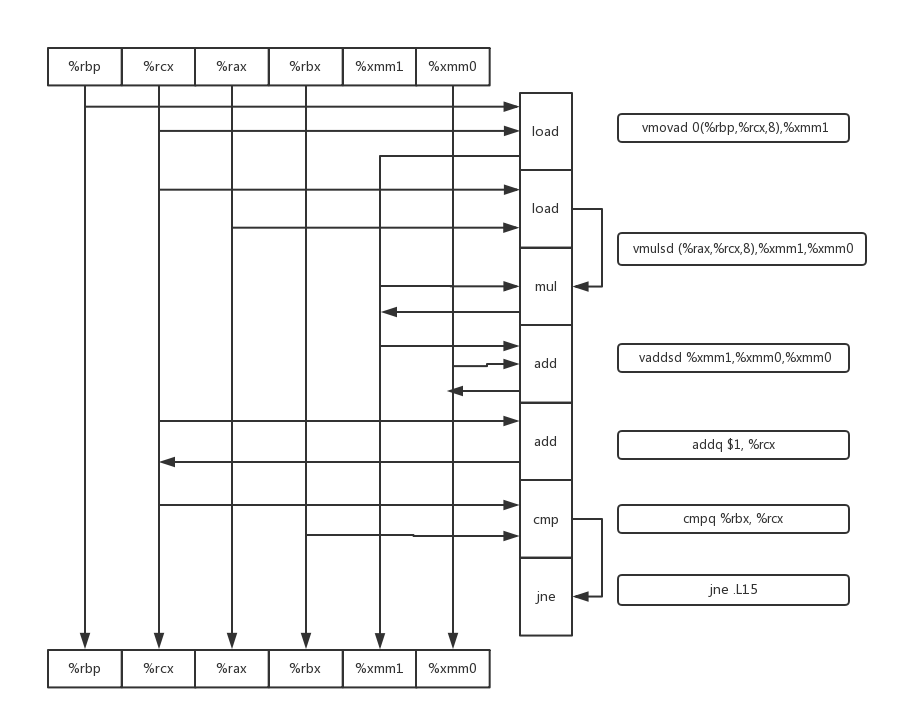
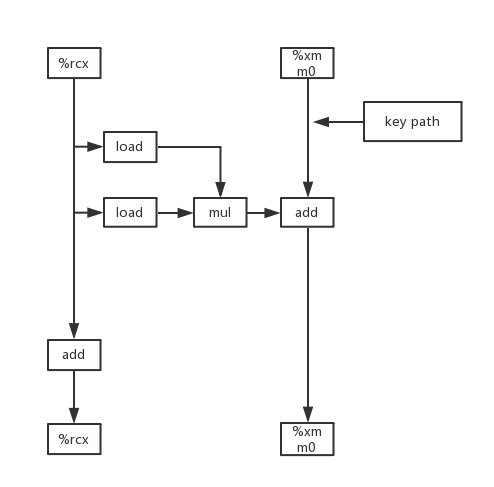
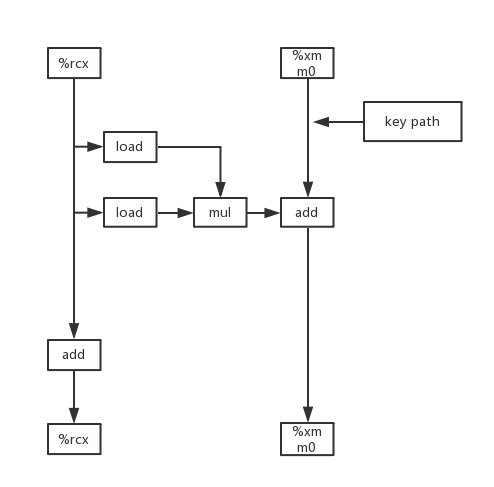
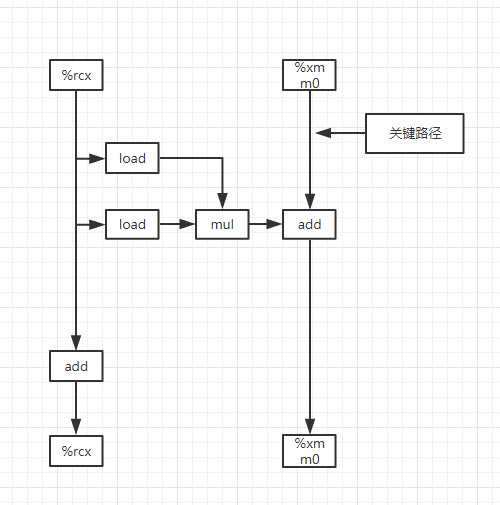
# 5.13

A



****



B

浮点数相加的模块，需要三个时钟周期

C

Long数据类型相加的模块，需要1.0个时钟周期

D

因为关键路径上只有浮点数相加。

# 5.15

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; **//初始化**

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) {  **//6X6循环展开**

sum0 = sum0 + udata[i+0] \* 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 += udata[i] \* vdata[i];

}

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

}

限制原因是功能单元的容量限制。只有两个加载单元，一个周期只能加载两个值。

# 5.17

void \*new\_memset(void \*s, int c, size\_t n) {

unsigned long w;

unsigned char \*pw = (unsigned char \*)&w;

size\_t cnt = 0;

while (cnt < K) {

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

cnt++;

}

size\_t i;

unsigned char \*schar = s;

for (i = 0; (size\_t)schar % K != 0 || i == n; i++) { **//为了达成对齐写入，目的地址是K的倍数的时候，要用字节级的写**

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

}  **//达成对齐的目的**

size\_t limit = n - K + 1; **//开始一次8个写入**

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

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

schar += K;

}

for (; i < n; i++) { **//最后在再对不满八个的不能一次性操作的一个字节一个字节操作**

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

}

return s;

}

# 5.19

**函数和测试代码如下**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void psum\_4(float a[], float p[], long n)

{

long i;

float val, last\_val;

float tmp, tmp1, tmp2, tmp3;

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

for (i = 1; i < n - 4; i+=4) **//每次处理四次循环操作,i+=4**

{

tmp = last\_val + a[i];

tmp1 = tmp + a[i+1];

tmp2 = tmp1 + a[i+2];

tmp3 = tmp2 + a[i+3];

p[i] = tmp;

p[i+1] = tmp1;

p[i+2] = tmp2;

p[i+3] = tmp3;

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

}

for (; i < n; i++)  **//如果不满四次最后部分单步操作**

{

last\_val += a[i];

p[i] = last\_val;

}

}

int main()

{

float a[10] = { 1, 2, 3, 4, 5,6,7,8,9,10};

float q[10];

psum\_4(a, q, 10);

for(int i=0; i<10; i++)

{

printf("%f\n",q[i]);

}

return 0;

**}输出其前面所有项的和**

