5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 代码 | Min | Max | Incr | Square |
| A | 1 | 91 | 90 | 90 |
| B | 91 | 1 | 90 | 90 |
| C | 1 | 1 | 90 | 90 |

A中，在确认i<max（x，y）之后，就不会再执行函数了。

B中，max执行一次，后面需要91次判断i是否大于min函数的返回值，最后一次判断的时候小于这个返回值了，于是终止循环，这也是为什么min函数比incr和square多循环一次的原因。

C中，因为有给low和high赋值，所以其实只在给它们赋值的时候调用过min&max。这个代码片段对i和high的大小比较了91次，调用incr和aquare90次。

5.4

A .O1和O2区别是O2比O1少了一个“store product at dest”，O1把xmm0作为一个临时值使用，而O2直接做了累加乘积。

B．实现了combine3的C语言代码，包括储存器名称都是一样的。

C．因为除了第一次，每一次循环开始和结束准备进行第二次循环的时候，dest读出的值和之前最后一次写进去的值是不改变的，因此可以合并指令，消除不必要的储存器引用。

5.10

A：在0<=i<=998的范围中，把a[i]更改为a[i+1]

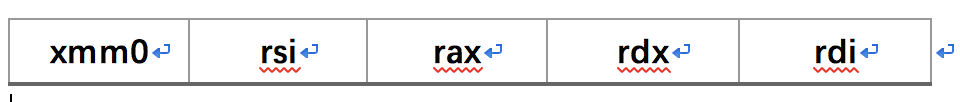
B：在1<=i<=998的范围中，把a[i]更改为a[0]也就是0

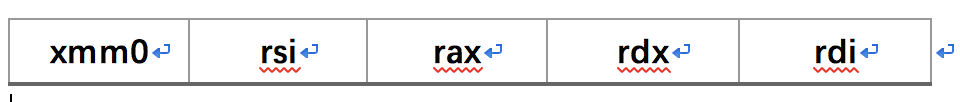
C：第一种是直接赋值，第二种在赋值的时候，需要用到之前迭代时更改过的值，所以有了一个存储的过程。

D：CPE是2，在0<=i<=998的范围中，不改变a[i]

5.11

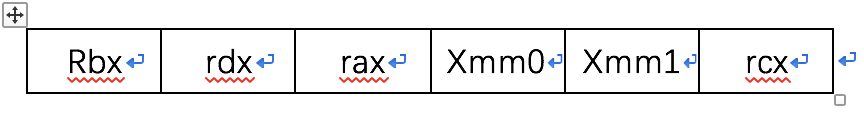
|  |
| --- |
| Load |
| Load |
| Addss |
| S\_addr |
| S\_data |
| addq |
| cmpq |
| jg |



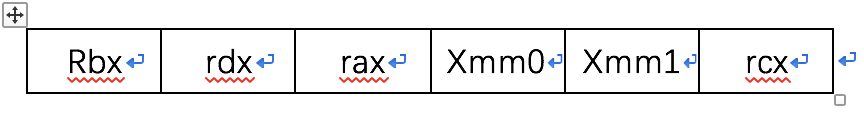


需要用到读和写，这两个都会占用时钟周期。

5.15



|  |
| --- |
| Load |
| Load |
| mulss |
| Addss |
| Addq |
| cmpq |
| jg |



关键路径

Load2

Load1

addq

addss

mul

B：由浮点数加法的延迟，CPE的下界应该是3。P344

C：由整数加法的延迟，CPE的下界应该是1。P344

D：每次循环时的乘法需要的数据不用依赖上一次乘法的结果所得出来的数据，所以他们可以各自独立进行。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addss。浮点数加法的延迟为3个周期，所以CPE为3.00。