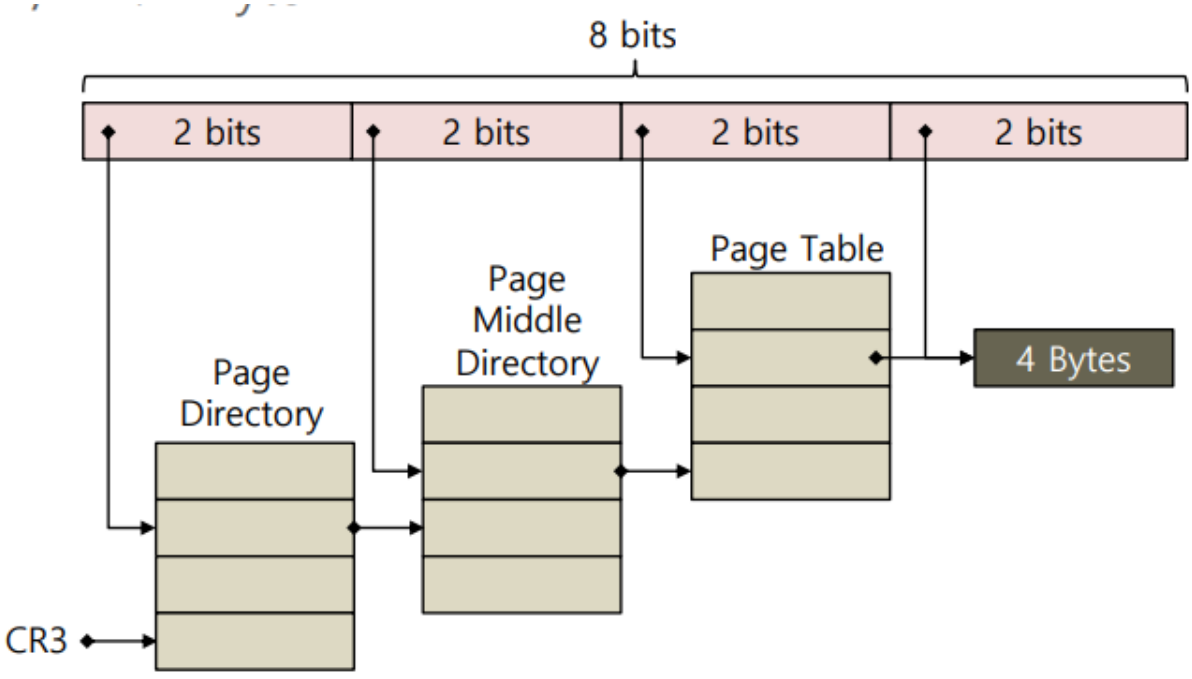
Operating Systems MMU 구현

컴퓨터공학과

201611297 정제윈

1. 구현 내용

리눅스 상에 구현되어 있는 MMU를 직접 시뮬레이션 하는 기능을 직접 구현한다. 운영체제는 MMU에 맞게 다양한 기능들을 구현해야한다. 본 프로그램에서는 운영체제가 하는 역할을 응용프로그램으로 구현한다. 가상메모리가 들어오면 그 가상 메모리를 물리 메모리로 바꿔줘야 하는 Address Translation이 일어난다. 우리는 8bit addressing을 관리할 것이다. Address space의 크기는 256Bytes고, Page size는 4Bytes, PDE/PTE는 1Byte를 가진다.



PDE/PTE의 구조는 6bits는 pfn 1bit는 unused, 마지막 1bit는 present bit이다. Present bit은 물리 메모리에 할당이 되어 있는지 없는지를 표시하는 지표이다. 그리고 swap이 되면 PDE/PTE구조는 swap space offset 7bits, present bit 1 bit로 이루어져 있다.

이런 주어진 정보를 가지고 MMU를 구현하면 된다.

2. 프로그램 구조와 설계

우선 ku_cpu.c 즉 cpu에서는 pid와 va 정보를 가지고 있는 txt 파일을 읽는다. 그리고 physical memory의 크기와 swap space의 크기를 인자로 받는다. 받아주면 ku_mmu_init 함수에서 physical memory와 swap 공간을 할당을 해준다. 이때 page의 input 정보를 가져올 수 있는 free list도 선언을 해준다. Page는 4byte 단위이기 때문에 mem_size/4 로 크기를 설정을 해 줄 수 있다. 그렇게 ku_mmu_init에서 pmem과 swap 공간, free list를 초기화 해준다.

그리고 input 파일에 들어온 pid와 va를 가져온다. 그 정보를 가져와서 현재 실행중인 process와 새로 들어온 process를 context switch를 해준다. 여기서 context switch를 해주는 함수는 ku_run_proc이다. Ku_run_proc의 인자값으로는 다음에 실행할 pid와 &ku_cr3를 가진다. 우선 실행할 pid가 pcb에 있는지 확인을 해준다. Pcb에 중복되는 pid가 있다면 page directory가 현재 할당 되어있다는 뜻이기 때문에 아무것도 할당을 안해주고, page directory의 시작 주소를 이중포인터를 이용해 ku_cr3에 저장을 해준다. 그런데 만약 pcb에 다음에 실행시키려하는 pid가 존재하지 않으면 그 process를 pcb에 추가를 해준다. 그리고 page directory를 물리 메모리 상에 할당을 해준다. 그리고 page가 할당되었기 때문에 pmem_freelist를 꽉 차있다는 표시인 1로 셋팅을 해준다.

그런 후 ku_traverse()를 이용해 va를 pa로 바꿔준다. 그런데 이때 page fault가 발생하면서 ku_page_fault 함수를 실행시켜준다. 그렇게 실행된 함수는 pid를 가지고 process의

pdbr을 가져온다. Pdbr은 page directory의 시작 주소이다. 여기서 va를 bit operation을 통해 pd의 index, pmd의 인덱스 pt(page table)의 index를 가져온다. 가져온 index에 page directory의 시작수로를 더해주면 pd에 할당된 pde위치를 가져온다. 이제 여기서 & 연산자를 이용해 present bit이 0인지 1인지 확인해주고, 만약 0이라면 pmem_freelist를 순회하면서 빈공간을 찾아 page middle directory를 pmem에 할당해준다. 그리고 page middle directory의 pfn을 pde의 상위 6bits에 넣어주고 , 이제 이 entry는 present 하니까 1로 바꿔준다. Pmde도 똑같이 설정을 해주고 page가 할당 되었을 때 replacement를 위해 queue에 page의 page frame number를 넣어준다. 그렇게 메모리 공간이 다 차면, deleteq를 이용해 dequeue를 해주고 빠져나온 page frame number를 이용해 page를 eviction 시켜주고 swap 공간에 넣어준다. 그리고 그 페이지를 가리키고 있는 pte에는 swap space number를 7비트에 할당해주고, present bit은 0으로 만들어 준다. 그 후, 넣으려는 page를 물리 메모리 공간의 빈 공간에 넣어준다. 만약 스왑공간이 꽉 찰 경우에는 swap space is full이 라는 문구를 띄워주고, return -1을 해준다. 그 외에도 만약 pd와 pmd pt가 스왑을 하려 할 때는 무시하기로 했기 때문에 return -1을 해준다.

결론적으로 va를 이용해 pa를 구할 수 있도록 시뮬레이션 해 주었다.

3. 결과물

- **실행 결과물**

Input file

```
1 100
2 100
3 100
1 104
2 30
1 108
```

실행결과

```
jewin@jewin-VirtualBox:~/newfolder$ ./ku_cpu newtxt.txt 64 64
[1] VA: 100 -> Page Fault
[1] VA: 100 -> PA: 16
[2] VA: 100 -> Page Fault
[2] VA: 100 -> PA: 32
[3] VA: 100 -> Page Fault
[3] VA: 100 -> PA: 48
[1] VA: 104 -> Page Fault
[1] VA: 104 -> PA: 52
ku_cpu: Fault handler is failed
```

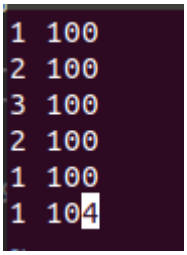
Input file

```
1 100
2 100
3 100
4 100
5 100
6 100
```

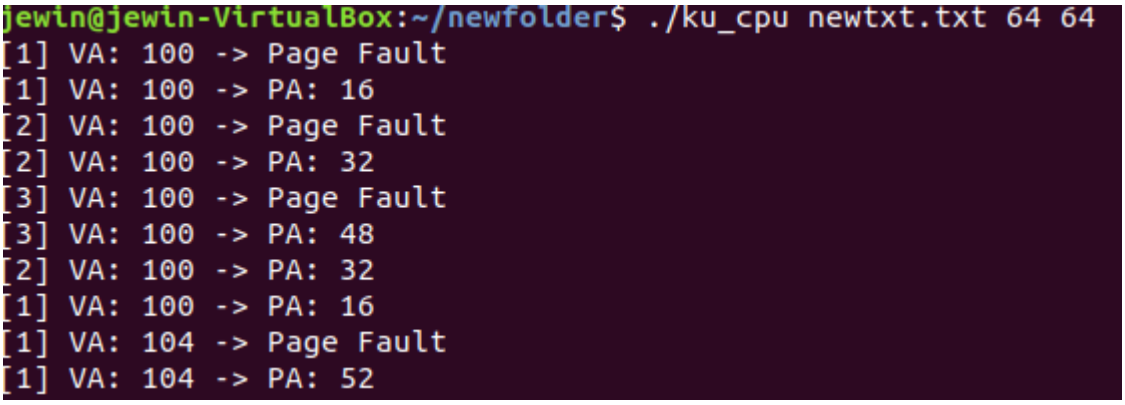
실행결과

```
[1] VA: 100 -> Page Fault
[1] VA: 100 -> PA: 16
[2] VA: 100 -> Page Fault
[2] VA: 100 -> PA: 32
[3] VA: 100 -> Page Fault
[3] VA: 100 -> PA: 48
ku_cpu: Fault handler is failed
```

Input file



실행 결과



4. Description for important functions

addq	Functionality	Page의 정보를 queue에 넣어준다.
	Parameters	Int value
	Return Value	X

Deleteq	Functionality	가장 처음 들어온 page의 정보를 출력해준다.
	Parameters	X
	Return Value	가장 처음 들어온 page의 정보

Node_insert	Functionality	링크드리스트에 pcb를 넣어준다.
	Parameters	(char pid, long long pdbr) Pid : 프로세스의 pid pdbr: page directory의 시작 주소
	Return Value	X

Check_pid	Functionality	pcb내부의 중복된 pid가 있는지 확인
	Parameters	(char pid)
	Return Value	중복된 pid의 pcbr

Pt_index	Functionality	Pt를 위한 비트 오퍼레이션
	Parameters	(char va) Va : 가상주소
	Return Value	비트 오퍼레이션 결과

Pmd_index	Functionality	Pmd를 위한 비트 오퍼레이션
	Parameters	(char va) Va : 가상주소
	Return Value	비트 오퍼레이션 결과

Pd_index	Functionality	Pd를 위한 비트 오퍼레이션
	Parameters	(char va) Va : 가상주소
	Return Value	비트 오퍼레이션 결과

Cal_pfn	Functionality	pfn계산
	Parameters	(int index) Index = pfn
	Return Value	Present bit을 씌워준 ku_pte

Check_freelist	Functionality	Free list가 가득 차있는지 확인
	Parameters	X
	Return Value	가득 차 있으면 0 가득 차 있지 않으면 1

Check_swap_freelist()	Functionality	Swap freelist가 가득 차 있는지 확인
	Parameters	X
	Return Value	가득 차 있으면 0 가득 차 있지 않으면 1

Ku_mmu_init	Functionality	Physical memory, swap space , free list 초기화
	Parameters	(mem_size, swap_size) Mem_size : 메모리 크기 Swap_size : 스왑 공간 크기
	Return Value	Pmem의 주소

Ku_run_proc	Functionality	Context switch
	Parameters	(pid, **ku_cr3) Pid : 다음 프로세스 아이디 Ku_cr3 : page directory를 가리키는 주소
	Return Value	성공하면 0 실패하면 -1

Ku_page_fault	Functionality	Page fault hanlder
	Parameters	(pid, va) Pid : 프로세스 아이디 va : virtual address
	Return Value	성공하면 0 실패하면 -1

Ku_run_proc, ku_mmu_init, ku_page_fault에 대한 설명은 프로그램 구조와 설계에 더 자세히 적혀 있습니다.

감사합니다.