Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Unit 4: Cache Optimization

Dr. D. C. Kiran

Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus

~~Unit 1: Basic Processor Architecture and Design~~

~~Unit 2: Pipelined Processor and Design~~

~~Unit 3: Memory~~

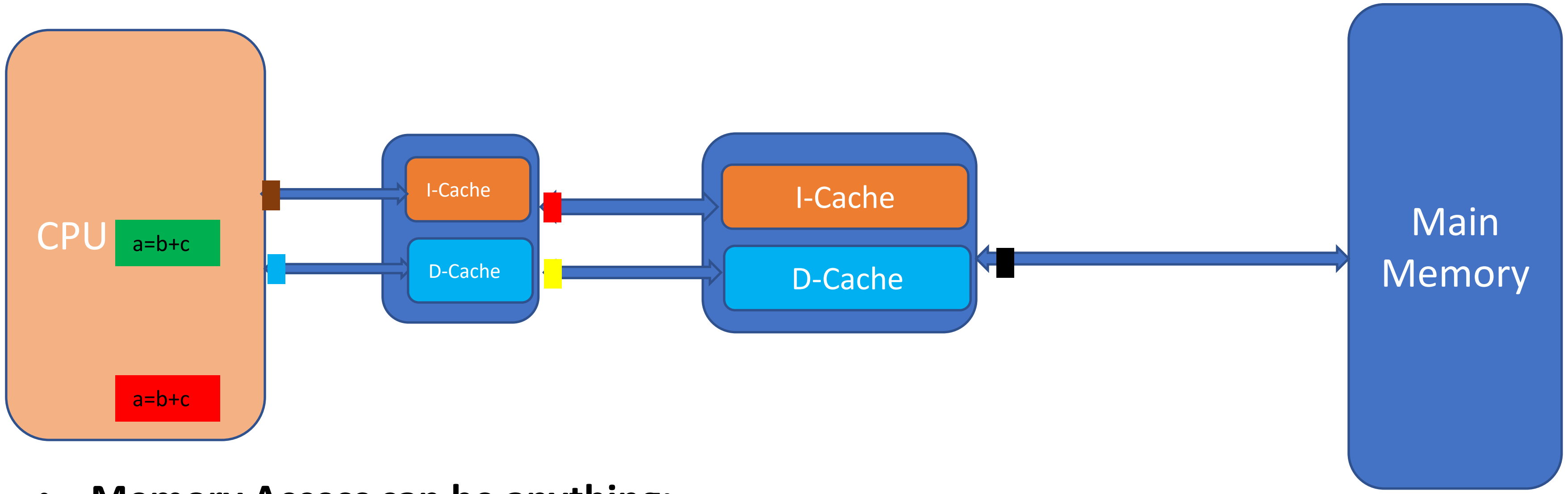
Unit 4: Input/Output Device Design

Cache Optimization

Unit 5: Advanced Architecture

Microprocessor & Computer Architecture (µpCA)

Why Average Memory Access Time?



- **Memory Access can be anything:**
  Instruction Fetch form I-Cache in L1
  Instruction Fetch from I-Cache in L2, if Miss in I-Cache of L1
  Data Access from D-Cache in L1.
  Data Access from D-Cache in L2, if Miss in D-Cache of L1
- **Hit time for L1 Cache is different Hit time for L2 Cache**
- **Miss Penalty for L1 Cache is different from Miss Penalty for L2 Cache**
- **Access depend on other factors such as Width of the bus, External & Internal Cache**

Cache Optimization

Aim: To Improve the Performance of the cache

- What is Performance of the Cache?

  Reduced Access time, to keep CPU Busy.

- How to Improve the Performance of the Cache?

?

Average Memory Access Time

AMAT = Hit Time + (Miss Rate x Miss Penalty)

Cache Performance can be improved by

- Reducing the miss rate
- Reducing the miss penalty
- Reducing the time to hit in the cache

Cache Optimizations

AMAT = Hit Time + (Miss Rate x Miss Penalty)

Cache Performance can be improved by

Six basic cache optimizations:

- Larger block size.
- Larger total **cache** capacity to reduce miss rate.
- Higher associativity.
- Higher number of **cache** levels.
- Giving priority to read misses over write.
- Avoiding address translation in **cache** indexing

AMAT = Hit Time + (Miss Rate x Miss Penalty)

Cache Performance can be improved by

- **Reducing the miss rate**
  - Larger block size, Larger cache size, and higher associativity.
- **Reducing the miss penalty**
  - Multilevel caches and giving reads priority over the writes.
- **Reducing the time to hit in the cache**
  - Avoiding address translation when indexing the cache.

Reducing Miss Rate!!!

What is a MISS ☹?

Three Categories Misses

- **Compulsory :**
  - The very first access to block cannot be in the cache. So, the block must be brought into the cache.
  - These are called cold-start misses or first reference misses.
- **Capacity:**
  - If the cache cannot contain all the blocks needed during execution of a program, capacity misses [in addition to compulsory misses] will occur because of blocks being discarded and later retrieved.

Situation: item has been in the cache, but space was tight, and it was forced out

- **Conflict:**
  - If the block placement strategy is set associative or direct mapped, conflict misses [in addition to compulsory and capacity misses] will occur because a block may be discarded and later retrieved if too many blocks map to its set. These misses are also called collision misses.
  - The idea is that hits in a fully associative cache that become misses in an n-way set associative cache, due to more than n requests on some popular sets.

Situation: item was in the cache, but the cache was not associative enough, so it was forced out

Cache Optimization: Reduce Miss Rates.

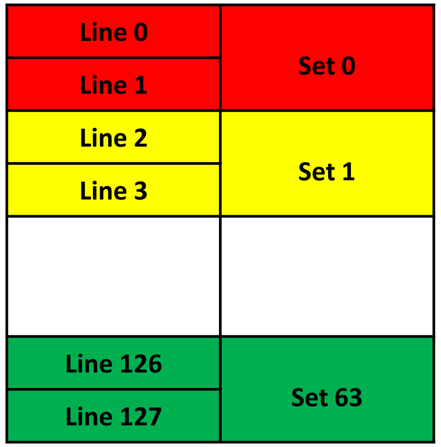
How to Categories the Miss

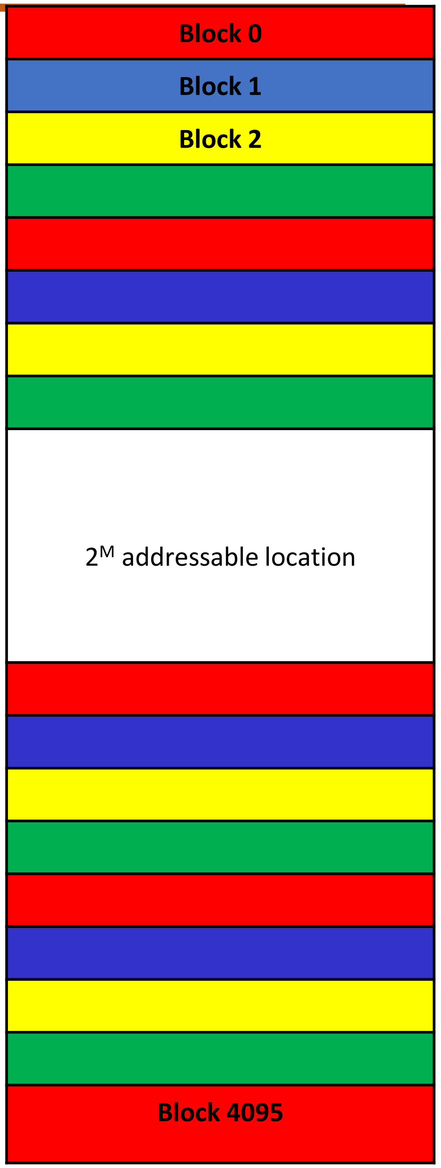
- **Compulsory Misses :**
  - Those that occur in an infinite cache.
- **Capacity Misses:**
  - Those that occur in a fully associative cache.
- **Conflict Misses:**
  - Those that occur going from fully associative to eight-way associative, four–way associative , and so on…

Microprocessor & Computer Architecture (μpCA)

Example: How to Categories the Miss

Consider a 2-way set associative cache with 128 Lines and 64 Sets



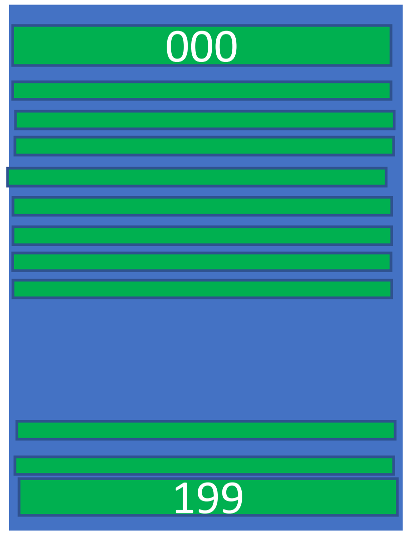


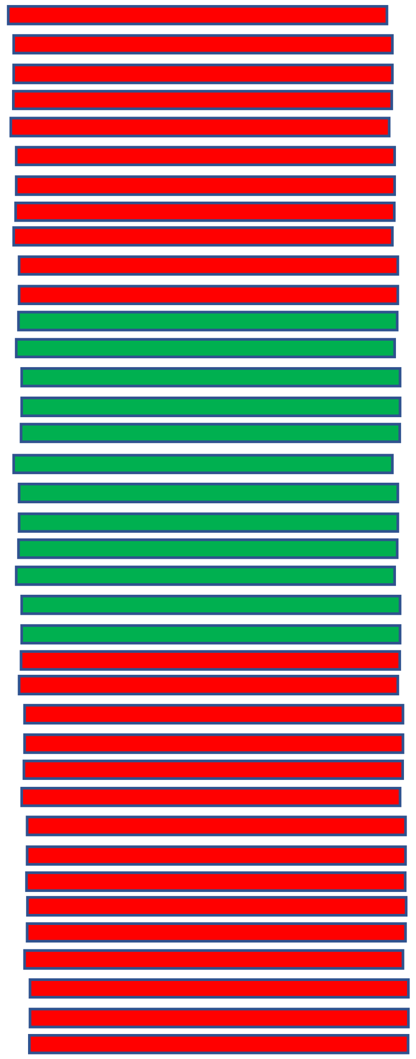
If you come across 1000 Misses, when a program is executed.

How many Compulsory Misses?
How many Capacity Misses?
How many Conflict Misses?

Step1: Identifying Compulsory Miss

Execute the program on an Infinite Size Cache and Fully Associative





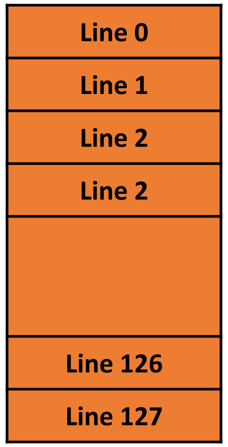
- Suppose there are 200 Misses
- When data is accessed for 1st time, you will have 200 Misses.
- Other Access are Hit.
- So 200 out of 1000 Misses are ***Compulsory Miss***

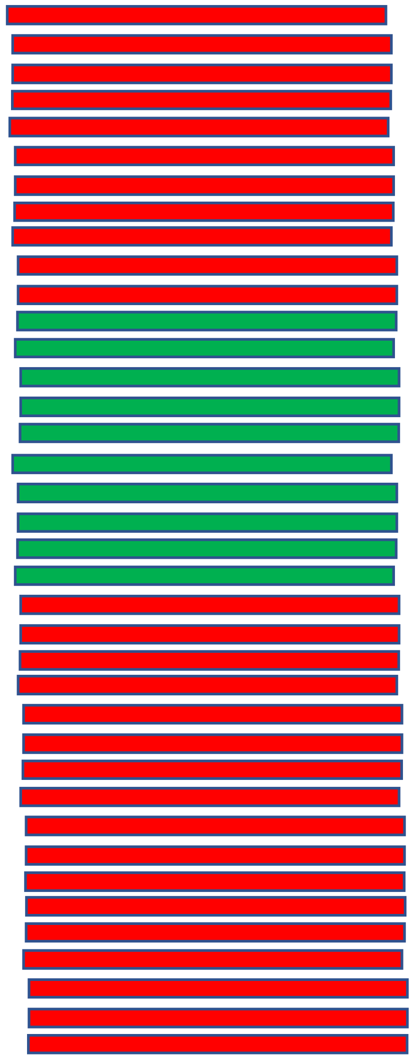
Microprocessor & Computer Architecture (µpCA)

Step2: Identifying Capacity Miss

Execute the program on a 128 Line Cache and Fully Associative.

128 Line Cache and Fully Associative





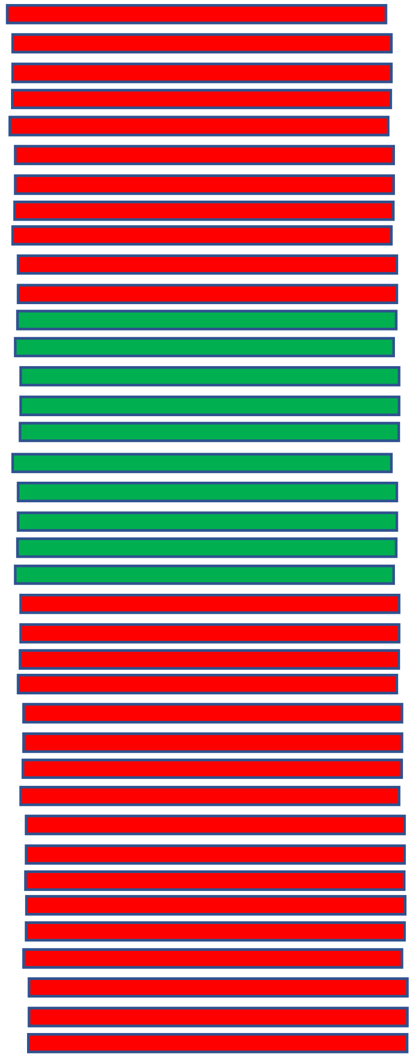
If the number of Misses are 400, 200 out of 1000 are Capacity Misses, as 200 are Compulsory Misses.

Microprocessor & Computer Architecture (μpCA)

Step3: Conflict Miss

Execute the same program on 128 Line, 2-way set Associative cache which will give 1000 Misses.

Line 0	Set 0
Line 1	
Line 2	Set 1
Line 3	
Line 126	Set 63
Line 127	

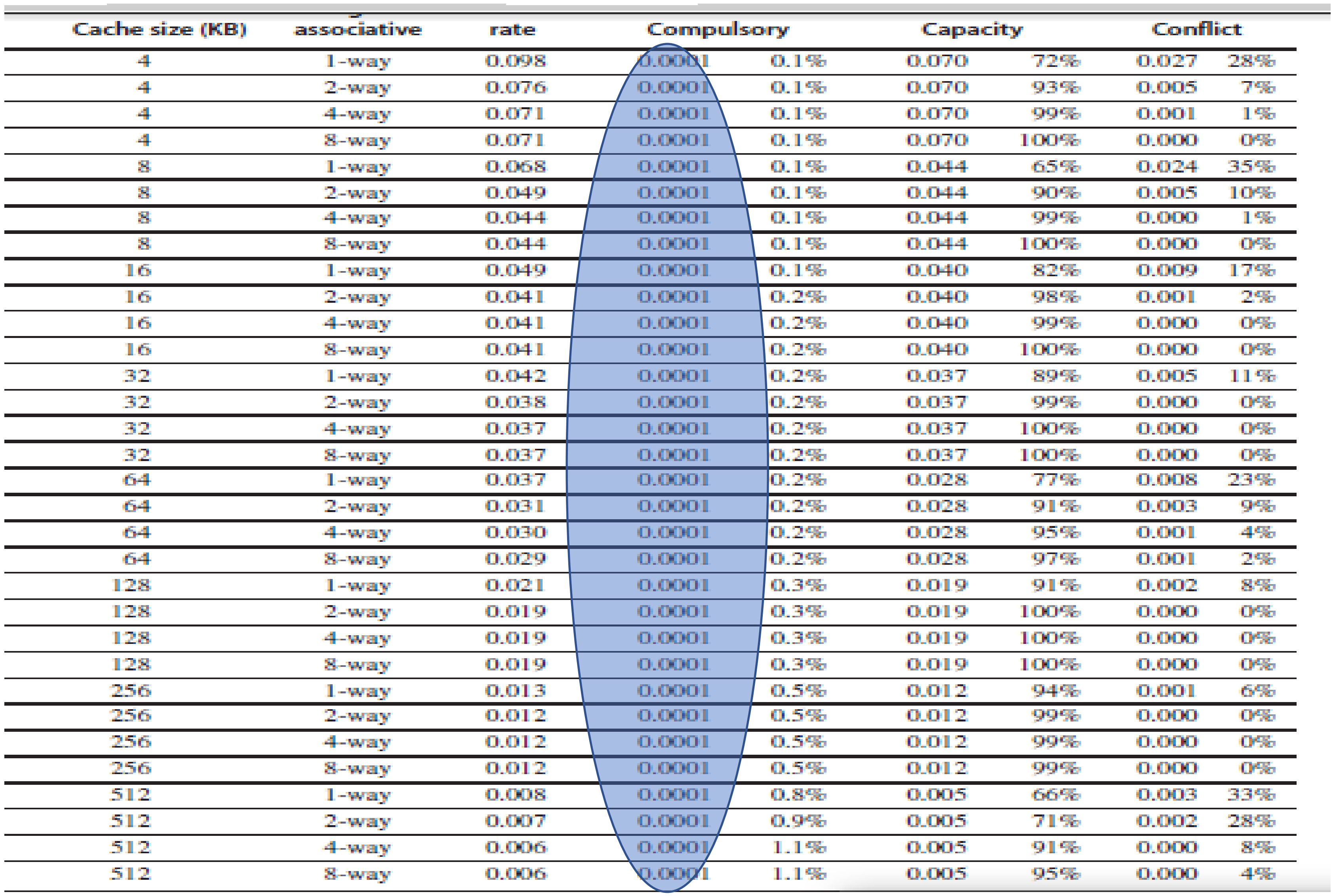


200 Compulsory Misses
200 Capacity Misses
600 Conflict Misses.

Few Facts of 3 C’s

Microprocessor & Computer Architecture (μpCA)

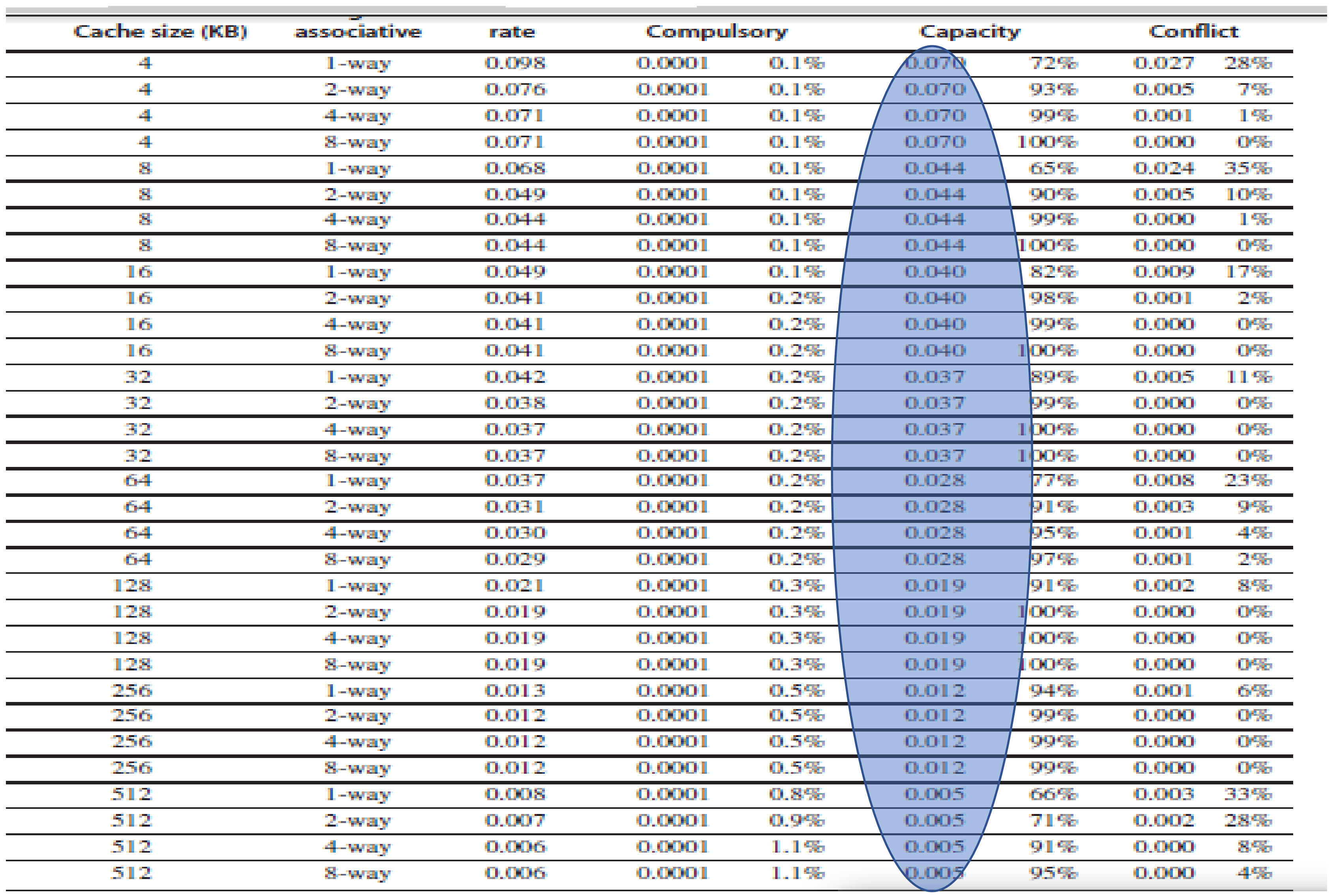
Compulsory Misses are Independent of Cache Size



Cache size (KB)	associative	rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

Microprocessor & Computer Architecture (μpCA)

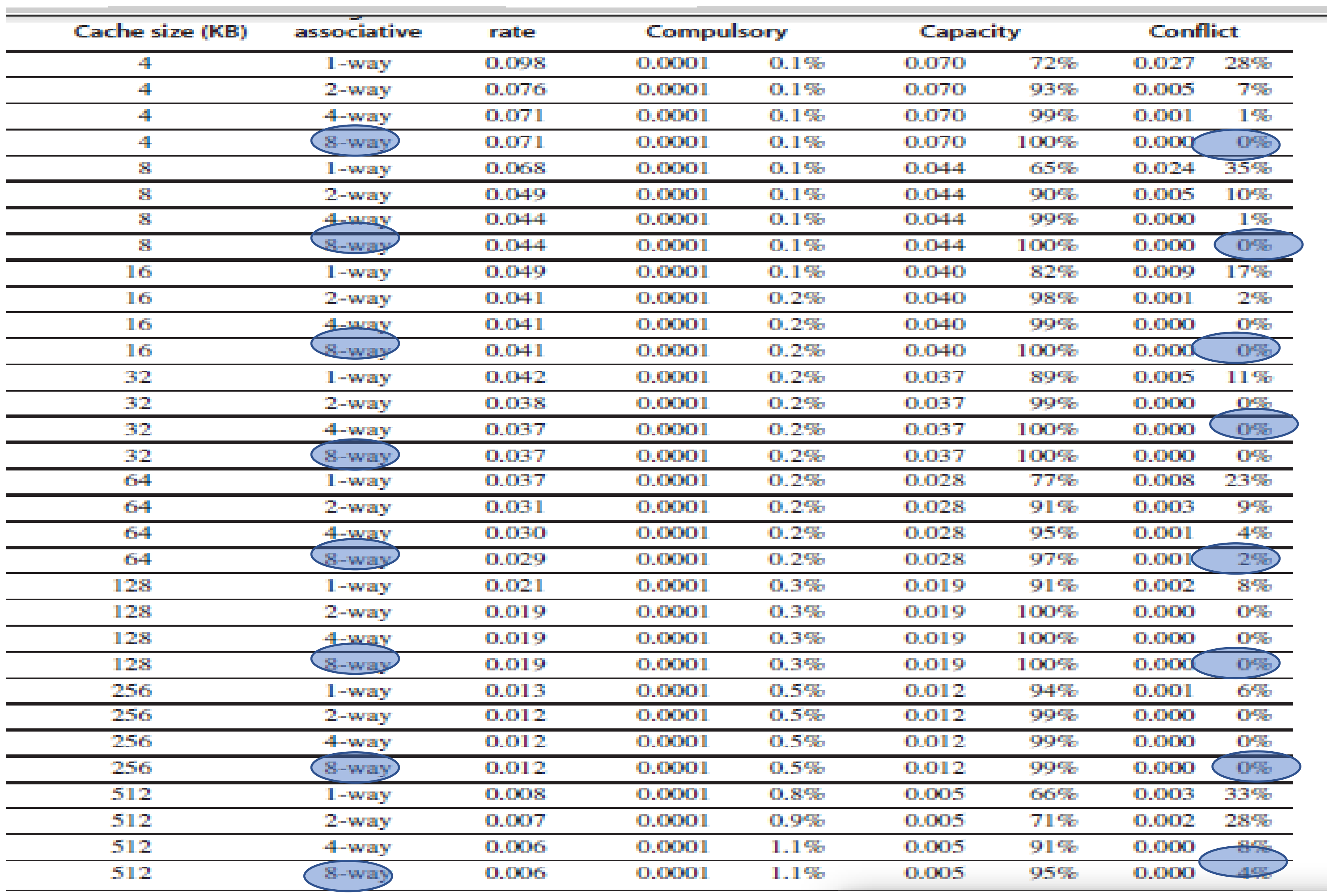
Capacity Misses Decrease as Cache Size Increases



Cache size (KB)	associative	rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

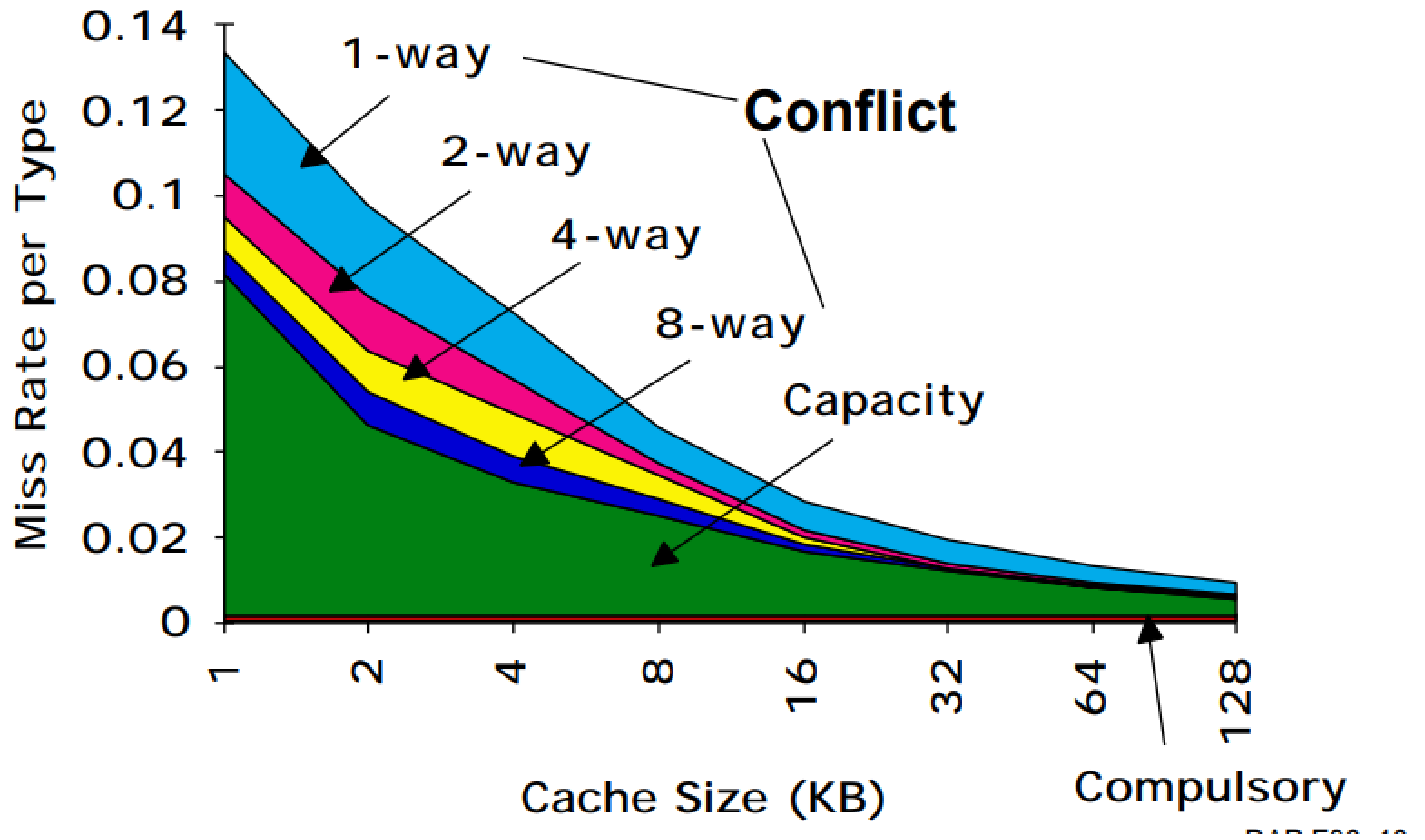
Microprocessor & Computer Architecture (μpCA)

Conflict Miss Decreases as the Associativity Increases



Cache size (KB)	associative	rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

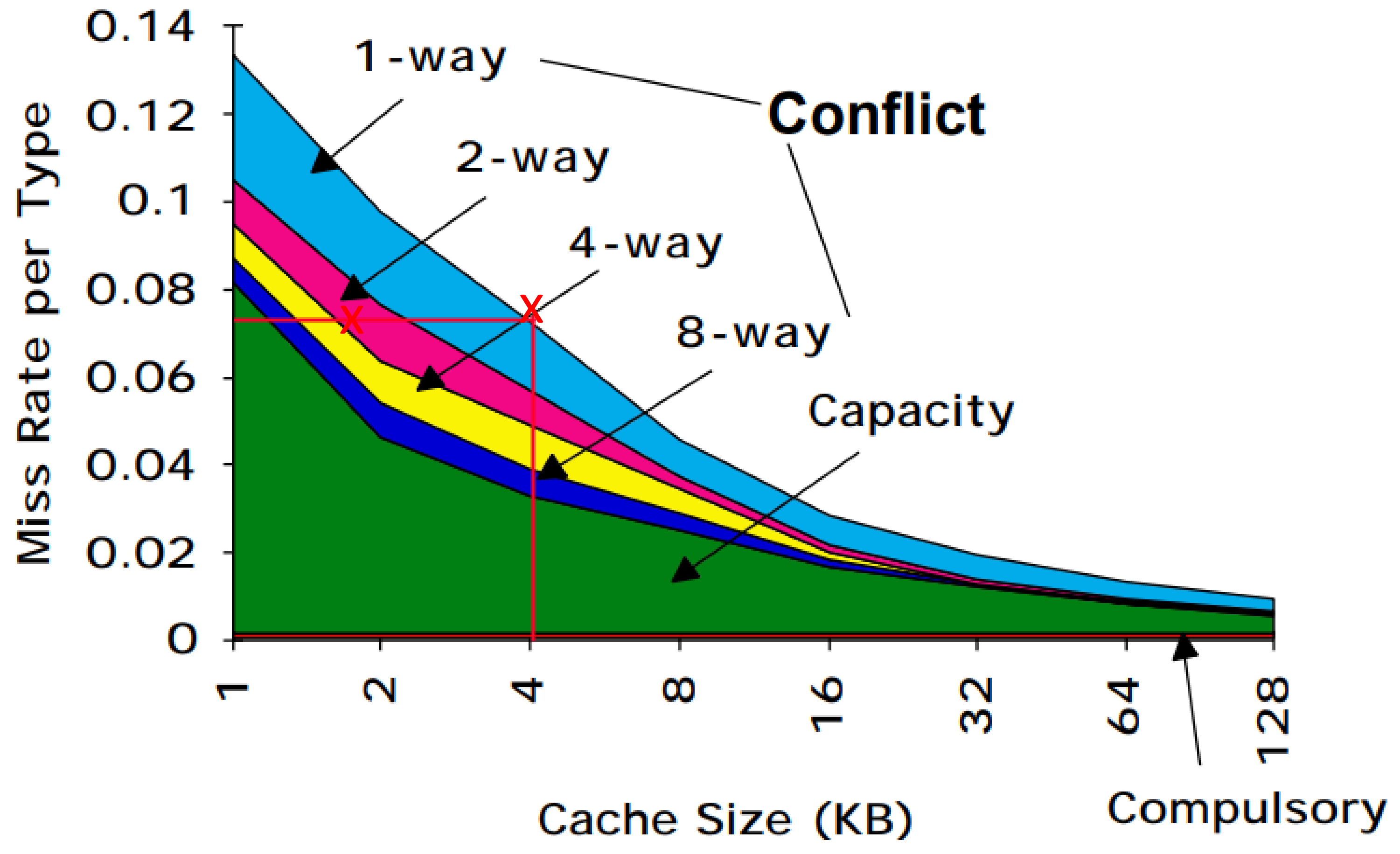
Microprocessor & Computer Architecture (μpCA)



Microprocessor & Computer Architecture (μpCA)

2:1 Rule

miss rate 1-way associative cache size X = miss rate 2-way associative cache size X/2



Reducing Miss Rate

Larger block size (to Reduce Compulsory Miss)

•Larger total **cache** capacity to reduce miss rate. (To Reduce Miss Capacity Miss)

•Higher associativity. (To Reduce Conflict Miss)

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135

Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Unit 4: Cache Optimization

Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus

~~Unit 1: Basic Processor Architecture and Design~~

~~Unit 2: Pipelined Processor and Design~~

~~Unit 3: Memory~~

Unit 4: Input/Output Device Design

- ~~3 C~~
- ~~Introduction to Cache Optimization~~
- **Recuse Miss Rate**

Unit 5: Advanced Architecture

Reducing Miss Rate!!!

Optimization 1: Large Block Size

Optimization 2: Large Cache Capacity

Optimization 3: Higher Associativity

Optimization 1: Large Block Size to Reduce Miss Rate

Advantage:

- Satisfy Spatial Locality
- Reduces Compulsory Misses

Example: Instead of 4 word Block, use 8 word Block or 16 word Block

Optimization 1: Large Block Size to Reduce Miss Rate

Disadvantage:

- Increase Miss Penalty (Bus width Issues)

Example: If Width of Bus is 4 word, 2 transfers for 8 word Block and 4 transfers for 16 word Block are required.

- Increase Conflict Miss, as Number of Cache Blocks reduces due increase in Block Size.

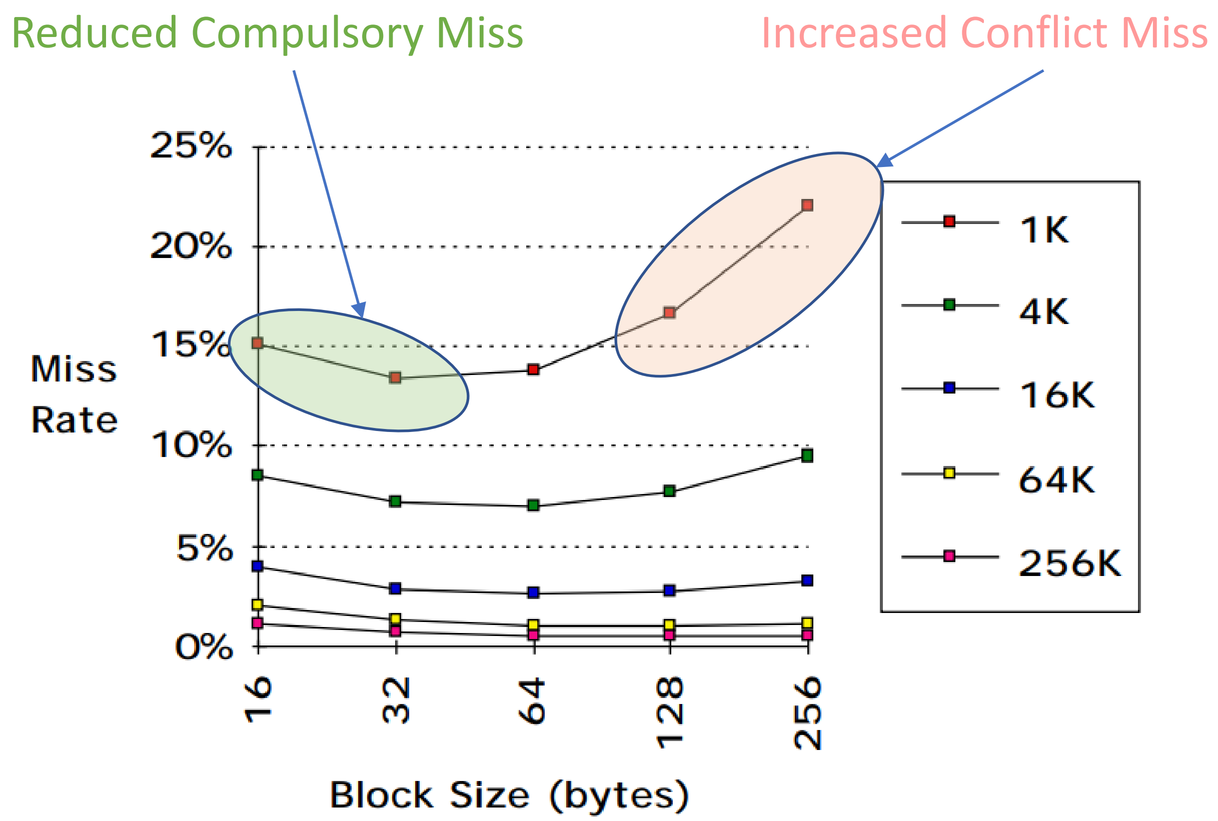
Example: A 32 word cache can be designed as

  - Eight, 4 word block
  - Four, 8 word block or
  - Two, 16 word block

- May bring useless data into cache as Spatial Locality cannot be maintained in entire program consistently.

Microprocessor & Computer Architecture (μpCA)

Optimization 1: Large Block Size to Reduce Miss Rate



- Performance keeps improving to a limit.
- With fewer number of cache block, increases conflict misses and thus overall cache miss rate

Optimization 2: Large Cache to Reduce Miss Rate

Advantage:

- Reduces Capacity Miss
- Can accommodate large memory footprint.

Example: If number of Blocks are 4096:

- On a cache with 128 Lines, 32 possible Blocks are mapped under direct mapping.

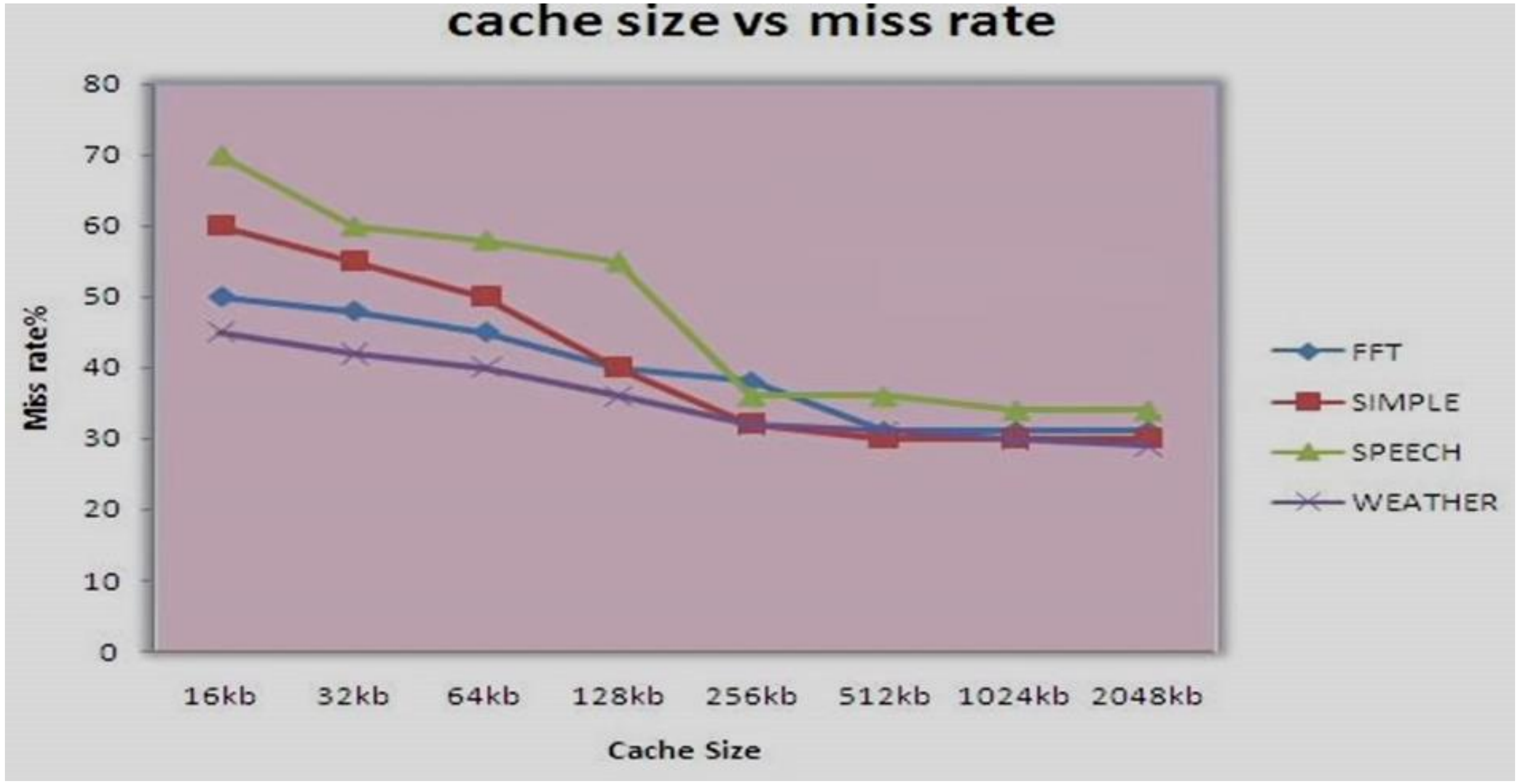
- On a cache with 256 Lines, Only 16 possible Blocks are mapped under direct mapping.

Optimization 2: Large Cache to Reduce Miss Rate

Disadvantage:

- Increases Hit Time
- High Cost, Area and Power

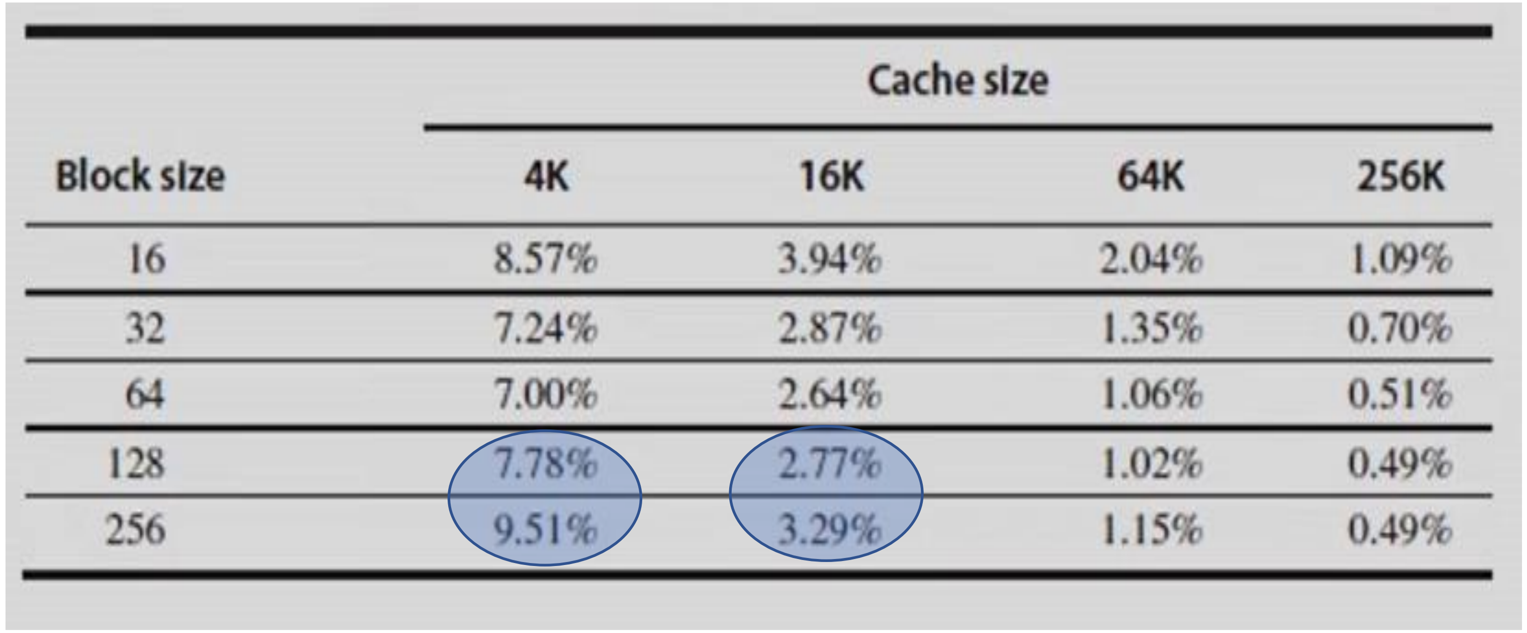
Optimization 2: Large Cache to Reduce Miss Rate



Microprocessor & Computer Architecture (μpCA)

Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



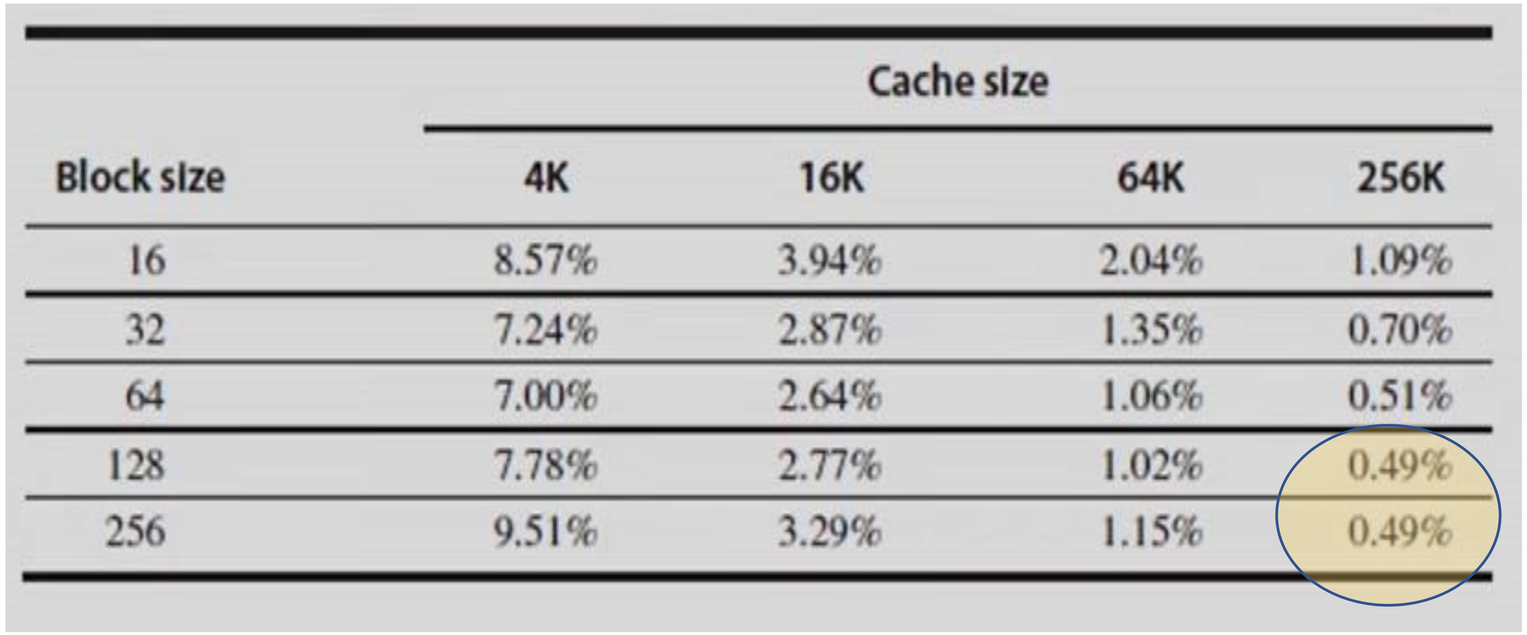
	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Observation1:
Miss Rate Increases if Cache Size is small and Block size is Large

4 K Cache with 256 Block size has highest Miss Rate

Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

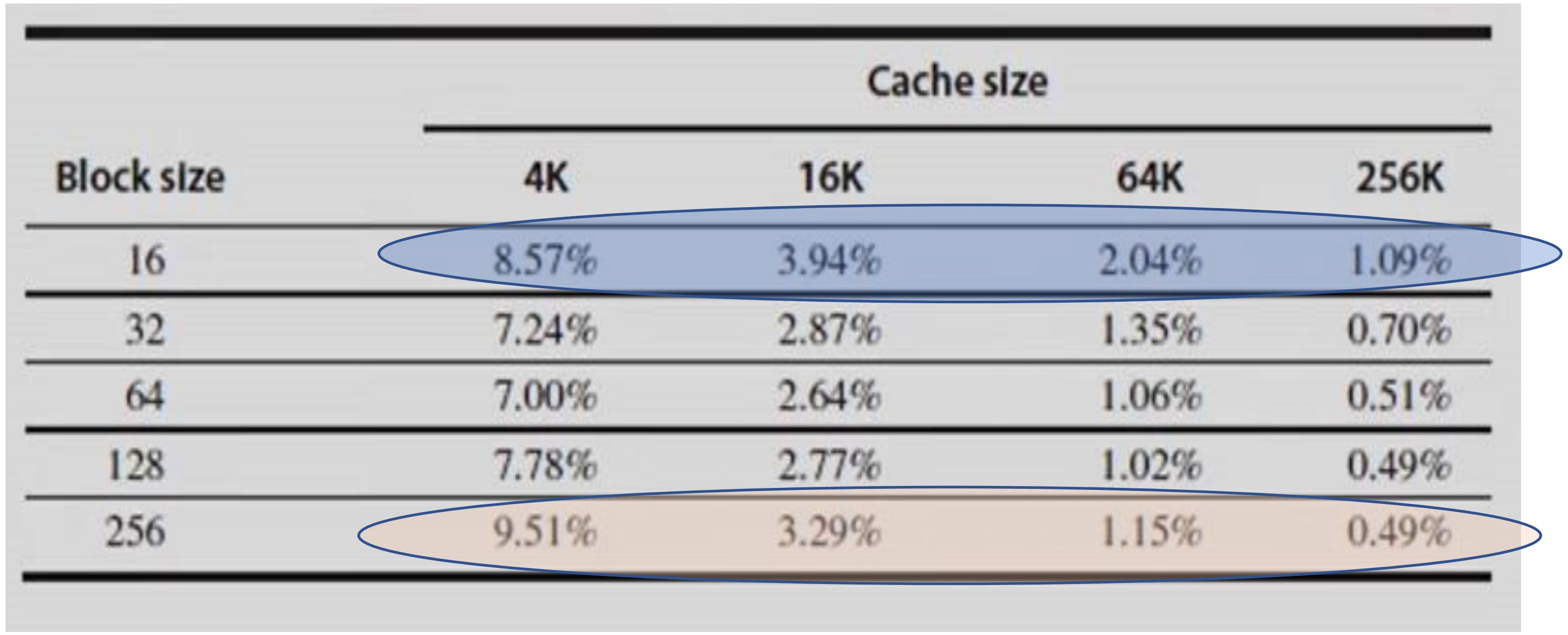
Observation2:
Miss Rate Decreases if Cache Size is Large and Block size is Large

256 k Cache with 256 Block Size has less Miss Rate

Microprocessor & Computer Architecture (μpCA)

Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Observation 3:

Miss Rate Decreases if Cache Size is Large and Block size is Small

Miss Rate Decreases if Cache Size is Large and Block Size is Large

However, If Cache Size if large, Hit Time, Space, Power and Cost Increases.

Microprocessor & Computer Architecture (μpCA)

Optimization 1 & 2: Reduce Miss Rate

Example 1: Assume the memory system takes 80 clock cycles of overhead and then delivers 16 bytes every 2 clock cycles. That is, it can supply 16 bytes in 82 clock cycles, 32 bytes in 84 clock cycles, and so on… Which block size has the smallest average memory access time for each cache size?

Ans: Average access time = Hit time + Miss rate x Miss penalty

If we assume that the hit time is 1 clock cycle independent of the block size, then,

The access time for a 16- byte block in a **4KB cache is**

Average access time = 1 + (8.57% x 82)

= 1 + 7.0274

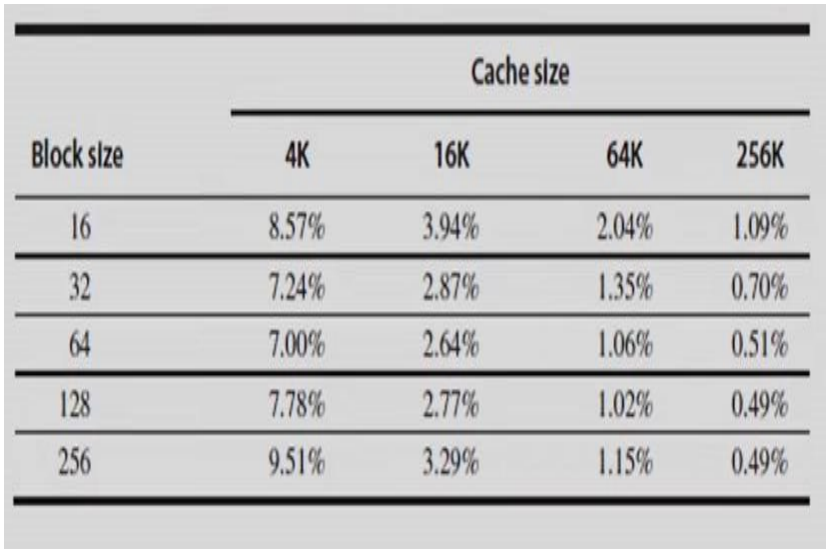
= **8.0274 clock cycles.**

For, 256 byte block in a 256 KB cache the

Average access time = 1 + (0.49% x (80+32))

= 1 + (0.5488)

= **1.5488 clock cycles.**



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Optimization 3: Large Associativity to Reduce Miss Rate

Higher Associativity to Reduce Miss Rate

- Fully Associative cache are best, as no replacement is required till cache is full.
- Set Associative cache balances the Search Time and Replacement.

Advantage:

- Reduce Conflict Miss
- Reduce Miss Rate and Eviction rate.

Disadvantage:

- Increases Hit Time (Due to increase in search time by comparing each TAG)
- Complex design than Direct Map.

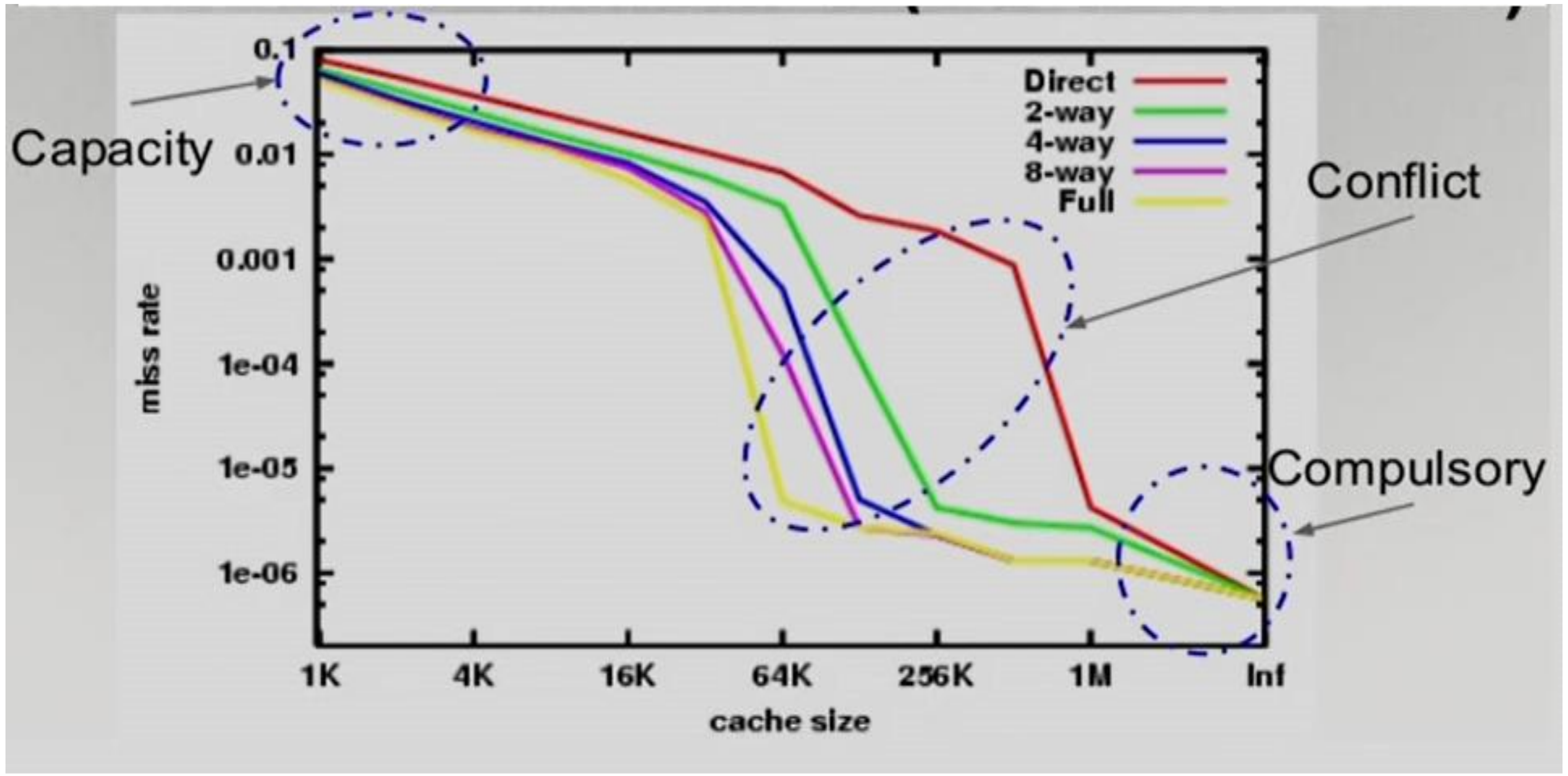
Replacement

- 2-way is better than Direct Mapping
- 4-way is better than 2-way
- 8-way is better than 4-way,

Search time

8-way, 16-way may become closer to Fully Associative.

Optimization 3: Large Associativity to Reduce Miss Rate



Replacement

- 2-way is better than Direct Mapping
- 4-way is better than 2-way
- 8-way is better than 4-way,

Search time

8-way, 16-way may become closer to Fully Associative.

Optimization 3: Large Associativity to Reduce Miss Rate

What about AMAT?
AMAT vs Cache Associativity

Assumption 1: Higher Associativity would increase the clock cycle time and is as listed below:

Clock Cycle Time$_{2\_way}$= 1.36 x Clock Cycle Time $_{1\_way}$

Clock Cycle Time$_{4\_way}$= 1.42 x Clock Cycle Time $_{1\_way}$

Clock Cycle Time$_{8\_way}$= 1.52 x Clock Cycle Time $_{1\_way}$

Assumption 2: Hit time is 1 Clock and Miss Penalty= 25 Clock Cycle

AMAT= Hit Time + Miss Rate x Miss Penalty

access time $_{8\text{-way}}$ = 1.52 + Miss Rate x 25

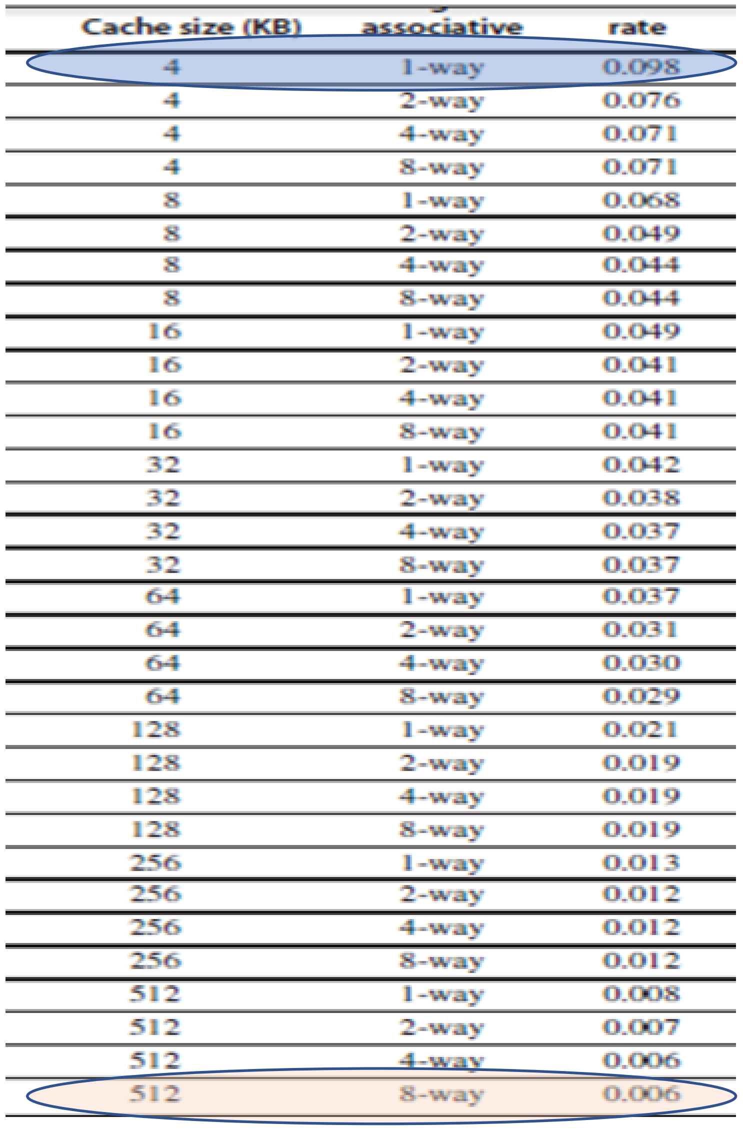
access time 4$_{\text{-way}}$ = 1.44 + Miss Rate x 25

access time 2$_{\text{-way}}$ = 1.36 + Miss Rate x 25

access time $_{1\text{way}}$ = 1.00 + Miss Rate x 25

Microprocessor & Computer Architecture (μpCA)

Optimization 3: Large Associativity to Reduce Miss Rate



Cache size (KB)	associative	rate
4	1-way	0.098
4	2-way	0.076
4	4-way	0.071
4	8-way	0.071
8	1-way	0.068
8	2-way	0.049
8	4-way	0.044
8	8-way	0.044
16	1-way	0.049
16	2-way	0.041
16	4-way	0.041
16	8-way	0.041
32	1-way	0.042
32	2-way	0.038
32	4-way	0.037
32	8-way	0.037
64	1-way	0.037
64	2-way	0.031
64	4-way	0.030
64	8-way	0.029
128	1-way	0.021
128	2-way	0.019
128	4-way	0.019
128	8-way	0.019
256	1-way	0.013
256	2-way	0.012
256	4-way	0.012
256	8-way	0.012
512	1-way	0.008
512	2-way	0.007
512	4-way	0.006
512	8-way	0.006

AMAT vs Cache Associativity

Example

The time for a 4KB Direct–Mapped cache is

Average memory access time $_{1\text{-way}}$ = 1.00 + (0.098 x 25) = 3.45

The time for a 512 KB, Eight-Way Set Associative cache is

Average memory access time $_{8\text{-way}}$ = 1.52 + (0.006 x 25) = 1.67

Example will make us to believe that

Average memory access time$_{8\text{-way}}$ < Average memory access time$_{4\text{-way}}$

Average memory access time$_{4\text{-way}}$ < Average memory access time$_{2\text{-way}}$

Average memory access time$_{2\text{-way}}$ < Average memory access time$_{1\text{-way}}$

However ☹

Optimization 3: Large Associativity to Reduce Miss Rate

AMAT vs Cache Associativity

	Associativity			
Cache size (KB)	1-way	2-way	4-way	8-way
4	3.44	3.25	3.22	**3.28**
8	2.69	2.58	2.55	**2.62**
16	2.23	**2.40**	**2.46**	**2.53**
32	2.06	**2.30**	**2.37**	**2.45**
64	1.92	**2.14**	**2.18**	**2.25**
128	1.52	**1.84**	**1.92**	**2.00**
256	1.32	**1.66**	**1.74**	**1.82**
512	1.20	**1.55**	**1.59**	**1.66**

High Associativity Leads to Higher Access Time

Microprocessor & Computer Architecture (μpCA)

Next Session

Reducing Miss Penalty

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135

Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Unit 4: Cache Optimization

Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus

~~Unit 1: Basic Processor Architecture and Design~~

~~Unit 2: Pipelined Processor and Design~~

~~Unit 3: Memory~~

Unit 4: Input/Output Device Design

~~3 C~~

~~Introduction to Cache Optimization~~

~~Recuse Miss Rate~~

Reduce Miss Penalty

4th Optimization: Multilevel Caches to Reduce Miss Penalty

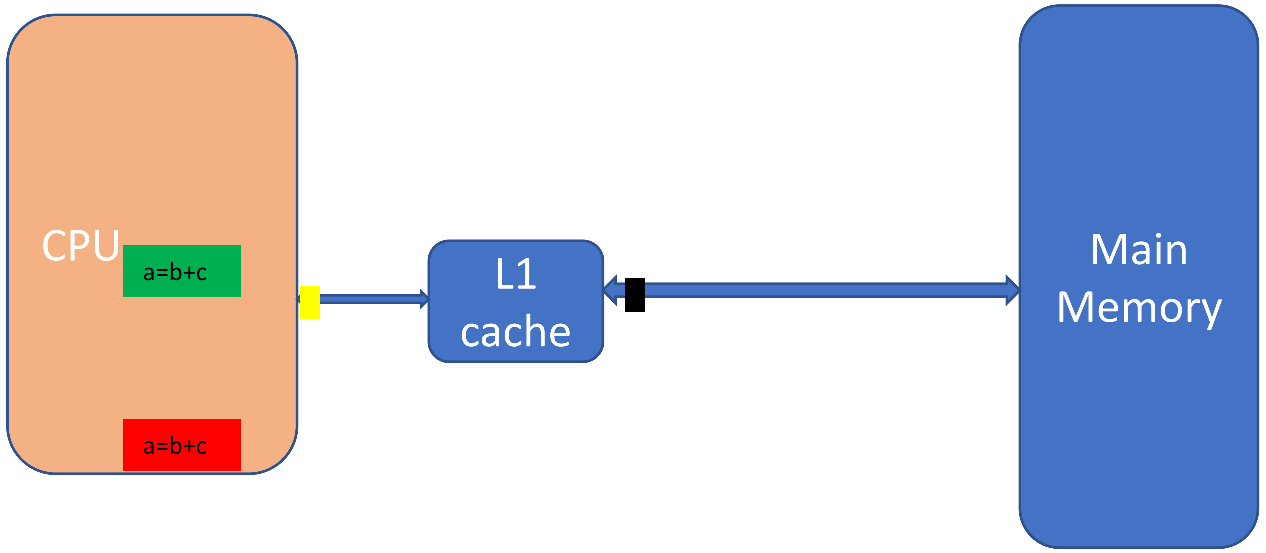
5th Optimization: Giving priority to Read misses over Write misses to reduce miss penalty

Unit 5: Advanced Architecture

Microprocessor & Computer Architecture (μpCA)

What is the Problem?

AMAT = Hit time + Miss rate x Miss penalty



- **Reducing Miss penalty is also important similar to Reducing miss rate.**

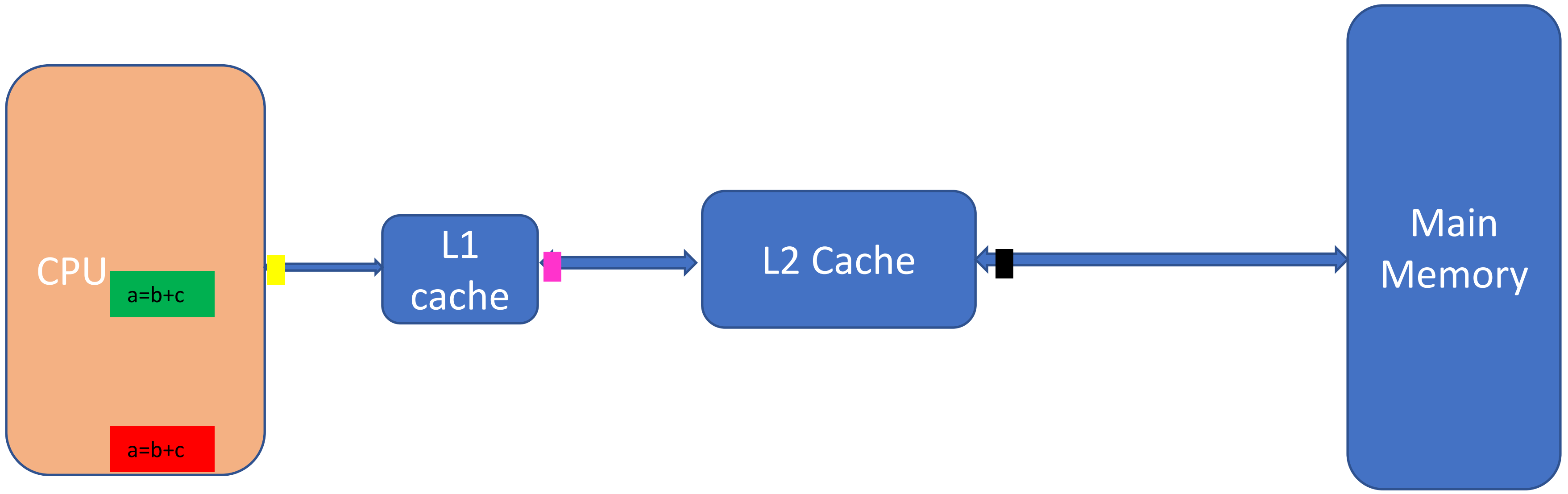
•The performance gap between processor & memory raises a question:

–Should I make the cache faster to keep the pace with the speed of the processor? **Or**

–Make the cache larger to overcome the widening gap between the processor & the main memory?

What is the Problem?

- Answer for these questions is to do both.
  - –**Adding another level of cache between memory & original cache simplifies the decision.**
- First level cache can be small enough to match the clock cycle time of the processor.
- Second level cache be can be large enough to capture many accesses that would go to main memory, thus reducing miss penalty.



AMAT? For two Level Cache

•The original formula is:

$$\textbf{AMAT} = \textbf{Hit Time}_{L1} + \textbf{Miss Rate}_{L1} \times \textbf{Miss Penalty}_{L1}$$

Miss in Level 1 Cache, leads to access data from Level 2 cache

$$\textbf{Miss Penalty}_{L1} = \textbf{Hit Time}_{L2} + \textbf{Miss Rate}_{L2} \times \textbf{Miss Penalty}_{L2}$$

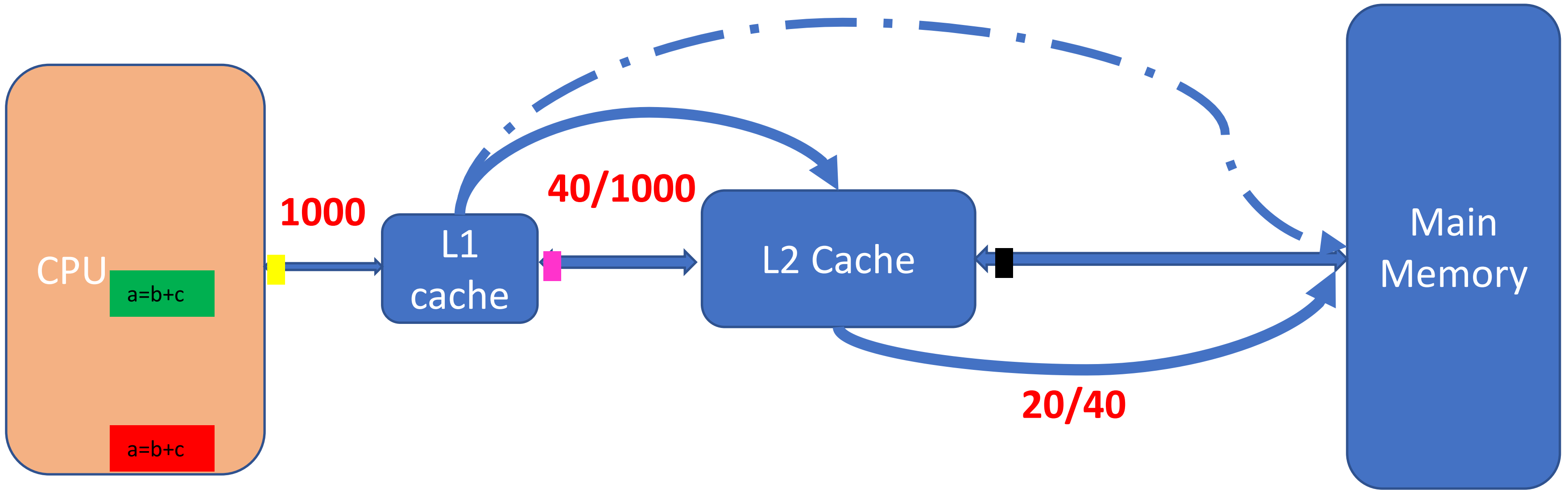
- Hit Time_{L2}= Item found in the Level 2 Cache
- Miss Rate_{L2}= How frequently element not found in Level 2 Cache
- Miss Penalty_{L2} = Extra time spent to bring item from RAM.

AMAT? For two Level Cache

$$\text{AMAT} = \text{Hit Time}_{L1} + \text{Miss Rate}_{L1} \times [\text{Hit Time}_{L2} + \text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2}]$$

Microprocessor & Computer Architecture (µpCA)

4th Optimization: Multilevel Caches to Reduce Miss Penalty



Suppose that in 1000 memory references there are 40 misses in the first level cache and 20 misses in the second –level cache.

What is Miss Rate of Level 1 Cache? **40/1000**

What is Miss Rate of Level 2 Cache? **20/1000** or **20/40**

4th Optimization: Multilevel Caches to Reduce Miss Penalty

Local miss rate:

•The number of misses in the cache divided by the total number of memory accesses to this cache.

Ex: For first level cache it is, **Miss Rate$_{L1}$** **(40/1000)**

For second level cache it is, **Miss Rate$_{L2}$** **(20/40)**

Global miss rate:

•The number of misses in the cache divided by the total number of memory accesses generated by the processor.

Ex: Global miss rate for level1 cache is still **Miss Rate$_{L1}$** **(40/1000)**

but, for level2 cache it is : **Miss Rate$_{L1}$ x Miss Rate$_{L2}$**

(40/1000) x (20/40)= (20/1000)

4th Optimization: Multilevel Caches to Reduce Miss Penalty

Suppose that in 1000 memory references there are 40 misses in the first level cache and 20 misses in the second –level cache. ***What are the various miss rates?***

Assume the miss penalty from the L2 cache to memory is 200 clock cycles, the hit time of the L2 cache is 10 clock cycles, hit time for L1 cache is 1 clock cycle.

What is the average memory access time ?

Solution:

The Miss rate [global & local] for the 1st level cache is 40/1000= 4%
The local miss rate for 2nd Level cache is 20/40 = 50%
The Global miss rate of 2nd Level cache is 20/1000=2%

$$\text{AMAT} = \text{Hit Time}_{L1} + \text{Miss Rate}_{L1} \times [\text{Hit Time}_{L2} + \text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2}]$$

$$\text{AMAT} = 1 + \underline{4\%} \times [10 + 50\% \times 200]$$

$$= 1 + 0.04 \times [10 + 0.5 \times 200]$$

$$= 1 + 0.04 \times 110$$

$$= 5.4$$

Average Memory Stalls Per Instruction

Average Memory Stalls Per instruction = (AMAT - Hit time$_{L1}$) x Average # of memory references per instruction

or

Average memory stalls per instruction = Misses per instruction $_{L1}$ x Hit time$_{L2}$ + Misses per instruction $_{L2}$ x Miss Penalty $_{L2}$

or

Average memory stalls per instruction = (Miss Rate $_{L1}$ x Hit time$_{L2}$ + Miss Rate $_{L2}$ x Miss Penalty $_{L2}$) *Memory Reference per Instruction

or

Average memory stalls per instruction = (Miss Rate $_{L1}$ x Miss Penalty$_{L1}$ + Miss Rate $_{L2}$ x Miss Penalty $_{L2}$)* Memory Reference per Instruction

Reference

Average Memory Stalls Per Instruction

Suppose that in 1000 memory references there are 40 misses in the first level cache and 20 misses in the second –level cache. ***What are the various miss rates?***

Assume the miss penalty from the L2 cache to memory is 200 clock cycles, the hit time of the L2 cache is 10 clock cycles, hit time for L1 cache is 1 clock cycle and **there are 1.5 memory references per instruction.**

What is the Average Stall Cycles Per Instruction?

Solution:

# of misses per instruction $= \dfrac{\#\,of\ memory\,references}{\#\,of\ memory\,references\,per\ instructions} = \dfrac{1000}{1.5}$ **= 667 ins.**

Thus, for L1 cache ,

# misses for 40 memory accesses for 1000 instructions = 40 x 1.5 = 60 misses and for 20 misses for L2 cache it is 1.5 x 20 = 30 misses.

Microprocessor & Computer Architecture (μpCA)

Average Memory Stalls Per Instruction

Average memory stalls per instruction = Misses per instruction $_{L1}$ x Hit time$_{L2}$ + Misses per instruction $_{L2}$ x Miss Penalty $_{L2}$

= (60/1000) x 10 + (30/1000) x 200

= 0.06 x 10 + 0.03 x200 = 0.06 + 6

= 6.6 clock cycles.

or

Average memory stalls per instruction = (AMAT - Hit time$_{L1}$) x Memory references per instruction

= (5.4 - 1.0) x 1.5 = 6.6 clock cycles.

or

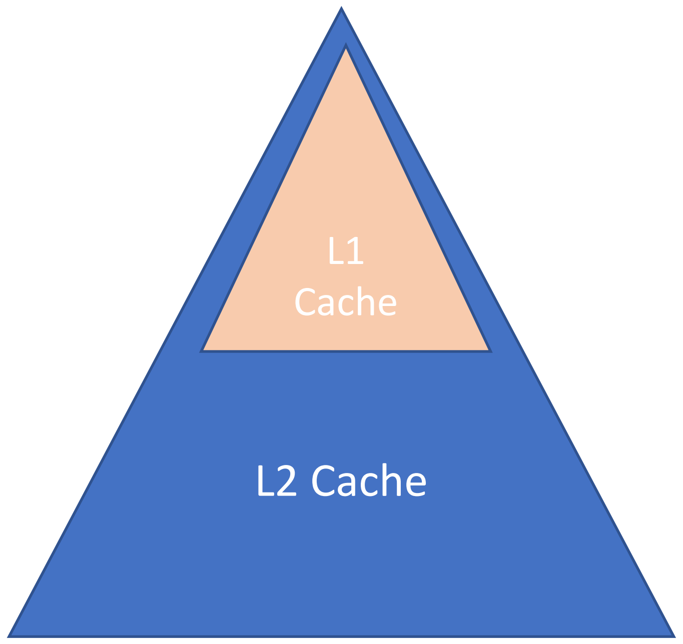
Average memory stalls per instruction = (Miss Rate $_{L1}$ x Hit time$_{L2}$ + Miss Rate $_{L2}$ x Miss Penalty $_{L2}$)* Memory Reference per Instruction

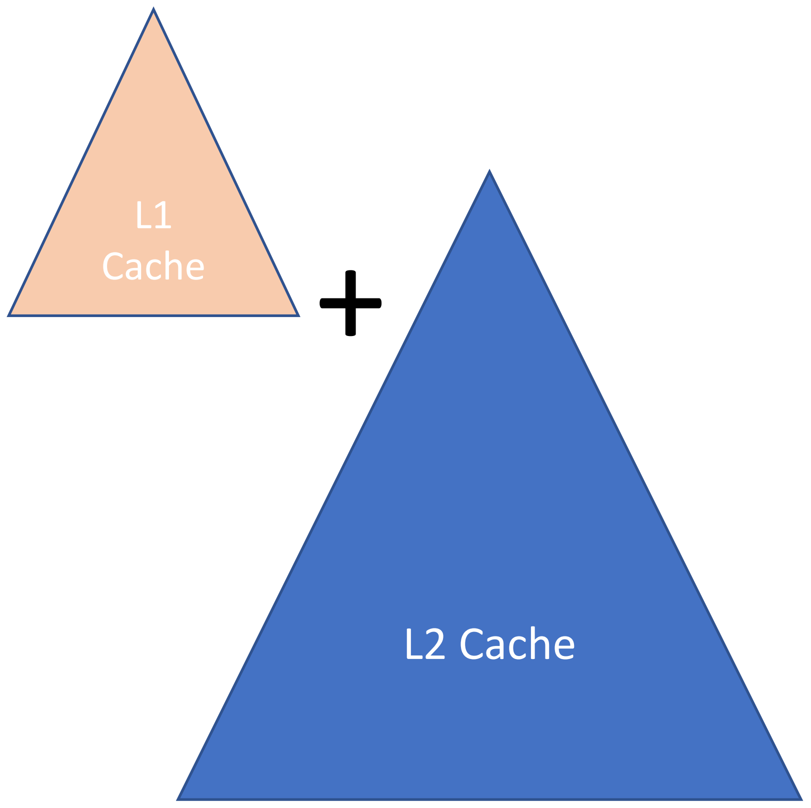
= ((40/1000*10)+(20/1000*200)) *1.5

= 6.6 Clock Cycles

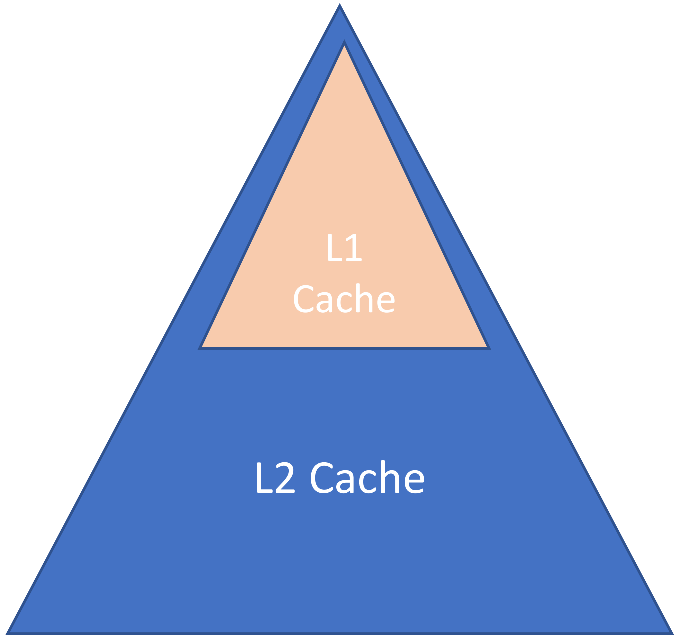
Why Global Miss Rate of 2nd Level Cache is Important?

Which one is Correct?





Why Global Miss Rate of 2nd Level Cache is Important?



First perspective :

Global cache miss rate is very similar to the single cache miss rate of the second level cache.

Second Perspective:

- Local cache miss rate is not a good measure of secondary caches.
- It is a function of the miss rate of the first level cache.
- Can vary by changing the first – level cache.

Note: Global cache miss rate should be used when evaluating second level caches.

Parameters for choosing 2nd Level Cache

- Miss Rate of 2nd level cache is function of 1st Level Cache.
- Speed of 2nd Level Cache will affect the Miss Penalty of 1st Level Cache.
- Design of 2nd Level cache should compensate all the deficiency in designing of 1st Cache.
- Thus, the strategies or Policy used in 2nd Level cache need not be same as design of 1st Level Cache.

Microprocessor & Computer Architecture (µpCA)

Parameters for choosing 2nd Level Cache

Example Given the data below, what is the impact of second-level cache associativity on its miss penalty?

- Hit time$_{L2}$ for direct mapped = 10 clock cycles.
- Two-way set associativity increases hit time by 0.1 clock cycle to 10.1 clock cycles.
- Local miss rate$_{L2}$ for direct mapped = 25%.
- Local miss rate$_{L2}$ for two-way set associative = 20%.

Answer For a direct-mapped second-level cache, the first-level cache miss penalty is

$$\text{Miss penalty}_{\text{1-way L2}} = 10 + 25\% \times 200 = 60.0 \text{ clock cycles}$$

Adding the cost of associativity increases the hit cost only 0.1 clock cycle, making the new first-level cache miss penalty:

$$\text{Miss penalty}_{\text{2-way L2}} = 10.1 + 20\% \times 200 = 50.1 \text{ clock cycles}$$

Microprocessor & Computer Architecture (μpCA)

5th Optimization: Giving priority to Read Misses over Write

Case 1: Write Back Policy

Scenario

- **P is referred by the Processor.**
- **P Should replace A.**
- **A is Dirty**

Known Solution

- **Write A back in Memory**
- **Replace P by A**

Cache

Valid Bit	Dirty Bit	TAG	Data
1	1	10100	A=11
1	0	10100	B=5
1	0	10100	C=6

Main Memory

10100.....1010..0000	**A= 0**
10100.....1010..0001	B=5
10100.....1010..0010	C=6
00010.....1010..0000	P=100

Optimized Solution (Give Priority for CPU Reference / Read)

- **Place A on to buffer to save time.**
- **Replace P by A, to provide quick access .**
- **Write A into memory parallelly when Processor is using P.**

Microprocessor & Computer Architecture (μpCA)

5th Optimization: Giving priority to Read Misses over Write

Case 2: Write Through

Scenario

- **P is referred by the Processor.**
- **P Should replace A.**

Main Memory

10100…..1010..0000	**A= 0**
10100…..1010..0001	B=5
10100…..1010..0010	C=6
00010…..1010..0000	P=100

Cache

Valid Bit	TAG	Data
1	10100	A=11
1	10100	B=5
1	10100	C=6

		A=11

Write Buffer

Microprocessor & Computer Architecture (μpCA)

5th Optimization: Giving priority to Read Misses over Write

Case 2: Write Through

Scenario

- **P is referred by the Processor.**
- **P Should replace A.**

Known Solution

- **Place A on the Write Buffer**
- **Replace P by A**

What may go wrong?

If A is referred again!

Cache

Valid Bit	TAG	Data
1	00010	P=100
1	10100	B=5
1	10100	C=6

		A=11

Write Buffer

Main Memory

10100.....1010..0000	**A= 0**
10100.....1010..0001	B=5
10100.....1010..0010	C=6
00010.....1010..0000	P=100

Solution1: Wait till A is written back in Memory

Solution2: (Give Priority for CPU Reference / Read)

Search for A in buffer, if found compare A in Memory,

i. if both are same, Replace.

ii. Otherwise, read from Buffer

Microprocessor & Computer Architecture (μpCA)

Consider the following code sequence.

```
Ex:   STR   R3, [R0,512]
      LDR   R1,  [ R0.1024])
      LDR  R2,  [R0,512]     Assume Direct mapped:
```

Write –through cache that maps 512 and 1024 to the same block. Four word write buffer that is not checked on a read miss. Will the value in R2 always be equal to the value in R3? o R2!

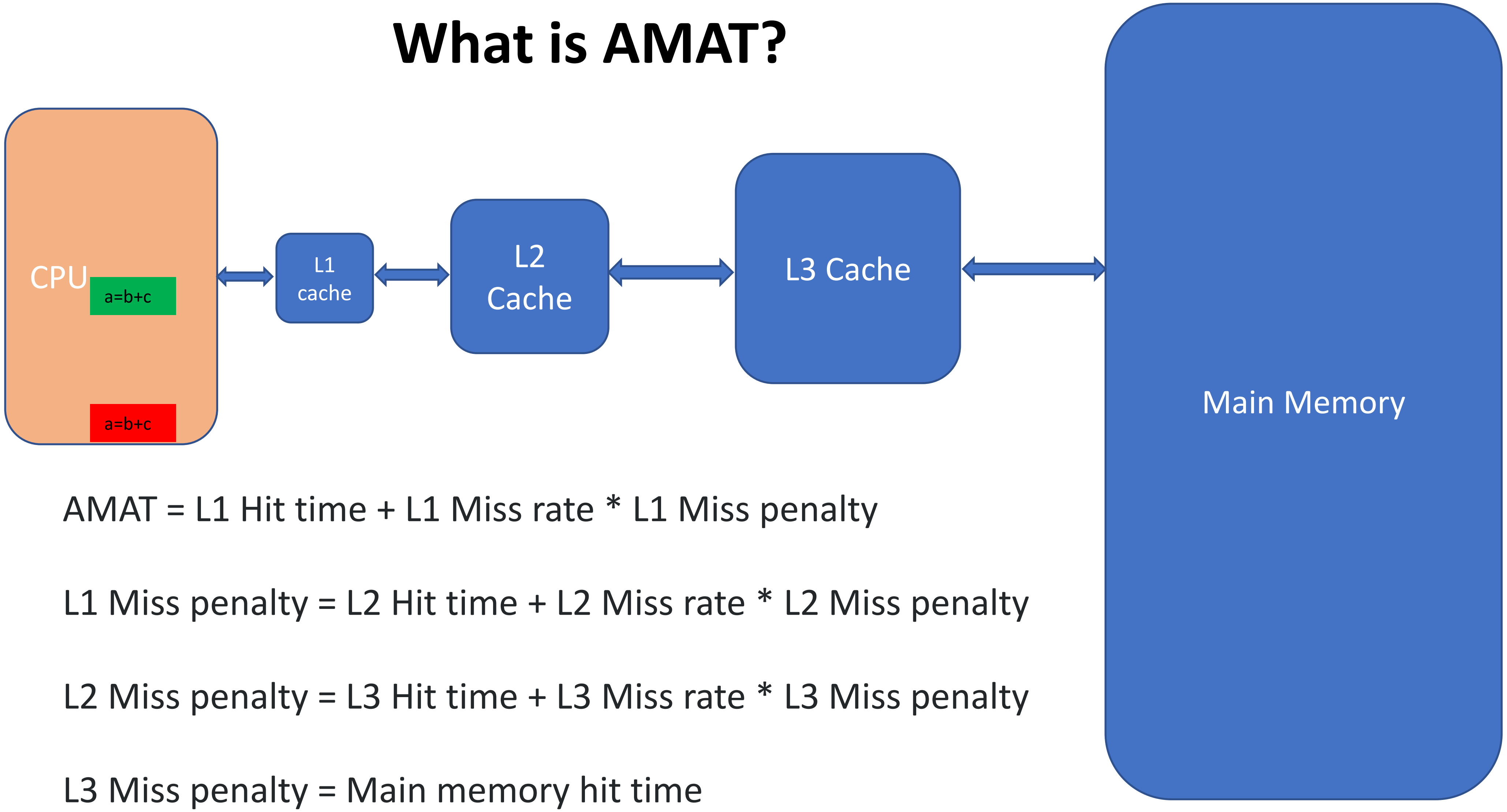
Microprocessor & Computer Architecture (μpCA)

Ans:

•This is a read-after-write data hazard in memory.
•The data in R3 are placed into the write buffer after the STR.
•The following LDR instruction uses the same cache index and is therefore a miss.
•The second LDR instruction, tries to put the value in location 512 into the register R2.
•This also results in a miss.
•If the write buffer hasn't completed writing to location 512 in memory,
•The read of location 512 will put the old, wrong value into the cache block and then into R2.
•Without proper precautions, R3 would not be equal to R2!

Think About it?

What is AMAT?



AMAT = L1 Hit time + L1 Miss rate * L1 Miss penalty

L1 Miss penalty = L2 Hit time + L2 Miss rate * L2 Miss penalty

L2 Miss penalty = L3 Hit time + L3 Miss rate * L3 Miss penalty

L3 Miss penalty = Main memory hit time

Microprocessor & Computer Architecture (μpCA)

Next Session

Improving Hit Time

PES UNIVERSITY ONLINE

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135

Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Unit 4: Cache Optimization

Dr. D. C. Kiran

Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus

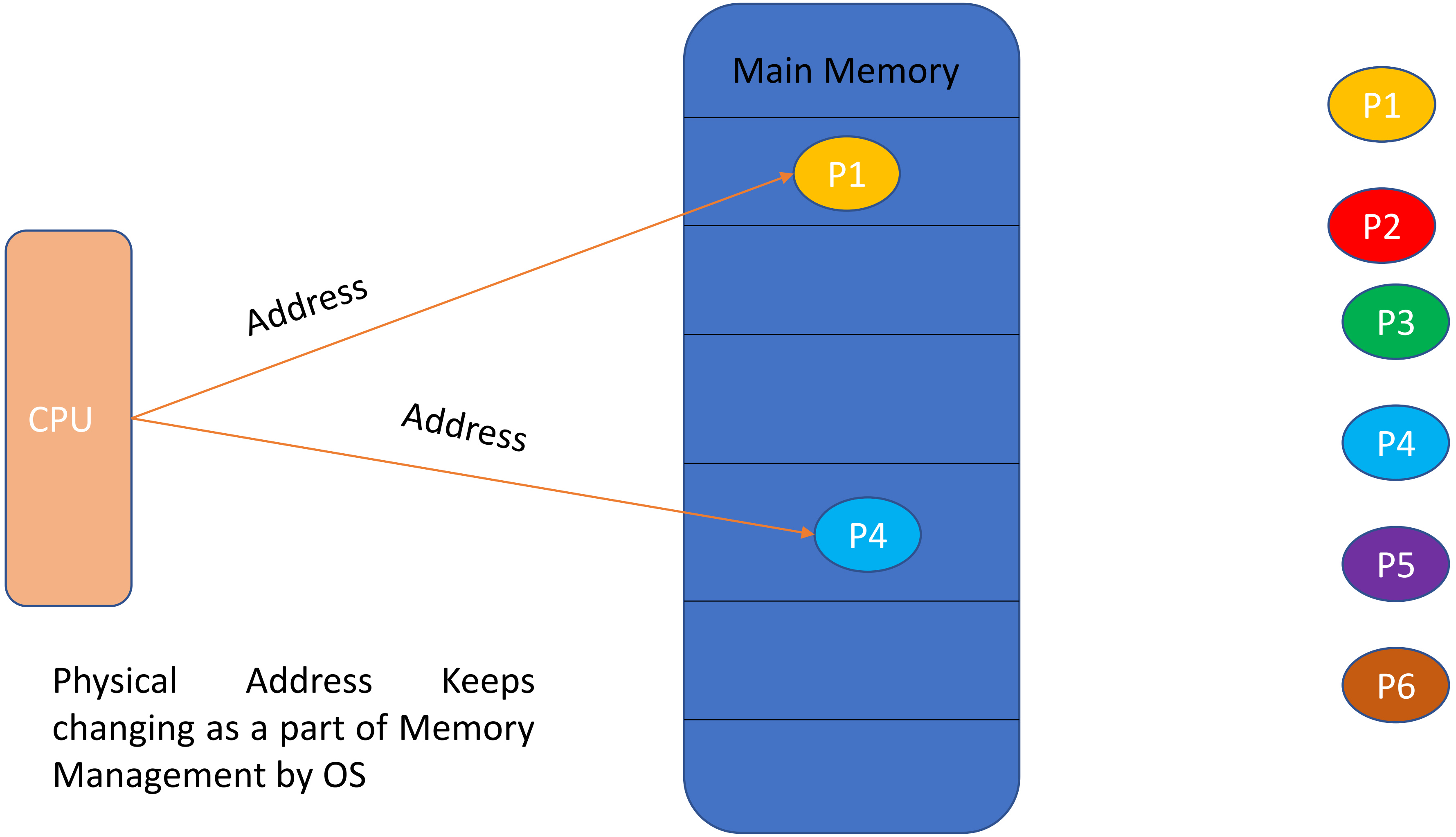
~~Unit 1: Basic Processor Architecture and Design~~
~~Unit 2: Pipelined Processor and Design~~
~~Unit 3: Memory~~
Unit 4: Input/Output Device Design

~~3 C~~
~~Introduction to Cache Optimization~~
~~Reduce Miss Rate~~
~~Reduce Miss Penalty~~
~~4th Optimization: Multilevel Caches to Reduce Miss Penalty~~
~~5th Optimization: Giving priority to Read misses over Write misses to~~
Reduce Hit Time

Unit 5: Advanced Architecture

6th Optimization: Reduce Hit Time

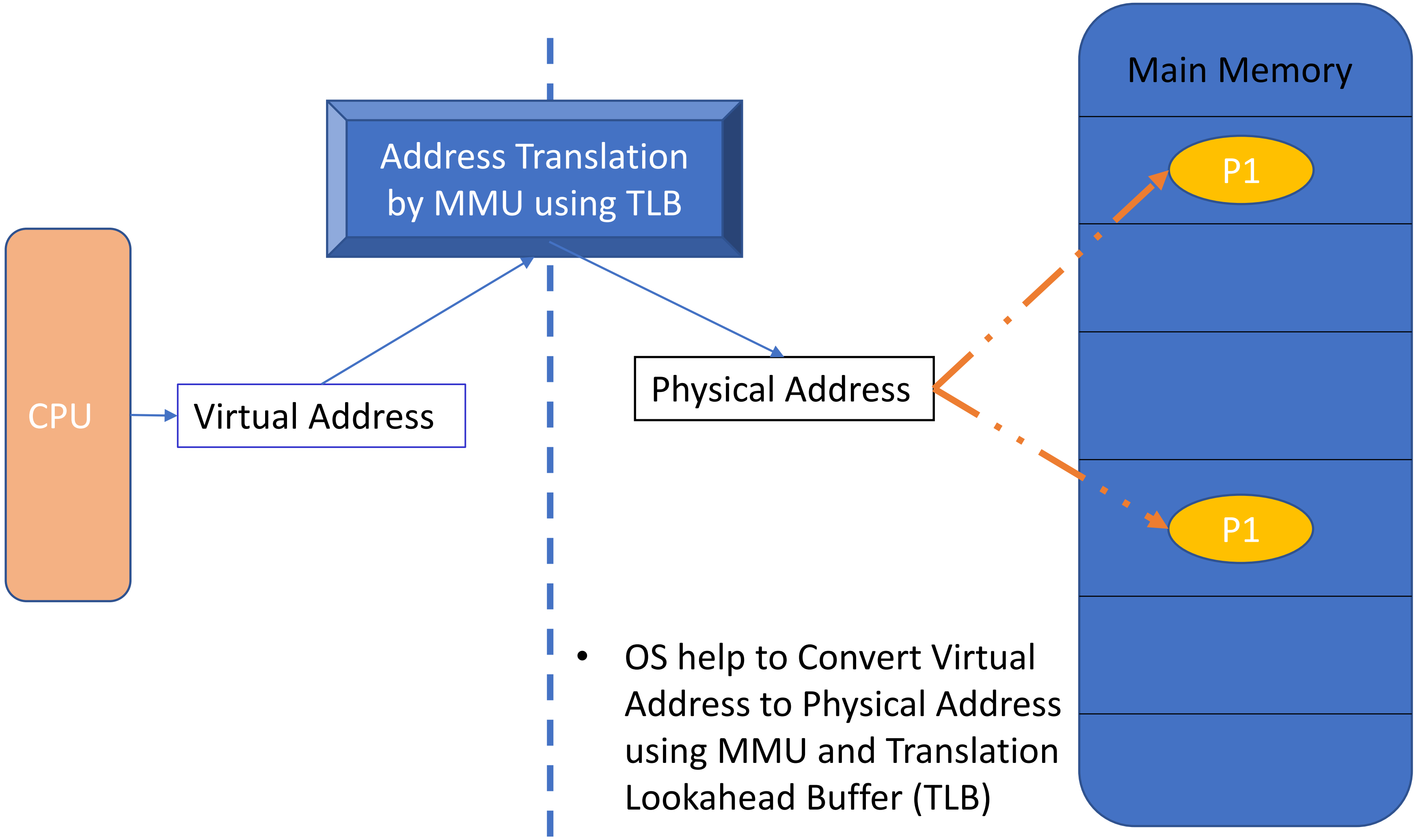
Avoid Address Translation In Cache Indexing To Reduce Hit Time



Physical Address Keeps changing as a part of Memory Management by OS

6th Optimization: Reduce Hit Time

Avoid Address Translation In Cache Indexing To Reduce Hit Time



- OS help to Convert Virtual Address to Physical Address using MMU and Translation Lookahead Buffer (TLB)

What is the Problem?

Accessing data from Cache involve:

- Indexing
- Tagging

Converting Virtual Address to Physical Address take some time.

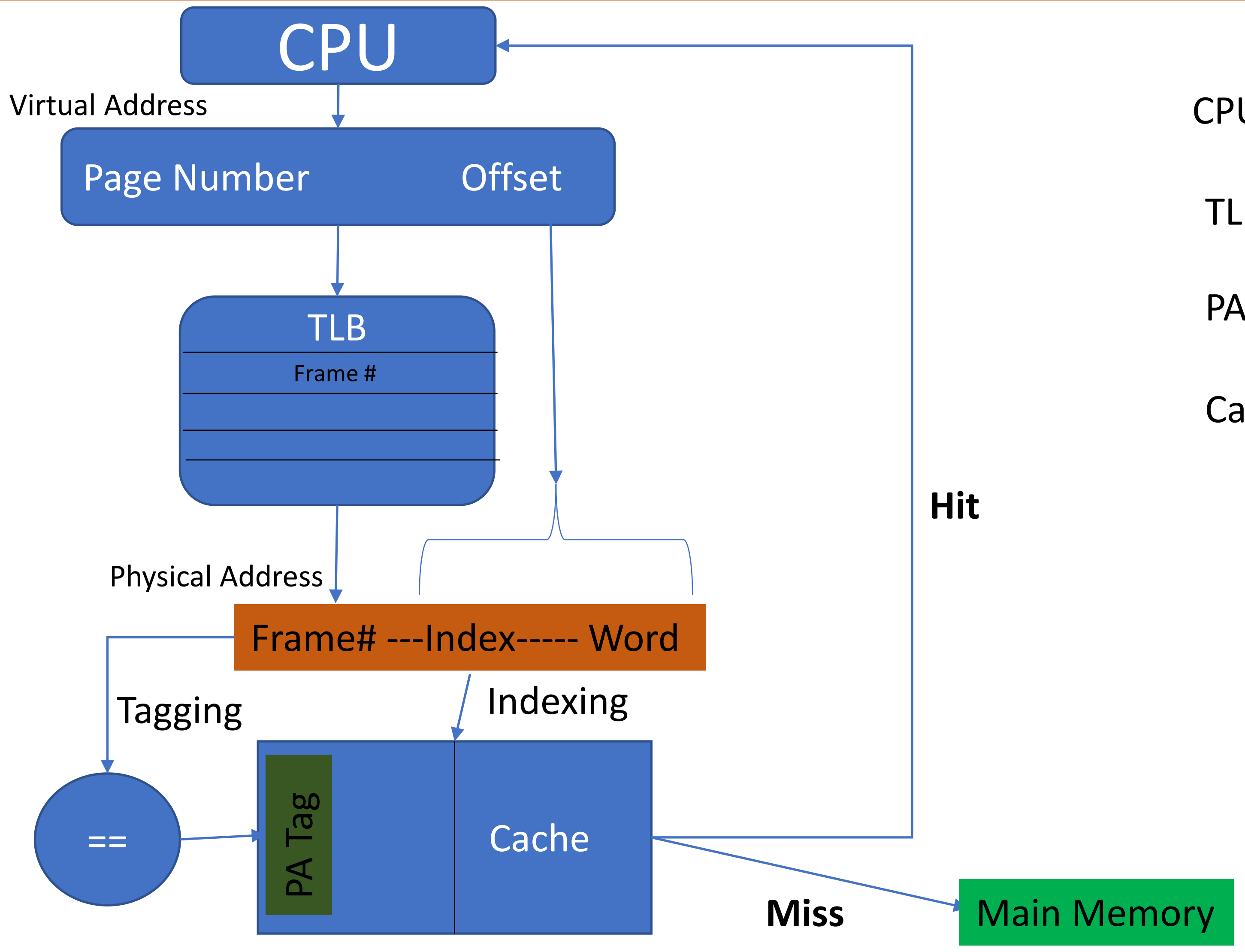
Solution ☺

Virtually Indexed & Physically Tagged

- Do not wait for VA to PA translation.
- Extract Index information from Virtual Address
- Extract Tag Information from the Physical Address.

Microprocessor & Computer Architecture (μpCA)

Physically Indexed and Physically Tagged (PIPT)



CPU generate VA

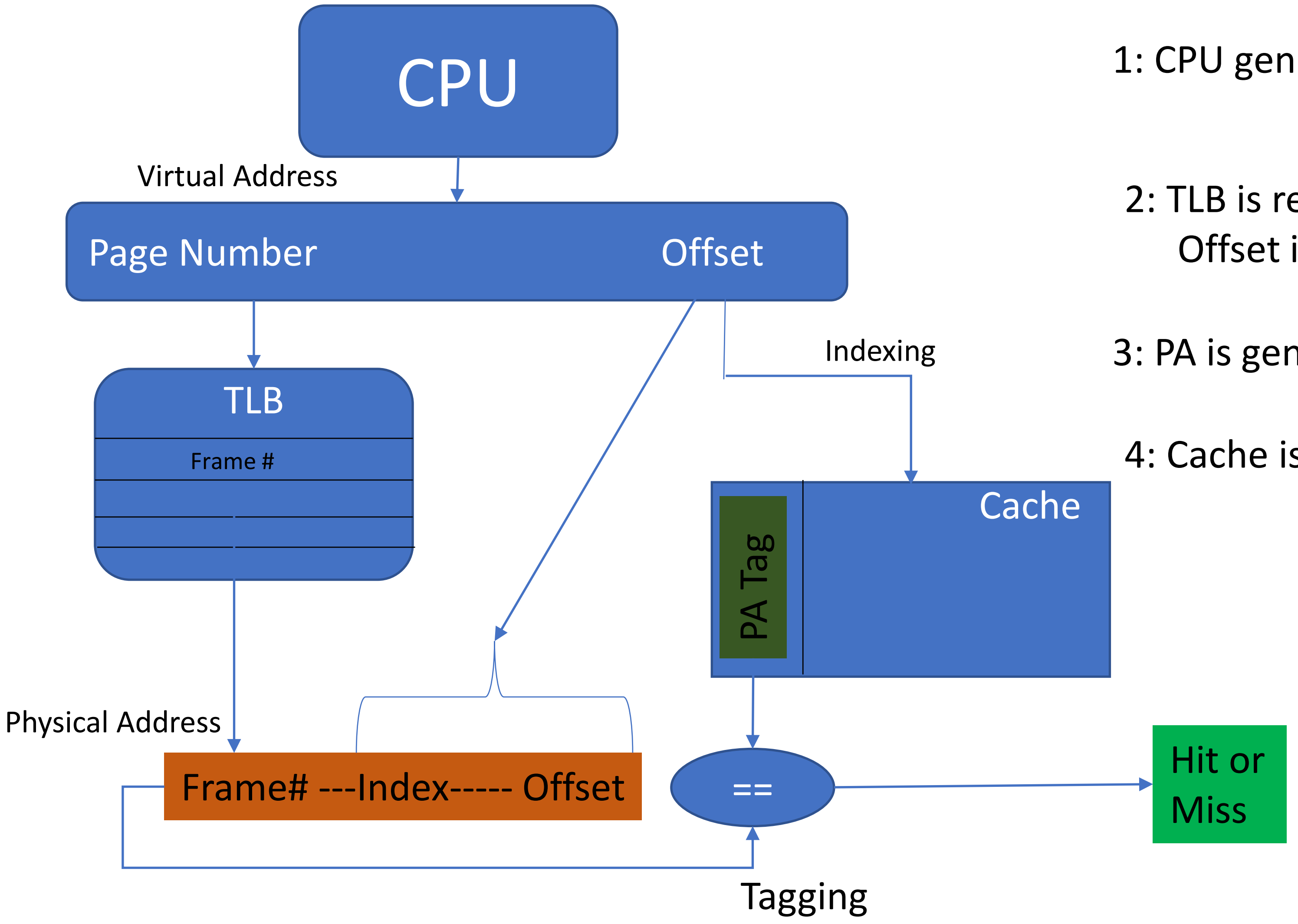
TLB is referred to check if required Frame present or not

PA is generated

Cache is Indexed and Tagged using PA

Microprocessor & Computer Architecture (μpCA)

Virtually Indexed and Physically Tagged



1: CPU generate VA

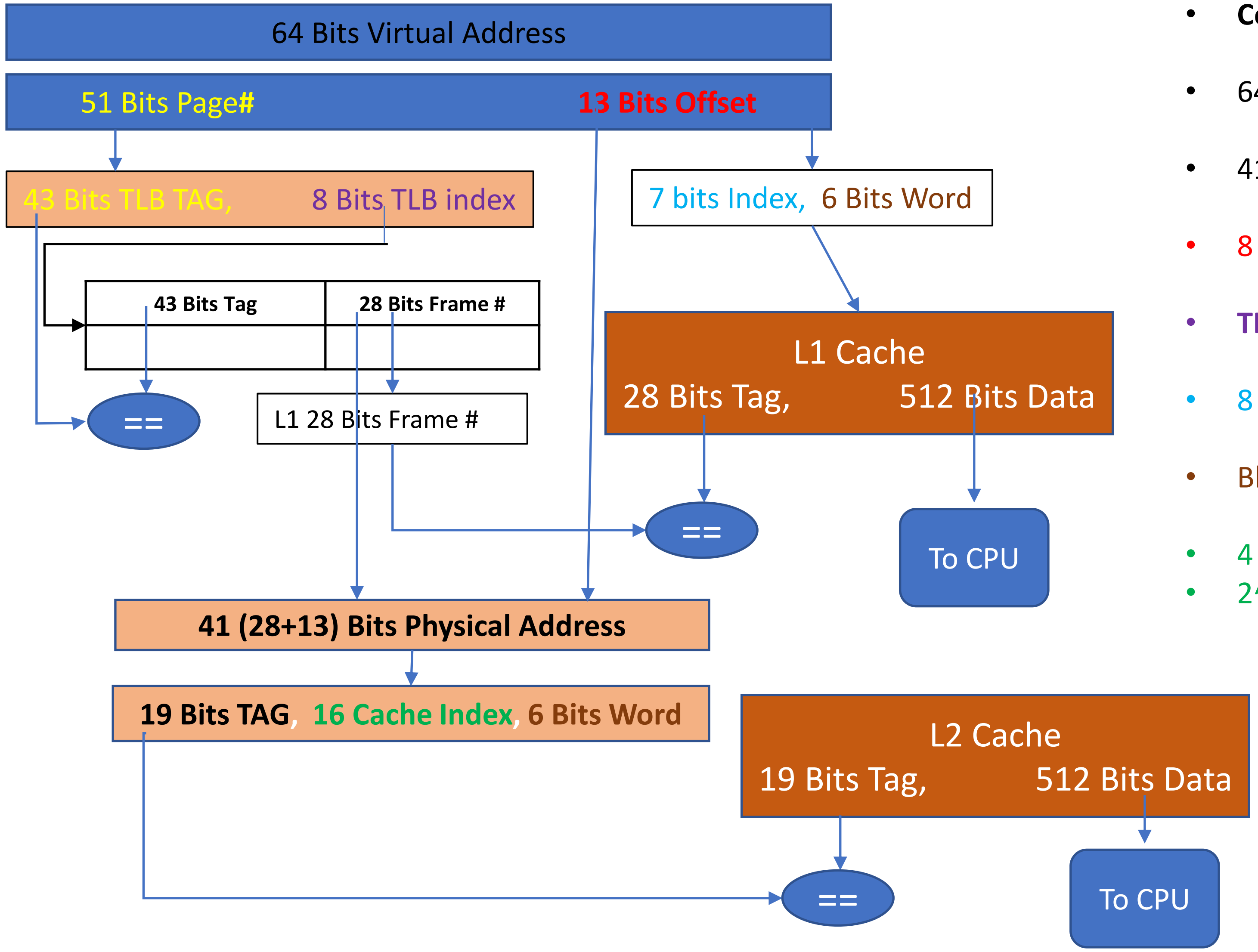
2: TLB is referred to check if required Frame present or not & Offset is used to Index Cache

3: PA is generated

4: Cache is Tagged using PA

Microprocessor & Computer Architecture (μpCA)

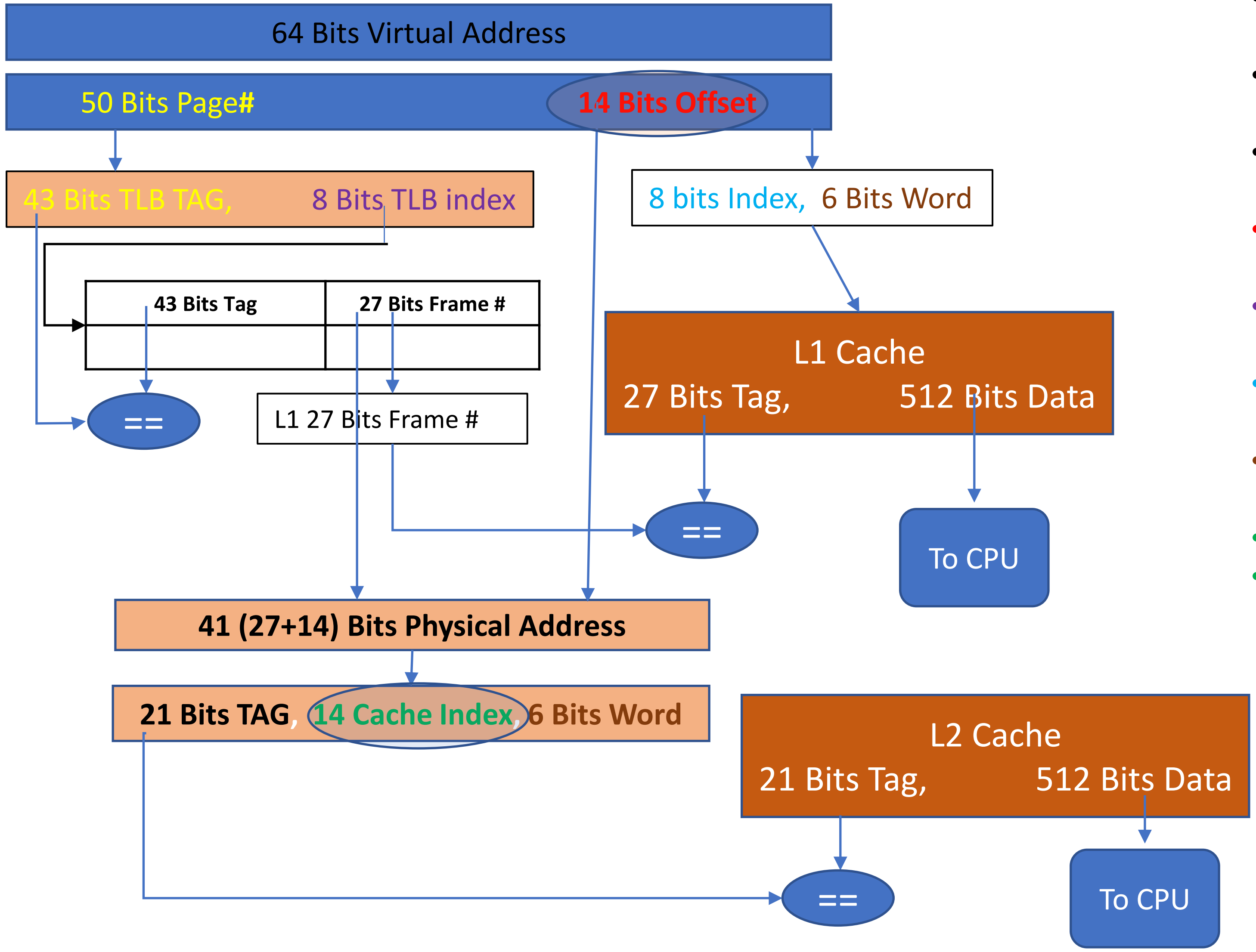
6th Optimization : Case Study



- **Consider the following Specification**
- 64 bit Virtual Address
- 41 bit Physical Address
- 8 KB (2^13)Page Size
- **TLB is Direct Mapped with 256 Entries**
- 8 KB (2^13) Direct Mapped L1 Cache
- Block Size = 64 Bytes (2^6 words)
- 4 MB Direct Mapped L2 Cache
- 2^22/2^6= 2^16

Microprocessor & Computer Architecture (μpCA)

6th Optimization : Optimized to Improve the Hit Time



- **Modified Specification for Cache**
- 64 bit Virtual Address
- 41 bit Physical Address
- 16 KB (2^14)Page Size
- **2 Way, 256 Entry TLB**
- 16 KB (2^14) Direct Mapped L1 Cache
- Block Size = 64 Bytes (2^6 words)
- 4 MB, 4 Way, L2 Cache
- 2^22/2^6x 2^2= 2^14

Microprocessor & Computer Architecture (μpCA)

Think About It?

Why Can’t we Virtually Index and Virtually TAG?

Reference

Microprocessor & Computer Architecture (μpCA)

Next Session

Input & Output devices

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135

Microprocessor & Computer Architecture (μpCA)

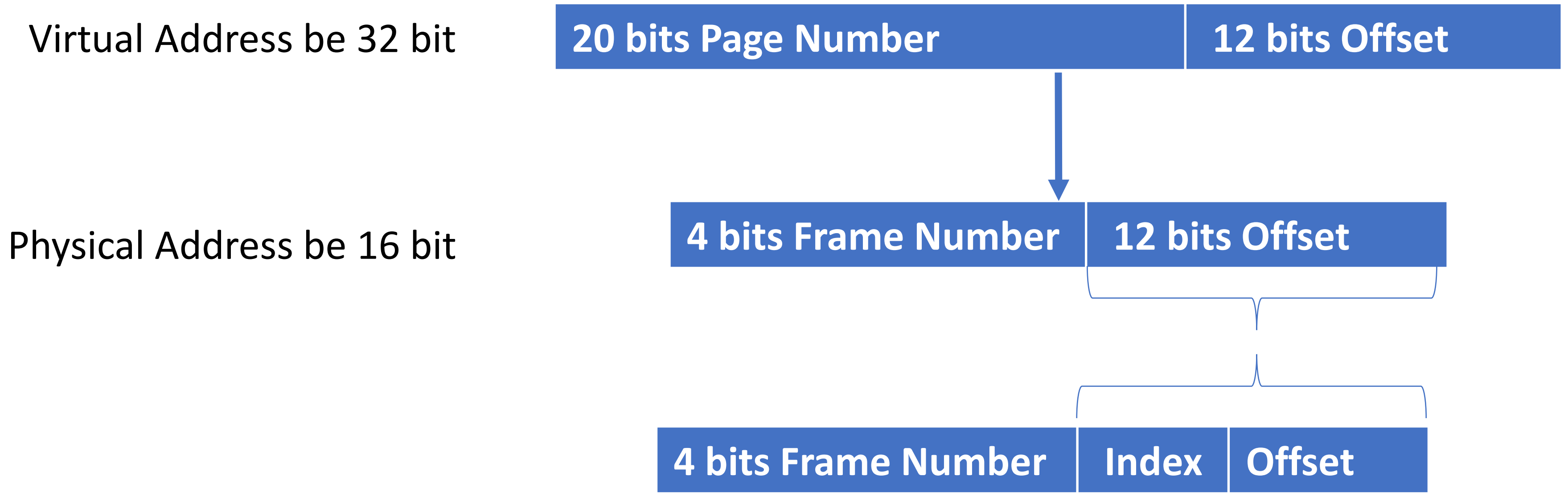
What is the Problem?

Accessing data from Cache involve:

- Indexing
- Tagging

Converting Virtual Address to Physical Address take some time.

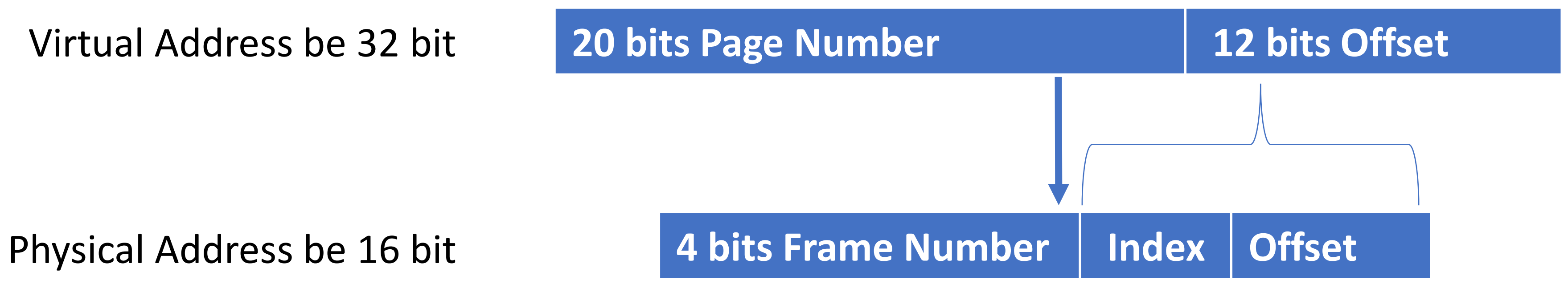
Solution ☺ Example:



Microprocessor & Computer Architecture (μpCA)

6th Optimization: Reduce Miss Time

Avoid Address Translation In Cache Indexing To Reduce Hit Time



Solution ☺

Do not wait for VA to PA translation.

Extract Index information from Virtual Address

Extract Tag Information from the Physical Address.

6th Optimization facilitate Virtual Indexing and Physically Tagging (VIPT)

Microprocessor & Computer Architecture (μpCA)

UE19CS252

V R BADRI PRASAD

Department of
Computer Science and Engineering

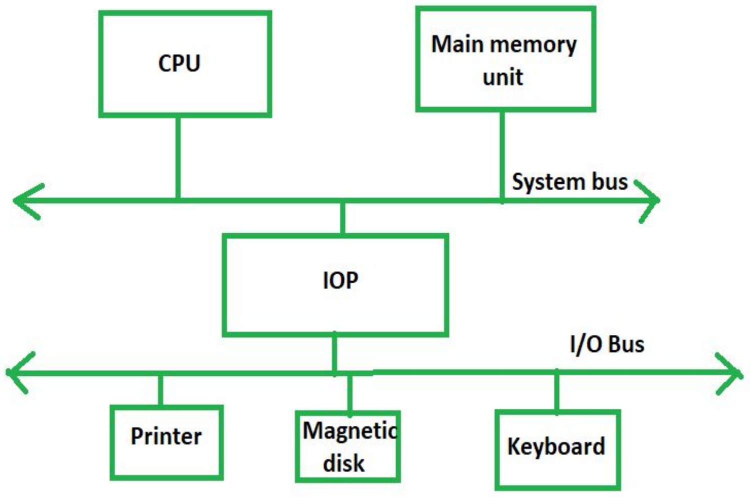
I/O and Bus Architecture

V R BADRI PRASAD

Department of
Computer Science and Engineering

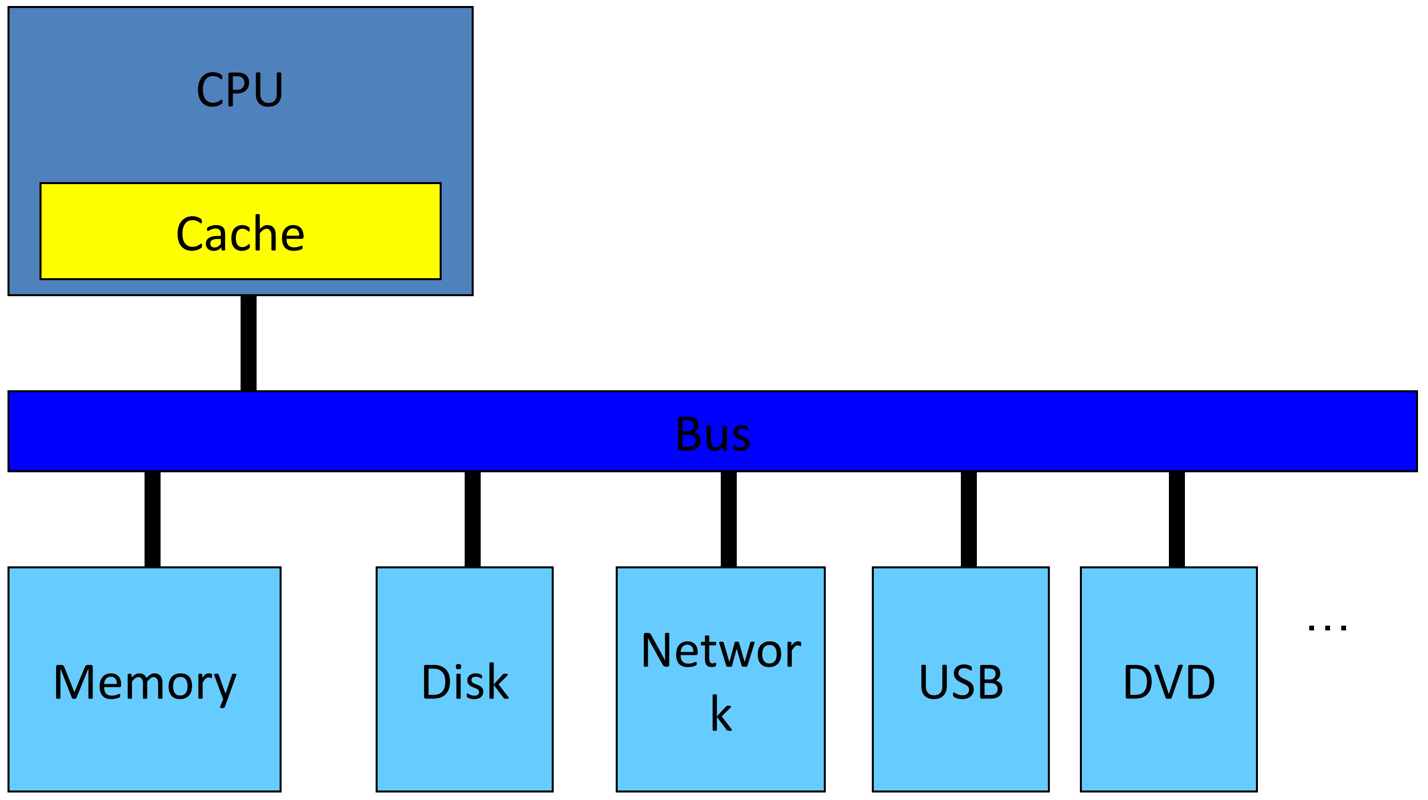
Accessing I/O Devices - Introduction

- The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface.
- A typical link between the processor , memory and several peripherals.
- These devices work at varying speeds.
  - Ex: Monitors, Mouse, Keyboards, Video cameras, etc.,
  - Data transfer rate can either be regular or irregular.

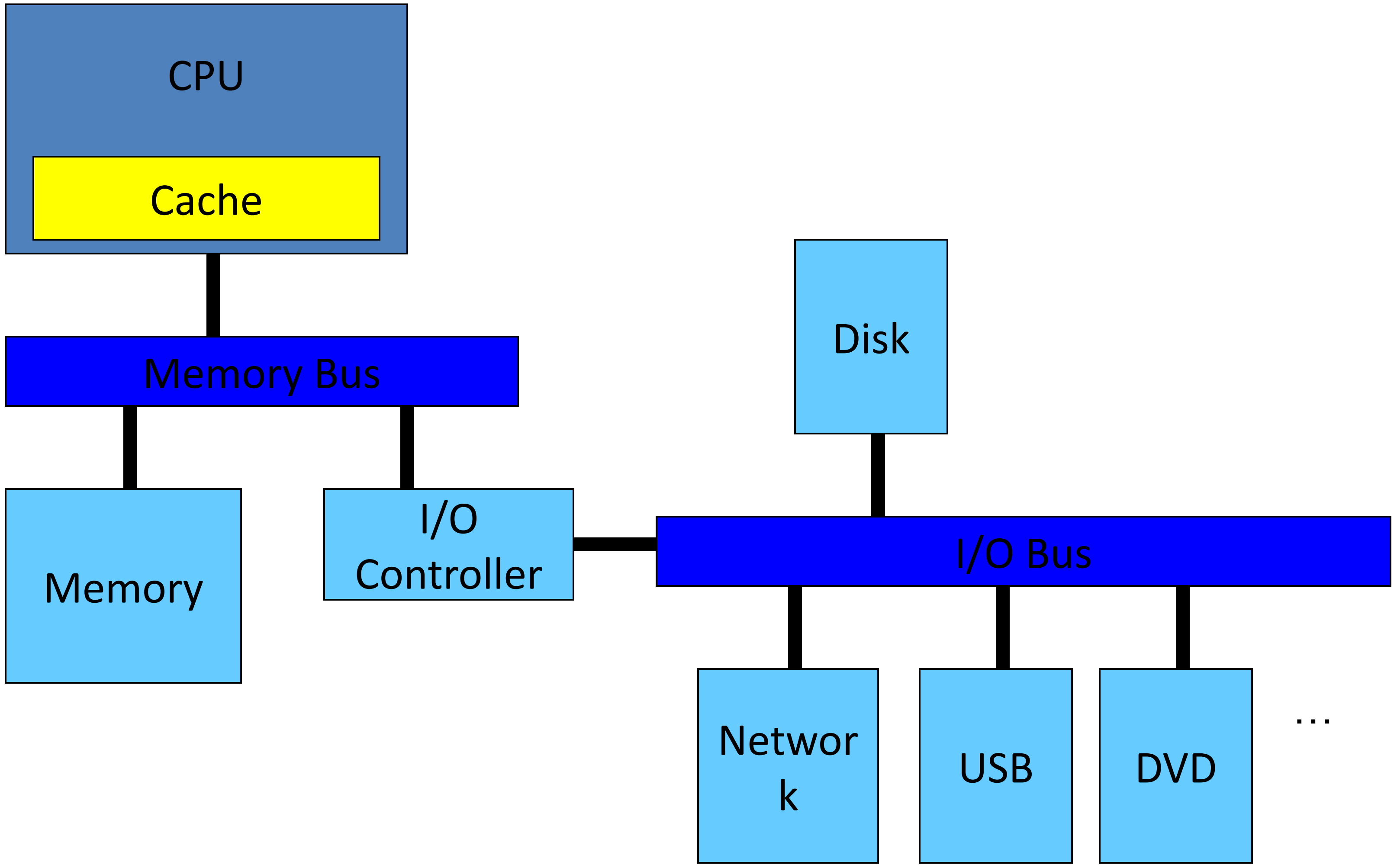


Input-Output Processor

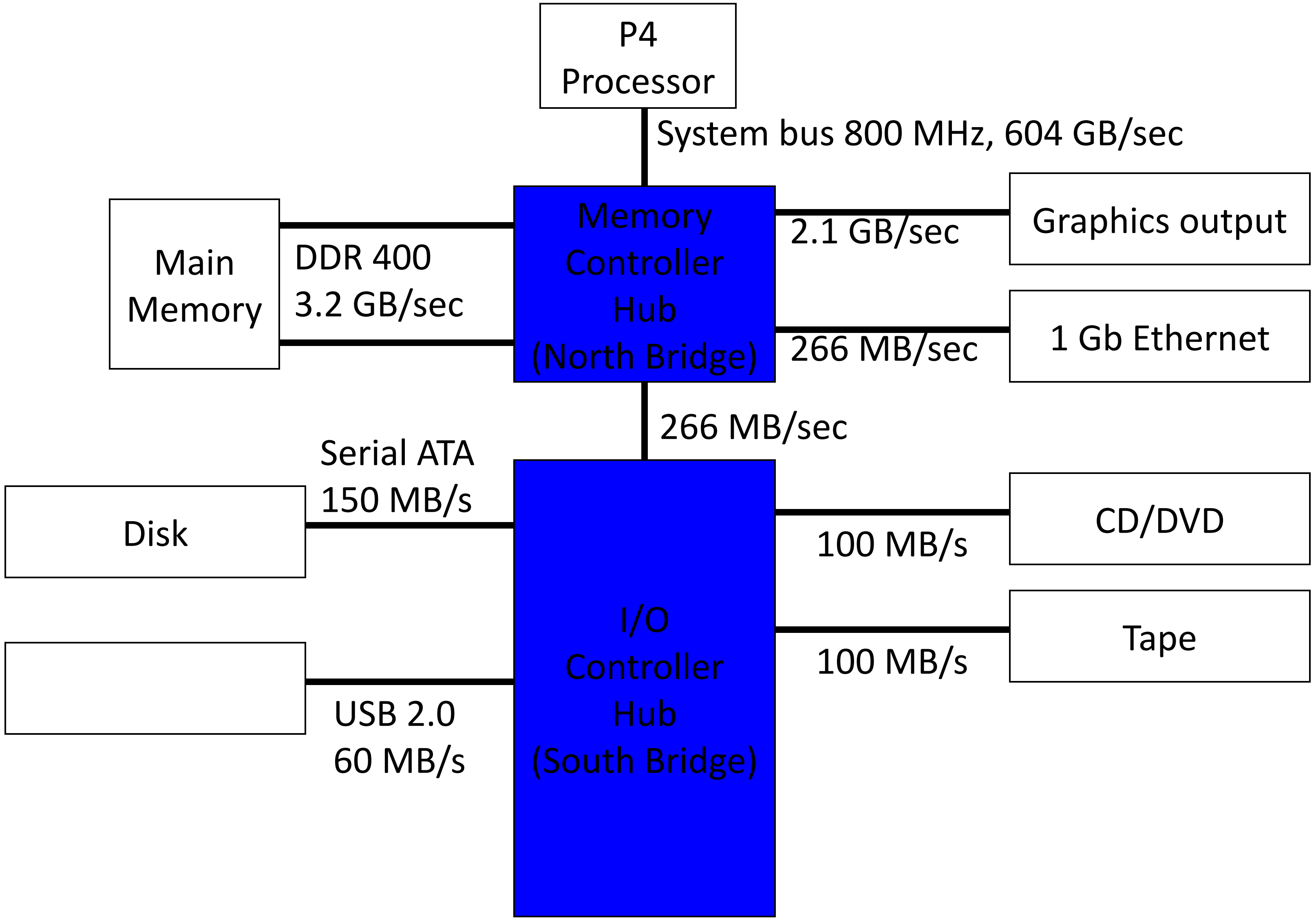
Input/Output



I/O Hierarchy

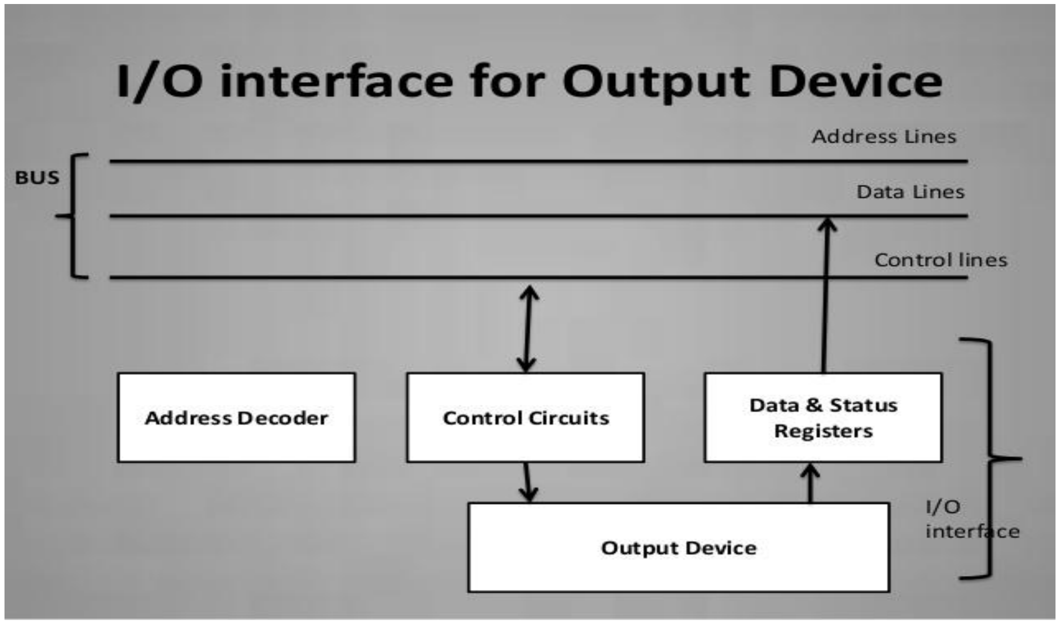


Intel Example



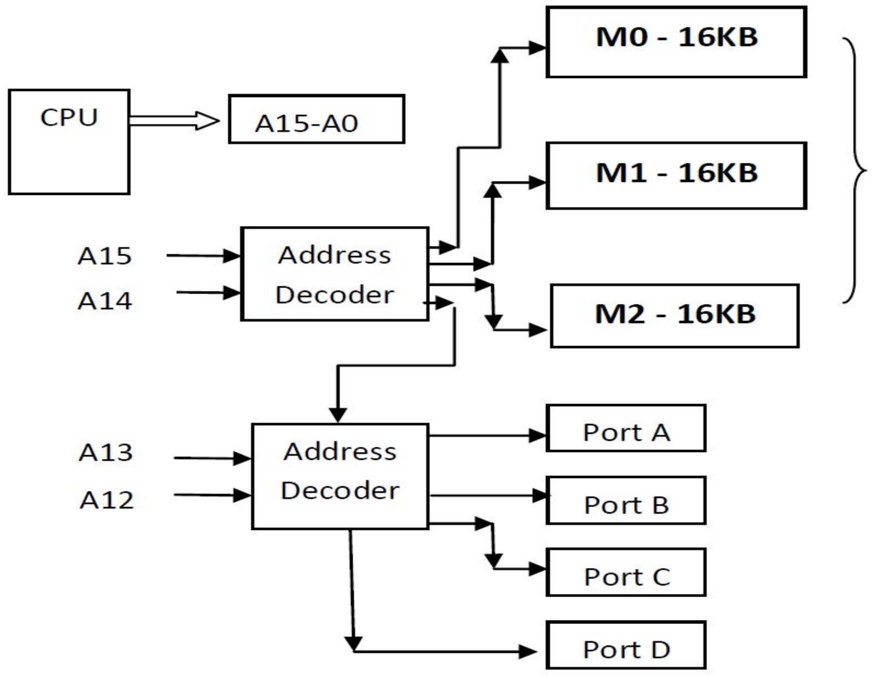
Accessing I/O Devices - Introduction

- There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.
- The I/O **Bus** consists of data lines, address lines and control lines.
- The I/O **bus** from the processor is attached to all peripherals interface.
- To communicate with a particular device, the processor places a device address on address lines.



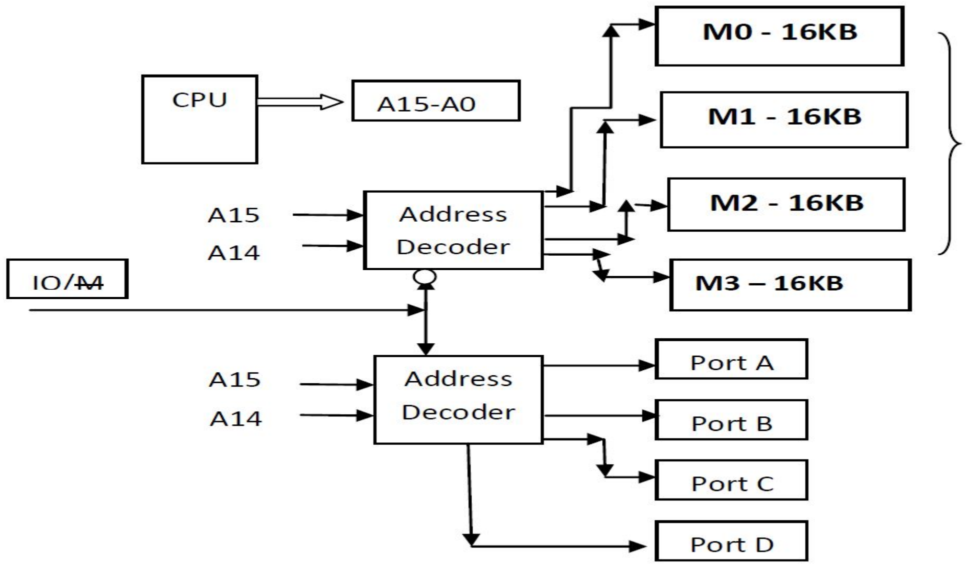
Accessing I/O Devices :
Memory Mapped device interface

- **Memory Mapped I/O device interface:**
  - Same address decoder selects memory and I/O ports.
  - Some memory address space is occupied by the I/O devices.
  - All data transfer instructions to / from memory an be used to transfer to/from I/O devices.
  - Processor need not have separate instructions for I/O.



Accessing I/O Devices :
I/O Mapped device interface

- **I/O Mapped I/O device interface:**
  - Separate instructions for I/O data transfer [IN / OUT].
  - Processor signal identifies whether a generated address refers to memory or I/O device.
  - Separate address decoder for selecting memory and I/O ports.
  - Complete memory address space can be utilized.



Accessing I/O Devices – Data Transfer Techniques

- Data transfer to and from the peripherals may be done in any of these ways:

  1. **Programmed I/O:** **CPU executes a program and that transfers data between I/O device and Memory**

     a. **Synchronous** **:** Fixed rate of transfer

     b. **Asynchronous :** Handshaking – polling for sending / receiving the data

     c. **Interrupt- Driven :(next slide)**

  2. Direct memory access(DMA): **An external controller directly transfers data between I/O device and Memory without CPU intervention.**

What is Interrupt/Exception?

- Main ()
- {
- :
- Doing something
- (e.g.
- browsing)
- :
- } ring

Can happen anytime
Depends on types of interrupts

Phone rings

Phone rings

```
_isr() // Interrupt service routine
{

  some tasks (e.g. answer
                    telephone)

}  //when finished,
  //goes back to main
```

Accessing I/O Devices – Data Transfer Techniques

c. **Interrupt- Driven:**

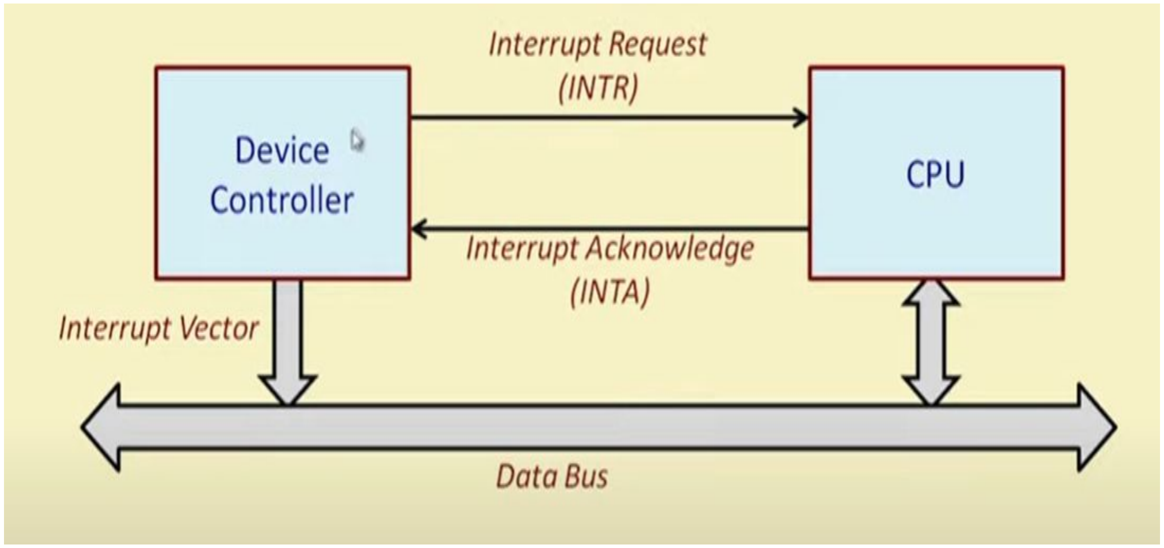
- CPU initiates the data transfer and proceeds to perform some other task.
- When I/O module is ready for data transfer, it informs the CPU by activating a signal (Interrupt request).
- The CPU suspends the task it was doing, services the request from the device and return back to the task it was doing.

- **Advantages:**
  - CPU time is not wasted by polling the device
  - CPU time is required only during the data transfer plus some overheads for transferring and returning the control.

Accessing I/O Devices – Data Transfer Techniques

c. What happens when an interrupt request arrives?

- At the end of the execution of the current instruction, PC and the status register contents are saved in the stack automatically.
- Interrupt is acknowledged, interrupt vector is obtained based which the control transfers to the appropriate ISR.
- After handling the interrupt, the ISR executes a special instruction *return from interrupt(RTI)*.



Accessing I/O Devices – Data Transfer Techniques

c. What happens when an interrupt request arrives?

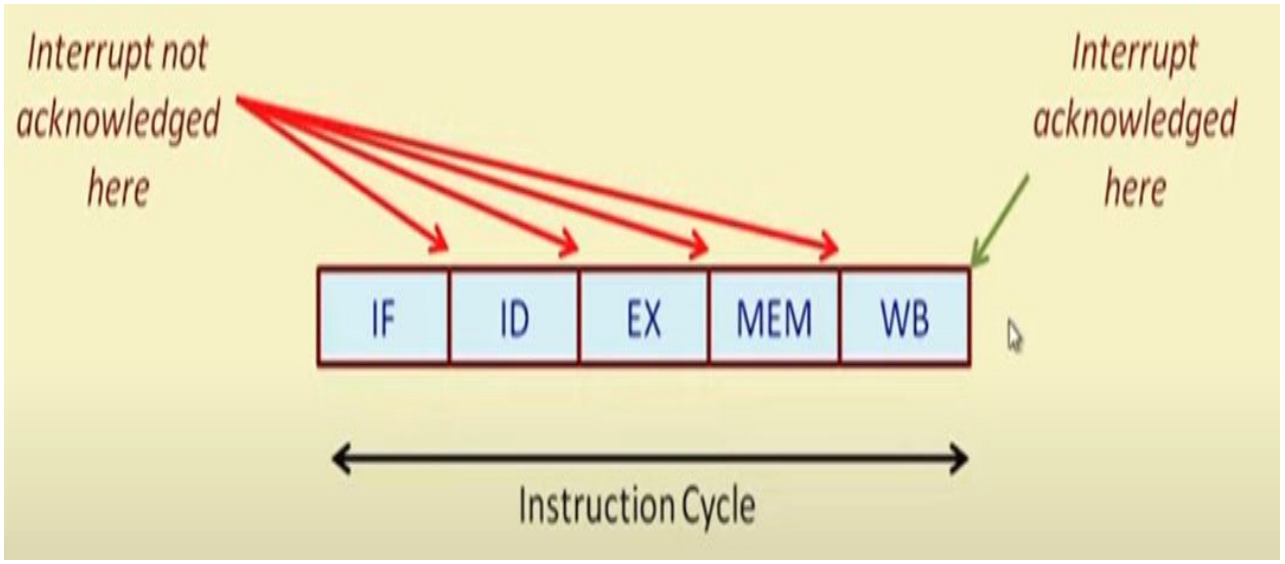
- At the end of the execution of the current instruction, PC and the status register contents are saved in the stack automatically.
- Interrupt is acknowledged, interrupt vector is obtained based which the control transfers to the appropriate ISR.
- After handling the interrupt, the ISR executes a special instruction *return from interrupt(RTI)*.

ARM Interrupt Vector Table

	Address	Exception	Mode in Entry
1	0x00000000	Reset	Supervisor
2	0x00000004	Undefined instruction	Undefined
3	0x00000008	Software Interrupt	Supervisor
4	0x0000000C	Abort (prefetch)	Abort
5	0x00000010	Abort (data)	Abort
x	0x00000014	Reserved	Reserved
6	0x00000018	IRQ (external interrupt)	IRQ
7	0x0000001C	FIQ (fast interrupt)	FIQ

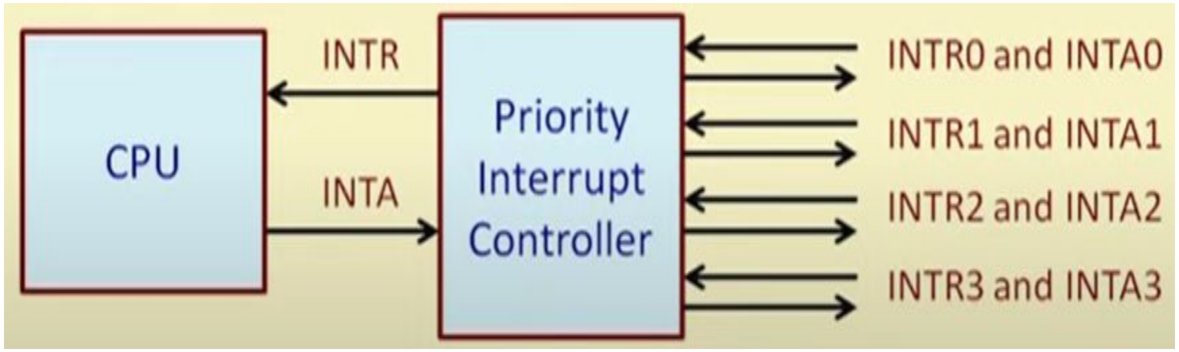
Accessing I/O Devices :

Interrupt request during the execution of an instruction.



Accessing I/O Devices :
Multiple devices interrupting CPU

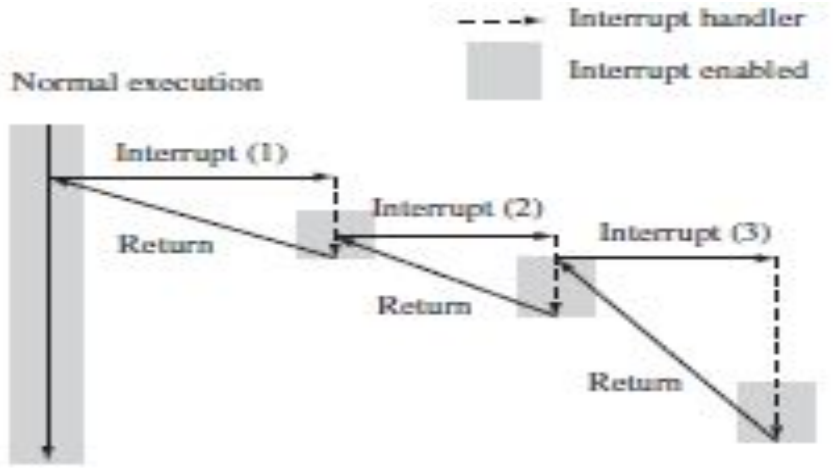
- Common solution is to use a **Priority Interrupt Controller.**
  - Interrupt controller interacts with CPU on one side and multiple devices on the other side.
  - For simultaneous interrupt requests, interrupt priority is defined.
  - Interrupt controller is responsible for sending the interrupt vector to the CPU.



Accessing I/O Devices :
How is interrupt nesting handled?

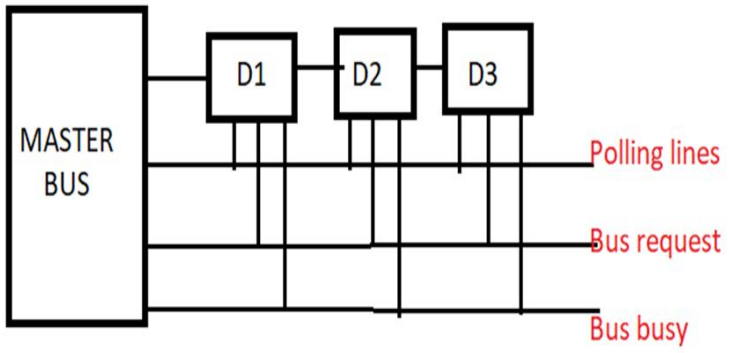
- Consider a scenario.
- A device D0 has interrupted and the CPU is servicing the (executing the ISR) device D0.
- In the meantime, device D1 has interrupted.
- Two possible scenarios are here:

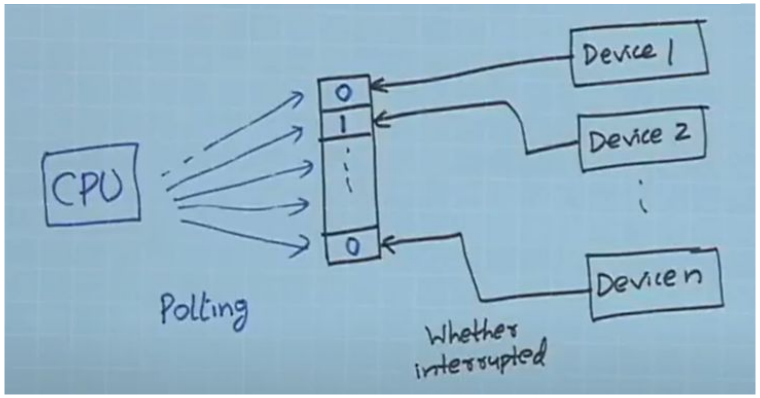
1. D1 will interrupt the ISR for D0, get processed first, and then the ISR for device D0 will be resumed.
   This creates problem for multi nesting.
2. Disable the interrupt system automatically whenever an interrupt is acknowledged.
   This will not require nested interrupts to be handled.



Accessing I/O Devices – Polling Technique

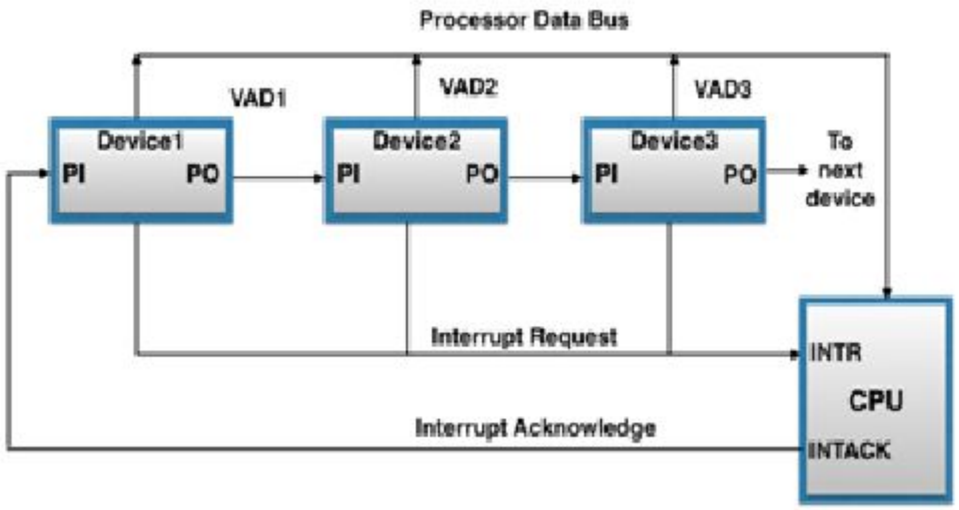
- Each device can have a status bit whether the device has interrupted?
- CPU can poll the status bit to check for the device which has interrupted.





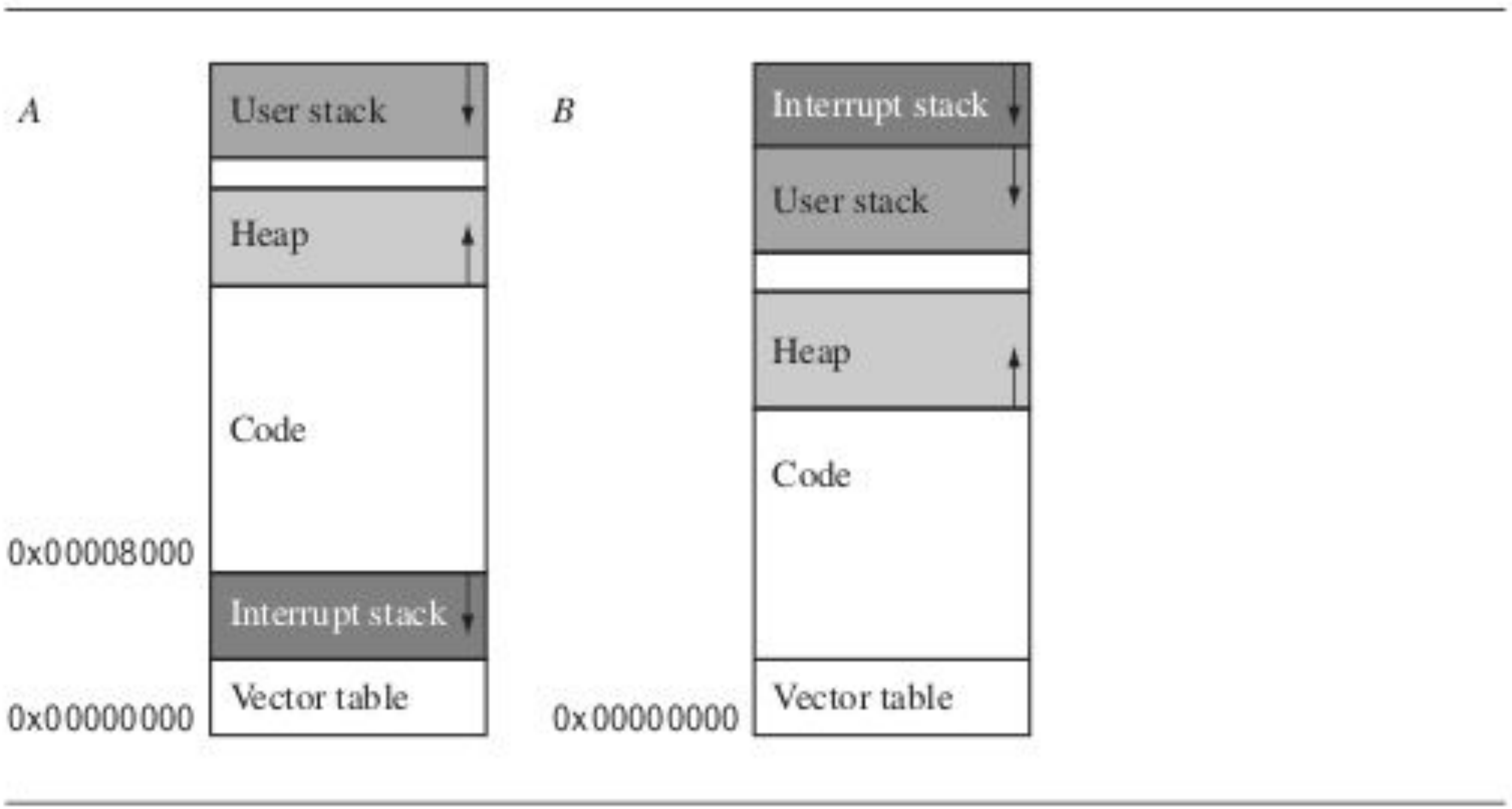
Accessing I/O Devices – Daisy Chain Technique

- In Daisy chain technique, INTR line is common to all the devices.
- INTA line is connected in a daisy chain fashion [as shown].
- This allows to propagate serially through the devices.
- A device when it receives INTA, passes the signal to the next device only if it has not interrupted.
- Else, it stops the propagation of INTA and puts the identifying code on the data bus.
- Thus the device that is electrically closest to the CPU will have the highest priority.



Basic Interrupt Stack Design and Implementation

錺 Stack Layouts



Basic Interrupt Stack Design and Implementation

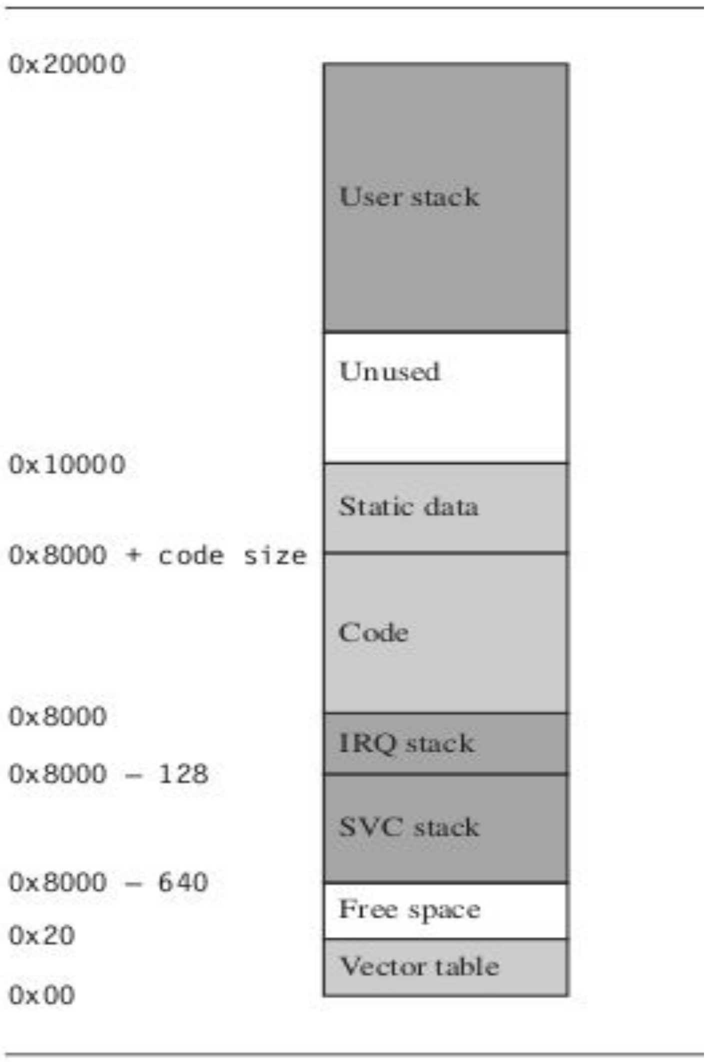
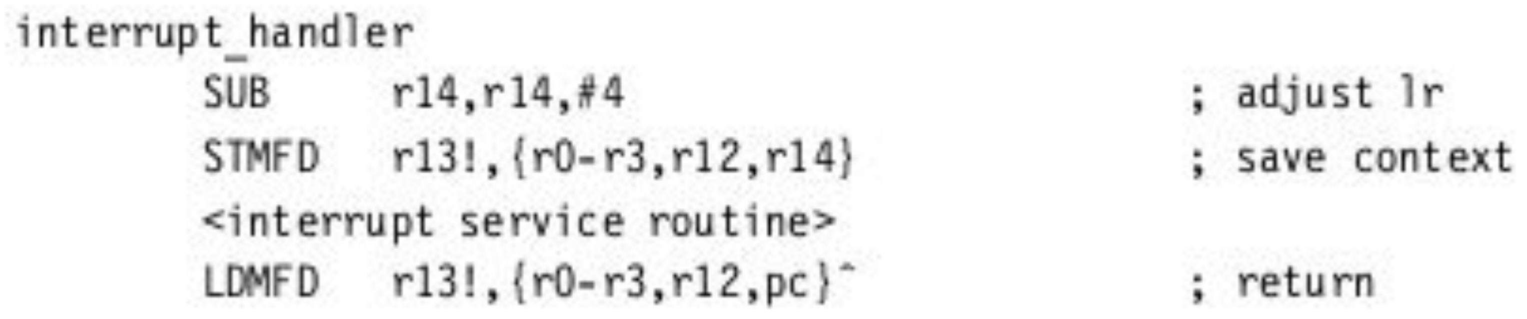


Figure 9.7 Example implementation using layout *A*.

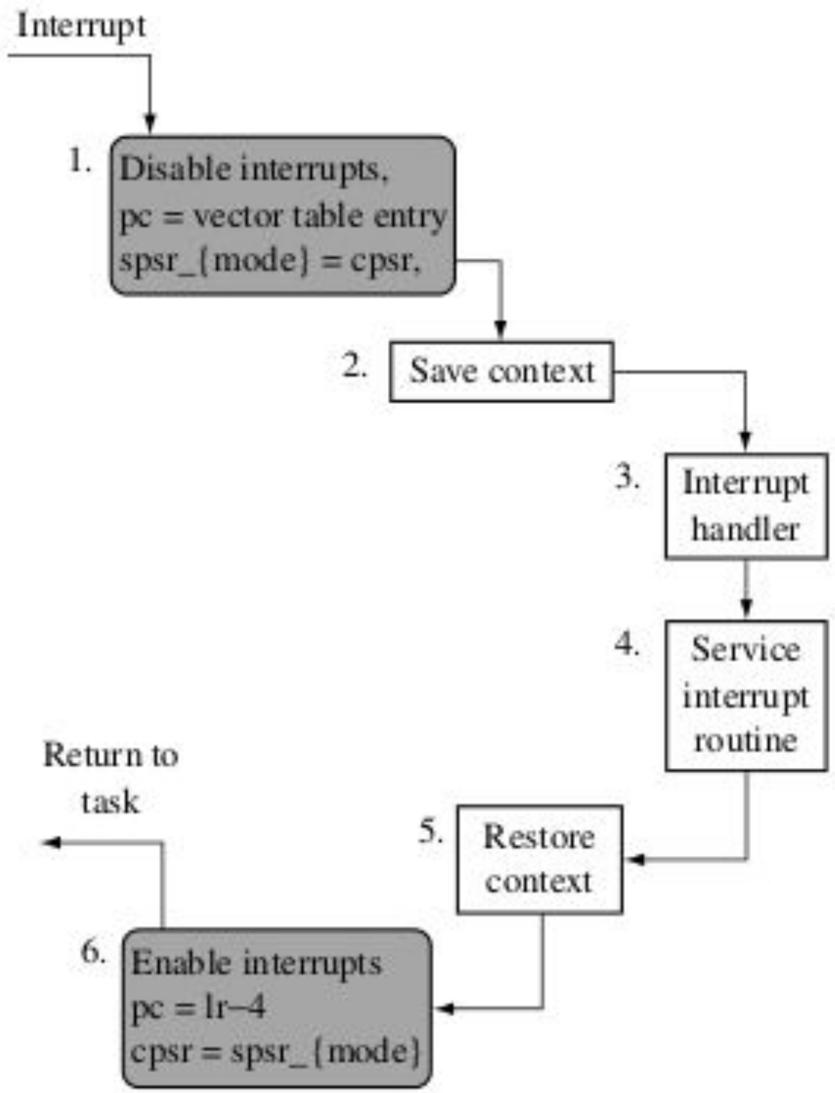
Interrupt Handling Schemes Non-Nested Interrupt Handler

- Various stages in a NNIH –
  - Enable Interrupts
    - spsr ← spsr_{interrupt request mode}
    - pc is restored



```
interrupt_handler
        SUB     r14,r14,#4                          ; adjust lr
        STMFD   r13!,{r0-r3,r12,r14}                ; save context
        <interrupt service routine>
        LDMFD   r13!,{r0-r3,r12,pc}^                ; return
```

Interrupt Handling Schemes Non-Nested Interrupt Handler



Microprocessor & Computer Architecture (μpCA)

Thank You

V R BADRI PRASAD

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

UE19CS252

V R BADRI PRASAD

Department of
Computer Science and Engineering

USB, PCI, SCSI, AMBA and ASB Bus Architecture

V R BADRI PRASAD

Department of
Computer Science and Engineering

Universal Serial Bus Architecture

- Universal Serial Bus is a data interface used with computers enabling the computer to send and receive data as well as providing power to some peripherals like disc drives, Flash memory sticks and the like so that separate power sources are not needed for each item

Universal Serial Bus Architecture

- USB Peripherals are slaves responding to commands from hosts
- When a peripheral is attached to the USB network, the host communicates with the device.
  - To learn its identity
  - To discover which device driver is required
    - This is called Enumeration
    - It is supported as the device driver for the USB port on the host.
- Power :
  - USB devices can pull limited amount power from the bus.
- Speed:
  - Low : 1.5Mbps
  - Full : 12Mbps
  - High : 480 Mbps

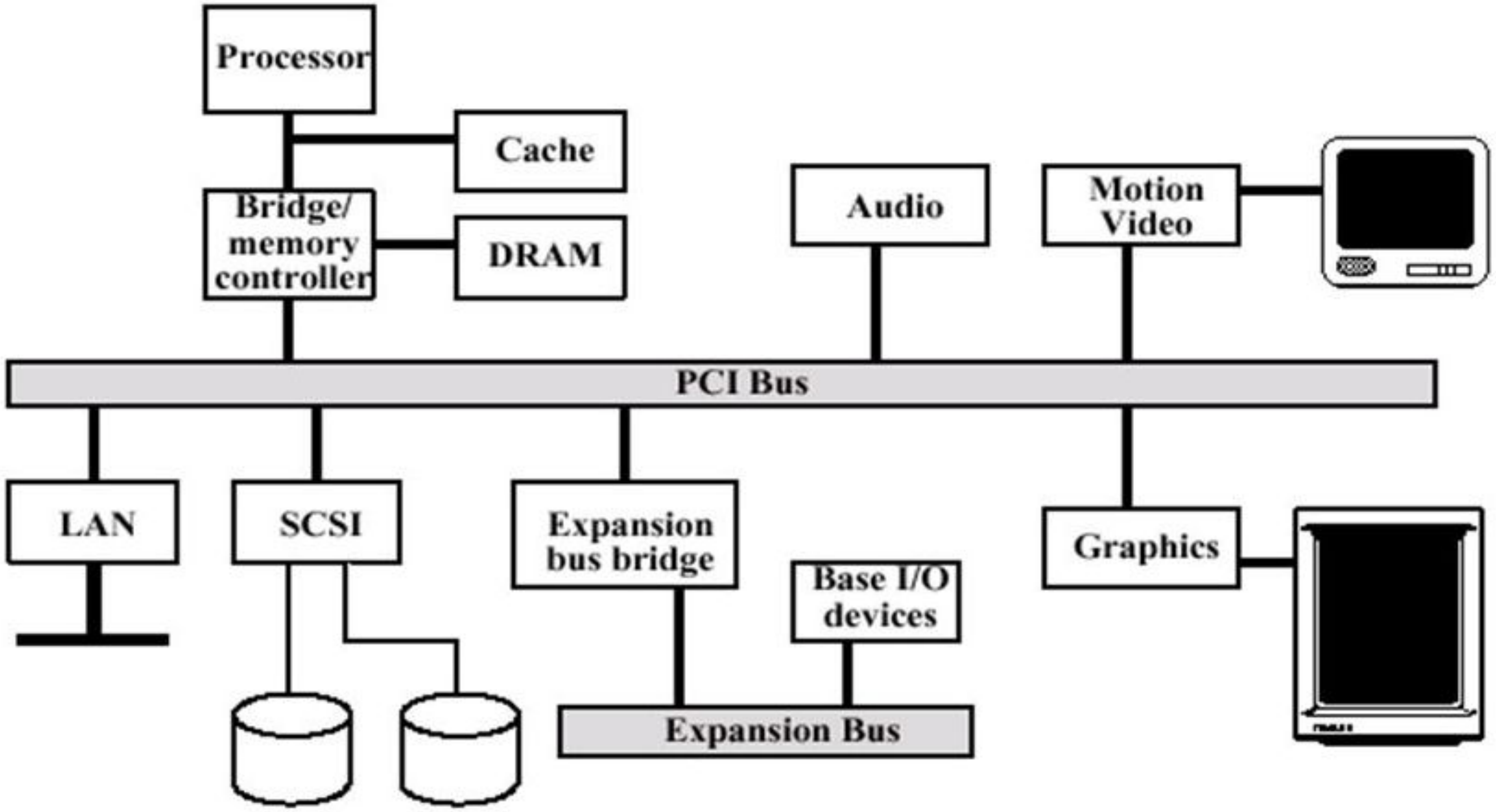
Universal Serial Bus Architecture

- USB Devices are of two types.
  - Stand-alone:
    - Single Function Units like Mouse,etc.,
  - Compound Devices:
    - More than one peripheral sharing a USB port.

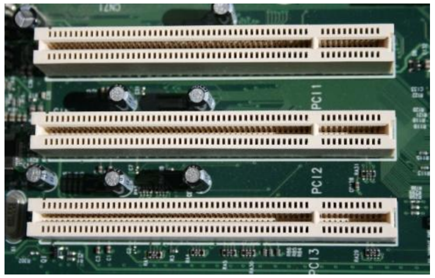
      Ex: Video Camera (both audio and video).
- USB Hubs:
  - Hubs are bridge
  - Themselves are USB devices
  - Hubs detect topology changes due to insertion and deletion of devices.

Peripheral Component Interconnect : PCI-Bus Architecture

- PCI is a local computer bus for attaching hardware devices in a computer.



Peripheral Component Interconnect : PCI-Bus Architecture



Features:

- **32 bit bus**
- **Transfer rate : 133MB/s**
- Any PCI device may initiate a transaction.
- First, it must request permission from a PCI bus arbiter on the motherboard.
- The arbiter grants permission to one of the requesting devices.
- The initiator begins the address phase by broadcasting a 32- bit address plus a 4-bit command code, then waits for a target to respond.
- All other devices examine this address and one of them responds a few cycles later.

SCSI -Bus Architecture

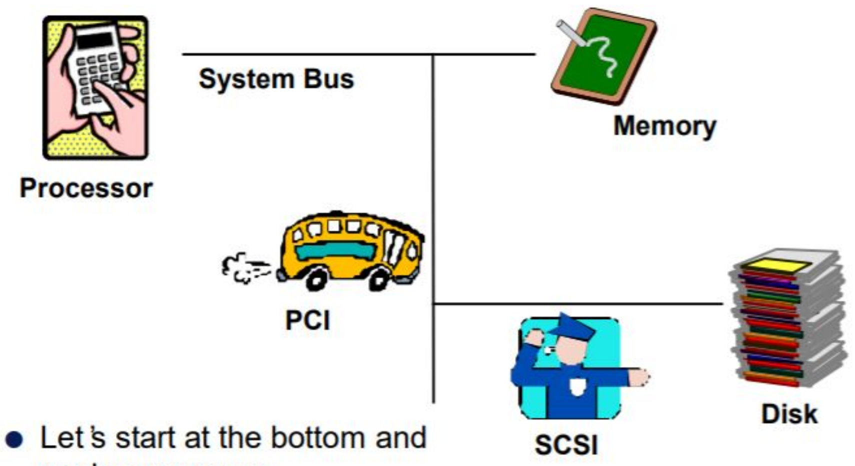
Small Computer System Interface – SCSI :

- Derived from **SASI –** Shugart Associates System Interface
- Provided as a bridge between hard disk low-level interface and a host computer
- Standard Interface and communication protocol for connecting computer peripherals.
- SCSI is used to increase performance and deliver faster
- Bus can address upto 8 devices (0 to 7).
- Provide larger expansion for devices such as CDs, scanners, etc.,
- Based on the Client Server Architecture
- Clients are Initiators – creates(begins)and sends SCSI commands to the target.
- Servers are Targets – Collection of logical units – carries out the requested task.
- Early SCSI assumes a bus based architecture

SCSI -Bus Architecture

Small Computer System Interface – SCSI :

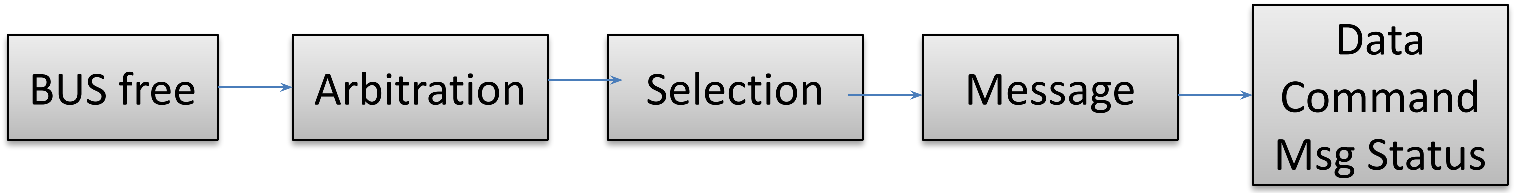
Parallel SCSI: SCSI-1 : 8 bit wide bus – speed 3.5MHz
SCSI-2 : 16 /32 bit wide – speed 10MHz to 20MHz
SCSI-3 : Also called as ultra SCSI.
Cable length- 3 mts.
Speed – 20MHz



- Let's start at the bottom and work our way up...

SCSI -Bus Architecture

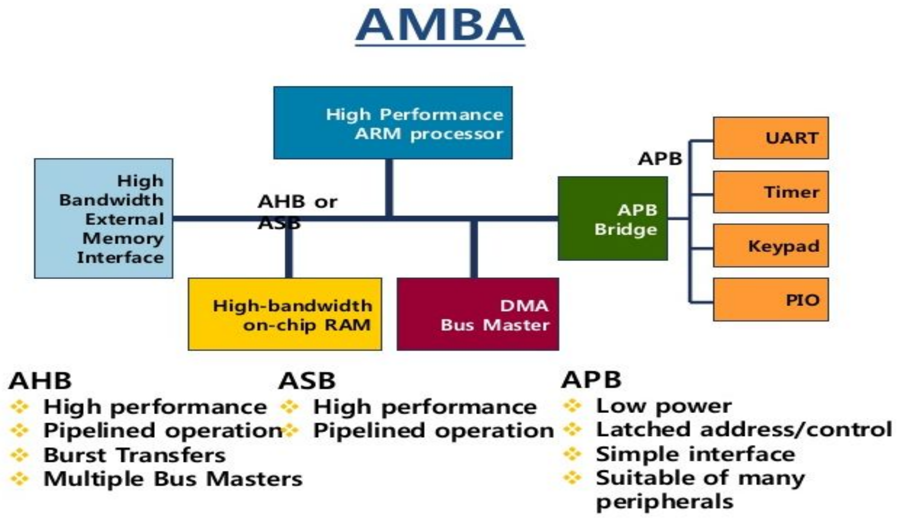
SCSI Bus Operation in phases:



- Bus Free
- Arbitration
- Selection / Reselection : target in control
  - Target decides when to recv msg, cmd, data from initiator.
  - OR send status to initiator
- Message
  - Used by target to send / recv protocol information.
    Eg : Identify, cmd complete, msg parity error, etc.,
- Data, command, message, status

AMBA (Advanced Microprocessor Bus Architecture) protocol

- The Arm AMBA protocols are an open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC).
- It facilitates right-first-time development of multi-processor designs with large numbers of controllers and peripherals.
- Features of AMBA Interfaces:
  - IP reuse is essential in reducing SoC development costs and timescales. AMBA specifications provide interface standards that enables IP reuse if the following essential requirements are met:
  - Flexibility:
  - Wide Adaption:
  - Compatibility:
  - Support:



APB (ADVANCED PERIPHERAL BUS) PROTOCOL

- The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) protocol family.
- It defines a low-cost interface that is optimized for minimal power consumption and reduced interface complexity.
- The APB protocol is not pipelined, use it to connect to low-bandwidth peripherals that do not require the high performance of the AXI protocol.
- The APB protocol relates a signal transition to the rising edge of the clock, to simplify the integration of APB peripherals into any design flow.
- Every transfer takes at least two cycles.
- The APB can interface with:
- • AMBA Advanced High-performance Bus (AHB)
- • AMBA Advanced High-performance Bus Lite (AHB-Lite)
- • AMBA Advanced Extensible Interface (AXI)
- • AMBA Advanced Extensible Interface Lite (AXI4-Lite)

Microprocessor & Computer Architecture (μpCA)

Thank You

V R BADRI PRASAD

Department of
Computer Science and Engineering