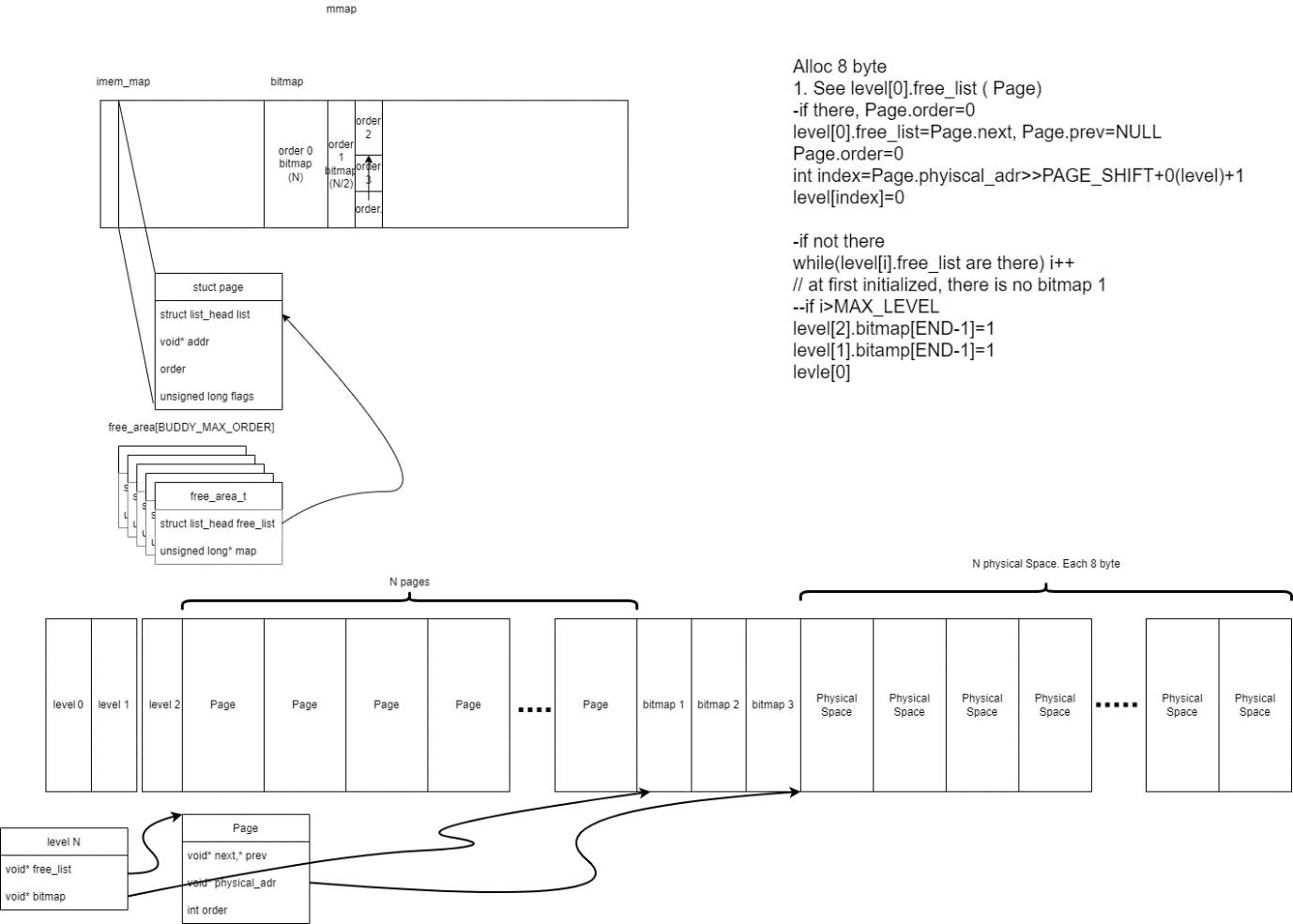
Mymalloc project

20180546

1. 초기 계획 : Buddy System



처음에는 리눅스에서 실제로 프로세스에게 물리 메모리 페이지를 할당해줄 때 사용하는 buddy 시스템 형태로 구현하려고 했다. Buddy 시스템이란 같은 사이즈의 페이지를 하나의 비트맵으로 관리한다

(Page Size << N)의 크기의 페이지를 할당 받고 싶다면 Level N의 할당된 페이지를 먼저 확인한다. 만약 Level N에 적합한 크기의 페이지가 없다면 N+1 ~ MAX\_ORDER까지의 LEVEL을 탐색한 뒤 ,페이지를 divide 해주고 level N ~ 할당된 LEVEL까지 관련된 bitmap들을 모두 toggle한다.

페이지를 할당 해제 하고 싶다면 해당 페이지의 해당하는 bitmap을 확인하고 1인 경우에 merge/리스트 삭제 , 다시 상위 ORDER의 연관되는 페이지의 bitmap을 확인한 후 1이면 merge/리스트 삭제,0이면 거기서 break 하는 식으로 반복문을 통해 페이지를 병합/ 리스트 삭제를 한다

전체적인 메모리 사용 구조는 global 변수로 free\_area[MAX\_ORDER] 구조체 배열를 저장하고 나머지 page 구조체, bitmap, 실제 물리메모리 주소공간은 남는 공간에 배열하는 형태였다.

Free\_area 배열 // Page 구조체 배열 // bitmap // 물리메모리 공간으로 나누어지는데,

struct free\_area size :24 byte

struct page size : 40 byte

가장 최적화된 Page 크기와 물리메모리 공간 크기를 구하려고 하다 보니..

텍스트, 화이트보드이(가) 표시된 사진

자동 생성된 설명

위 형태의 비선형방정식이 도출되었다

그리고 계산결과 가장 본 메모리 과제 최대 한계치인 20MB 내에서 최적의 형태는 Page 크기 : 256Byte , 물리 메모리 공간 크기 : 2^24 Byte 였다

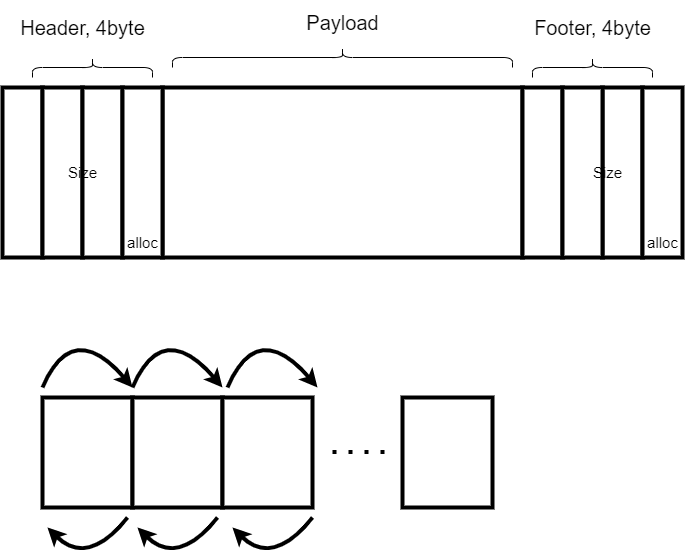
두 파라미터로 결정된 공간은 아래와 같다

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Free\_area | Page | Bitmap | Physical area |
| Size | 24\*16 | 40\*(2^16) | 8192 | 2^24 |
|  | 384 | 2621440 | 8192 | 16777216 |

그러나, 해당 buddy 시스템으로 구현한 allocator는 mm\_free를 구현하고 보니 해당 프로젝트의 평가 indicator에 부합하지 못하였다. Buddy는 초기에 2621440의 page 구조체를 할당해줘야하고, 해당 page의 물리주소도 physical area에 매핑해주기 때문에 처음부터 끝까지 mem\_brk가 384+2621440+8192+16777으로 고정되어있다.

1. Next fit

그래서 그냥 CSAPP에 있는 next fit으로 구현했다



메모리 할당 –

마지막 할당된 block의 포인터 혹은 free 된 포인터 지점부터 메모리 allocation을 시작한다. 할당이 되어있지않고, 사이즈가 충분하다면 그곳을 할당해준다

메모리 해제 –

이전 블록/다음 블록의 할당여부를 확인한 후 HEADER와 FOOTER의 할당여부, SIZE 값들을 바꿔준다.

메모리 재할당 –

재할당 사이즈가 더 작으면 변화 없이 그대로 리턴해준다

재할당 사이즈가 더 크다면

다음 블록이 할당되어있지 않고 재할당 사이즈가 다음 블록 크기와 현재 블록의 합보다 더 작다면 바로 그곳에 할당하고

아니면 새로운 곳을 할당하고 원래의 포인터를 해제한다

결과 값 :

