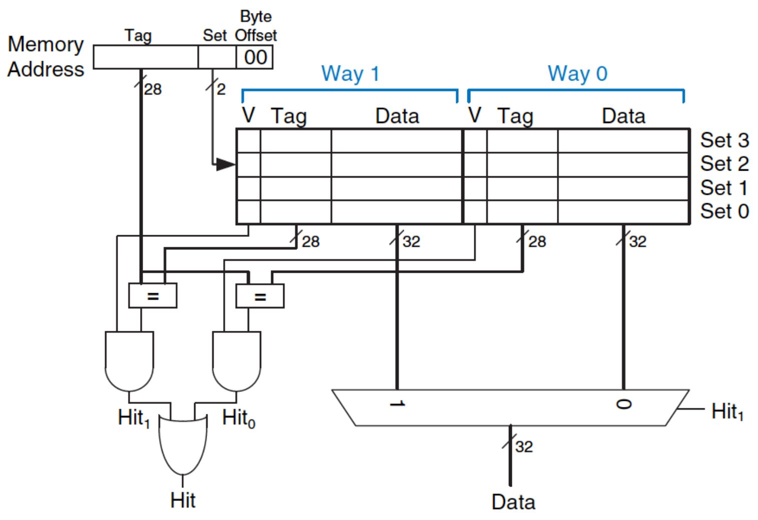
How does the cache work for the loop given a SAC versus Direct Mapped Cache?

What does the cache look like at the end of the iterations?

How many misses are there?

```
      addi $t0, $0, 5
loop: beq  $t0, $0, done
      lw   $t1, 0x4($0)
      lw   $t2, 0x24($0)
      addi $t0, $t0, -1
      j    loop
done:
```



	Way 0			Way 1	
V	Tag	Data	V	Tag	Data

```
      addi $t0, $0, 5
loop: beq  $t0, $0, done
      lw   $t1, 0x4($0)
      lw   $t2, 0x24($0)
      addi $t0, $t0, -1
      j    loop
done:
```

Fully Associative Cache

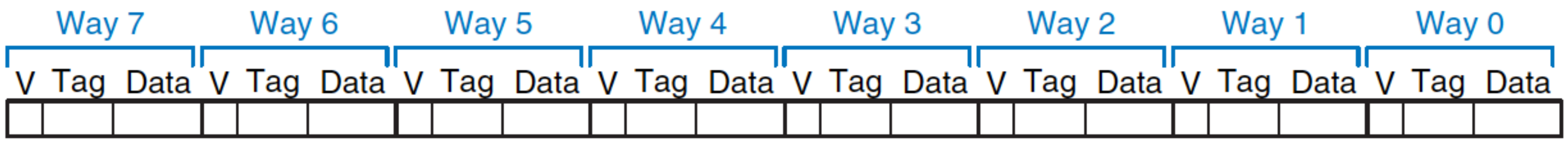
Set-associative cache with 1 set

For each request, eight tag comparisons must be made

- An 8:1 MUX chooses the proper data if a hit occurs.

Fully associative caches tend to have the fewest conflict misses for a given capacity but require more hardware for tag comparisons

- Best suited to small caches because of the large number of comparators.



What Data is Replaced?

In direct mapped cache, each address maps to a unique block and set

- If a set is full and new data is needed, block is replaced with new data
- No choice to be made

In associative caches, the cache must choose which block to evict

Question: Which way should we replace when both ways are full?

- Hint: How could we use locality to guide our choice?

		Way 0				Way 1	
V	Tag	Data	LRU	V	Tag	Data	LRU

Temporal Locality

In two-way associative cache, a use bit indicates which way within a set was least recently used.

- Flip between the two as used.
- For set associative caches with more than two ways, it's a bit more complicated.
- Generally divide the ways into two groups and track which group was recently used.
- New block replaces a random block within the least recently used group

Pseudo-LRU - good enough in practice.

The Evolution of MIPS Caches

Year	CPU	MHz	L1 Cache	L2 Cache
1985	R2000	16.7	none	none
1990	R3000	33	32 KB direct mapped	none
1991	R4000	100	8 KB direct mapped	1 MB direct mapped
1995	R10000	250	32 KB two-way	4 MB two-way
2001	R14000	600	32 KB two-way	16 MB two-way
2004	R16000A	800	64 KB two-way	16 MB two-way
2010	MIPS32 1074K	1500	32 KB	variable size

Write Policy

Write-through cache

- Data written to the cache is written simultaneously to memory.

Write-back Cache

- A dirty bit is assigned to each block
- 1 when the block has been written and 0 otherwise
- Dirty blocks are written back to main memory only when they are evicted from the cache.

Write through doesn't require the bit, but generally requires more main memory writes.

- Modern caches are usually write-back because main memory access time is so large.