Cache Memory

Technology, Organization and Architecture

Computer System Architecture (CS5202)

IIT Tirupati

March 2020

Jaynarayan t tudu

jtt@iittp.ac.in

Last Lecture

- Trends in Computer Architecture
- Fundamentals of Computer System Design
- Performance Measurement: Bench mark and Metrics
- Analysis of Instructions
  - Addressing Modes
  - Example ISA: RISC and CISC types

Memory Design

We will be discussing following topics with respect to memory design. The basic idea starts from storing a single bit of information to a set of bits and be able to locate them and access them for some purpose.

- Moore’s Law and memory wall (identifying the problem)
- Memory hierarchy system (why we need it)
- Memory cell design (technology to store a bit)
- Introductory concepts of memory system design

Review of Data Representation

- This is one of the fundamental ideas in computer science which enables to represent the objects of physical world into the computer. So, we may think of computer as another world where each of the objects are identified with different rules. Data representation is one of the techniques which maps the objective in physical world into the objects of computational world.
- Generally the physical world is represented with a set of Characters and Numbers. The characters could be from any natural language scripts such as English, Devanagari etc. There are many possible ways in which the character and numbers are represented in computers.
- Considering advantage and disadvantages with respect to size of bits and computational complexity there are various coding scheme for specific purposes. Generally, for human interaction ASCII and Universal code (Unicode) systems are used, whereas, for computation binary representation(2's complement and floating-point) are used.

ASCII (Example)

- Uses 8 bits to represent A-Z, a-z, 0-9, and special characters/symbols

- Used for I/O interface; keyboard and display

Review of Data Representation

- This is one of the fundamental ideas in computer science which enables to represent the objects of physical world into the computer. So, we may think of computer as another world where each of the objects are identified with different rules. Data representation is one of the techniques which maps the objective in physical world into the objects of computational world.
- Generally the physical world is represented with a set of Characters and Numbers. The characters could be from any natural language scripts such as English, Devanagari etc. There are many possible ways in which the character and numbers are represented in computers.
- Considering advantage and disadvantages with respect to size of bits and computational complexity there are various coding scheme for specific purposes. Generally, for human interaction ASCII and Universal code (Unicode) systems are used, whereas, for computation binary representation(2's complement and floating-point) are used.

Unicode (Universal Code)

- Variable coding scheme
- Used to code all the scripts on this earth
- Used primarily for HTML pages

Review of Data Representation

Data Representation for Computation

Binary: Integer

- 1’s and 2’s complement coding system
- 2’s complement is currently being used
- Different coding scheme are possible, could be used for specific purpose

Floating point:

- 32 bit standard or single precision
- 64 bit standard or double precision
- (smaller representation using 8 or 16 bit is also possible)

Memory System Design

Requirements:

- Able to store a set of bits
- Able to locate
- Able to retrieve as needed

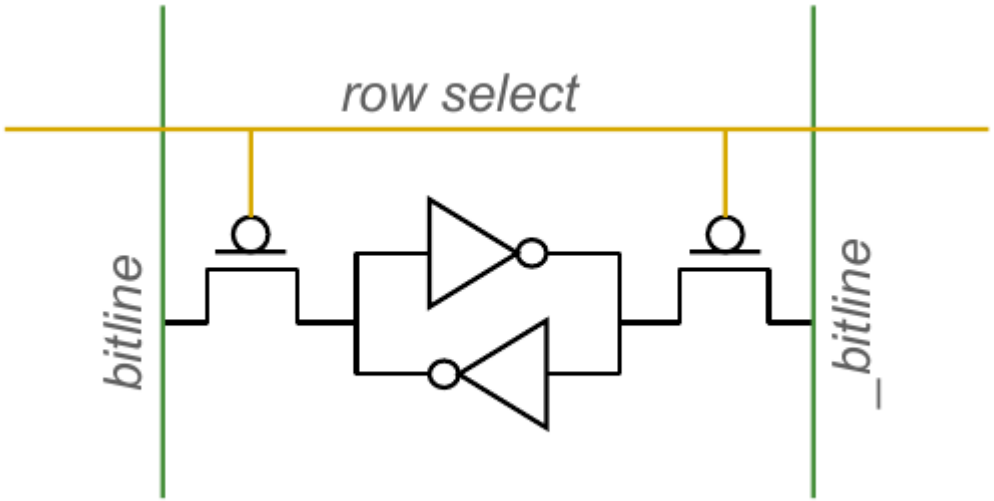
How to store a bit?

- Technology: CMOS transistor, capacitor, Magnetic surface, and (currently under research: phase and resistor)

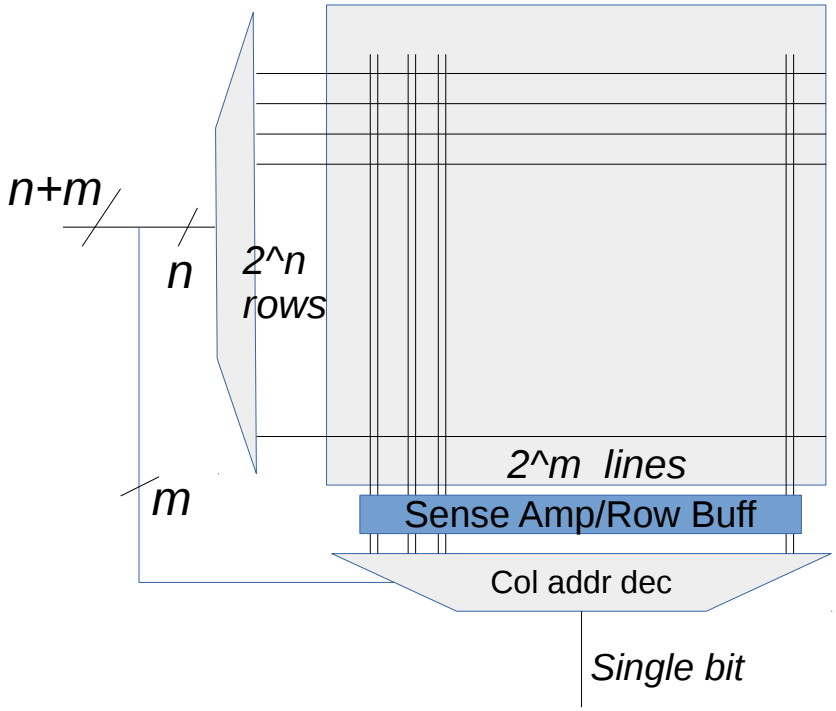
- Design: Static random access memory (SRAM)
  Dynamic random access memory (DRAM)
  (For more refer: ITRS 2015 report on recent memory design)

Memory Cell: storing a bit

SRAM Cell (6 Transistors cell)



SRAM Memory Array



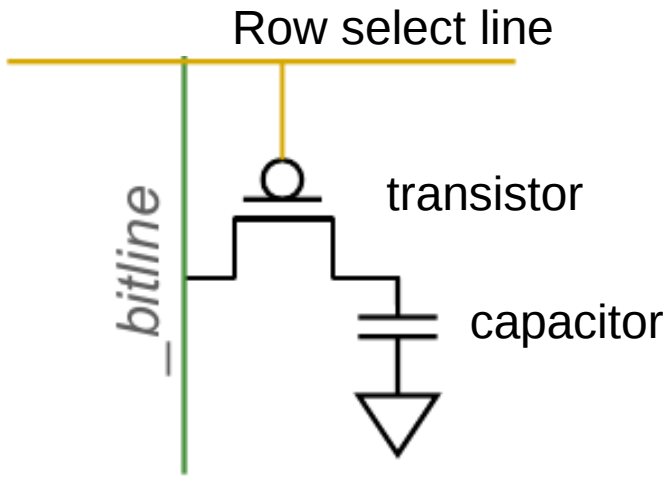
1) Address decode
2) Drive row select
3) Selected bit-cells drive bitlines
4) Sense amplifier senses the bit difference
5) Column address decode and select
6) Pre-charge all the bitlines for next read/write

The same structure can be replicated for simultaneous read or write of multiple data

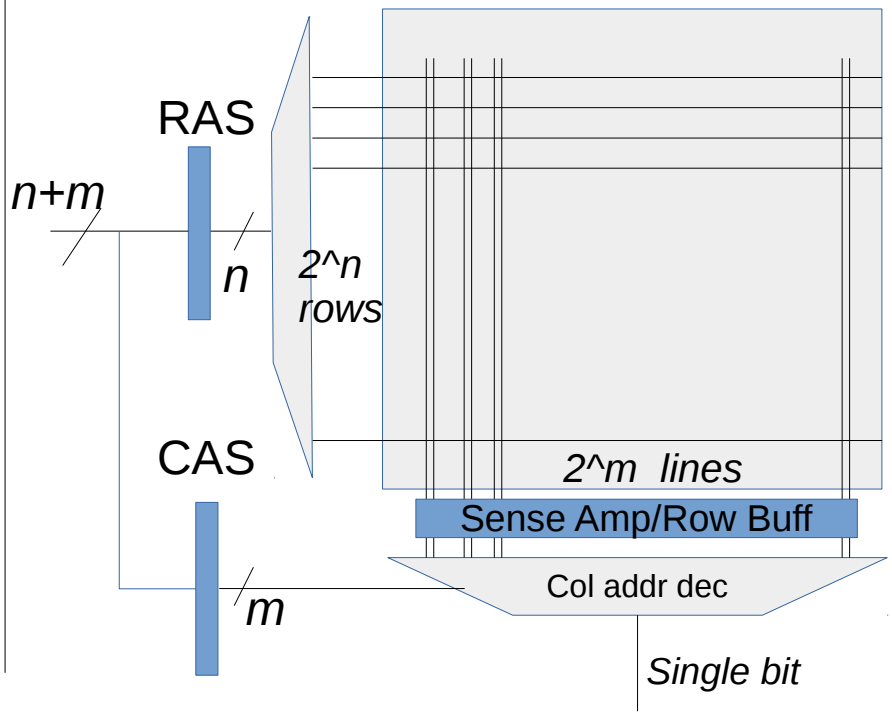
Step 2 and 3 dominates the access time,
Step 2, 3 and 5 dominates the cycle time

Memory Cell: storing a bit

DRAM Cell (1 transistor + 1 capacitor)
[Dynamic RAM]



DRAM Memory Array



1) Row address decode from RAS
2) Drive row select
3) Selected bit-cells drive bitlines
4) Sense amplifier senses the bit difference
5) Column address decode from CAS and select
6) Pre-charge all the bitlines for next read/write

RAS: Row Address Strobe
CAS: Column Address Strobe

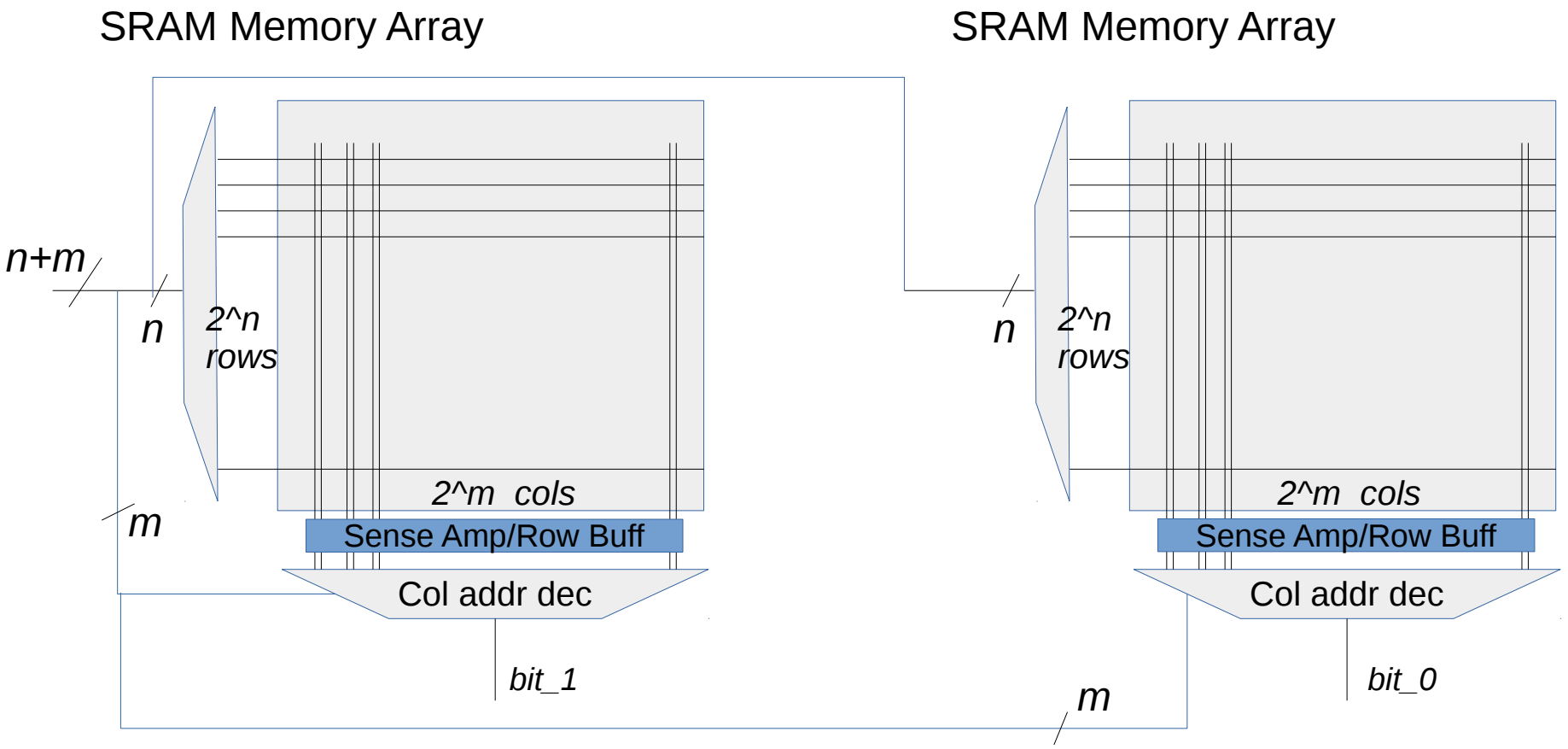
The same structure can be replicated for simultaneous read or write of multiple data

Refresh: Needed to restore the charge stored in capacitor.
This has to be done peridically (typically 10s of ms)

Detail in Main Memory Lecture

SRAM Memory Array

How to access more than just one bit simultaneously?

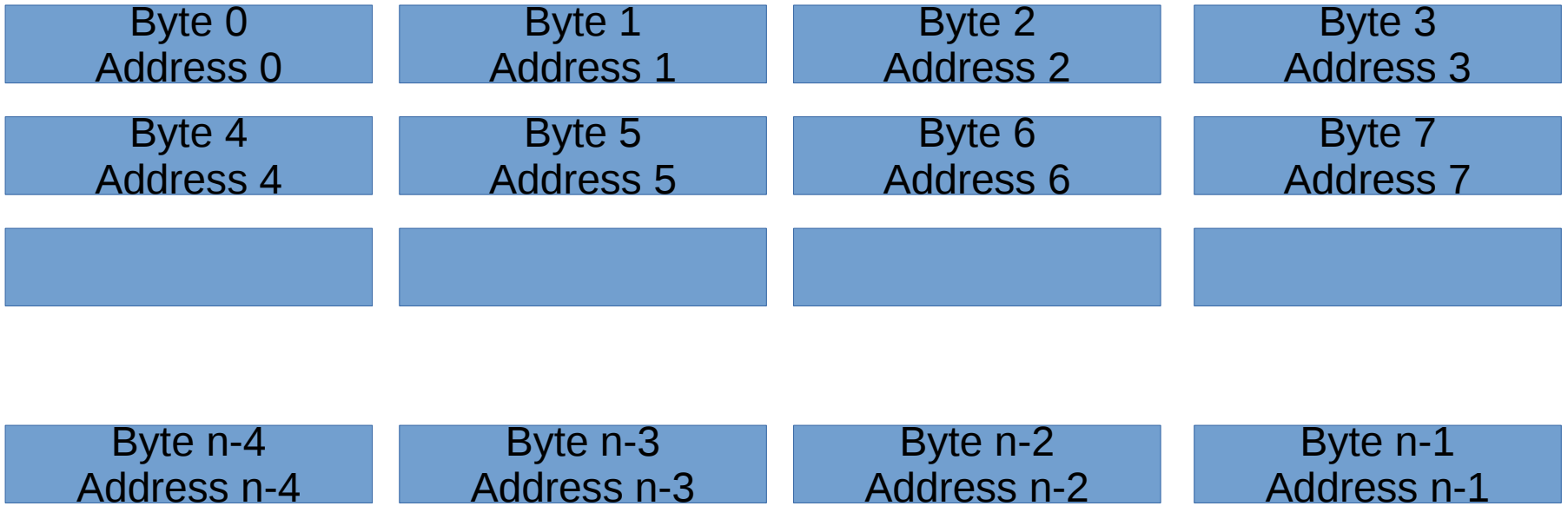


- *Above design to access 2 bit simultaneously.*
- *Similarly for 8 bit access, 8 such memory array can be replicated.*

Byte Address-ability

- The row and column decides the address of a bit (or a group of bits)
- Address can be assigned to single bit, to two bits, or to 8 bits or to any number of bits.
- Byte address-ability means that an 8 bits (1 byte) data is having unique address
- Byte addressable memory can be designed by replicating 8 memory arrays

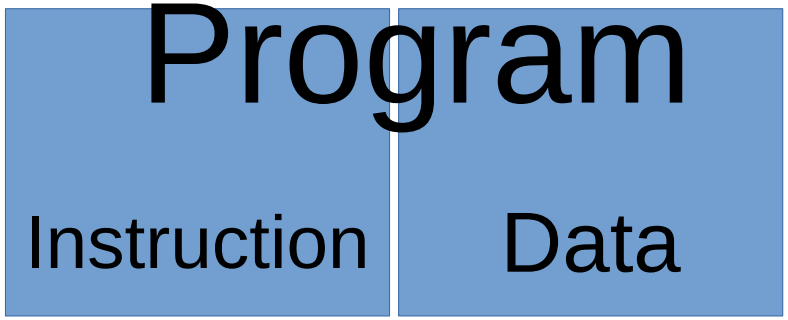
Block Diagram: (Logical Arrangement of Bytes – this is little endian)



Notes: Each of these bytes could go to different devices in physical memory array, For example: interleaving way of arrangement

Storing and Accessing Program

Program: is defined as a well define set of instructions
Instruction: is defined as an operation to be performed on operands



As discussed during ISA:

Instruction and Data both have to have some sort of format.

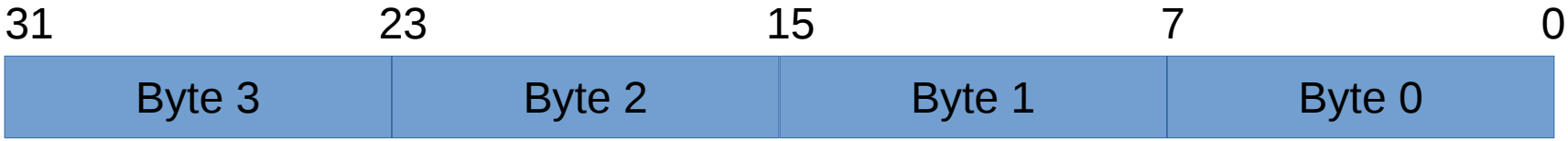
Data can be stored in many places (register, immediate, I/O etc.),
here we will examine the memory that store data.

Example:
Instruction: can be of fixed format (RISC type) or variable (CISC type) or hybridized.
Data: can be 1 byte (8 bits), 1 word (could be 16 or 32 bits),
double word etc (this is design dependent)

So, the question is: How to store instructions and data?

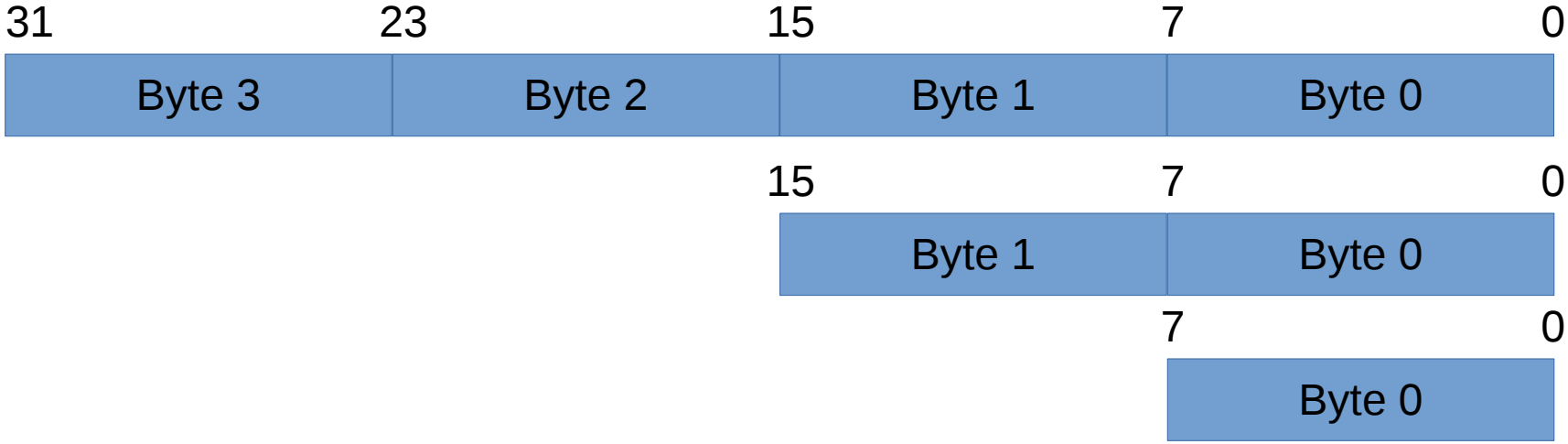
Storing and Accessing Program

Example of Fixed instruction format of 32 bit:



Some architecture define this as one word
(MIPS and RISC), x86_64 has 16 bit word size

Similarly for Data (could 8 bit, 16 bit, 32 bit or 64 bit):
(why did most of the architect chose the data size to be power of 2?)



Concept of Memory Block

Recall the issues of Memory Alignment!

	Value of 3 low-order bits of byte address							
Width of object	0	1	2	3	4	5	6	7
1 byte (byte)	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned
2 bytes (half word)	Aligned		Aligned		Aligned		Aligned	
2 bytes (half word)		Misaligned		Misaligned		Misaligned		Misaligned
4 bytes (word)	Aligned				Aligned			
4 bytes (word)		Misaligned				Misaligned		
4 bytes (word)			Misaligned				Misaligned	
4 bytes (word)				Misaligned				Misaligned
8 bytes (double word)	Aligned							
8 bytes (double word)		Misaligned						
8 bytes (double word)			Misaligned					
8 bytes (double word)				Misaligned				
8 bytes (double word)					Misaligned			
8 bytes (double word)						Misaligned		
8 bytes (double word)							Misaligned	
8 bytes (double word)								Misaligned

You have to pay penalty of memory access if it is miss-aligned (crossing the word boundary)!

Concept of Memory Block

The idea is whenever you access the memory, don't just access a byte or a word rather do it for group of words! Will it be helpful anyway?

The program analysis show that, the program exhibits a property called locality. Whenever an Instruction is being accessed very soon the next instruction also will be accessed. And, whenever an instruction is being accessed very likely that same instruction be accessed again.

Example:

This instruction is likely to be repeated soon: **Temporal Locality**

```
loop:lw   r1,0(r1)
     and  r1,r1,r2
     lw   r1,0(r1)
     lw   r1,0(r1)
     beq  r1,r0,loop
```

Each of these will be executed one after another **(Spatial Locality)**

Therefore, this can be a memory block.

Concept of Memory Block

Principle of Locality:

Locality in general could have many reasons, the idea is how does any two instructions or data being accessed in very short duration of time (short is tricky here).

- Temporal locality
  (this tells you keep the same thing for longer time)

- Spatial locality
  (this tells you keep two or more things together)

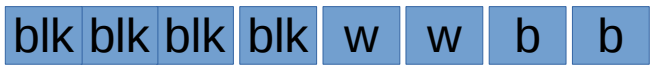
- Algorithm locality
  (Two separately located objects are being accessed in pattern.
  This tells you to keep two non consecutive blocks in cache)

Concept of Memory Block

It is good, therefore, to group a multiple words to form a memory block.

Byte 3	Byte 2	Byte 1	Byte 0	Word 0	BLOCK 0
Byte 3	Byte 2	Byte 1	Byte 0	Word 1	
Byte 3	Byte 2	Byte 1	Byte 0	Word 2	
Byte 3	Byte 2	Byte 1	Byte 0	Word 3	
Byte 3	Byte 2	Byte 1	Byte 0	Word 0	BLOCK 1
Byte 3	Byte 2	Byte 1	Byte 0	Word 1	
Byte 3	Byte 2	Byte 1	Byte 0	Word 2	
Byte 3	Byte 2	Byte 1	Byte 0	Word 3	

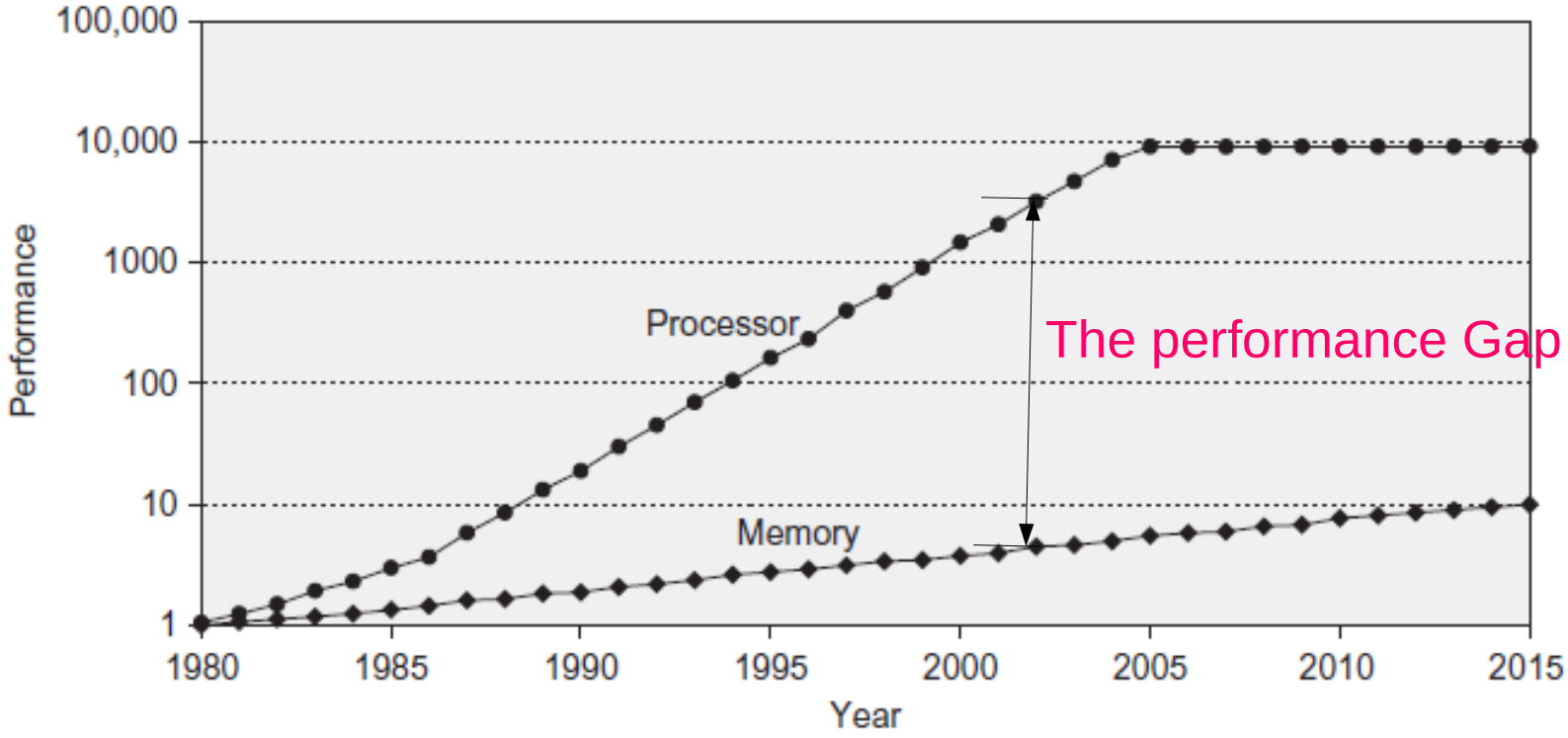
Addressing:



b – byte
w – word
blk – Block

The Cache Memory

How to solve the problem of performance gap?

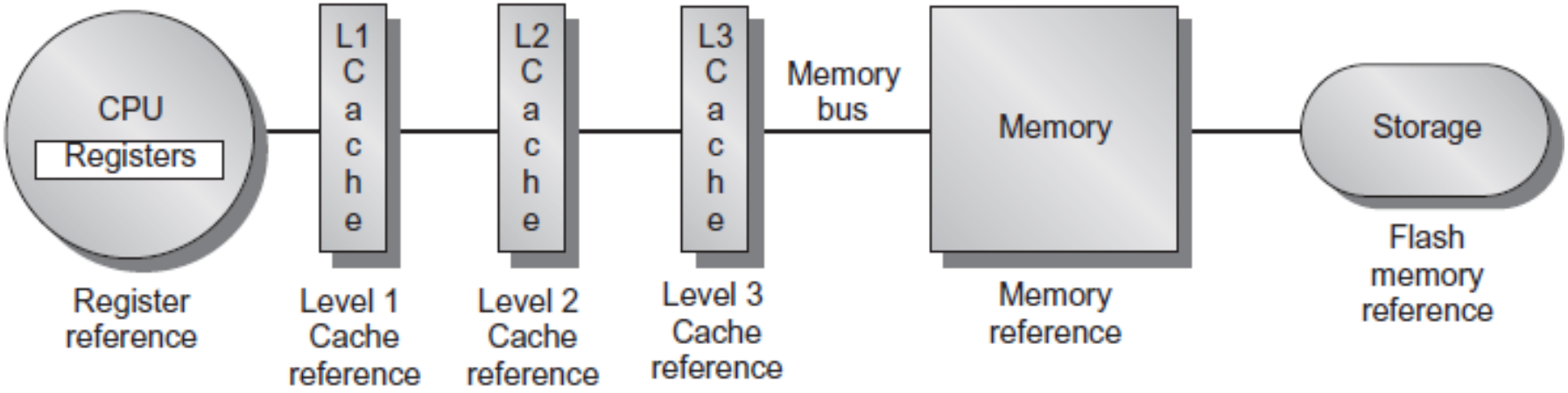


The gap is in the order of 1000!

One solution is memory hierarchy!

The Cache Memory

Memory hierarchy system of Personal Computers



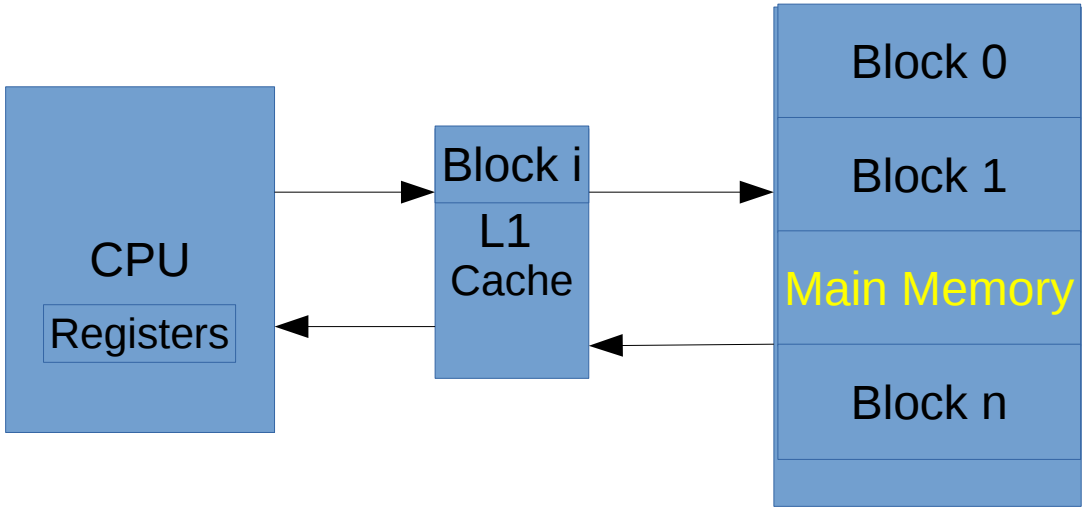
Laptop	Size:	1000 bytes	64 KB	256 KB	4-8 MB	4–16 GB	256 GB-1 TB
	Speed:	300 ps	1 ns	3–10 ns	10–20 ns	50–100 ns	50-100 uS
Desktop	Size:	2000 bytes	64 KB	256 KB	8-32 MB	8–64 GB	256 GB-2 TB
	Speed:	300 ps	1 ns	3–10 ns	10–20 ns	50–100 ns	50-100 uS

Cost per bit Reduces, size and density increases

Performance increases

The Cache Memory

Consider the simplified hierarchy system.



Fixed block size
= *n* number of words

- The goal is to bring the blocks from main memory and place them in cache!

- The block size in main memory and cache memory need to be equal.
  (Can the block size of main memory and cache be different, if so how?)

- CPU finally needs a word, it does not process a block at a time!

The Cache Memory

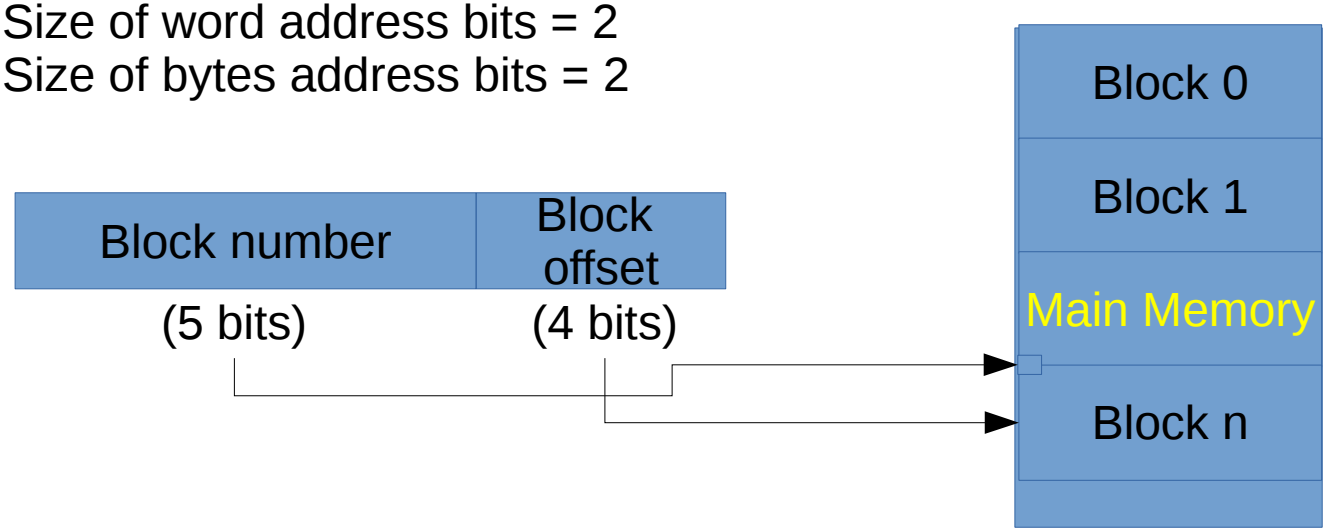
Addressing in Block:

Example:

Given a memory (does not matter cache or main) of size 512 bytes
Block size = 4 words
Word size = 4 bytes
How many blocks are possible? And what would be minimum address size?

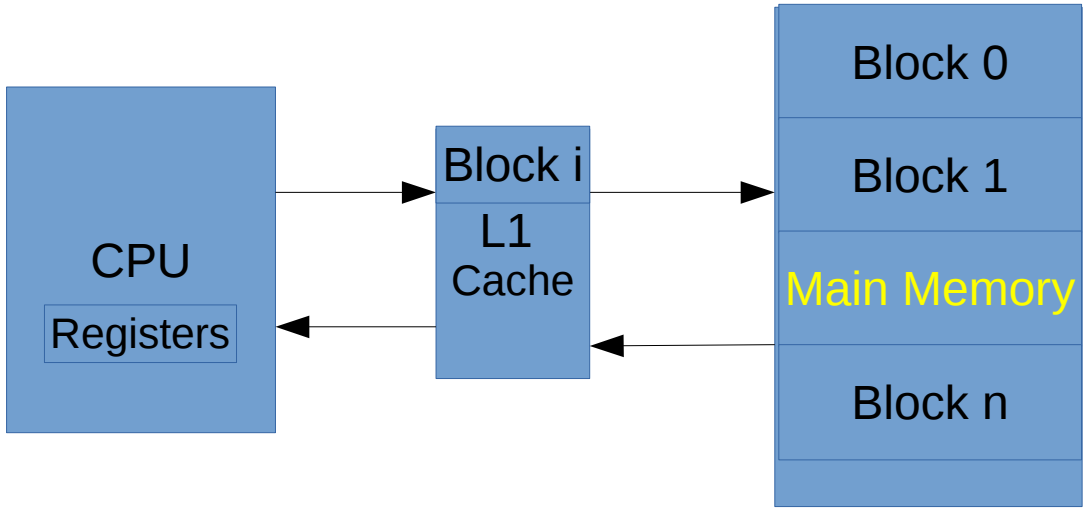
The total number of blocks = 512 / 16 = 32 blocks

Size of block address bits = 5 bits
Size of word address bits = 2
Size of bytes address bits = 2



How do you decide What should be the size of Block? (Research!)

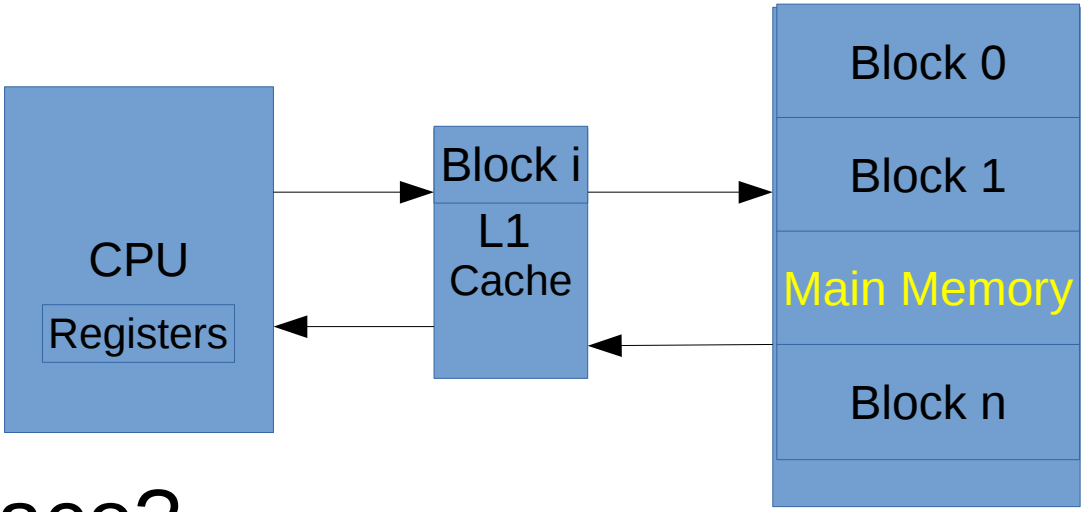
Cache Memory: Four Questions



Its a mapping between main and cache

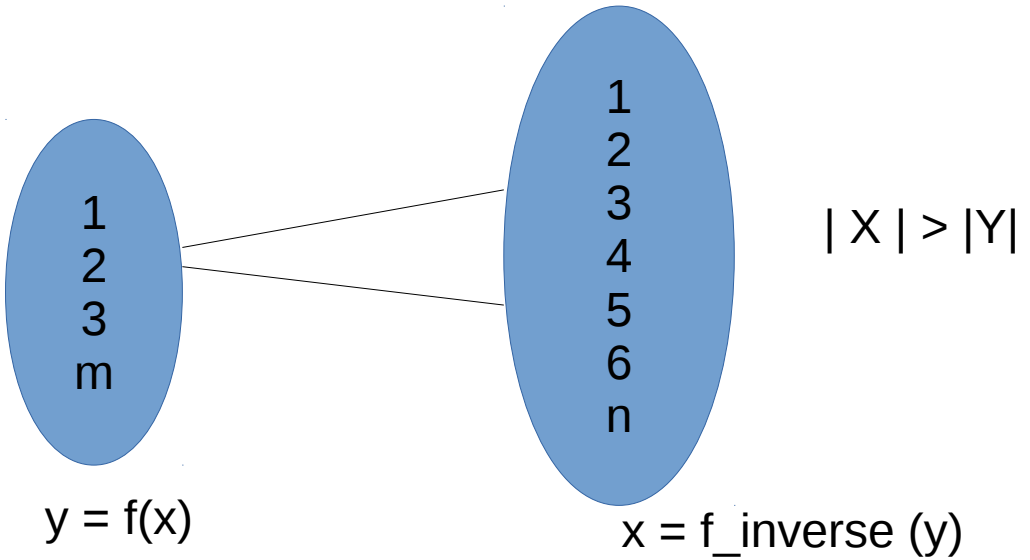
- Where to place a block in the cache?
- How to find the placed block?
- Which block should be replaced on miss?
- What happen when content of block is updated?

The First Question

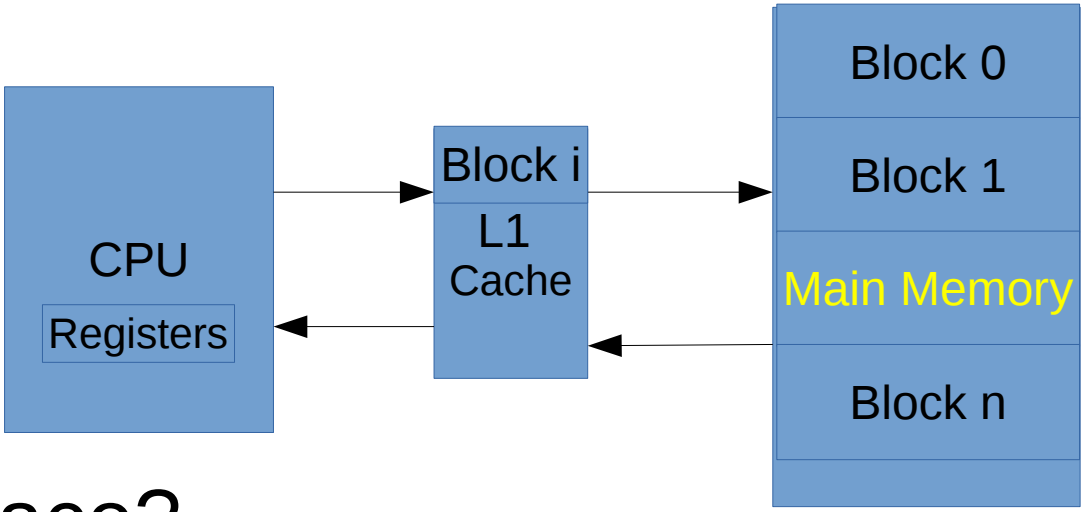


Where to place?

Why to worry about this question? Because size of Cache is smaller than size of main.



The First Question



Where to place?

General thoughts:

Thought 1: Let any main memory bock occupy any cache block
Thought 2: Only a selected main memory block occupy a designated set of cache block
Thought 3: Let a designated set of main block occupy a fixed cache block
Thought 4: Any other possibilities? Looks like no possibility!

Standard name:

Thought 1: Fully associative mapping
Thought 2: Set associative mapping
Thought 3: Direct mapping

Last page

Reference:

- Bruce Jacob, Spencer Ng, and David Wang; *Memory Systems: Cache, DRAM, Disk;* 2008, Elesevier. (Refer: Chapter 1 and Chapter 5)

- Hennessy & Patterson, *Computer Architecture Quantitative Approach*, Appendix B, Review of Memory Hierarchy,

- Patterson & Hennessy, *Computer Organization and Design*, Chapter 5, Large and Fast Exploiting Memory Hierarchy .

Thank You

Additional Pages