Project 3 - Memory Management

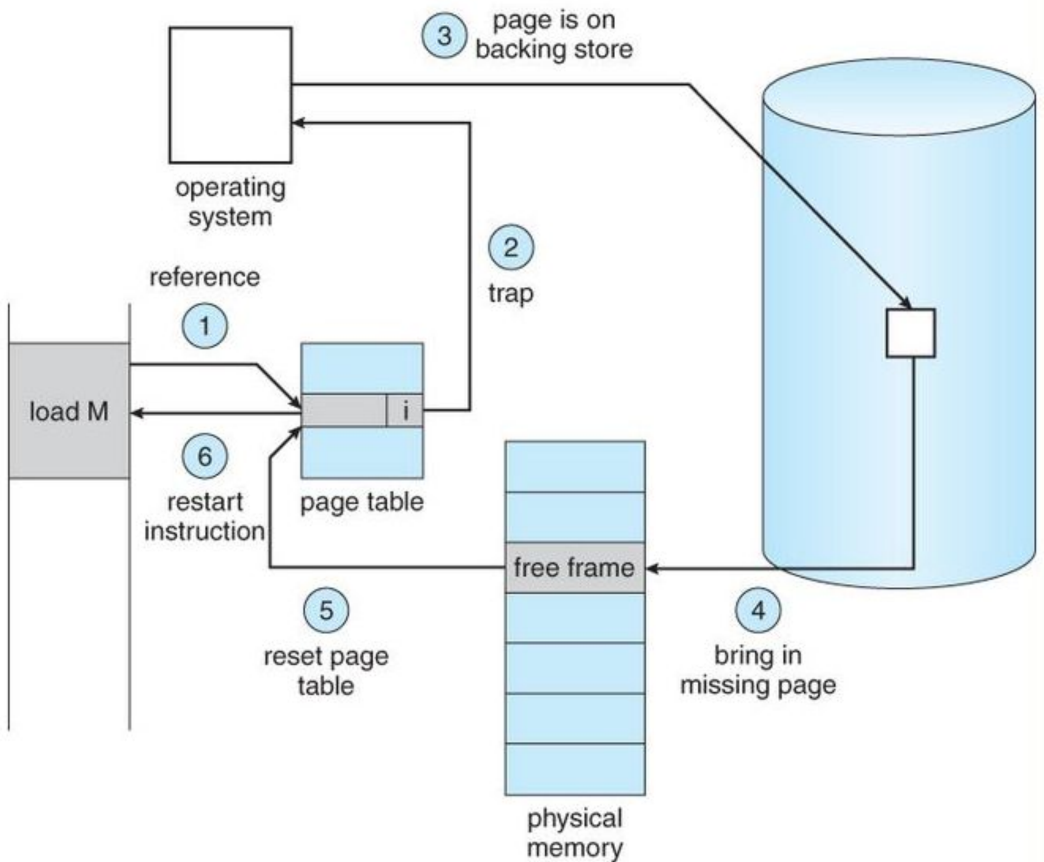
Goal of this project

- Both the test cases below require lots of memory.
  - /test/matmult.c
  - /test/sort.c
- Goal:
  - Run this two programs concurrently, and get the correct result.
  - Remember to specify the scheduling algorithm you used in the report.
  - DO NOT pass by simply modify the memory size in machine.h

Goal of this project (cont.)

- ./nachos -e ../test/matmult
  - Should return 7220
- ./nachos -e ../test/sort
  - Should return 1
- ./nachos -e ../test/matmult -e ../test/sort
  - Should return 7220 and 1

Demand paging



Hints

- File system - swap space
- Maintain three tables to record the information
  - PageTable
  - FrameTable
  - SwapTable
- Think how to “load” the page correctly and deal with the PageFaultException
- Design your own “Virtual Memory Manager”

Maintain three tables

- **PageTable**
    - One page table per process
    - Decide your virtual page number
- **FrameTable**
    - Each frame represent one physical page
    - Record every physical page’s information
- **SwapTable**
    - Each entry represent one frame in the disk
    - Record every sector’s information in swap

PageTable

```
TranslationEntry {
    unsigned int virtualPage; //virtual memory page
    unsigned int physicalPage; //if in physical memory
    bool valid; //if in physical memory
    bool use; //been used(read or write)
    bool dirty; //been modified
};
```

FrameTable

```
FrameInfoEntry {
    bool valid; //if being used
    bool lock;
    AddrSpace *addrSpace; //which process is using this page
    unsigned int vpn; //which virtual page of the process is stored in
                             //this page
};
```

SwapTable (same as the FrameTable)

```
FrameInfoEntry {
    bool valid; //if being used
    bool lock;
    AddrSpace *addrSpace; //which process is using this page
    unsigned int vpn; //which virtual page of the process is stored in
                            //this page
};
```

Addrspace::Load

- First ask for 1 page, if the FrameTable is full, select 1 frame and put it into SwapTable
  - Design your own page replacement method to get the frame
  - Specify the method in your report
- Mapping virtual address to physical address
- Invoke executable->ReadAt(&(kernel->machine->mainMemory[physical address]), sizeToLoadNow, inFileAddr)

Address mapping

- Map the Virtual Address to Physical Address
- Like the way you used in Project 1
- physicalAddr = pageTable[(virtualAddr / PageSize)].physicalPage * PageSize + (virtualAddr % PageSize)

Page-fault handler

- Put the pages in SwapTable into FrameTable
- When all physical memory frames are occupied, design a page replacement method to get a frame
    - Remember to specify it in your report!
- You can do this work in your own way

Memory manager (optional)

- You can design your own manager

```
Class MemoryManager{
      int TransAddr(AddrSpace *space, int virtAddr);
      // return phyAddr (translated from virtAddr)

      Bool AcquirePage(AddrSpace *space, int vpn);
      // ask a page(frame) for vpn

      Bool ReleasePage(AddrSpace *space, int vpn);
      // free a page

      Void PageFaultHandler();
      // will be called when manager want to swap a page from
      // SwapTable to FrameTable and the FrameTable is full
};
```

Some files that might be useful

- For the disk usage details, see:
  - /filesys/synchdisk.*
- For the swap space initialization:
  - /userprog/userkernel.*
- For the table maintaining, see:
  - /machine/machine.*
- For the loading of pages:
  - /userprog/addrspace.*

Some files that might be useful (cont.)

- Always see the header and comments first
- Based on your implementation, there might be different files that you need to see and modify

Report

- Motivation
    - Motivation and the problem analysis
    - Wha't your plan to deal with the problem
- Implementation
    - How do you implement to solve the problem in NachOS
    - You can include some of your code and explain it
- Result
    - Experiment result and some discussion
    - Extra effort or observation
- Save your report as the format report.pdf

Submission format

- Create a folder for the source code and report
  - {Student ID}_nachos3/
    - nachos-4.0/
    - report.pdf
- Submission format: {Student ID}_nachos3.tar.gz
  - e.g. r11223344_nachos3.tar.gz
- Compress your folder
  - `tar zcvf {Student ID}_nachos3.tar.gz {Student ID}_nachos3/`
- Submit your compressed file to NTU cool
- Deadline: 2022/12/16 23:59

Grading policy

- NachOS source code:(40%)
- Report: (40%)
- Correct format: (20%)
- No plagiarisim

Late policy

- 10% penalty per day
- After 7 days, you will get 70% penalty, but no more penalty after that