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

memory

15．下面关于存储器的说法，错误的是_____

A．SDRAM 的速度比 FPM DRAM 快

B．SDRAM 的 RAS 和 CAS 请求共享相同的地址引脚

C．磁盘的寻道时间和旋转延迟大致在一个数量级

D．固态硬盘的随机读写性能基本相当

D

10、以下关于存储器的说法中，正确的是：

A．SRAM 单元比 DRAM 单元的能耗更大

B．DRAM 通常用做高速缓存

C．DRAM 单元比 SRAM 单元的速度更快

D．固态硬盘的读写速度基本相当

A

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

B

13. 以下计算机部件中，通常不用于存储器层次结构（Memory Hierarchy）的是：

A. 高速缓存　　B. 内存　　C. 硬盘　　D. 优盘（U 盘）

D

16. 某磁盘的旋转速率为 7200 RPM，每条磁道平均有 400 扇区，则一个扇区的平均传送时间为____

A. 0.02 ms

B. 0.01 ms

C. 0.03 ms

D. 0.04 ms

A

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

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

D

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。

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。

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。

B

13．以下关于存储结构的讨论，那个是正确的：

A．增加额外一级存储，数据存取的延时一定不会下降

B．增加存储的容量，数据存取的延时一定不会下降

C．增加额外一级存储，数据存取的延时一定不会增加

D．以上选项都不正确

D

16、如果直接映射高速缓存大小是 4KB，并且块（block）大小为 32 字节，请问它每组（set）有多少行（line）？答：（　　）

A. 128　　B. 64　　C. 32　　D. 1

D

17. 某高速缓存 （E=2，B=4，S=16），地址宽度 14，当引用地址 0x9D28 处的 1 个字节时，tag 位应为：

A. 01110100

B. 001110000

C. 1110000

D. 10011100

A

18．关于高速缓存的说法正确的是 _____

A．直写（write through）比写回（write back）在电路实现上更复杂

B．固定的高速缓存大小，较大的块可提高时间局部性好的程序的命中率

C．随着高速缓存组相联度的不断增大，失效率不断下降

D．以上说法全不正确

D

19. 如果直接映射高速缓存大小是 32KB，并且块（block）大小为 32 字节，那么它每组（set）有_____行（line）。

20. 如果二路组相联高速缓存大小是 8KB，并且块（block）大小为 32 字节，那么它每路（way）有_____行（line）。

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

C

13. 设一种全相联缓存共包含 4 个缓存块，如果循环地顺序访问 5 个不同的缓存块，下列哪种替换算法会产生最多的命中？

A) 最近最少使用替换策略 LRU
B) 先入先出替换策略 FIFO
C) 随机替换策略 Random
D) 后入先出替换策略 LIFO

A

11、以下关于缓存的说法中，正确的是：

A．LRU 不会比 MRU（most-recently used）替换策略差

B．先入先出（FIFO）比随机替换命中率高

C．保持缓存容量、缓存块大小（B）不变，增加路数（E），命中率不会下降

D．以上说法都不正确

D

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：（ ）

C

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

A

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

C

optimization

12. 针对程序优化，请挑出下面唯一正确的陈述：

A. 用 add/sub 和 shift 替代 multiply/divide 永远能提高程序的运行速度。

B. 最有效的提高程序运行效率的方法是提高 compiler 的优化级别。

C. 跨 procedure 优化的障碍之一是因为使用了全局变量。

D. 程序中，*a += *b; *a += *b;永远可以用*a +=2*(*b);代替。

C

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

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

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：（ ）

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

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

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

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

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

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：（ 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++;

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

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();

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;

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])

D