刘宇 2014213404 19 班

8.11 Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

First-fit:
six memory partitions turned into : 300->185 600->100 350->150 200 750->392->17 125
Efficiency=(115+500+358+200+375)/(300+600+350+750)=1548/2000=0.774
Best-fit:
six memory partitions turned into : 300 600->100 350 200->10 750->392->17 125->10
Efficiency=(115+500+358+200+375)/(600+200+750+125)=1548/1675=0.924
Worst-fit:
six memory partitions turned into : 300 600->242 350->150 200 750->635->135 125
Efficiency=(115+500+358+200)/(600+750+350)=1173/1700=0.69

From the prompt above, only the worst-fit has the program to be wait. And the best-fit is more efficient that first-fit. Best Fit is the most efficient use of the memory, leaving the smallest remainder in the least amount of partitions, next comes first fit for doing the next best job, worst fit comes last because it is least efficient but also could not complete the task and ran out of space to put the last one.
Therefore, the rank will be best-fit > first-fit > worst-fit.

8.25 Consider a paging system with the page table stored in memory.
a. If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?
b. If we add TLBs, and 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 nanoseconds, if the entry is present.)

- 2*50=100ns
- EAT=(1+e)*a+(2+e)*(1-a)
  =(50+2)*0.8+(100+2)*0.2
  =62ns

8.26 Why are segmentation and paging sometimes combined into one scheme?

By combining the two, the benefits of each can be claimed. Paging prevents programs from fragmentation all over the memory space, and segmentation protects the operating system and other programs from badly written or malicious software.

Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single-segment table entry with a page-table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

In a combined paging/segmentation system a user address space is broken up into a number of segments at the discretion of the programmer. Each segment is in turn broken up into a number of fixed-size pages which are equal in length to a main memory frame. If a segment is less than a page in length, the segment occupies just one page. From the programmer`s point of view, a logical address still consists of a segment number and a segment offset. From the system`s point of view, the segment offset is viewed as a page number and page offset for a page within the specified segment.

8.28 Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0,430
b. 1,10
c. 2,500
d. 3,400
e. 4,112

a.219+430=649
b.2300+10=2310
c. Error, because 500>100
d.1327+400=1727
e. Error, because 112>96