# 과제 #3 : Physical Page Frame Tracking XV6

○ 과제 개요

- xv6 운영체제의 물리 메모리 관리자는 기본적으로 사용 가능한 페이지 프레임들을 연결 리스트(free list)로 관리하고 있으며, 어떤 프레임이 어떤 프로세스에 사용 중인지에 대한 전역 추적 테이블은 제공하지 않고 있음
  - ✓ 본 설계 과제에서는 xv6 커널의 메모리 할당기(kalloc/kfree)를 확장하여 모든 물리 메모리 프레임의 실시간 사용 현황을 추적하는 기능을 구현하고 이를 통해 어떤 프로세스가 어떤 프레임을 사용 중이며, 언제부터 사용했는지를 실시간으로 파악할 수 있게 함
- 이 기능은 메모리 누수 디버깅이나 메모리 사용 패턴 분석 등에 활용할 수 있으며, 운영체제의 메모리 관리 동작을 가시화할 수 있음

○ 과제 구현 목표

- (필수)
  - ✓ 커널에 전역 프레임 정보 테이블을 도입하여 모든 물리 페이지 프레임의 할당 여부, 소유 PID, 사용 시작 tick을 지속적으로 추적
  - ✓ kalloc/kfree를 수정하여 프레임 할당.해제 시 해당 엔트리를 즉시 갱신하고, 기존 free list 기반 할당기에 추적 기능 결함
  - ✓ dump_physmem_info() 시스템 콜을 구현해 전역 테이블을 유저 프로세스가 프레임 상태를 조회 및 출력
- (추가)
  - ✓ 하드웨어 페이지 테이블(PGDIR/PTE) 의존 없이 소프트웨어 루틴만으로 가상주소→물리주소 변환을 수행하는 기능 구현
  - ✓ 역페이지 테이블(IPT, Inverted Page Table)을 도입해 물리페이지→(프로세스, 가상주소) 역방향 매핑 구현

○ A. 구현해야 할 기능(필수)

- 1. 전역 프레임 정보 테이블 유지
  - ✓ 물리 메모리의 각 프레임(frame)에 대한 정보를 저장하는 전역 테이블을 커널에 생성하고 유지. xv6의 메모리 할당 함수인 kalloc()과 해제 함수인 kfree()를 수정하여, 프레임이 할당되거나 해제될 때마다 이 테이블을 갱신해야 함.
  - ✓ 이 전역 테이블에는 시스템의 모든 물리 페이지 프레임에 대한 항목이 존재하며, free list 기반으로만 동작하는 기존 xv6 메모리 할당기에 프레임 사용 추적 기능을 추가하는 것임.
  - ✓ 반드시 전역 프레임 정보 테이블의 원소 개수는 60000으로 설정.
- 2. 프레임당 저장 정보: 전역 테이블의 각 엔트리에 다음 정보를 저장
  - ✓ (1) 할당 여부 – 해당 프레임이 현재 할당되어 사용 중인지 여부 (불리언 또는 플래그 값).
  - ✓ (2) 소유 프로세스 PID – 프레임을 현재 점유하고 있는 프로세스의 PID (없을 경우 특정 예약 값 사용).
  - ✓ (3) 사용 시작 시각 – 해당 프레임이 현재 소유 프로세스에 할당된 시점을 운영체제의 tick 카운트로 기록 (시스템 부팅 이후 경과한 tick 수). (참고. tick 값은 xv6 커널에서 시계 인터럽트마다 증가하는 전역 시간 단위로, ticks 변수로 누적 관리됨. 이를 통해 프레임이 얼마나 오래 사용되고 있는지 추적할 수 있음)
- 3. 시스템 콜 dump_physmem_info 구현
  - ✓ 커널 영역의 전역 프레임 정보를 사용자 공간으로 출력하기 위한 새로운 시스템 콜 int dump_physmem_info(void *addr, int max_entries)를 구현. 이 시스템 콜은 커널 내 전역 테이블에서 현재 저장된 프레임 사용 정보를 최대 max_entries개까지 읽어들여, 이를 사용자 제공 버퍼(addr가 가리키는 메모리)에 구조체 배열 형태로 복사
  - ✓ 커널에서 직접 printf로 출력하는 방식이 아니라, 시스템 콜의 반환 데이터로서 사용자 프로그램이 활용할 수 있도록 구조체 형태의 정보를 복사. 이를 위해 xv6 커널의 copyout()를 이용하여 커널 → 유저 공간 안전 복사를 수행
  - ✓ 사용자 영역에서도 동일한 구조체 정의를 제공하여(user.h 등에 선언) 커널과 사용자가 정보의 형식을 공유하도록 함. 예. 아래와 같은 구조체를 정의할 수 있음.

```
struct physframe_info {
    uint frame_index;   // 물리 프레임 번호
    int allocated;      // 1이면 할당됨, 0이면 무료 상태
    int pid;            // 소유 프로세스 PID (없으면 -1 등)
    uint start_tick;    // 이 프레임을 현재 PID가 사용 시작한 시각 (tick)
};
#define PFNNUM 60000
struct physframe_info pf_info[PFNNUM];
```

  - ✓ dump_physmem_info() 시스템 콜은 반환 값으로 복사된 엔트리 개수를 리턴하거나, 혹은 전체 프레임 개수 등 유의미한 값을 리턴할 수 있도록 함 (각자 알아서 구현 편의에 따라 결정)
- 4. 메모리 할당기 연동
  - ✓ kalloc()이 새 페이지 프레임을 할당할 때, 그리고 kfree()가 페이지를 해제할 때 전역 테이블을 정확히 갱신해야 함
  - ✓ kalloc(): 새로운 페이지를 할당하면 해당 물리 주소에 대응되는 테이블 엔트리를 찾아 allocated = 1로 표시하고, pid는 현재 실행 중인 프로세스의 PID로 설정. 커널이 사용하는 페이지의 경우 추적할 필요 없음. 또한 start_tick 필드에 현재 전역 tick 값을 기록. Tick 값은 ticks 전역 변수에서 얻을 수 있으며, tick 접근 시에는 동기화에 유의 (참고. ticks는 인터럽트 핸들러에서 증가하므로 tickslock으로 보호됨).
  - ✓ kfree(): 페이지가 해제되면 해당 프레임의 엔트리를 찾아 allocated = 0으로 표시하고, pid 필드를 초기화(예: -1로 설정)하여

더 이상 어떤 프로세스에도 속하지 않음을 나타냄. start_tick은 초기화하거나 0으로 설정(더 이상 유효하지 않은 시간). 이러한 갱신은 실제 프레임을 free list에 반환하기 직전에 수행되어야 하며, 반드시 메모리 할당기의 락을 획득한 상태에서 이뤄져야 함 (참고. xv6의 kmem 구조는 free list와 함께 스핀락으로 보호됨)

- 5. 동시성 및 정확성

✓ 다중 프로세스/다중 코어 환경에서 여러 프로세스가 동시 실행되더라도 프레임 추적 정보가 일관되고 정확하게 유지되어야 함.

✓ 메모리 할당/해제와 테이블 갱신 동작을 하나의 임계구역으로 취급하여 동기화(예. 이미 존재하는 kmem.lock을 재사용하거나 별도 락으로 보호).

✓ dump_physmem_info() 시스템 콜이 전역 테이블을 읽는 동안 다른 CPU나 프로세스가 테이블을 수정하면 안 되므로, 필요하다면 해당 syscall 구현에서 메모리 할당기 락을 잠시 획득하여 안전하게 복사하거나, 혹은 복사 중 일시적으로 인터럽트/스케줄링을 막는 등의 방법으로 일관된 스냅샷을 가져옴. 복사 완료 후에는 락을 해제. (참고. 이렇게 하면 실시간으로 변경 중인 데이터도 안정적으로 사용자에게 제공 가능)

○ B. A에서 구현한 기능 테스트 예제 및 출력

- 테스트 출력

✓ 아래 3개 프로그램의 소스코드는 일부만 별도로 공개 예정(10월 10일 전후)

✓ 주어진 부분은 절대 수정 불가하며, 반드시 아래와 같은 결과가 나와야 함

✓ memdump : dump_physmem_info() 시스템 콜로부터 받아온 프레임 정보를 가공하여 아래 표 형태로 출력. 각 행(row)은 하나의 물리 프레임을 나타내며, 필드들은 대괄호로 구분해서 출력하는 프로그램.
  - 할당된 프레임만 출력
  - 옵션 -a: 전체 프레임 테이블 출력 (free 포함)
  - 옵션 -p 〈PID〉: 특정 PID가 점유한 프레임만 필터

✓ memstress.c - 동적 할당을 만들어 상태 변화를 유도하는 도구. 지정한 개수 만큼의 페이지를 할당하고, 필요 시 해당 페이지에 write를 수행하는 프로그램.
  - 옵션 -n 〈N〉: N 페이지를 sbrk로 확보 (기본 64)
  - 옵션 -t 〈ticks〉: 확보 후 ticks 동안 유지했다가 종료 (기본 200)
  - 옵션 -w: 확보한 각 페이지의 첫 바이트를 쓰기(접근)하여 실제 물리 페이지 할당을 확실히 트리거

✓ memtest : memstress와 memdump를 사용하여 테스트하는 프로그램.

```
$ memtest
[memstress] pid=4 pages=31 hold=500 ticks write=0
[memstress] pid=5 pages=31 hold=500 ticks write=0
[memdump] pid=6  // pid 4 프로세스 확인
[frame#]       [alloc] [pid]   [start_tick] // start_tick는 memtest 시작 시간에 따라 달라질 수 있음
56994           1       4       152
56995           1       4       152
56996           1       4       152
56997           1       4       152
56998           1       4       152
56999           1       4       152
57000           1       4       152
57001           1       4       152
57002           1       4       152
57003           1       4       152
57004           1       4       152
57005           1       4       152
57006           1       4       152
57007           1       4       152
57008           1       4       152
57009           1       4       152
57010           1       4       152
57011           1       4       152
57012           1       4       152
57013           1       4       152
57014           1       4       152
57015           1       4       152
```

| | | | |
|---|---|---|---|
| 57016 | 1 | 4 | 152 |
| 57017 | 1 | 4 | 152 |
| 57018 | 1 | 4 | 152 |
| 57019 | 1 | 4 | 152 |
| 57020 | 1 | 4 | 152 |
| 57021 | 1 | 4 | 152 |
| 57022 | 1 | 4 | 152 |
| 57023 | 1 | 4 | 152 |
| 57024 | 1 | 4 | 152 |
| 57025 | 1 | 4 | 152 |
| 57026 | 1 | 4 | 152 |
| 57027 | 1 | 4 | 152 |
| 57028 | 1 | 4 | 152 |
| 57029 | 1 | 4 | 152 |
| 57030 | 1 | 4 | 152 |
| 57031 | 1 | 4 | 152 |
| 57032 | 1 | 4 | 152 |
| 57033 | 1 | 4 | 152 |
| 57034 | 1 | 4 | 152 |
| 57035 | 1 | 4 | 152 |
| 57036 | 1 | 4 | 152 |
| 57037 | 1 | 4 | 152 |
| 57038 | 1 | 4 | 152 |
| 57039 | 1 | 4 | 152 |
| 57040 | 1 | 4 | 152 |
| 57041 | 1 | 4 | 152 |
| 57042 | 1 | 4 | 152 |
| 57043 | 1 | 4 | 152 |
| 57044 | 1 | 4 | 152 |
| 57045 | 1 | 4 | 152 |
| 57046 | 1 | 4 | 152 |
| 57047 | 1 | 4 | 152 |
| 57048 | 1 | 4 | 152 |
| 57049 | 1 | 4 | 152 |
| 57050 | 1 | 4 | 152 |
| 57051 | 1 | 4 | 152 |
| 57052 | 1 | 4 | 151 |
| 57053 | 1 | 4 | 151 |
| 57054 | 1 | 4 | 151 |
| 57128 | 1 | 4 | 151 |
| 57129 | 1 | 4 | 151 |
| 57130 | 1 | 4 | 151 |
| 57131 | 1 | 4 | 151 |
| 57132 | 1 | 4 | 151 |
| 57133 | 1 | 4 | 151 |
| 57204 | 1 | 4 | 151 |
| 57205 | 1 | 4 | 152 |
| 57206 | 1 | 4 | 152 |
| 57277 | 1 | 4 | 151 |
| 57313 | 1 | 4 | 152 |

```
57314           1       4       152
57315           1       4       152
57316           1       4       152
57317           1       4       152
57318           1       4       152
57319           1       4       152
57320           1       4       152
57321           1       4       152
57322           1       4       152
57323           1       4       152
57324           1       4       152
57325           1       4       152
57326           1       4       152
57327           1       4       152
57328           1       4       152
57329           1       4       152
57330           1       4       152
57331           1       4       152
57332           1       4       152
57333           1       4       152
57334           1       4       152
57335           1       4       152
57336           1       4       152
57337           1       4       152
57338           1       4       152
57339           1       4       152
57340           1       4       152
57341           1       4       152
[memdump] pid=7  // pid 5 프로세스 확인
[frame#]        [alloc] [pid]   [start_tick]
56893           1       5       251
56894           1       5       251
56895           1       5       251
56896           1       5       251
56897           1       5       251
56898           1       5       251
56899           1       5       251
56900           1       5       251
56901           1       5       251
56902           1       5       251
56903           1       5       251
56904           1       5       251
56905           1       5       251
56906           1       5       251
56907           1       5       251
56908           1       5       251
56909           1       5       251
56910           1       5       251
56911           1       5       251
56912           1       5       251
```

| | | | |
|---|---|---|---|
| 56913 | 1 | 5 | 251 |
| 56914 | 1 | 5 | 251 |
| 56915 | 1 | 5 | 251 |
| 56916 | 1 | 5 | 251 |
| 56917 | 1 | 5 | 251 |
| 56918 | 1 | 5 | 251 |
| 56919 | 1 | 5 | 251 |
| 56920 | 1 | 5 | 251 |
| 56921 | 1 | 5 | 251 |
| 56922 | 1 | 5 | 251 |
| 56923 | 1 | 5 | 251 |
| 56924 | 1 | 5 | 251 |
| 56925 | 1 | 5 | 251 |
| 56926 | 1 | 5 | 251 |
| 56927 | 1 | 5 | 251 |
| 56928 | 1 | 5 | 251 |
| 56929 | 1 | 5 | 251 |
| 56930 | 1 | 5 | 251 |
| 56931 | 1 | 5 | 251 |
| 56932 | 1 | 5 | 251 |
| 56933 | 1 | 5 | 251 |
| 56934 | 1 | 5 | 251 |
| 56935 | 1 | 5 | 251 |
| 56936 | 1 | 5 | 251 |
| 56937 | 1 | 5 | 251 |
| 56938 | 1 | 5 | 251 |
| 56939 | 1 | 5 | 251 |
| 56940 | 1 | 5 | 251 |
| 56941 | 1 | 5 | 251 |
| 56942 | 1 | 5 | 251 |
| 56943 | 1 | 5 | 251 |
| 56944 | 1 | 5 | 251 |
| 56945 | 1 | 5 | 251 |
| 56946 | 1 | 5 | 251 |
| 56947 | 1 | 5 | 251 |
| 56948 | 1 | 5 | 251 |
| 56949 | 1 | 5 | 251 |
| 56950 | 1 | 5 | 251 |
| 56951 | 1 | 5 | 251 |
| 56952 | 1 | 5 | 251 |
| 56953 | 1 | 5 | 251 |
| 56954 | 1 | 5 | 251 |
| 56955 | 1 | 5 | 251 |
| 56956 | 1 | 5 | 251 |
| 56957 | 1 | 5 | 251 |
| 56958 | 1 | 5 | 251 |
| 56959 | 1 | 5 | 251 |
| 56960 | 1 | 5 | 251 |
| 56961 | 1 | 5 | 251 |
| 56966 | 1 | 5 | 251 |

```
56967           1       5       251
56968           1       5       251
56969           1       5       251
56970           1       5       251
56971           1       5       251
56972           1       5       251
56973           1       5       251
56974           1       5       251
56975           1       5       251
56976           1       5       251
56977           1       5       251
56978           1       5       251
56979           1       5       251
56980           1       5       251
56981           1       5       251
56982           1       5       251
56983           1       5       251
56984           1       5       251
56985           1       5       251
56986           1       5       251
56987           1       5       251
56988           1       5       251
56989           1       5       251
56990           1       5       251
56991           1       5       251
56992           1       5       251
56993           1       5       251
57124           1       5       251
57126           1       5       251
57311           1       5       251
[memstress] pid=4 done
[memstress] pid=5 done
[memdump] pid=8  // 종료된 pid 5 프로세스 확인. 출력 없는 것이 정상
[frame#]        [alloc] [pid]   [start_tick]
$
```

○ C. 추가 기능 구현

- (1)소프트웨어 페이지 워커(sw_vtop()) 구현

✓ int sw_vtop(pde_t *pgdir, const void *va, uint32_t *pa_out, uint32_t *pte_flags_out);

✓ 주어진 프로세스의 최상위 디렉터리(pgdir)와 가상주소(va)로부터 소프트웨어만으로 PTE를 찾아 물리주소/플래그를 계산해 리턴

✓ 하드웨어(에뮬레이터) 접근 없이 메모리 내 페이지 테이블 엔트리 구조를 직접 파싱(PDE/PTE 인덱스 계산, present/perm 비트 확인, PTE→프레임번호→물리주소 조합).

- (2)역페이지 테이블(IPT) 구현

✓ IPT를 위한 자료 구조 예.

```
struct ipt_entry {
  uint32_t pfn;           // 물리 프레임 번호
  uint32_t pid;           // 소유 프로세스 PID
  uint32_t va;            // 매핑된 가상주소(페이지 기준)
  uint16_t flags;         // PTE 권한(P/W/U 등) 스냅샷
  uint16_t refcnt;        // 역참조 카운트(옵션)
  struct ipt_entry *next; // 해시 체인
};
extern struct ipt_entry *ipt_hash[IPT_BUCKETS];
```

✓ 삽입/삭제 트리거: mappages(), uvmalloc()/uvmdealloc(), loaduvm() 등에서 매핑/해제 시 동일하게 IPT 갱신
✓ 중복 매핑 처리: 동일 물리프레임이 여러 (pid,va)에 매핑될 수 있으므로 체인으로 보관. refcnt 관리.
✓ 동기화: 간단히 스핀락 하나로 보호해도 좋으나, 성능 항목에서 비용을 분석할 것

- (3)SW기반 TLB

✓ 최근 N개 조회 (pid, va_page)→pa_page 을 고정 크기 Direct-mapped 또는 2-way set associative 캐시로 저장
✓ 히트율/미스율을 /proc 스타일 인터페이스나 디버그로 출력

- (4)일관성(Consistency)

✓ 페이지 테이블 변경 시 IPT와 소프트 TLB 일관 유지: remap, munmap류 동작(해제/권한 변경)에서 IPT 제거 및 갱신하여 소프트 TLB invalidation 수행

- (5)테스트 프로그램

✓ 페이지 테이블 변경 시 IPT와 소프트 TLB 일관 유지
  - remap, munmap류 동작(해제/권한 변경)에서 IPT 제거 및 갱신하여 소프트 TLB invalidation 수행다양한 권한 조합(P/R/W/U) 페이지 생성 후 vtop 결과 검증
✓ 동일 물리페이지에 대한 COW 시나리오(간단 복제)로 IPT 중복 체인 확인
✓ exit() 후 IPT 정리(좀비가 없게) 검증

- (힌트) 사용자/커널 인터페이스

✓ int sys_vtop(void *va, uint32_t *pa_out, uint32_t *flags_out);
  - 현재 프로세스 컨텍스트에서 sw_vtop을 호출해 결과 반환.
✓ 시스템콜 2: int sys_phys2virt(uint32_t pa_page, struct vlist *out, int max);
  - IPT를 조회하여 해당 물리페이지를 참조하는 (pid, va_page, flags) 리스트를 최대 max개 반환(역방향 질의).
✓ 사용자 인터페이스 프로그램 : vtop, pfind
  - vtop 0xBEEF1234 → PA=... flags=... hit/miss=...
  - pfind 0x12345000 → 이 물리페이지를 참조 중인 (pid,va,flags) 목록.
✓ 기본적인 구현을 위해 수정이 필요한 파일 (예-자유롭게 스스로 알아서)
  - 커널 : vm.c(mappages/alloc/free), proc.c, sysproc.c, sysfile.c syscall.h/.c, usys.S, defs.h, mmu.h

○ A를 위한 간단한 힌트

- 사용자/커널 인터페이스

✓ 사용자 공간에서 동일한 구조체/프로토타입을 사용할 수 있도록 user.h(또는 별도 공용 헤더)에 아래 선언을 추가

```
// user.h (추가)
struct physframe_info {
  uint frame_index;   // 물리 프레임 번호(인덱스)
  int  allocated;     // 1: 사용 중, 0: free
  int  pid;           // 소유 PID (커널 전용인 경우 -1 또는 0 등 약속)
  uint start_tick;    // 이 PID가 사용 시작한 tick
};

int dump_physmem_info(void *addr, int max_entries);
```

○ 과제 제출 마감

- 25년 11월 02일(일) 23시 59분 59초까지 구글클래스룸으로 제출 (강의계획서 확인). 1초라도 늦으면 0점 처리
- 가산점 2 부여 마감 기한 : 25년 10월 19일(일)/ 10월 26일(일) 23시 59분까지 구글클래스로 제출 1초라도 늦으면 가산점2 없음
- 가산점 1, 2는 필수기능(A, B)만 구현한 경우, 가산점2는 부여하지 않으며, 가산점1의 50%를 부여. (강의계획서 상 모든 기능의 구현이라고 되어 있는 가산점1의 부여 기준은 본 과제에서는 무시하고 필수기능 구현 시 가산점 1의 대상(50% 미만의 학생 구현

등)이 되는 경우, 가산점1을 부여)

○ 주의사항
- 보고서 (hwp, doc, docx 등으로 작성 - 수행된 결과 (캡쳐 등) 반드시 포함시킬 것
- xv6에서 변경한 소스코드 및 테스트 쉘 프로그램 소스코드 (주어진 코드는 제외, 자신이 조금이라도 수정한 것은 반드시 제출)
- 보고서에는 본 설계명세서에서 주어진 소스코드의 분석도 반드시 포함되어야 함

○ 필수 구현
- A, B

○ 배점 기준
- A: 40점
- B: 10점
- C: 50점