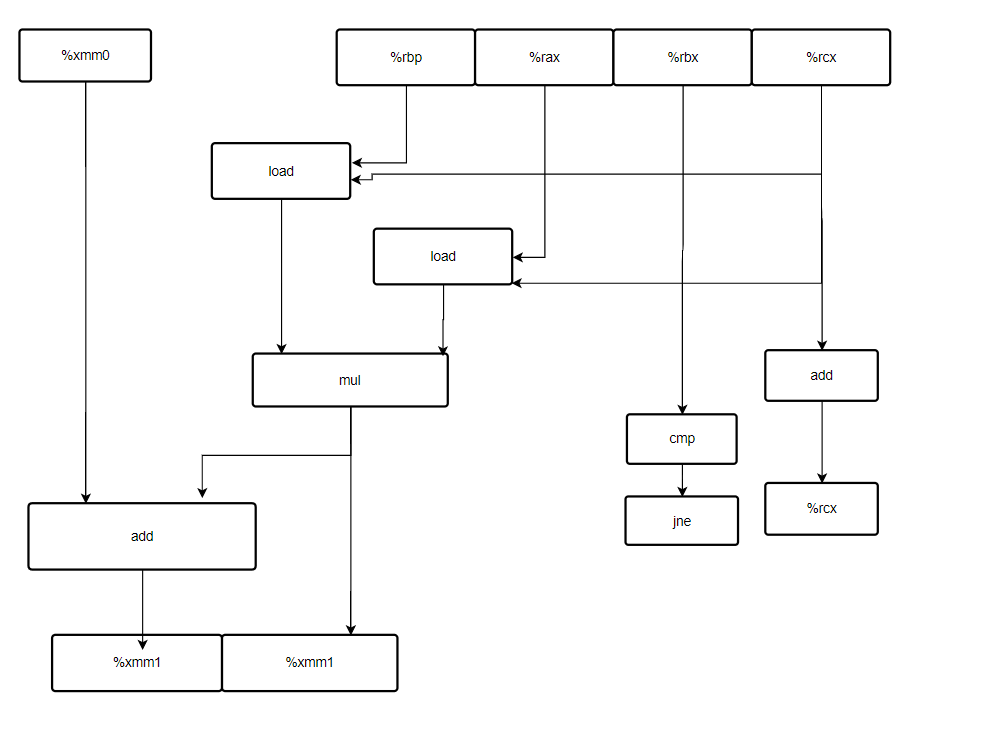
5.13

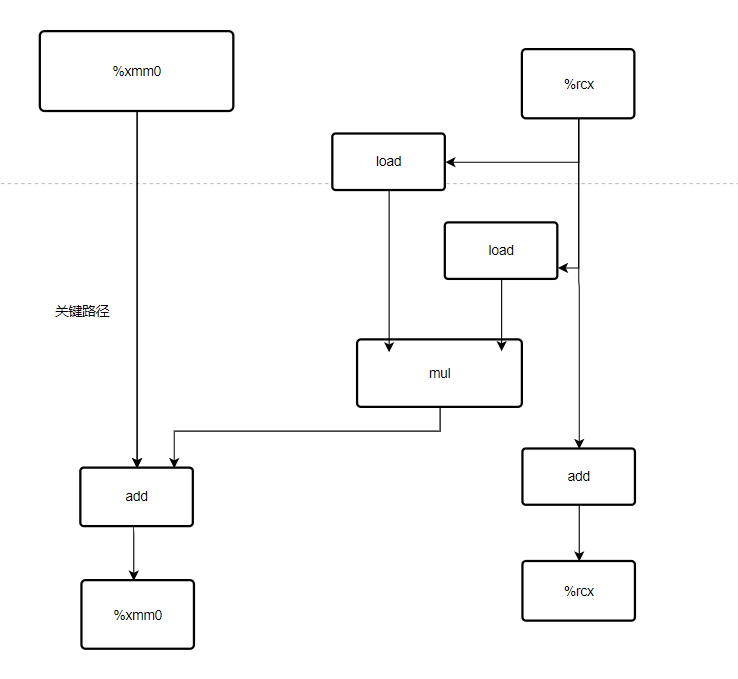
A.

操作抽象成数据流图：

（1）



（2）加法操作形成了限制性能的关键路径



B.下界为浮点加法的延迟界限，CPE为3.00

C.下界为整数加法的延迟界限，CPE为1.00

D.每次循环时乘法可以独立进行，不依赖上次结果，所以关键路径上只有浮点加法，而浮点数加法延迟为3个周期，所以CPE为3.00

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;

    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;

}

限制：加载器的数量为2，，一个时钟周期只能加载两个值

5.17

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

    size\_t K = sizeof(unsigned long);

    unsigned long word;

    unsigned char \*pword = (unsigned char \*)&word;

    size\_t cnt = 0;

    while (cnt < K) {                               //将K个c作为一个字

        \*pword++ = (unsigned char)c;

        cnt++;

    }

    size\_t i;

    unsigned char \*schar = s;

    //处理n很小的情况，逐个写为c

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

        \*schar++ = (unsigned char)c;

    }

    size\_t limit = n - K + 1;

    for (; i < limit && (int)limit > 0; i += K) {         //每次写K个c

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

        schar += K;

    }

     //剩下的部分逐个写为c

    for (; i < n; i++) {

        \*schar++ = (unsigned char)c;

    }

    return s;

}

5.19

void psum\_4\_1a(float a[], float p[], long n) {

    long i;

    float last\_val;

    last\_val = p[0] = a[0];

    float tmp0, tmp1, tmp2, tmp3 = 0;

    for (i = 0; i < n - 3; i += 4) {

        tmp0 = last\_val + a[i];

        tmp1 = tmp0 + a[i + 1];

        tmp2 = tmp1 + a[i + 2];

        tmp3 = tmp2 + a[i + 3];

        p[i] = tmp0;

        p[i + 1] = tmp1;

        p[i + 2] = tmp2;

        p[i + 3] = tmp3;

        last\_val = tmp3;

    }

    for (; i < n; i++) {

        last\_val += a[i];

        p[i] = last\_val;

    }

}