Project #4: Virtual Memory

Operating Systems (CSE4070) Project

Fall 2021

Prof. Youngjae Kim, Prof. Sungyong Park

TA: Yonghyeon Cho (01)
Jungwook Han (02)
Hongsu Byun (03)

Overview

- **Reading pintos document is highly recommended especially for this project (pp. 39-49)**
- Basically, the 'vm' directory contains only 'Makefile's
- In project 1, 2 and 3, a program was terminated when a page fault occurs
- In this project, you will make the pintos to be more reliable from page faults and to run the programs properly
- All code you write will be in new files or in files introduced in earlier projects

Overview

- **Use the code implemented in 'User Programs' project.**

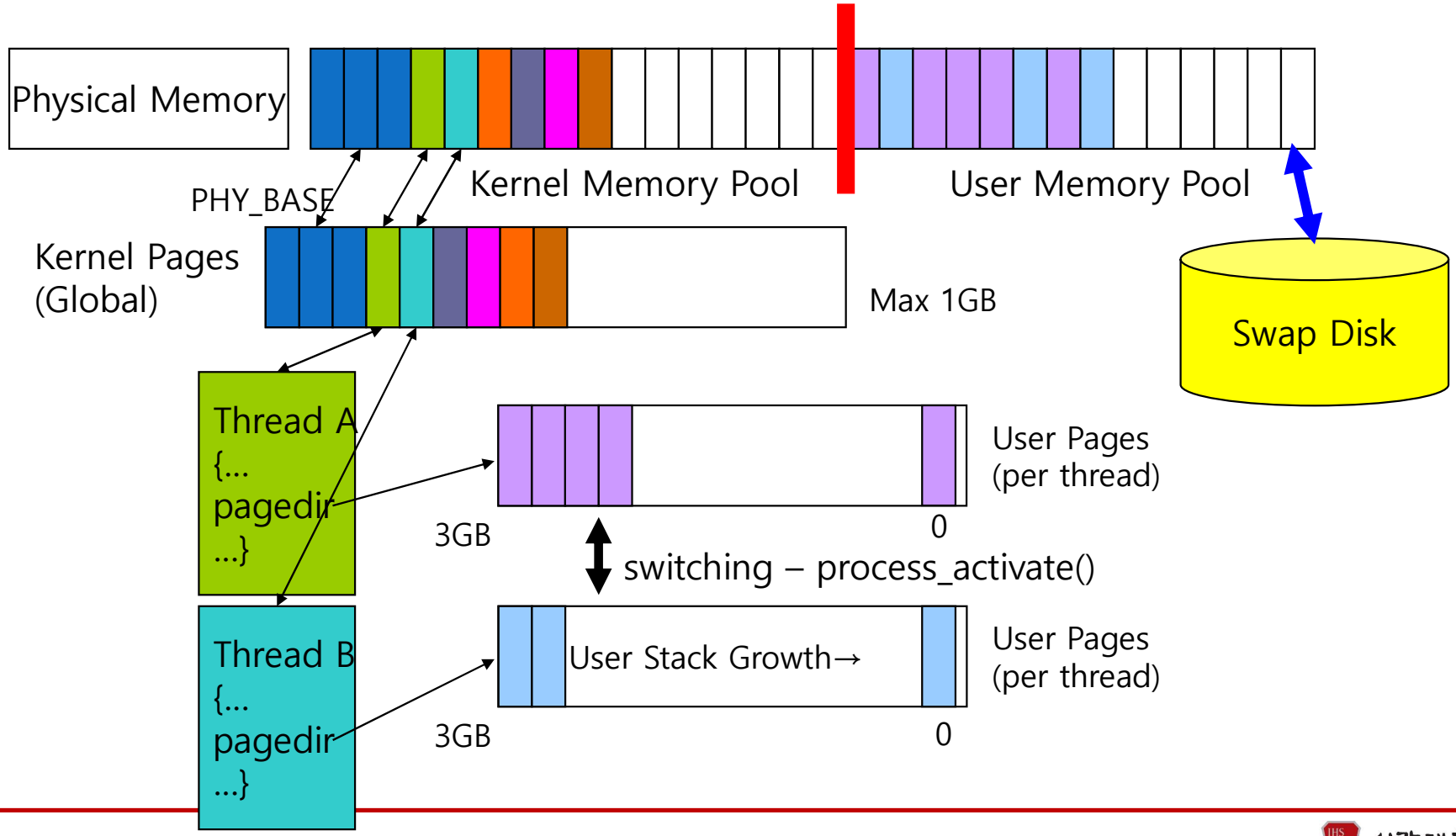
- Refer 2nd paragraph in pg.39.

> You will build this assignment on top of the last one. Test programs from project 2 should also work with project 3. You should take care to fix any bugs in your project 2 submission before you start work on project 3, because those bugs will most likely cause the same problems in project 3.

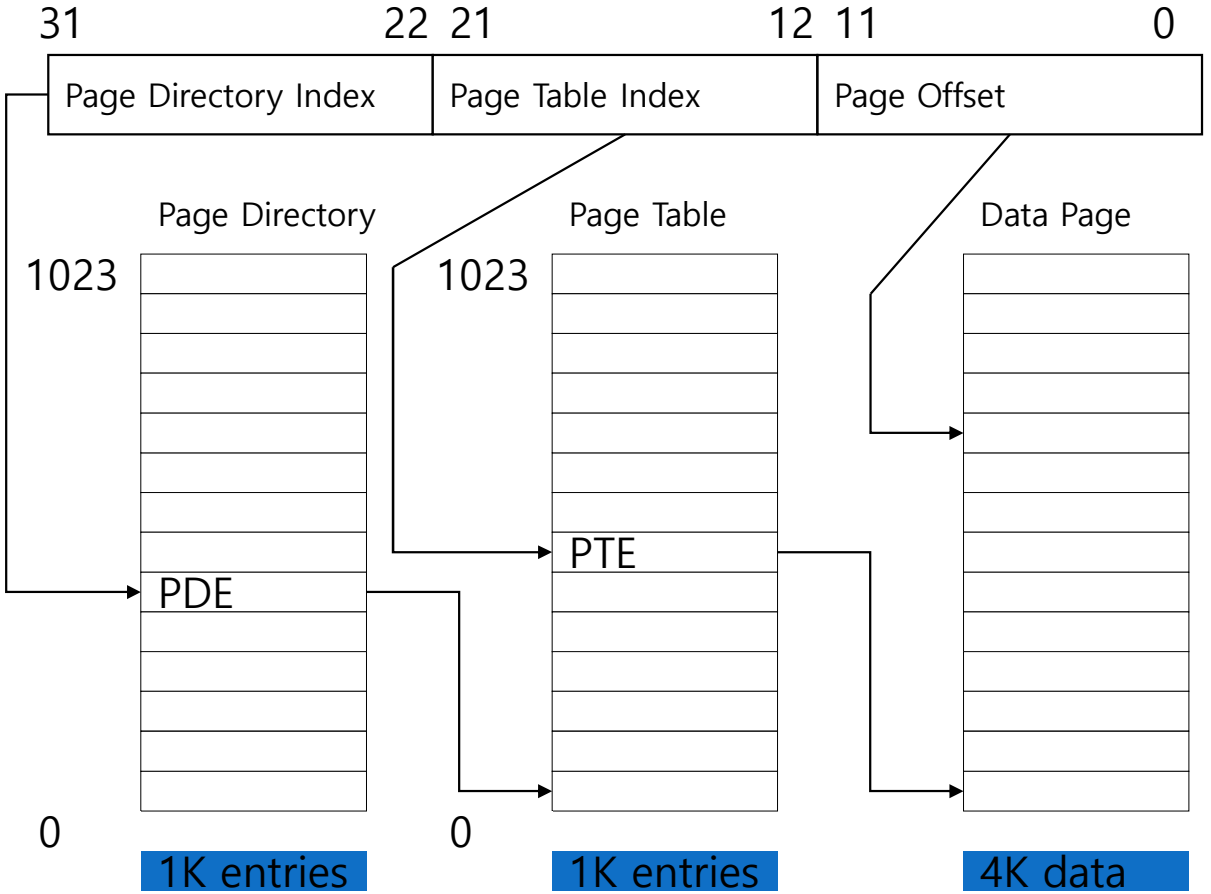
Project Requirement

- Page Table Management
  - Supplemental page table and page fault handling
- Paging to and from (swap) disk
  - Implement pseudo-LRU policies (second chance)
- Stack Growth

Virtual Memory Overview



Page Table

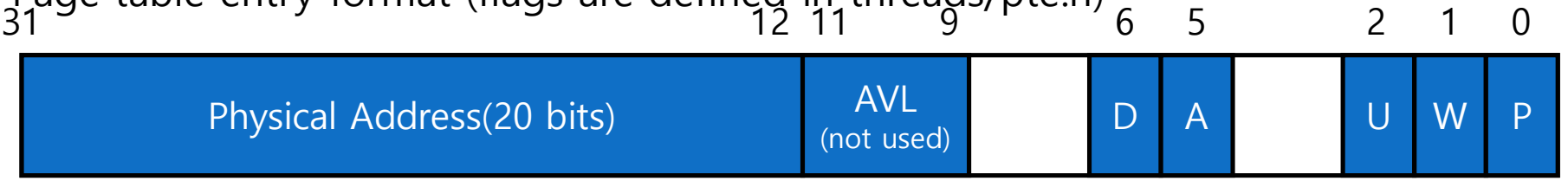


Page Table Management

- 기존 Page Table에 필요한 정보 추가
  - 주로 Page fault handling을 위한 정보를 추가
- Page Fault handler 구현

Supplemental Page Table

- Since the given page table in pintos has limitations, we need to supplement the page table with additional data about each page
- You can exploit the functions in userprog/pagedir.c to implement supplemental page table
- Page table entry format (flags are defined in threads/pte.h)



PTE_P	present bit
PTE_W	read(0)/write(1) bit
PTE_U	kernel(0)/user(1) bit
PTE_A	accessed bit
PTE_D	dirty bit
PTE_AVL	not used in pintos

Page Fault Handler

- userprog/exception.c의 page_fault()

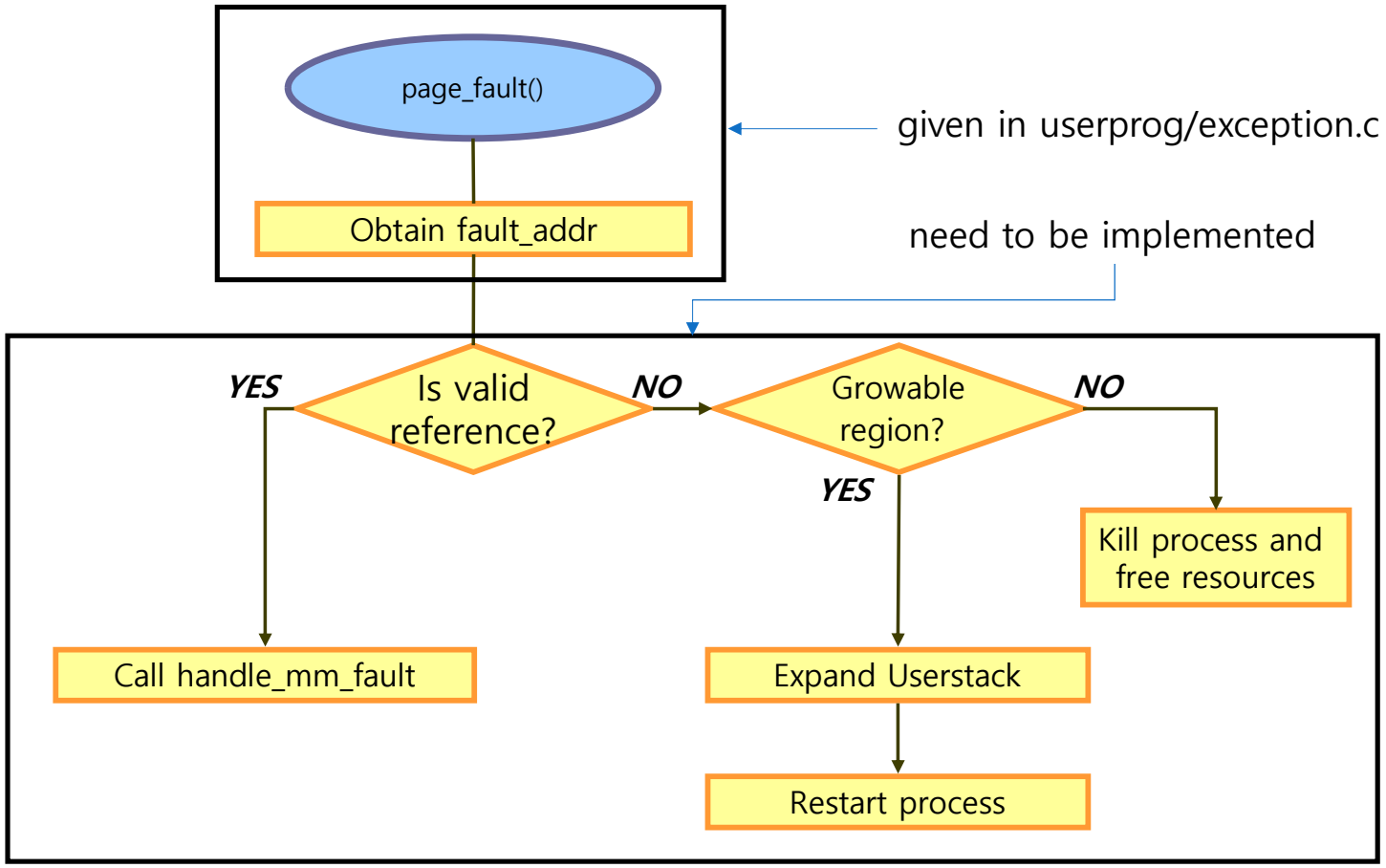
```
static void
page_fault (struct intr_frame *f)
{
  bool not_present;  /* True: not-present page, false: writing r/o page. */
  bool write;        /* True: access was write, false: access was read. */
  bool user;         /* True: access by user, false: access by kernel. */
  void *fault_addr;  /* Fault address. */

  asm ("movl %%cr2, %0" : "=r" (fault_addr));
```

- Page Fault situation
  - page from a file or swap
- Access is invalid
  - If the page is unmapped, that is, if there's no data where the page is referenced
  - If the access is an attempt to write to a read-only page (unprivileged access)
- CR2 : register storing faulted address
- Some boolean variables in page_fault() will help you
  - bool not_present; // not present in memory or rights violation
  - bool write; // write or read fault
  - bool user; // fault from user or kernel space

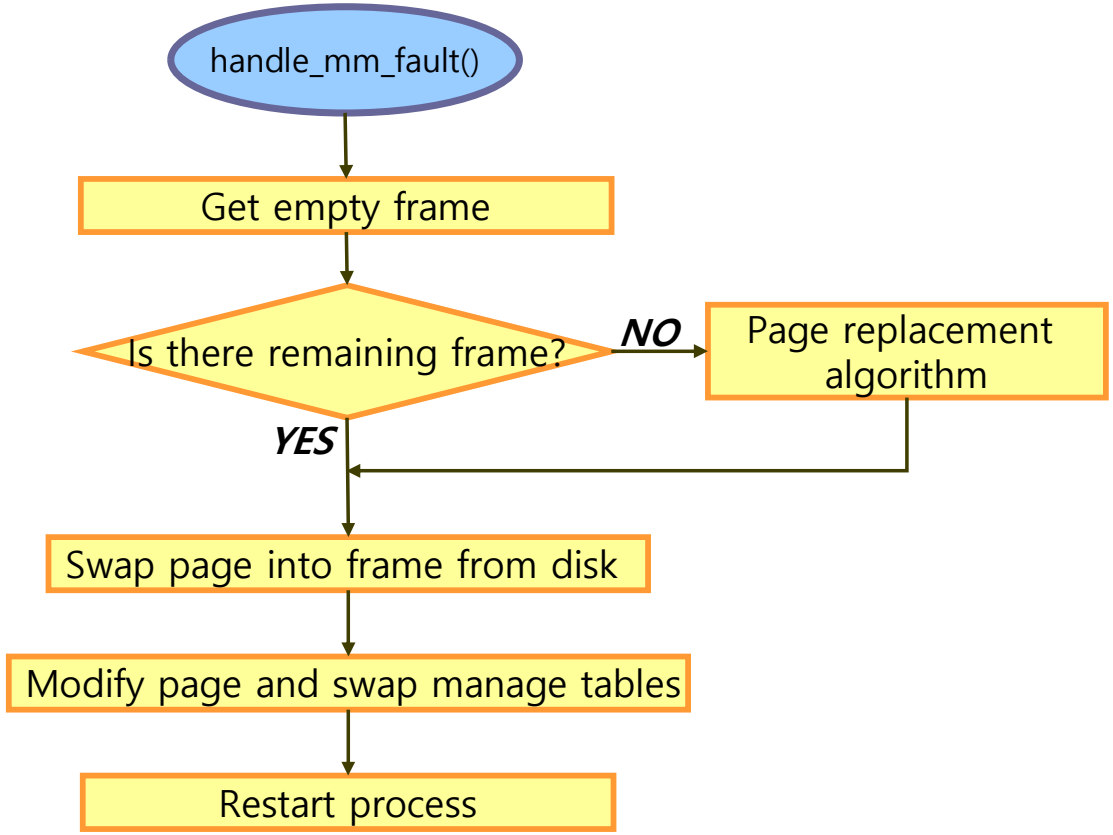
Page Fault Handler

- Page Fault handling procedures
  1. Processor (CPU) triggers page fault
  2. Control is passed to the kernel, which calls the page fault handler (userprog/exception.c:page_fault())
  3. Get the faulted address from CR2 register
  4. If the memory reference is valid
     - Obtain a frame to store the page
     - Fetch the data into the frame, by reading it from the file system or swap, zeroing it, etc.
     - Point the page table entry for the faulting virtual address to the physical page
  5. If the access is invalid
     - Any invalid access terminates the process and thereby frees all of its resources

Page Fault Handler: page_fault()



Page Fault Handler: handle_mm_fault()



Paging to and from (swap) disk

- Swap disk
  - Process에 할당해 줄 Physical memory가 부족할 때(User page pool에 free page가 없을 때) disk로 swap-out이 일어남
  - Swap 할 page의 결정은 page replacement algorithm 사용(LRU, LFU …)
- Swap table 작성
  - swap disk가 현재 사용하고 있는 슬롯과 빈 슬롯 관리
- devices/block.c 의 block_read() / block_write() 활용
- You may use the **BLOCK_SWAP block device** for swapping, obtaining the **struct block** that represents it by calling **block_get_role()**
- BLOCK_SWAP에 대해서는 devices/partition.c, thread/init.c 참조
- swap disk 생성
  - vm/build에서
    - pintos-mkdisk swap.dsk --swap-size=*n* --> swap.dsk 라는 이름으로 *n* MB swap disk 생성
    - swap.dsk는 pintos의 실행 시 자동으로 hdb1에 attach됨
  - Pintos 실행 시 argument로 '--swap-size=*n*'을 추가해도 *n* MB swap disk가 생성됨
    - swap.dsk는 pintos의 실행 시 자동으로 hda4에 attach됨

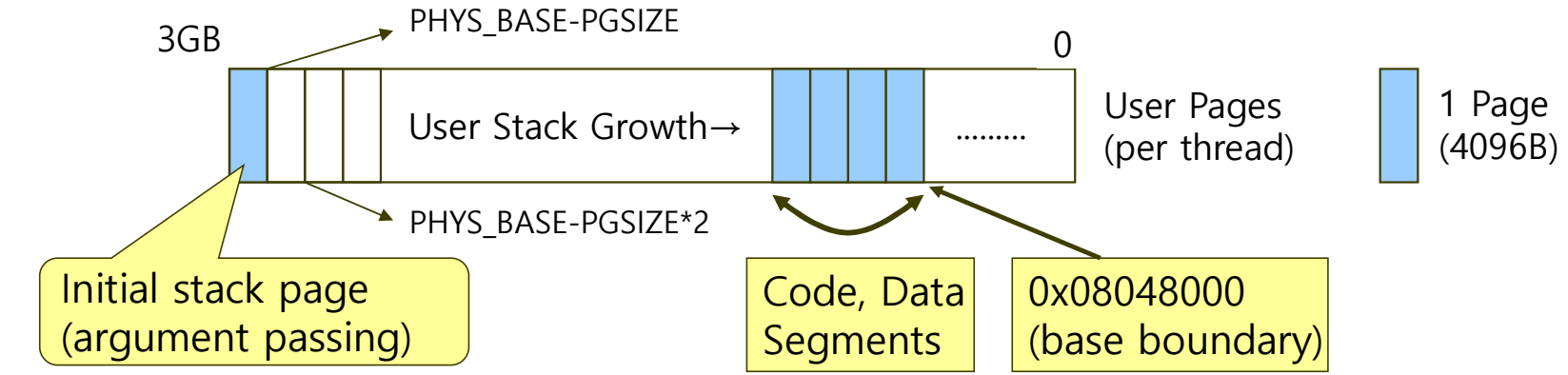
Stack Growth

- If page faults on an address that "appears" to be a stack access, allocate another stack page

- You should impose some absolute limit on stack size, as do most OSes.
  On many GNU/Linux systems, the default limit is 8 MB.

- First stack page can still be loaded at process load time (in order to get arguments, etc.)

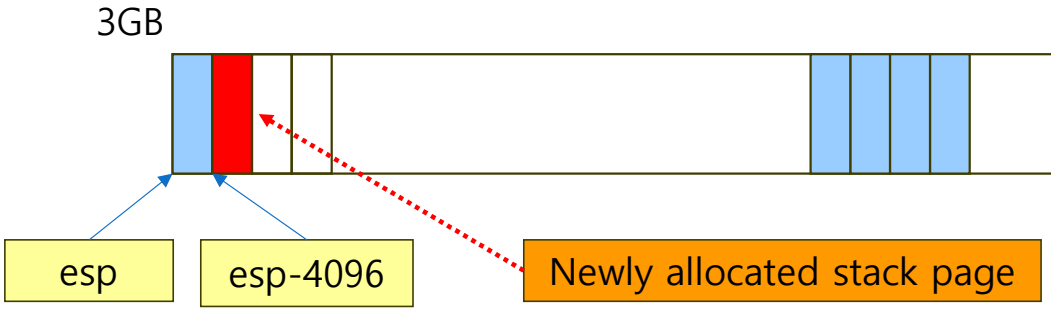
Stack Growth (Example 1)




```
22 void main(void)
23 {
24   char stack_object[4096];
25 }
```

esp = esp-4096 ;allocate stack memory

but the new esp, "esp-4096", will cause page fault

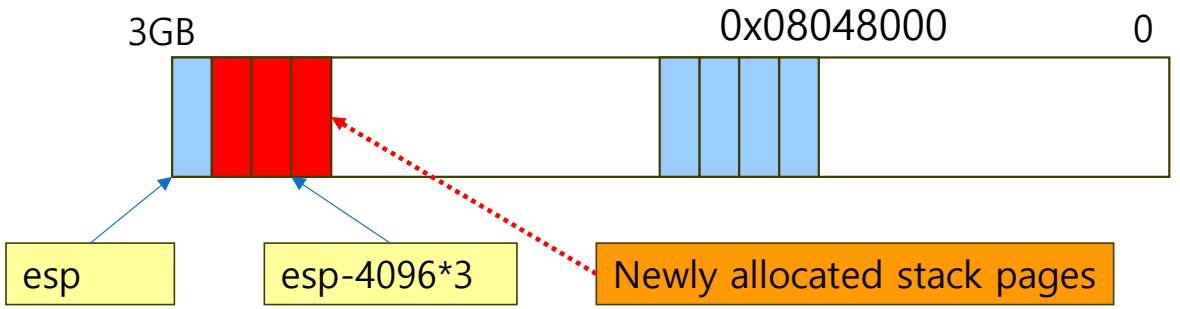


Stack Growth (Example 2)

What if user program tries to allocate more than a single PAGE size?

```
22 void main(void)
23 {
24   char big_stack_object[4096*3];
25 }
```

esp = esp-4096*3 ;allocate 3 stack pages



What about page shrinking? Do not consider about shrinking
Consider only expanding!

Reference

- You can add new files to src/vm/ if it is required.
  - **Newly added files should be written to src/Makefile.build**

```
Makefile.build       |   4
devices/timer.c      |  42 ++
threads/init.c       |   5
threads/interrupt.c  |   2
threads/thread.c     |  31 +
threads/thread.h     |  37 +-
userprog/exception.c |  12
userprog/pagedir.c   |  10
userprog/process.c   | 319 ++++++++++++++-----
userprog/syscall.c   | 545 ++++++++++++++++++++++++++++-
userprog/syscall.h   |   1
vm/frame.c           | 162 +++++++++
vm/frame.h           |  23 +
vm/page.c            | 297 ++++++++++++++++
vm/page.h            |  50 ++
vm/swap.c            |  85 ++++
vm/swap.h            |  11
17 files changed, 1532 insertions(+), 104 deletions(-)
```

You can follow this approach if you want

Evaluation

- 16 tests will be graded.

  (Refer to the test case list in the next slide)

- Total score is 100 which consists of 80 for test cases and 20 for documentation.

- Refer to

  src/tests/vm/Grading

  src/tests/vm/Rubric.functionality

  src/tests/vm/Rubric.robustness

  to check the points of each test.

Evaluation

Functionality		
No.	Test Case	Point
1	pt-grow-stack	3
2	pt-grow-stk-sc	3
3	pt-big-stk-obj	3
4	pt-grow-pusha	3
5	page-linear	3
6	page-parallel	3
7	page-shuffle	3
8	page-merge-seq	4
9	page-merge-par	4
10	page-merge-mm	4
11	page-merge-stk	4
Total		37

Robustness		
No.	Test Case	Point
1	pt-bad-addr	2
2	pt-bad-read	3
3	pt-write-code	2
4	pt-write-code2	3
5	pt-grow-bad	4
Total		14

Documentation

- Use the document file uploaded on e-class
- Documentation accounts for 20% of total score.
  (Development 80%, Documentation 20%)

Submission

- Make 'ID' directory and copy **'src' directory** in the pintos directory and the document file (**[ID].docx**).
- Compress 'ID' directory to '**os_prj4_[ID].tar.gz**'.
- We provide the script 'submit.sh' to make tar.gz file which contains 'src' directory and document file.
  학생들의 편의를 위해 pintos 디렉토리 내 submit.sh 스크립트를 제공합니다.
  이 스크립트는 src 디렉토리와 document file을 포함한 tar.gz 파일을 생성합니다.
- **Disclaimer**
  - **Any result produced from the 'submit.sh' script is at your own risk.**
  - **You must check the contents of the tar.gz file before submission.**
  - **'submit.sh' 스크립트로 생성된 결과의 모든 책임은 사용자에게 귀속됩니다.**
  - **제출하기 전, tar.gz 파일의 내용물을 반드시 다시 한 번 체크하기 바랍니다.**

Submission

- **Notice – 'submit.sh'**
  - The 'submit.sh' script should be executed on a directory where 'src' folder is located.
    submit.sh 스크립트는 src 폴더가 위치한 디렉토리에서 실행되어야 합니다.
  - 'ID' folder should not be in the directory.
    해당 디렉토리에 '학번' 폴더가 없어야 합니다.
  - 'ID.docx' file should be located in the directory.
    Also, report file with extensions other than 'docx' will not be compressed.
    해당 디렉토리에 '학번.docx' 파일이 있어야 함께 압축됩니다.
    또한 'docx' 이외의 확장자를 가진 보고서 파일은 압축되지 않습니다.
  - Be sure to backup your code in case of an unexpected situation.
    만일의 경우를 대비해 반드시 코드를 백업하여 주세요.

Submission

- It is a **personal project**.
- Due date : **2021. 11. 27 23:59**
- Submission
  - The form of submission file is as follows:

Name of compressed file	Example (ID: 20191111)
os_prj4_[ID].tar.gz	os_prj4_20191111.tar.gz

  - No hardcopy.
  - **Copy will get a penalty (1st time: 0 Point and downgrading, 2nd time: F grade)**

Submission

- **Contents**
  - ① Pintos source codes (Only **'src' directory** in pintos directory)
    최소한의 용량을 위해 **src 디렉토리만** 압축파일에 포함합니다.
  - ② Document: [**ID].docx** (e.g. 20191111.docx; Other format is not allowed such as .hwp)
- **How to submit**
  1) Make tar.gz file.
     - Copy the document file ([ID].docx) to pintos directory.
     - **Execute submit.sh script in the pintos directory and follow the instructions of the script.**
       pintos 디렉토리 내의 submit.sh 스크립트를 실행하고 스크립트의 지시를 따르십시오.
     - Check that **os_prj4_[ID].tar.gz** is created.
     - Decompress **os_prj4_[ID].tar.gz** and check the contents in it. ($ **tar -zxf os_prj4_[ID].tar.gz**)
       (Only **[ID].docx** and **src** directory should be contained in the tar.gz file.)
     - For example, if your ID is 20191111, os_prj4_20191111.tar.gz should be created.
       To decompress the tar.gz file, execute **tar -zxf os_prj4_20191111.tar.gz**
     - **Please check the contents of tar.gz file after creating it.**
  2) Upload the **os_prj4_[ID].tar.gz** file to e-class.