COMP2121: Microprocessors and Interfacing

Caches

http://www.cse.unsw.edu.au/~cs2121

Lecturer: Hui Wu

Term 2, 2019

1

1

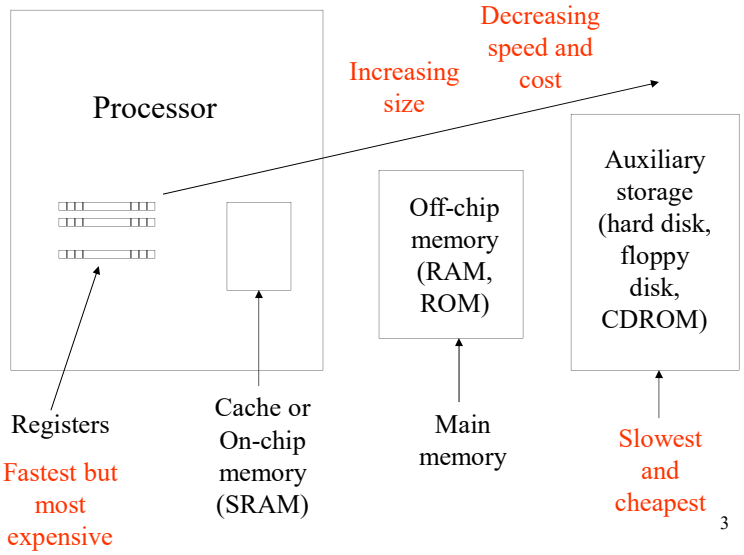
Contents

- Directed Mapped Cache
- Set Associative Cache
- Fully-Associative Cache

2

2

Revisiting Memory Hierarchy (1/2)



3

3

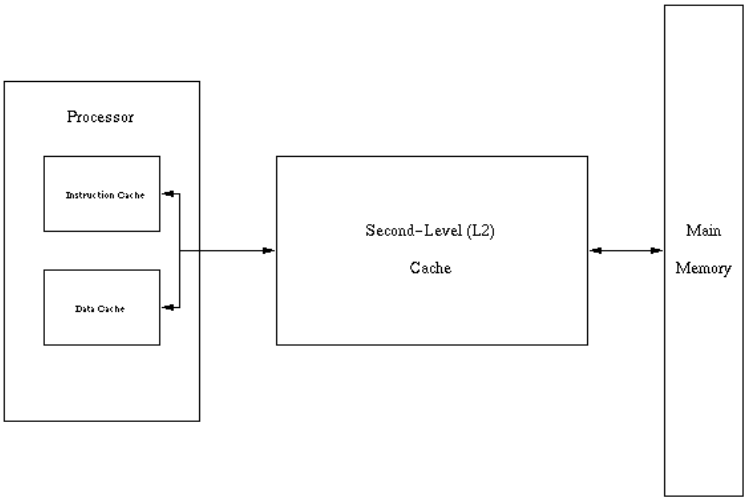
Revisiting Memory Hierarchy (2/2)

- Better performance by having a hierarchy of storages
- The closer to CPU a storage, the faster it is
- Caches are used to speed up accesses to main memory
  - A cache is a high speed buffer to store data and instructions
  - A cache is managed by hardware and typically invisible to programmers
  - Typically there are separate level-one (L1) caches for instructions and data and a shared level-two (L2) cache for both data and instructions
  - For each memory access, the processor looks up the L1 cache, then the L2 cache, and lastly the main memory

4

4

Two Levels of Caches



5

5

Why Caches Work?

- **Temporal Locality**: If a memory location is referenced, it is very likely that the memory location will be referenced again in the near future
  - Loops
- **Spatial Locality**: If a memory location is referenced, it is very likely that a nearby memory location will also be referenced in the near future
  - Sequential accesses to an array

6

6

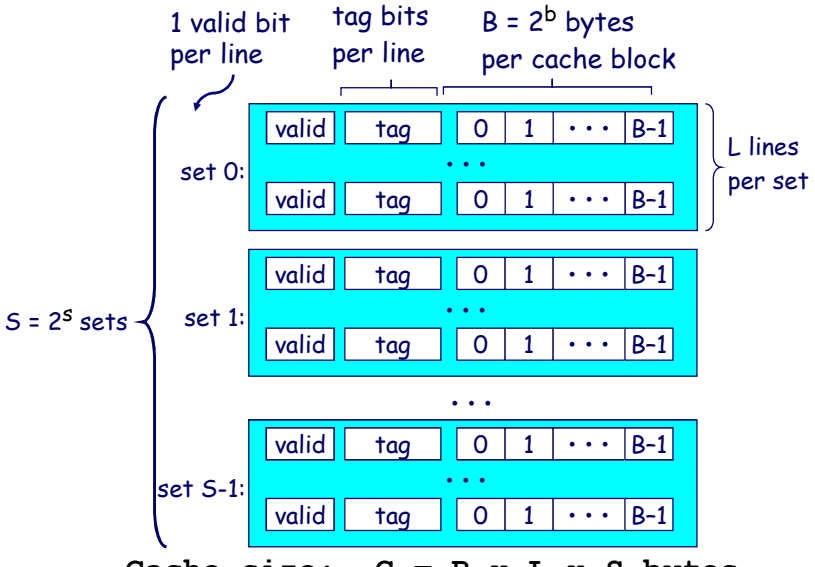
General Organisation of Caches (1/3)

- A cache is an array of sets
- Each set contains one or more cache lines
- Each cache line holds a block of data
  - Information transfer between the cache and the memory is in terms of complete cache lines, rather than individual bytes. Thus if a program needs a particular byte, the entire cache line containing that byte is obtained from the memory

7

7

General Organisation of Caches (2/3)



Cache size: C = B x L x S bytes

8

8

General Organisation of Caches (3/3)

- The main memory is partitioned into contiguous memory blocks such that each memory block exactly fits one cache line
- The tag serves as a unique identifier for a group of data. Because different memory blocks may be mapped into the same cache line, the tag is used to differentiate between them
- The valid bit indicates whether the data in a block is valid (1) or not (0).

9

Cache Organisations

- Direct mapped cache
  - Each set has only one cache line
  - Each main memory block maps to exactly one cache line
- Set associative cache
  - Each set has a fixed number of cache lines
  - Each memory block can map to any cache line of a set
- Fully associative cache
  - There is only one set
  - Each memory block can map to any cache line of a set

10

10

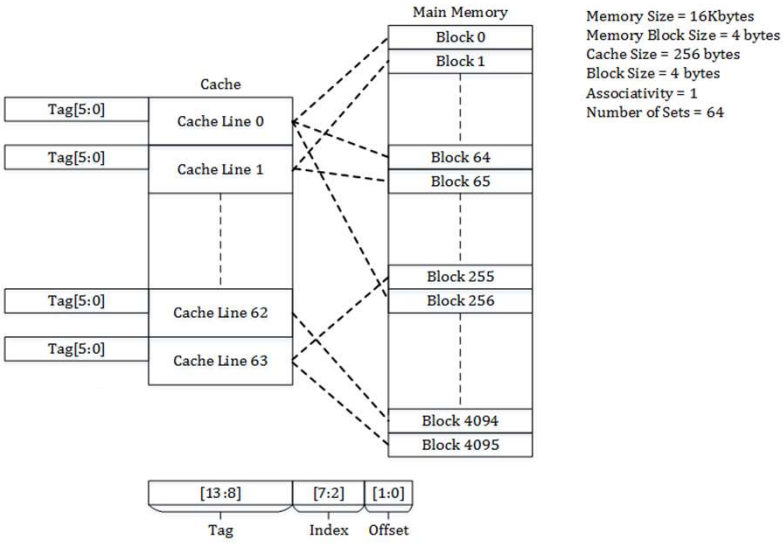
Direct Mapped Cache (1/5)

- Each memory block maps to only one cache line
- Address is in two parts: Least significant bits identify a unique word, and most significant bits specify one memory block
- The MSBs are split into an index field (cache line number) and a tag field

11

11

Direct Mapped Cache (2/5)



Taken from https://en.wikipedia.org/wiki/Cache_Placement_Policies

12

12

Direct Mapped Cache (3/5)

To place a memory block in the cache

- The set is determined by the index bits derived from the address of the memory block
- The memory block is placed in the set identified and the tag is stored in the tag field associated with the set
- If the cache line is previously occupied, then the new data replaces the memory block in the cache

13

13

Direct Mapped Cache (4/5)

To locate a word in the cache

- The set is identified by the index bits of the address
- The tag of the memory address is compared with the tag of the set. If the tag matches, there is a cache hit and the cache block is returned to the processor. Otherwise, there is a cache miss and the memory block is fetched from the main memory

14

14

Direct Mapped Cache (5/5)

Advantages

- This placement policy is power efficient as it needs only one comparator
- The placement policy and the replacement policy is simple
- It requires cheap hardware as only one tag needs to be checked at a time.

Disadvantage

- It has lower cache hit rate, as there is only one cache line available in a set
- Every time a new memory is referenced to the same set, the cache line is replaced

15

15

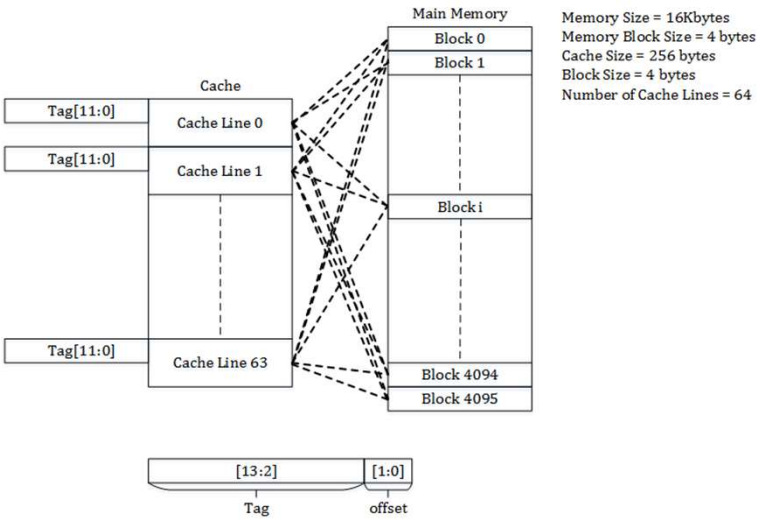
Fully Associative Cache (1/5)

- There is only one set
- Each memory block can map to any cache line
- Address is in two parts: Least significant bits identify a unique word, and most significant bits are used as tag

16

16

Fully Associative Cache (2/5)



Taken from https://en.wikipedia.org/wiki/Cache_Placement_Policies

17

17

Fully Associative Cache (3/5)

To place a memory block in the cache

- The cache line is selected based on the valid bit associated with it. If the valid bit is 0, the new memory block can be placed in the cache line, else it has to be placed in another cache line with valid bit 0
- If the cache is completely occupied, then a block is evicted and the memory block is placed in that cache line
  - The eviction of memory block from the cache is decided by the replacement policy

18

18

Fully Associative Cache (4/5)

To locate a word in the cache

- The tag of the memory address is compared with the tags of all cache lines. If it matches, the block is present in the cache and is a cache hit. Otherwise, it is a cache miss and has to be fetched from the lower memory
- Based on the *offset*, a byte is selected and returned to the processor

19

19

Fully Associative Cache (5/5)

Advantages

- Fully associative cache provides us the flexibility of placing memory block in any of the cache lines and hence full utilization of the cache
- The placement policy provides better cache hit rate
- It offers the flexibility of utilizing a wide variety of replacements algorithms if a cache miss occurs

Disadvantages

- Needs n comparators, where n is the number of cache lines, and thus infeasible

20

20

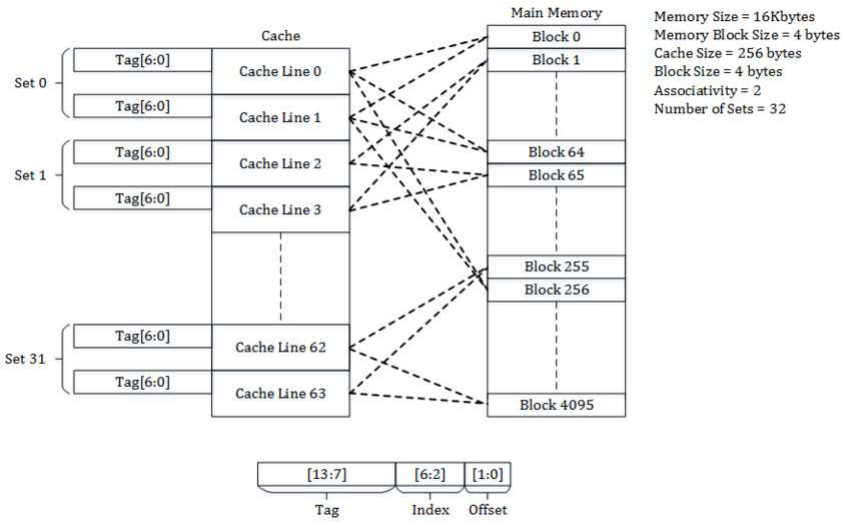
Set Associative Cache (1/5)

- A trade-off between direct mapped cache and fully associative cache
- Each set has a fixed number of cache lines
- Each memory block can map to any cache line of the set it belongs to

21

21

Set Associative Cache (2/5)



Memory Size = 16Kbytes
Memory Block Size = 4 bytes
Cache Size = 256 bytes
Block Size = 4 bytes
Associativity = 2
Number of Sets = 32

Taken from https://en.wikipedia.org/wiki/Cache_Placement_Policies

22

22

Set Associative Cache (3/5)

To place a memory block in the Cache

- The set is determined by the index bits derived from the memory address
- The memory block is placed in the set identified and the tag is stored in the tag field associated with the set
- If the cache line is occupied, then the new data replaces the cache block identified with the help of replacement policy

23

23

Set Associative Cache (4/5)

To locate a word in the Cache

- The set is determined by the index bits derived from the address of the memory block
- The tag is compared with the tags of all cache lines of the selected set. If the tag matches, then it is a cache hit and the appropriate word is fetched and delivered to the processor. If the tag does not match, it is a cache miss and is fetched from the main memory

24

24

Set Associative Cache (5/5)

Advantages

- It makes a good trade-off between hardware complexity and cache hit rate

Disadvantages

- The placement policy will not effectively use all the available cache lines in the cache

25

25

Cache Replacement Algorithm (1/2)

For direct mapped Cache

- No choice
- Each block only maps to one line
- Replace that line

26

26

Cache Replacement Algorithm (2/2)

For fully associative cache and set associative cache

- Hardware implemented algorithm (speed)
- Least Recently used (LRU): replace the block with no reference longest time
- First in first out (FIFO): replace the block that has been in the cache longest
- Least frequently used (LFU): replace block which has had fewest hits
- Random (implemented using counter)

27

27

Write Policy

- A cache line must not overwritten by a block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly and it may cause inconsistency

28

28

Write Through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep L1 cache up to date
- Lots of traffic
- Slows down writes

29

29

Write Back

- Updates initially made in cache only
- Update bit for the cache line is set when update occurs
- If a memory block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync

30

30