

13조(+5조) 협업 진행 과정

[WEEK12-13] PintOS - Virtual Memory

```
pass tests/vm/cow/cow-simple
All 141 tests passed.
make[1]: Leaving directory '/workspaces/pintos_lab_docker/pintos/vm/build'
❖ jung1e@f5ef8745262d:/workspaces/pintos_lab_docker/pintos/vm$
sjk2915* 16:01 0 0 1 [Week 12-13]VM Debug (pintos_lab_docker)
```

서정

Repo 관리

- Base Repo 선정 및 커밋 컨벤션
- Code Style 통일
- git에서 협업 진행중 자신의 브랜치 커밋을 유지하면서 main 동기화

Base Repo 선정 및 커밋 컨벤션

가장 많은 Project2 테스트를 통과한 레포 선정
README.md로 커밋 컨벤션 및 참고자료 공지

README

[WEEK12~13] Pintos VIRTUAL MEMORY

커밋 컨벤션

Commit Type

Type	설명
Feat:	새로운 기능 추가
Fix:	버그 수정 또는 typo
Refactor:	리팩토링
Design:	CSS 등 사용자 UI 디자인 변경
Comment:	필요한 주석 추가 및 변경
Style:	코드 포맷팅, 세미콜론 누락, 코드 변경이 없는 경우
Test:	테스트(테스트 코드 추가, 수정, 삭제, 비즈니스 로직에 변경이 없는 경우)
Chore:	위에 걸리지 않는 기타 변경사항(빌드 스크립트 수정, assets image, 패키지 매니저 등)
Init:	프로젝트 초기 생성
Rename:	파일 혹은 폴더명 수정하거나 옮기는 경우
Remove:	파일을 삭제하는 작업만 수행하는 경우

[커밋 컨벤션 참고 자료](#)

Packages

No packages published
[Publish your first package](#)

Contributors 5

sjk2915 sjk2915
Owns this repository
Committed to this repository in the past week

Suggested workflows
Based on your tech stack

MSBuild based projects
Build a MSBuild based project.
Configure

C/C++ with Make
Build and test a C/C++ project using Make.
Configure

Code Style 통일

clang-format 을 통해 저장 시 자동으로 코드 스타일이 통일되도록 유지

```
≡ .clang-format
1   BasedOnStyle: Microsoft
2   IndentWidth: 4
3   TabWidth: 4
4   UseTab: Never
5   SortIncludes: Never
6   ColumnLimit: 100
7   PointerAlignment: Right
```

git에서 협업 진행중 자신의 브랜치 커밋을 유지하면서 main 동기화

잔디유지를 위해 개인 레포와 팀 레포를 둘다
쓰는 경우 main 동기화를 위해 추가 셋팅

목표

- 팀 레포에서는: 내 브랜치 = main과 완전히 동일 (충돌 없음, 깔끔)
- 개인 레포에서는: 오늘 실제 작업 커밋 스냅샷 남겨서 기록/잔디 유지

0) 1회 세팅

```
# 개인(잔디) 레포 원격 추가
git remote add grass 개인레포주소
git remote -v 로 확인

# 잔디 푸시 alias: origin엔 현재 브랜치, grass엔 main으로
git config --global alias.pushgrass '!f(){ \
  git push origin HEAD && \
  git push --force-with-lease grass HEAD:main; \
}; f'

# 미팅 후 동기화 alias: 베이스=main, 내용=main로 맞춘 뒤 origin에만 푸시
git config --global alias.sync-main '!f(){ \
  git fetch origin && \
  git rebase -X ours origin/main && \
  git restore -SW --source=origin/main . && \
  (git diff --quiet --staged || git commit -m "sync: match origin/main exactly after rebase"); \
  git push --force-with-lease origin HEAD; \
}; f'
```

✓ 데일리 루틴

1) 작업 → 커밋 (각자 브랜치에서)

2) 미팅 전: 오늘 작업 스냅샷 + 잔디

```
git pushgrass
```

- origin: 내 브랜치로 푸시
- grass: main으로 푸시(잔디 + 스냅샷 기록)

3) 리뷰/미팅

4) 미팅 후: 팀 레포 내 브랜치 = main과 완전히 동일화

```
git sync-main
```

주의

- pushgrass는 “미팅 전”에만, sync-main은 “미팅 후(origin만)” 실행.
- origin 보호 브랜치 정책에 따라 force push가 막힐 수 있음(개인 브랜치는 허용 권장).
- 잔디 반영되려면 `git config user.email` 이 GitHub 계정 이메일과 매칭되어야 함.

👍 1 🗨️ 1 😊

Main 관리

- 각 날마다 정해진 인원이 자신의 코드를 main 에 Pull request
- 코어 타임 진행하며 팀원이 돌아가며 Review 진행 후 Approve→Merge
- 코어 타임 끝날때 모두가 동기화 진행

각 날마다 정해진 인원이 자신의 코드를 main 에 Pull request

인원은 로테이션 → 정해진 분량만큼 개발하지 못하면 상의하여 가장 잘 구현한 사람의 코드

Merged

feat: spt 및 일부 vm 함수 완성 #31
seongwonjung merged 6 commits into `main` from `sjk2915` 3 weeks ago

sjk2915 commented 3 weeks ago

Owner

...

No description provided.

sjk2915 added 6 commits 3 weeks ago

Update vm.h

92a5d87

style: 코드포맷팅

729344a

Fix: UNUSED 명시

5c9380b

Fix: spt_find_page 수정

440014a

comment: 나중에 개발할 예정 코멘트

211489c

style: 코드 정렬

15bad74

seongwonjung approved these changes 3 weeks ago

View reviewed changes

seongwonjung left a comment • edited

Collaborator

...

load_segment -> vm_alloc_page_with_initializer를 호출할 때 넘겨야 할 데이터들 잘 참고하겠습니다!

Reviewers

seongwonjung

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

Notifications

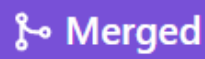
Customize

Unsubscribe


You're receiving notifications because you authored the thread.


코어 타임 진행하며 팀원이 돌아가며 Review 진행 후 Approve→Merge



리뷰 진행하며 수정할 점이 보이면 토의 후 추가
커밋후 Approve 리뷰 후에 Merge 진행

 **Merged**

feat: 커널의 유저 스택 접근 폴트 처리 기능 추가 #49
sisyphusman merged 16 commits into `main` from `seongwonjung` 2 weeks ago

 feat: 커널의 유저 스택 접근 폴트 처리 기능 추가 `0c75751`

 Fix: rsp 정상화, sys_read 버퍼 검증 추가 `0127fa0`

  sjk2915 approved these changes 2 weeks ago [View reviewed changes](#)


pintos/vm/vm.c **Outdated**

167 169 bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)

168 170 {



171 + if (!user && is_user_vaddr(addr))


172 + f->rsp = thread_current()->user_rsp;

 sjk2915 2 weeks ago Owner ...



void *rsp = user ? f->rsp : thread_current()->user_rsp;

예시입니다

  1

 Reply...

Resolve conversation

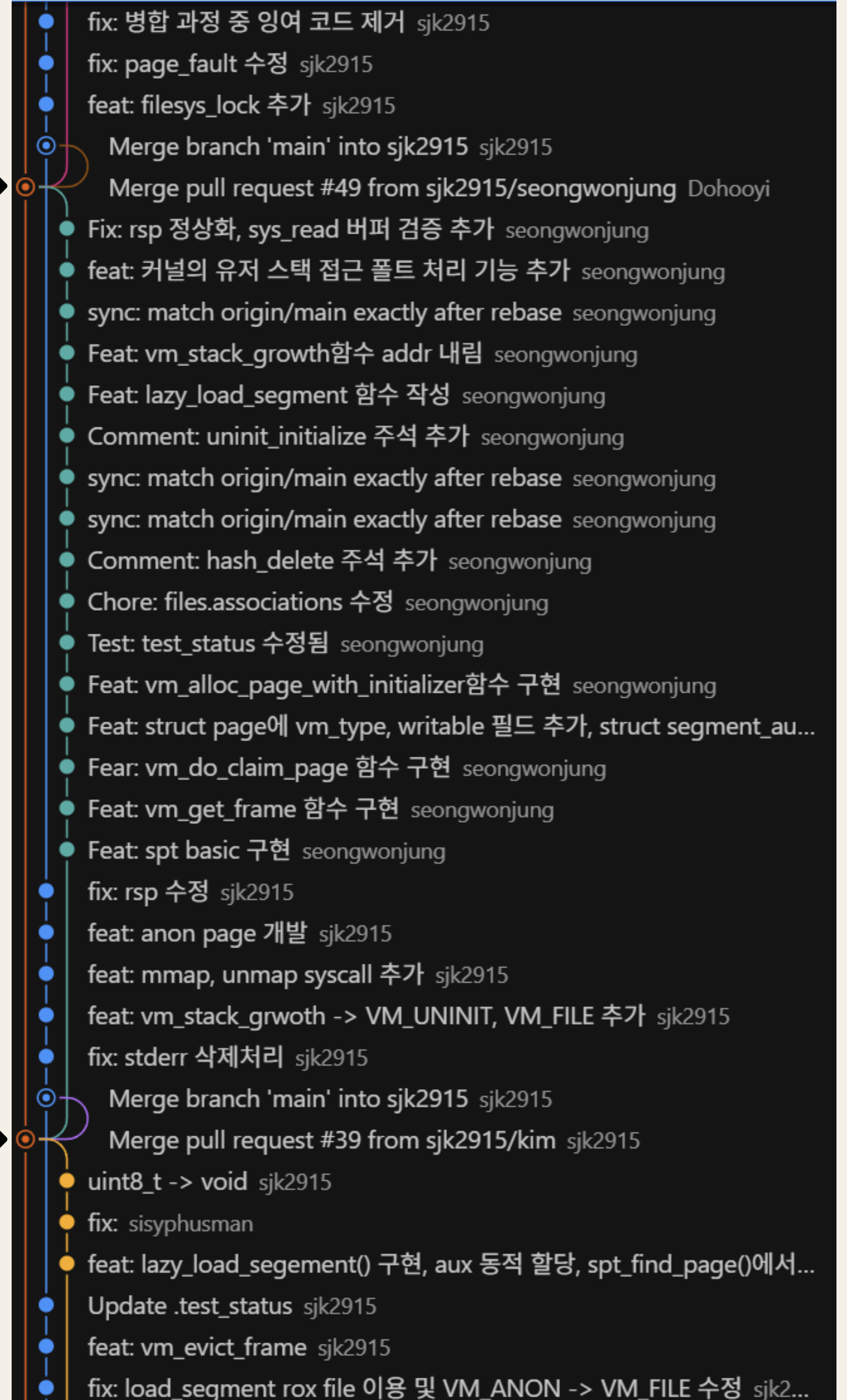
  sisyphusman merged commit `5a10aef` into `main` 2 weeks ago Revert

코어 타임 끝날때 모두가 동기화 진행

코어 타임 종료시에 반드시 모두가 동기화 하여
다음 Pull request 시에 Main과 Conflict 일어
나는 일 없도록 진행

10/1 코어타임 →

9/30 코어타임 →



issue 관리

- 코어타임 때 다음 날 개발해야할
기능 상의후 새 issue 추가
- 깃허브내 칸반보드와 로드맵으로
issue 관리
- 모두가 기능 구현 후 Assign시
issue 처리 완료

코어타임 때 다음 날 개발해야할 기능 상의회 새 issue 추가

정해진 기간까지 개발해야할 기능 상의회
issue에 추가, 대략적으로 자신이 공부하여
구현해야할 기능 상세 설명

feat: supplemental_page_table_copy 함수 구현 #51

Edit

New issue



Closed



sjk2915 opened 2 weeks ago · edited by sjk2915

Edits

Owner



Assignees



crucial-sub
seongwonjung
sisyphusman
sjk2915

Labels



No labels

src 즉 부모의 spt 를 보고 dst 즉 자식의 spt 에 동일하게 복사해주기
-> alloc만 된 상태면 uninit page 일텐데 동일하게 alloc만 해주기
• 이 page가 어떤 page가 될지 알아낸후 vm_alloc_page_with_initializer() 호출
• aux 도 복사해주기 !!!!! 부모거 참조시키면 안됨

-> alloc 후 calim까지 된 상태면 anon page 또는 file page 일텐데 동일하게 처리 해주기

Create sub-issue



코어타임 때 다음 날 개발해야할 기능 상의후 새 issue 추가

정해진 기간까지 개발해야할 기능 상의후
issue에 추가, 대략적으로 자신이 공부하여
구현해야할 기능 상세 설명

feat: vm_stack_growth 함수 구현 #48

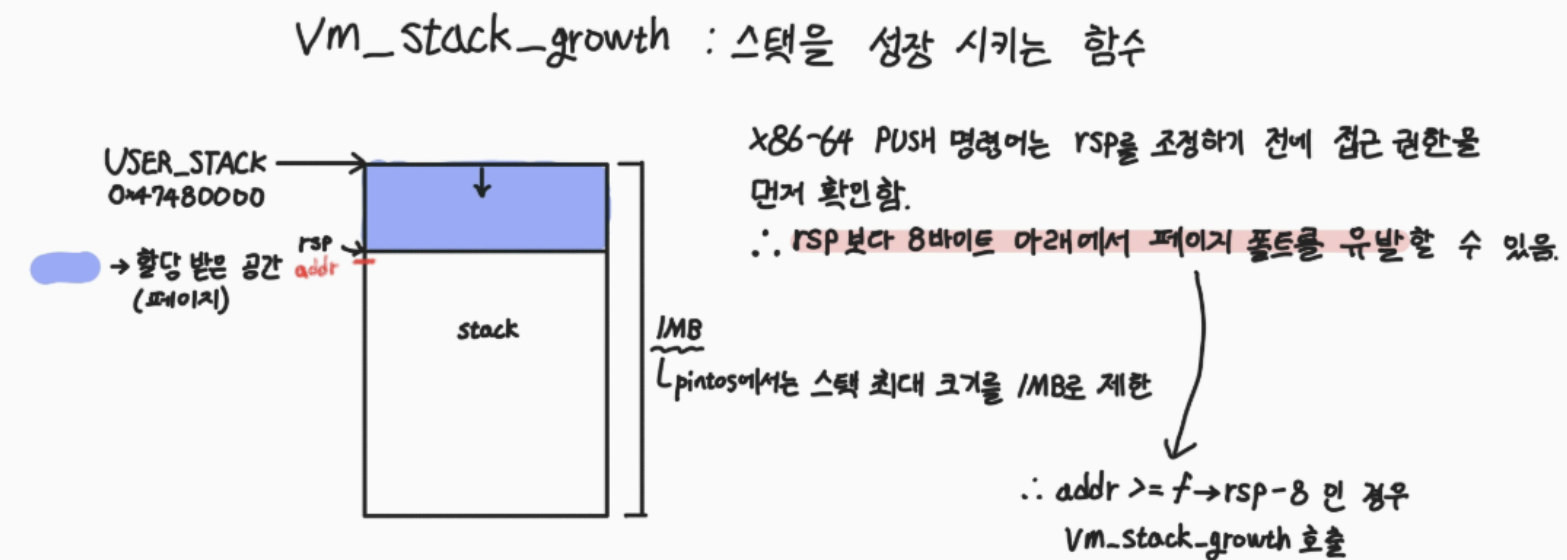
Closed

sjk2915/pintos_lab_docker Public



seongwonjung opened 2 weeks ago

Collaborator ...



내부동작은 setup 때와 같음

vm_alloc_page로 페이지를 할당 및 uint_page를 만들어 주고.

vm_claim_page로 프레임 할당 및 pml4 매핑을 해준다.

* 주의할 점 페이지 단위 할당 이므로 pg_round_down 후 할당 해줘야 함

Create sub-issue



Assignees

- crucial-sub
- seongwonjung
- sisyphusman
- sjk2915

Labels

No labels

Projects

12-13주차 정글 끝까지(PintOS) - Virtual Mem...

Status Done

Priority Choose an option

Size Choose an option

Estimate Enter number...

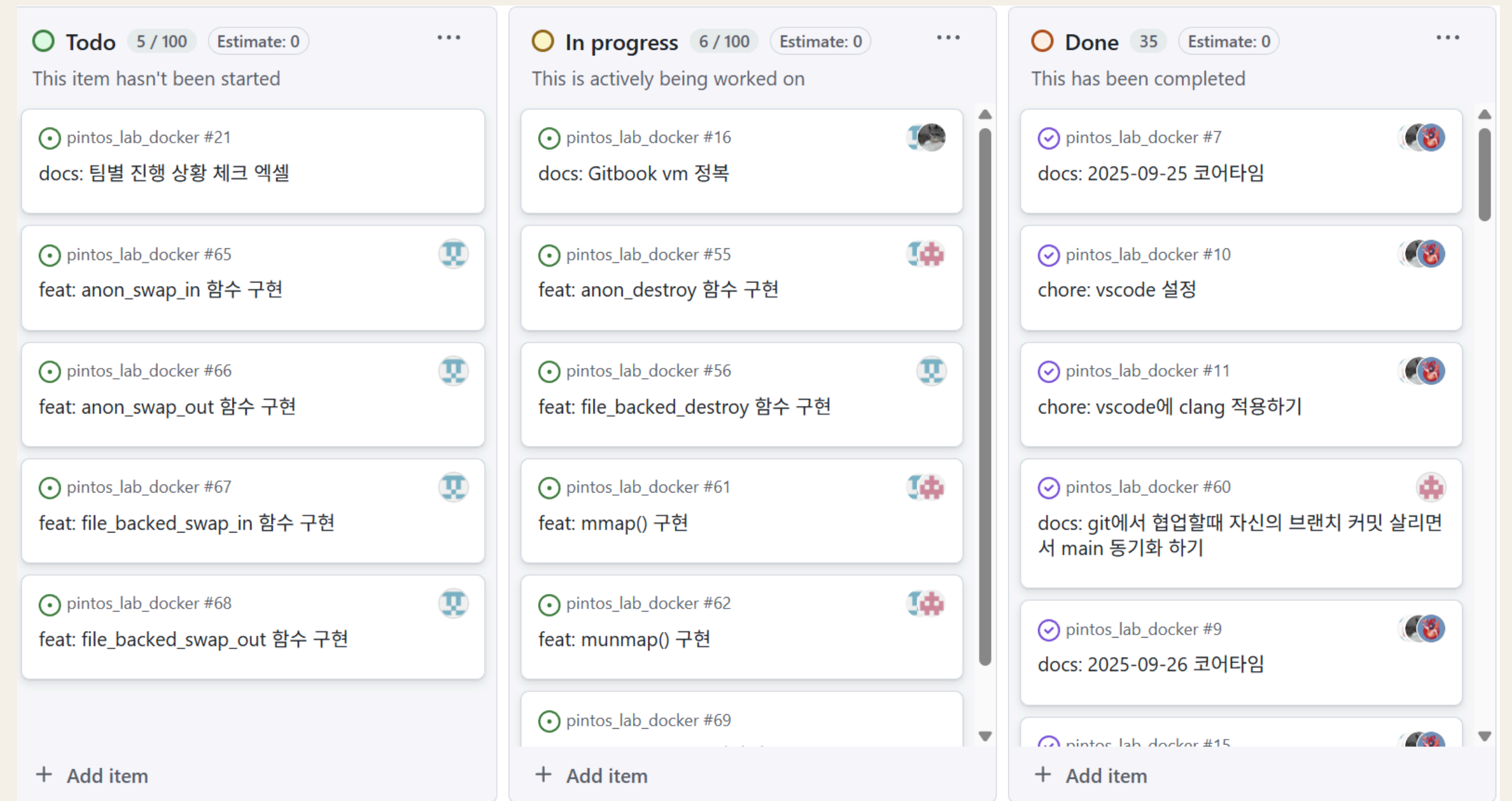
Start date Sep 30, 2025

End date Oct 1, 2025

Milestone

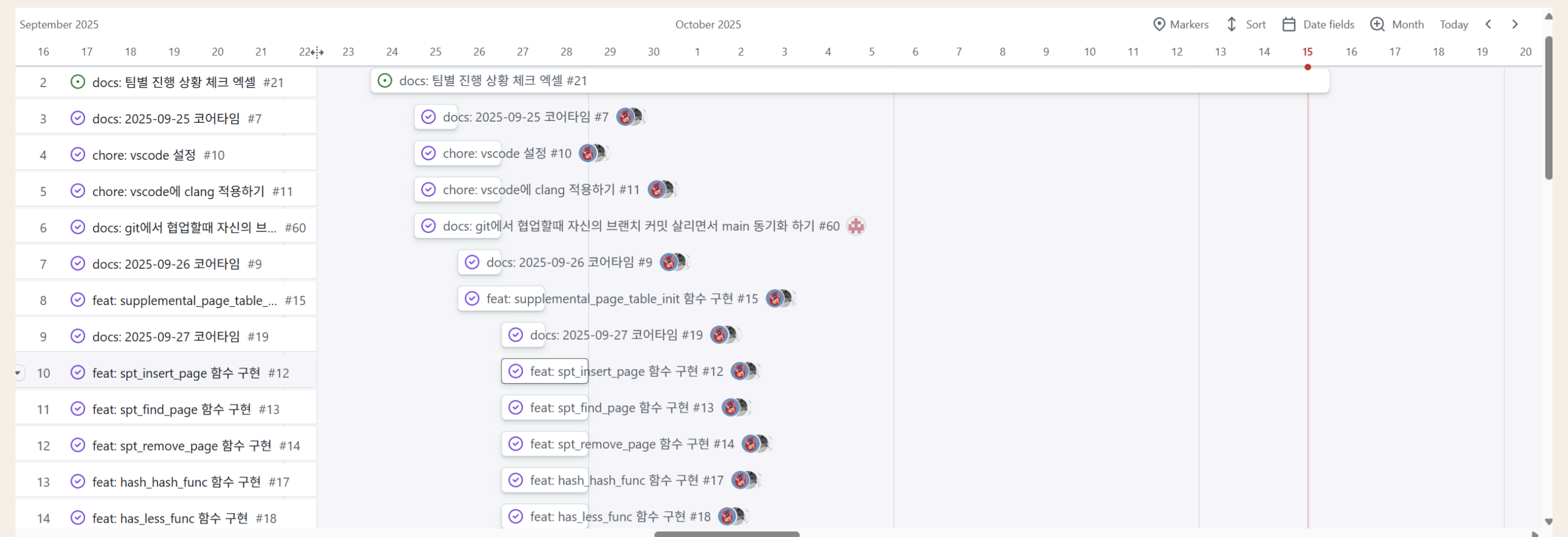
깃허브내 칸반보드와 로드맵으로 issue 관리

개발기간과 현재 진행중인 작업, 다음에 해야 할 일을 칸반보드, 로드맵 등으로 관리



깃허브내 칸반보드와 로드맵으로 issue 관리

개발기간과 현재 진행중인 작업, 다음에 해야
할 일을 칸반보드, 로드맵 등으로 관리



모두가 기능 구현 후 Assign시 issue 처리 완료

각자 기능 구현 시에 자신의 구현 결과를 커멘트
로 남기고 Assign, 4명 모두 assign시에 해당
issue를 Done으로 이동 후 처리 완료

🔒 Closed

feat: supplemental_page_table_kill 함수 구현 #52

sjk2915/pintos_lab_docker

📅

sjk2915 added this to 📅 12-13주차 정글 끝까지(PintOS) - Virtual Memory 2 weeks ago

🧩

sisyphusman 5 days ago · edited by sisyphusman

Edits Collaborator ⋮

함수 정의

-> 현재 스레드의 보조 페이지 테이블(SPT) 이 가지고 있는 모든 페이지 엔트리를 파괴
해서 테이블 자체를 없애는 건 아니고, 그 안에 들어 있는 데이터들만 전부 파괴 하는 함수

함수 구현

```
void supplemental_page_table_kill(struct supplemental_page_table *spt UNUSED)
{
    hash_clear(&spt->pages, spt_destroy_func);
}
```

😊

👤

sisyphusman self-assigned this 5 days ago

🔗

sisyphusman added a commit that references this issue 5 days ago

feat: supplemental_page_table_kill(), spt_destroy_func() 함수 구현 (#52, #53 ...)

4c1e4fb

📅

sjk2915 moved this from Todo to In progress in 📅 12-13주차 정글 끝까지(PintOS) - Virtual Memory 5 days ago

👤

sjk2915 self-assigned this 5 days ago

👤

seongwonjung 5 days ago · edited by sisyphusman

Edits Collaborator ⋮

supplemental_page_table_kill()에서 hash_clear() 대신 hash_destroy() 사용 시

- process.c (process_exec() 함수)

```
/* We first kill the current context */
process_cleanup();
supplemental_page_table_init(&thread_current()->spt);
```

process_cleanup() 함수에서 supplemental_page_table_kill() 호출
load -> load_segment -> vm_alloc_page_with_initializer -> spt_find_page() 에서 buckets 필요하므로
supplemental_page_table_init(&thread_current()->spt)로 spt init 해줘야 합니다.

😊

crucial-sub

seongwonjung

sisyphusman

sjk2915

Labels

No labels

Projects

📅 12-13주차 정글 끝까지(PintOS) - Virtual Mem...

Status Done ▾

Priority Choose an option

Size Choose an option

Estimate Enter number...

Start date Oct 10, 2025

End date Oct 11, 2025

Milestone

No milestone

Relationships

None yet

Development

Create a branch for this issue or link a pull request.

Notifications

Customize

🔔 Unsubscribe

You're receiving notifications because you're subscribed to this thread.

Participants

🧩 🧑 🧑 🧑

→ Transfer issue

📄 Duplicate issue

🔒 Lock conversation

📌 Pin issue

SPT COPY() 함수

(cow 미적용)

```
/* Copy supplemental page table from src to dst */  
bool supplemental_page_table_copy(struct supplemental_page_table *dst,  
| | | | | | | | | | struct supplemental_page_table *src)
```

*file, offset, read_bytes,
zero_bytes...

spt_copy()

*file, offset, read_bytes,
zero_bytes...

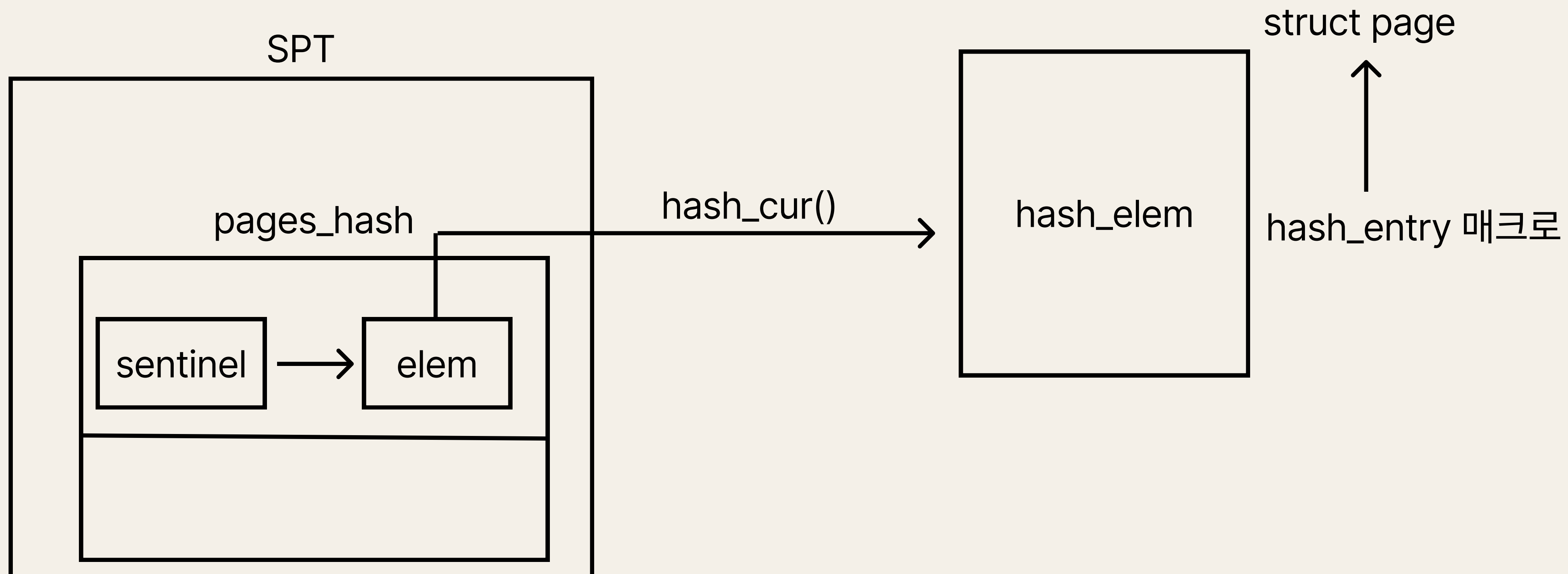
uninit이면 → 메타(aux)만 복제

ANON/FILE면 → 새 프레임+매핑+복사


```

struct hash_iterator i;
hash_first(&i, &src->pages);
while (hash_next(&i))
{
    struct page *src_page = hash_entry(hash_cur(&i), struct page, elem);
    struct page *dst_page;
    enum vm_type type = VM_TYPE(src_page->operations->type);

```



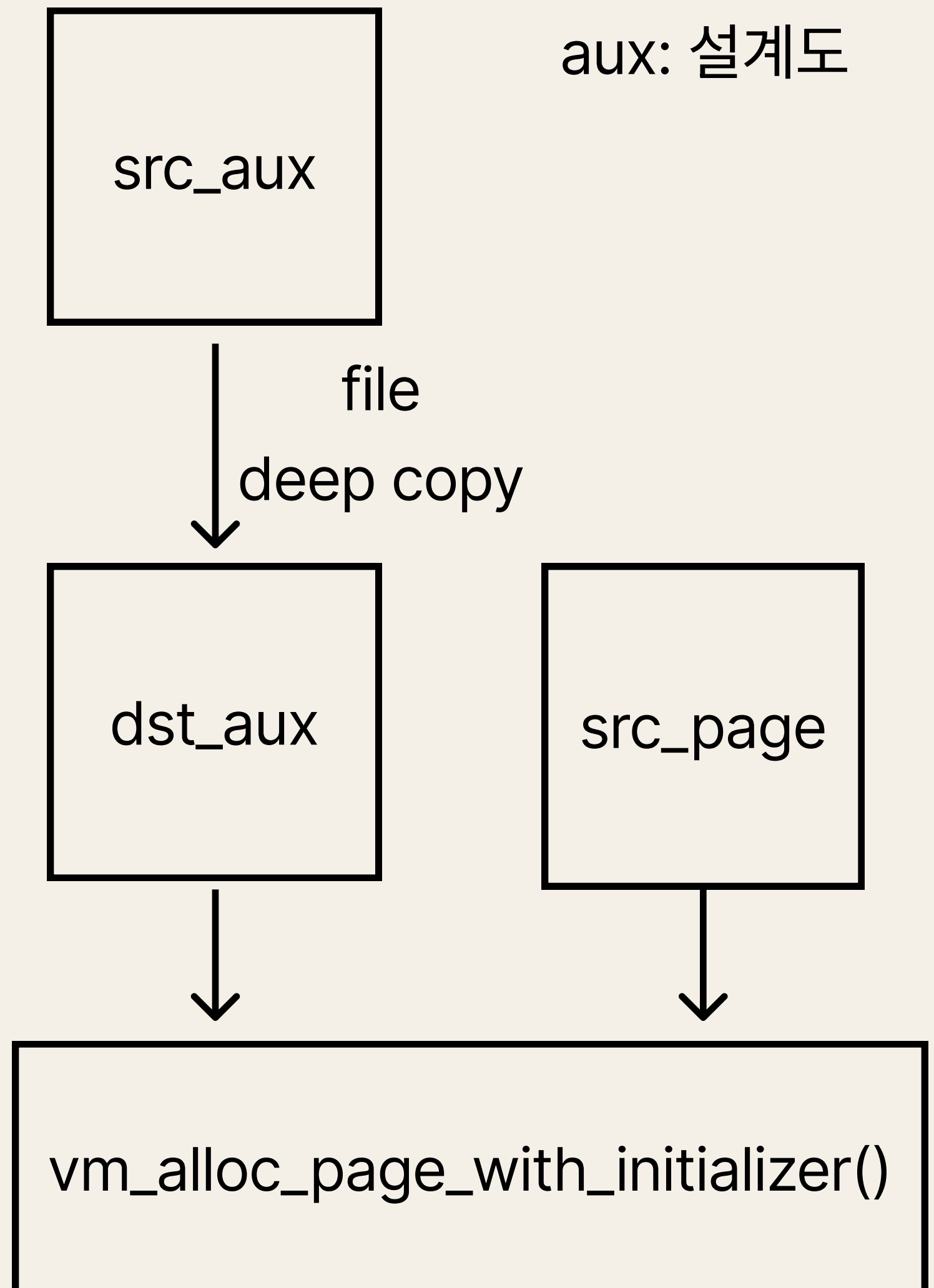
```

switch (type)
{
// 로드 안된 페이지
case VM_UNINIT: {
    struct segment_info *src_aux = src_page->uninit.aux;
    struct segment_info *dst_aux =
        (struct segment_info *)malloc(sizeof(struct segment_info));
    if (dst_aux == NULL)
        return false;
    *dst_aux = (struct segment_info){
        .file = file_reopen(src_aux->file),
        .ofs = src_aux->ofs,
        .read_byte = src_aux->read_byte,
        .zero_byte = src_aux->zero_byte,
    };

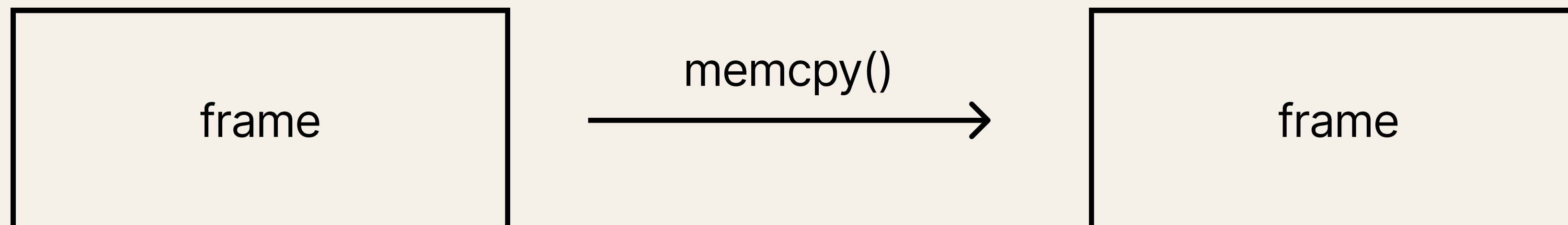
    if (!vm_alloc_page_with_initializer(page_get_type(src_page), src_page->va,
                                        src_page->writable, src_page->uninit.init, dst_aux))
    {
        file_close(dst_aux->file);
        free(dst_aux);
        return false;
    }
    break;
}
}

```

VM_UNINIT: 메모리에 없을때



```
// 로드 된 페이지
case VM_ANON:
case VM_FILE: {
    if (!(vm_alloc_page(type, src_page->va, src_page->writable) &&
        vm_claim_page(src_page->va)))
        return false;
    dst_page = spt_find_page(dst, src_page->va);
    memcpy(dst_page->frame->kva, src_page->frame->kva, PGSIZE);
    break;
}
default:
    return false;
}
```



vm_try_handle_fault

박종섭

In Project 2 (User Prog)

페이지 폴트 = 커널 또는 사용자 프로그램의 버그로 종료

In Project 3 (VM)

페이지 폴트 = 메모리를 효율적으로 관리하기 위한 '정상적이고 필수적인 과정'일 수 있음

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가
    struct supplemental_page_table *spt;
    struct page *page = spt_find_page(spt);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : 0;
            if (addr >= rsp - 8 && ((USE_STACK_GROWTH) && !user))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

page_fault(struct intr_frame *f)


```

/* Determine cause. */
not_present = (f->error_code & PF_P) == 0;
write = (f->error_code & PF_W) != 0;
user = (f->error_code & PF_U) != 0;

/* Count page faults. */
page_fault_cnt++;

#ifdef VM
/* For project 3 and later. */
if (vm_try_handle_fault(f, fault_addr, user, write, not_present))
    return;

```




```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한
    struct supplemental_page_table *spt = &th
    struct page *page = spt_find_page(spt, ad

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* The main system call interface */
void syscall_handler(struct intr_frame *f)
{
    // user_rsp 저장
    thread_current()->user_rsp = f->rsp;

    // ...
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

```

/* Return true on success */
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    // 주소 유효성 검사
    if (addr == NULL || is_kernel_vaddr(addr))
        return false;

    // 주어진 addr로 보조 페이지 테이블에서 폴트가 발생한 페이지를 찾기
    struct supplemental_page_table *spt = &thread_current()->spt;
    struct page *page = spt_find_page(spt, addr);

    if (not_present)
    {
        if (page == NULL)
        {
            uintptr_t rsp = user ? f->rsp : thread_current()->user_rsp;
            if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
            {
                vm_stack_growth(addr);
                return true;
            }
            return false;
        }
        return vm_do_claim_page(page);
    }
    return false;
}

```

성원

STACK_GROWTH

WHAT

In Project 2 (User Prog)

USER_STACK 에서 시작하는 단일 페이지

In Project 3 (VM)

필요에 따라 추가 페이지를 할당

HOW

setup_stack

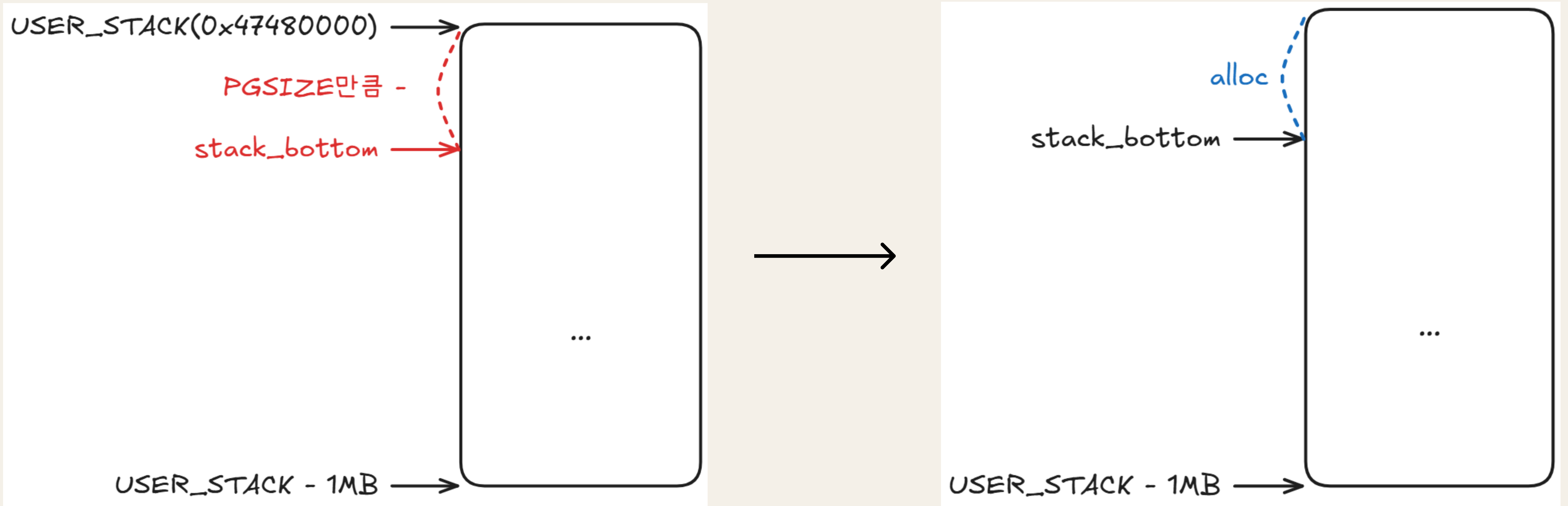
페이지 크기 만큼의 스택 공간을 할당해준다.

vm_stack_growth

필요에 따라 스택 공간에 추가적으로 페이지를 할당해준다.

setup_stack

1. 한 페이지 내려준다.
2. 페이지를 할당한다.
3. rsp를 설정한다.



HOW

setup_stack

```
/* Create a PAGE of stack at the USER_STACK. Return true on success. */
static bool setup_stack(struct intr_frame *if_)
{
    bool success = false;
    void *stack_bottom = (void *)(((uint8_t *)USER_STACK) - PGSIZE);

    /* TODO: Map the stack on stack_bottom and claim the page immediately.
     * TODO: If success, set the rsp accordingly.
     * TODO: You should mark the page is stack. */
    /* TODO: Your code goes here */

    if (vm_alloc_page(VM_ANON | VM_STACK, stack_bottom, true) && vm_claim_page(stack_bottom))
    {
        success = true;
        if_>rsp = USER_STACK;
    }

    return success;
}
```

1. 한 페이지 내려준다.

2. 페이지를 할당한다.

3. rsp를 설정한다.

vm_stack_growth

할당 받은 스택 공간을 넘어가는 곳에 접근할 때 → 페이지 추가 할당

```
/* Growing the stack. */
static void vm_stack_growth(void *addr)
{
    vm_alloc_page(VM_ANON | VM_STACK, pg_round_down(addr), true);
    vm_claim_page(addr);
}
```

고려할 점

addr를 PGSIZE에 맞게 내림 필요 → pg_round_down

스택영역에 대한 접근인지 확인 필요 → 호출 하는 곳에서

vm_stack_growth

스택영역에 대한 접근인지 확인 필요 → 호출 시점(vm_try_handle_fault)

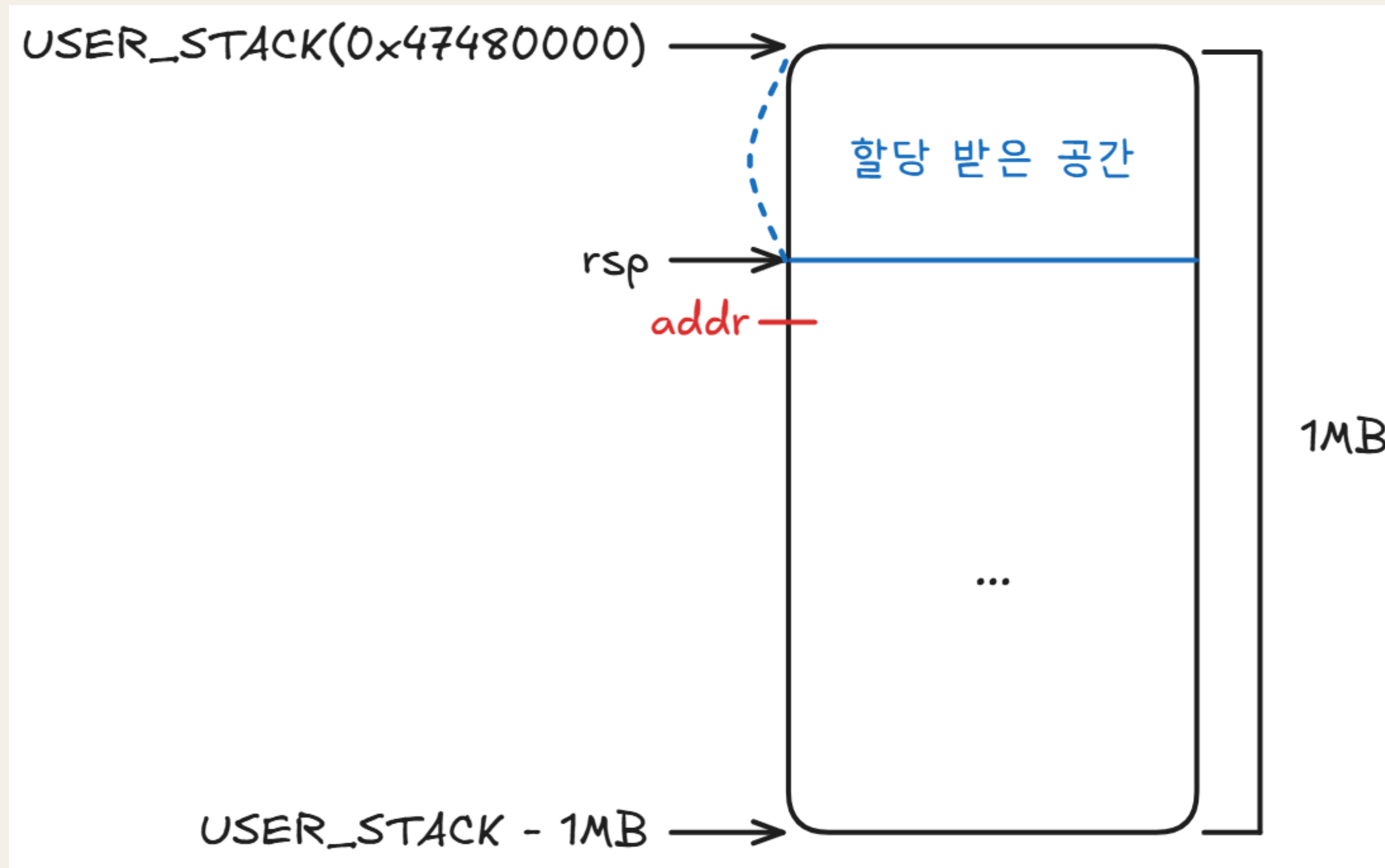
may interrupt a process at any time to deliver a "signal," which modifies data on the stack. However, the x86-64 PUSH instruction checks access permissions before it adjusts the stack pointer, so it may cause a page fault 8 bytes below the stack pointer.

x86-64의 PUSH 명령어는 스택 포인터를 조정하기 전에 접근 권한을 확인하기 때문에 스택 포인터 아래 8바이트에서 페이지 폴트가 발생할 수 있습니다.

그렇다면?

→ 유저 가상 메모리의 스택 영역 안인지? addr이 rsp 보다 8바이트 아래인지?

vm_stack_growth



호출 시 (vm_try_handle_fault)

```
bool vm_try_handle_fault(struct intr_frame *f, void *addr, bool user, bool write, bool not_present)
{
    if (page == NULL)
    {
        if (addr >= rsp - 8 && ((USER_STACK - (1 << 20)) < addr) && (addr < USER_STACK))
        {
            vm_stack_growth(addr);
            return true;
        }
        return false;
    }
}
```

addr이 rsp 보다 8바이트 아래인지?

유저 가상 메모리의 스택 영역 안인지?



확인 후 vm_stack_growth 호출

Thank you!