Baker's Treadmill collector

A non-moving collector

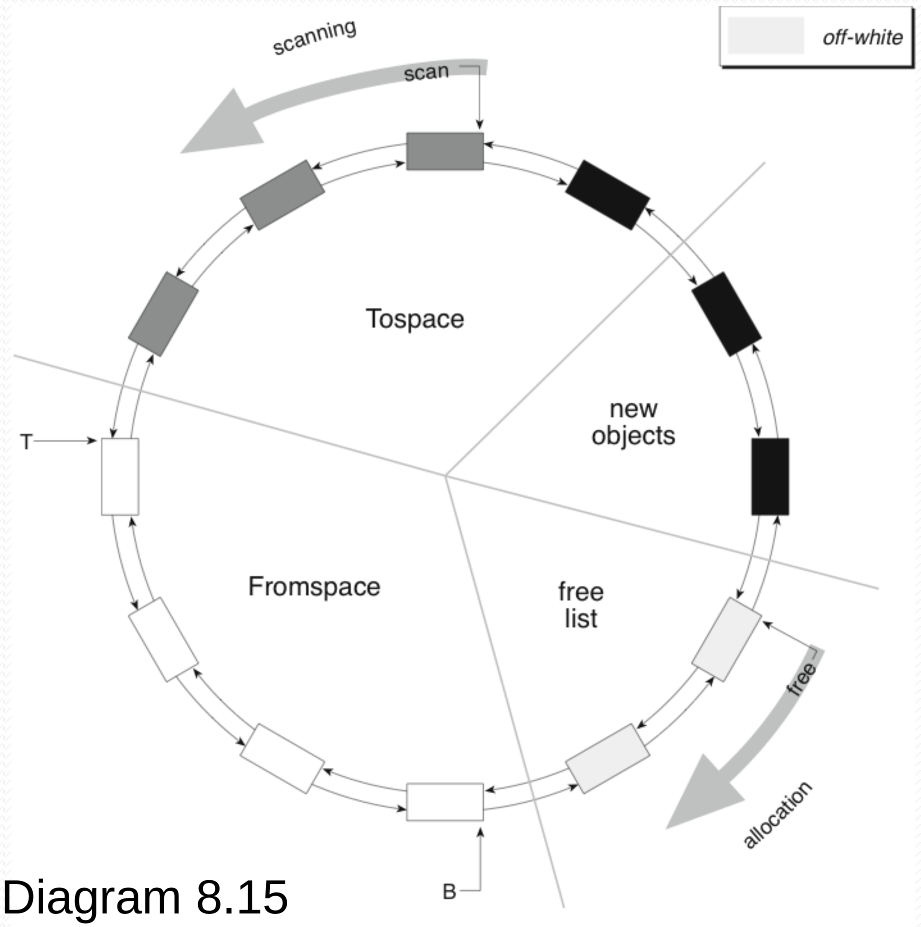
Organization of heap memory in GC

- Heap memory falls in 4 sets in a GC world
  - Scanned objects
  - Visited but unscanned objects
  - Objects not yet visited
  - Free space
- Semi-space copying collector attempts to implement these spaces
- Baker's treadmill collector offers another arrangement of these sets in a non-moving collector

Advantages of non-moving collector

- Better suited for uncooperative environments
- Mutator does not need to be protected from changes made by collector
- Collector does not move objects
  - NB: asynchronous movement may be disruptive to compiler optimization

Baker’s treadmill

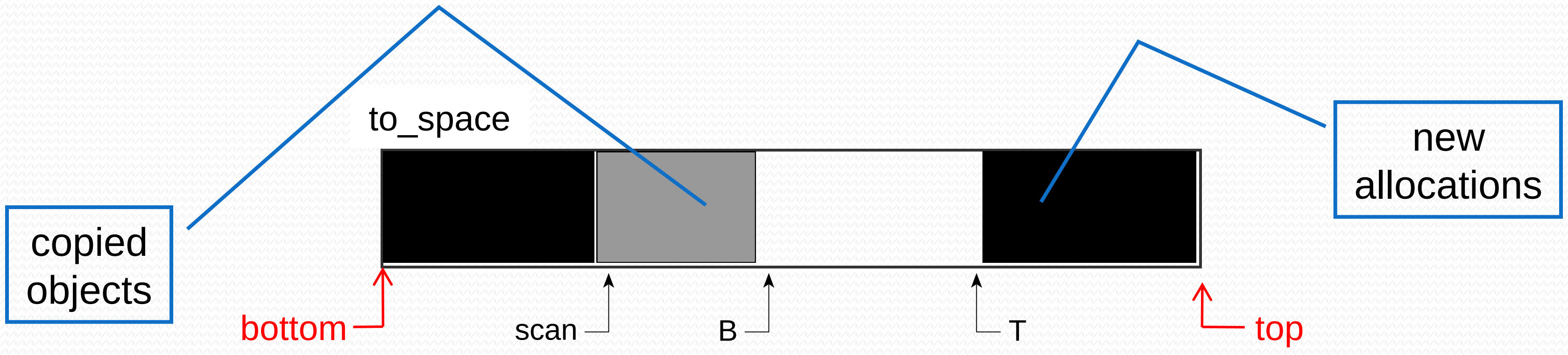


Jones and Lin: Diagram 8.15

Organization of Baker’s Treadmill

- All objects organized into cyclic doubly-linked list
  - Hence the name treadmill
- Each color segment in the list is arranged contiguously
- Fourth color, ***off-white*** used for free list
- The four segments delimited by four pointers
  - free
  - B
  - T
  - Scan
    - Similar to his incremental copying collector (see next slide)

Best known read-barrier collector



- Allocation occurs at top of to_space

Operation of Treadmill collector

- How is allocation done?
- What about marking? How is it done?
- No manipulation of color bits is necessary. Why?
- If scanned pointer refers to a black or grey object no action is required
- If object is white, what actions must be taken?

Effects of snapping

- Snapping is a constant time operation
  - Offers algorithm potential to meet real-time bounds
- Only point at which color needs to be discriminated
  - Is object white or not
- If object is snapped at T end of grey segment
  - Traversal is breadth-first
  - More page faults
- If object is snapped at scan end of grey segment
  - Traversal is depth-first
  - Not auxiliary stack needed

More on algorithm

- GC cycle is complete when no grey cells are left
    - When scan pointer meets T pointer
- Flip when free pointer meets B pointer
    - Only two colors at this point: black and white
    - Black segment → white
    - White segment → off-white
    - B and T pointers are exchanged
    - Treadmill advances its segments

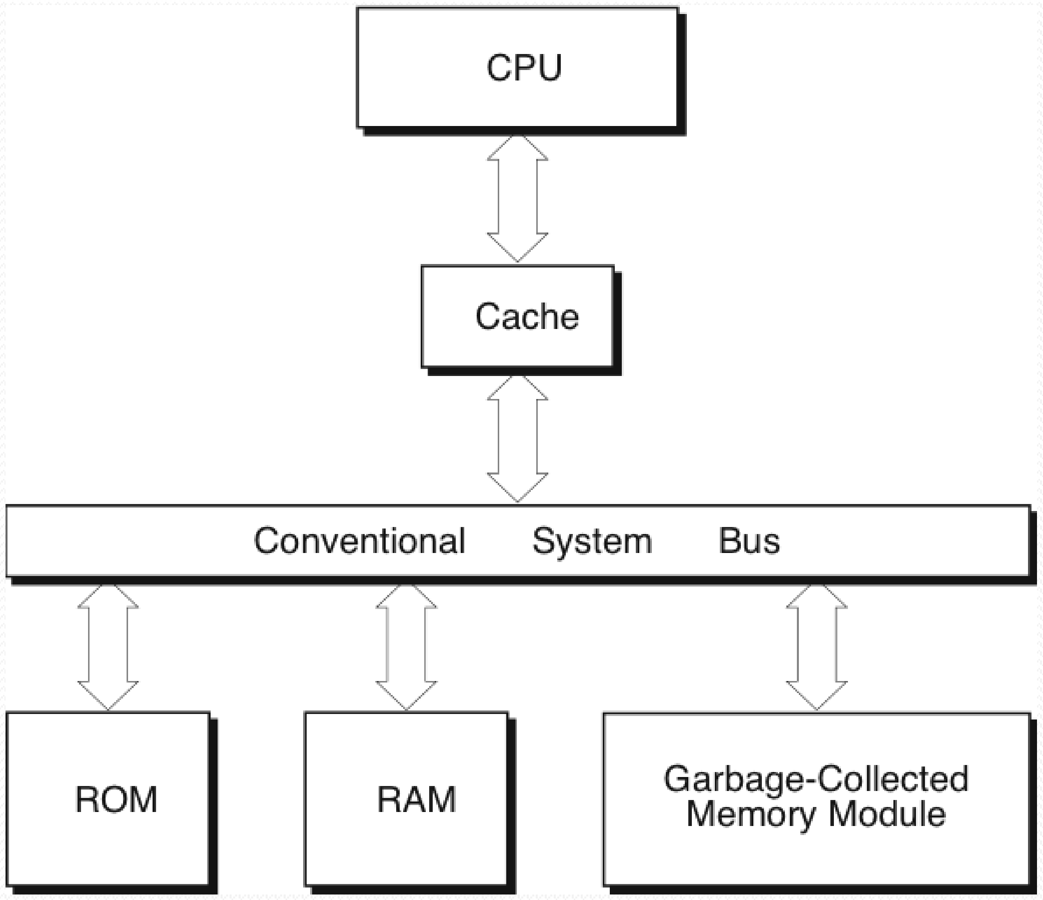
Cost of treadmill algorithm

- Expensive with regards to space compared with non-moving collectors
  - Space overhead for Links
- Memory utilization no more than copying collector
- Allocation more expensive than bumping a pointer
- Has problems with handling variable size objects
- Uses read-barrier to synchronize collector with mutator
  - Read-barriers are expensive

Hardware support for real-time GC

- No software GC has yet to demonstrate convincing hard real-time performance
  - Read-barrier techniques expensive
  - Write-barrier techniques vary in the face of virtual memory
- Nilsen and Schmidt argue that hard real-time systems must have hardware support

Nilsen’s hardware architecture



Jones and Lin: Diagram 8.15

Motivation for Nilsen’s architecture

- General purpose computers, besides supercomputers , that rely on specialized hardware have not had commercial success
- Nilsen isolates GC hardware in a special memory module that interfaces with the CPU through memory bus
- Rational: technology investment will be shared between different processor architecture