南開大学
Nankai University

5.8

5.8.1

给出的是字地址，由于块大小为32个字节，所以每个块可以容纳四个字。因为程序是刚开始启动，所以肯定每四次访问就有一次缺失，所以缺失率为 $\frac{1}{4}$。所以根据3C模型，这些缺失属强制缺失，因此对64B随着cache容量和工作集的改变并不会有改变。

5.8.2

16字节的缺失率 $= \frac{1}{2}$　64字节缺失率 $= \frac{1}{8}$　128字节缺失率 $= \frac{1}{16}$

该负载所采用的是空间局部性。

5.8.3

在这种预取情况下，如果包括开始时的冷启动导致的强制缺失，那么缺失率为0。因为预取缓冲区会准备好下一个请求。

5.18

5.18.1

4KB = 4096B = 2^{12}B

所以字节偏移位是12位。所以标记位是 43-12 = 31位。

所以页表一共有 2^{31} 个页表项。每个页表项大小是4字节，所以总共的大小是 $2^{31} \times 4B = 2^{30} \times 8 = 8GB$

5.18.2

需要两级页表就可以，当TLB缺失时，需要2次访问存储器才能转换地址。

5.18.3

4字节大小已经足够大了。从 5.18.1 可知 2^{31} 个页表项可以覆盖 $2^{31} \times 4KB$ = 8TB的物理地址，这已经远远超过16GB。

5.18.4

每个高级页表有 $\frac{4KB}{4B} = 2^{10}$ 个页表项，所以每次转换费用10位。所以要用的转换次数为 $\left\lceil \frac{43-12}{10} \right\rceil = 4$，也就是要用4级转换。

5.18.5 在反向页表中，页表项的数量可以减少到哈希表的大小，加上冲突的代价。而在这种情况下，当TLB缺失时候就需要一个额外的引用来比较存储在哈希表中的标签。

5.23

模拟不同的事件ISA需要对该ISA的API进行特定的处理，而每个API都有特定的行为，这些行为将在指令执行中断、捕获内核模式等情况下发生。因此必须对其进行模拟，为了模拟更多指令，可能需要执行更多的指令，这可能会导致性能大幅下降，并使其难以与外部设备进行正常通讯。如果可以动态检查和优化仿真代码，则仿真系统可能比在其本机ISA上运行得更快。

5.27

5.27.1 srcIP和rettime。平均会引发2次缺失。

5.27.2 由5.27.1可知可以把srcIP和rettime字段分到一个单独的数组里面，其余的字段放到另一个单独的数组里面。

5.27.3
```
struct entry{
  Peak_hour;
  {srcIP, rettime, status};
}
```

6.4

6.4.1

我们从MIPS代码中可以看出，该循环要执行999次，而每次循环需要16个时钟周期，再加上循环外的3个时钟周期，所以一共需要16×999+3=15987个时钟周期。

6.4.2

重排后的代码为：

```
    li   $s0, 8000
    add  $s1, $a0, $s0
    addi $s2, $a0, 16
loop: l.d  $f0, -16($s2)
    l.d  $f2, -8($s2)
    add.d $f4, $f0, $f2
    addi $s2, $s2, 8
    s.d  $f4, 0($s2)
    bne  $s2, $s1, loop
```

重排之后，我们可以减少一个周期的阻塞，所以需要15×999+3=14961

6.4.3

数组元素D[i]和D[i-1]有这种循环相关性，其中相关的为D[i]，在迭代i期间加载到D[i]的值是迭代i-1期间产生的

6.4.4

```
    li   $s0, 8000
    add  $s1, $a0, $s0
    addi $s2, $a0, 16
    l.d  $f0, 0($s2)
    l.d  $f2, 8($s2)
    addi $s2, $s2, 8
    stall
    stall
    stall
    stall
    stall
loop: add.d $f4, $f0, $f2
    addi $s2, $s2, 8
    mov.d $f0, $f2
    mov.d $f2, $f4
    stall
    s.d  $f2, 0($s2)
    ble  $s2, $s1, loop
```

所以总共需要的周期数为 7×999+10=7003

6.4.5

```
        l.d   $f0, 0($s2)
        l.d   $f2, 8($s2)
        li    $s0, 8100
        add   $s1, $a0, $s0
        addi  $s2, $s2, 16
        NOP
        NOP
        NOP
Loop:   add.d $f4, $f0, $f2
        NOP
        NOP
        NOP
        NOP
        add.d $f0, $f4, $f2
        s.d   $f2, 0($s2)
        NOP
        NOP
        NOP
        NOP
        add.d $f2, $f4, $f0
        s.d   $f0, 0($s2)
        addi  $s2, $s2, 24
        NOP
        NOP
        s.d   $f2, -8($s2)
        bne   $s2, $s1, Loop
```

南開大学
Nankai University

因为每次循环要处理原先3次循环所要执行的操作，所以该代码只执行 $\frac{999}{3}=333$ 次，所以一共需要的时钟周期数为 $17\times333+10=5671$.

6.4.6

要想解决这个问题，我们可以复制原来的循环，一个当做原始循环另一个则当做循环展开的循环。假设我们展开循环X次，我们运行展开的循环，直到剩余的循环次数小于X，这时候，我们就切换到原始循环上，这样就能保证程序能正确执行。

6.4.7

不可以使用消息传递来提高性能，即使消息传递没有任何延迟它也不能使用多个CPU来完成并行工作。

南開大学
Nankai University

6.6

6.6.1

四核相对于单核的加速比大概是4倍。

6.6.2

因为每次更新都会产生缓存缺失的成本，因此会将获得的加速降低为缓存缺失成本的3倍。

6.6.3

要想解决虚假共享问题，我们可以通过跨列而不是行遍历来计算C的元素，因为这些元素会被映射到不同的缓存，然后我们只需要确保我们处理的矩阵索引是在同一个核上计算的(i,j)和(i+1,j)，就会消除虚假共享的问题。