习题课 3

郭海林

2014.5.27

4.9

```
for (i=0;i<300;i++) {
    c_re[i] = a_re[i] * b_re[i] – a_im[i] * b_im[i];
    c_im[i] = a_re[i] * b_im[i] + a_im[i] * b_re[i];
}
```

a. 运算密度 = 浮点操作个数 / 内存访问字节数

六次浮点数操作

4 次读操作：a_re, a_im, b_re, b_im + 2 次写操作：c_re, c_im

运算密度 = 6/(6*4) = 1/4

4.9 cont.

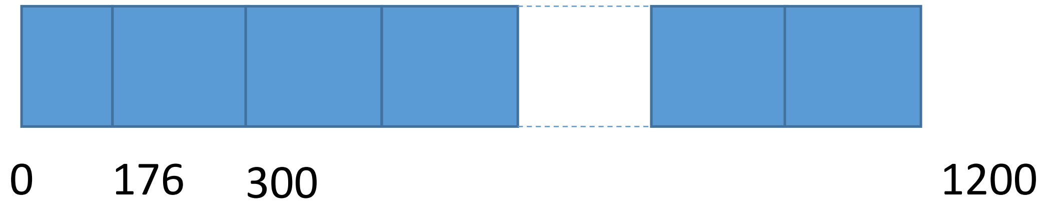
- b.

```
li          $VL,44                   # perform the first 44 ops
li          $r1,0                    # initialize index

loop:
            lv          $v1,a_re+$r1     # load a_re
            lv          $v3,b_re+$r1     # load b_re
            mulvv.s     $v5,$v1,$v3      # a+re*b_re
            lv          $v2,a_im+$r1     # load a_im
            lv          $v4,b_im+$r1     # load b_im
            mulvv.s     $v6,$v2,$v4      # a+im*b_im
            subvv.s     $v5,$v5,$v6      # a+re*b_re - a+im*b_im
            sv          $v5,c_re+$r1     # store c_re
            mulvv.s     $v5,$v1,$v4      # a+re*b_im
            mulvv.s     $v6,$v2,$v3      # a+im*b_re
            addvv.s     $v5,$v5,$v6      # a+re*b_im + a+im*b_re
            sv          $v5,c_im+$r1     # store c_im
```

```
            bne         $r1,0,else        # check if first iteration
            addi        $r1,$r1,#176      # first iteration,
                                            increment by 176
j loop                    # guaranteed next iteration

else:
            addi        $r1,$r1,#256      # not first iteration,
                                            increment by 256
skip:
            blt         $r1,1200,loop     # next iteration?
```



4.9 cont.

- c.

```
1.  mulvv.s   lv          # a_re * b_re (assume already loaded), load a_im
2.  lv      mulvv.s       # load b_im, a_im*b_im
3.  subvv.s   sv          # subtract and store c _re
4.  mulvv.s   lv          # a_re*b_im, load next a_re vector
5.  mulvv.s   lv          # a_im*b_re, load next b_re vector
6.  addvv.s   sv          # add and store c_im
```

6 次钟鸣

d. 每次迭代所用的时钟数：6*64 + 15 * 6 + 8 * 4 + 5 * 2 = 516

产生64 个结果：516 / 64 = 8

4.9 cont.

- e. 三条内存流水线

```
1. mulvv.s                  # a_re*b_re
2. mulvv.s                  # a_im*b_im
3. subvv.s  sv              # subtract and store c_re
4. mulvv.s                  # a_re*b_im
5. mulvv.s  lv  lv          # a_im*b_re, load next a_re, load next b_re
6. addvv.s  sv lv   lv      # add, store c_im, a_im,b_im
```

仍然是 6 次钟鸣，所需时钟数不受影响。

4.10

- Vector :
  - execution time: 400 ms
  - mem access: (200MB+100MB)/(30GB/s) = 10ms
  - total time: 410ms

- Hybrid:
  - execution time: 400 ms
  - mem access: (200MB+100MB)/(150GB/s) = 2ms
  - host IO: (200MB+100MB)/(10GB/s) = 30ms
  - latency: 10 ms
  - total time: 442ms

4.12

```
for (int x=0; x<NX-1; x++) {
 for (int y=0; y<NY-1; y++) {
 for (int z=0; z<NZ-1; z++) {
 int index = x*NY*NZ + y*NZ + z;
 if (y>0 && x >0) {
                material = IDx[index];
                dH1 = (Hz[index] – Hz[index-incrementY])/dy[y];
                dH2 = (Hy[index] – Hy[index-incrementZ])/dz[z];
                Ex[index] = Ca[material]*Ex[index]+Cb[material]*(dH2-dH1);
}}}}
```

a. 浮点操作个数：共8 FLOPS
   内存访问字节数：9次reads + 1次write
   运算密度： 8 / 40

b. 可以

c. 30 GB/s * 8/40 FLOPS/B = 6 GFLOPS/s
   如果峰值性能大于6 GFLPOS/s，则访存受限；否则，计算受限。

d. 单个浮点数运算密度： 85/4 = 21.25 GFLOPS/s

4.13

- a. 1.5 GHz × .80 × .85 × 0.70 × 10 cores × 32/4 = 57.12 GFLOPs/s
- b.
  - (1) Speedup = 16/8 = 2
  - (2) Speedup = 15/10 = 1.5
  - (3) Speedup = 0.95/0.85 = 1.11

4.14

- a.

```
for (i=0;i<100;i++) {
   A[i] = B[2*i+4];
   B[4*i+5] = A[i];
}
```

GCD 测试：

gcd(4, 2) | (4-5) ?

所以，不存在相关。

4.14 cont.

- b.

```
for (i=0;i<100;i++) {
        A[i] = A[i] * B[i]; /* S1 */
        B[i] = A[i] + c;    /* S2 */
        A[i] = C[i] * c;    /* S3 */
        C[i] = D[i] * A[i]; /* S4 */
}
```

真相关：

S2-S1 A[i];   S4-S3 A[i];

反相关：

S1-S2 B[i];   S2-S3 A[i];

S1-S3 A[i];   S3-S4 C[i];

输出相关：

S1-S3 A[i]

重命名消除反相关和输出相关：

```
for (i=0;i<100;i++) {
        A[i] = A[i] * B[i];   /* S1 */
        B1[i] = A[i] + c;     /* S2 */     （消除  s1-s2 关于 B[i] 的反相关）
        A1[i] = C[i] * c;     /* S3 */     （消除  s1-s3 关于 A[i] 的输出相关；
                                            同时消除 s1-s3,s2-s3 关于A[i] 的反相关）
        C1[i] = D[i] * A1[i]; /* S4 */     （消除  s3-s4 关于 C[i] 的反相关）
}
```

4.14 cont.

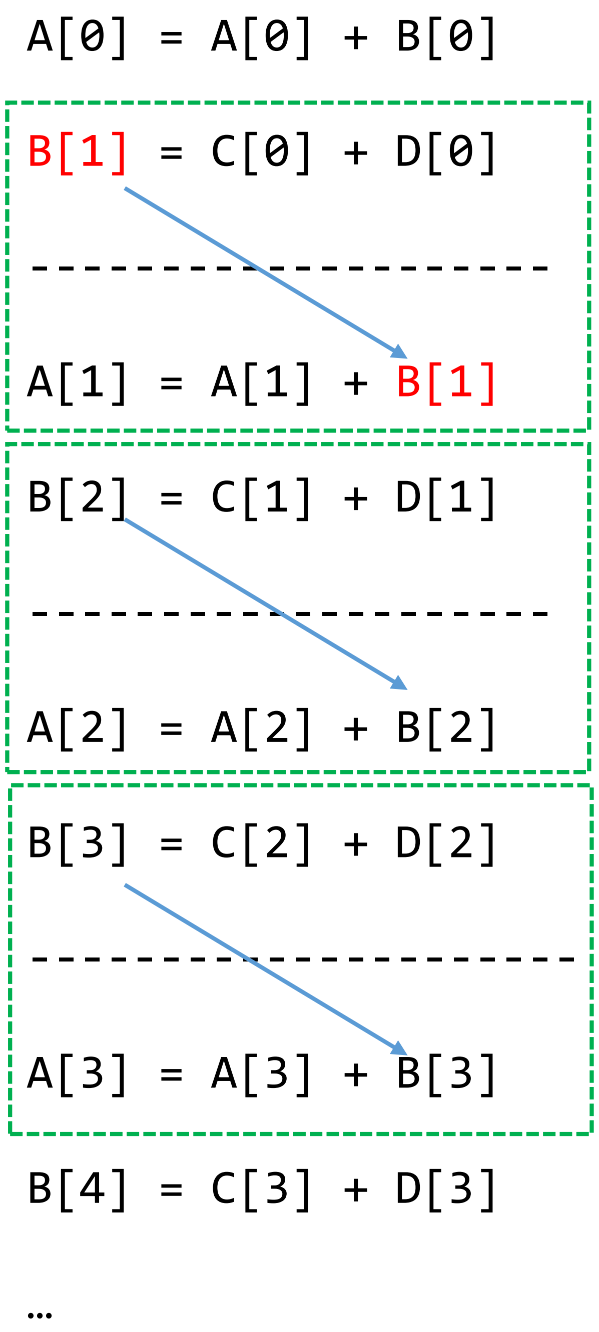
- c.

```
for (i=0;i < 100;i++) {
        A[i] = A[i] + B[i];   /* S1 */
        B[i+1] = C[i] + D[i]; /* S2 */
}
```

关于 B 存在循环间依赖，因此不能并行。

修改为：

```
A[0] = A[0] + B[0];
for (i=1; i<100; i++) {
        B[i] = C[i-1] + D[i-1];
        A[i] = A[i] + B[i];
}
B[100] = C[99] + D[99];
```



```
A[0] = A[0] + B[0]

B[1] = C[0] + D[0]
--------------------
A[1] = A[1] + B[1]

B[2] = C[1] + D[1]
--------------------
A[2] = A[2] + B[2]

B[3] = C[2] + D[2]
--------------------
A[3] = A[3] + B[3]

B[4] = C[3] + D[3]

…
```