Daniel McDonough
4/10/19
CS4515
HW5

4.9)

Consider the following code, which multiplies two vectors that contain single-precision complex values:

```
for (i=0;i <300;i++) {
        c_re[i] = a_re[i] * b_re[i] - a_im[i] * b_im[i];
        c_im[i] = a_re[i] * b_im[i] + a_im[i] * b_re[i];
}
```

Assume that the processor runs at 700 MHz and has a maximum vector length of 64. The load/store unit has a start-up overhead of 15 cycles; the multiply unit, 8 cycles; and the add/subtract unit, 5 cycles.

a) What is the arithmetic intensity of this kernel? Justify your answer.

There are four variables being used loaded each loop:

- a_re[i]
- b_re[i]
- a_im[i]
- b_im[i]

There are 2 variables being written in each loop:

- *c_re[i]*
- *c_im[i]*

There are 6 operations per loop:

- ** (4)*
- *+ (1)*
- *- (1)*

Therefore the arithmetic intensity of the loop is (4+2) / 6 = 1

b) Convert this loop into RV64V assembly code using strip mining.

Based on the code shown on p. 296 of Hennessey and Patterson

vsetdcfg	2 DP FP	Enable 2 64b Fl. Pt. registers
vld	v4,x4	load vector *c_re*
vld	v5,x5	load vector *c_im*
setvl (loop)	t0,a0	vl = t0 = min(mvl,n)
slli	tl,t0,3	tl = vl * 8
vld	v0,x0	load vector *a_re*
add	x0,x0,t1	increment pointer to *a_re* by vl*8
vld	v1,x1	load vector *b_re*
add	x1,x1,t1	increment pointer to *b_re by vl*8*
vld	v2,x2	load vector *b_im*
add	x2,x2,t1	increment pointer to *b_im by vl*8*
vmul	v0,v2,v0	a_re * *b_im*
vld	v3,x3	load vector *a_im*
add	x3,x3,t1	increment pointer to *a_im by vl*8*
vmul	v3,v1,v1	*a_im * b_re*
vmul	v3,v2,v2	*a_im * b_im*
vmul	v0,v1,v3	a_re * b_re
vadd	v0,v1,v0	*(a_re * b_im) - (a_im * b_re)*
vsub	v3,v2,v1	*(a_re * b_re) - (a_im * b_im)*
add	x4,x4,t1	increment pointer to c_*re* by vl*8
add	x5,x5,t1	increment pointer to c_*im* by vl*8
vst	v1,x4	store the value of *(a_re * b_re) - (a_im * b_im) in c_re*
vst	v0,x5	store the value of *(a_re * b_im) - (a_im * b_re) in c_im*
bnez	a0,loop	repeat if n != 0
vdisable		disable vector registers

c) Assuming chaining and a single memory pipeline, how many chimes are required? How many clock cycles are required per complex result value, including start-up overhead?

vld	vld	load a_re and b_re
vmul	vld	a_re * b_re and load a_im
vld	vmul	load b_im and a_im*b_im
vsub	vst	do subtraction and store into c_re
vmul	vld	a_re*b_im and load a_re
vmul	vld	a_im*b_re and load b_re
vadd	vst	do addition and store into c_im

Here **6 chimes** are required, with one extra overhead for the initial startup for the loading of a_re and b_re.

Assume a maximum vector length of 64. The load/store unit has a start-up overhead of 15 cycles; the multiply unit, 8 cycles; and the add/subtract unit, 5 cycles.

Therefore total cycles per iteration =

chimes * elements + overhead

Here the overhead is the first loop:

6 loads * 15 cycles + 4 multiplication * 8 cycles + 2 add/sub * 4 cycles = 132

So total cycles per iteration = 6 * 64 + 132 = 516

Cycles per result = 516/(64 * 2 (because 2 vectors are calculated)) = 4