Cache

↳ because of locality of reference

↳ memory tend to refer into one particular area

Block size (main memory) = Line size (cache)

↳ · Block

unit of transfer between cache & main memory

main ⟷ cache (block)

· Frame

distinguish between data transferred & reference to cache

Frame | Data } Block

· Line

portian of cache capable of holding one block

memory: 1 block → cache: 1 line (block ≡ line)

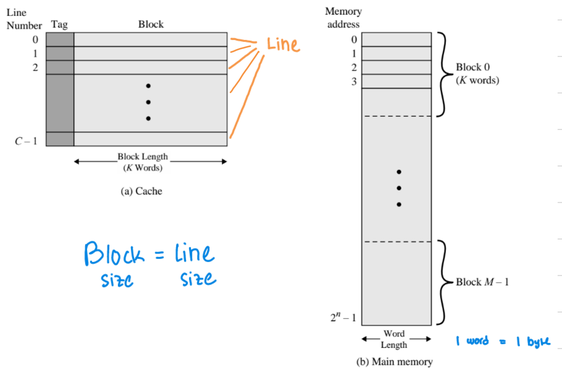
· Tag

portion of line used for addressing

line { Tag | Data

· Line size

num. of data bytes in a line



Block size = Line size

1 word = 1 byte

MM size = 64 word

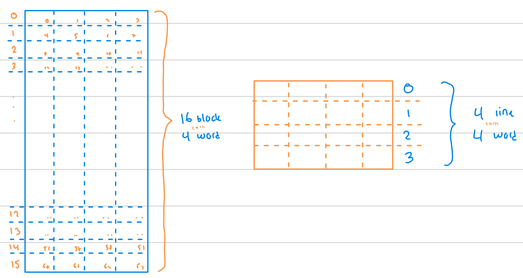
Block size = 4 word

Num of Block = 64/4 = 16 block

cache size = 16 word

Line size = Block size = 4 word

Num of line = 16/4 = 4 line



↳ Hit

when data is found in cache

Miss

when data is not found in cache

↳ Cache performance

- **Hit time:** Time to access the cache
- **Miss time:** Time to process a cache miss
  - Move data from lower level memory to the cache and CPU
- **Hit ratio:** % of time the data is found in the cache
- **Miss ratio:** (1 - hit ratio)
- **Cache block size or cache line size:** the amount of data that gets transferred on a cache miss
- **Effective access time:** (Hit Ratio * Hit Time) + (Miss Ratio * Miss Time)

↳

- Take the following access time of A computer system, (T_{ma} = **500ns)** (execution access time)
- Based on the above, the overall access time to the memory with no cache is, (main memory)
  - $T_{acc} = T_{ma}$ = **500ns** (all time)
- If the above system has a cache with an access time of **50ns** (T_{ca}) and a hit ratio of **90%**, the overall access time is, (cache access time)
- $T_{acc} = (T_{ca}$ X 0.90) + (T_{ma} X (1-0.9)) (hit time, hit ratio, miss ratio, miss ratio) } hit doesn't contain in cache, must get from memory
- T_{acc} = (50 X 0.90) + (500 x 0.10)
- T_{acc} = (45) + (50)
- T_{acc} = **95ns**

Cache acces method

↳ ① Direct Mapped

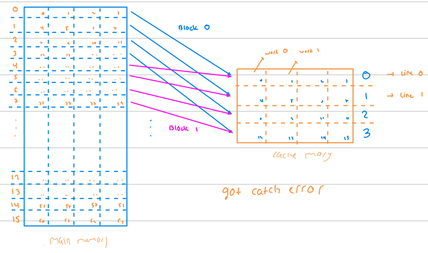
↳ each block (main memory) map to only one cache line

many-to-one

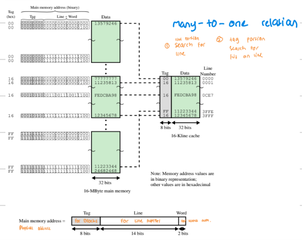
Tag (s-r)	Line or Slot	Word
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- **No two blocks in the same line have the same Tag field**
- Check contents of cache by **finding line** and **checking Tag**

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
- Number of lines in cache = m = 2^r
- Size of cache = 2^{r+w}
- Size of tag = (s – r) bits



got catch error



↳ ② Associative mapping

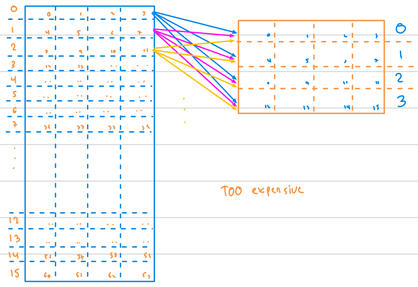
↳ each block can map to any cache line

many-to-many

S Tag	Word
22 bit	2 bit

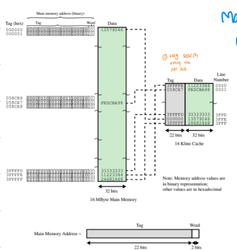
- 22 bit tag stored with each 32 bit block of data
- Compare **tag field with tag entry in cache to check for hit**
- Least significant 2 bits of address identify which 8 bit word is required from 32 bit data block
- e.g.

	Address	Tag	Data	Cache line
–	FFFFFC	3FFFFF	24682468	3FF



too expensive

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
- Number of lines in cache = undetermined
- Size of tag = s bits



Many-to-many relations

↳ ③ Set associative Mapping

↳ cache consist number of sets

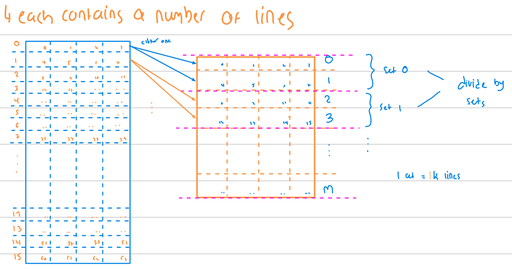
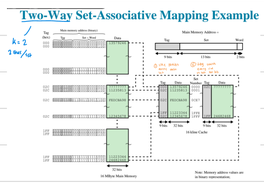
↳ each contains a number of lines

Tag 9 bit	Set 13 bit	Word 2 bit

- Use **set field** to determine cache set to look in
- Compare **tag field** to see if we have a hit
- e.g

	Address	Tag	Data	Set number
–	167FFC	02C	12345678	1FFF
–	FFFFF8	1FF	11223344	1FFE

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = 2^s
- Number of lines in set = k
- Number of sets = v = 2^d
- Number of lines in cache = kv = k * 2^d
- Size of tag = (s – d) bits



Two-Way Set-Associative Mapping Example