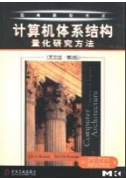


5.8 Main Memory and Organizations for Improving Performance

- **Main memory** is the next level down in the hierarchy.
We usually call it inter-memory(EMS memory)
Main memory is usually made from DRAM while caches use SRAM.
- **Performance** of main memory
 - **Latency** *harder to reduce latency*
Important for caches.
 - **Bandwidth** *easier to improve bandwidth with new organizations*
Important for I/O.
Also for cache with second-level and larger block sizes.
- The **previous** sections describe cache organization to reduce CPU-DRAM performance gap
- This section we analyze techniques for **organizing memory to improve bandwidth.**





Performance basic memory organization

First-level caches are often organized with a physical width of **1 word** because most CPU accesses are that size

Assume

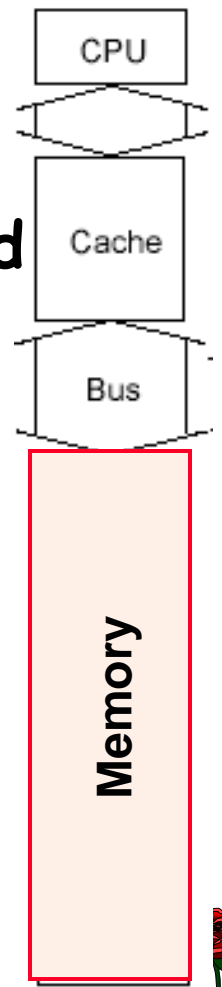
- 4 clock cycles to send the address
- 56 clock cycles for the access time per word
- 4 clock cycles to send a word of data
- Block size is 4 words
- Every word is 8 bytes

The miss penalty:

$$4 \times (4 + 56 + 4) = 256 \text{ CLKs}$$

Bandwidth :

$$\frac{4 \times 8}{256} = \frac{1}{8}$$

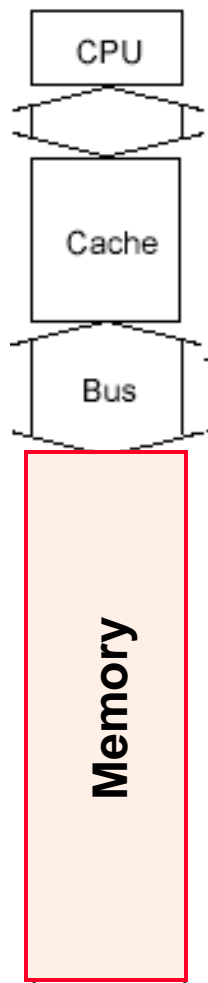


First Technique for Higher Bandwidth: Wider Main Memory

- Amdahl's law suggested that memory capacity should grow linearly with CPU speed.
- Retrospecting
 - Memory capacity grows four-fold every three years to supply this demand.
 - The CPU-DRAM performance gap is a problem, however, since DRAM performance improvement is only about 7% per year.
 - We will now look at innovations in main memory organizations that are more cost effective.
- **Wider Main Memory**
 - Doubling or quadrupling the width of the cache and the memory
 - It will therefore double or quadruple the memory bandwidth.



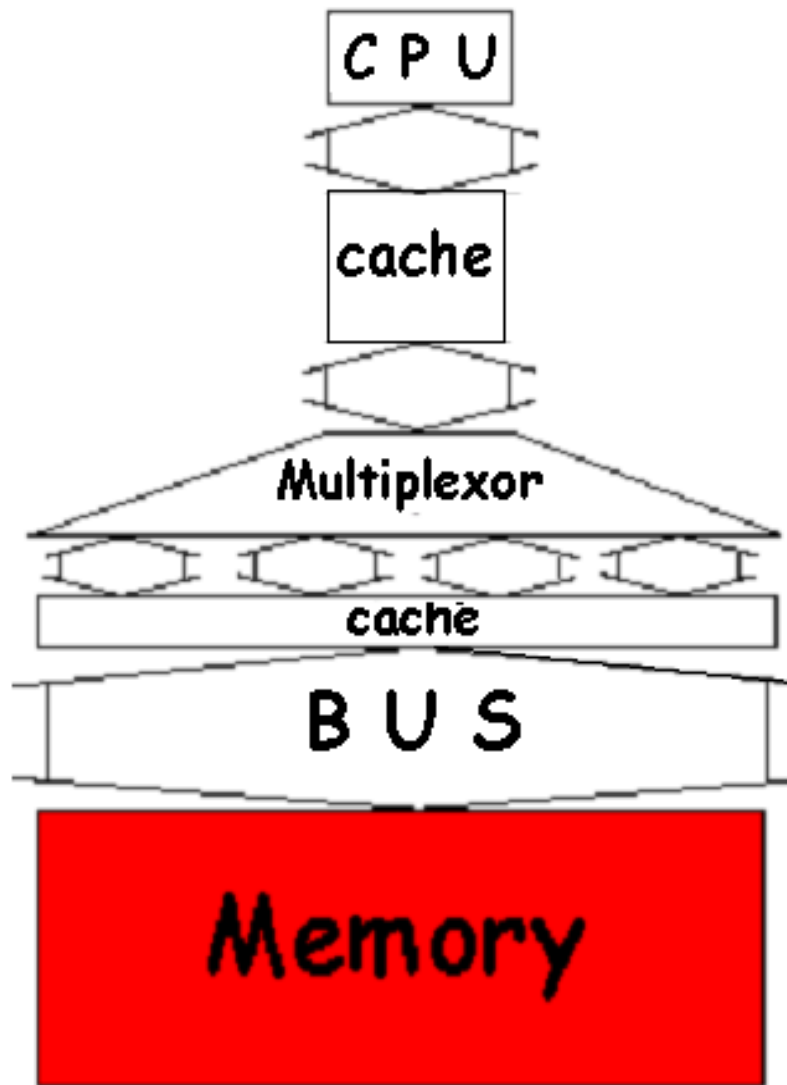
many words in Single bank

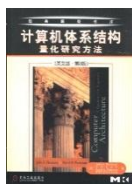


**Widening memory:
Doubles /quadruples
Memory and Bandwidth**



Disadvantages: bus
MUX required on critical
path to allow word access.
Increase minimum memory
increment purchased by
customer.
Complicates error correction





Performance in the Technique

- With a main memory width of 2 words

The miss penalty: 4words/Block

$$2 \times (4 + 56 + 4) = 128 \text{ CLKs}$$

Bandwidth :

$$\frac{4 \times 8}{128} = \frac{1}{4}$$

- With a main memory width of 4 words

The miss penalty: 4words/Block

$$1 \times (4 + 56 + 4) = 64 \text{ CLKs}$$

Bandwidth :

$$\frac{4 \times 8}{64} = \frac{1}{2}$$

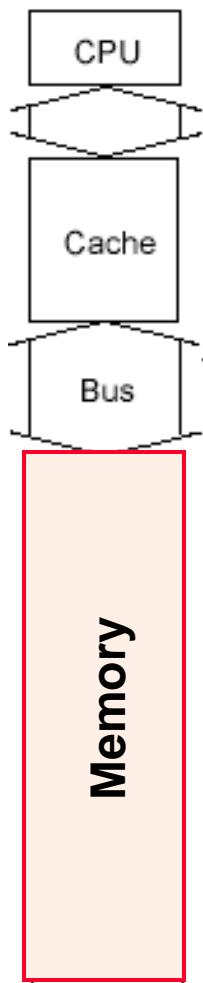


Second Technique for Higher Bandwidth: simple Interleaved Memory

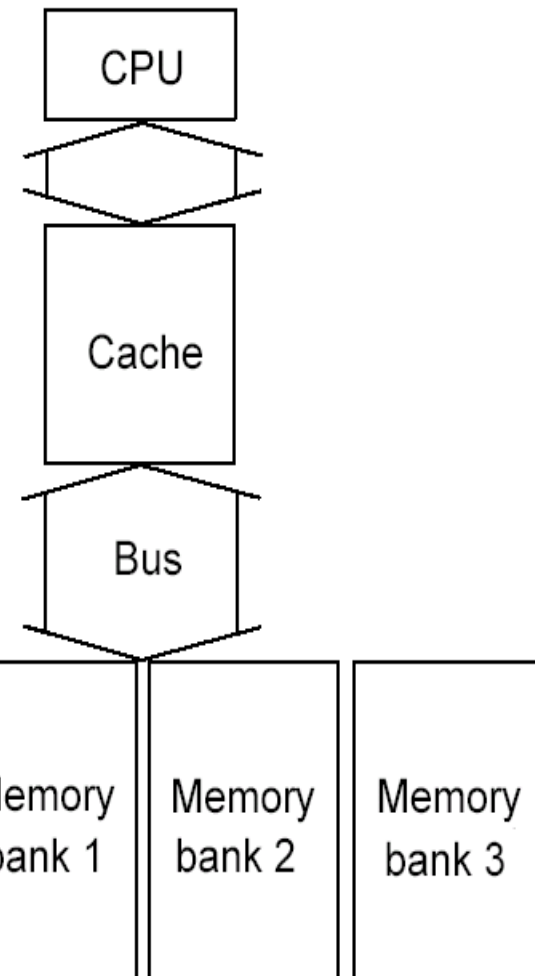
Advantages:

Take advantage of the potential parallelism of having many DRAMs in a memory system.

Such a memory organization is especially important for write through with accessing sequentially

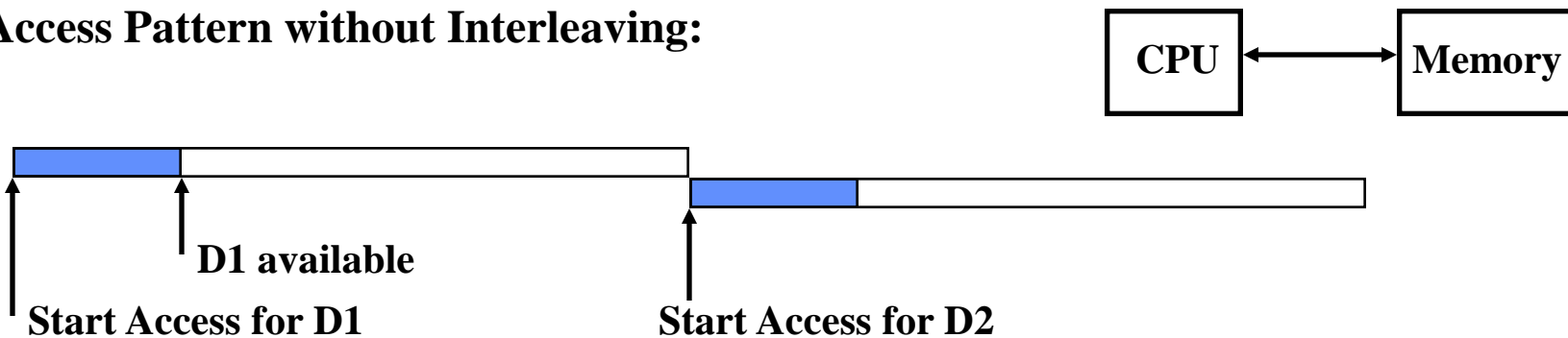


Memory chips can be organized in banks to read or write multiple words at a time. The banks are often 1 word wide so that width of the bus and cache need not change. But sending addresses to several banks permits them all to read simultaneously.

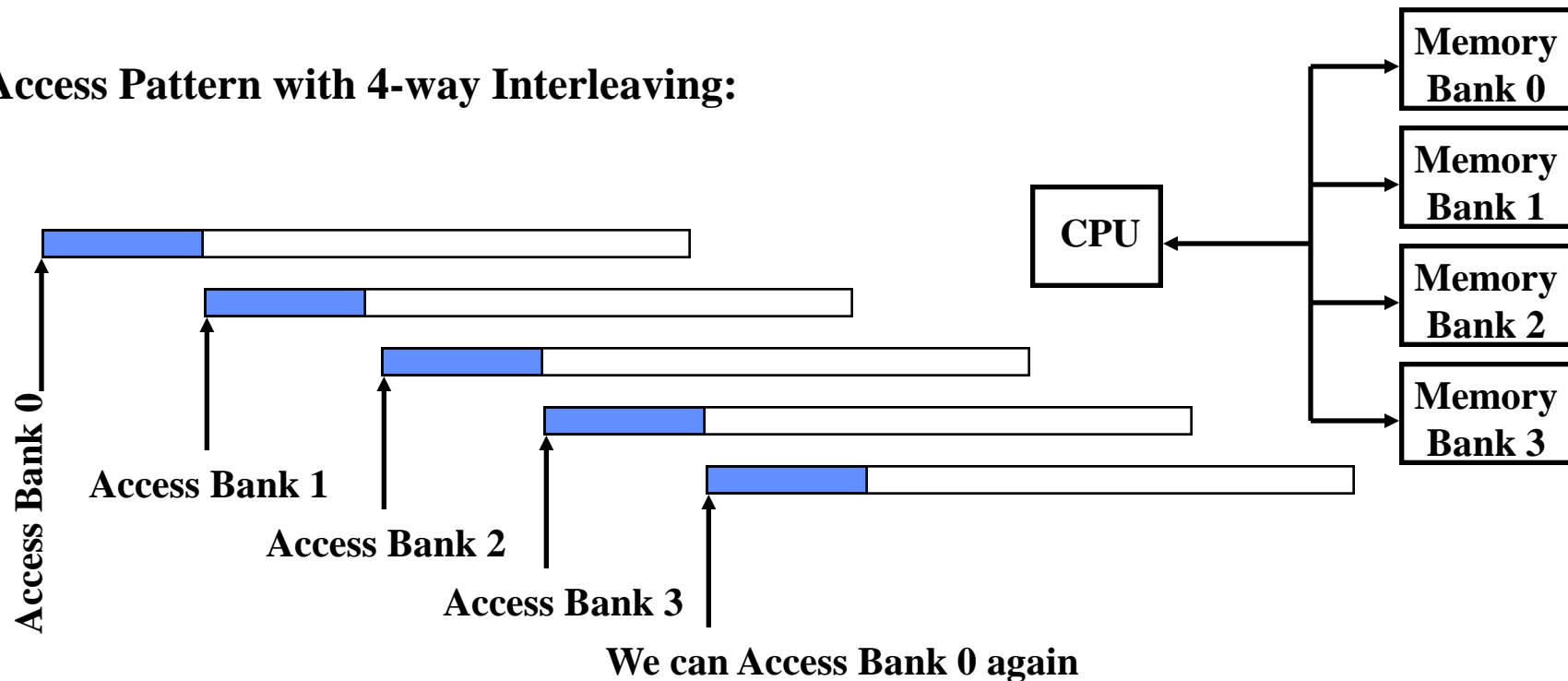


How Access Banks

Access Pattern without Interleaving:



Access Pattern with 4-way Interleaving:





Performance in the Technique

• With 4 banks Interleaved Memory

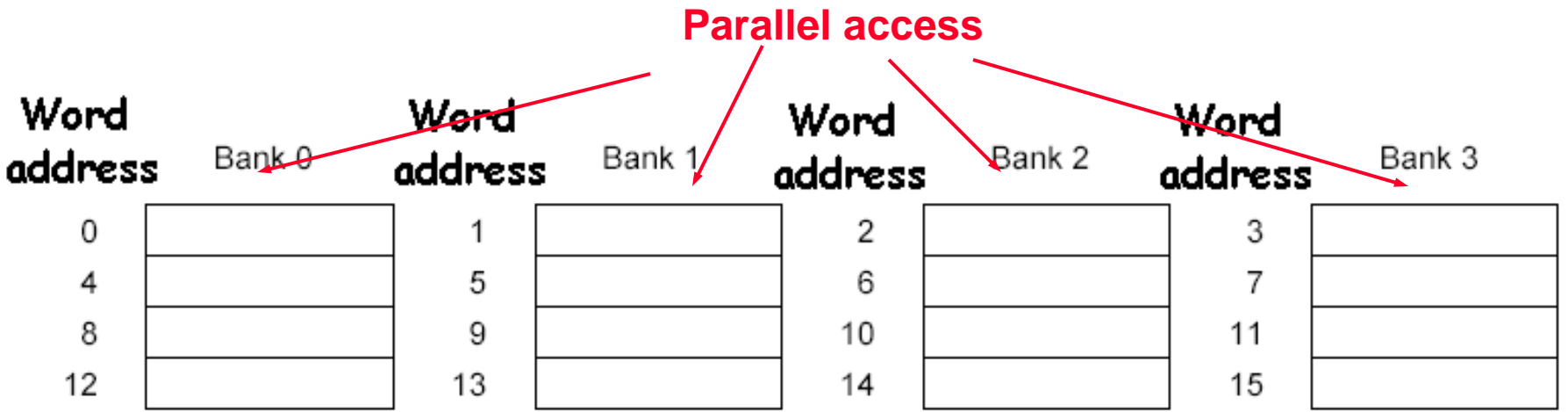
The miss penalty: 4words/Block

$$4+56 + (4 \times 4) = 76$$

Bandwidth :

$$\frac{4 \times 8}{76} = 0.4$$

Four-way interleaved memory



Optimizes sequential address access patterns





Example 14-1 Interleaved Memory

- **Assume:**

Block size = 1 word

Memory bus width = 1 word

Miss rate = 3%

Memory accesses per instruction = 1.2

Cache miss penalty = 64 cycles (as above)

Average cycles per instruction (ignoring cache misses) = 2

- **What can interleaving and wide memory buy?**

- **Answer** The CPI for the base computer using 1-word blocks is

$$\text{CPI} = 2 + (1.2 \times 3\% \times 64) = 4.30$$

we can calculate performance improvement by just comparing CPI.



Example 14-2 Interleaved Memory

- **Answer**

- Increasing the block size to two words gives the following options (miss rate is 2%):

64-bit bus and memory, no interleaving

$$\text{CPI} = 2 + (1.2 \times 2\% \times 2 \times 64) = 5.07$$

64-bit bus and memory, interleaving

$$\text{CPI} = 2 + (1.2 \times 2\% \times (4 + 56 + 8)) = 3.63$$

128-bit bus and memory, no interleaving

$$\text{CPI} = 2 + (1.2 \times 2\% \times 1 \times 64) = 3.54$$

Thus, doubling the block size slows down the straightforward implementation (5.07 versus 4.30), while interleaving or wider memory is 1.19 or 1.22 times faster, respectively.

- **If we increase the block size to four, How about performance?**



Example 14- 4 Interleaved Memory

- **Answer** **Assume: miss rate is 1.2%**
- Increasing the block size to four words gives the following options:
64-bit bus and memory, no interleaving

$$\text{CPI} = 2 + (1.2 \times 1.2\% \times 4 \times 64) = 5.69$$

How many banks should be included? One metric, used in vector computers, is as follows:

Number of banks \geq Number of clock cycles to access word in bank

$$\text{CPI} = 2 + (1.2 \times 1.2\% \times 2 \times 64) = 3.84$$

Again, the larger block hurts performance for the simple case (5.69 versus 4.30), although the interleaved 64-bit memory is now fastest—1.39 times faster versus 1.12 for the wider memory and bus.



third Technique for Higher Bandwidth: Independent Memory Banks

- **Independent memory banks**
 - The interleaved memory concept can be extended to remove all restrictions on memory access.
 - » Independent memory controller was present for every bank.
 - » Using separate address line and data bus
 - » This allowed the interleaving of sequential access patterns.





Example 15 Avoiding Memory Bank Conflicts

Assume

Memory banks **128**, interleaved on a word basis, and execute following code:

Solutions There are both software and hardware **solutions**

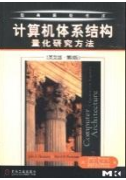
The compiler :

- Loop interchange optimization to avoid accessing the same bank.
- For the programmer or the compiler to **expand the size of the array** so that it is **not a power of 2**, thereby forcing the addresses above to go to different banks.

The hardware: Using a prime number of memory banks $2^n - 1$

on data cache misses no matter how sophisticated a CPU or memory system.





5.9 Memory Technology

This section describes the technology inside the memory chips.

Two measures—access time and cycle time.

- **Access time** ----- time between when a read is requested and when the desired word arrives,
- **Cycle time** ----- minimum time between requests to memory.
- One reason that cycle time is **greater than** access time is that the memory needs the address lines to be stable between accesses.





Main Memory Background

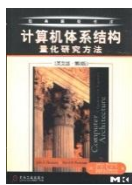
- Random Access Memory (vs. Serial Access Memory)
- Different flavors at different levels
 - Physical Makeup (CMOS, DRAM)
 - Low Level Architectures (FPM, EDO, BEDO, SDRAM)
- Cache uses **SRAM**: Static Random Access Memory
 - No refresh (6 transistors/bit vs. 1 transistor)
 - Size*: DRAM/SRAM 4-8,
 - Cost/Cycle time*: SRAM/DRAM 8-16
- Main Memory is **DRAM**: Dynamic Random Access Memory
 - Dynamic since needs to be **refreshed** periodically (8 ms, 1% time)
 - Addresses divided into 2 halves (Memory as a 2D matrix):
 - » **RAS** or **Row Access Strobe**
 - » **CAS** or **Column Access Strobe**



Embedded Processor Memory Technology: ROM and Flash

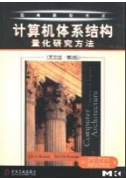
- Embedded computers usually have small memories, and most do not have a disk to act as non-volatile storage.
- Two memory technologies are found in embedded computers to address this problem.
 - The first is *Read-Only Memory (ROM)*.
 - » ROM is programmed at time of manufacture, nothing the computer can do can modify the contents of this memory.
 - The second *Flash memory*.
 - » Allows the embedded device to alter nonvolatile memory after the system is manufactured,
 - » Allows reading at almost DRAM speeds but writing flash is 10 to 100 times slower.





- ROM (掩膜式)
- PROM(熔丝或二极管)
- EPROM
- EEPROM
- FLASH MEMORY





5.10 virtual Memory

What is **virtual memory**?

- Technique that allows execution of a program that
 - » can reside in discontinuous memory locations
 - » does not have to completely reside in memory
- Allows the computer to "fake" a program into believing that its
 - » memory is contiguous
 - » memory space is larger than physical memory,
Provides illusion of very large memory

Why is VM **important**?

- Cheap - no longer have to buy lots of RAM
- Removes burden of memory resource management from the programmer
- Enables multiprogramming, time-sharing, protection





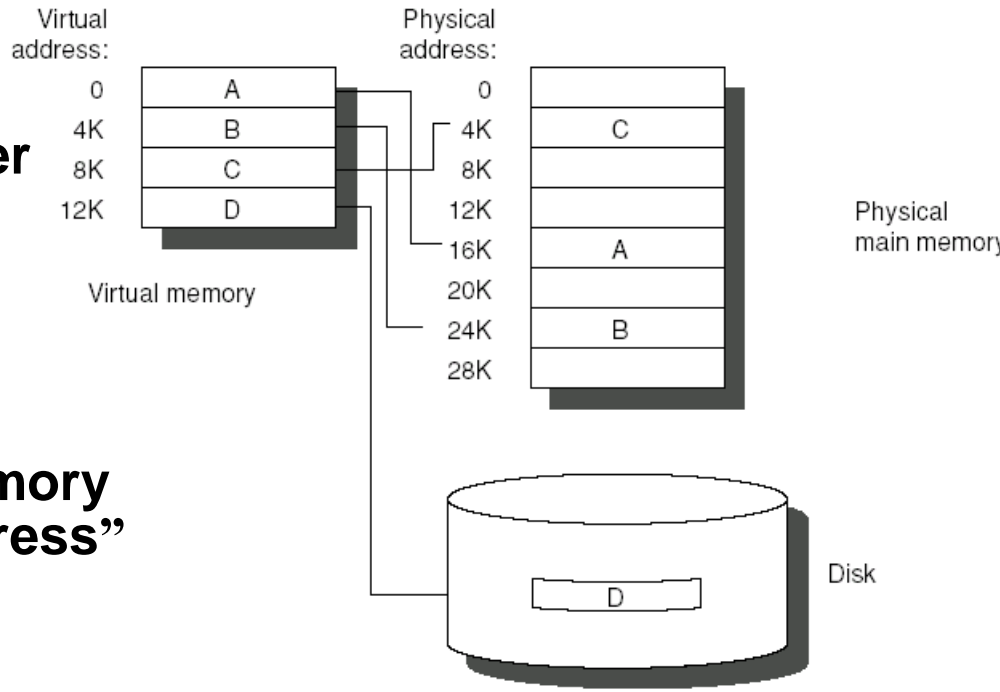
Advantages

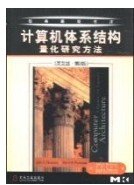
- Main memory (physical memory) can act as a *cache* for the secondary storage (disk)
- illusion of having more and contiguous physical memory
- program relocation by "pages" or "segment"
- protection in multiprogramming

Virtual Address :
address used by the programmer

Virtual Address Space:
collection of such addresses

Memory Address:
address of word in physical memory
also known as "physical address"
or "real address"





How Does VM Work

Two memory "spaces"

- **Virtual memory space** - what the program "sees"
- **Physical memory space** - what the program runs in (size of RAM)

On program startup

- OS copies program into RAM
- If there is not enough RAM, OS stops copying program & starts running the program with some portion of the program loaded in RAM
- When the program touches a part of the program not in physical memory, OS copies that part of the program from disk into RAM
- In order to copy some of the program from disk to RAM, OS must evict parts of the program already in RAM
 - OS copies the evicted parts of the program back to disk if the pages are dirty (ie, if they have been written into, and changed)



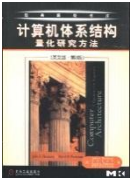
Memory Hierarchy Parameters for Virtual Memory.

Terms are different

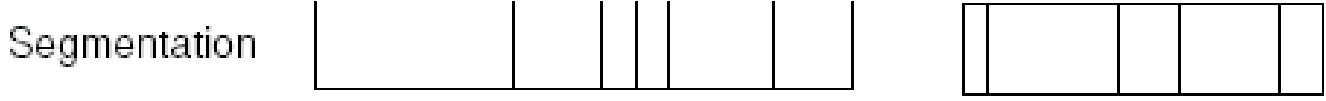
Parameter	First-level cache	Virtual memory
Block (page) size	16-128 bytes	4096-65,536 bytes
Hit time	1-3 clock cycles	50-150 clock cycles
Miss penalty (Access time)	8-150 clock cycles (6-130 clock cycles)	1,000,000-10,000,000 clock cycles (800,000-8,000,000 clock cycles)
(Transfer time)	(2-20 clock cycles)	(200,000-2,000,000 clock cycles)
Miss rate	0.1-10%	0.00001- 0.001%
Address mapping	25- 45 bit physical address to 14- 20 bit cache address	32-64 bit virtual address to 25-45 bit physical address

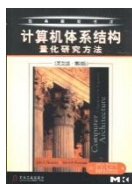


Paging versus segmentation



	Page	Segment
Words per address	One	Two (segment and offset)
Programmer visible?	Invisible to application programmer	May be visible to application programmer
Replacing a block	Trivial (all blocks are the same size)	Hard (must find contiguous, variable-size, unused portion of main memory)
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused pieces of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments may transfer just a few bytes)





Four Memory Hierarchy Questions Revisited

Q1: Where can a block be placed in main memory?

- **The high miss penalty**
 - Quite high
 - » access to a rotating magnetic storage device and
- **Must be lower miss rates**
 - choosing a simpler placement algorithm
 - operating systems designers normally pick lower miss rates because of the exorbitant miss penalty.
- **Fully associative strategy.**
 - Thus, operating systems allow blocks to be placed anywhere in main memory.



Q2: How is a block found if it is in main memory?

Virtual address



To reduce the size table

- Apply a hashing function to the virtual address.
- The hash allows the data structure to be the length of the number of *physical pages* in main memory. Such a structure is called an *inverted page table*.

To reduce address translation time

- Computers use a cache dedicated to these address translations, called a translation look-aside buffer, or simply translation buffer.

$$\frac{2^{32}}{2^{12}} \times 2^2 = 2^{22} \text{B} = 4 \text{ MB}$$



Q3: Which block should be replaced on a virtual memory miss?

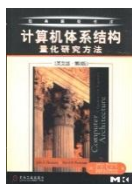
For minimizing page faults

- Almost all operating systems try to replace the least-recently used (LRU) block
- because if the past predicts the future, that is the one less likely to be needed.

Mechanism

- many processors provide a *use bit or reference bit*
 - » which is logically set whenever a page is accessed.
 - » The operating system periodically clears the use bits and later records them
 - » By keeping track in this way, the operating system can select a page that is among the least-recently referenced.





Q4: What happens on a write?

write strategy

- The level below main memory contains rotating magnetic disks that take millions of clock cycles to access.
 - Thus, the write strategy is always write back.

Dirty bit

- cost of an unnecessary access Disk is so high, it allows blocks to be written to disk only if they have been altered since being read from the disk.

