N*j;`
		`int temp = i*N;`
		`for (j= 0 ; j < N ; j ++)`
		`  m + = temp * j;`

B.	`for (j = 0 ; j < N ; j ++)`
	`  B[i] *= A[j];`
		`int temp = B[i];`
		`for (j= 0 ; j < N ; j ++)`
		`  temp *= A[j];`
		`B[i] = temp;`

C.	`for (j = 0 ; j < N ; j ++)` `  m = (m + A[j]) + B[j];`	`for (j = 0 ; j < N ; j ++)` `  m = m + (A[j] + B[j]);`

D.	`for (j = 0 ; j < foo(N) ; j ++)`
	`  m++;`
		`int temp = foo(N);`
		`for (j= 0 ; j < temp ; j ++)`
		`  m++;`

B。数组A和B可能指向同一个地址，内存别名使用
C。m为浮点数，浮点数运算时，不能结合
D。函数调用可能有副作用

A

12. 假设已有声明 `int i, int j, const int n, int r, int a[n], int b[n], int mul(int, int)`

以下程序优化编译器一般不会进行的是：

	优化前	优化后

A.	`for (j = 0; j < n; ++j)`
	`  a[n * 8] += b[j];`
		`int tmp = (n << 3);`
		`for (j = 0; j < n; ++j)`
		`  a[tmp] += b[j];`

B.	`for (j = 0; j < n; ++j)` `  r = (r * a[j]) * b[j];`	`for (j = 0; j < n; ++j)` `  r = r * (a[j] * b[j]);`

C.	`for (j = 0; j < n; ++j)`
	`  a[mul(n, i) + j] = b[j];`
		`int ni = mul(n, i);`
		`for (j = 0; j < n; ++j)`
		`  a[ni + j] = b[j];`

D.	`for (j = 1; j < n; ++j)`
	`  a[0] += a[j];`
		`int tmp = 0;`
		`for (j = 1; j < n; ++j)`
		`  tmp += a[j];`
		`a[0] += tmp;`

C。函数调用可能存在副作用

C

11. 假设已有声明 int i, int sum, int *p, int *q, int *r, const int n = 100, float a[n], float b[n], float c[n], int foo(int), void bar(), 以下哪项程序优化编译器总是可以进行？

A	for(i = 0; i < n; ++i) {
	  a[i] += b[i];
	  a[i] += c[i];
	}
		float tmp;
		for(i = 0; i < n; ++i) {
		  tmp = b[i] + c[i];
		  a[i] += tmp;
		}

B	*p += *q;
	*p += *r;
		int tmp;
		tmp = *q + *r;
		*p += tmp;

C	for(i = 0; i < n; ++i)
	  sum += i * 4;
		int N = n * 4;
		for(i = 0; i < N; i += 4)
		  sum += i;

D	for(i = 0; i < foo(n); ++i)
	  bar();
		int tmp = foo(n);
		for(i = 0; i < tmp; ++i)
		  bar();

A。数组a，b,c可能指向同一个地址，内存别名使用
B。p,q,r可能指向同一地址，内存别名使用
C。不涉及浮点运算
D。函数调用可能有副作用

C

8、仅考虑以下代码，哪个或哪些代码片段在当前主流编译器的标准优化选项下一定会被优化？（假设int i, int j, int A[N], int B[N], int *p都是局部变量，int foo(int) 是一个函数）

	优化前	优化后

A.	for (j = 0 ; j < N ; j++)
	  B[i] *= A[j];
		int temp = B[i];
		for (j= 0 ; j < N ; j++)
		  temp *= A[j];
		B[i] = temp;

B.	for (j = 0 ; j < N ; j++)
	  m + = i*N*j;
		temp = i*N;
		for (j= 0 ; j < N ; j++)
		  m + = temp * j;

C.	i = foo(N);
	j = foo(N)+1;
	if (i != j)
	  m = j ;
		m = j ;

D.	if(m==1){
	  i=3;j=4;
	}else{
	  i=4;j=3;
	}
	m=i+j;
		m = 7;

A.数组A和B可能存在内存别名
C 函数调用可能有副作用
D i和j可能会被使用，不能直接删除赋值操作，这可能导致程序行为不一致

B

14. 假设已有声明 int i, int j, float x, int y, const int n, int a[n], int b[n], int *p, int *q int *r, int foo(int), 以下哪项程序优化编译器总是可以进行：

A	for (i = 0; i < n; i++) x = (x+ a[j]) + b[j]	for (i = 0; i < n; i++) x= x + (a[j] + b[j])
B	*p += *q *p += *r	int tmp; tmp = *q + *r *p += tmp

C	for(i = 0; i < foo(n);
	i++)
	sum += i;
		int tmp = foo(n)
		for(i = 0; i < tmp; i++)
		sum += i;

D	for (i = 0; i < n; i++) y = (y * a[j]) * b[j]	for (i = 0; i < n; i++) y = y * (a[j] * b[j])

A：x为浮点数，浮点数不可以结合
B：可能出现内存别名使用
C：函数调用可能有副作用
D：没有浮点运算，可以结合

D