ECE 571 – Advanced Microprocessor-Based Design Lecture 9

Vince Weaver

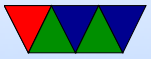
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

16 February 2016

Announcements

- HW#4 Posted, Branch prediction

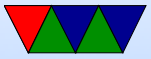


Notes about HW#3

- Energy

	sleep	stream	MMM	iozone
cores	0.05J	125.8J	20.17J	3.94J
gpu	0J	0J	0J	0J
package	30.8J	211J	30J	31.5J
dram	6.52J	25.4J	1.17J	6.18J
time	10s	9.5s	1.3s	8.38s

- Power

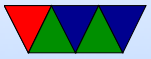


	sleep	stream	MMM	iozone
cores	0.005W	13.2W	15.5W	0.47W
gpu	0W	0J	0J	0J
package	3.1W	22.2W	23.0W	3.76W
dram	0.65W	2.67W	0.9W	0.74W
time	10s	9.5s	1.3s	8.38s

What does the various things exercise?

sleep? stream? MMM? iozone?

- It's a i7-4770, 84W TDP, 22nm, 4-core (8 thread)
- equake_l

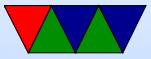


threads	time	E	ED	ED2
1	168.5	2836	476k	80M
2	137.1	3155	432k	59M
4	131.8	4174	550k	72M
8	134.2	4538	608k	81M

fastest run time? Some have it with 4 vs 8. Think it's 4 cores, 8 threads.

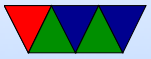
interesting with values so close, some people got 2 instead of 1 on best ED.

- Why doesn't it scale better? Get whole thesis out of it.

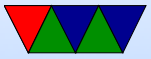


Would think it would.

How would you find out why it doesn't scale?

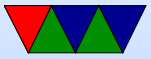


Oh No, More Caches!



Other Cache Types

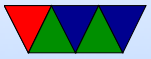
- Victim Cache – store last few evicted blocks in case brought back in again, mitigate smaller associativity
- Assist Cache – prefetch into small cache, avoid problem where prefetch kicks out good values
- Trace Cache – store predecoded program traces instead of (or in addition to) instruction cache



Virtual vs Physical Addressing

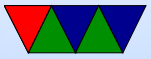
Programs operate on Virtual addresses.

- PIPT, PIVT (Physical Index, Physical/Virt Tagged) – easiest but requires TLB lookup to translate in critical path
- VIPT, VIVT (Virtual Index, Physical/Virt Tagged) – No need for TLB lookup, but can have aliasing between processes. Can use page coloring, OS support, or ASID (address space id) to keep things separate



Cache Miss Types

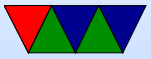
- Compulsory (Cold) — miss because first time seen
- Capacity — wouldn't have been a miss with larger cache
- Conflict — miss caused by conflict with another address (would not have been miss with fully assoc cache)
- Coherence — miss caused by other processor



Fixing Compulsory Misses

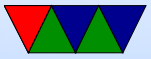
Prefetching

- Hardware Prefetchers – very good on modern machines. Automatically bring in nearby cachelines.
- Software – loading values before needed
  also special instructions available
- Large-blocksize of caches. A load brings in all nearby values in the rest of the block.



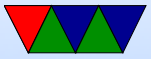
Fixing Capacity Misses

- Build Bigger Caches



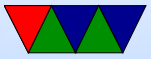
Fixing Conflict Misses

- More Ways in Cache
- Victim Cache
- Code/Variable Alignment, Cache Conscious Data Placement



Fixing Coherence Misses

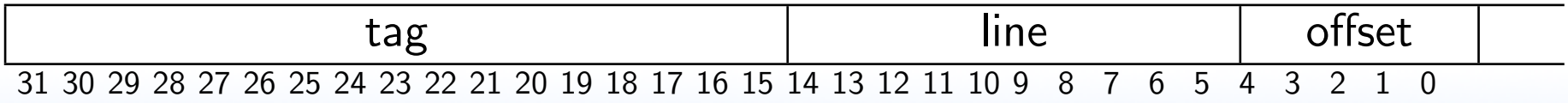
- False Sharing – independent values in a cache line being accessed by multiple cores

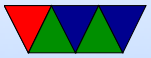


Cache Example 1 – Finding the Parameters

32kB cache (2^{15}), direct mapped (2^0)
32 Byte linesize (2^5), 32-bit address size (2^{32})

offset = $log_2(linesize)$ = 5 bits
lines = $log_2((cachesize/ways)/linesize)$ = 1024 lines (10 bits)
tag = addresssize - (offset bits + line bits) = 17 bits

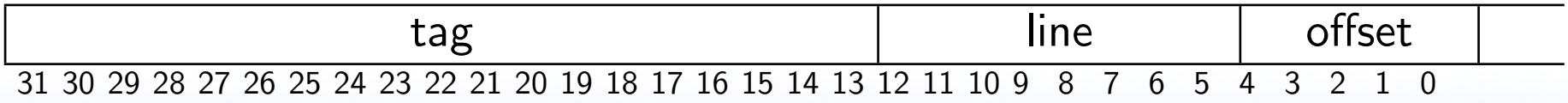


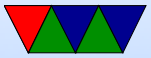


Cache Example 2 – Finding the Parameters

32kB cache (2^{15}), 4-way (2^2)
32 Byte linesize (2^5), 32-bit address size (2^{32})

offset = $log_2(linesize)$ = 5 bits
lines = $log_2((cachesize/ways)/linesize)$ = 256 lines (8 bits)
tag = addresssize - (offset bits + line bits) = 19 bits

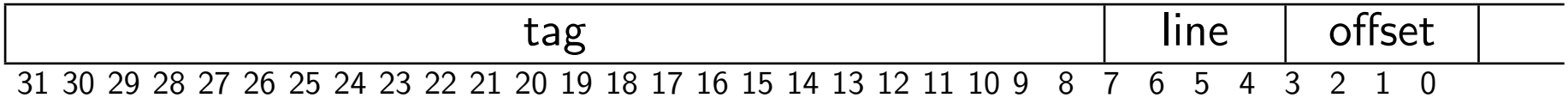


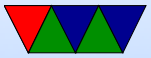


Cache Example

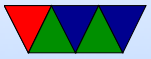
512 Byte cache, 2-Way Set Associative, with 16 byte lines, LRU replacement.

24-bit tag, 16 lines (4 bits), 4-bit offset.





Cache Example 1

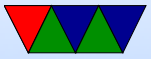


Cache Example – Instruction 1

```
ldb r1, 0x00000000
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

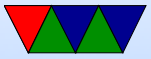


Cache Example – Instruction 2

```
ldb r1, 0x00000001
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Hit

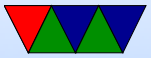


Cache Example – Instruction 3

```
ldb r1, 0x00000010
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	1	0	0	0000 00	0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

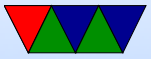


Cache Example – Instruction 4

```
ldb r1, 0x80000010
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	1	0	1	0000 00	1	0	0	8000 00
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

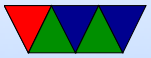


Cache Example – Instruction 5

```
ldb r1, 0xC0000010
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	1	0	0	C000 00	1	0	1	8000 00
2	0				0			
3	0				0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold (never in cache previously)

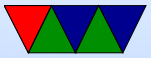


Cache Example – Instruction 6

```
ldb r1, 0xC0000002
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	1	0000 00	1	0	0	c000 00
1	1	0	0	C000 00	1	0	1	8000 00
2	0				0			
3	0				0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

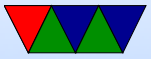


Cache Example – Instruction 7

```
ldb r1, 0x00000010
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	1	0000 00	1	0	0	c000 00
1	1	0	1	C000 00	1	0	0	0000 00
2	0				0			
3	0				0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Conflict

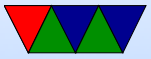


Cache Example – Instruction 8

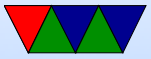
```
stb r1, 0x00000005
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	1	0	0000 00	1	0	1	c000 00
1	1	0	1	C000 00	1	0	0	0000 00
2	0				0			
3	0				0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Hit



Cache Example 2

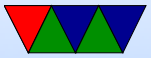


Cache Example – Instruction 1

```
ldrb r1, 0x00000000
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

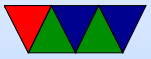


Cache Example – Instruction 2

```
str r1, 0x00000001
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 00	0			
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Hit

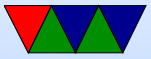


Cache Example – Instruction 3

```
strb r1, 0x00000105
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	1	0000 00	1	1	0	0000 01
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

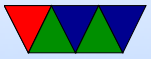


Cache Example – Instruction 4

```
ldr r1, 0x00000206
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	0	0000 02	1	1	1	0000 01
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

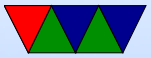


Cache Example – Instruction 5

```
ldb r1, 0x00000000
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	1	0000 02	1	0	0	0000 00
1	0				0			
2	0				0			
3	0				0			
4	0				0			
5	0				0			
...								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Conflict

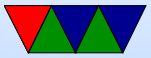


Cache Example – Instruction 6

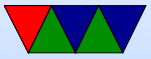
```
ldb r1, 0x00000030
```

	Way 0				Way 1			
line	V	D	LRU	Tag	V	D	LRU	Tag
0	1	0	1	0000 02	1	0	0	0000 00
1	0				0			
2	0				0			
3	1	0	0	0000 00	0			
4	0				0			
5	0				0			
. . .								
b	0				0			
c	0				0			
d	0				0			
e	0				0			
f	0				0			

Miss, Cold

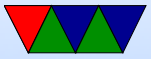


CMP Issues

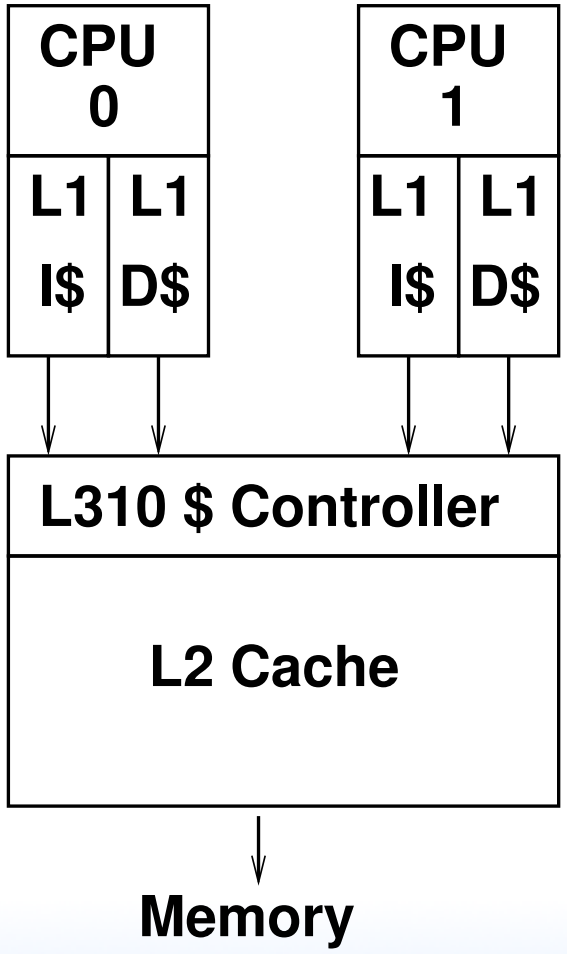


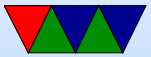
Cache Coherency

- Protocols such as MESI (Modified, Exclusive, Shared, Invalid)
- Snoopy vs Directory



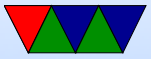
Cortex A9 Cache Layout





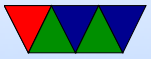
Cortex A9 Cache Layout

- OMAP4430 processor
- 32kB 4-way associative, separate L1-I and L1-D
  pseudo-round-robin or pseduo random replacement
  8-word line size (32B)
  critical-word first filling
  instruction: VIPT, data: PIPT
- Optional L2 cache controller
  Pandaboard has L310 L2 cache controller, 1MB 16-way

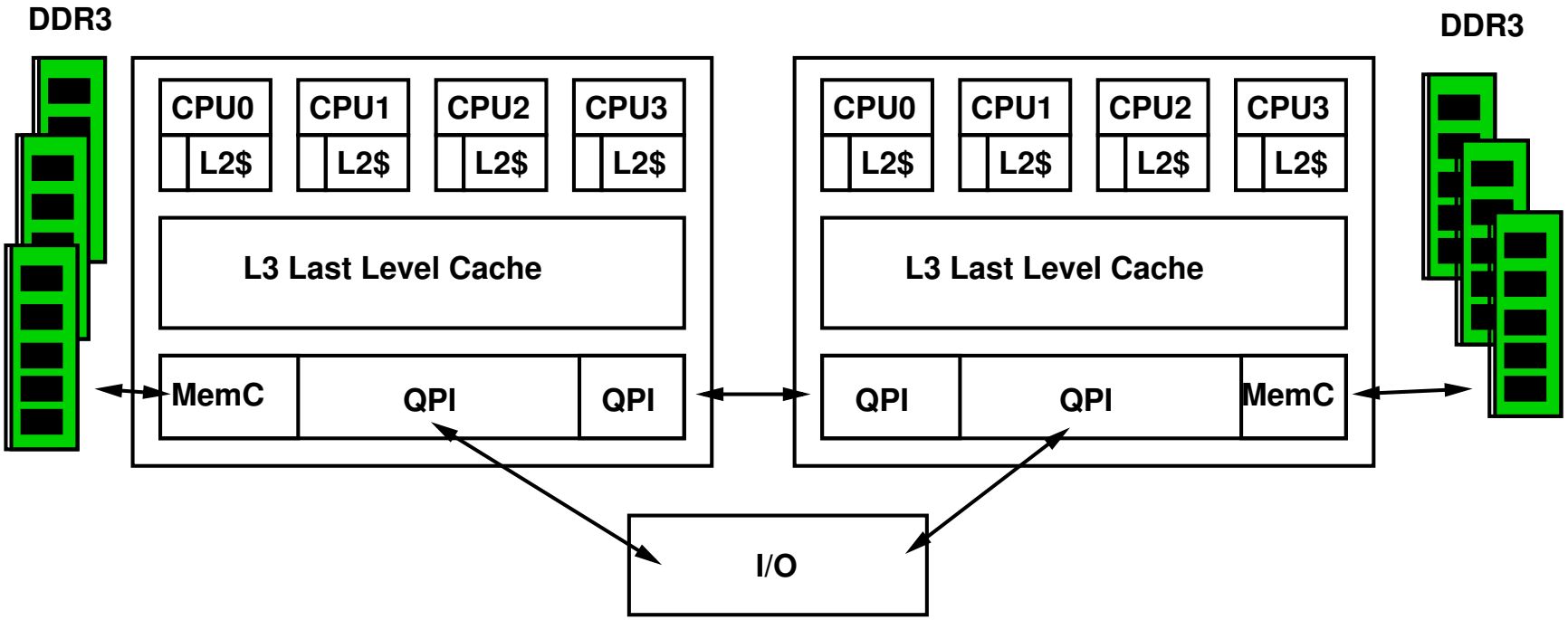


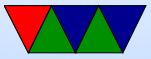
Optional prefetcher

- data cache reads/writes non-blocking, 4 outstanding misses
  write buffer: 4 64-bit, allowing write combining



SandyBridge Cache Layout



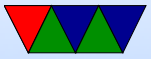


SandyBridge Cache Layout

- per core 32kB L1 I/D – 4 clocks
  64B/line, 8-Way
  (shared if hyper-threaded)
  writeback

- mOp cache? 1.5K instructions, 8-way, 6Mop/line
  Loop stream detector, can execute w/o accessing icache

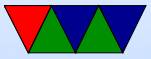
- per core 256kB L2 unified – 12 clocks



64B/line, 8-way
writeback. non-inclusive

- shared L3 1MB-20MB – 26-31 clocks
  64B/line. 12-way (varies)
  writeback, inclusive

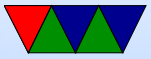
- various hw prefetchers operating



Cache Performance Measurement

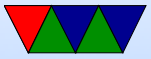
Matrix-Matrix multiply is the typical example.

Despite being a big deal in HPC, MMM happens in embedded world too.



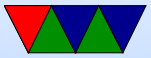
Naive Matrix-Matrix Multiply 1

```
#define MATRIX_SIZE 512
static double a[MATRIX_SIZE][MATRIX_SIZE];
static double b[MATRIX_SIZE][MATRIX_SIZE];
static double c[MATRIX_SIZE][MATRIX_SIZE];

for(j=0;j<MATRIX_SIZE;j++) {
    for(i=0;i<MATRIX_SIZE;i++) {
        for(k=0;k<MATRIX_SIZE;k++) {
            c[i][j]+=a[i][k]*b[k][j];
        }
    }
}
```

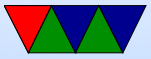


Naive Matrix-Matrix Multiply 1 – what's the issue?

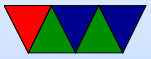
- Branch Misses?
- TLB Misses?
- ICache Misses?
- DCache Misses?
- L2 Cache Misses?



Naive Matrix-Matrix Multiply 1 – perf results

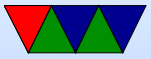


```
Matrix multiply sum: s=27665734022509.746094
 Performance counter stats for './matrix_multiply':

     11296.203614 task-clock                #    0.999 CPUs utilized
              20 context-switches          #    0.000 M/sec
               0 CPU-migrations            #    0.000 M/sec
           1,633 page-faults               #    0.000 M/sec
     9,032,356,979 cycles                  #    0.800 GHz
         6,547,102 stalled-cycles-frontend   #    0.07% frontend cycles id
     8,213,005,758 stalled-cycles-backend    #   90.93% backend  cycles id
     1,176,144,886 instructions            #    0.13  insns per cycle
                                           #    6.98  stalled cycles per
       137,651,296 branches                #   12.186 M/sec
           795,064 branch-misses           #    0.58% of all branches

      11.303802490 seconds time elapsed
```



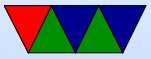
Naive Matrix-Matrix Multiply 1 – DCache results

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e cache-misses,cache-r
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply':

       171,786,441 cache-misses              #   42.072 % of all cache ref
       408,318,876 cache-references

      10.680664062 seconds time elapsed
```



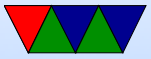
Naive Matrix-Matrix Multiply 1 – ICache results

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e l1-icache-load-misse
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply':

           536,719 l1-icache-load-misses
     1,174,927,869 instructions              #    0.00  insns per cycle

      12.203002930 seconds time elapsed

0.04% icache misses
```



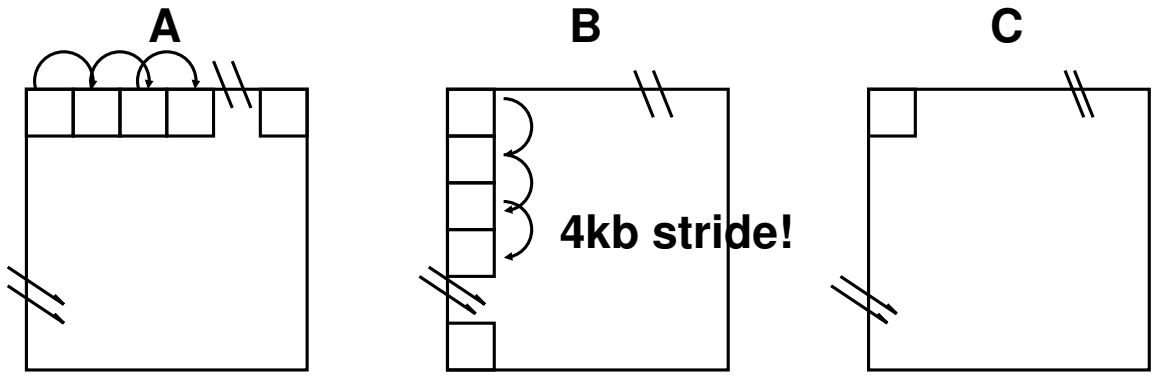
Naive Matrix-Matrix Multiply 1 – TLB Misses

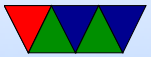
```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e dTLB-load-misses,dTL
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply':

       135,253,464 dTLB-load-misses
       135,253,464 dTLB-store-misses

      12.443572998 seconds time elapsed
```



Naive Matrix-Matrix Multiply 1 – What's the Issue?

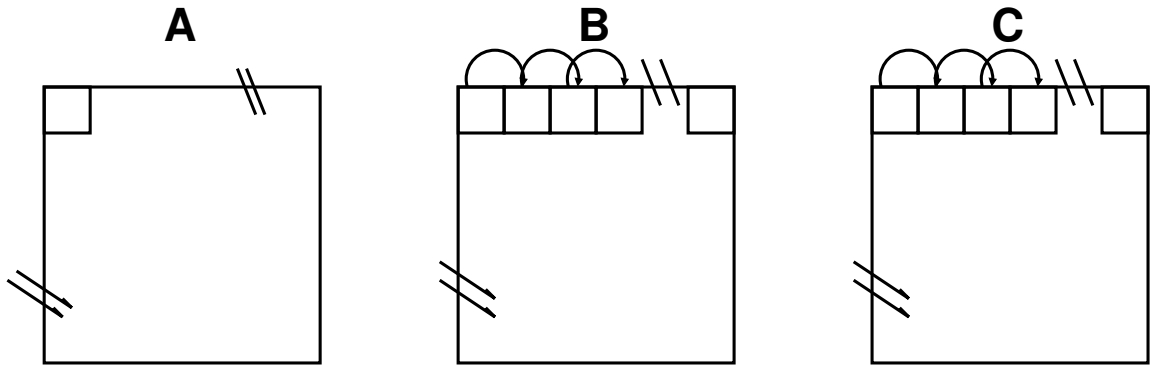
400M memory accesses about right (512x512x512x3)

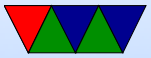




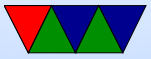
Switch the loop ordering

```
for(i=0;i<MATRIX_SIZE;i++) {
    for(k=0;k<MATRIX_SIZE;k++) {
        for(j=0;j<MATRIX_SIZE;j++) {
            c[i][j]+=a[i][k]*b[k][j];
        }
    }
}
```

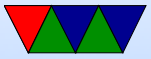




Naive Matrix-Matrix Multiply 2 – perf results

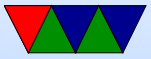


```
Matrix multiply sum: s=27665734022509.746094
 Performance counter stats for './matrix_multiply_swapped':

       3443.267822 task-clock                #    0.999 CPUs utilized
                 5 context-switches          #    0.000 M/sec
                 0 CPU-migrations            #    0.000 M/sec
             1,633 page-faults               #    0.000 M/sec
     2,849,573,010 cycles                    #    0.828 GHz
         2,913,607 stalled-cycles-frontend   #    0.10% frontend cycles id
     1,893,138,507 stalled-cycles-backend    #   66.44% backend  cycles id
       965,962,767 instructions              #    0.34  insns per cycle
                                             #    1.96  stalled cycles per
       136,649,964 branches                  #   39.686 M/sec
           553,643 branch-misses             #    0.41% of all branches

       3.447875977 seconds time elapsed
```



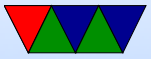
Naive Matrix-Matrix Multiply 2 – DCache results

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e cache-misses,cache-r
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply_swapped':

       38,628,043 cache-misses              #    9.409 % of all cache ref
      410,528,663 cache-references

      5.585754395 seconds time elapsed
```



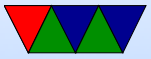
Naive Matrix-Matrix Multiply 2 – ICache results

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e l1-icache-load-misse
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply_swapped':

          254,041 l1-icache-load-misses
      964,335,795 instructions              #    0.00  insns per cycle

      4.245208740 seconds time elapsed

0.02%
```



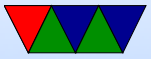
Naive Matrix-Matrix Multiply 1 – TLB Misses

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e dTLB-load-misses,dTL
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply_swapped':

          486,039 dTLB-load-misses
          486,039 dTLB-store-misses

       5.242126465 seconds time elapsed
```



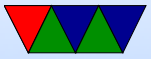
Other Ways to Optimize

- Tiling
- Parallelizing



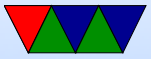
Use ATLAS/BLAS

```
cblas_dgemm(CblasRowMajor,
      CblasNoTrans,CblasNoTrans,
      512,512,512,
      1.0,(const double *)a,512,
      (const double *)b,512,
      1.0,(double *)c,512);
```



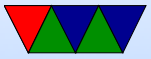
Matrix-Matrix Mul ATLAS – perf results

```
Matrix multiply sum: s=27665734022509.746094
 Performance counter stats for './matrix_multiply_atlas':

       1158.325193 task-clock                #    1.678 CPUs utilized
                12 context-switches          #    0.000 M/sec
                 1 CPU-migrations            #    0.000 M/sec
             2,017 page-faults               #    0.002 M/sec
       597,931,712 cycles                    #    0.516 GHz
         2,043,500 stalled-cycles-frontend   #    0.34% frontend cycles id
       258,860,537 stalled-cycles-backend    #   43.29% backend  cycles id
       519,715,833 instructions              #    0.87  insns per cycle
                                             #    0.50  stalled cycles per
        36,716,368 branches                  #   31.698 M/sec
           815,440 branch-misses             #    2.22% of all branches

       0.690429687 seconds time elapsed
```



Matrix-Matrix Mul ATLAS – DCache

```
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply_atlas':

       11,988,047 cache-misses             #    8.128 % of all cache ref
      147,494,664 cache-references

      0.598632813 seconds time elapsed
```



Matrix-Matrix Mul ATLAS – TLB

```
vince@arm:~/class/ece571/lecture10_code$ perf stat -e dTLB-load-misses ./m
Matrix multiply sum: s=27665734022509.746094

 Performance counter stats for './matrix_multiply_atlas':

           224,299 dTLB-load-misses

       0.755981446 seconds time elapsed
```