Review:
Stride, Memory Banks,
Formula for number banks vs memory latency in clocks

Today:
7.4 Model of performance
4 pieces
$T_{element}$. the time for each vector operation in the loop to process one element, ignoring the start-up costs. This time depends only on the execution of vector instructions.
The overhead for each strip-mined block of vector instructions. 2 parts:
T_{loop}, the cost of executing the scalar code for strip mining of each block
T_{start} the vector start-up cost for each block, .
T_{base} the overhead from computing the starting addresses and setting up the vector control. This time consists solely of scalar overhead instructions. This occurs once for the entire vector operation.

These components can be used to state the total running time for a vector sequence operating on a vector of length n, which we will call T_n:

$$T_n = T_{base} + \text{ceiling}(\frac{n}{MVL}) * (T_{loop} + T_{start}) + n * T_{element}$$

The values of T_{start} and T_{loop} are both compiler and machine dependent, while the value of $T_{element}$ depends mainly on the hardware. The exact vector sequence affects all three values; the effect on $T_{element}$ is probably the most pronounced, with T_{start} and T_{loop} less affected.

R_∞—The MFLOPS rate on an infinite-length vector.
(R_n is the MFLOPS rate for a vector of length n.)
$N_{1/2}$—The vector length needed to reach one-half of R_∞.
(This is a good measure of the impact of T_{base}.)
N_v—The vector length needed to make vector mode faster than scalar mode.
(This measures $T_{base}+T_{loop}+T_{start}$ and the speed of scalars relative to vectors.)

Chaining

What about data hazards between vector instructions?

```
MULTV   V1,V2,V3
ADDV    V4,V1,V5
```

Time is

$T_{element}$ * Vector length + Start-up $time_{ADDV}$ + stall time + Start-up_{MULTV}
= 2 * Vector length + 6 + 4 + 7
= 2 * Vector length + 17

ADDV waits for MULTV to complete, just as if scalar isntruction

Suppose treat each vector register as 64 invididual elements vs monolith?

Break:
Evening Midterm 7-9PM in room change: 608-7?
David Green tape: Friday April 20

Compiler Technology for Vectors

```
          do 10 i=1,100
S1           A(i+1) = A(i) + B(i)
S2           B(i+1) = B(i) + A(i+1)
10        continue
```

1. S1 uses a value computed by S1 in an earlier iteration. This is true for S1 since iteration i+1 uses the value `A(i)` that was computed in iteration i as `A(i+1)`. The same is true of S2 for `B(i)` and `B(i+1)`.
2. S1 uses a value computed by S2 in an earlier iteration. This is true since S1 uses the value of `B` in iteration i+1 that is computed by S2 in iteration i.
3. S2 uses a value computed by S1 in the same iteration. This is true for the value `A(i+1)`.

1 or 2 (loop carried dependence)=> not vectorizable
3 => chained

a *recurrence*, does contain a loop-carried dependence on Y:

```
   do 10 i=2,100
Y(i) = Y(i-1) + Y(i)
10 continue
```

Effectiveness of Vectorization
p. 376, Figures 7.12 and 7.13
Programs must change

Can be vectorized by hand:

Pitfalls

Pitfall: Concentrating on peak performance and ignoring start-up overhead.

Pitfall: Increasing vector performance, without comparable increases in scalar performance.

p. 391 Fig 7.17 Livermore Loops: same micro, slower scalar
Smaller harmonic Mean

*Fallacy: The scalar performance of the **best** supercomputers is low.*

Spice: CPI 2.1 MIPS vs 4.1 CRAY; IC 738 MIPS vs 665 Cray

Fallacy: You can get vector performance without providing memory bandwidth.

SAXPY: 1.5 Memory ref per FL. PT. Operations

Discussion:

7.15 [Discussion] Some proponents of vector machines might argue that the vector processors have provided the best path to ever-increasing amounts of computer power by focusing their attention on boosting peak vector performance. Others would argue that the emphasis on peak performance is misplaced because an increasing percentage of the programs are dominated by nonvector performance. (Remember Amdahl's Law?) The proponents would respond that programmers should work to make their programs vectorizable. What do you think about this argument?

7.16 [Discussion] Consider the points raised in the Concluding Remarks (Section 7.9). This topic—the relative advantages of pipelined scalar machines versus FP vector machines—is the source of much debate in the early 1990s. What advantages do you see for each side? What would you do in this situation?