Advanced Computer Architecture (0630561)

Lecture 8

Memory Hierarchy

Prof. Kasim M. Al-Aubidy

Computer Eng. Dept.

Introduction:

Goal: Illusion of large, fast, cheap memory. Let programs address a memory space that scales to the disk size, at a speed that is usually as fast as register access

Solution: Put smaller, faster "cache" memories between CPU and DRAM. Create a "memory hierarchy".

- Main memory: fast, random access, expensive, located close (but not inside) the CPU.
  Is used to store program and data which are *currently manipulated* by the CPU.
- Secondary memory: slow, cheap, direct access, located remotely from the CPU.

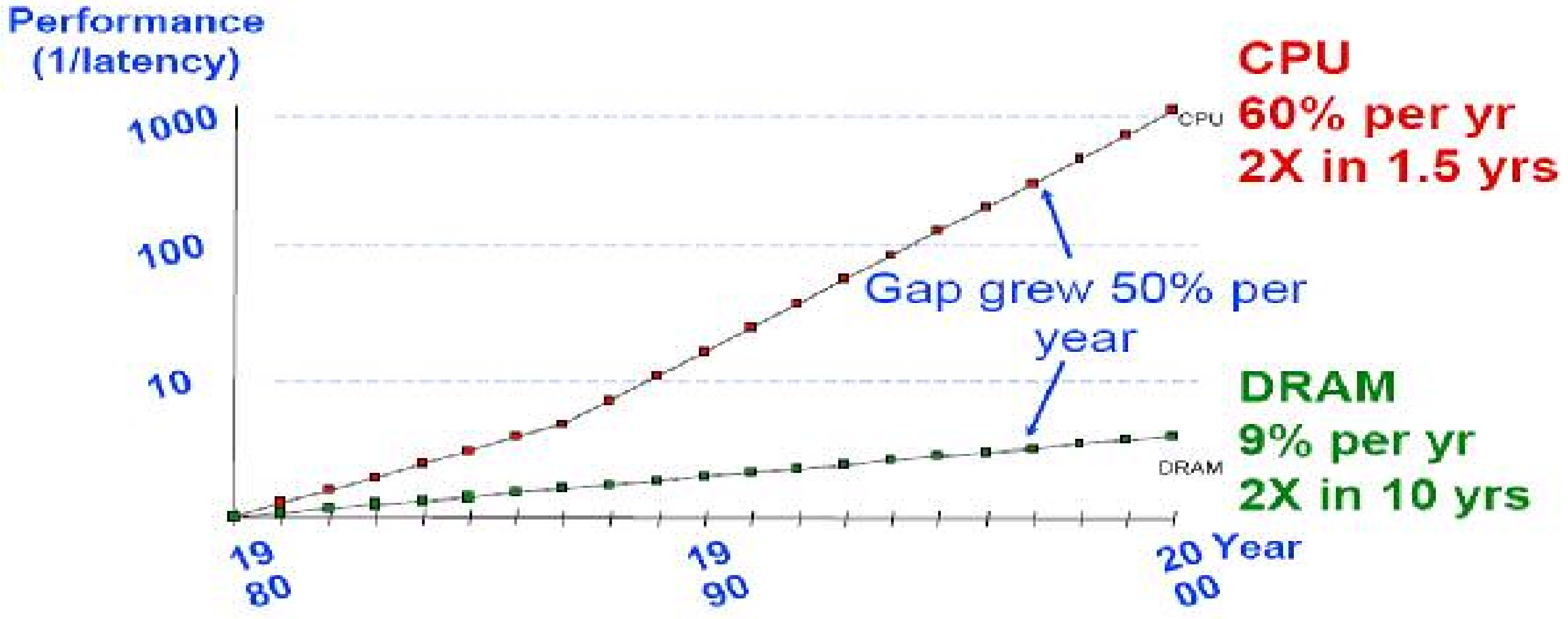
What do we need?

We need memory to fit very large programs and to work at a speed comparable to that of the microprocessors.

Main problem:

- microprocessors are working at a very high rate and they need large memories;
- memories are much slower than microprocessors;

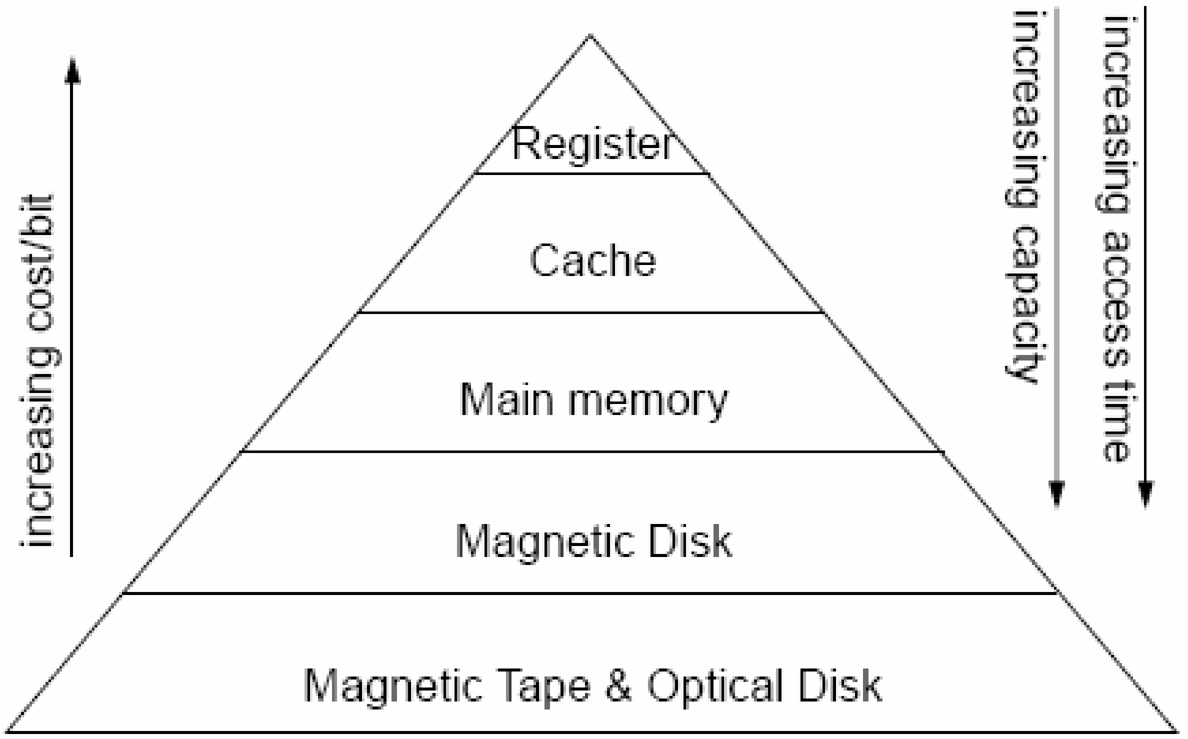
Four-issue 2GHz superscalar accessing 100ns DRAM could execute 800 instructions during time for one memory access!



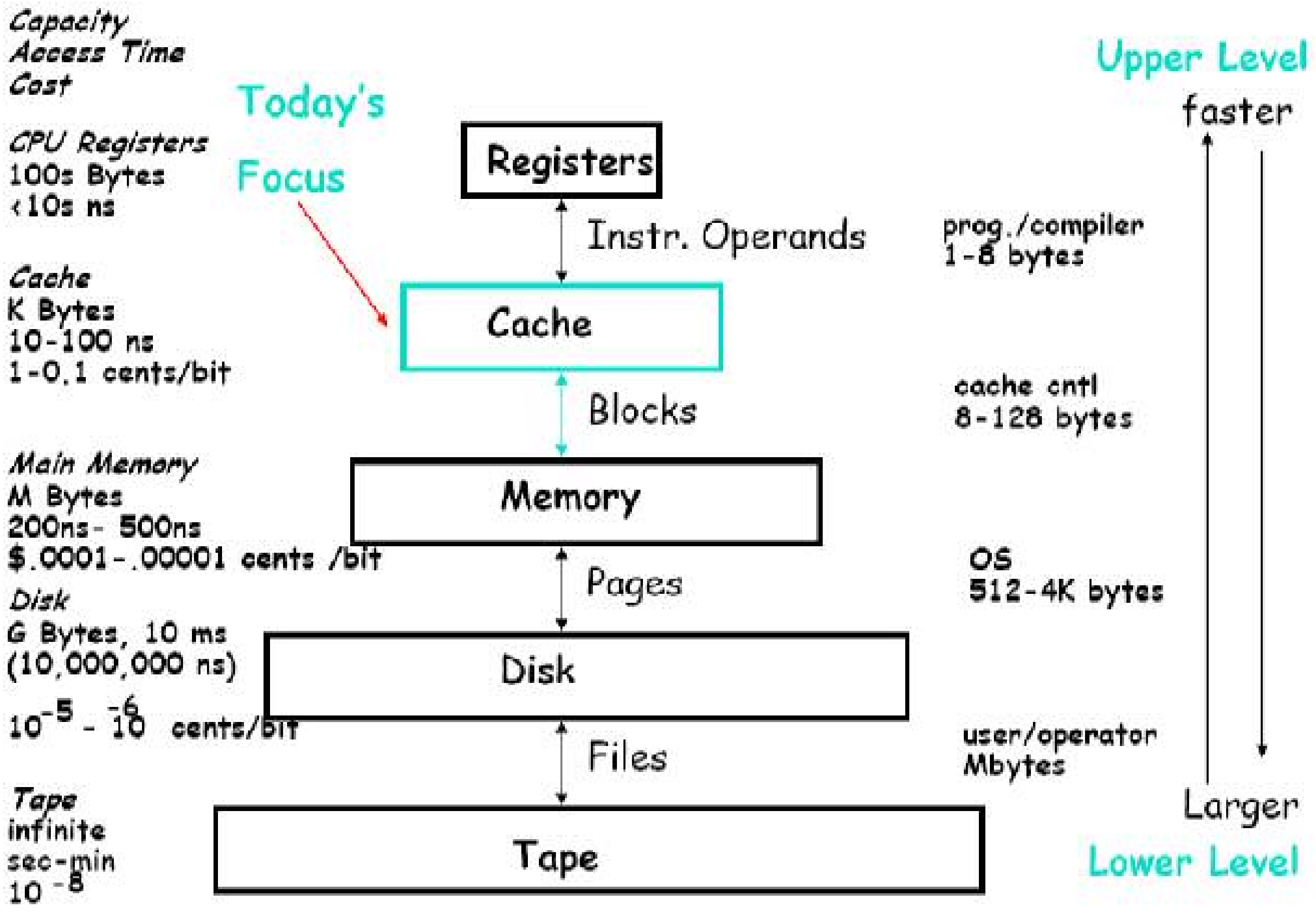
A Solution

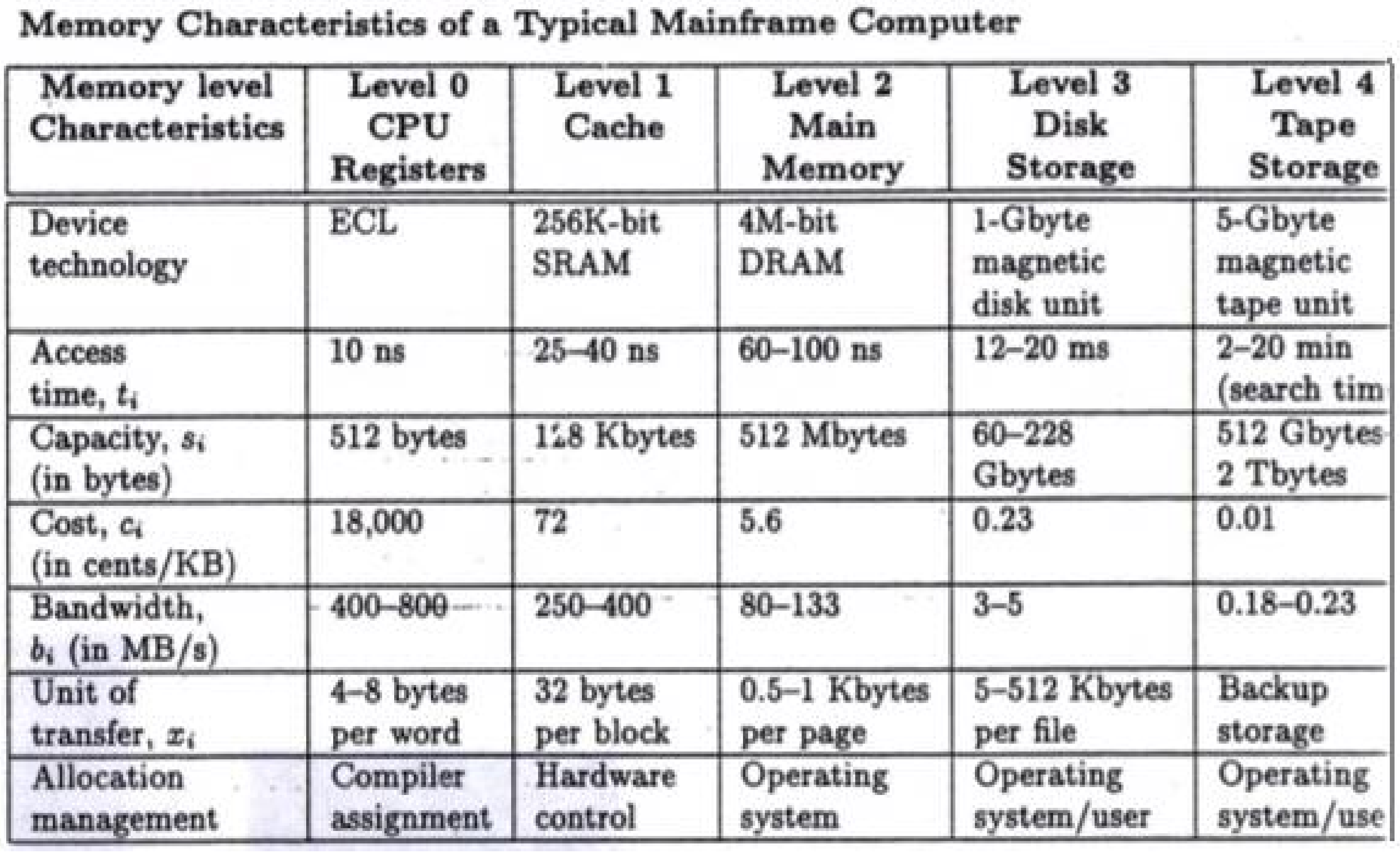
It is possible to build a composite memory system which combines a *small, fast memory* and a *large slow main memory* and which behaves (most of the time) like a large fast memory.

The two level principle above can be extended into a *hierarchy of many levels* including the secondary memory (disk store).



Levels of the Memory Hierarchy:



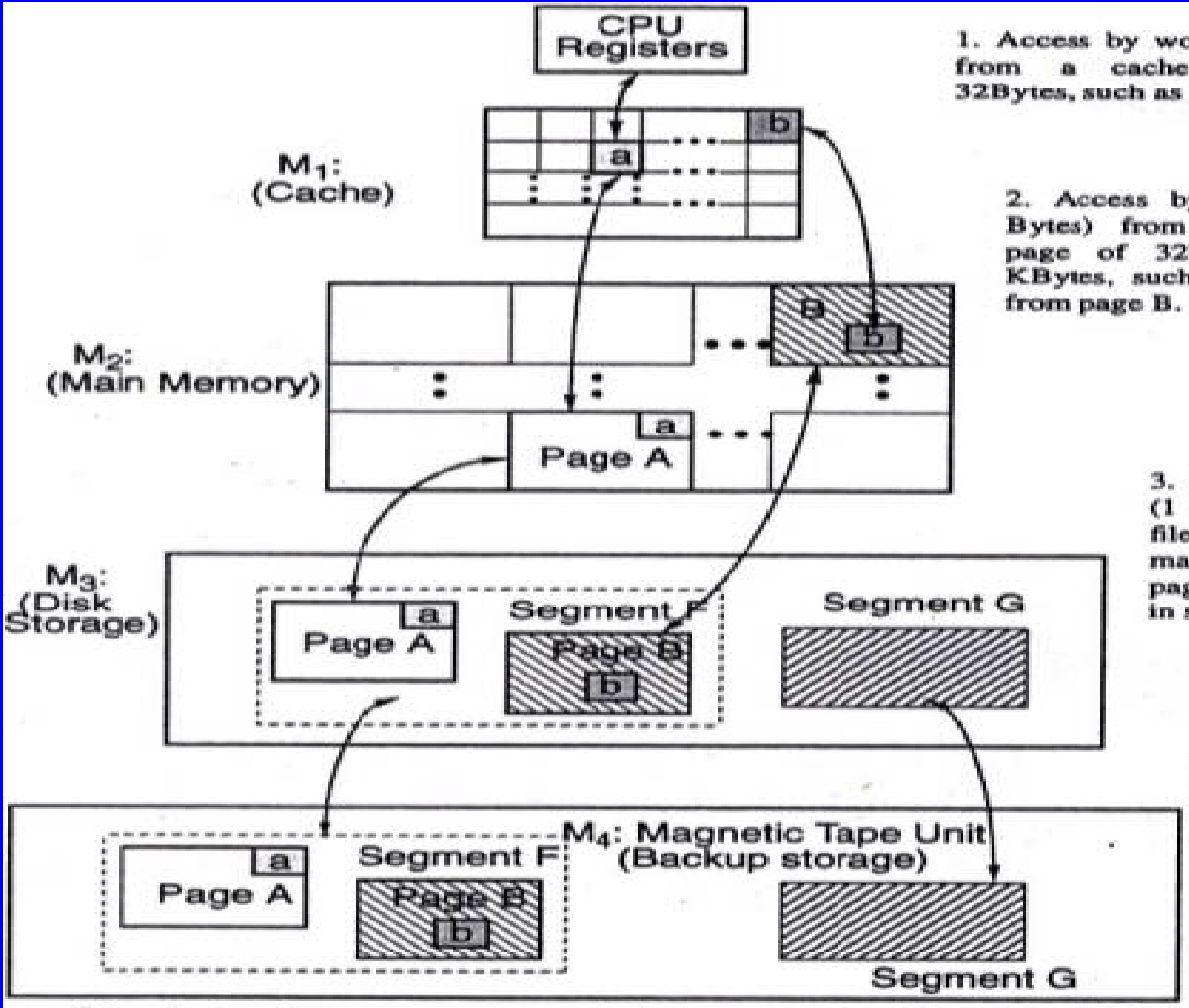


Memory Characteristics of a Typical Mainframe Computer

Memory level Characteristics	Level 0 CPU Registers	Level 1 Cache	Level 2 Main Memory	Level 3 Disk Storage	Level 4 Tape Storage
Device technology	ECL	256K-bit SRAM	4M-bit DRAM	1-Gbyte magnetic disk unit	5-Gbyte magnetic tape unit
Access time, t_i	10 ns	25–40 ns	60–100 ns	12–20 ms	2–20 min (search tim
Capacity, s_i (in bytes)	512 bytes	128 Kbytes	512 Mbytes	60–228 Gbytes	512 Gbytes– 2 Tbytes
Cost, c_i (in cents/KB)	18,000	72	5.6	0.23	0.01
Bandwidth, b_i (in MB/s)	400–800	250–400	80–133	3–5	0.18–0.23
Unit of transfer, x_i	4–8 bytes per word	32 bytes per block	0.5–1 Kbytes per page	5–512 Kbytes per file	Backup storage
Allocation management	Compiler assignment	Hardware control	Operating system	Operating system/user	Operating system/use

Memory Hierarchy Properties:

- Information stored in a memory hierarchy (M_1, M_2,..M_n) satisfies three important properties:
- **Inclusion Property:** it implies that all information items are originally stored in level M_n. During the processing, subsets of M_n are copied into M_{n-1}. similarity, subsets of M_{n-1} are copied into M_{n-2}, and so on.
- **Coherence Property:** it requires that copies of the same information item at successive memory levels be consistent. If a word is modified in the cache, copies of that word must be updated immediately or eventually at all higher levels..
- **Locality of References:** the memory hierarchy was developed based on a program behavior known as locality of references. Memory references are generated by the CPU for either instruction or data access. Frequently used information is found in the lower levels in order to minimize the effective access time of the memory hierarchy.



1. Access by word (4 Bytes) from a cache block of 32Bytes, such as block a.

2. Access by block (32 Bytes) from a memory page of 32block or 1 KBytes, such as block b from page B.

3. Access by page (1 KBytes) from a file consisting of many pages, such as page A and page B in segment F.

4. Segment transfer with different number of pages.

The inclusion property and data transfers between adjacent levels

Memory Capacity Planning:

- The performance of a memory hierarchy is determined by the effective access time (T_{eff}) to any level in the hierarchy. It depends on the hit ratio and access frequencies at successive levels.

- **Hit Ratio (h):** is a concept defined for any two adjacent levels of a memory hierarchy. When an information item found in Mi, it is a hit, otherwise, a miss. The hit ratio (h_i) at Mi is the probability that an information item will be found in Mi. the miss ratio at M_i is defined as $1-h_i$.

- The **access frequency** to Mi is defined as

$$f_i = (1-h_1)(1-h_2)\ldots.(1-h_i)$$

Effective Access Time (T_{eff}):

- In practice, we wish to achieve as high a hit ratio as possible at M1. Every time a miss occurs, a penalty must be paid to access the next higher level of memory.

  The T_{eff} of a memory hierarchy is given by:

$$\begin{aligned} T_{eff} &= \sum_{i=1}^{n} f_i \cdot t_i \\ &= h_1 t_1 + (1-h_1)h_2 t_2 + (1-h_1)(1-h_2)h_3 t_3 + \cdots + \\ &\quad (1-h_1)(1-h_2)\cdots(1-h_{n-1})t_n \end{aligned}$$

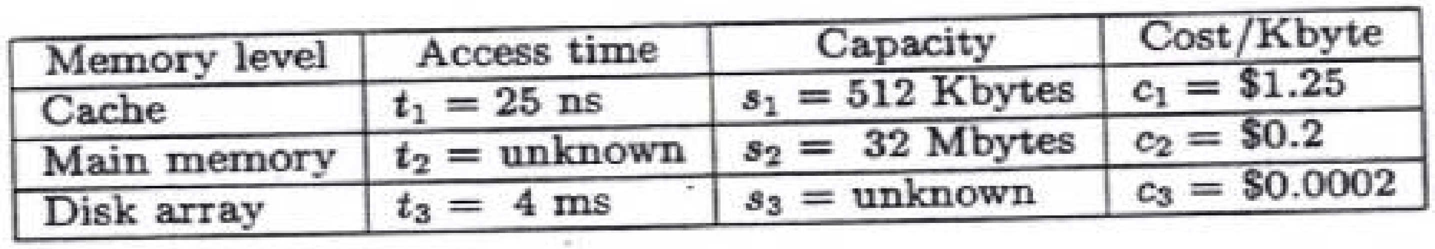
Hierarchy Optimization:

The total cost of a memory hierarchy is estimated as:

$$C_{total} = \sum_{i=1}^{n} c_i \cdot s_i$$

Example:

Consider the design of a three-level memory hierarchy with the following specifications for memory characteristics:



Memory level	Access time	Capacity	Cost/Kbyte
Cache	$t_1 = 25$ ns	$s_1 = 512$ Kbytes	$c_1 = \$1.25$
Main memory	$t_2 =$ unknown	$s_2 = 32$ Mbytes	$c_2 = \$0.2$
Disk array	$t_3 = 4$ ms	$s_3 =$ unknown	$c_3 = \$0.0002$

The design goal is to achieve an effective memory access time (t=10.04 μs) with a cache hit ratio (h_1=0.98) and a main memory hit ratio (h_2=0.9). The total cost of memory hierarchy is limited by $15000.

Solution:

Memory cost is calculated by;

$$C_{total}= C_1S_1+C_2S_2+C_3S_3 \leq 15000, \text{ then } S_3=39.8$$

The effective memory access time is calculated as

$$T_{eff}=h_1t_1+(1-h_1)h_2t_2+(1-h_1)(1-h_2)h_3t_3 \leq 10.04, \text{ then } t_2= 903 \text{ ns.}$$

Note: If one wants to double the main memory to 64 Mbytes at the expense of reducing the disk capacity under the same budget limit. This change will not affect the cache hit ratio. But it may increase the hit ratio in the main memory if a proper page replacement algorithm is used.

Cache Algorithm:

Look at Processor Address, search cache tags to find match. Then either

HIT - Found in Cache

Return copy of data from cache

MISS - Not in cache

Read block of data from Main Memory
Wait ...
Return data to processor
and update cache

Hit Rate = fraction of accesses found in cache

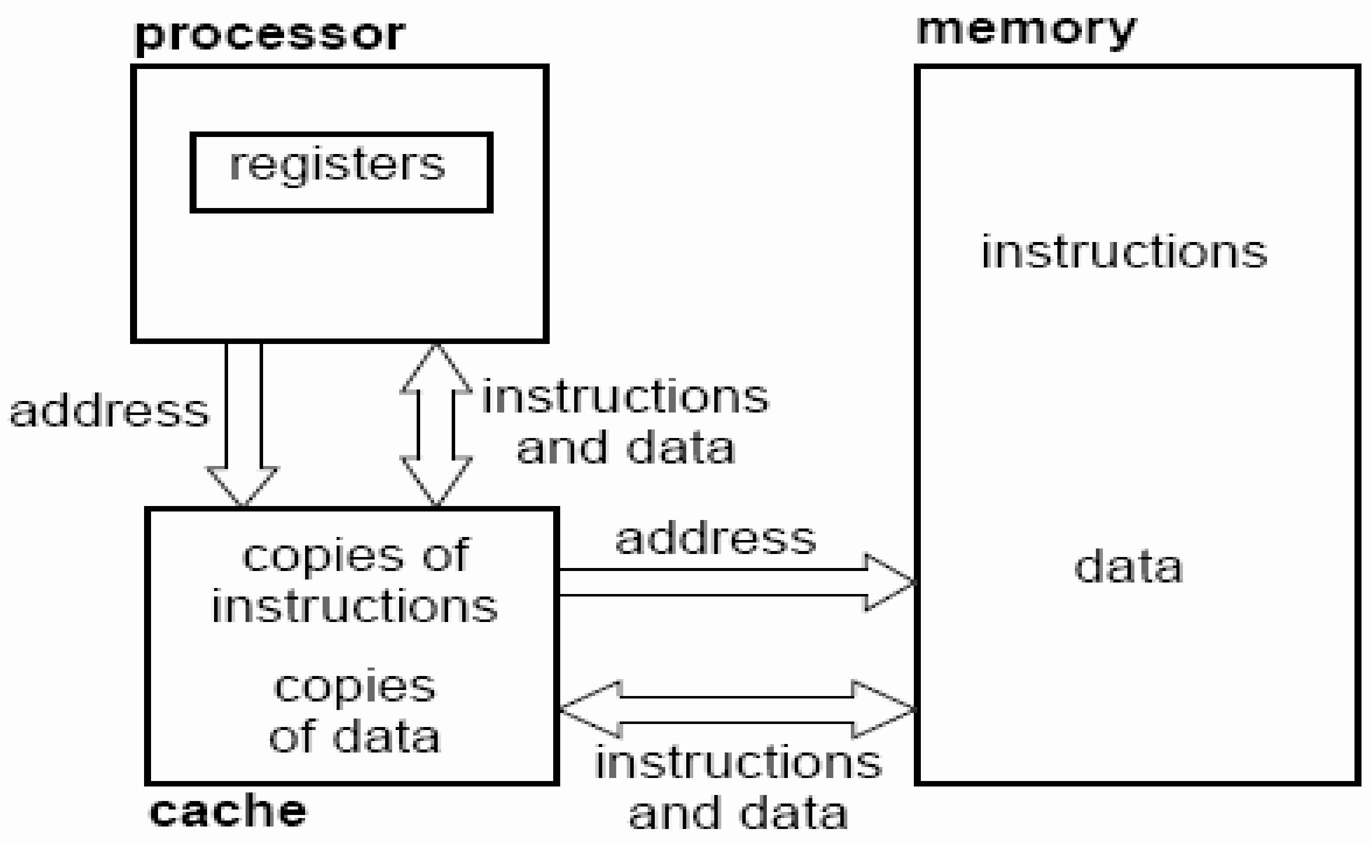
Miss Rate = 1 – Hit rate

Hit Time = RAM access time + time to determine HIT/MISS

Miss Time = time to replace block in cache + time to deliver block to processor

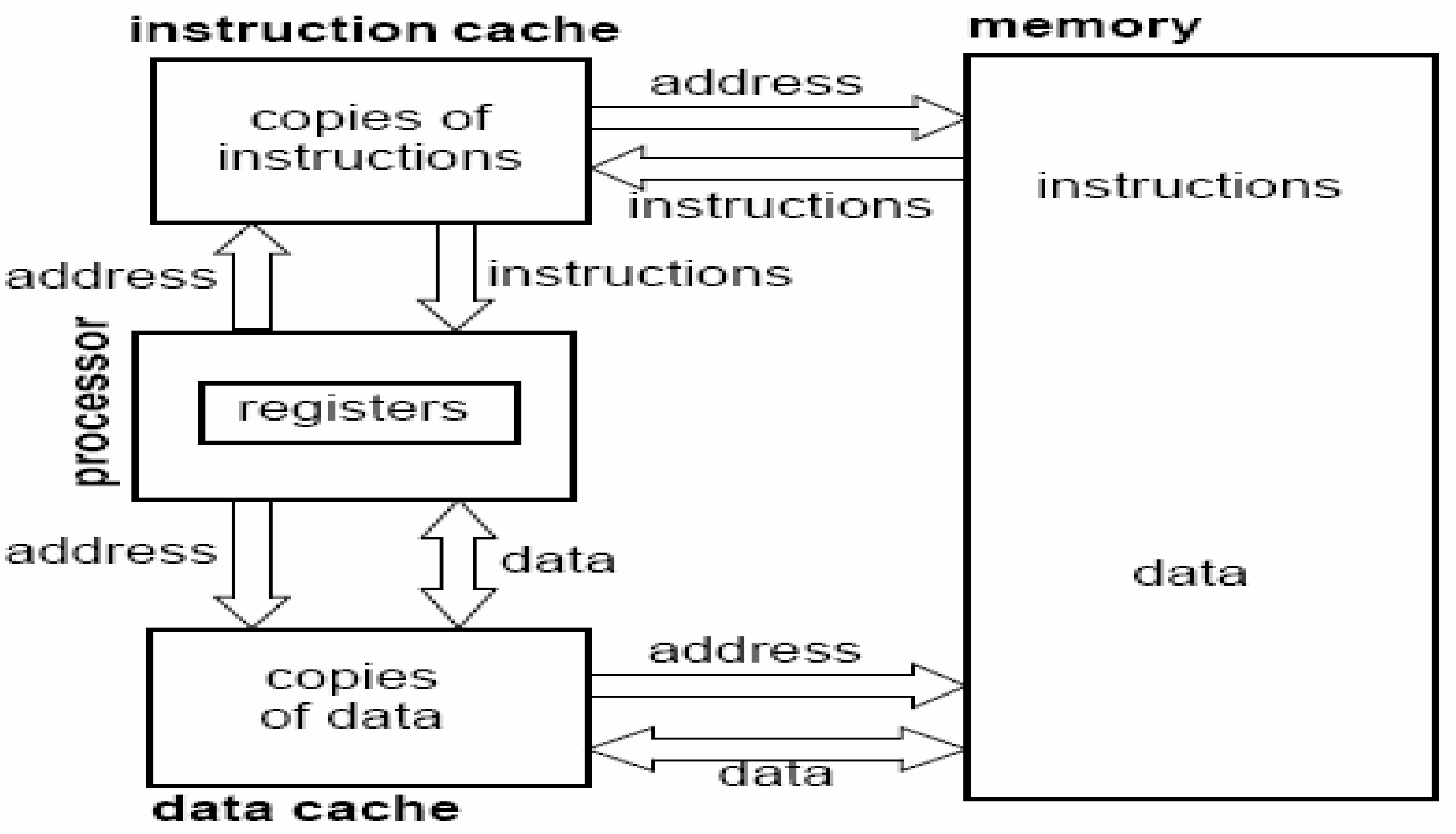
Cache Memory:

A cache memory is a small, very fast memory that retains copies of recently used information from main memory. It operates transparently to the programmer, automatically deciding which values to keep and which to overwrite.



Cache Memory:

It is common also to split the cache into one dedicated to instructions and one dedicated to data.


instruction cache
copies of instructions
address
instructions
memory
instructions
address
instructions
processor
registers
address
data
data
copies of data
address
data
data cache