Memory Management

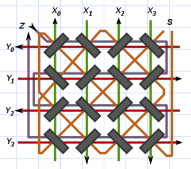
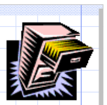
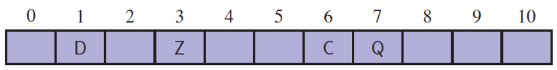


Diagram of a 4×4 plane of magnetic core memory in an X/Y line coincident-current setup.

Computer Memory



- In order to implement any data structure on an actual computer, we need to use computer memory.
- Computer memory is organized into a sequence of words, each of which typically consists of 4, 8, or 16 bytes (depending on the computer).
- These memory words are numbered from 0 to N −1, where N is the number of memory words available to the computer.
- The number associated with each memory word is known as its memory **address**.

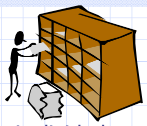


Object Creation

- With Python, all objects are stored in a pool of memory, known as the **memory heap** or Python heap (which should not be confused with the "heap" data structure).
- Consider what happens when we execute a command such as:

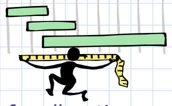
  w = Widget()
- A new instance of the class is created and stored somewhere within the memory heap.

Free List



- The storage available in the memory heap is divided into blocks, which are contiguous array-like "chunks" of memory that may be of variable or fixed sizes.
- The system must be implemented so that it can quickly allocate memory for new objects.
- One popular method is to keep contiguous "holes" of available free memory in a linked list, called the **free list**.
- Deciding how to allocate blocks of memory from the free list when a request is made is known as **memory management.**

Memory Management

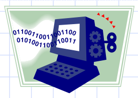


- Several heuristics have been suggested for allocating memory from the heap so as to minimize fragmentation.
  - The **best-fit algorithm** searches the entire free list to find the hole whose size is closest to the amount of memory being requested.
  - The **first-fit algorithm** searches from the beginning of the free list for the first hole that is large enough.
  - The **next-fit algorithm** is similar, in that it also searches the free list for the first hole that is large enough, but it begins its search from where it left off previously, viewing the free list as a circularly linked list.
  - The **worst-fit algorithm** searches the free list to find the largest hole of available memory.

Garbage Collection

- The process of detecting "stale" objects, deallocating the space devoted to those objects, and returning the reclaimed space to the free list is known as **garbage collection**.
- In order for a program to access an object, it must have a direct or indirect reference to that object.
  - Such objects are **live objects**.
- We refer to all live objects with direct reference (that is a variable pointing to them) as **root objects**.
- An **indirect reference** to a live object is a reference that occurs within the state of some other live object, such as a cell of a live array or field of some live object.

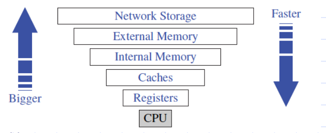
Mark-Sweep Algorithm



- In the **mark-sweep garbage collection** algorithm, we associate a "mark" bit with each object that identifies whether that object is live.
- When we determine at some point that garbage collection is needed, we suspend all other activity and clear the mark bits of all the objects currently allocated in the memory heap.
- We then trace through the active namespaces and we mark all the root objects as "live."
- We must then determine all the other live objects—the ones that are reachable from the root objects.
- To do this efficiently, we can perform **a depth-first search** (see Section 14.3.1) on the directed graph that is defined by objects reference other objects.

Memory Hierarchies

- Computers have a hierarchy of different kinds of memories, which vary in terms of their size and distance from the CPU.
- Closest to the CPU are the internal **registers**. Access to such locations is very fast, but there are relatively few such locations.
- At the second level in the hierarchy are the memory **caches**.
- At the third level in the hierarchy is the **internal memory**, which is also known as main memory or core memory.
- Another level in the hierarchy is the **external memory**, which usually consists of disks.



Virtual Memory

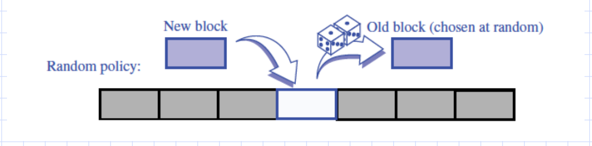
- **Virtual memory** consists of providing an address space as large as the capacity of the external memory, and of transferring data in the secondary level into the primary level when they are addressed.
  - Virtual memory does not limit the programmer to the constraint of the internal memory size.
- The concept of bringing data into primary memory is called **caching**, and it is motivated by **temporal locality**.
- By bringing data into primary memory, we are hoping that it will be accessed again soon, and we will be able to respond quickly to all the requests for this data that come in the near future.

Page Replacement Strategies

- When a new block is referenced and the space for blocks from external memory is full, we must evict an existing block.
- There are several such **page replacement** strategies, including:
  - FIFO
  - LIFO
  - Random

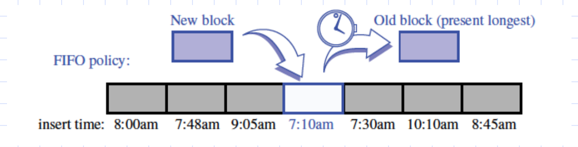
The Random Strategy

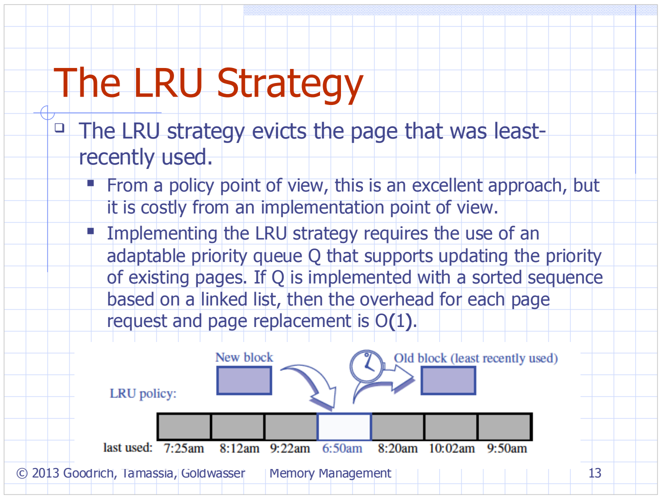
- Choose a page at random to evict from the cache.
  - The overhead involved in implementing this policy is an O(1) additional amount of work per page replacement.
  - Still, this policy makes no attempt to take advantage of any temporal locality exhibited by a user's browsing.



The FIFO Strategy

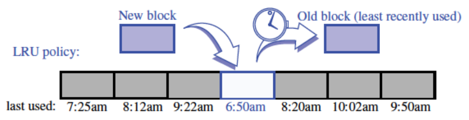
- The FIFO strategy is quite simple to implement, as it only requires a queue Q to store references to the pages in the cache.
  - Pages are enqueued in Q when they are referenced, and then are brought into the cache.
  - When a page needs to be evicted, the computer simply performs a dequeue operation on Q to determine which page to evict. Thus, this policy also requires O(1) additional work per page replacement.
  - Moreover, it tries to take some advantage of temporal locality.


New block
Old block (present longest)
FIFO policy:
insert time: 8:00am 7:48am 9:05am 7:10am 7:30am 10:10am 8:45am



The LRU Strategy

- The LRU strategy evicts the page that was least-recently used.
  - From a policy point of view, this is an excellent approach, but it is costly from an implementation point of view.
  - Implementing the LRU strategy requires the use of an adaptable priority queue Q that supports updating the priority of existing pages. If Q is implemented with a sorted sequence based on a linked list, then the overhead for each page request and page replacement is O(1).


New block
Old block (least recently used)
LRU policy:
last used:
7:25am
8:12am
9:22am
6:50am
8:20am
10:02am
9:50am