# Phase 2 help

***Process Loading:**

- Read files one by one. (Each file is for a program)
- Memory is divided into frames of size 128 bytes.
- Each process would have two page tables.
- After writing 128 bytes of the data of the program (i.e. a page) to a frame, assign that frame to that page on the data page table of that process.
- Same goes for code, after writing a page of the code to a frame, assign that frame to that page on the code page table of that process.
- Check code and data size is according to the limit specified.
- After loading a process into memory, create its PCB (i.e. a class), which have all the process's info including the reference to the page table you have created for that process. Put the reference to that class into a ready queue according to the priority given.
- After loading all the processes, add remaining frames to free page table.

**=>File Structure:**

1. 1st byte: priority.
2. 2nd & 3rd byte: Process ID (PID)
3. 4th & 5th byte: data size
4. 6th, 7th, 8th bytes: ignore them
5. $8^{th}$ onwards is data and data will be loaded in data segment according to the size given
6. After data, code is given in the files

At this point, you have loaded processes, divided memory into frames, assigned pages in page tables and you have processes into queues.

**CPU Scheduling:**

- There could be a function or class of scheduler which would tell which process to execute next.
- That scheduled process's registers (i.e. stored in PCB) are mapped to actual registers of the processor.
- Move that process (i.e. its PCB's object reference) into running queue (contains only one process).
- Run the process (this you have done in the previous assignment)
- After its time quantum finishes (if it is 8 cycles, then it becomes 4 instruction), remove that process from running queue and store its state (i.e. replaces its PCB's registers with actual processor's registers).
- Update every process's running time and waiting time and call scheduler, which may rearrange the queue if need and give you next process to run.

(Note: * = updated).

**Process Termination:**

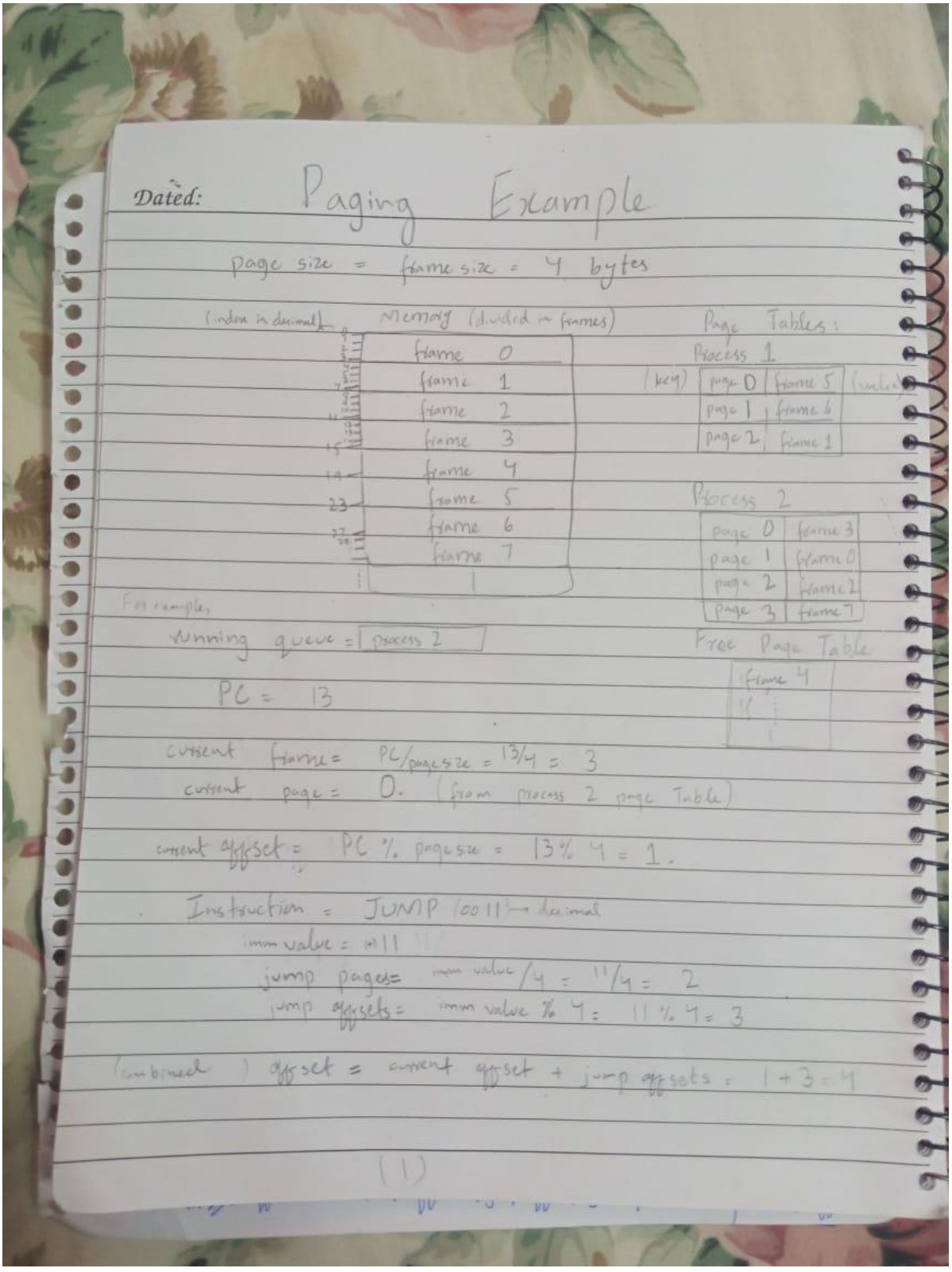
- After a process terminates, either because of an error or successfully, print its memory dump (its memory contents) and PCB on screen and into a file as well.
- Remove that process from ready queue.
- Add its frames (from its page table) to free page table.
- Reinitialize quantum and schedule another process

**Memory Management:**

Memory is not contiguous rather divided into segments (pages), therefore when jumping to an instruction, or next instruction pointing to another frame, or accessing data of that process, you need to check whether the corresponding page is of that process or not.

(PTO)

***Paging Example:**



Dated:

if offset >= page size (4) {
jump pages = jump pages + 1 = 2+1 ⇒ 3
offset = offset % pagesize (4) = 4 % 4 ⇒ 0
}

page No (page to jump) = current page + jump pages
= 0 + 3 ⇒ 3

→ check if process have frame of that page in Page Table.
if yes, frame No. (frame to jump) = 7 (from page table of process 2)
else (error)

(value of) PC = (frame No. × frame size) + offset

PC = (7 × 4) + 0 ⇒ 28