

Lab3

2016-18221 이동현

1. 실행 결과

Results for mm malloc:						
trace	valid	util	ops	secs	Kops	
0	yes	97%	5694	0.000328	17360	
1	yes	96%	5848	0.000284	20592	
2	yes	97%	6648	0.000364	18254	
3	yes	99%	5380	0.000256	21057	
4	yes	79%	14400	0.000473	30450	
5	yes	93%	4800	0.001079	4448	
6	yes	91%	4800	0.000330	14563	
7	yes	63%	12000	0.000388	30912	
8	yes	54%	24000	0.004044	5935	
9	yes	80%	14401	0.000504	28579	
10	yes	82%	14401	0.000369	39027	
Total		85%	112372	0.008419	13348	

Perf index = 51 (util) + 40 (thru) = 91/100

사진 1. 실행 결과

2. 구현

Heap을 관리하기 위해 Segregated list를 사용한다. size class의 linked list를 사용해서 heap의 free list를 가리키게 된다. segregated list에서 header, prev node pointer, next node pointer, footer로 구성되는데 header와 footer는 size와 state를 가진다.

- extend_heap

size 만큼의 heap을 늘려주는 기능을 한다. heap 끝에 Epilogue header를 할당하고 seg_insert를 통해 리스트에 삽입한다. 그 후 coalesce를 통해 free block을 합쳐서 리턴한다.

- seg_insert

free list 포인터를 list에 넣어준다. 각 index는 $2^i - 2^{(i+1)}$ 를 표현하기 때문에 size에 따라 정렬하고 location을 결정해서 block을 넣어줄 수 있다.

- **seg_delete**

seg_list의 포인터를 삭제한다. 리스트의 **index**가 사이즈를 나타내기 때문에 인덱스에 노드가 한개일 경우 해당 **index** 포인터를 **NULL**로 할당한다. 한개 이상일 경우 **location**을 찾아 제거하고 포인터를 적절하게 연결해준다.

- **seg_search**

인풋 사이즈에 해당하는 **free block**을 찾아서 반환한다. **size class**를 찾아서 타겟 사이즈 이상의 **free block**을 순차적으로 검색한다.

- **place**

heap에 할당하는 동작을 한다. **heap block**에 **allocation bit**를 표시하는데 **bp**와 **block**의 사이즈가 다를 때 생기는 **remaining block**을 최소 사이즈를 비교해서 적절한 동작을 수행한다. 최소 사이즈보다 작다면 한번에 끊고 크다면 쪼개서 **seg_insert**를 수행한다.

- **coalesce**

연속된 **free list**를 합쳐주는 역할을 한다. **prev**, **next**의 할당 상태에 따라 적절하게 합쳐서 리턴해준다. 합치는 블록의 **header**와 **footer**를 적절하게 제거 후 연결하고 기록하는 동작을 한다.

- **mm_init**

SEG_LIST를 통해 **segregate list**와 **heap**을 초기화한다. 최초에 **padding**으로 공간을 잡고, **extend heap**으로 **page size**의 **heap** 공간을 확보한다. 전역 변수인 **heap_listp**으로 **header** 다음 주소를 표현한다.

- **mm_malloc**

seg_search를 통해 타겟 사이즈의 크기를 갖는 **free list**를 검색하고 결과값이 있을 경우 **place**를 호출해 **heap**에 할당한다. 없다면 **extend_heap**을 통해 **heap**을 확장하여 **malloc**을 시켜준다.

- **mm_realloc**

free후 **malloc** 동작을 한다. **input**의 **ptr**이 **NULL**인 경우 **malloc**, **size**가 0인 경우에는 **free**와 동일한 동작을 한다. **size**가 같은 경우 포인터를 그대로 리턴하고, **size**가 작은 경우 **remaining block**을 **seg_insert** 한다. **size**가 큰 경우 다음 블록이 **free**라면 합쳤을 때 **size**를 만족하는지 확인하고, 아니라면 **malloc**을 수행한다.

3. 어려웠던 점

함수를 동작 별로 쪼개고 설계하기 위해 처음 구조를 생각할 때 어려웠다. 시행착오를 거치며 구성했는데, **pointer**를 연쇄적으로 많이 사용하기 때문에 **segfault** 디버깅이 어려웠다.

4. 새롭게 배운 점

mm_realloc에서 필요한 **size**가 현재 **size**보다 클 경우 **size_list**에서 검색 후 합쳐서 공간을 만드는 방법과 새로 **malloc**을 하는 방법 중 **space/time trade off**를 고민해 보며 경험을 쌓을

수 있었다. 또한 **#define**을 이용하여 코드를 효율적으로 작성하는 것에 더욱 익숙해질 수 있었다.