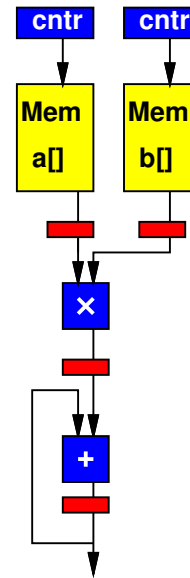


1. When the adder is adding $a[0]*b[0]$ into the sum,
 - (a) For what i is the multiplier performing $a[i]*b[i]$?
 - (b) For what i is the memory looking up $a[i]$ and $b[i]$?



Running:

```
for (i=0;i<MAX;i++)
    c+=a[i]*b[i];
```

2. Below is the assembly for the above loop body. Schedule this loop body onto a VLIW datapath with two ALUs, a multiplier, a branch unit, and a ld/st unit. (destination register is last – so sub r2,r1,r3 is $r3=r2-r1$)

```
// r1 -- i; r2 -- MAX
top: bzneq r3, end // branch if zero or negative
    addi r4,#4,r4 //&(a[i]) from &(a[i-1])
    addi r5,#4,r5 //&(b[i]) from &(b[i-1])
    ld r4,r6 // r6=a[i]
    ld r5,r7 // r7=b[i]
    mul r6,r7,r7 // r7=a[i]*b[i]
    add r7,r8,r8 // r8=c+=a[i]*b[i]
    addi r1,#1,r1 // i++
    sub r2,r1,r3 // r3=MAX-i
    br top
end:
```

Cycle	Branch	ALU	ALU	Multiply	Ld/St
0					
1					
2					
3					
4					
5					
6					

3. Schedule the following (slightly different) loop body onto a VLIW datapath with two ALUs, a multiplier, a branch unit, and a ld/st unit.

```
// r1 -- i; r2 -- MAX
top: bzneq r3, end // branch if zero or negative
    add r8,r9,r9 // r9=c+=a[i]*b[i]
    mul r6,r7,r8 // r8=a[i+1]*b[i+1]
    ld r4,r6 // r6=a[i+2]
    ld r5,r7 // r7=b[i+2]
    addi r4,#4,r4 //&(a[i+3]) from &(a[i+2])
    addi r5,#4,r5 //&(b[i+3]) from &(b[i+2])
    addi r1,#1,r1 // i++
    sub r2,r1,r3 // r3=MAX-i
    br top
```

end:

Cycle	Branch	ALU	ALU	Multiply	Ld/St
0					
1					
2					
3					
4					
5					
6					

4. How many operators of each type are needed in order to achieve the target II based on resource bound limits alone?

```
for (xptr=&x;xptr<XMAX;xptr++) { // count < as an add, ignore branch for this one
    // x[i]=*xptr
    yptr++; zptr++;
    res[i]=sqrt(x[i]*x[i]+y[i]*y[i]+z[i]*z[i]);
}
```

For which, we will break down the loop body into the following psuedo-assembly-level operations:

```
<XMAX; xptr++; yptr++; zptr++; ld x; ld y; ld z; x[i]2; y[i]2; z[i]2; x[i]2+y[i]2; +z[i]2;
sqrt; st res
```

Target	Branch	ALU	Multiply	Read Ports	Write Ports	incr	Sqrt
II=1							
II=2							
II=3							