OS 4/9

Reagan Shirk

April 9, 2020

Chapter 9

- The CPU has access to the MBR, MAR, and I/O registers
  - It doesn't really see the memory
- The load functions requests some memory through MAR or I/O
- So far, we've been assuming that the memory box has all of the memory we need (picture to be added at some point of this), but that's not exactly the case
- Principle of Locality (missed what he said about this)
- There are two primary things that the CPU does: fetch and execute
  - Fetch is slow, execute is really fast
- Memory fault: request for a memory address and it is not there (also known as trap)
- Given two processes, Process A and Process B, what happens when you try to malloc memory in the middle of Process A's memory?
  - Each process has a base address and an offset, so we can determine where an address is using those things
- Operating Systems are in charge of memory protection
  - If Process B tries to access Process A's memory, there will be a trap/fatal error: invalid memory access. Everything should crash
- In summary, the OS is responsible for:
  - Translate memory requests quickly
  - Memory protection

Address Binding

- Memory from source programs have symbolic addresses
- You start with your source program that has the symbolic addresses
- You go through the compiler to create object files
- The object files go through a linker to create executable files
- Your executable files go through a loader to create the running program
- Going through the linker is called load time, this creates absolute addresses
- Going through the loader is called run time, brings the shared libraries into memory
- Every program that runs C has the `<stdlib.h>` installed, which means that they also have the location of routines and variables for use. They have all of the object files needed for that library

Shared Libraries

- Shared Libraries are useful for redundant routines
  - Instead of having these routines added at linking time, they're added at loading time
  - `.dll` files are dynamically loaded libraries
  - `.so` files: shared object files
    - These are small executables and are shared among many processes

Fixed Partitioning

- Physical memory is permanently cut into fixed-sized paritions
  - Each process is allocated one of the free partitions
- Disadvantages:
  - A program may be too big to fit into one parition, will need to do overlays (using multiple paritions for one memory address which may not be desirable)
  - Utilization is inefficient; you must use the whole partition every time which can lead to wasted space
    - Leads to **internal fragmentation**: wasted space due to the block of data being smaller than the partition
  - Number of partitions is fixed

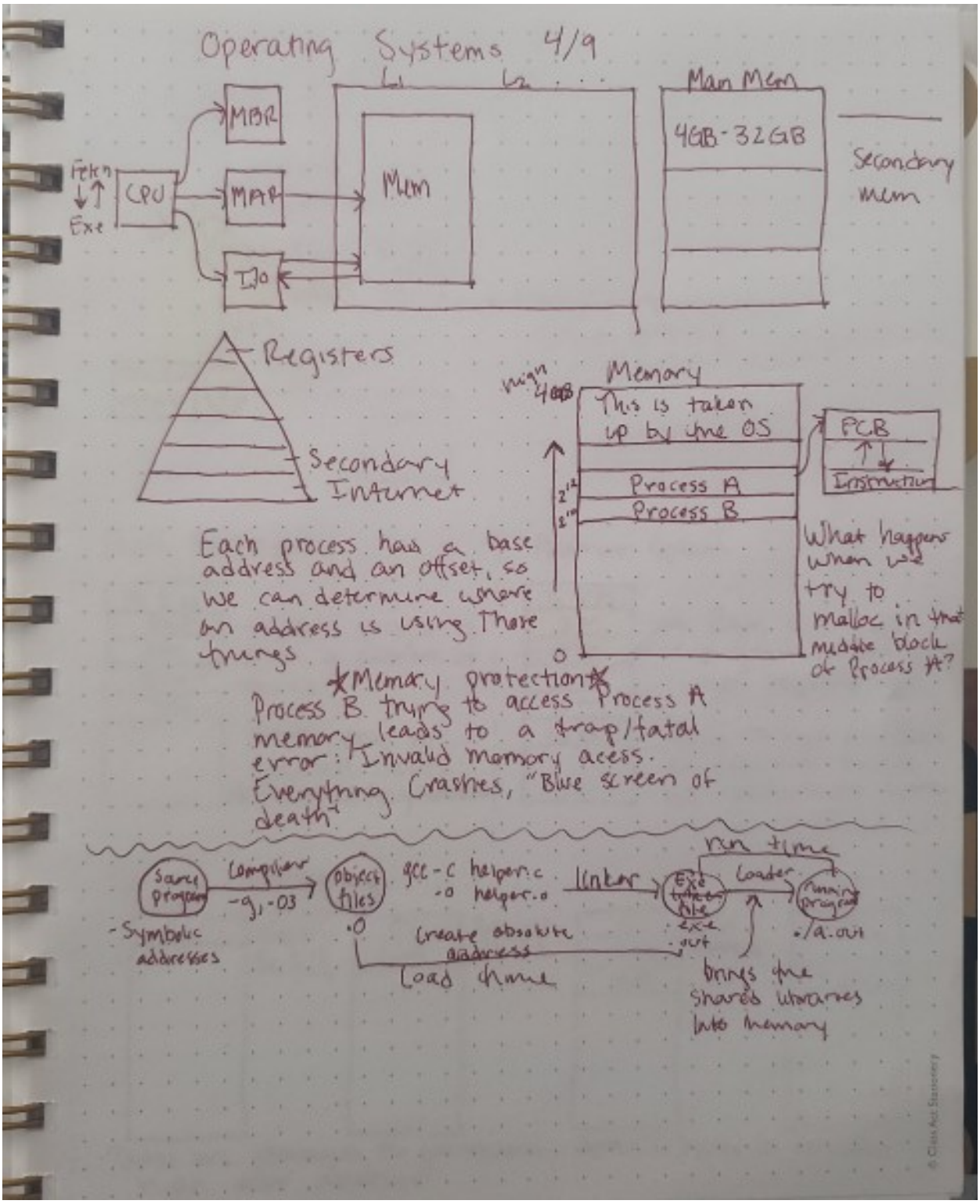
Dynamic Partitioning this is what project 2 will be

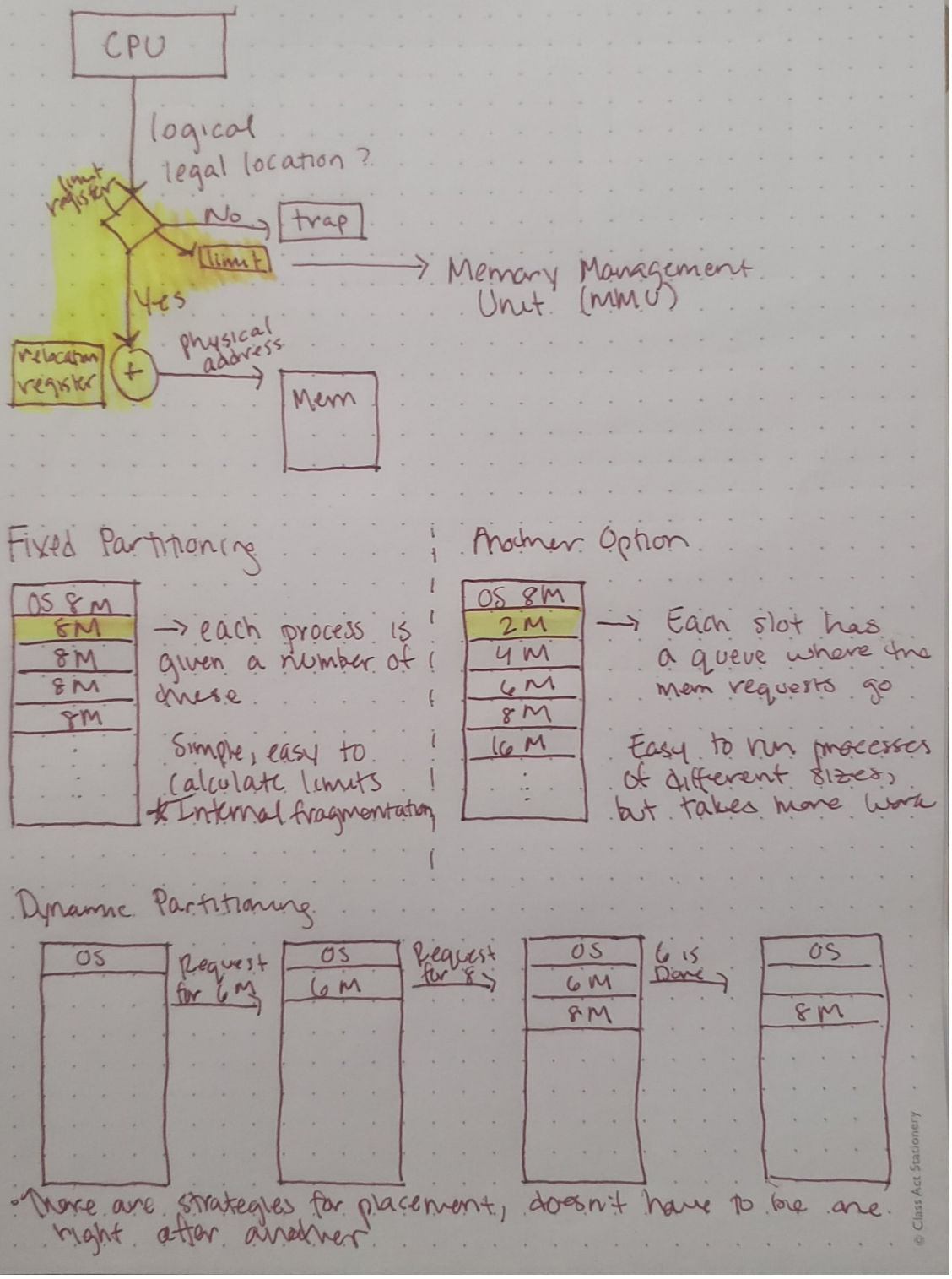
- Partitions are a variable lengths and the lengths can change depending on the process
- Each process will be allocated only exactly what it needs

Project 2

- Ways to allocate memory: first fit, best fit, worst fit, next fit
- The program will take in three commands: string that says which one to use, integer that tells you how much memory you have to use, text file with commands that are described in the project assignment (each command will be one line, ignore the new line that the chart might create for you)
- You can simulate or you can manually do the memory allocation, just make sure you have a game plan
  - Allocation at the top = address 0, allocation at the end = address N - n

Pictures/Diagrams




CPU
logical
legal location ?
limit register
No
trap
limit
Memory Management Unit (MMU)
Yes
relocation register
+
physical address
Mem
Fixed Partitioning
OS 8M
8M
8M
8M
8M
→ each process is given a number of these
Simple, easy to calculate limits
* Internal fragmentation
Another Option
OS 8M
2M
4M
6M
8M
16M
→ Each slot has a queue where the mem requests go
Easy to run processes of different sizes, but takes more work
Dynamic Partitioning
OS
Request for 6M
OS
6M
Request for 8
OS
6M
8M
6 is Done
OS
8M
• There are strategies for placement, doesn't have to be one right after another
© Class Act Stationery