Subject: Computer architecture. Lec 1 - Memory.

Main Memory:
- RAM ⇒ Random access memory
- ROM ⇒ Read only memory.

RAM:
- DRAM ⇒ Dynamic RAM.
- SRAM ⇒ Static RAM.

DRAM
- Consists Capacitors which slowly leaks their charge. Thus, They must be refreshed periodicly to prevent data loss.
- Simple in design and cheap.

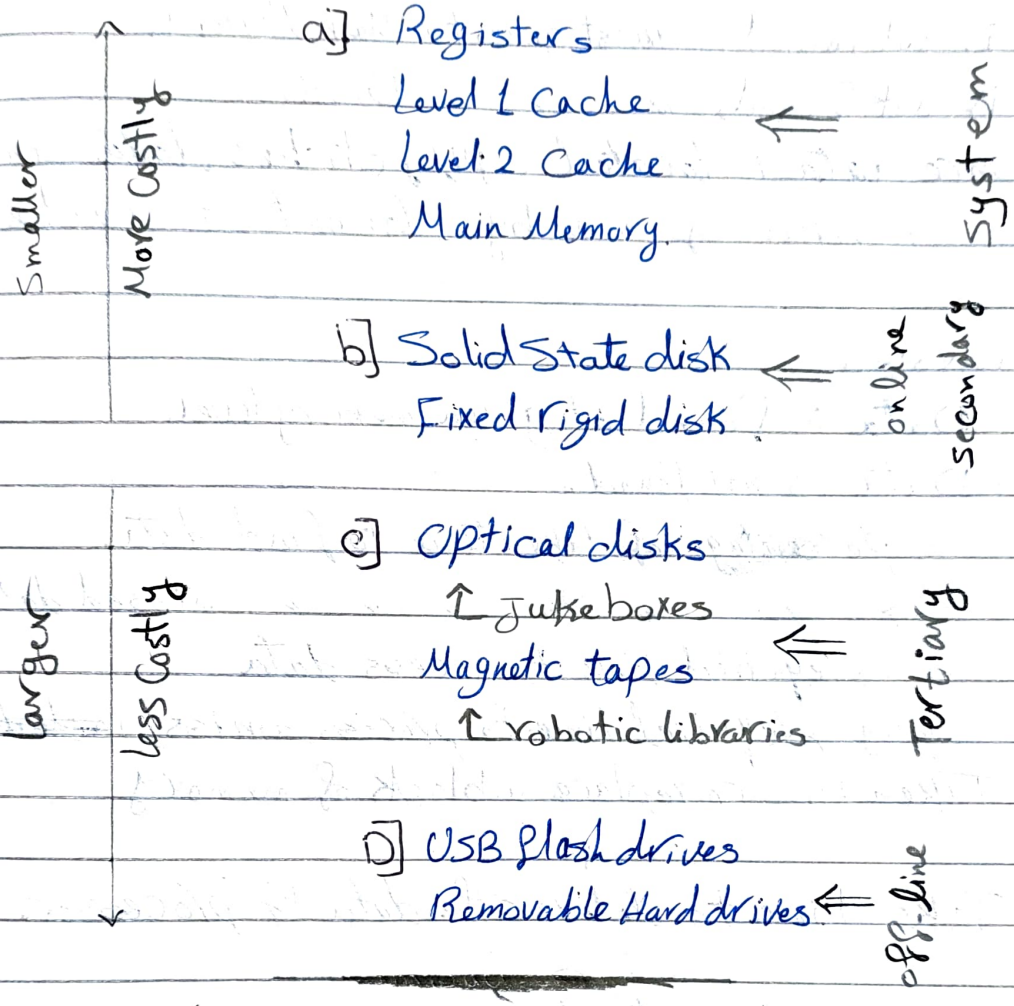
SRAM
- Consists Circuits similar to D Flipflops
- Doesn't need to be refreshed
- Very fast memory.
- Used to build ~~the Cache~~ Cache memory.

ROM
- Doesn't need to be refreshed. "needs alittle charge".
- Used to store permanent or Semi-permanent data.
- Data ~~persists~~ persits even while System turned off.
- Cheaper than RAM and slower than RAM.

*
* Generally faster memory is more expensive than slower memory.
* Hierarchical fashion used to provide **best performance at the lowest cost**
* Small, fast storage elements are kept in the **CPU**
* Larger, permanent storage are kept further from CPU.

Subject : ..... Date: / /

Storage organization:-



- Registers:- Storage locations available on the processor itself.
- Virtual Memory - implemented using a hard disk.
  - Used to extend address space from RAM to hard drive.
  - Provides more space.

- Cache Memory - provides speed.

Subject: ..... Date: / /

- Data Fetching:
  1. CPU send request to cache. "Nearest memory"
  2. if not found, request goes to Main Memory. (data ←)
  3. if not found, " " " Disk. (data ←)
  4. Once data is located, CPU fetches data and number of nearby elements to cache.

- Definitions:
  - hit → Data is found at given memory level.
  - miss → Data is not found.
  - hit rate → percentage of taken time to find data.
  - miss rate → " " " " " missed data.
  - hit time → required time to access data
  - miss penalty → required time to process a miss, ~~includes~~:
    - = Taken time to replace a block of memory
    - + Taken time to deliver the data to processor.

* Miss rate = 1 - hit rate

- Locality:
  principle of locality: Once a byte is accessed, the nearby data will be needed soon.
  so, while hitting, ~~a block~~ an entire block is copied.

Subject : ..... Date: / /

- locality forms:
  - Temporal locality → Recently accessed data tends to be accessed again.
  - Spatial locality → Accessed data tends to be cluster. (تجميع البيانات التي تم الوصول إليها)
  - Sequential locality → Accessed instructions tends to be accessed sequentially.

- Cache Memory:
  - purpose → Speed up data accessing by storing recently used data closer to CPU, instead of Main memory.
  - Content addressable Memory → accessed by Content.
  - Smaller than Main Memory but faster than it.
  - Single large Cache Memory not desirable because it takes longer time to search and that is because it searched by Content not address.

* Main Memory's data accessed by address while ← fast
  Cache Memory's data accessed by Content ← slow

- Cache Mapping Schemes:

1. Direct Mapped Cache.
2. Fully associative mapped Cache.
3. N-Way set associative Mapped cache.

Subject : .....

Date: / /

1- Direct cache mapping :-

- The simplest cache mapping scheme.
- $Y = X \bmod N$.

$Y \rightarrow$ Number of block in Cache memory.

$X \rightarrow$ Number of block in Main ~~M~~memory.

$N \rightarrow$ ~~Num~~ Total blocks number in Cache.

- Main memory address :-

Tag	Block	offset

- ~~Tag. offset.~~
- offset : identifies address within block.
يحدد مكان الداتا جوه الـ Block
- Block : identifies which block contains data.
يحدد الـ Block اللي جواه الداتا
- Tag : Marks data in block.

- Mapping Mechanism :

1- Determine address format for mapping :-

- number of bits for offset $\Rightarrow 2^n = x$

$x \rightarrow$ Bytes per block. "Size".

- n-bits for Block $\Rightarrow 2^n = x$

$x \rightarrow$ Number of Blocks in Cache

- n-bits for Tag $\Rightarrow$ remaining bits.
- Use modulo algorithm to get mapped block in Cache.
- n-bits for address $\Rightarrow$ Memory Size $= 2^{(n)}$
main

Subject : ..... Date: / /

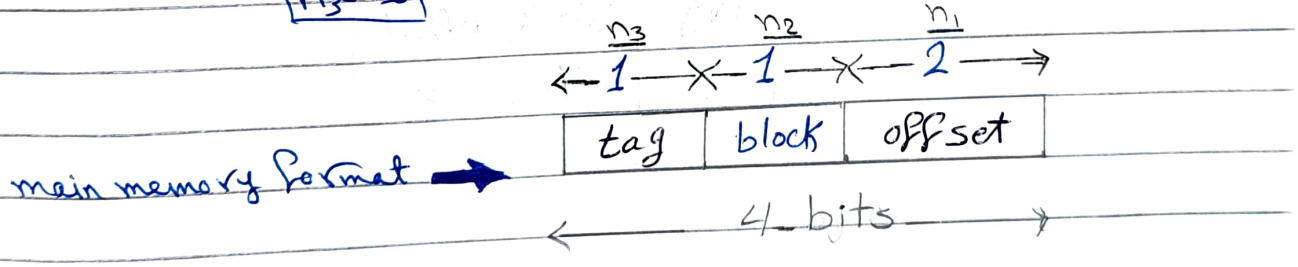
ex:- Byte addressable main memory has 4 blocks, and a cache with 2 blocks where each block is 4-byte.

⇒ M-blocks in main memory = 4
N-blocks in Cache = 2
Address = ~~4 byte~~ 16-bit = 2^4 ⇒ address bits = 4

(a) n-bits for offset ⇒ $2^n = (4)$ ← each block is 4-byte "size"
$\boxed{n_1 = 2}$

(b) n-bits for block ⇒ $2^n = (2)$ ← 2 blocks in Cache.
$\boxed{n_2 = 1}$

(c) n-bits for tags ⇒ Size - (bits for offset) - (bits for block)
n_1 n_2
$n = 4 - (2) - (1)$
$\boxed{n_3 = 1}$



Date: / /

ex 2:- Assume a byte addressable memory consist of 2^{14} byte, Cache has 16 block, each block has 8 byte.

→ word = 1 byte

memory size = 2^{14} byte

∴ Address bits = 14 bit.

n-bits for offset ⇒ block size = 8 = 2^3 byte

n = 3 bits.

n-bits for block ⇒ Number of blocks = 16 = 2^4

n = 4 bits.

n-bits for tag ⇒ remainder bits

n = 14 - (3+4)

n = 7 bits.

tag	block	offset
7	4	3

Subject : ..... Date: / /

ex: A 16-bit memory address and 64 block of Cache where each block contains 8 byte.

⇒ Address bits = 16-bit.

offset ⇒ block size = 8 = 2^3 ⇒ offset bits = 3 bits

Block ⇒ Number of blocks = 64 = 2^6 ⇒ block = 6 bits

Tag ⇒ Address bits - [offset bits - block bits]
Tag = 7 bits.

- Direct cache mapping depends on.
  - Number of bits in the main memory address. ⇒ address bits
  - Number of blocks in Cache memory. ⇒ Size of block field
  - Number of addresses in block. ⇒ Size of offset field.

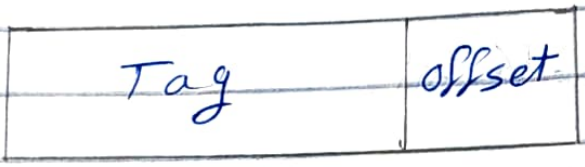
- Direct Cache mapping issue:
  Memory block is replacing Cache block depends on memory address without checking for free space, which leads to Data loss in Cache.

Subject : .....

Date: / /

2. Fully associative cache mapping:-

- Main memory address:-



- When Searching in Cache, all tags are being Searched in parallel to retrieve data quickly.

- Requires special and costly hardware.

- Mechanism:-
  - Searching for ~~empty~~ free space in Cache, then Memory block mapped to this place.
  - if there is no free space, there is an algorithm used to find Victim block which is evicted.

Subject : ..... Date: / /

3. Set associative Cache: "N-Way set associative".

- It Combines the ideas of direct mapping and fully associative cache.

- Memory reference is being mapped to free space in subset of cache slot.

- Number of cache blocks in a set varies according to System design, ex:
  - 2 way set associative cache
    - ↳ Set = 2 blocks

- Main memory address:-

Tag	Set	offset

- offset → Chooses the byte within cache block. ~~to be mapped.~~

- Set → Determines which set has the block to be mapped.

Subject : ..... Date: / /

ex: Suppose a byte addressable main memory contain 1 MB. Cache consist of 32 line, each block contains 16 byte, using:

A) Direct:- Memory size = 2^{20}

Address bits = 20 bit.

offset → block size = 16 = 2^4

offset bits = 4 bit.

Block → Number of blocks = 32 = 2^5

Block bits = 5 bit.

Tag → 20 - [4+5]

Tag = 11 bit.

Tag	Block	offset
11	5	4

B) Fully associative:-

Address bits = 20 bit.

offset bits = 4 bits

Tag bits = 16 bits

Tag	offset
16	4

Subject : ..... Date: / /

c) (4) Way set associative:-

Address bits = 20 bit

~~n.sets = 4 sets~~

offset bits = 4 bit.

number of sets → Number of blocks in a set = (4)

$$n.set = \frac{\text{Number of blocks in Cache}}{\text{Number of blocks in a set}} = \frac{32}{4} = 8$$

$n.set = 8 = 2^3$

set bits = 3 bit

Tag = 20 - [3+4] = 13 bit.

Tag	set	offset
13	3	4