Lab 11 malloc lab

20200445박은하

이번 랩은 동적할당을 구현하는 것을 목표로 했다. 나는 allocator를 구현하기 위해서 segregation free list를 사용했다. Segregation free list는 size 별로 list를 만들어 관리하는데 한 줄의 list는 2의 배수 사이 만큼의 사이즈를 관리한다. 따라서

 int tempsize=asize;

    void \* list=NULL;

    while(index<SEG\_LIM\_NUM)//search for segregated list

    {

    if((tempsize<=1)&&(SEG\_LIST(index)!=NULL))//

    {

    list=SEG\_LIST(index);

와 같은 코드로 크기에 해당하는 seglist를 찾을 수 있다.

그리고 나는 allocate할때 first fit 방식을 이용했는데 요구하는 size보다 큰 block이 나올시 바로 그것을 택하는 방식이다. 이 방식은 다음과 같은 코드를 통해 구현했다.

   while((list!=NULL)&&(asize>GET\_SIZE(HDRP(list))))//first fit

    {

        list=SEG\_PREV(list);

    }

    if(list!=NULL)

    return list;

free block을 seg list에 insert할 때는 ascending order를 택했는데 만약 size가 현재 search\_ptr보다 클 경우 계속 다음과 같은 반복문을 돌린다.

 while((prev\_ptr!=NULL)&&(size>GET\_SIZE(HDRP(prev\_ptr))))

    {

    next\_ptr=prev\_ptr;

    prev\_ptr=SEG\_PREV(prev\_ptr);

    }

따라서 prev\_ptr에는 현재보다 큰 값의 size가 저장되어있고 next\_ptr에는 현재 사이즈 보다 작은 free block이다.

Allocate할 때는 first fit 방식을 사용하는데, free block은 segmentation list에서 ascending order로 정렬되어있기 때문에 best fit의 효과를 갖는다.

더 큰 최적화를 진행하기 위해 place함수의 splitting 부분에서 manipulation을 하였다.

 else if(asize>100)//asize is bigger than 100 insert small sub block first to reallocate well

    {

        PUT(HDRP(block), PACK(sub, 0));

        PUT(FTRP(block), PACK(sub, 0));

        PUT(HDRP(NEXT\_BLOCK\_PTR(block)), PACK(asize, 1));

        PUT(FTRP(NEXT\_BLOCK\_PTR(block)), PACK(asize, 1));

        insert\_block(block, sub);

        return NEXT\_BLOCK\_PTR(block);

    }

    else{

        PUT(HDRP(block), PACK(asize, ALLOCATED));

        PUT(FTRP(block), PACK(asize,ALLOCATED));

        PUT(HDRP(NEXT\_BLOCK\_PTR(block)), PACK(sub, FREE));

        PUT(FTRP(NEXT\_BLOCK\_PTR(block)), PACK(sub, FREE));

        insert\_block(NEXT\_BLOCK\_PTR(block),sub);

    }

만약 size가 클 경우 small sub block를 앞에 두고 할당하는 부분을 블록의 뒷부분에 두어 reallocation될 때 앞의 블록과 colasce가 가능하게 한다.

Size가 작을 경우 블록의 앞부분에 할당한다. Size의 경계값을 찾기 위해서 테스팅해 본 결과 90~110의 영역에서 97점에 해당하는 최고 값을 얻을 수 있었다.