ECE 571 – Advanced Microprocessor-Based Design Lecture 9

Vince Weaver

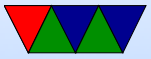
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

16 February 2016

Announcements

- HW#4 Posted, Branch prediction

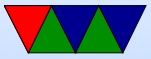


Notes about HW#3

- 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

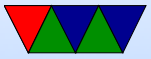


	sleep	stream	MMM	iozone
cores	0.005W	13.2W	15.5W	0.47W
gpu	0W	0J	0J	0J
package	3.1W	22.2W	23.0W	3.76W
dram	0.65W	2.67W	0.9W	0.74W
time	10s	9.5s	1.3s	8.38s

What does the various things exercise?

sleep? stream? MMM? iozone?

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

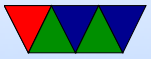


threads	time	E	ED	ED2
1	168.5	2836	476k	80M
2	137.1	3155	432k	59M
4	131.8	4174	550k	72M
8	134.2	4538	608k	81M

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

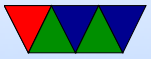
interesting with values so close, some people got 2 instead of 1 on best ED.

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

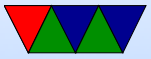


Would think it would.

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

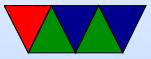


Oh No, More Caches!



Other Cache Types

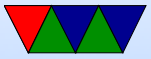
- Victim Cache – store last few evicted blocks in case brought back in again, mitigate smaller associativity
- Assist Cache – prefetch into small cache, avoid problem where prefetch kicks out good values
- Trace Cache – store predecoded program traces instead of (or in addition to) instruction cache



Virtual vs Physical Addressing

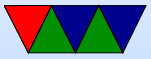
Programs operate on Virtual addresses.

- 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

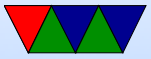
- Compulsory (Cold) — miss because first time seen
- Capacity — wouldn't have been a miss with larger cache
- Conflict — miss caused by conflict with another address (would not have been miss with fully assoc cache)
- Coherence — miss caused by other processor



Fixing Compulsory Misses

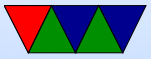
Prefetching

- 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.



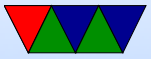
Fixing Capacity Misses

- Build Bigger Caches



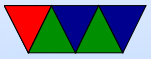
Fixing Conflict Misses

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



Fixing Coherence Misses

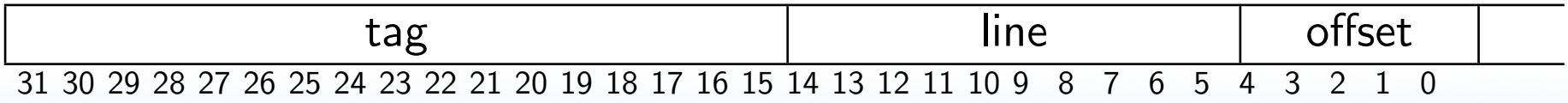
- False Sharing – independent values in a cache line being accessed by multiple cores

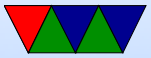


Cache Example 1 – Finding the Parameters

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

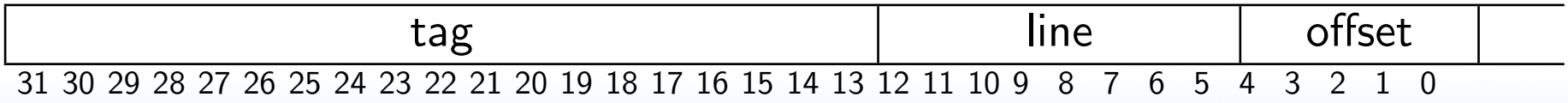


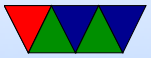


Cache Example 2 – Finding the Parameters

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

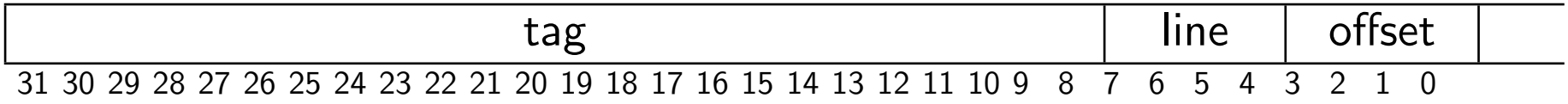


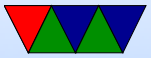


Cache Example

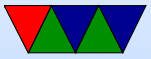
512 Byte cache, 2-Way Set Associative, with 16 byte lines, LRU replacement.

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





Cache Example 1

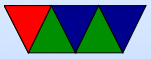


Cache Example – Instruction 1

```
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

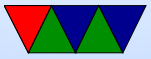


Cache Example – Instruction 2

```
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

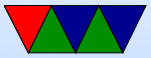


Cache Example – Instruction 3

```
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

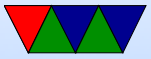


Cache Example – Instruction 4

```
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

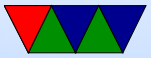


Cache Example – Instruction 5

```
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)

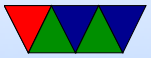


Cache Example – Instruction 6

```
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

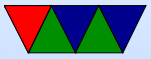


Cache Example – Instruction 7

```
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