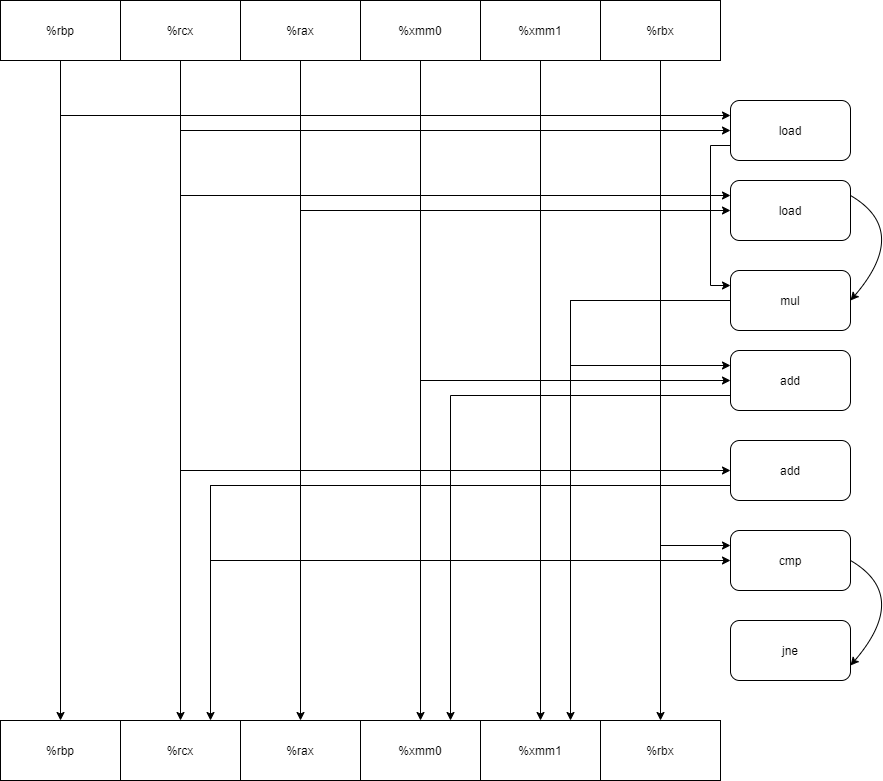
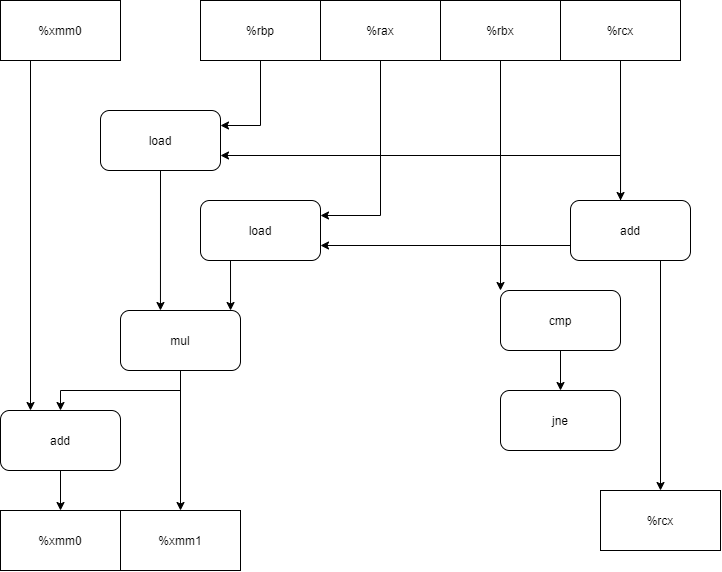
5.13

A.



B.3.0

C.1.0

D.关键路径上没有乘法，只有浮点加法

5.15

代码：

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

long i;

long length = vec\_length(u);

long limit = length - 5;

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

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

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

data\_t sum1 = (data\_t)0;

data\_t sum2 = (data\_t)0;

data\_t sum3 = (data\_t)0;

data\_t sum4 = (data\_t)0;

data\_t sum5 = (data\_t)0;

for (i = 0; i < limit; i += 6) {

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

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

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

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

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

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

}

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

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

}

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

}

原因：一次最多加载两个值，那么就限制了CPE的下限是1.0。而且在HasWell架构中加法延迟为一个周期，因此这么编写CPE下限不可能低于1.0。

5.17

#include <stdio.h>

#include <stdint.h>

#include <stdlib.h>

#include <string.h>

#include <assert.h>

#include<Windows.h>

void\* memset\_5\_17(void \*s, int c, size\_t n) {

size\_t K = sizeof(unsigned long);

size\_t cnt = 0;

unsigned long ul;

unsigned char \*temp=(unsigned char \*)&ul;

unsigned char \*schar = (unsigned char\*)s;//强制类型转换

while (cnt < K) {

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

cnt++;

}

cnt=0;

while (cnt < n)

{

//对齐

if ((size\_t)schar % K == 0) {

break;

}

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

cnt++;

}

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

size\_t res = n - cnt;

size\_t res2 = res % K;

size\_t num = res / K;

for (size\_t i = 0; i < num; i++) {

\*ulong++ = ul;

}

schar = (unsigned char \*)ulong;

for (size\_t i = 0; i < res2;i++) {

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

}

return s;

}

5.19

void psum4(float a[], float p[], long n)

{

long i;

float last\_val,val;

float temp0, temp1, temp2, temp3;

last\_val = p[0] = a[0];

for (i = 1; i < n - 4; i++) {

temp0 = last\_val + a[i];

temp1 = temp0 + a[i+1];

temp2 = temp1 + a[i+2];

temp3 = temp2 + a[i+3];

p[i] = temp0;

p[i+1] = temp1;

p[i+2] = temp2;

p[i+3] = temp3;

last\_val += (a[i] + a[i+1] + a[i+2] + a[i+3]);

}

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

last\_val += a[i];

p[i] = last\_val;

}

}