ECE 571 – Advanced Microprocessor-Based Design Lecture 10

Vince Weaver

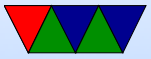
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

29 September 2022

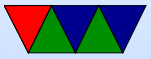
Announcements

- Don't forget HW#4. A little trickier than previous as you run on 3 different machines



HW#3 Review – The Benchmarks

- sleep – does nothing
- stream – stresses memory subsystem
- matrix – loads the CPU
- iozone – disk I/O



HW#3 Review – The System

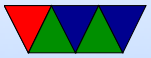
- Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
  20MB cache, 22nm, 90W TDP

  `https://ark.intel.com/content/www/us/en/ark/products/83359/intel-xeon-processor-e5-2640-v3-20m-ca`

  `html`

- 80GB DDR4 RAM

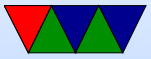
- Regular spinning hard drive



HW#3 Energy

Energy

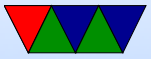
	sleep	stream	matrix	iozone
cores	0.00	0.00	0.0	0
sleep	203	337	228	46
ram	26	74	37	9.7
time	10.0s	6.3s	4.6s	1.3s



HW#3 Power

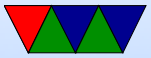
Power

	sleep	stream	matrix	iozone
cores	0.00	0.00	0.0	0
pkg	20	54	50	36
ram	2.6	12	8	8



HW#3 Notes

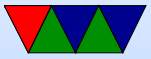
- highest power pkg: stream
- highest power dram: stream
- cores: always zero
  This is likely a bug with the Haswell-EP chips
  Intel slow to acknowledge this
- server class does not have integrated GPU
- guesses: old, virtual machine
- HPL (20k): pkg: 117W dram: 11W
- stream stresses memory. iozone, CPU is waiting?



HW#3 Energy-Delay

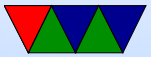
Energy-delay

	1	2	4	8	16	32	64
E	9k	6k	4.4k	3.6k	3.6k	3.9k	4.6k
time	169s	99s	59s	31s	28s	26s	33s
ED	1500k	614k	256k	111k	101k	101k	152k
ED2	257M	61M	14M	3.5M	2.8M	2.6M	5M
Power	53W	62W	75W	116W	129W	150W	134W

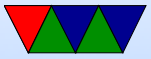


HW#3 Energy-Delay Discussion

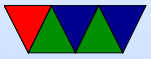
- interesting, half class got results where 32 best
- a) fastest time = 16/32 (close)
- b) lowest energy = 8/16
- c) lowest E-D = 16/32
- d) lowest E-D-D = 32
- e) scaling? only can show strong (problem size same)
  - Poorly written benchmark? (possible)
  - Not enough memory? (unlikely, benchmark from 2001)
- f) 32 threads, but only 16 cores



- g) TDP=90W for one package, but we have two



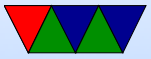
Oh No, More Caches!



Virtual vs Physical Addressing

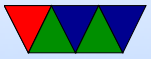
Programs operate on Virtual addresses.

- 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

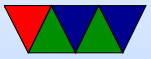
- Compulsory (Cold) — miss because first time seen
- Capacity — wouldn't have been a miss with larger cache
- Conflict — miss caused by conflict with another address (would not have been miss with fully assoc cache)
- Coherence — miss caused by other processor



Fixing Compulsory Misses

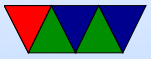
Prefetching

- 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.



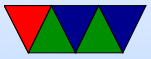
Fixing Capacity Misses

- Build Bigger Caches



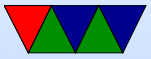
Fixing Conflict Misses

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



Fixing Coherence Misses

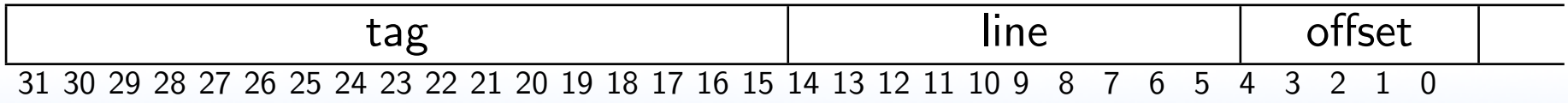
- False Sharing – independent values in a cache line being accessed by multiple cores

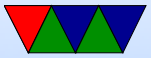


Cache Parameters Example 1

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

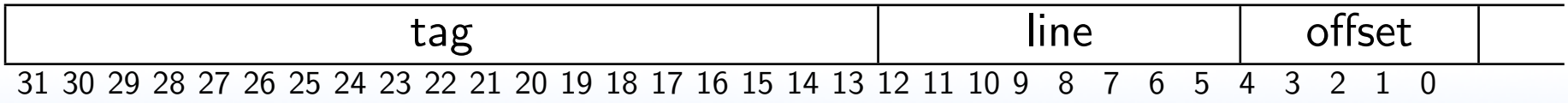


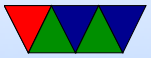


Cache Parameters Example 2

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

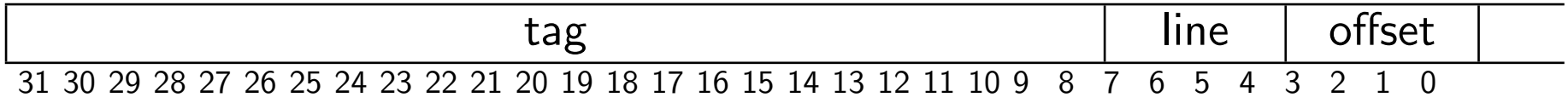


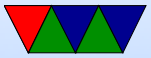


Cache Example

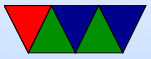
512 Byte cache, 2-Way Set Associative, with 16 byte lines, LRU replacement.

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





Cache Example 1

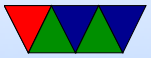


Cache Example – Instruction 1

```
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

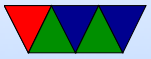


Cache Example – Instruction 2

```
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

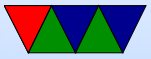


Cache Example – Instruction 3

```
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

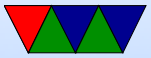


Cache Example – Instruction 4

```
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

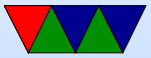


Cache Example – Instruction 5

```
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)

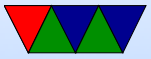


Cache Example – Instruction 6

```
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

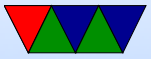


Cache Example – Instruction 7

```
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

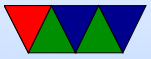


Cache Example – Instruction 8

```
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



Capacity vs Conflict Miss

- It's hard to tell on the fly what kind of miss
- For example: to know if cold, need to keep list of every address that's ever been in cache
- To know if it's capacity, need to know if it would have missed even in a fully associative cache
- Otherwise, it's a conflict miss