Multicore Programming

Project 3

학번 : 20191617

전공 : 컴퓨터공학

이름 : 이규호

1. 개발 목표

- C program에서 작동하는 dynamic memory allocator를 구현한다. malloc, free, realloc 함수가 작동해야하고, 설계 공간을 정확하고 효율적으로 탐색하여 효율적인 allocator를 구현해야 한다.

2. 개발 범위 및 내용

A. 개발 범위

1. mm_init

malloc을 위한 초기화를 진행한다. 아래의 3가지 함수(mm_malloc, mm_free, mm_realloc)을 호출하기 전에, 초기 heap 영역 할당, 초기 가용블럭 생성 등을 수행한다. 초기화 수행 여부에 따라 -1 혹은 0 을 return한다.

2. mm_malloc

최소 크기의 free block에 size만큼 할당한 후, block의 payload pointer를 반환한다. 할당된 모든 block들은 heap 영역 내부에 존재해야하고 다른 할당된 chunk와 겹치지 않아야한다.

3. mm_free

mm_free는 할당된 공간을 가르키는 ptr에 대하여 block을 해제한다. return값이 없고 allocated된 ptr에 대해서만 작동한다.

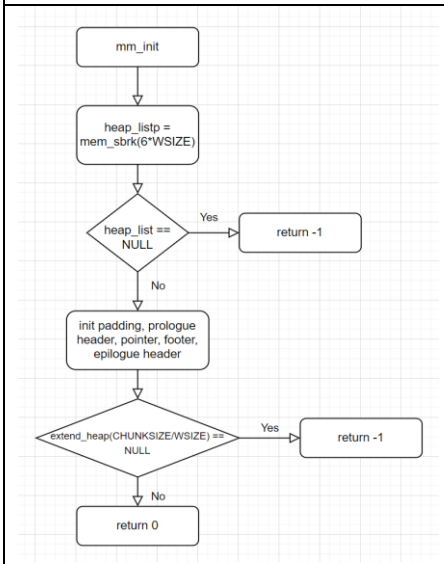
4. mm_realloc

인자로 넘어온 ptr에 대하여 재할당을 한다. ptr이 null일 경우, mm_malloc과 동일한 시행을 한다. size가 0일 경우, mm_free와 동일한 시행을 한다. 그 외의 경우 ptr이 point하는 block의 크기를 변경하여 새 block 주소를 반환한다.

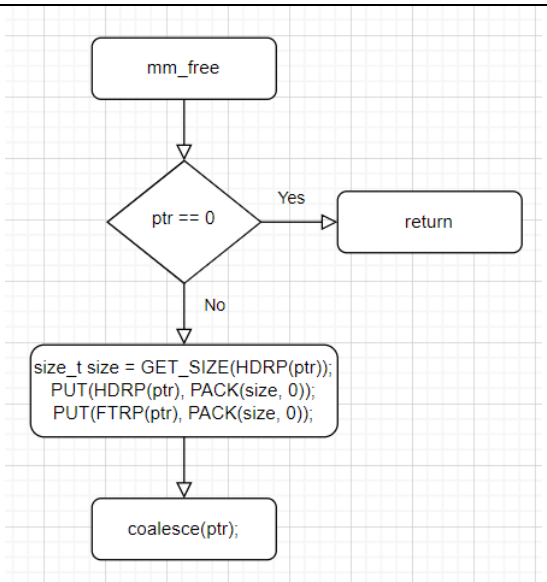
B. 개발 내용 및 방법

-flowchart

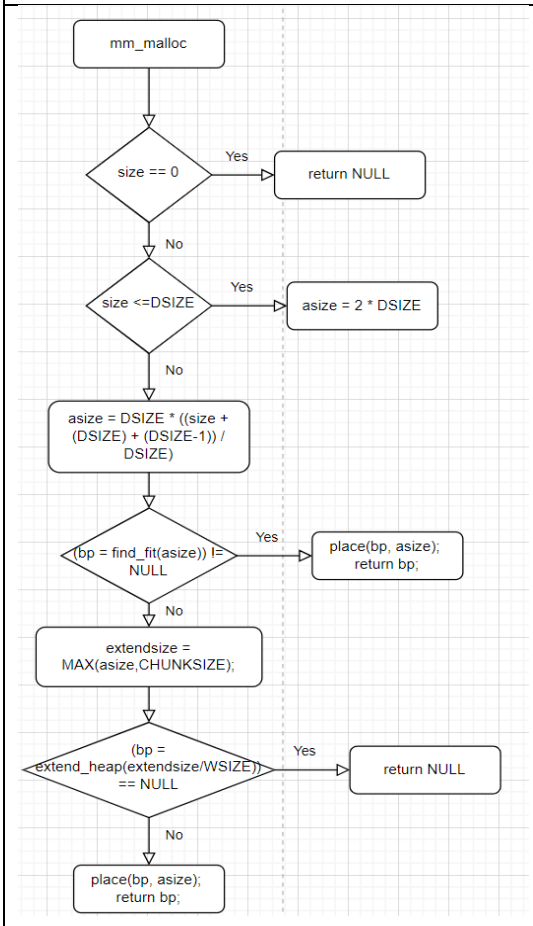
* mm_init



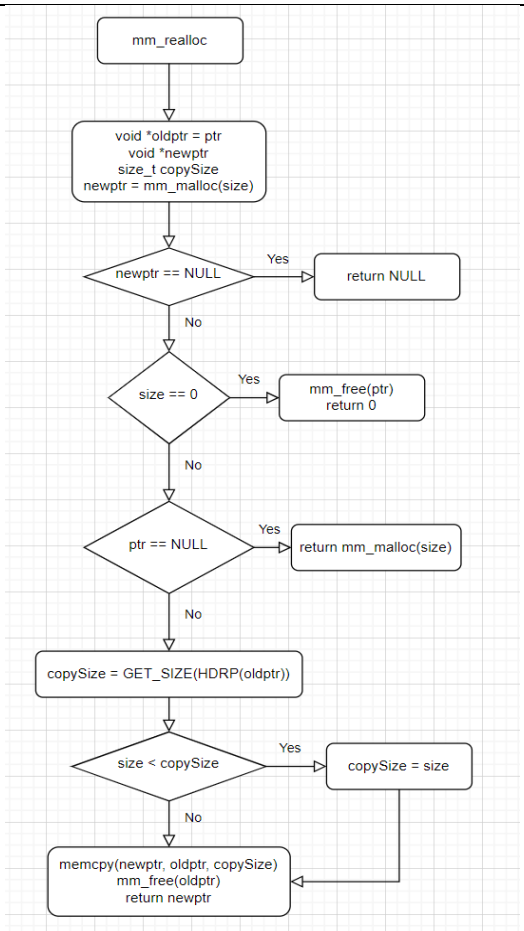
* mm_free



* mm_malloc



* mm_realloc



- Macro

WSIZE 4 : word size를 4로 설정

DSIZE 8 : double word size를 8로 설정

CHUNKSIZE (1<<12) : 초기 init에서 heap 확장을 위한 기본 크기를 선언 (4096 byte)

MAX(x,y) ((x) > (y) ? (x):(y)) : MAX 함수 선언

PACK(size,alloc) ((size) | (alloc)) : size와 alloc을 인자로 넘겨 header, footer를 위한 인자를 Packing

GET(p) (*(unsigned int*)(p)) : 인자 p가 참조하는 word 값을 읽어서 return한다. p는 (void *)이기 때문에 형변환한다.

PUT(p,val) (*(unsigned int*)(p) =(val)) : 인자 p가 point하는 word에 val을 저장한다.

GET_SIZE(p) (GET(p) & ~0x7) : 주소 p에 있는 header 혹은 footer의 size를 return한다.

GET_ALLOC(p) (GET(p) & 0x1) : 주소 p에 있는 header 혹은 footer의 alloc 비트를 return한다.

HDRP(bp) ((char*)(bp) - WSIZE) : block의 header를 가르키는 pointer를 return한다.

FTRP(bp) ((char*)(bp) + GET_SIZE(HDRP(bp)) - DSIZE) : block의 header를 가르키는 pointer를 return 한다.

NEXT_BLKP(bp) ((char*)(bp) + GET_SIZE((char*)(bp)-WSIZE)) : 다음 block의 block pointer를 return 한다. 현재 block에 해당하는 block size만큼 움직인다.

PREV_BLKP(bp) ((char*)(bp) - GET_SIZE((char*)(bp)-DSIZE)) : 이전 block의 block pointer를 return한다. 이전 block에 해당하는 footer를 통해 block size만큼 앞으로 움직인다.

NEXT_FREE_PTR(bp) (*(void**)(bp+WSIZE)) : 다음 free block의 bp pointer를 return한다.

PREV_FREE_PTR(bp) (*(void**)(bp)) : 이전 free block의 bp pointer를 return한다.

- Global variables

static char *heap_listp = 0 : 첫번째 block의 pointer이다.

static char *free_listp = 0 : free list를 구현하기 위한 pointer이다.

- **Functions**

int mm_init(void) : malloc을 위한 초기화를 진행한다. 먼저 6*WSIZE size만큼 heap을 할당한다. 이는 padding, 초기 값을 가지는 prologue header와 footer, 그리고 그 사이에 prologue에 해당하는

prev, next free pointer, 그리고 마지막으로 마지막을 뜻하는 epilogue header를 위한 word를 할당하는 것이다. 그 후, 이에 해당하는 값을 넣고 free_listp의 값 또한 prologue를 가르키도록 설정한다. 그후 extend_heap 함수에 CHUNKSIZE/WSIZE 값을 넘겨 초기 free block을 생성하여 0을 return 하도록하고, 실패할 경우 -1을 return 한다.

void *mm_malloc(size_t size) : malloc에 해당하는 함수이다. 먼저 size를 parameter로 받아 0일 경우 NULL을 return한다. 이후 요청한 값을 size가 DSIZE보다 작을 경우 2*DSIZE 로 설정하고, 아닐 경우 DSIZE의 배수로 설정한다. 이후 find_fit 함수를 통해 free list에서 알맞는 free block을 찾은 후, place함수를 통해 할당하였다. 만약 알맞는 free block을 찾지 못한 경우, heap size를 extend_heap 함수로 연장한다. 이 함수는 할당된 bp를 return한다.

void mm_free(void *ptr) : parameter로 전달받은 ptr이 0일 경우 return 하고, 아닐 경우 free를 진행한다. 먼저 ptr의 header와 footer의 allocate bit를 0으로 설정하고 coalesce를 호출한다.

void *mm_realloc(void *ptr, size_t size) : realloc에 해당하는 함수이다. oldptr와 newptr을 선언하여 newptr에 parameter로 받은 size만큼 할당하고, newptr이 NULL인 경우 NULL을 return한다. size가 0 인 경우에는 free와 같은 역할을 하기 때문에 mm_free를 호출하고 0을 return한다. 기존에 존재하던 block의 size를 copySize에 저장하고, realloc으로 받은 사이즈와 비교하여 size가 작을 경우 copySize를 기존 size로 줄여 newptr에 할당한 후, return 한다.

static void *extend_heap(size_t words) : parameter로 요청한 size에 대해 추가적인 heap 공간을 연장한다. 새로운 heap 공간 연장 후 새 공간에 대한 initialize 또한 진행한다.

static void place(void *bp, size_t asize) : bp의 header size만큼 block을 할당한다. 할당 후 남은 공간이 2* DSIZE일 경우, 나머지 공간을 free공간으로 바꾸고, 그렇지 않은 경우엔 할당만 수행한다. free block이 된 부분 free list에 insert한다.

static void *find_fit(size_t asize) : free_listp를 순회하며 parameter로 전달받은 size에 맞는 free block을 찾고 반환한다. 맞는 block이 없을 시 NULL을 return 한다.

static void *coalesce(void *bp) : 4가지 경우에 따라서 free된 block을 합친다. 현재 free 시키는 block에서 block의 앞, 뒤의 할당 여부를 확인하여 block을 병합한 후 free list에 insert한다.

static void insert_free(void *bp) : parameter로 받은 bp를 free list에 삽입한다.

static void remove_free(void *bp) : free list에서 block을 삭제한다. parameter로 받은 bp가 free list의 처음일 경우 다음 free block의 prev pointer를 NULL로 만들고 free list pointer를 수정한다. 처음이 아닐 경우 앞뒤 free block과 이어준다.

3. 구현 결과

```
cse20191617@cspro:~/test$ ./mdriver -V
[20191617]::NAME: Kyuho Lee, Email Address: rbgh0114@sogang.ac.kr
Using default tracefiles in ./tracefiles/
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: amptjp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cccp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cp-decl-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: expr-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: coalescing-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   89%    5694  0.000405 14045
 1       yes   92%    5848  0.000271 21555
 2       yes   95%    6648  0.000604 11003
 3       yes   96%    5380  0.000449 11982
 4       yes   88%   14400  0.000309 46527
 5       yes   88%    4800  0.000854  5620
 6       yes   85%    4800  0.000818  5869
 7       yes   55%   12000  0.023504   511
 8       yes   51%   24000  0.015737  1525
 9       yes   26%   14401  0.113180   127
10       yes   42%   14401  0.004730  3044
Total          73%  112372  0.160864   699

Perf index = 44 (util) + 40 (thru) = 84/100
```

주어진 mdriver(test driver program)를 이용하여 성능을 평가하였다.