103011227 周延儒

1. [10 points] 8.15 Consider a logical address space of 256 pages, with a 4KB page size, mapped onto a physical memory of 64 frames.
   1. How many bits are required in the logical address?

      Ans:

      2^m=256*4k=2^(8+12) -> m=20bits
   2. How many bits are required in the physical address?

      2^m=64*4k=2^(6+12)-> m=18bits
2. 8.16 Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512-MB of physical memory. How many entries are there in each of the following?
   1. A conventional single-level page table

      2 ^11 <4000 <2^12 , so we need 12 out of 32 bit logical address for the offset. Then we have 32-12=20 bits left for the page number. There are, 2^20=1048576 entries in a conventional single-level page table.
   2. An inverted page table

      2 ^11 <4000 <2^12, so we still need 12 out of 32 bit logical address for the offset. Then we have 32-12 =20 bits left for the page number and process id. There are, therefore, 2^(20/2)=1024 entries
3. [10 points] 8.20 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 : 219+430=649

b. 1, 10 : 2300+10=2310

c. 2, 500 : invalid

d. 3,400 : 1327+400=1727

e. 4, 112 : invalid

Bonus:

Well, the key idea, in my opinion, is the disadvantage of Best-Fit. To be more specific, it will generate many small spaces. And then, if we can design the testcase with the trap of making many small spaces, the Best-fit will fail. In the end, we could show First-Fit succeeding and Best-Fit and Worst-Fit fail. And, here are my details to support my claims.

First of all, we allocate the memory like this

-------|-------|---------|-----------|----------

a=2, b=1, c=9 , d=1 , e=7 (it presents the **size of memory** for each element)

That is, the symbols={'a': 0, 'b': 2, 'c': 3, 'd': 12, 'e': 13} holes=[] allocation={0: 2, 2: 1, 3: 9, 12: 1, 13: 7}

Secondly, we builds up the trap by freeing a,c,e:

-----------|-------|---------|-----------|----------

free=2, b=1 , free=9, d=1 , free=7

In other words, symbols={'b': 2, 'd': 12} holes=[(0, 2), (3, 9), (13, 7)] allocation={2: 1, 12: 1}

Lastly, we can yield the results by allocating 2,4,4,6,space in sequence. And to be more accurate, we could illustrate the sequence of the first-fit and best-fit.

In fact, there are 3 holes with 2 , 9 , 7 spaces.

Adding 2: Consume the first hole with 2 space

FirstFit symbols={'b': 2, 'd': 12, **'f': 0}** holes=[(3, 9), (13, 7)] allocation={2: 1, 12: 1, 0: 2}

BestFit symbols={'b': 2, 'd': 12, **'f': 0**} holes=[(3, 9), (13, 7)] allocation={2: 1, 12: 1, 0: 2}

Adding 4: First fit chooses the first one with 9 spaces while the best-fit choose the second one with 7 space.

-------|-------|----------------------|--------|-----------|----------

f=2, b=1 , **g from first fit = 4,free=5**, d=1 , free=7

FirstFit symbols={'b': 2, 'd': 12, 'f': 0, '**g': 3**} holes=[(7, 5), (13, 7)] allocation={2: 1, 12: 1, 0: 2, 3: 4}

-------|-------|---------|-----------|----------------------|----

f=2, b=1 , free=9, d=1 , **g from best-fit=4, free=3**

BestFit symbols={'b': 2, 'd': 12, 'f': 0, '**g': 13**} holes=[(3, 9), (17, 3)] allocation={2: 1, 12: 1, 0: 2, 13: 4}

Adding 4: First fit chooses 4 from first 5 space and the on contrary, Best-Fit chooses 4 from the first 9 space.

-------|-------|------|--------|---------|-----|----------

f=2, b=1 , g = 4,h=4, **free=1**, d=1 , free=7

FirstFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 3, '**h': 7**} holes=[(11, 1), (13, 7)] allocation={2: 1, 12: 1, 0: 2, 3: 4, 7: 4}

-------|-------|---------|----------|-----|--------|---------

f=2, b=1 , **h=4** ,free=5, d=1 , g =4, free=3

BestFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 13, **'h': 3**} holes=[(7, 5), (17, 3)] allocation={2: 1, 12: 1, 0: 2, 13: 4, 3: 4}

Adding 6: the FirstFit contains free space for 7 ,but the bestFit contains no free space with length at least 6. Besides, we can observe that both have the equal size of free spaces but differs from the size of each fragment.

-------|-------|------|--------|---------|-----|-----|-----

f=2, b=1 , g = 4,h=4, free=1, d=1 , **i=6**, free=1

FirstFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 3, 'h': 7, **'i': 13**} holes=[(11, 1), (19, 1)] allocation={2: 1, 12: 1, 0: 2, 3: 4, 7: 4, 13: 6}

-------|-------|---------|----------|-----|--------|---------

f=2, b=1 , h=4 ,free=5, d=1 , g =4, free=3

BestFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 13, 'h': 3, **'i': None**} holes=[(7, 5), (17, 3)] allocation={2: 1, 12: 1, 0: 2, 13: 4, 3: 4}

And also, the worstFit fails.

WorstFit symbols={'b': 2, 'd': 12, 'f': 3, 'g': 5, 'h': 13, **'i': None**} holes=[(0, 2), (9, 3), (17, 3)] allocation={2: 1, 12: 1, 3: 2, 5: 4, 13: 4}

In brief, the final result will be:

i=malloc(6):

FirstFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 3, 'h': 7, **'i': 13**} holes=[(11, 1), (19, 1)] allocation={2: 1, 12: 1, 0: 2, 3: 4, 7: 4, 13: 6}

BestFit symbols={'b': 2, 'd': 12, 'f': 0, 'g': 13, 'h': 3, **'i': None}** holes=[(7, 5), (17, 3)] allocation={2: 1, 12: 1, 0: 2, 13: 4, 3: 4}

WorstFit symbols={'b': 2, 'd': 12, 'f': 3, 'g': 5, 'h': 13, **'i': None}** holes=[(0, 2), (9, 3), (17, 3)] allocation={2: 1, 12: 1, 3: 2, 5: 4, 13: 4}