ECE 571 – Advanced Microprocessor-Based Design Lecture 9

Vince Weaver

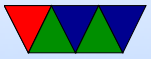
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

1 October 2019

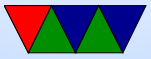
Announcements

- HW#4 was posted. About branch predictors
- Don't wait until last minute, is a bit more complex. Log into three machines.
- Any trouble with Jetson? Was a pain getting a working version of perf (perf_event is forwards compatible but not backwards compatible)



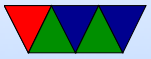
HW#3 Review

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

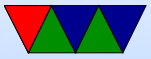
html



HW#3 Energy

Energy

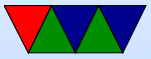
	sleep	stream	matrix	iozone
cores	0.00	0.00	0.0	0
pkg	205	376	30	3.13
ram	26	74	5	0.58
time	10.0	7.0	0.77s	0.084



HW#3 Power

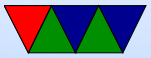
Power

	sleep	stream	matrix	iozone
cores	0.00	0.00	0.0	0
pkg	20	53.7	40	37
ram	2.6	10.6	6.5	6.9



HW#3 Notes

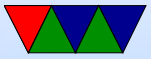
- highest power pkg: stream
- highest power dram: stream
- cores: always zero
  This is likely a bug with the Haswell-EP chips
  Intel slow to acknowledge this
- server class does not have integrated GPU
- HPL (20k): pkg: 117W dram: 11W
- stream stresses memory. iozone, CPU is waiting?



HW#3 Energy-Delay

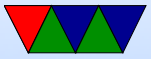
Energy-delay

	1	2	4	8	16	32
E	10.6k	7.6k	5.3k	4.3k	4.6k	5.1k
time	193s	105s	56s	39s	35s	37s
ED	2.0M	798k	297k	167k	161k	188k
ED2	395M	83.8M	16M	6.5M	5.6M	7.0M
Power	55W	72W	94W	110W	131W	137W

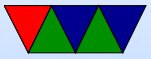


HW#3 Energy-Delay Discussion

- a) fastest time = 16
- b) lowest energy = 8
- c) lowest E-D = 16
- d) lowest E-D-D = 16
- e) scaling? only can show strong (problem size same)
  - Poorly written benchmark (possible)
  - Not enough memory (not really, this is a 2001 benchmark)
- f) 32 threads, but only 16 cores

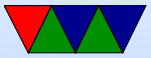


Oh No, More Caches!



Cache Miss Types

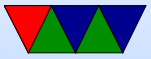
- Compulsory (Cold) — miss because first time seen
- Capacity — wouldn't have been a miss with larger cache
- Conflict — miss caused by conflict with another address (would not have been miss with fully assoc cache)
- Coherence — miss caused by other processor



Fixing Compulsory Misses

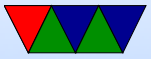
Prefetching

- 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.



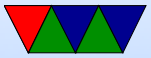
Fixing Capacity Misses

- Build Bigger Caches



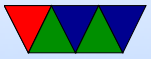
Fixing Conflict Misses

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



Fixing Coherence Misses

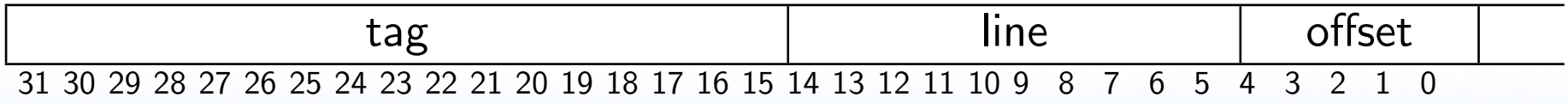
- False Sharing – independent values in a cache line being accessed by multiple cores

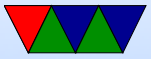


Cache Parameters Example 1

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

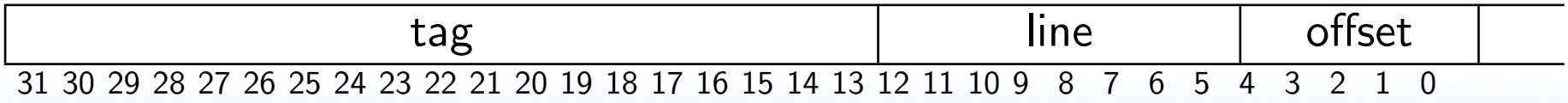


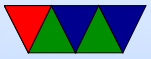


Cache Parameters Example 2

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

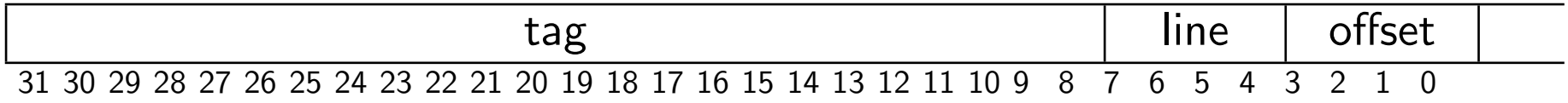


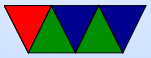


Cache Example

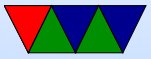
512 Byte cache, 2-Way Set Associative, with 16 byte lines, LRU replacement.

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





Cache Example 1

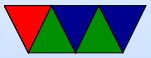


Cache Example – Instruction 1

```
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

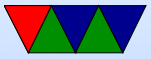


Cache Example – Instruction 2

```
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

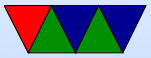


Cache Example – Instruction 3

```
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

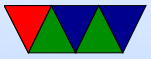


Cache Example – Instruction 4

```
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

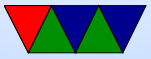


Cache Example – Instruction 5

```
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)

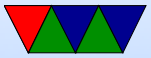


Cache Example – Instruction 6

```
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

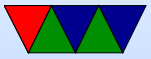


Cache Example – Instruction 7

```
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

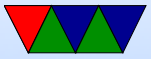


Cache Example – Instruction 8

```
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