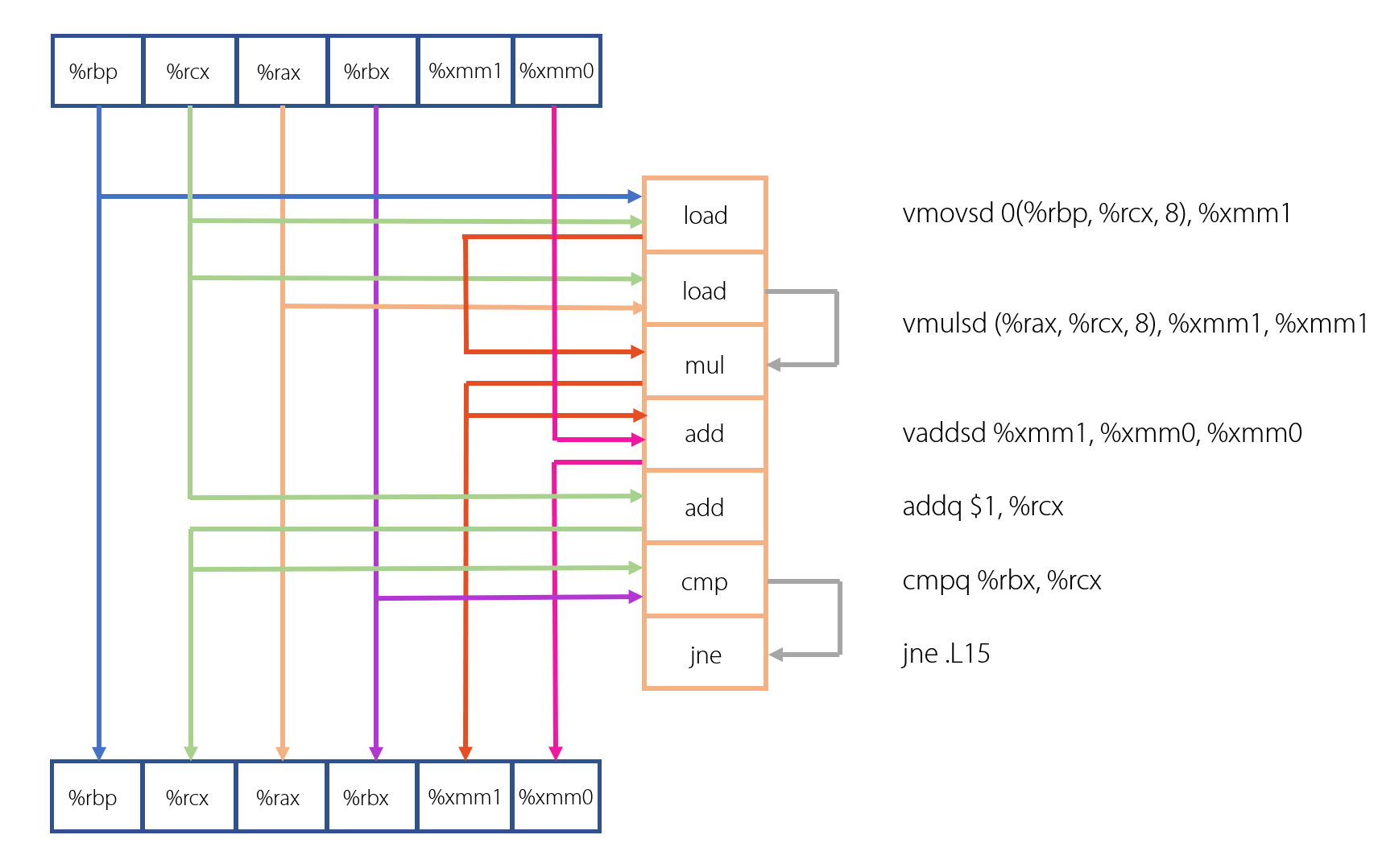
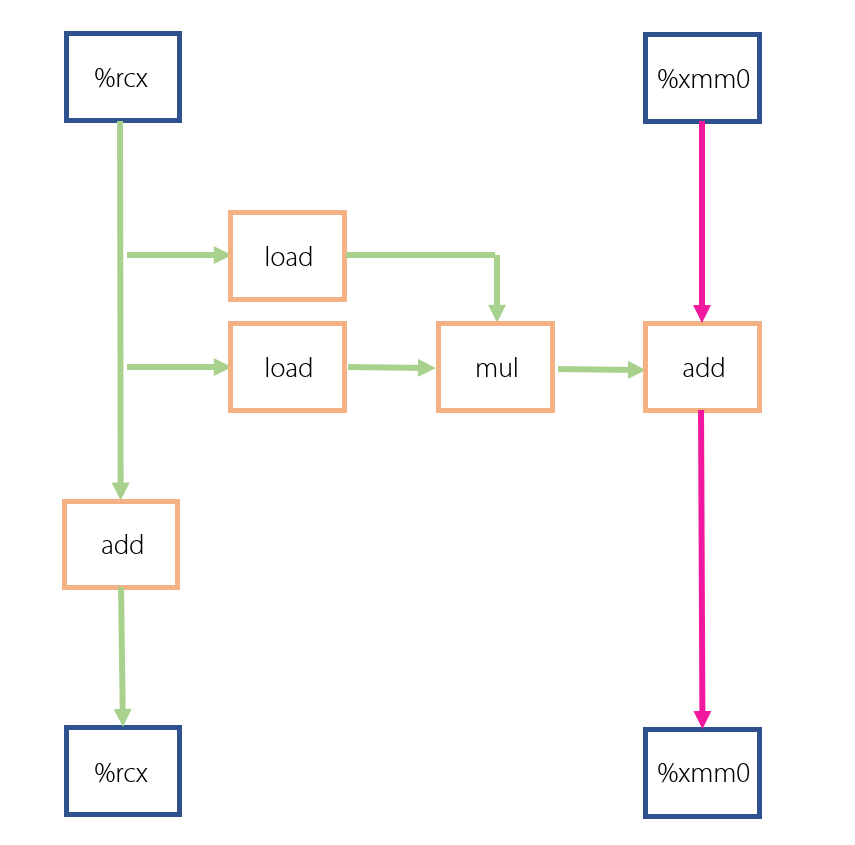
# 5.13

A.



如何形成关键路径：



B.

浮点加法的延迟

C.

两个load操作固有的吞吐量界限

D.

因为乘法不在关键路径上，故整个程序的限制因素为浮点数加法的延迟，对于浮点数据和double类型，这个延迟都是3.00。

# 5.15

void inner4(vec\_ptr *u*, vec\_ptr *v*, data\_t \**dest*) {

  long i;

  long length = vec\_length(*u*);

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

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

  data\_t sum = (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 < length-6; i+=6) {

    sum = sum + 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++) {

    sum = sum + udata[i] \* vdata[i];

  }

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

}

# 5.17

#include <assert.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

{

    size\_t cnt = 0;

    unsigned char \*schar = *s*;

    while (cnt < *n*)

    {

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

        cnt++;

    }

    return *s*;

}

void \*effective\_memset(void \**s*, unsigned long *cs*, size\_t *n*)

{

    size\_t K = sizeof(unsigned long);

    size\_t cnt = 0;

    unsigned char \*schar = *s*;

    while (cnt < *n*)

    {

        if ((size\_t)schar % K == 0) break;

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

        cnt++;

    }

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

    size\_t rest = *n* - cnt;

    size\_t loop = rest / K;

    size\_t tail = rest % K;

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

        \*slong++ = *cs*;

    schar = (unsigned char \*)slong;

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

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

    return *s*;

}

# 5.19

#include <stdio.h>

#include <assert.h>

void psum1a(float *a*[], float *p*[], long *n*) {

  long i;

  float last\_val, val;

  last\_val = *p*[0] = *a*[0];

  for (i = 1; i < *n*; i++) {

    val = last\_val + *a*[i];

*p*[i] = val;

    last\_val = val;

  }

}

void psum\_4\_1a(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++) {

    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++) {

    last\_val += *a*[i];

*p*[i] = last\_val;

  }

}