2021 Fall System Programming

Malloc Lab 2

2021. 11. 29

유주원
clearlyhunch@gmail.com

Embedded System Lab.
Dept. of Computer Science & Engineering
Chungnam National University

CNU Embedded System Laboratory

실습 소개

- 과목 홈페이지
  - 충남대학교 사이버캠퍼스 (http://e-learn.cnu.ac.kr/)
- 연락처
  - 유주원
  - 공대 5호관 533호 임베디드 시스템 연구실
  - clearlyhunch@gmail.com
    - Email 제목은 '**[시스템프로그래밍][01][학번_이름]용건**' 으로 시작하도록 작성

CNU Embedded System Laboratory

공지사항

- Implicit 관련
  - 미리 정의된 매크로, define 값 수정 금지
    - ex. Chunksize 등 수정금지 (알고리즘만 수정해서 성능 향상)
- 구현 배점 implicit : explicit = 50 : 20

- Copy 관련 전부 찾아서 0점 처리하겠습니다.
  (이때까지 과제 포함, explicit 카피 제출시 implcit도 0점 처리)

개요

- 실습 명
  - Malloc Lab
- 목표
  - Dynamic Allocator를 구현
- 구현사항
  - malloc, realloc, free를 구현
  - 다양한 알고리즘을 적용하여 높은 성능을 갖도록 함

Malloc Lab

- Malloc Lab에서는 C언어로 dynamic storage allocator를 구현하게 된다.
- 이를 통해 각자의 malloc, free, realloc, calloc 알고리즘을 구현해야 한다.
- 또한, 메모리 공간의 생성 디자인을 고려하면서 정확하고 효율이 좋은 빠른 allocator를 구현해야 한다.
- Malloc Lab은 총 4가지의 방식으로 진행된다.
  - Naive
  - Implicit
  - Explicit
  - Segregated
    - Segregated는 실습 일정상 진행하지 않음

Malloc Lab – Free Block 추가 알고리즘

- Explicit 방식은 Free block들을 list 형식으로 관리하는 방법이다. 따라서 Free block들이 새로 추가되는 경우, 이 block을 어느 위치에 추가하는지에 따라 성능이 좌우된다.

- Free block을 추가하기 위한 알고리즘은 다음과 같다.
  - LIFO(Last In First Out) 정책
    - Free Block을 List의 맨 앞에 끼워 넣는 방법
    - 장점: 간단하고, 상수 시간 소요
    - 단점: 단편화가 Address-ordered 정책보다 나쁨
  - 주소 정렬(Address-ordered) 정책
    - Free Block List의 block 순서를 유지하면서 삽입
      - ex) addr(pred) < addr(curr) < addr(next)
    - 장점: LIFO 정책보다 단편화 성능이 우수
    - 단점: List 탐색 후 삽입해야 함

CNU Embedded System Laboratory

Malloc Lab – 설치 및 빌드

- Malloc Lab은 각 allocation 방식에 맞게 설정 후 빌드를 해주어야 한다.
  - 아래는 각 allocation 방식 설정 방법이다.

Allocation 방식	make 명령어
Naive	make clean; make naive; make
Implicit	make clean; make implicit; make
Explicit	make clean; make explicit; make

- 설정 후에 'make'명령을 입력하여 컴파일 한다.

- make 명령을 통해 각 방식 별로 컴파일 하기 이전에 'make clean' 명령을 통해 이전에 컴파일 된 파일들을 제거하고 수행해준다.

CNU Embedded System Laboratory

Malloc Lab – 설치 및 빌드

- Allocation 방식 설정 및 빌드

```
sys02@host-192-168-0-5:~/test/malloclab-handout$ make explicit
rm -f mm.c mm.o; ln -s mm-explicit.c mm.c
sys02@host-192-168-0-5:~/test/malloclab-handout$ make
gcc -Wall -O2 -g -DDRIVER   -c -o mdriver.o mdriver.c
```

- 빌드 결과

```
sys02@host-192-168-0-5:~/test/malloclab-handout$ ls
clock.c   fcyc.o    ftimer.o       memlib.c       mm-implicit.c  traces
clock.h   fsecs.c   Makefile       memlib.h       mm-naive.c
clock.o   fsecs.h   malloclab.pdf  memlib.o       mm.o
config.h  fsecs.o   mdriver        mm.c           mm-orig.c
fcyc.c    ftimer.c  mdriver.c      mm-explicit.c  mm-seglist.c
fcyc.h    ftimer.h  mdriver.o      mm.h           README
sys02@host-192-168-0-5:~/test/malloclab-handout$
```

CNU Embedded System Laboratory

Malloc Lab – 필수 사항

- 각 방식 별, 소스파일 상단에 학번과 이름을 입력한다.
  - mm-explicit.c

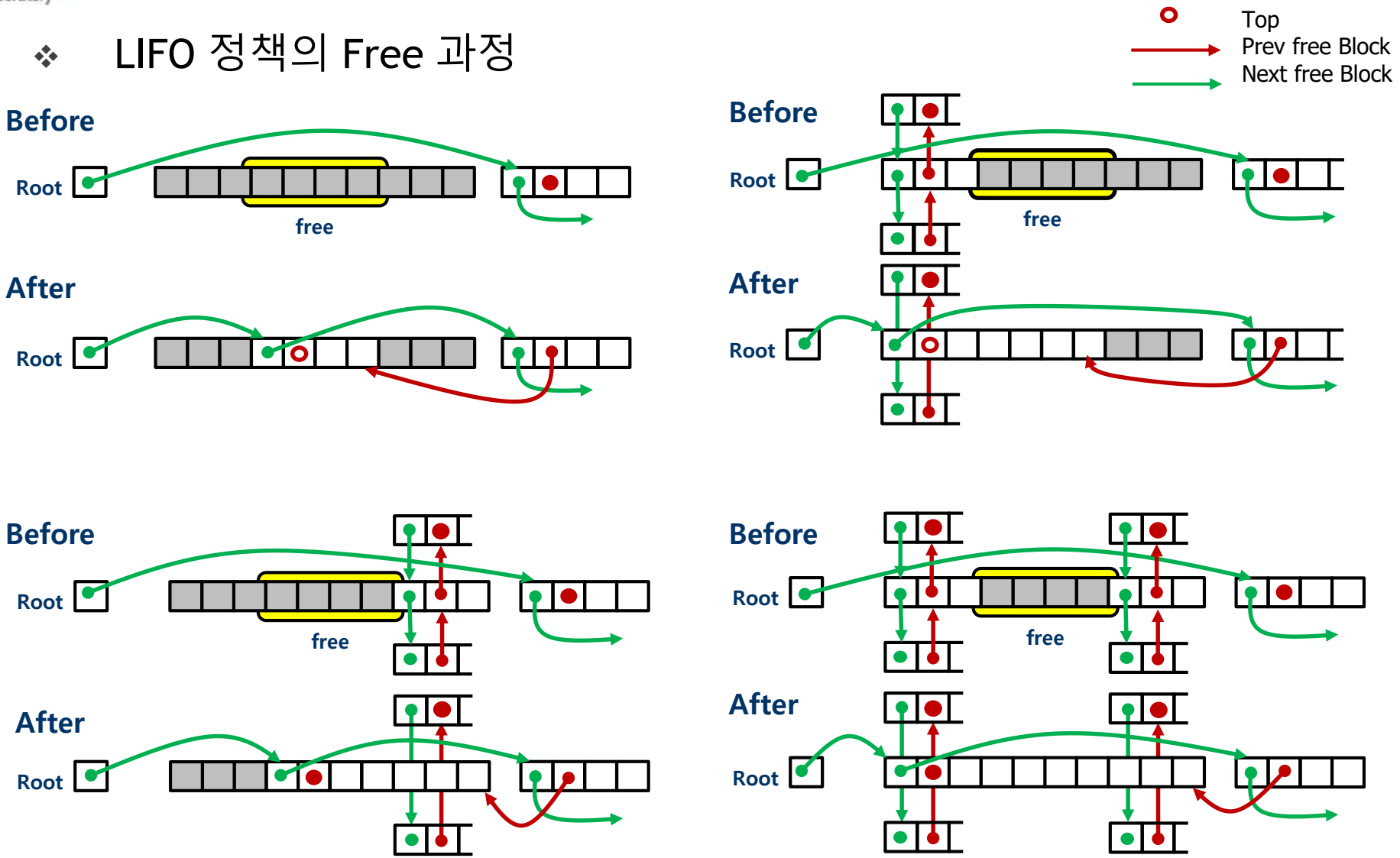
```
/*
 * mm-explicit.c - an empty malloc package
 *
 * NOTE TO STUDENTS: Replace this header comment with your own header
 * comment that gives a high level description of your solution.
 *
 * @id : 학 번
 * @name : 이 름
 */
```

- 해당 파일 외에는 절대 수정을 하지 않는다.(mm.h 등)
- 외부 메모리 관리 라이브러리 및 시스템 콜 사용 불가.
- 배열, 구조체, 트리, 리스트 같은 전역 자료 구조체의 선언 금지.
  - 공간이 많이 필요한 경우 mem_sbrk 활용

CNU Embedded System Laboratory

(참고) Malloc Lab – Free Block 추가 알고리즘

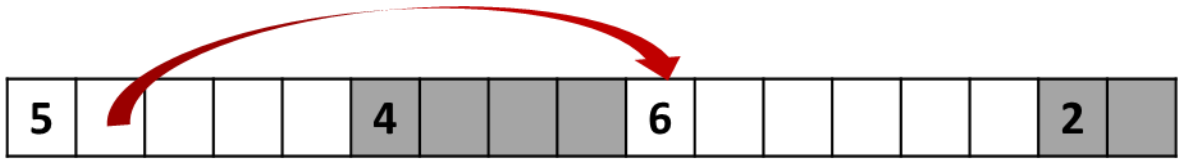
❖ LIFO 정책의 Free 과정



CNU Embedded System Laboratory

(참고) Malloc Lab – Explicit

- Explicit list 방식은 Free block에 저장된 있는 pointer를 이용하여 Free block들을 연결하고, 연결된 모든 Free block들을 탐색한다.

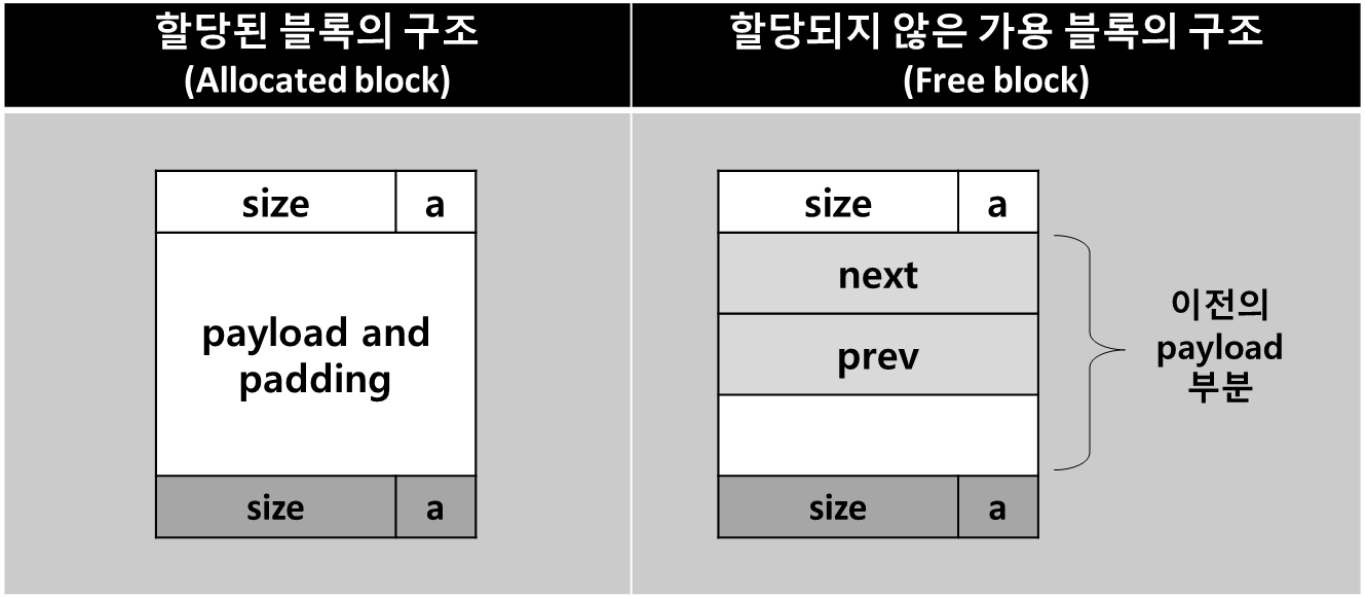


- Free block만 탐색하므로 높은 throughput을 보인다.

CNU Embedded System Laboratory

(참고) Malloc Lab – Explicit

❖ 자료구조

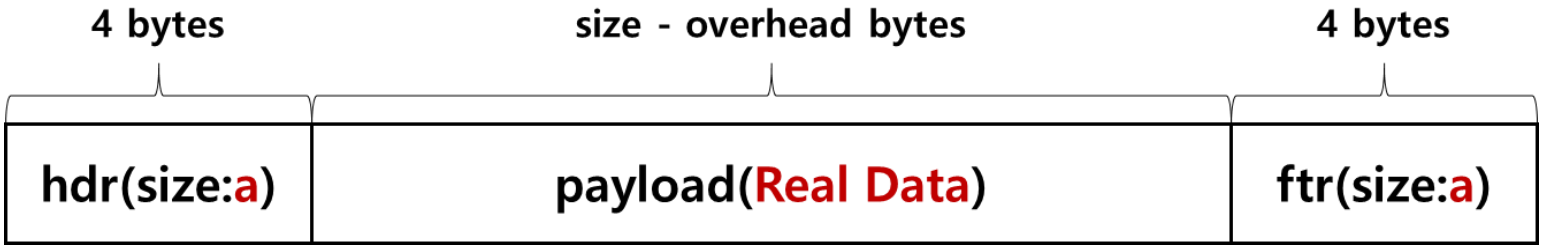


- 할당된 블록의 경우, implicit에서 사용했던 구조를 그대로 이용
- Free 블록의 경우, 다음 Free 블록을 가리키는 next와 이전 Free 블록을 가리키는 prev를 이전 payload 영역에 가짐

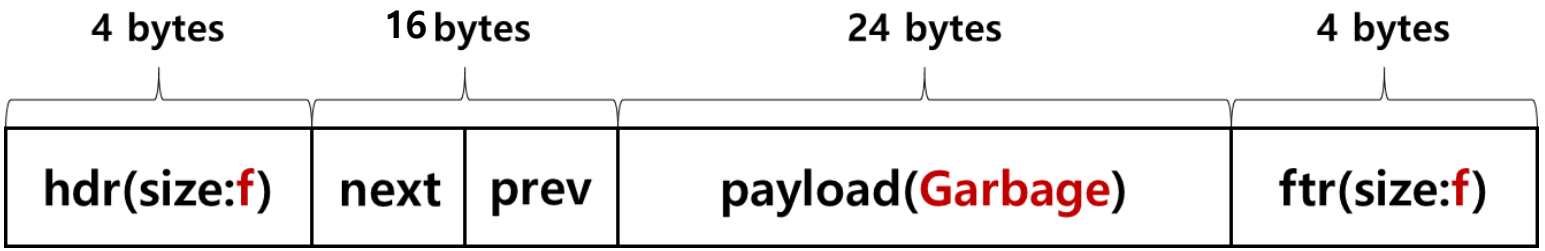
CNU Embedded System Laboratory

(참고) Malloc Lab – Explicit

- 자료구조
  - <Allocated block>



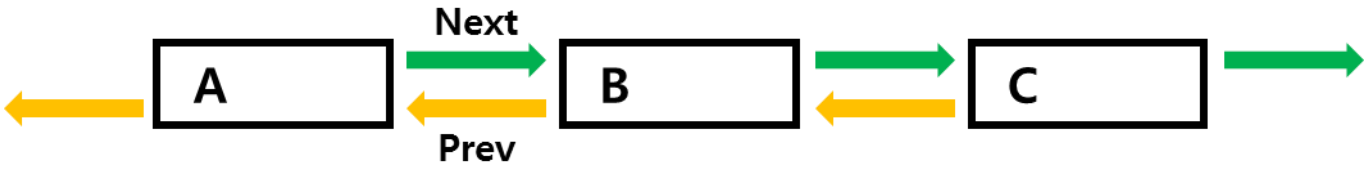
  - <Free block>



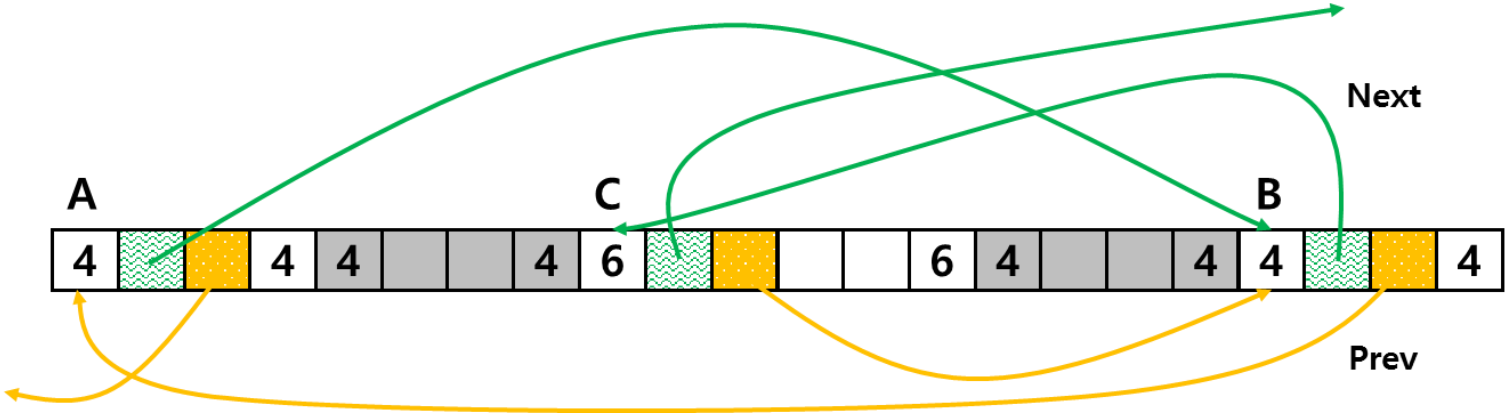
CNU Embedded System Laboratory

(참고) Malloc Lab – Explicit

- 논리 구조
  - 논리상으로는 각 Free 블록들이 순서대로 연결된 형태로 되어있다.



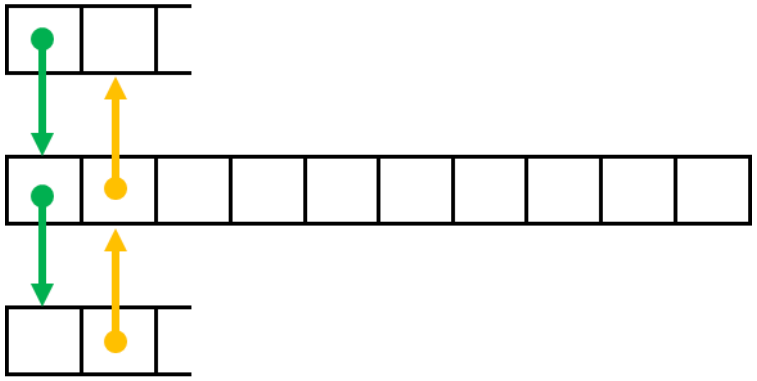
- 실제 구조
  - 실제 연결 링크는 메모리 블록의 순서와 무관하다.



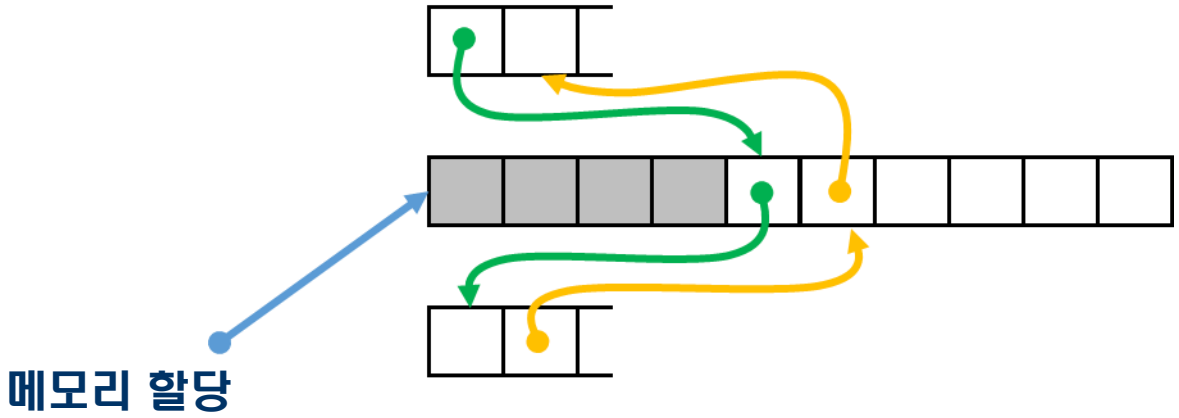
CNU Embedded System Laboratory

(참고) Malloc Lab – Explicit

- Explicit의 메모리 할당
  - 할당 전



  - 할당 후



CNU Embedded System Laboratory

Malloc Lab – Explicit 사용 매크로

```
/* single word (4) or double word *(8) alignment */
#define ALIGNMENT 8

/* Basic constants and macros */
#define HDRSIZE     4       /* header size (bytes) */
#define FTRSIZE     4       /* footer size (bytes) */
#define WSIZE       4       /* word size (bytes) */
#define DSIZE       8      /* doubleword size (bytes) */
#define CHUNKSIZE  (1<<12)  /* initial heap size (bytes) */
#define OVERHEAD    8       /* overhead of header and footer (bytes) */

#define MAX(x, y) ((x) > (y)? (x) : (y))
#define MIN(x, y) ((x) < (y)? (x) : (y))

/* Pack a size and allocated bit into a word */
#define PACK(size, alloc)  ((unsigned)((size) | (alloc)))

/* Read and write a word at address p */
#define GET(p)       (*(unsigned *)(p))
#define PUT(p, val)  (*(unsigned *)(p) = (unsigned)(val))
#define GET8(p)      (*(unsigned long *)(p))
#define PUT8(p, val) (*(unsigned long *)(p) = (unsigned long)(val))
```

CNU Embedded System Laboratory

Malloc Lab – Explicit 사용 매크로

```
/* Read the size and allocated fields from address p */
#define GET_SIZE(p)  (GET(p) & ~0x7)
#define GET_ALLOC(p) (GET(p) & 0x1)

/* Given block ptr bp, compute address of its header and footer */
#define HDRP(bp)       ((char *)(bp) - WSIZE)
#define FTRP(bp)       ((char *)(bp) + GET_SIZE(HDRP(bp)) - DSIZE)

/* Given block ptr bp, compute address of next and previous blocks */
#define NEXT_FREEP(bp) ((char*)(bp))
#define PREV_FREEP(bp) ((char*)(bp) + DSIZE)

/* Given free block pointer bp, compute address of next and previous free blocks */
#define NEXT_FREE_BLKP(bp)  ((char*)GET8((char*)(bp)))
#define PREV_FREE_BLKP(bp)  ((char*)GET8((char*)(bp) + DSIZE))

/* Given free block pointer bp, compute address of next pointer and prev pointer */
#define NEXT_BLKP(bp)  ((char*)(bp) + GET_SIZE((char*)(bp) - WSIZE))
#define PREV_BLKP(bp)  ((char*)(bp) - GET_SIZE((char*)(bp) - DSIZE))

/* rounds up to the nearest multiple of ALIGNMENT */
#define ALIGN(p)    (((size_t)(p) + (ALIGNMENT-1)) & ~0x7)
```

CNU Embedded System Laboratory

Malloc Lab - Explicit의 함수

- mm_init 함수 (참고용)

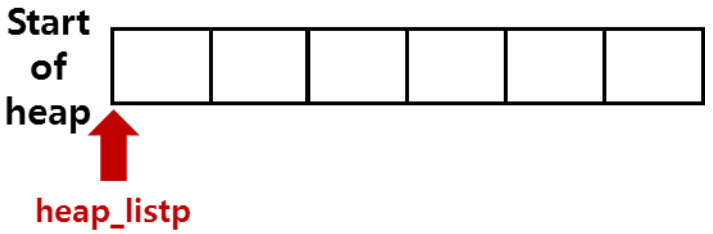
```
/*
 * Initialize: return -1 on error, 0 on success.
 */
int mm_init(void) {
    /* Request memory for the initial empty heap */
    if((h_ptr = mem_sbrk(DSIZE + 4 * HDRSIZE)) == NULL)
        return -1;
    heap_start = h_ptr;

    PUT(h_ptr, NULL);
    PUT(h_ptr + WSIZE, NULL);
    PUT(h_ptr + DSIZE, 0);                                  // alignment padding
    PUT(h_ptr + DSIZE + HDRSIZE, PACK(OVERHEAD,1));         // prologue header
    PUT(h_ptr + DSIZE + HDRSIZE + FTRSIZE, PACK(OVERHEAD,1));   // prologue footer
    PUT(h_ptr + DSIZE + 2 * HDRSIZE + FTRSIZE, PACK(0,1));      // epilogue header

    /* Move heap pointer over to footer */
    h_ptr += DSIZE + DSIZE;

    /* Leave room for the previous and next pointers, place epilogue 3 words down */
    epilogue = h_ptr + HDRSIZE;

    /* Extend the empty heap with a free block of CHUNKSIZE bytes */
    if(extend_heap(CHUNKSIZE/WSIZE) == NULL)
        return -1;

    return 0;
}
```

CNU Embedded System Laboratory

Malloc Lab – Explicit의 함수

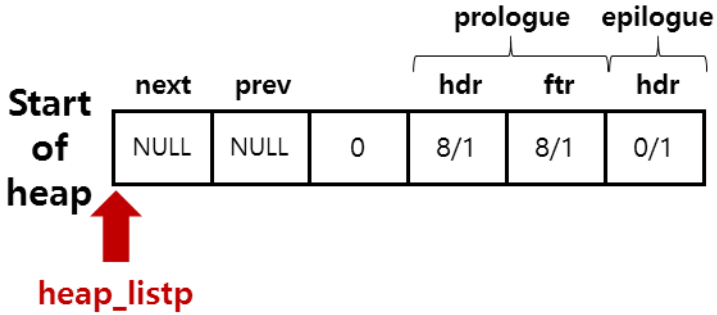
- mm_init 함수 (참고용)
  1) 초기 empty heap을 생성한다.

```
/* Request memory for the initial empty heap */
if((h_ptr = mem_sbrk(DSIZE + 4 * HDRSIZE)) == NULL)
    return -1;
heap_start = h_ptr;
```



  2) Root free block의 next와 prev를 생성하고, padding block과 prologue, epilogue를 할당한다.

```
PUT(h_ptr, NULL);
PUT(h_ptr + WSIZE, NULL);
PUT(h_ptr + DSIZE, 0);
PUT(h_ptr + DSIZE + HDRSIZE, PACK(OVERHEAD,1));
PUT(h_ptr + DSIZE + HDRSIZE + FTRSIZE, PACK(OVERHEAD,1));
PUT(h_ptr + DSIZE + 2 * HDRSIZE + FTRSIZE, PACK(0,1));
```

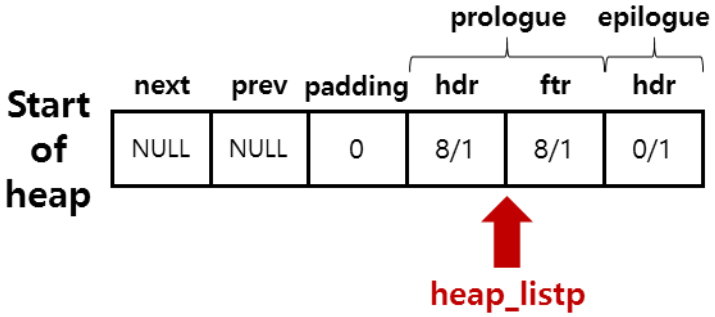


CNU Embedded System Laboratory

Malloc Lab – Explicit의 함수

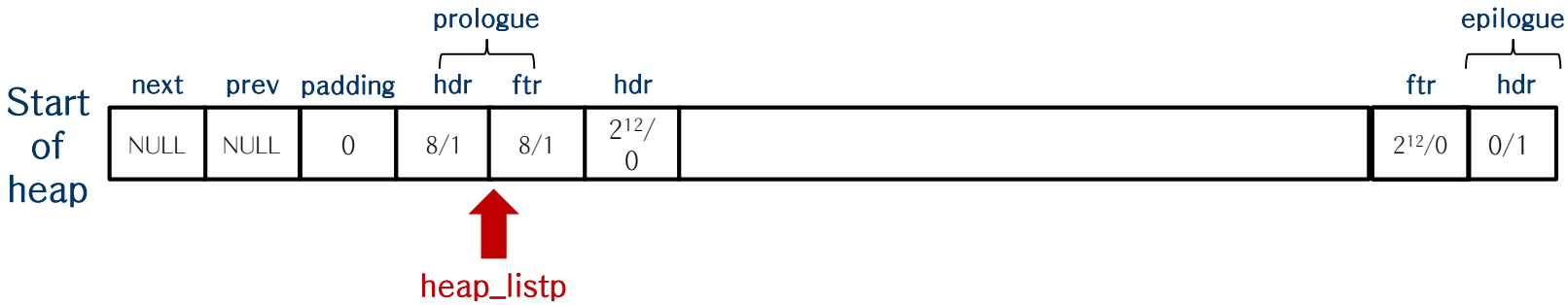
- mm_init 함수 (참고용)
  3) Heap pointer를 footer 위치로 이동시킨다.
  4) epilogue를 초기화 한다.

```
h_ptr += DSIZE + DSIZE;
epilogue = h_ptr + HDRSIZE;
```



5) 사용할 최대 크기의 heap을 미리 할당한다.

```
if(extend_heap(CHUNKSIZE/WSIZE) == NULL)
    return -1;
```



CNU Embedded System Laboratory

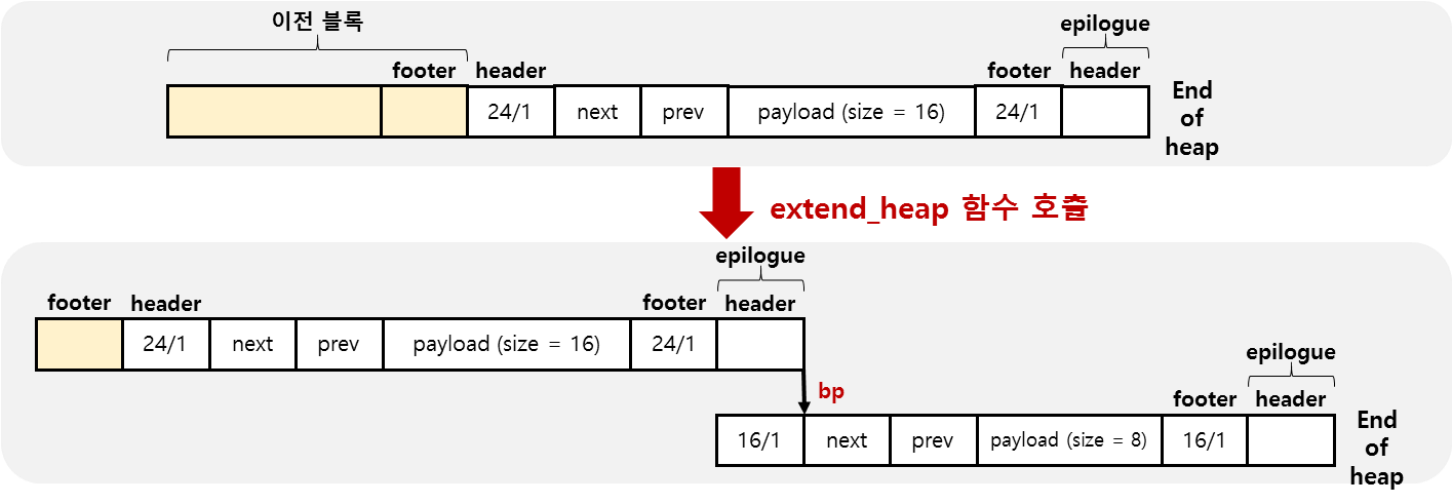
Malloc Lab - Explicit의 함수

- extend_heap 함수 (참고용)

```
/*
 * extend_heap
 */
inline void *extend_heap(size_t words) {
    unsigned *old_epilogue;     // Temp storage for current epilogue
    char *bp;                   // New block pointer after heap extension
    unsigned size;              // Request size for heap memory

    /* Allocate an even number of words to maintain alignment */
    size = (words % 2) ? (words+1)*WSIZE : words*WSIZE ;

    /* Request more memery from heap */
    if((long)(bp = mem_sbrk(size)) < 0)
        return NULL;

    /* Save the old epilogue pointer */
    old_epilogue = epilogue;
    epilogue = bp + size - HDRSIZE;

    /* Write in the header, footer, and new epilogue */
    PUT(HDRP(bp), PACK(size,0));
    PUT(FTRP(bp), PACK(size,0));
    PUT(epilogue, PACK(0,1));

    return coalesce(bp);
}
```

Malloc Lab – Explicit의 함수

- extend_heap 함수 (참고용)
  - extend_heap 함수가 호출 되면, 요청된 크기만큼 heap을 늘리고 이전 블록의 epilogue가 다음 블록의 header가 된다.



  - 새로운 free 블록이 할당되었으므로 free 블록을 통합하는 coalesce() 함수를 호출하도록 되어있다.
    - LIFO 정책의 free 과정과 동일하게 free 블록을 통합하도록 함수 작성
    - free() 함수 내부에서도 호출

CNU Embedded System Laboratory

Malloc Lab - Explicit의 함수

- malloc 함수 (참고용)
  - 주석 부분을 채워야 한다.
  - free list를 검색하는 find_fit() 함수를 작성해야 한다.
  - free 블록에 할당하는 함수인 place() 함수를 작성한다.
  - LIFO 정책의 free 과정을 coalesce() 함수에 작성한다.

```
void *malloc (size_t size) {
        char *bp;               // Block pointer, points to first byte of payload
        unsigned asize;         // Block size adjusted for alignment and overhead
        unsigned extendsize;    // Amount to extend heap if no fit

        // size가 올바르지 않을 때 예외처리

        // block의 크기 결정

        // 결정한 크기에 알맞은 블록을 list에서 검색하여 해당 위치에 할당
        if((bp = find_fit(asize)) != NULL)
        {
                place(bp, asize);
                return bp;
        }

        // free list에서 적절한 블록을 찾지 못했으면 힙을 늘려서 할당
        extendsize = MAX(asize, CHUNKSIZE);
        if((bp = extend_heap(extendsize/WSIZE)) == NULL)
                return NULL;
        place(bp, asize);
        return bp;
        return NULL;
}
```

CNU
Embedded System
Laboratory

과제

1. Malloc Lab
   I. implicit방식의 malloc 구현
   II. Explicit방식의 malloc 구현

2. Malloc Lab 보고서
   I. mm-naive 대한 설명
      1) naive에 사용된 매크로 및 구현 함수 설명
   II. mm-implicit 대한 설명
      1) implicit에 사용된 매크로 및 구현 함수 설명
   III. mm-explicit 대한 설명
      1) explicit에 사용된 매크로 및 구현 함수 설명
   IV. explicit 방식에 대한 ./mdriver 결과 첨부

3. 제출
   I. malloc-handout 디렉토리를 통째로 압축
      1) 파일명: [sys01]HW4_학번.tar.gz
   II. 결과 보고서를 작성
      1) 파일명: [sys01]HW4_학번.pdf
   III. I.과 II. 두개를 하나로 압축
      1) 파일명:[sys01]HW4_학번_이름.zip

CNU Embedded System Laboratory

제출 사항

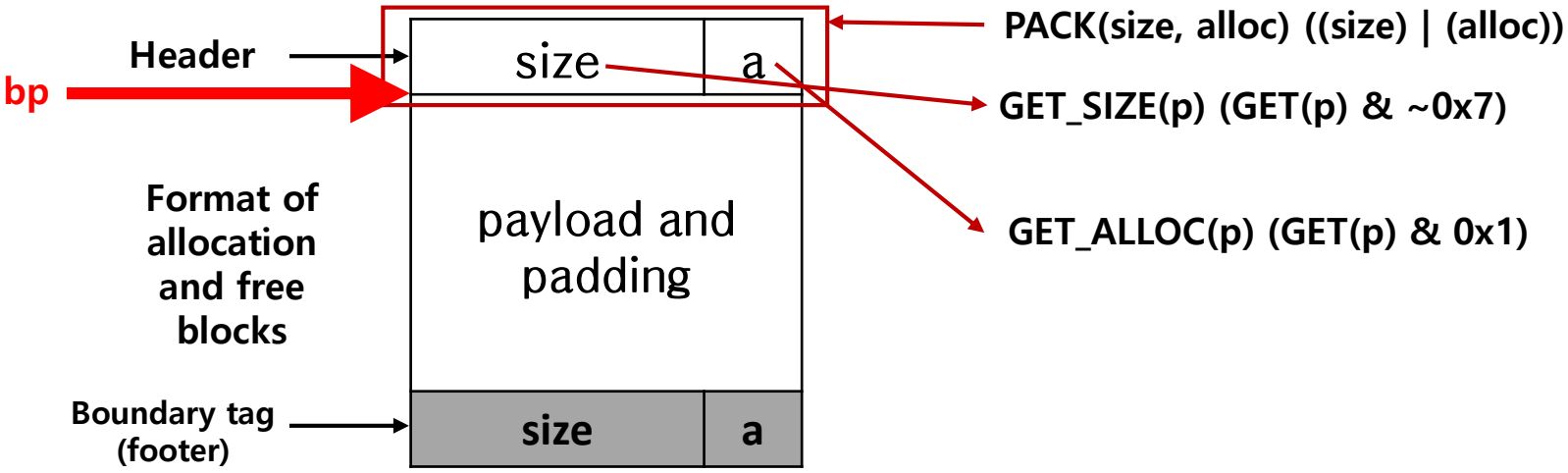
- 사이버캠퍼스에 제출
  - 자세한 양식은 앞장 슬라이드 참고.
  - 파일 제목: [sys01]HW4_학번_이름.zip
- 제출일자
  - 사이버 캠퍼스: 2021년 12월 6일 월요일 10시 59분 59초까지

참 고

CNU Embedded System Laboratory

Malloc Lab – Implicit 매크로 사용 예제

- 매크로를 통해서 필요한 값을 쉽게 획득할 수 있다.



- HDRP(bp) ((char *)(bp) – WSIZE)
  FTRP(bp) ((char *)(bp) + GET_SIZE(HDRP(bp)) - DISZE)
  GET(p) (*(size_t *)(p))
  - 사용 예
    - if(GET(HDRP(bp)) != GET(FTRP(bp)))
      - printf("ERROR : header dose not match footer\n");
    - 현재 블록 포인터 bp를 이용해서 Header와 Footer의 정보가 동일한지 검사

CNU Embedded System Laboratory

Malloc Lab – Implicit 매크로 사용 예제

- PUT(p, val) (*(size_t *)(p) = (val))
  - 사용 예
    - PUT(HDRP(bp), PACK(size, 0))
      - bp의 header에 block size와 alloc = 0 을 저장
      - 즉, 데이터를 지워주는 효과를 주고자 할 때 사용

- NEXT_BLKP(bp) ((char *)(bp) + GET_SIZE(((char *)(bp) - WSIZE)))
  PREV_BLKP(bp) ((char *)(bp) - GET_SIZE(((char *)(bp) - DSIZE)))
  - 사용 예
    - size = GET_SIZE(HDRP(PREV_BLKP(bp)))
      - 이전 블록의 크기를 얻어온다. PREV_BLKP를 사용해서 bp의 이전 블록을 쉽게 알 수 있다.
    - PUT(FRTP(NEXT_BLKP(bp)), PACK(size, 0))
      - 다음 블록의 footer에 size와 alloc = 0 을 할당