Readme file

Group Project Members

1. **Naveen Kumar Reddy Inaganti(Net Id: NXI180004).**
2. **Kiran Noolvi(Net Id: KXN180017)**

Contribution

- **loader.c, swap.c, cpu.c and process.c - Kiran Noolvi**
- **paging.c and cpu.c - Naveen Kumar Reddy Inaganti**

Points

1. **Loader.c**
   a. **load_process_to_swap function**
      - The given program file is read for the number of instructions and the data.
      - Number of pages required for the process is calculated
      - For each page of the process, A buf having size of pageSize is created, Instructions/data is stored in the buf and the buf is written to the disk using insert_swapQ function, Also the page table of the process is updated with the information of where the page of the process is stored presently.

   b. **load_pages_to_memory**
      - load_memory_from_loader functions is called for initial pages load to memory when a file is submitted.

   c. **load_instruction function**
      - Prints each instruction which is given in the program file.

   d. **load_data function**
      - Prints each data which is given in the program file.

2. **Paging.c**
   a. **calculate_memory_address**
      - The addr is calculated and returned to the calling function.
      - If the page of the process executing in the CPU is nullpage in the page table pointer and read request has been made, mError is returned.
      - If the page of the process executing in the CPU is diskpage/nullpage in page table pointer and write

request has been made then page fault interrupt is set and mPFault is returned.

b. **Get_data**
   - Based on the address received data from memory is copied to CPU.MBR.

c. **Put_data**
   - Based on the address received data from CPU.MBR is copied to Memory, The respective frame of memory is made as dirty.

d. **get_instruction**
   - Based on the address received the instruction is retrieved from memory, From the instruction opcode and operand are separated. Opcode is saved in CPU.IRopcode and operand is saved in CPU.IRoperand.

e. **Dump_one_frame**
   - One frame based on findex is printed.

f. **dump_free_list**
   - From the entire memory frames, The list of frames which are free are printed.

g. **update_frame_info**
   - A particular frame based on findex is updated with the respective pid, page of process, highest age and set as used frame.

h. **addto_free_frame**
   - A particular frame is added to the free list**.**

i. **get_free_frame**
   - A frame which is free is returned, If there are no free frames then the frame with lowest age is returned.

j. **initialize_memory**
   - The memory is initialized.

k. **update_process_pagetable**
   - Information regarding the location of a particular page of the process is stored in the process page table.

l. **free_process_memory**
   - After a process has finished, The memory frames used by the process are added to free list.

m. **dump_process_pagetable**
   - The pagetable of the process is printed.

n. **page_fault_handler**

- Whenever called the required page is read from the disk to the memory using insert swapq, If the frame which is returned from the get_free_frame is dirty then first the contents of the memory are stored back to disk and then the required page is read from disk to the memory.

o. memory_agescan

- Age of each memory frame is right shifted by 1 bit.
- If the age becomes zero after the right shift, The frame is added to free frame if it is not free.

p. load_memory_from_loader

- Called by load_pages_to_memory for initial pages load to memory when a file is submitted.

q. select_agest_frame

- Called by get_free_frame if no free frame is available, a frame with lowest is selected and returned as said requirements doc.

3. Swap.c

a. read_swap_page

- Used the same code given in Project 3 swap.c function.
- Disk mutex(disk_mutex) is used to make sure that while read_swap_function is executed, write_swap_page/ dump_process_swap_page will be waiting.
- This function reads from the disk and loads to the memory.

b. write_swap_page

- Used the same code given in Project 3 swap.c function.
- Disk mutex(disk_mutex) is used to make sure that while write_swap_page is executed, read_swap_page/ dump_process_swap_page will be waiting.
- This function writes data in the buf to disk.

c. dump_process_swap_page

- Used the same code given in Project 3 swap.c function.
- Disk mutex(disk_mutex) is used to make sure that while dump_process_swap_page is executed, read_swap_page/ write_swap_page will be waiting.
- This function prints the contents of the process page.

d. process_one_swap

- The thread will be waiting on swap_semaq semaphore until insert_swapQ signals it.
- Using swapq_mutex the region between if(swapQhead==NULL) and the end of the function is made mutually exclusive.

e. Semaphore initialization

- swap_semaq is initialized to 0
- disk_mutex and swapq_mutex is initialized to 1.

4. cpu.c

a. handle_interrupt

- If the interrupt is pFaultException then page_fault_handler() is called and then the interrupt pFaultException is cleared.
- If the interrupt is actAgeInterrupt then memory_agescan() is called and then the interrupt actAgeInterruptthen is cleared.