CSE205
Computer Organization and Architecture

Internal Memory

What you are going to study?

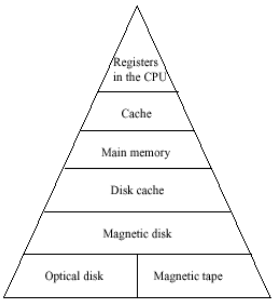
- Key Characteristics of Memory Systems
  - Location,Capacity,Unit of transfer,Access method,Performance,Physical type,Physical characteristics,Organisation
- Memory Hierarchy-Locality of Reference Principle
- Static/Dynamic RAM,Types of ROM, Organization
- Error Detection and Correction - Hamming Code
- Cache Memory
  - Typical organization
  - Operation -overview
  - Elements of Cache Design
    - Mapping - Direct, Associative, Set Associative
    - Replacement Algorithms
    - Write Policy
- Newer RAM technologies

Key Characteristics of Memory Systems

- Location
- Capacity
- Unit of transfer
- Access method
- Performance
- Physical type
- Physical characteristics
- Organisation

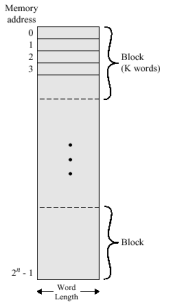
Location

- CPU - registers
- Internal- main memory and cache systems
- External – peripheral storage devices (disk/tape)



Capacity

- Word size
  - The natural unit of organisation
- Number of words
  - or Bytes



Unit of Transfer

- Internal
  - no.of bits read out of or written into memory at a time.
  - Usually determined by data bus width
  - May not be equal to word length
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - In many Systems, addressable unit is word
  - Some systems allow byte addressing
  - Relationship between length in bits A of an address and number N of addressable units is $2^A=N$

Access Methods (1).....

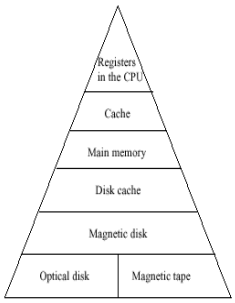
- Sequential
  - memory is organized into units of data called records
  - Start at the beginning and read through in order (specific linear sequence)
  - Access time depends on location of data and previous location. Example: tape
- Direct
  - Involves a shared read-write mechanism
  - Individual blocks have unique address
  - Access by jumping to vicinity plus sequential search,count to reach the final location.
  - Access time depends:location and previous location
  - Example: Disk

Access Methods (2)

- Random
  - Individual addresses identify locations exactly
  - Involves wired-in addressing mechanism
  - Access time: independent of location or previous access. Example: main memory and RAM
- Associative
  - Word is retrieved by a comparison with portion of the contents rather than its address
  - Access time is independent of location or previous access. Example: Cache

Memory Hierarchy

- Registers: In CPU
- Internal or Main memory:
  - May include one or more levels of cache.
- External memory: Backing store



Performance(1).....

- Access time
  - Time between presenting the address and getting the valid data and store or made available for use. (time taken to perform read/write operation). Apply in Random Access Memory
  - Time taken to position the read-write mechanism at the desired location. Apply in Non Random Access Memory
- Memory Cycle time
  - Time may be required for the memory to "recover" before next access
  - The additional time required for transients to die out on signal lines or to regenerate data if they are read destructively.
  - Cycle time is access time + recovery time
  - Only apply to Random Access Memory

PERFORMANCE (2)

- Transfer Rate
  - Rate at which data can be moved into or out of memory unit.
    - For Random Access Memory, it is equal to 1/cycle time
    - For Non-random access memory,
      - $T_N = T_A + N/R$
        where T_N = Average time to read or write N bits
        T_A = Average access time
        N = Number of bits
        R = transfer rate, in bits per second (bps)

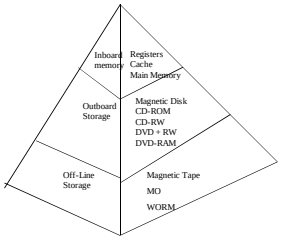
Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD

Physical Characteristics

- Volatile memory (R/W Memory)- information decays or lost when power is switched off
- Non volatile memory - no electrical power is needed (Magnetic surface memories/ROM)
- Non Erasable – memory cannot be altered (ROM)
- ORGANIZATION: For Random Access memory, key design issue
  - physical arrangement of bits to form words

Memory Hierarchy (1).....

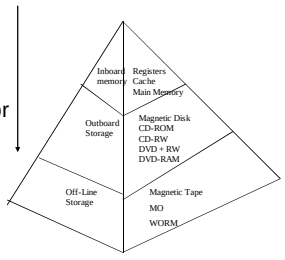


Memory Hierarchy (2)....

- **Design constraints on a computer's memory - How much? How fast? How expensive?**
- Trade off among the three key characteristics – cost, capacity & access time
  - Faster access time,greater cost per bit
  - Greater capacity,smaller cost per bit
  - Greater capacity, slower access time
- **Need for memory hierarchy**
  - designer would like to use memory technologies for larger capacity memory
  - To meet performance requirement - needs to use expensive,lower capacity memories with fast access time
  - way out of this dilemma - not to rely on a single memory component - but to employ memory hierarchy

Memory Hierarchy (3)....

- Down hierarchy
  - decreasing cost per bit
  - increasing capacity
  - increasing access time
  - decreasing frequency of access of memory by processor



Memory Hierarchy (4)- Example...

Suppose that the processor has access to two levels of memory. Level 1 contains 1000 words and has an access time of 0.1 microseconds; level 2 contains 100,000 words and has an access time of 1 microsecond.

Assume that if a word is to be accessed is in level 1, then the processor accesses it directly. If it is in level 2, then the word is first transferred to level 1 and then accessed by the processor. For simplicity, ignore the time required for the processor to determine whether the word is in level 1 or level 2.

Suppose 95% of the memory accesses are found in cache (level1). Calculate the average time to access a word?

Memory Hierarchy (5)-Example

Average time to access a word

(0.95) (0.1 microsec.) + (0.05) (0.1 microsec. + 1 microsec)

= 0.095+0.055

= 0.15 microsec.

Locality of Reference Principle

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops, subroutines
- Once a loop or subroutine is entered, there are repeated references to a small set of inst.
- Operations on tables or arrays involve access to clustered set of data words.
- Over a short period of time, processor is working with fixed clusters.
- It is possible to organize data across hierarchy - percentage of accesses to each lower level is less than that of level above
- principle can be applied more than two levels

Additional levels in hierarchy

- Disk Cache
  - a portion of main memory used as a buffer to hold data to be read out to disk
  - improves disk performance - instead of small transfers of data, few large transfers
  - data destined for write-out may be referenced by a program before transfer to the disk - data is retrieved rapidly than slowly from the disk

Semiconductor Memory

- RAM
  - Misnamed as all semiconductor memory is random access - individual words of memory are directly accessed through wired-in addressing logic
  - Read/Write
  - Writing and reading through electrical signals
  - Volatile
  - Temporary storage
  - Static or dynamic

21

Dynamic RAM

- Bits stored as charge in capacitors
- presence or absence of charge in capacitor-1/0
- Charges leak
- Need periodic refreshing even when powered
- Simpler construction
- Smaller per bit
- Less expensive
- Need refresh circuits
- Slower compared to static RAM - R/W mem.

22

Static RAM

- Bits stored as on/off switches - using flip-flops
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache

23

Read Only Memory (ROM)

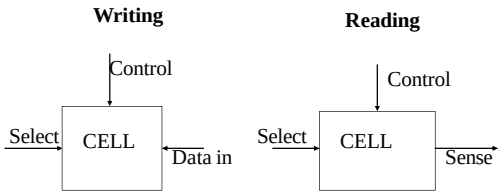
- Permanent storage
- Microprogramming
- Library subroutines
- Systems programs (BIOS)
- Function tables
- No need to load from secondary device

24

Types of ROM

- Written during manufacture
  - Very expensive for small runs
- Programmable (once)
  - PROM
  - Needs special equipment to program
- Read “mostly”
  - Erasable Programmable (EPROM)
    - Erased by UV
  - Electrically Erasable (EEPROM)
    - Takes much longer to write than read
  - Flash memory
    - Erase whole memory electrically

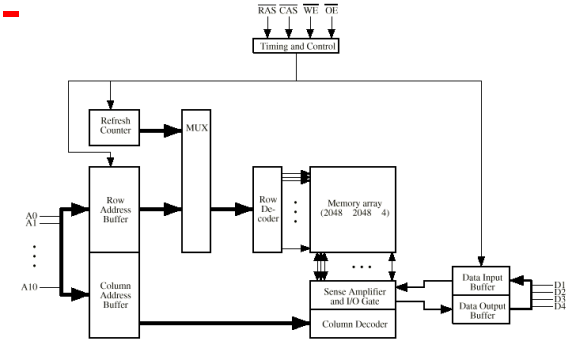
Operation of Memory Cell



Organisation in detail

- A 16Mbit chip can be organised as 1M of 16 bit words - organisation in which physical arrangement is same as logical arrangement
- One-bit-per-chip system has 16 lots of 1Mbit chip with bit 1 of each word in chip 1 and so on
- Ex: A 16Mbit chip can be organised as a 2048 x 2048 x 4bit array
  - elements of array are connected by both horizontal (row) and vertical (column) lines
    - Each horizontal line connects to select terminal
    - Each vertical line connects to Data-In/Sense terminal
  - Reduces number of address pins
    - Multiplex row address and column address
    - 11 pins to address (2^{11}=2048)
    - Adding one more pin doubles range of values so x4 capacity

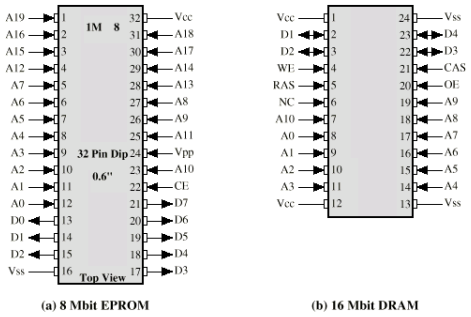
Typical 16 Mb DRAM (4M x 4)



Refreshing on DRAM

- First step, to disable chip while all data cells are refreshed. Then, the refresh counter steps through all of the row values
- For each row, the output lines from the refresh counter are supplied to the row decoder and the RAS (row address select) line is activated
- The data are read out and written back in the same location. This causes each cell in the row to be refreshed
- Effect - Takes time & slows down apparent performance

Packaging

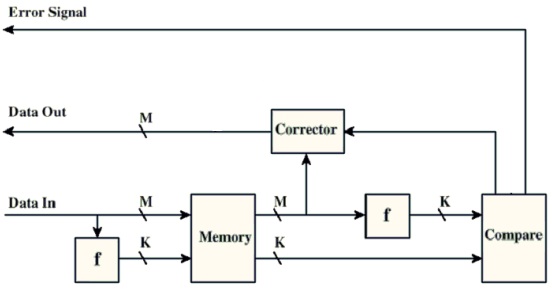


(a) 8 Mbit EPROM (b) 16 Mbit DRAM

Error Correction

- Hard Failure
  - Permanent physical defect
  - stuck at 0 or 1 or switch erratically between 0 and 1
  - caused by manufacturing defects, wear, environment abuse
- Soft Error
  - Random, non-destructive
  - No permanent damage to memory
  - caused by power supply problems, alpha particles
- Detected using Hamming error correcting code

Error Correcting Code Function



Error Correcting Code Function

When data are to be read into memory, a calculation, depicted as a function *f*, is performed on the data to produce a code. Both the code and the data are stored.

Thus, if an M bit word of data is to be stored, and the code is of length K bits, Then the actual size of the stored word is M + K bits.

When the previously stored word is read out, the code is used to detect and possibly correct errors.

A new set of K code bits is generated from the M data bits and compared with the fetched code bits.

The comparison yields one of three results:

a) No errors are detected. The fetched data bits are send out.
b) An error is detected, and it is possible to correct the error. Corrector produces a corrected set of M bits data
c) An error is detected, but cannot be corrected. Condition is reported

Hamming Error Correcting Code (SEC).....

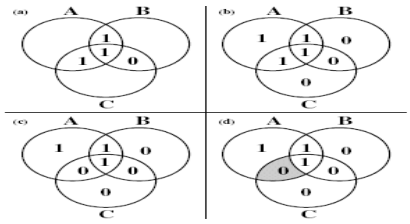


Figure 5.8 Hamming Error-Correcting Code

⌘ Venn diagram to illustrate the use of the code on 4 bit words. (M=4)

Hamming Code (Venn diagram)

⌘ 7 compartments
⌘ 4 data bits to the inner compartments
⌘ remaining compartments filled with parity bits- chosen so that the total no. of 1's in its circle is even.
⌘ If error changes one of the data bits, (fig. C), easily found.
⌘ Check the parity bits, discrepancies found in circle A and C but not B.
⌘ Only one of seven compartments in A and C but not in B. The error can be corrected by changing the bit.

Procedure for developing code that can detect and correct single bit errors in 8-bit words (SEC – single error detecting)(1)....

⌘ Step 1: Determine the length of the code. (number of bits needed to correct single bit error in a word containing M data bits)
Comparison logic receives two k-bit values as input and after XOR, produce syndrome word which is k bits wide.
⌘ 0 value indicates there is no error. Therefore 2^k-1 values to indicate which bit is in error. Because error can occur on any of M data bits or K check bits, we must have the following relationship.
⌘ $2^k-1 >= M+K$
⌘ Ex: For a word of 8 bits, 4 check bits required.

Procedure for developing code that can detect and correct single bit errors in 8-bit words (SEC) (2)....

- **Step 2: Arrangement of data and check bits into M+K bit word. (For ex. 8 bit word + 4 check bits)**
- **Bit positions are numbered from 1 to 12**
- **Bit positions whose position numbers are powers of 2 (1,2,4,8) are designated as check bits**

Bit position	Position Number	Check bit	Data bit
12	1100		M8
11	1011		M7
10	1010		M6
9	1001		M5
8	1000	C8	
7	0111		M4
6	0110		M3
5	0101		M2
4	0100	C4	
3	0011		M1
2	0010	C2	
1	0001	C1	

Procedure for developing code that can detect and correct single bit errors in 8-bit words (SEC) (3)....

- Step 3: Calculate the check bits.
- Each check bit operates on every data bit position whose position number contains 1 in the corresponding column position.
- There fore , for 8 bit word,
- C1= M1⊕ M2⊕ M4⊕ M5⊕ M7
- C2 = M1⊕ M3⊕ M4⊕ M6⊕ M7
- C4= M2⊕ M3⊕ M4⊕ M8
- C8= M5⊕ M6⊕ M7⊕ M8

Procedure for developing code that can detect and correct single bit errors in 8-bit words (SEC) (4)

- Verify the procedure by assuming 8 bit input word is 0011 1001, with data bit M1 in the right most position. (M1)
- C8=0,C4=1,C2=1,C1=1
- Suppose data bit 3 is in error-Recalculate the check bits C8 C4 C2 C1= 0 0 0 1
- Syndrome word = 0111 ⊕ 0001
- = 0110 (bit position 6 is in error which contains data bit 3 (M3))

Hamming SEC-DEC code

- SEC-DEC stands for Single –error-correcting, double- error-detecting

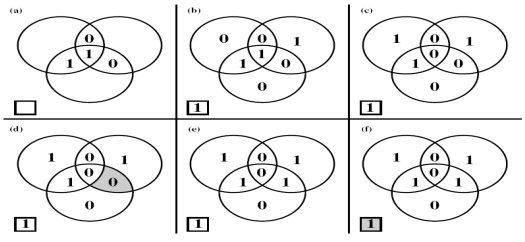
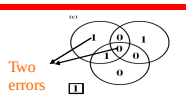


Figure 5.11 Hamming Error-Correcting Code

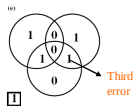
Hamming SEC-DEC code

Example: 4 bit data word

The sequence shows that if two errors occur (figure 5.11c),

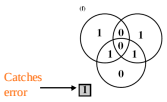


The checking procedure goes astray (d) and worsens the problem by creating a third error (e)



To overcome the problem, an eighth bit is added that is set so that the total number of 1s in the diagram is even.

The extra parity bit catches the error (f)



PROBLEM (1)

- Suppose an 8-bit data word stored in memory is 11000010. Using the Hamming algorithm, determine what check bits would be stored in memory with the data word. Show how you got your answer.

PROBLEM (2)

- For the 8-bit word 00111001, the check bits stored with it would be 0111. Suppose when the word is read from memory, the check bits are calculated to be 1101. What is the data word that was read from memory?

PROBLEM (3)

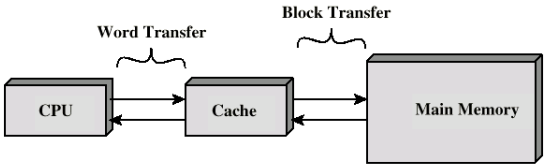
- How many check bits are needed if the Hamming error correction code is used to detect single bit errors in a 1024-bit data word?

PROBLEM (4)

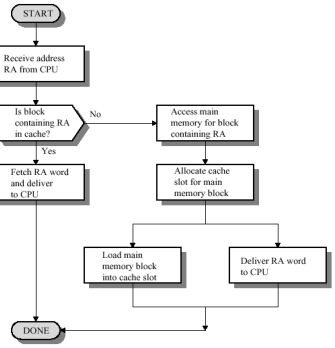
- Develop an SEC code for a 16-bit data word. Generate the code for the data word 0101 0000 0011 1001. Show that the code will correctly identify an error in data bit 4.

Cache

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module

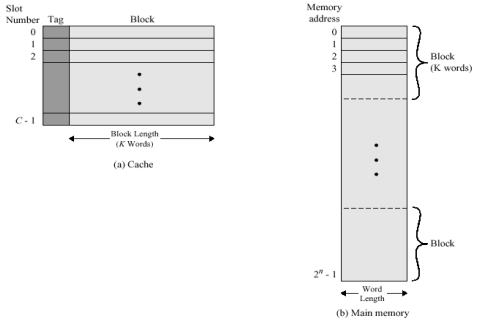


Cache



• RA – address of a word to be read that generated by processor

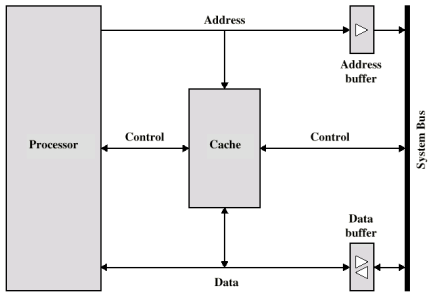
Cache operation - overview



Cache operation - overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot
- C<<M (Cache lines << Main memory blocks)

Typical Cache Organization



Elements of Cache Design

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Size does matter

- The size of cache should be designed small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone
- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Larger cache-larger gates involved-slow down
  - Checking cache for data takes time
  - Studies show that size bet. 1K and 512 K words-effective

Mapping Function

- There are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. (Direct/Associative/Set Associative)
- Means needed to determine which main memory block currently occupies a cache line.
- Example:
- Assume Cache of 64 kByte
  - block of 4 bytes - Data transfer between main memory and cache. Cache is organized as 16k (2^{14}) lines of 4 bytes each
  - Assume 16 MBytes main memory
    - each byte directly addressable by 24 bit address (2^{24}=16M)
    - thus, for mapping purposes, we can consider main memory to consist of 4M blocks of 4 bytes each.

Direct Mapping

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
  - The mapping is expressed as :

    i = j module m

    where i = cache line number

    j = main memory block number

    m = number of lines in the cache

Direct Mapping

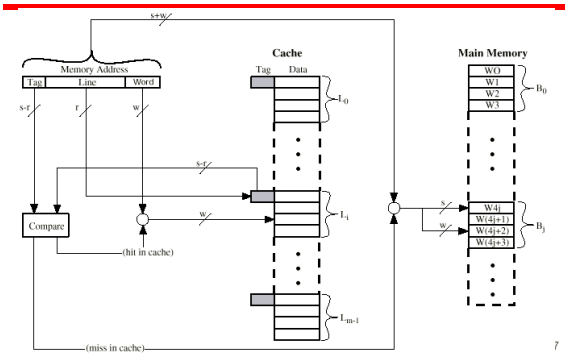
- Each main memory address consisting is three fields:
- Least Significant *w* bits identify unique word or byte within a block of main memory
- The remaining *s* bits specify one of 2^s memory block. The s bits are split into a tag of *s-r* (most significant) and a cache line field *r* bits.
- Line field identifies one of the $m = 2^r$ lines of the cache

Direct Mapping Address Structure

Tag s-r	Line or Slot r	Word w
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

Direct Mapping Cache Organization



Direct Mapping Cache

- 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 = $\frac{2^{s+w}}{2^w}$ = 2^s
- Number of lines in cache = $m = 2^r$
- Size of tag = ($s - r$) bits

Direct Mapping Cache Line Table

- The effect of the mapping is that blocks of main memory are assigned to lines of the cache as follows:

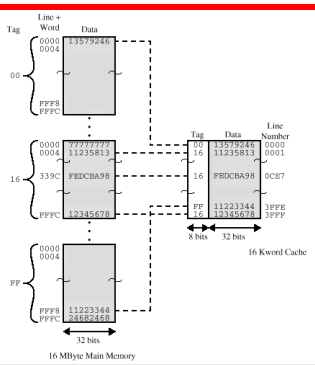
Cache Line	Main Memory blocks assigned
0	0, m, $2m$, ….., $2^s - m$
1	1, $m + 1$, $2m + 1$, …., $2^s - m + 1$
. .	. .
$m - 1$	$m - 1$, $2m - 1$, $3m - 1$, …, $2^s - 1$

Direct Mapping Example

i = j module m, where m = 16K = 2^{14}

Cache Line	Starting memory address assigned
0	000000, 010000, …., FF0000
1	000004, 010004, … , FF0004
. .	. .
$2^{14} - 1$	00FFFC, 01FFFC, …., FFFFFC

Direct Mapping Example



Direct Mapping pros & cons

- Pros:
  - Simple
  - Inexpensive
- Con:
  - Fixed location for given block
    - If a program accesses 2 blocks that map to the same line repeatedly, the blocks will be continually swapped in the cache and the hit ration will be low (cache misses are very high) – a phenomenon known as thrashing.

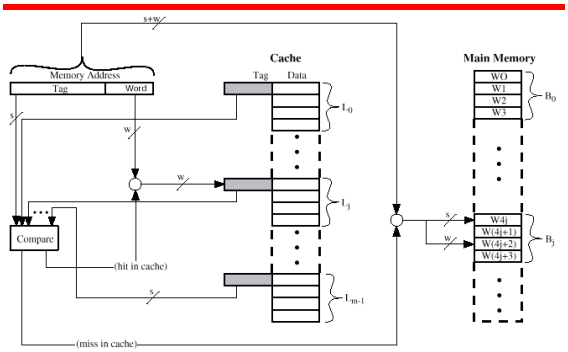
Associative Mapping

- Each main memory block can load into any line of cache (overcome the direct mapping problem)
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match

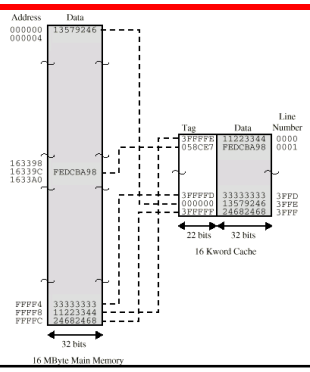
Associative Mapping

- 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 = $\frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in cache = undetermined
- Size of tag = s bits

Fully Associative Cache Organization



Associative Mapping Example



Associative Mapping Address Structure

Tag 22 bit	Word 2 bit

- A main memory address consists of a 22 bit tag and a 2 bit byte number.
- 22 bit tag stored with each 32 bit block of data for each line in the cache
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of memory address (word) identify which 8 bit data required from 32 bit data block
- The leftmost (most significant) 22 bits of the address form the tag

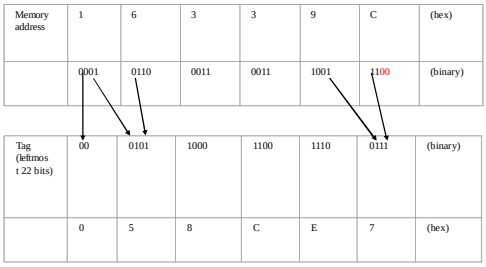
Associative Mapping Address Structure

Example:

Memory address (24 bit)	Tag (22 bit)	Data	Cache line
16339C	058CE7	FEDCBA98	0001
FFFFFC	3FFFFF	24682468	3FFF

Associative Mapping Address Structure

Example: Memory to tag working flow



Associative Mapping pros & cons

⌘Pros:

➢Flexibility as to which block to replace when a new block is read into the cache

➢Replacement algorithms are designed to maximize the hit ratio

⌘ Cons:

⌘Required complex circuitry to examine the tags of all cache lines in parallel

Set Associative Mapping

⌘Cache is divided into a number of sets, *v*

⌘Each set contains a number of lines, *k*

The relationship:

$m = v \times k$

$i = j$ modulo v

Where i = cache set number

j = main memory block number

m = number of lines in the cache

Set Associative Mapping

⌘A given block maps to any line in a given set

- e.g. Block B can be in any line of set i

⌘e.g. 2 lines per set

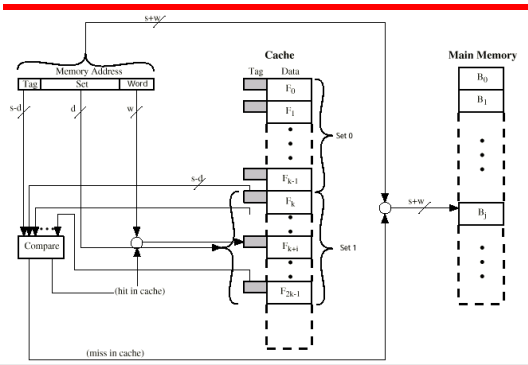
- 2 way associative mapping
- A given block can be in one of 2 lines in only one set

Tag 9 bit	Set 13 bit	Word 2 bit

Set Associative Mapping

- 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 = $\frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in set = k
- Number of sets $v = 2^d$
- Number of lines in cache = $kv = k \times 2^d$
- Size of tag = $(s - d)$ bits

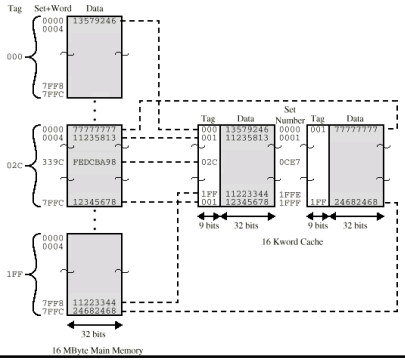
Two Way Set Associative Cache Organization



Set Associative Mapping

- Example fig 4.12 : two –way set associative
- 13 bit set number
- Block number in main memory is modulo 2^{13}
- 000000, 00A000, 00B000, 00C000 ... map to same set

Two Way Set Associative Mapping Example



Set Associative Mapping Address Structure

- 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
1FF 7FFC	1FF	12345678	1FFF
001 7FFC	001	11223344	1FFF

Replacement Algorithms (1)

For Direct Mapping

- each block only maps to one line
- when a new block is bought into the cache , replace that line, no choice is possible

Replacement Algorithms (2)

For Associative & Set Associative

- Algorithm implemented in hardware (to achieve high speed)

A) The most effective method : **Least Recently Used (LRU)**

- replace that block in the set that has been in the cache longest with no reference (hits) to it.

B) **First in first out (FIFO)**

- replace block that has been in cache longest

c) **Least frequently used**

- replace block which has had fewest hits

D) **Random**

- Pick a line at random from among the candidate lines

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Two problems to content with :
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache, to ensure main memory is valid
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Disadvantages: lots of traffic & slows down writes

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set (minimize the memory writes)
- Problems: I/O must access main memory through cache
- N.B. 15% of memory references are writes

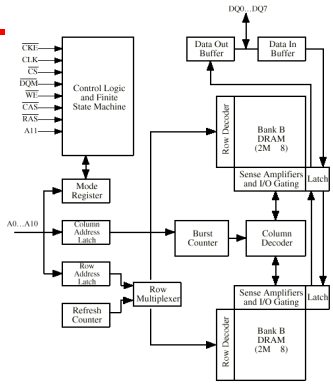
Newer RAM Technology (1)....

- Basic DRAM same since first RAM chips
- Enhanced DRAM
  - Contains small SRAM as well
  - SRAM holds last line read (c.f. Cache!)
- Cache DRAM
  - Larger SRAM component
  - Use as cache or serial buffer

Newer RAM Technology (2)...

- Synchronous DRAM (SDRAM)
  - currently on DIMMs
  - Access is synchronized with an external clock
  - Address is presented to RAM
  - RAM finds data (CPU waits in conventional DRAM)
  - Since SDRAM moves data in time with system clock, CPU knows when data will be ready
  - CPU does not have to wait, it can do something else
  - Burst mode allows SDRAM to set up stream of data and fire it out in block

SDRAM



Newer RAM Technology (3)

- Foreground reading
- DDR, DDR2, DDR4 etc.. (T –RAM, Z-RAM,TTRAM)
- Check out any other RAM you can find
- See Web site:
  - The RAM Guide