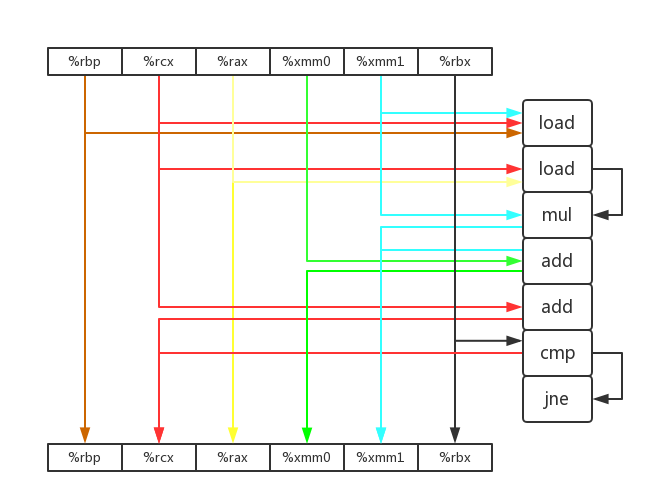
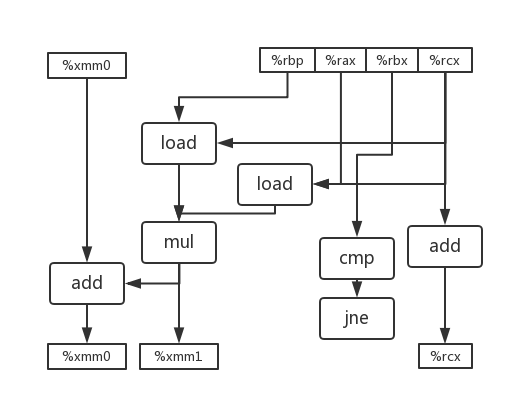
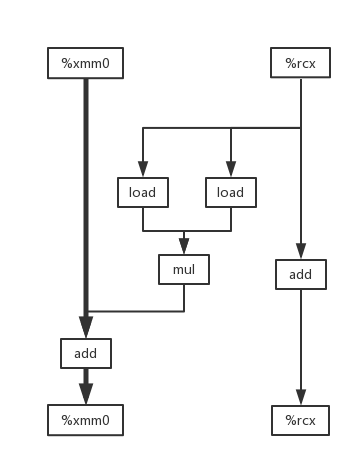
5.13

A.







B.3

C.1

D.每次循环时的乘法不需要依赖上一次乘法的结果，可以各自独立进行。但是加法是依赖于上一次的结果的，所以该循环的“关键路径”是加法这条链。而浮点数加法的延迟为3个周期，所以CPE为3.00。

5.15

A.

void inner6(vec\_ptr u, vec\_ptr v, data\_t \*dest) {

long i;

long length = vec\_length(u);

data\_t \*udata = get\_ver\_start(u);

data\_t \*vdata = get\_vec\_start(v);

data\_t sum0 = (data\_t)0;

data\_t sum1 = sum0, sum2 = sum0, sum3 = sum0, sum4 = sum0, sum5 = sum0;

for (i = 0; i < length - 5; i += 6) {

sum0 += udata[i] \* vdata[i];

sum1 += udata[i + 1] \* vdata[i + 1];

sum2 += udata[i + 2] \* vdata[i + 2];

sum3 += udata[i + 3] \* vdata[i + 3];

sum4 += udata[i + 4] \* vdata[i + 4];

sum5 += udata[i + 5] \* vdata[i + 5];

}

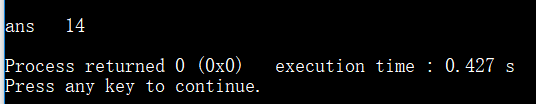
for (; i < length; i++)

sum0 += udata[i] \* vdata[i];

\*dest = sum0 + sum1 + sum2 + sum3 + sum4 + sum5;

}

测试：



B.

只有两个加载单元，一个时钟周期只能加载两个值，CPE 最低只能到 1.00

5.17

void \*memset(void \*s, int c, size\_t n)

{

unsigned char \*schar = s;

unsigned long w;

size\_t i = 0, limit = n - K + 1;

if (n < K)

{

while (i < n)

{

\*schar++ = (unsigned char)c;

i++;

}

}

else

{

for (i = 0; (size\_t)schar % K != 0 || i == n; i++) {

\*schar++ = (unsigned char)c;

}

for (; i < limit && (int)limit > 0; i += K) {

\*(unsigned long \*)schar = w;

schar += K;

}

for (; i < n; i++) {

\*schar++ = (unsigned char)c;

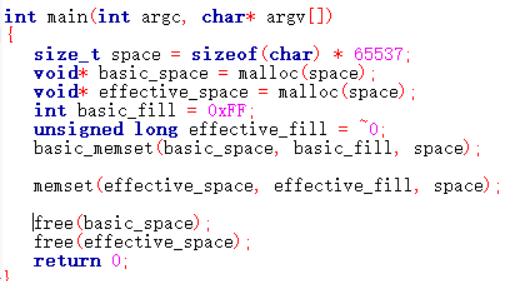
}

}

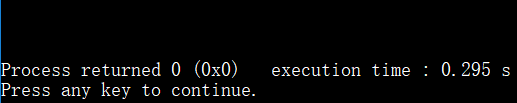
return s;

}

测试函数



测试结果



5.19

//浮点数加法分别对应3和1，所以这里选择3\*1a。

void faster(float a[], float p[], long n)

{

long i;

float val = 0;

for (i = 0; i + 2 < n; i += 3)

{

float t1 = a[i];

float t2 = t1 + a[i + 1];

float t3 = t2 + a[i + 2];

p[i] = var + t1;

p[i + 1] = var + t2;

p[i + 2] = var = var + t3;

}

for (; i < n; ++i)

{

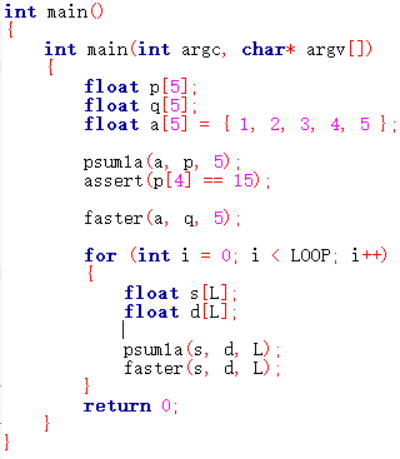
var += a[i];

p[i] = var;

}

}

测试函数



测试结果

