**Pintos Project 4: Virtual Memory**

담당 교수 : 박성용 교수

이름 / 학번 : 강민석 / 20181589

개발 기간 : 2022. 11. 15. ~ 2022. 12. 06.

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

src/vm디렉토리를 구현한다. 원래는 page fault가 발생하면 프로그램이 종료되는데, 이를 수정해서 page table을 만들어 이를 관리할 수 있는 자료구조와 함수들을 추가해준다. 그리고 Lazy load 방식으로 필요한 부분만을 물리 메모리에 올린다. 물리 메모리가 가득 찬 경우에는 swap disk라는 임시 저장 공간을 만들어 swap이 가능하도록 구현하고, page fault가 stack영역에서 발생하면 stack expand를 통해 추가 페이지를 할당한다

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목 개발의 필요성 또는 개발 시 기대되는 결과를 간략히 서술
  1. Page Table & Page Fault Handler

가상 주소를 물리 주소로 변환할 수 있는 page table을 구현한다. 가상 주소와 물리 주소는 일대일로 매핑되어 페이지 테이블에 자료구조로 존재하며, 해당 가상 주소를 가진 페이지 테이블에서 물리 주소를 찾으려 할 때 해당 물리 주소가 없으면 page fault가 발생한다. 기존에는 page fault가 발생하면 프로그램이 종료됐지만, page fault handler가 제대로 구현되면 해당 페이지가 할당돼 페이지 테이블에 추가된다. Page fault가 발생할 경우, 프로그램이 단순 종료되진 않고, 메모리 처리를 통해 계속 작동할 수 있다.

* 1. Disk Swap

물리 메모리가 부족하면, swap할 frame을 임시로 저장할 공간인 swap disk에다가 LRU algorithm을 이용해서 swap out할 frame을 넣는다. 그리고 새로운 frame을 할당 받아서 swap in을 진행한다. 이렇게 해주면 물리 메모리가 꽉 차도 프로그램이 죽지 않고 정상적으로 수행할 수 있다.

* 1. Stack Growth

확장 가능한 스택을 구현한다. 현재 스택의 크기는 4KB로 고정되어 있다. 현재 스택의 크기를 초과하는 주소에 접근이 발생했을 때, 유효한 스택 접근인지 segmentation fault인지 판별하는 휴리스틱을 적용하도록 코드 수정한다. 또한, 유효한 스택 접근으로 판별한 경우, 스택을 확장한다. 이때, 확장 가능한 스택의 최대 크기는 8MB가 되도록 코드를 수정한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술
  1. Page fault가 발생하는 이유와 이를 handling하는 전반적인 과정을 서술

가상 주소로 접근을 할 때, 가상 주소에 대응되는 물리 주소가 만약 page table에 존재하지 않는다면 이 때 page fault가 발생한다. Page fault가 발생하면 page fault를 handle하는 과정을 시도한다. 본인은 해당 가상주소의 virtual page를 표현하는 vm\_emtry라는 자료구조를 이용해서 관리하였다. Page fault가 일어날 때마다 가상 주소에 해당하는 vm\_entry를 탐색하여서, 이 vm\_entry에 해당되는 내용을 disk에서부터 읽어와서 물리 메모리에 frame으로 넣는다. 탐색을 빠르게 하기 위해 hash table 자료구조를 사용한다. 그리고, 가상주소의 유효성 검사를 해야 한다. 물리 메모리에 새로운 frame을 넣어줬다면 page table을 업데이트해서 page table에서 접근하려는 가상 주소에 대응되는 물리 주소를 맵핑해서 넣어준다.

* 1. Disk swap 발생 시 사용한 page replacement algorithm에 대해 서술

Page fault가 발생했을 때 물리 메모리가 다 차 있어서 할당할 공간이 없는 경우에 기존 물리 메모리에 있던 frame을 디스크로 swap out해서 공간을 만들고, 그 만들어진 공간에 접근하려는 가상 주소를 할당해서 맵핑한다. 어떤 frame을 swap out시킬지 결정하는 것은 LRU(Least Recently Used) 알고리즘을 따른다. 이 알고리즘은 가장 오랫동안 사용되지 않은 frame을 select해서 디스크로 evict 시킨다.

* 1. Stack growth 구현 시 stack 확장 여부를 판단할 수 있는 방법에 대해 서술

stack pointer인 esp로부터 grow limit 이내에 포함되는 영역을 접근한 경우, page fault 발생 시에 스택을 확장한다. Grow limit은 최대 8MB로 정해져 있다. stack의 시작 주소값에서 8MB를 뺀 것이 Maximum limit으로 설정되고, 이 안에 들어오지 않는 접근은 유효하지 않은 접근으로 간주하여 segmentation fault를 발생한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성
  1. Page Table & Page Fault Handler: 22.11.15. ~ 22.11.22.
  2. Disk Swap: 22.11.22. ~ 22.11.27.
  3. Stack Growth: 22.11.27~22.12.04.
  4. 코드 정리 및 보고서 작성: 22.12.04~22.12.06.
  5. **개발 방법**
* II. B의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드
  + 수정하거나 추가해야 하는 자료구조
  + 수정하거나 추가해야 하는 함수
* **Page fault의 발생과 handling**

<수정 파일>

 pintos/src/userprog/process.c

process.c에 handle\_mm\_fault() 함수를 구현한다. handle\_mm\_fault는 페이지 폴트 발생 시에 핸들링을 위해서 호출되는 함수이다. Page fault 발생 시에 물리페이지를 할당하는 함수이다.

 pintos/src/userprog/exception.c

원래는 page fault가 발생하면 exception.c의 page\_fault() 함수에 들어가서 경고메세지를 출력한 후에 프로세스를 kill 한다. 이를 지우고 vm\_entry를 찾아서 조건이 맞으면 handle\_mm\_fault 함수의 load\_page를 할 수 있도록 코드를 작성한다.

 pintos/src/userprog/syscall.c

syscall.c에서 가상주소의 유효성 검사를 수행한다. 시스템 콜 사용 시에 파라미터로 주어지는 string이나 buffer의 주소에 해당하는 vm\_entry가 존재하는지 검사한다.

 pintos/src/threads/thread.h

vm\_entry를 관리하기 위한 hash table을 thread 구조체에 선언한다.

<자료구조 및 추가 파일>

 pintos/src/vm/page.h & pintos/src/vm/page.c

vm\_entry 파일을 관리하기 위한 함수들을 구현한다. vm\_entry를 위한 vm\_entry structure를 구현한다. Page table은 user page와 kernel page가 일대일로 대응되기 때문에, vm\_entry의 탐색은 효율성을 위하여 hash table로 구현을 하였다.

물리메모리 할당 완료 후에 실제 디스크의 파일을 물리페이지로 load 해준다. page.c의 load\_file()에서 disk에 존재하는 page를 물리 메모리로 load하고, file\_seek와 file\_read를 이용하여 vm\_entry의 (file, offset)으로 한 페이지를 kaddr로 읽어들이는 함수를 구현한다.

* **Disk swapping (page replacement algorithm)**

**<수정 파일>**

 pintos/src/userprog/process.c

handle\_mm\_fault()에서 swapping을 지원할 수 있도록 수정한다. 또한, 원래 그냥 palloc\_get\_page()를 이용해서 메모리를 할당했는데, 이를 alloc\_page()라는 함수를 frame.c에 구현하고 process.c의 메모리 할당하는 부분을 alloc\_page()로 바꾼다.

 pintos/src/thread/init.c

main 내에서 swap과 lru\_list를 초기화해준다.

**<추가 파일>**

 pintos/src/vm/page.h

Page table을 위한 page structure를 구현한다. 그리고 clock algorithm을 이용하기 위한 lru list를 만든다.

 pintos/src/vm/frame.c

frame.c에서 물리 페이지가 부족할 때 try\_to\_free\_pages를 구현하여 LRU(Clock) 알고리즘을 사용하여 여유 메모리를 확보한다. Clock이 lru\_list를 순회하면서 victim을 선정하는데, 이 clock이 순회할 수 있도록 해주는 get\_next\_lru\_clock이라는 함수를 구현한다.

 pintos/src/vm/swap.c

swapping을 위한 swap\_init(), swap\_int(), swap\_out() 함수를 구현한다.

* **Stack growth**

**<수정 파일>**

 pintos/src/userprog/exception.c

page\_fault() 함수 내에서 page fault가 발생한 주소가 유효한 주소인지를 verify\_stack()을 이용해 확인하고, expand\_stack()을 이용해 stack 영역을 확장하는 부분을 수정한다.

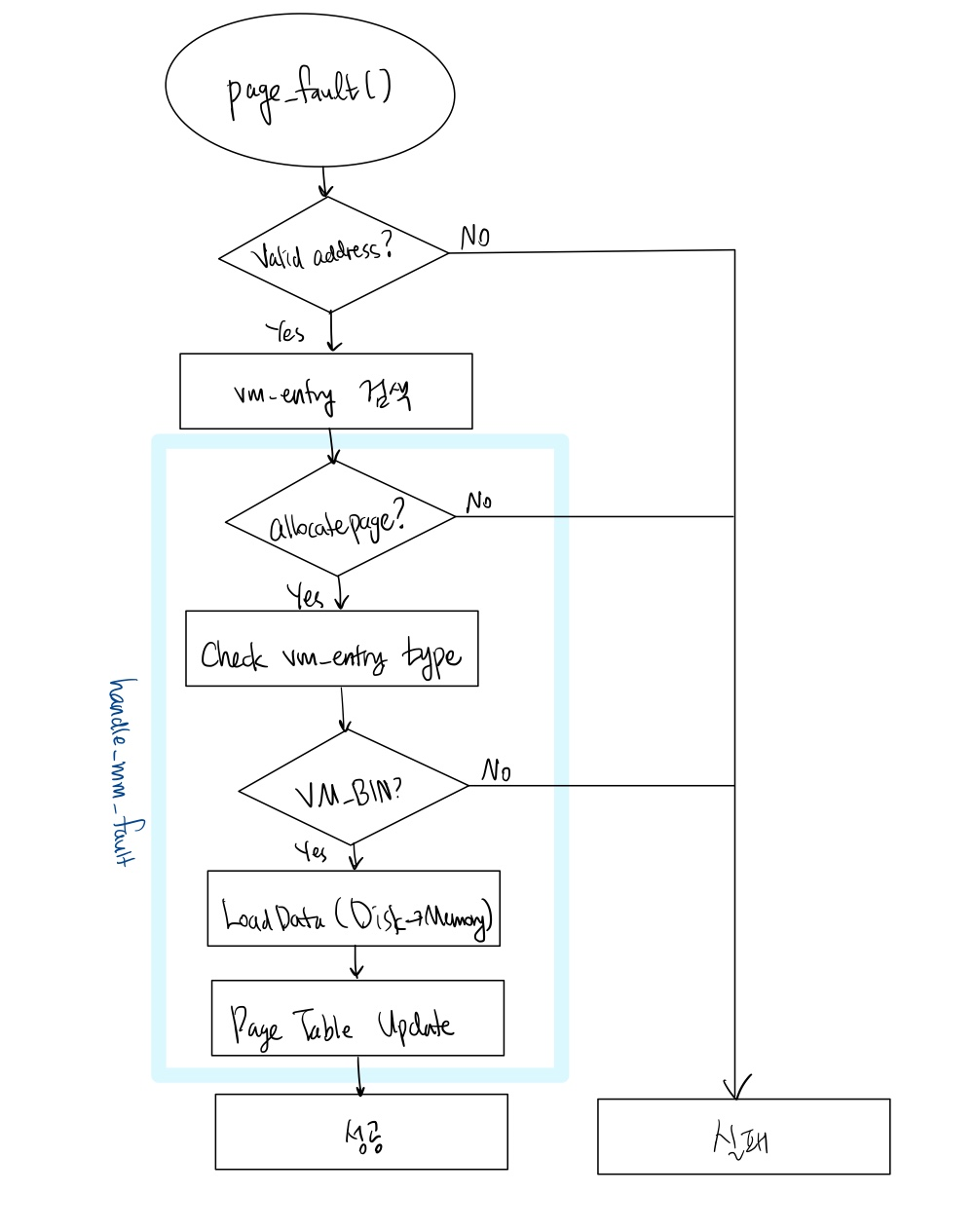
**<추가 함수>**

 pintos/src/userprog/process.c

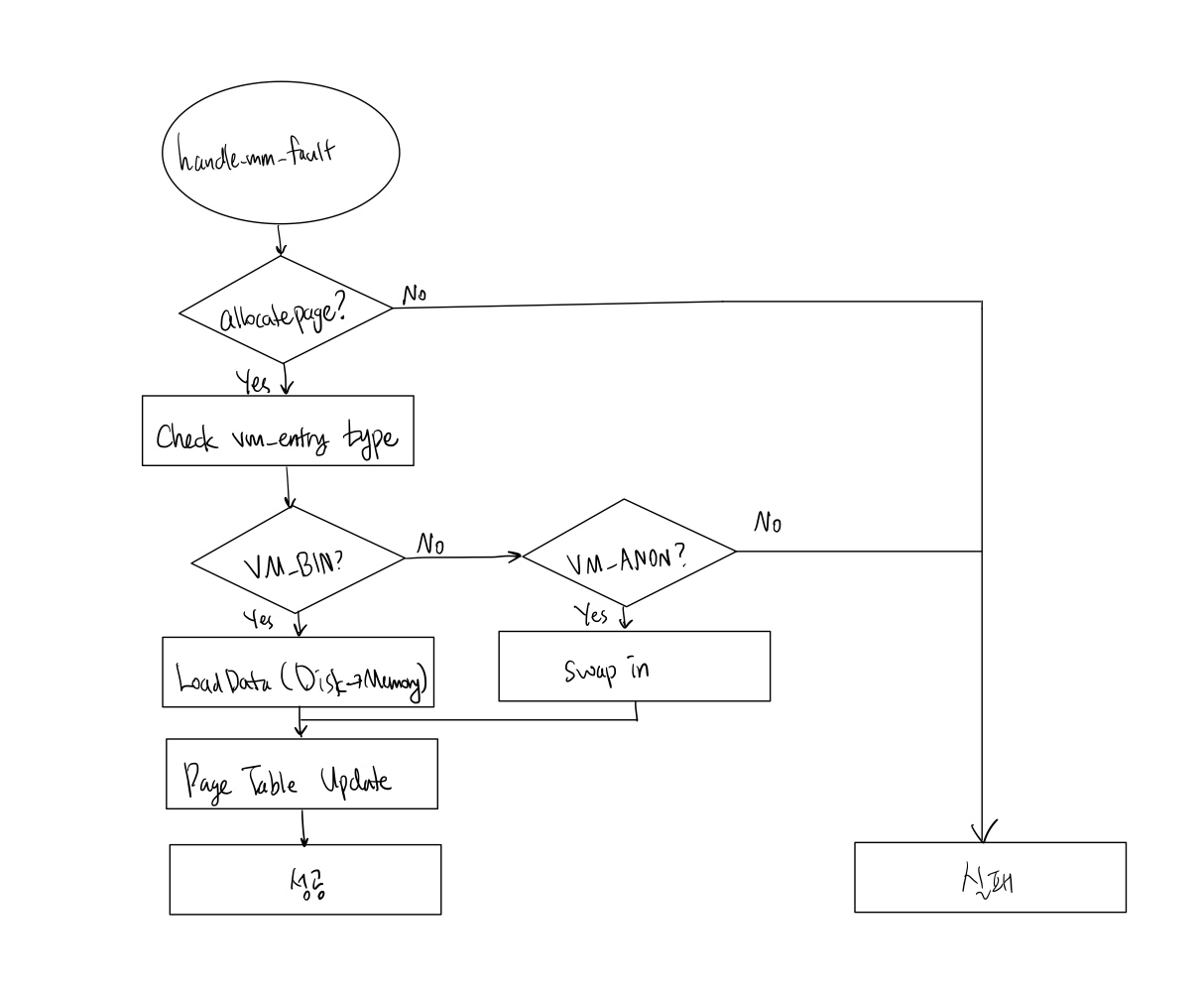
Grow limit 내의 물리 메모리 영역을 참조했는지 확인하는 verify\_stack()과, 만약 유효한 영역을 참조하였을 경우에 stack 영역을 확장해주는 expand\_stack() 함수를 구현한다.

1. **연구 결과**
   1. **Flow Chart**

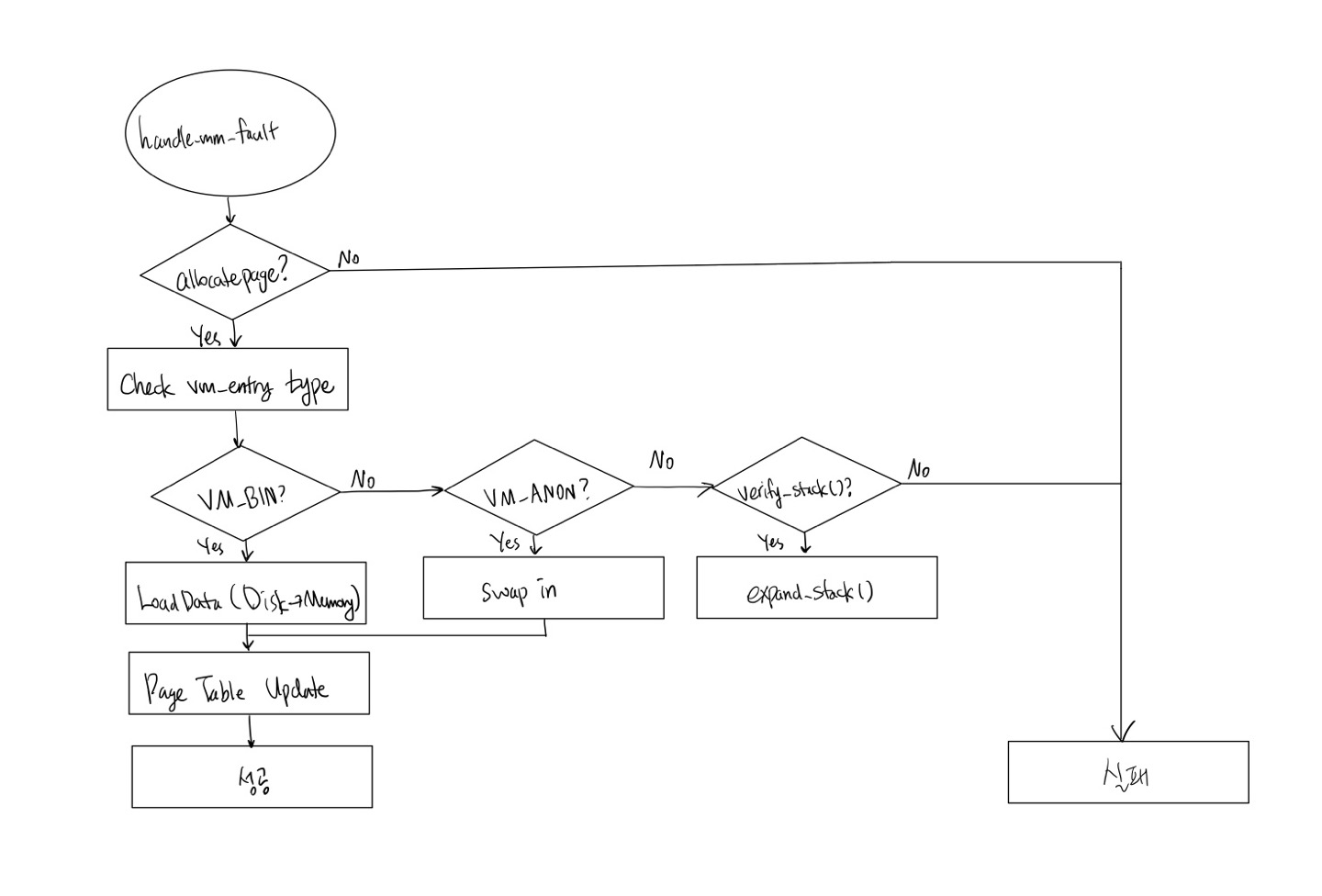
* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성
* **Page fault의 발생과 handling**



* **Disk swapping (page replacement algorithm)**



* **Stack growth**



* 1. **제작 내용**
* II. B. 개발 내용의 각 항목에 대하여 실질적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
  + 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명
* **Page fault의 발생과 handling**

<수정 파일>

 pintos/src/userprog/process.c

process.c에 handle\_mm\_fault() 함수를 구현한다. handle\_mm\_fault는 페이지 폴트 발생 시에 핸들링을 위해서 호출되는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

 pintos/src/userprog/exception.c

vm\_entry를 찾아서 조건이 맞으면 handle\_mm\_fault 함수의 load\_page를 할 수 있도록 코드를 작성한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

 pintos/src/userprog/syscall.c

syscall.c에서 가상주소의 유효성 검사를 수행한다. 시스템 콜 사용 시에 파라미터로 주어지는 string이나 buffer의 주소에 해당하는 vm\_entry가 존재하는지 검사한다. check\_user\_vaddr() 함수에 find\_vme 함수를 추가하여 해당 vm\_entry를 반환한다. check\_valid\_buffer는 buffer가 유효한지를 검사하고, check\_valid\_string은 string이 유효한 주소인지를 검사한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

 pintos/src/threads/thread.h

vm\_entry를 관리하기 위한 hash table을 thread 구조체에 선언한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<자료구조 및 추가 파일>

 pintos/src/vm/page.h & pintos/src/vm/page.c

vm\_entry 파일을 관리하기 위한 함수들을 구현한다. Page table은 user page와 kernel page가 일대일로 대응되기 때문에, vm\_entry의 탐색은 효율성을 위하여 hash table로 구현을 하였다. insert\_vme()는 해당 vme를 hash table에 insert한다. delete\_vme()는 해당 vme를 hash table에서 지운다. vm\_entry를 위한 vm\_entry structure를 구현한다. find\_vme()는 해당 가상 주소에 대응되는 vm\_entry를 찾아서 반환한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

물리메모리 할당 완료 후에 실제 디스크의 파일을 물리페이지로 load 해준다. page.c의 load\_file()에서 disk에 존재하는 page를 물리 메모리로 load하고, file\_seek와 file\_read를 이용하여 vm\_entry의 (file, offset)으로 한 페이지를 kaddr로 읽어들이는 함수를 구현한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* **Disk swapping (page replacement algorithm)**

**<수정 파일>**

 pintos/src/userprog/process.c

handle\_mm\_fault()에서 swapping을 지원할 수 있도록 수정한다. 또한, 원래 그냥 palloc\_get\_page()를 이용해서 메모리를 할당했는데, 이를 alloc\_page()라는 함수를 frame.c에 구현하고 process.c의 메모리 할당하는 부분을 alloc\_page()로 바꾼다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

 pintos/src/thread/init.c

main 내에서 swap과 lru\_list를 초기화해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**<추가 파일>**

 pintos/src/vm/page.h

Page table을 위한 page structure를 구현한다. 그리고 clock algorithm을 이용하기 위한 lru list를 만든다.



 pintos/src/vm/frame.c

frame.c에서 물리 페이지가 부족할 때 try\_to\_free\_pages를 구현하여 LRU(Clock) 알고리즘을 사용하여 여유 메모리를 확보한다. Clock이 lru\_list를 순회하면서 victim을 선정하는데, 이 clock이 순회할 수 있도록 해주는 get\_next\_lru\_clock이라는 함수를 구현한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

 pintos/src/vm/swap.c

swapping을 위한 swap\_init(), swap\_int(), swap\_out() 함수를 구현한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* **Stack growth**

**<수정 파일>**

 pintos/src/userprog/exception.c

page\_fault() 함수 내에서 page fault가 발생한 주소가 유효한 주소인지를 verify\_stack()을 이용해 확인하고, expand\_stack()을 이용해 stack 영역을 확장하는 부분을 수정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**<추가 함수>**

 pintos/src/userprog/process.c

Grow limit 내의 물리 메모리 영역을 참조했는지 확인하는 verify\_stack()과, 만약 유효한 영역을 참조하였을 경우에 stack 영역을 확장해주는 expand\_stack() 함수를 구현한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* 1. **시험 및 평가 내용**
* (채점 대상 테스트 케이스에 해당하는) make check 수행 결과를 캡처하여 첨부

텍스트이(가) 표시된 사진

자동 생성된 설명