Memory Hierarchy

Outline

- Memory Hierarchy
- Hit/Miss, IPC
- Cache: Set, Line size, Associativity

No Class on Friday (07-SEP-2012)

Typical specs of a computer today

http://www.flipkart.com

Dell XPS 14
14.0" WLED (1366 x 768), Intel Core i7-740QM
4 GB DDR3 1333 MHz, 500 GB, Windows 7 Home Premium, 8x CD/DVD burner (dual layer DVD+/-R drive), NVIDIA GeForce GT 425M
2 USB 2.0, HDMI, eSATA, 6-cell lithium-ion
Built-in 2.0-megapixel HD Price: 48,300

HP TouchSmart TM2 Series TM2-2102TU
(Modern Argento), * Intel Core i3, * 3 GB DDR3 RAM,
*** 12.1 Inch Screen, * Windows 7 Home Premium**
Price : Rs. 47,199.00

Memory technologies

- Semiconductor
  - Registers
  - SRAM
  - DRAM
  - FLASH
- Magnetic
  - FDD
  - HDD
- Optical
  - CD
  - DVD

Random Access

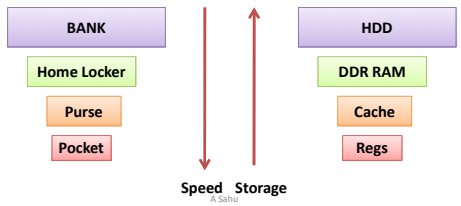
Array : x=A[i]

Random (Seq. Seek to Sector) + sequential

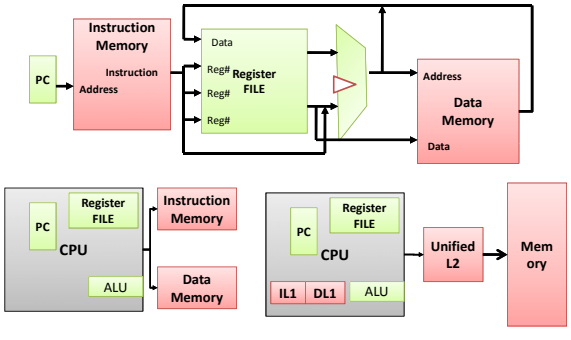
Array + Linked List

Memory Hierarchy

- Smaller is Faster - Bigger is Slower
- Places of Cash/Money



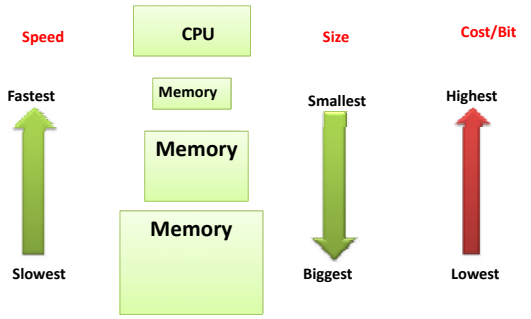
CPU Design with Memory Hierachy



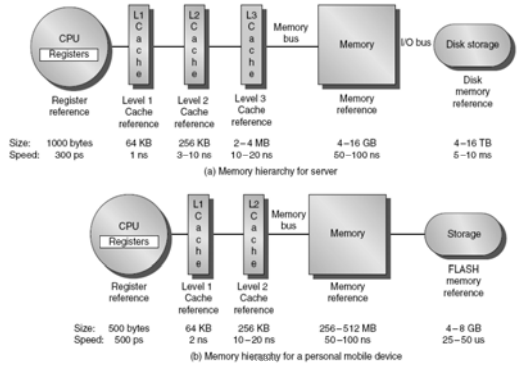
Memory Hierarchy

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor

Hierarchical structure



Memory Hierarchy



Memory Hierarchy Design

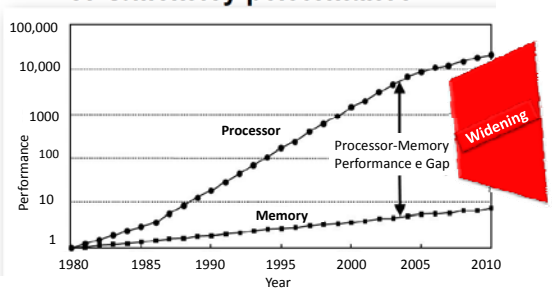
- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references
      - = 409.6 GB/s!
    - DRAM bandwidth is only 6% of this (25 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

Performance and Power

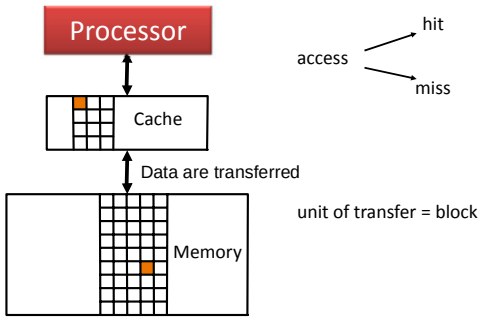
- High-end microprocessors have >10 MB on-chip cache
  - Consumes large amount of area and power budget

DRAM Speed Vs Processor Speed

CPU/Memory performance



Data transfer between levels



A Sahu 13

Principle of locality

- Temporal Locality
  - references repeated in time
- Spatial Locality
  - references repeated in space
  - Special case: Sequential Locality

```
for(i=0;i<100;i++){
    A[i] += sqrt(i);
}  // 1D SPLocality
Access A[i],  near future
will Access A[i+1], A[i+2]..
```

```
for(T=0;T<80;T++){
  for(i=0;i<10;i++)
     A[i] +=M[T]*i;
}
A[i] repeated after some Time
```

A Sahu 14

Cache Access

- Address is divided in to three part : TAG, Index, Offset
  - Offset = Address % Line Size,
  - Index = (Address/LineSize)%NumSet
  - TAG = Address/(LineSize*NumSet)
- If TAG matches with ExistingTAG then HIT else miss

```
if (TAG==CACHE[Index].TAG)
        Cache HIT
else    Cache MISS
```

- Assume LS=10, NumSet=100, Address 2067432
  - Offset = 2, Index =43, TAG=2067

A Sahu 15

Cache Example : Algorithmic

212012 212011

0	0
2120	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Direct mapped, Line size 10

A Sahu 16

Cache Example : Algorithmic

212335 212012 212011

0	0
2120	1
2	2
2123	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Direct mapped, Line size 10

A Sahu 17

Cache Example : Algorithmic

414368 212335 212012 212011

0	0
2120	1
2	2
2123	3
4	4
5	5
4143	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Direct mapped, Line size 10

A Sahu 18

Cache Example : Algorithmic

414318	414365	212335	212012	212011

0	0
2120	1
2	2
2123	3
4	4
5	5
4143	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Direct mapped, Line size 10

Cache Example : Algorithmic

414318	414365	212335	212012	212011

0	0
4143	1
2	2
2123	3
4	4
5	5
4143	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Direct mapped, Line size 10

Cache Example : Algorithmic

212012	212011

0	0
2120	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Two way Asso, Line size 10

Cache Example : Algorithmic

212335	212012	212011

0	0
2120	1
2	2
2123	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Two way Asso, Line size 10

Cache Example : Algorithmic

414368	212335	212012	212011

0	0
2120	1
2	2
2123	3
4	4
5	5
4143	6
7	7
8	8
9	9

Tag index Line

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Two way Asso, Line size 10

Cache Example : Algorithmic

414318	414365	212335	212012	212011

0	0
2120	1
2	2
2123	3
4	4
5	5
4143	6
7	7
8	8
9	9

Tag index Line

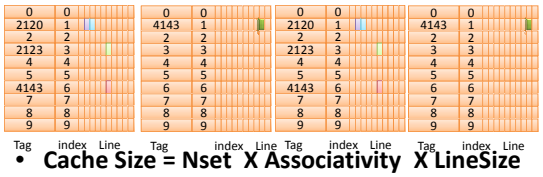
0	0
4143	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Tag index Line

Decimal Example, Two way Asso, Line size 10

Cache Size

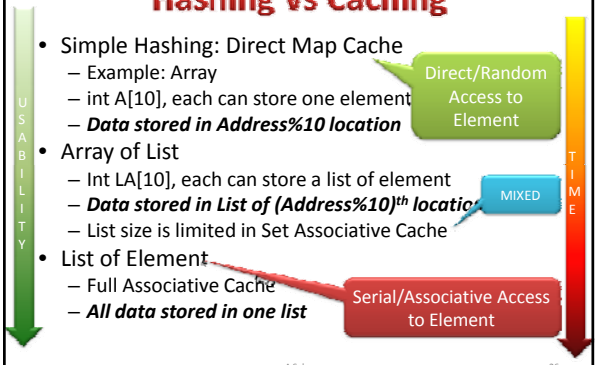
- No of Set (Depend on index field)
- Associatively (How many Tag)
- Line size (No of Addressable units/byte in a line)



- **Cache Size = Nset X Associativity X LineSize**

= 10 x 4 x 10 = 400 Byte

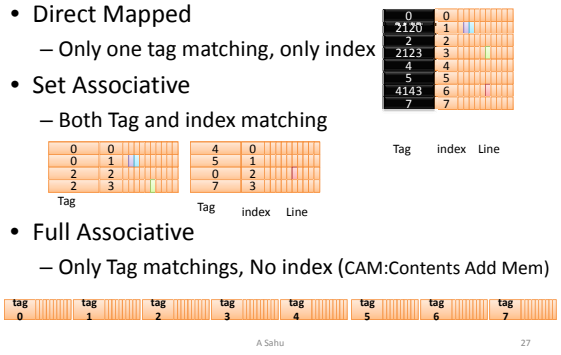
Hashing Vs Caching



- Simple Hashing: Direct Map Cache
  - Example: Array
  - int A[10], each can store one element
  - ***Data stored in Address%10 location***
- Array of List
  - Int LA[10], each can store a list of element
  - ***Data stored in List of (Address%10)th location***
  - List size is limited in Set Associative Cache
- List of Element
  - Full Associative Cache
  - ***All data stored in one list***

Cache: Placement

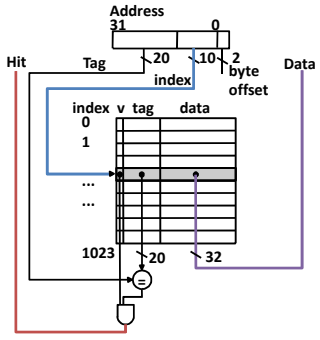
- Direct Mapped
  - Only one tag matching, only index
- Set Associative
  - Both Tag and index matching
- Full Associative
  - Only Tag matchings, No index (CAM:Contents Add Mem)



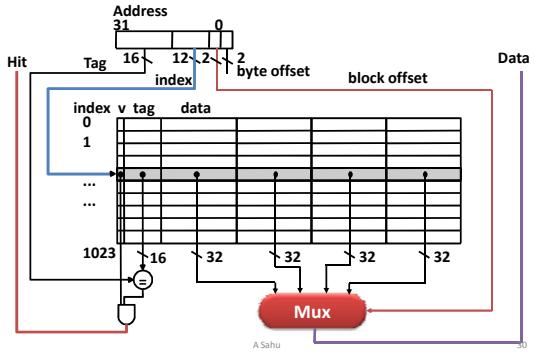
Addressing Cache

Early select: access data after tag matching
Late select: access data while tag matching

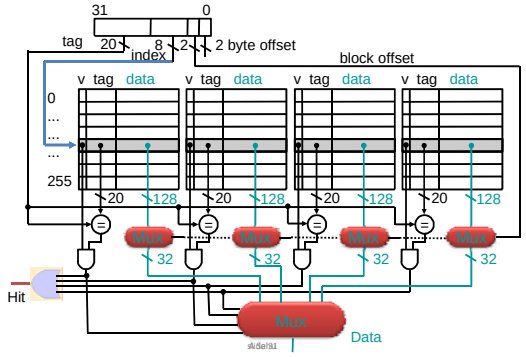
Cache access mechanism



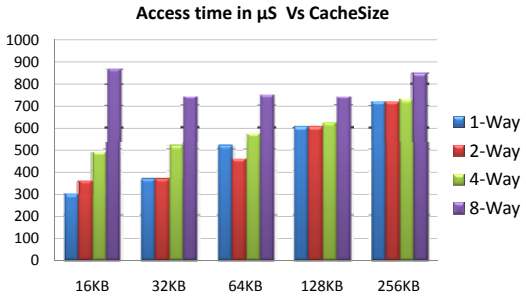
Cache with 4 word blocks



4-way set associative cache



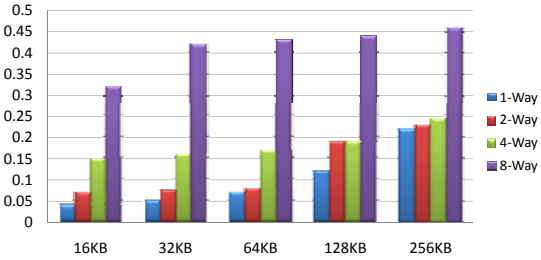
Access time: Size and Associativity



Access time vs. size and associativity

A Sahu 32

Energy: Size and Associativity



Energy per read vs. size and associativity

A Sahu 33