CSE 560 Computer Systems Architecture

Cache

Why Caches?

Programs 101

C Code

```
int sum(int x, int y)
{
  int t = x+y;
  return t;
}
```

Generated IA32 Assembly

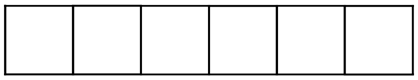
```
sum:
   pushl %ebp
   movl %esp,%ebp
   movl 12(%ebp),%eax
   addl 8(%ebp),%eax
   popl %ebp
   ret
```

High-level behavior: **Instructions that read from/write to memory…**

- Read data from memory (put in registers)
- Manipulate it
- Store it back to memory

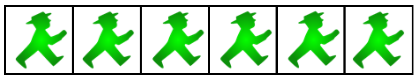
The Need for Speed

CPU Pipeline



The Need for Speed

CPU Pipeline

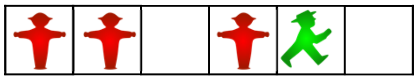


Instruction speeds:

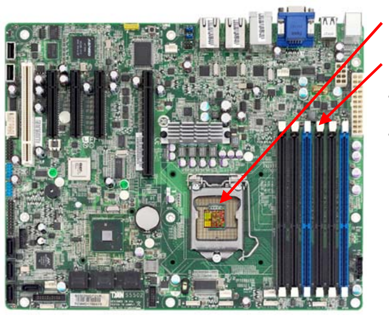
- **add,sub,shift**: 1 cycle
- **mult**: 3 cycles
- **load/store**: **100 cycles**
  off-chip 50(-70)ns
  2(-3) GHz processor → 0.5 ns clock

The Need for Speed

CPU Pipeline



What's the problem?



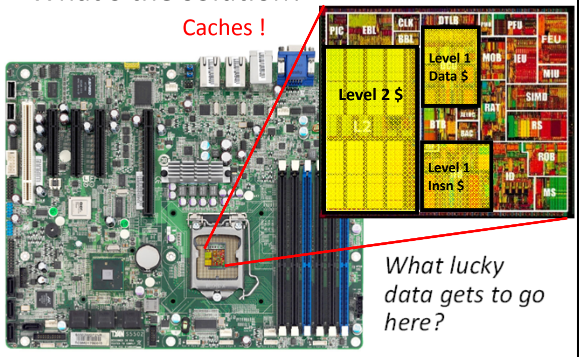
Processor

Main Memory
- too slow
- too far away

SandyBridge Motherboard, 2011
http://news.softpedia.com

What's the solution?

Caches !

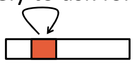


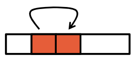
What lucky data gets to go here?

Intel Pentium 3, 1999

Locality Locality Locality

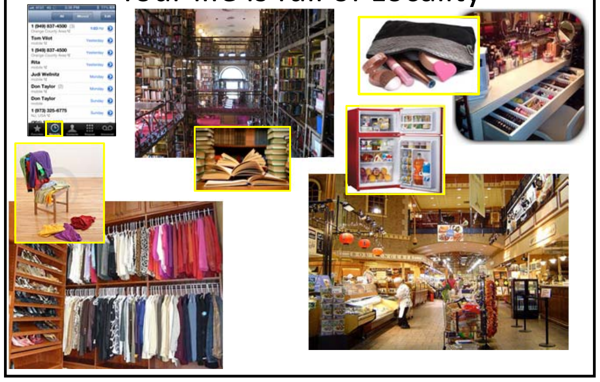
If you ask for something, you're likely to ask for:
- the same thing again soon
  → Temporal Locality
- something near that thing, soon
  → Spatial Locality



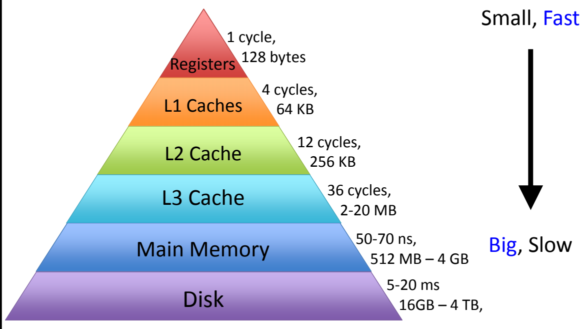


```
total = 0;
for (i = 0; i < n; i++)
    total += a[i];
return total;
```

Your life is full of Locality

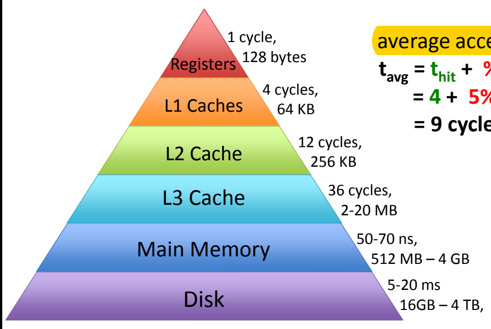


The Memory Hierarchy



Intel Haswell Processor, 2013

The Memory Hierarchy



average access time

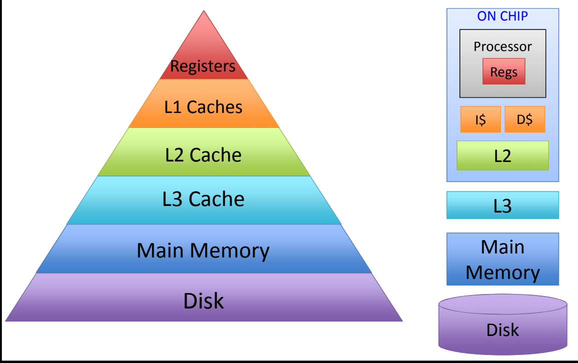
$t_{avg} = t_{hit} + \%_{miss} * t_{miss}$

$= 4 + 5\% \times 100$

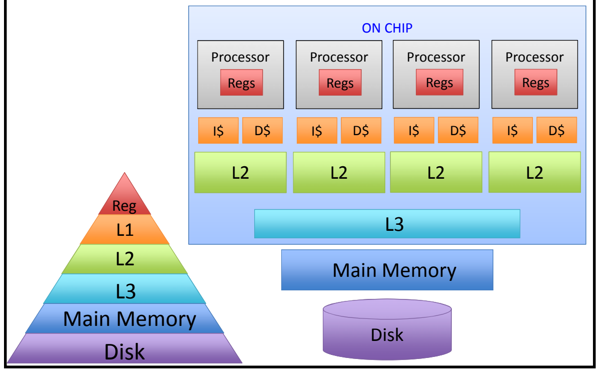
$= 9$ cycles

Intel Haswell Processor, 2013

The Memory Hierarchy



The Memory Hierarchy



Basic Cache Design

Direct Mapped Caches

16 Byte Memory

`load 0x1100 → r1`

- Byte-addressable memory
- 4 address bits → 16 bytes total
- b addr bits → 2^b bytes in memory

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

4-Byte, Direct Mapped Cache

CACHE

index	addr	data
00	xxxx	X
01	xxxx	X
10	xxxx	X
11	xxxx	X

- entry = row = **cache line** = **cache block**
- **Block Size:** 1 byte
- **Direct mapped:**
  - Each address mapped to specific cache block
  - 4 entries → 2 index bits (2^n → n bits)

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Least Significant Bits as Index

index
XXXX

CACHE

index	addr	data
00	0000	A
01	0001	B
10	0010	C
11	0011	D

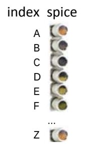
- Supports spatial locality

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Analogy to a Spice Rack

Spice Rack (Cache)



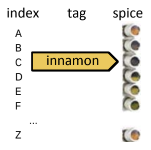
Spice Wall (Memory)

- Compared to your spice wall
  - Smaller
  - Faster
  - More costly (per oz.)

http://www.bedbathandbeyond.com

Analogy to a Spice Rack

Spice Rack (Cache)



Spice Wall (Memory)

- How do you know what's in the jar?
- Need labels

Tag = Ultra-minimalist label

4-Byte, Direct Mapped Cache

`tag|index`
`XXXX`

CACHE

index	tag	data
00	00	A
01	00	B
10	00	C
11	00	D

Tag: minimalist label/address

`address = tag + index`

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #1 of a 4-byte, DM Cache

`tag|index`
`XXXX`

CACHE

index	tag	data
00	00	0
01	00	0
10	00	0
11	00	0

Cache starts empty

`load 0x0000` `Hit?`

Lookup:
⇨ Index into $
⇨ Check tag

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

4-Byte, Direct Mapped Cache

CACHE

index	V	tag	data
00	0	xx	X
01	0	xx	X
10	0	xx	X
11	0	xx	X

One last tweak: **valid bit**

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #1 of a 4-byte, DM Cache

`tag|index`
`XXXX`

CACHE

index	V	tag	data
00	0	xx	X
01	0	xx	X
10	0	xx	X
11	0	xx	X

`load 0x1100` `Miss`

Lookup:
⇨ Index into $
⇨ Check tag
⇨ Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #1 of a 4-byte, DM Cache

tag|index
XXXX

CACHE

index	V	tag	data
00	1	11	N
01	0	xx	X
10	0	xx	X
11	0	xx	X

`load 0x1100` Miss

Lookup:
- Index into $
- Check tag
- Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #1 of a 4-byte, DM Cache

tag|index
XXXX

CACHE

index	V	tag	data
00	1	11	N
01	0	xx	X
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
. . .
load 0x1100   Hit!
```

Awesome!

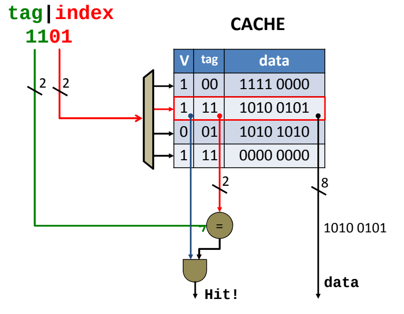
Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Block Diagram
4-entry, direct mapped Cache

tag|index
1101



Great!
Are we done?

27

Simulation #2: 4-byte, DM Cache

tag|index
XXXX

CACHE

index	V	tag	data
00	0	xx	X
01	0	xx	X
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101
load 0x0100
load 0x1100
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #2: 4-byte, DM Cache

tag|index
XXXX

CACHE

index	V	tag	data
00	1	11	N
01	0	xx	X
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101
load 0x0100
load 0x1100
```

Lookup:
- Index into $
- Check tag
- Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #2: 4-byte, DM Cache

tag|index
XXXX

CACHE

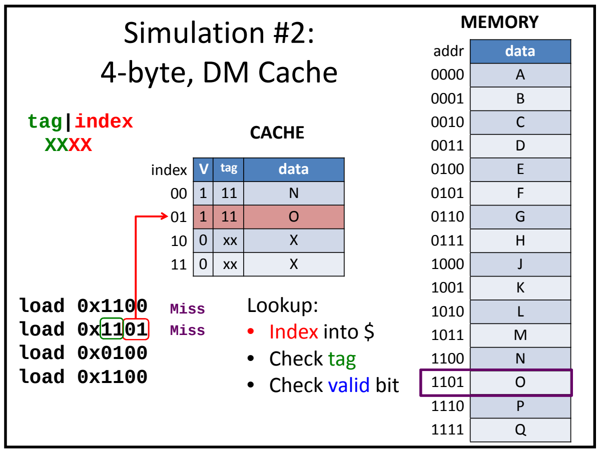
index	V	tag	data
00	1	11	N
01	0	xx	X
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101   Miss
load 0x0100
load 0x1100
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q



Simulation #2: 4-byte, DM Cache

tag|index
XXXX

MEMORY

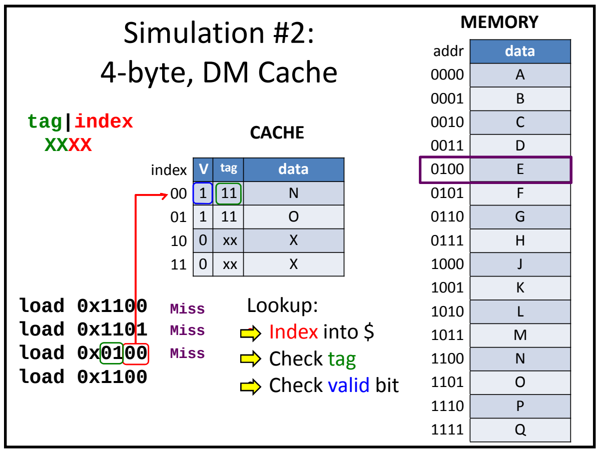
addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

CACHE

index	V	tag	data
00	1	11	N
01	1	11	O
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101   Miss
load 0x0100
load 0x1100
```

Lookup:
- Index into $
- Check tag
- Check valid bit



Simulation #2: 4-byte, DM Cache

tag|index
XXXX

MEMORY

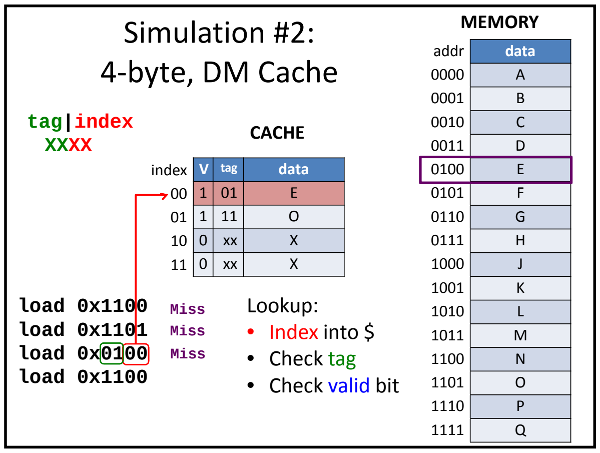
addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

CACHE

index	V	tag	data
00	1	11	N
01	1	11	O
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101   Miss
load 0x0100   Miss
load 0x1100
```

Lookup:
⇨ Index into $
⇨ Check tag
⇨ Check valid bit



Simulation #2: 4-byte, DM Cache

tag|index
XXXX

MEMORY

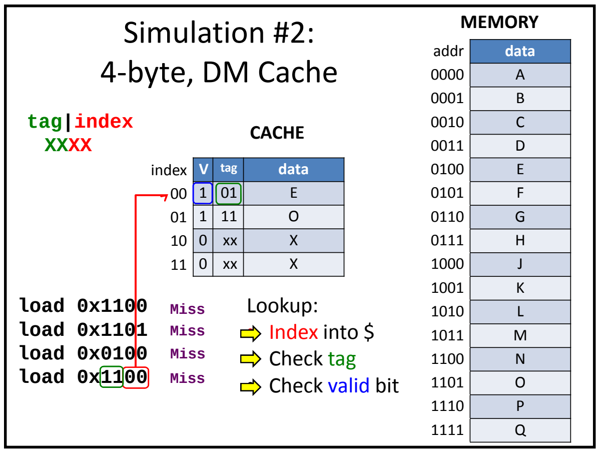
addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

CACHE

index	V	tag	data
00	1	01	E
01	1	11	O
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101   Miss
load 0x0100   Miss
load 0x1100
```

Lookup:
- Index into $
- Check tag
- Check valid bit



Simulation #2: 4-byte, DM Cache

tag|index
XXXX

MEMORY

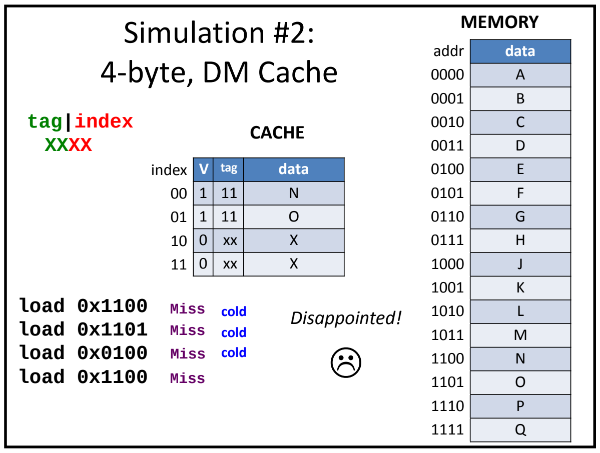
addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

CACHE

index	V	tag	data
00	1	01	E
01	1	11	O
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss
load 0x1101   Miss
load 0x0100   Miss
load 0x1100   Miss
```

Lookup:
⇨ Index into $
⇨ Check tag
⇨ Check valid bit



Simulation #2: 4-byte, DM Cache

tag|index
XXXX

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

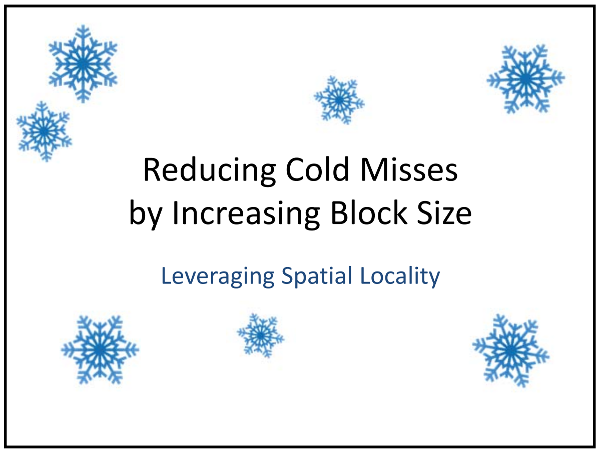
CACHE

index	V	tag	data
00	1	11	N
01	1	11	O
10	0	xx	X
11	0	xx	X

```
load 0x1100   Miss  cold
load 0x1101   Miss  cold
load 0x0100   Miss  cold
load 0x1100   Miss
```

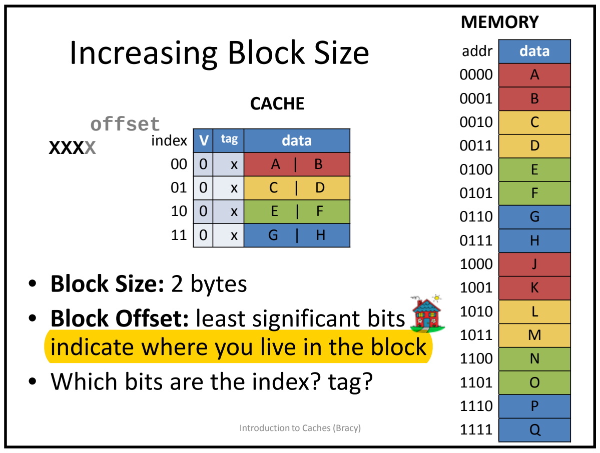
Disappointed!

☹



Reducing Cold Misses by Increasing Block Size

Leveraging Spatial Locality



Increasing Block Size

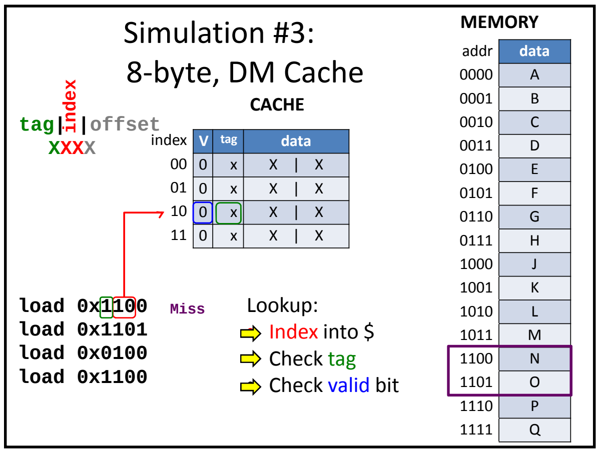
offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	A | B
01	0	x	C | D
10	0	x	E | F
11	0	x	G | H

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

- **Block Size:** 2 bytes
- **Block Offset:** least significant bits indicate where you live in the block
- Which bits are the index? tag?

Introduction to Caches (Bracy)



Simulation #3: 8-byte, DM Cache

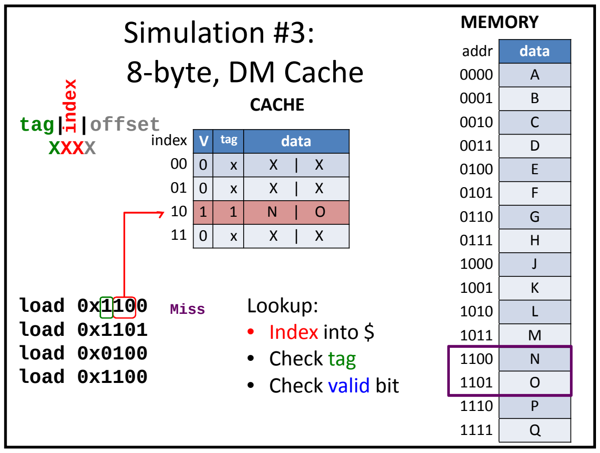
tag|index|offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	0	x	X | X
11	0	x	X | X

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

```
load 0x1100   Miss
load 0x1101
load 0x0100
load 0x1100
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit



Simulation #3: 8-byte, DM Cache

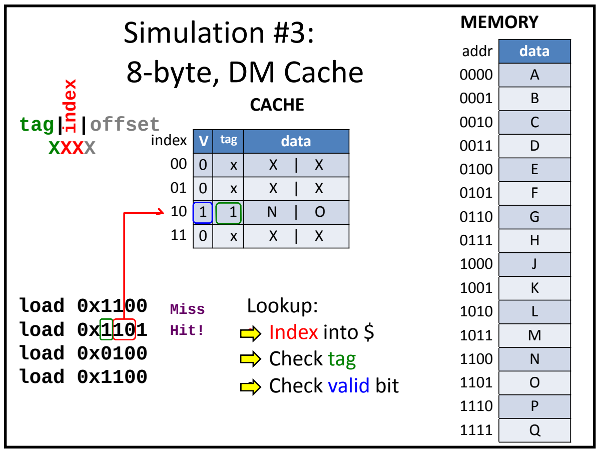
tag|index|offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	1	N | O
11	0	x	X | X

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

```
load 0x1100   Miss
load 0x1101
load 0x0100
load 0x1100
```

Lookup:
- Index into $
- Check tag
- Check valid bit



Simulation #3: 8-byte, DM Cache

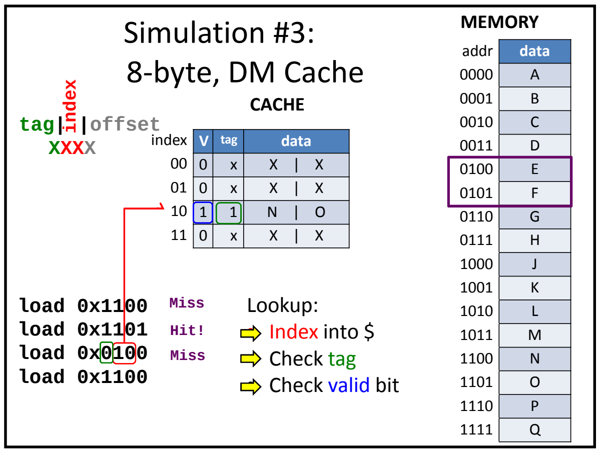
tag|index|offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	1	N | O
11	0	x	X | X

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100
load 0x1100
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit



Simulation #3: 8-byte, DM Cache

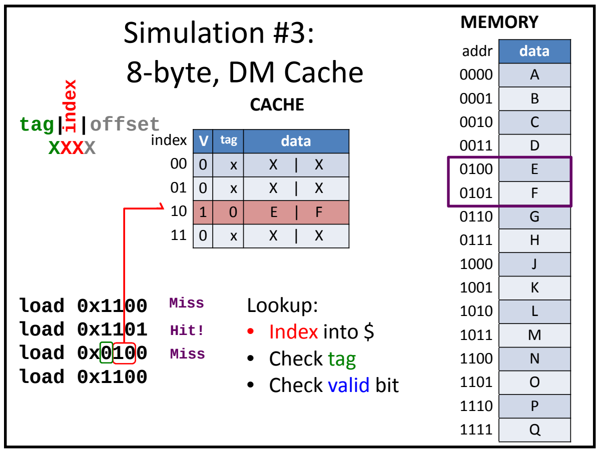
tag|index|offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	1	N | O
11	0	x	X | X

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100   Miss
load 0x1100
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit



Simulation #3: 8-byte, DM Cache

tag|index|offset
XXXX
CACHE
MEMORY

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	0	E | F
11	0	x	X | X

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100   Miss
load 0x1100
```

Lookup:
- Index into $
- Check tag
- Check valid bit

Simulation #3: 8-byte, DM Cache

tag|index|offset
XXXX

CACHE

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	0	E | F
11	0	x	X | X

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100   Miss
load 0x1100   Miss
```

Lookup:
- ⇨ Index into $
- ⇨ Check tag
- ⇨ Check valid bit

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #3: 8-byte, DM Cache

CACHE

index	V	tag	data
00	0	x	X | X
01	0	x	X | X
10	1	0	E | F
11	0	x	X | X

```
load 0x1100   Miss  cold
load 0x1101   Hit!
load 0x0100   Miss  cold
load 0x1100   Miss  conflict
```

1 hit, 3 misses

3 bytes don't fit in an 8 byte cache?

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Removing Conflict Misses with Fully-Associative Caches

8 byte, fully-associative Cache

XXXX

CACHE

V	tag	data	V	tag	data	V	tag	data	V	tag	data
0	xxx	X | X	0	xxx	X | X	0	xxx	X | X	0	xxx	X | X

What should the **offset** be?

What should the **index** be?

What should the **tag** be?

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #4: 8-byte, FA Cache

XXXX
tag|offset

CACHE

V	tag	data	V	tag	data	V	tag	data	V	tag	data
0	xxx	X | X	0	xxx	X | X	0	xxx	X | X	0	xxx	X | X

```
load 0x1100   Miss
load 0x1101
load 0x0100
load 0x1100
```

Lookup:
- ~~Index into $~~
- ⇨ Check tags
- ⇨ Check valid bits

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #4: 8-byte, FA Cache

XXXX
tag|offset

CACHE

V	tag	data	V	tag	data	V	tag	data	V	tag	data
1	110	N | O	0	xxx	X | X	0	xxx	X | X	0	xxx	X | X

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100
load 0x1100
```

Lookup:
- ~~Index into $~~
- ⇨ Check tags
- ⇨ Check valid bits

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #4: 8-byte, FA Cache

XXXX
tag|offset

CACHE

V	tag	data	V	tag	data	V	tag	data	V	tag	data
1	110	N | O	0	xxx	X | X	0	xxx	X | X	0	xxx	X | X

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100   Miss
load 0x1100
```

Lookup:
- ~~Index into $~~
- Check tags
- Check valid bits

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Simulation #4: 8-byte, FA Cache

XXXX
tag|offset

CACHE

V	tag	data	V	tag	data	V	tag	data	V	tag	data
1	110	N | O	1	010	E | F	0	xxx	X | X	0	xxx	X | X

```
load 0x1100   Miss
load 0x1101   Hit!
load 0x0100   Miss
load 0x1100   Hit!
```

Lookup:
- ~~Index into $~~
- Check tags
- Check valid bits

MEMORY

addr	data
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	J
1001	K
1010	L
1011	M
1100	N
1101	O
1110	P
1111	Q

Pros and Cons of Full Associativity

+ No more conflicts!
+ Excellent utilization!
But…
Parallel Reads
- lots of reading!

Serial Reads
- lots of waiting

$$t_{avg} = t_{hit} + \%_{miss} * t_{miss}$$

= 4 + 5% x 100
= 9 cycles

= 6 + 3% x 100
= 9 cycles