April 24

- 3 Classes to Go!
- Final Exam Saturday May 5 from 2 to 5pm (12 Days!)
- Matrix Multiply Example

Matrix Multiply

- A VERY common operation in scientific programs
- Multiply a LxM matrix by an MxN matrix to get an LxN matrix result
- This requires L*N inner products each requiring M * and +
- So 2*L*M*N floating point operations
- Definitely a FLOATING POINT INTENSIVE application
- L=M=N=100, 2 Million floating point operations

Matrix Multiply

```
const int L = 2;
const int M = 3;
const int N = 4;
void mm(double A[L][M], double B[M][N], double C[L][N])
{
    for(int i=0; i<L; i++)
        for(int j=0; j<N; j++) {
            double sum = 0.0;
            for(int k=0; k<M; k++)
                sum = sum + A[i][k] * B[k][j];
            C[i][j] = sum;
        }
}
```

Matrix Memory Layout

Our memory is a 1D array of bytes

How can we put a 2D thing in a 1D memory?

double A[2][3];

0 0	0 1	0 2
1 0	1 1	1 2

Row Major

0 0
0 1
0 2
1 0
1 1
1 2

addr = base+(i*3+j)*8

Column Major

0 0
1 0
0 1
1 1
0 2
1 2

addr = base + (i + j*2)*8

Where does the time go?

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: mul $t1, i, M
    add $t1, $t1, k
    mul $t1, $t1, 8
    add $t1, $t1, A
    l.d $f1, 0($t1)
    mul $t2, k, N
    add $t2, $t2, j
    mul $t2, $t2, 8
    add $t2, $t2, B
    l.d $f2, 0($t2)
```

```
mul.d $f3, $f1, $f2
add.d $f4, $f4, $f3
add k, k, 1
slt $t0, k, M
bne $t0, $zero, L1
```

Change Index * to +

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: l.d $f1, 0($t1)
    add $t1, $t1, AColStep
    l.d $f2, 0($t2)
    add $t2, $t2, BRowStep

    mul.d $f3, $f1, $f2
    add.d $f4, $f4, $f3
    add k, k, 1
    slt $t0, k, M
    bne $t0, $zero, L1
```

AColStep = 8

BRowStep = 8 * N

Eliminate k, use an address instead

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: l.d $f1, 0($t1)
    add $t1, $t1, AColStep
    l.d $f2, 0($t2)
    add $t2, $t2, BRowStep

    mul.d $f3, $f1, $f2
    add.d $f4, $f4, $f3
    bne $t1, LastA, L1
```

We made it faster

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: l.d $f1, 0($t1)
    add $t1, $t1, AColStep
    l.d $f2, 0($t2)
    add $t2, $t2, BRowStep

    mul.d $f3, $f1, $f2
    add.d $f4, $f4, $f3
    bne $t1, LastA, L1
```

Now this is FAST! Only 7 instructions in the inner loop!

BUT...

When we try it on big matrices it slows way down.

Whas Up?

Now where is the time?

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: l.d $f1, 0($t1)
    add $t1, $t1, AColStep
    l.d $f2, 0($t2)
    add $t2, $t2, BRowStep

    mul.d $f3, $f1, $f2
    add.d $f4, $f4, $f3
    bne $t1, LastA, L1
```

lots of time wasted here!

possibly a little stall right here

Why?

The inner loop takes all the time

```
for(int k=0; k<M; k++)
    sum = sum + A[i][k] * B[k][j];
```

```
L1: l.d $f1, 0($t1)
    add $t1, $t1, AColStep
    l.d $f2, 0($t2)
    add $t2, $t2, BRowStep

    mul.d $f3, $f1, $f2
    add.d $f4, $f4, $f3
    bne $t1, LastA, L1
```

This load usually hits (maybe 3 of 4)

This load always misses!

2