Mach

Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures,
by *R. Rashid and A. Tevanian and M. Young and D. Golub and R. Baron and D. Black and W. Bolosky and J. Chew,*
Proceedings of the 2nd International Conference on Architectural Support for Programming Languages and Operating System (ASPLOS), 1987.

1. What problem did Mach address?
2. At a high-level, what is their solution?
3. What 5 basic abstractions did Mach rely upon? What needs to be implemented efficiently for an extensible system?
4. What functionality is provided to manage an address space?
5. What 4 primary data structures are used for memory management in Mach?
6. What is the purpose of the **resident page table**? How is it organized? What info is tracked for each page?
7. Page entries from the resident page table may simultaneously be linked in 3 different lists. What are the 3 lists and what are their purposes?
8. What is the purpose of the **address map** per task address space? How is it organized? What are the advantages of this structure? How many entries does it typically contain?
9. What is the purpose of **memory objects?** What is the purpose of the reference counter for each memory object? What handles page faults for each memory object?
10. Why is efficient copy-on-write needed in Mach? How does copy-on-write work? What inefficiencies can their approach cause? How does read/write sharing work?
11. What is the purpose of the **pmap?** Why doesn't pmap need to contain every virtual to physical mapping? What is the minimal pmap?
12. How were large page tables dealt with in VAX/VMS? How does Mach on VAX deal with large page tables?
13. To put all the data structures together, describe what happens on a "page fault".
14. Conclusion?