

## Memory Management

- goal of MM
- fs vs MM
- paging
- process image
- 3 problems of paging

### ⟨goal of MM⟩

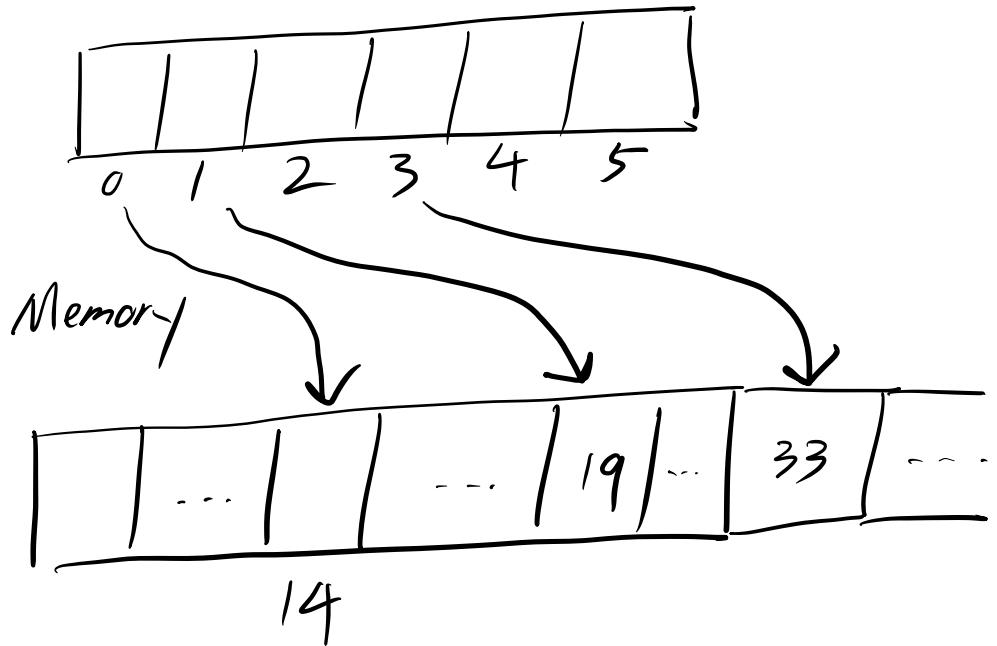
- store processes in memory efficiently
- solution (paging):  
$$\text{process} = \sum_{\text{page}} \text{page}$$

↑ page=page frame  
= 4KB

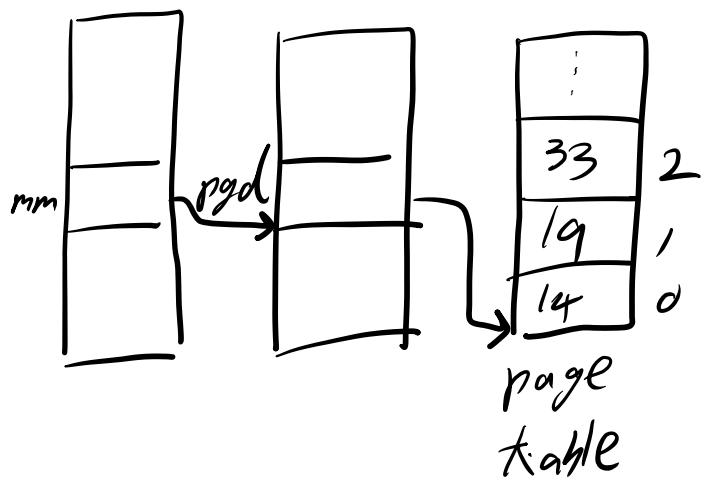
  - $\text{Memory} = \sum \text{page frame}$ .
  - store pages in page frames
  - remember page location in  
task\_struct → mm → pg

↑ process 를 가지고 있는 것

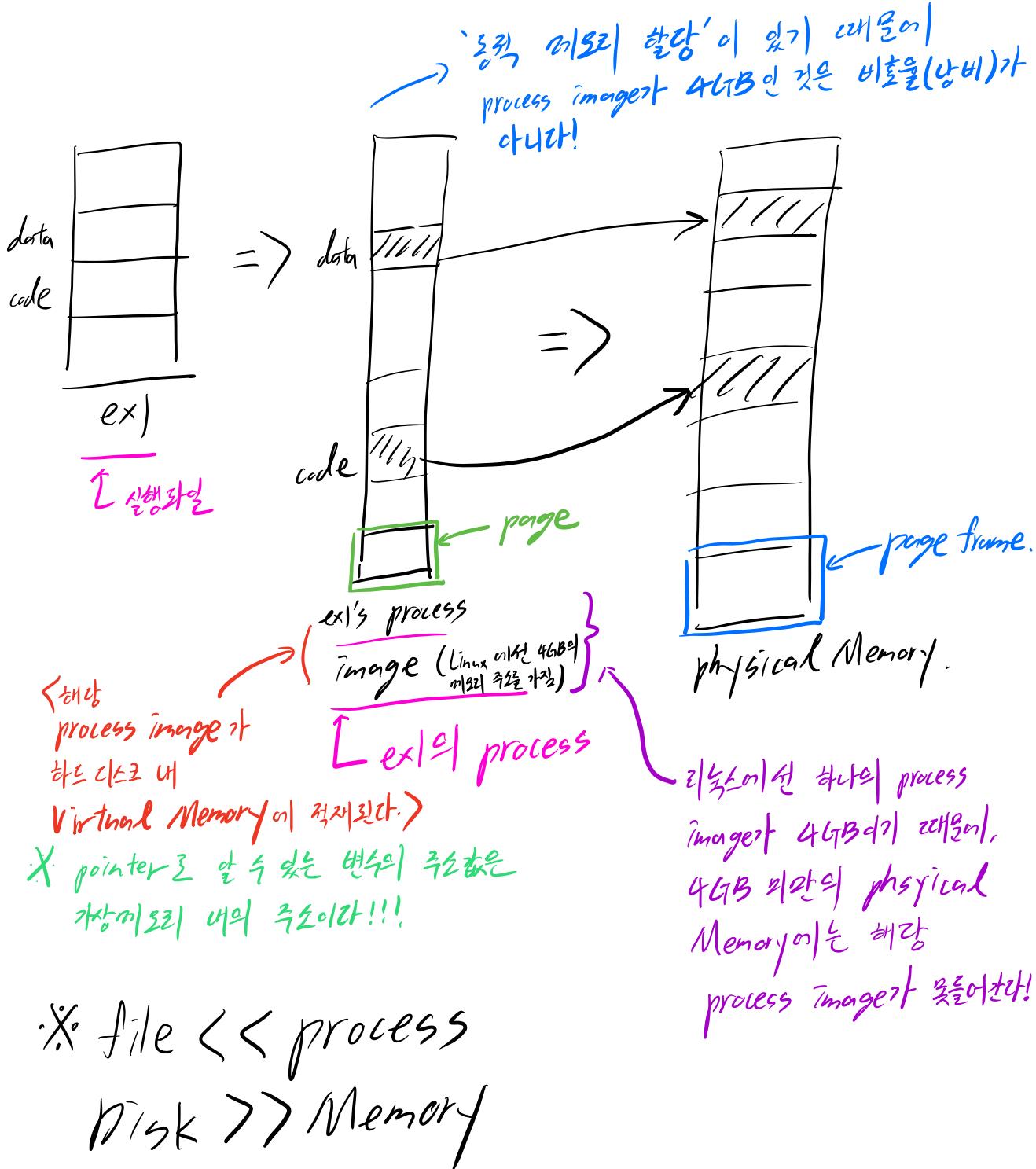
ex)  $P_1$



<  $P_1$ 's task\_struct >



ex) `gcc -O ex1 ex1.c`



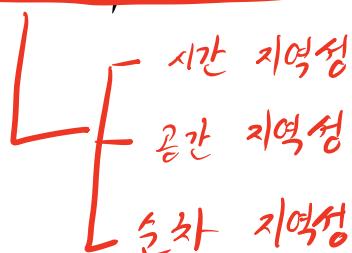
## < 3 problems of paging >

- {
  - process too big
  - page table too big
  - address computing time getting slow.

## < Solution for process too big >

- demand paging
  - < : 현재 필요한 페이지가 physical Memory에 부재한 경우'를 'page fault'라 한다.
  - 'page fault'가 발생하면, 해당 페이지를 가장 미리에서 찾아야 한다.
  - 운영체제가 page fault를 해결하는 과정을 'demand Paging'이라 한다. >

- Virtual Memory
- page fault (INT 14 → do\_page\_fault())
- locality of reference (참조의 지역성)



# < 중간 | 장로 >

- 실행파일 (.exe)는 '프로그램 파일'이다.
- 프로그램은 명령어(코드)와 정적 데이터의 묶음이다.  
(동적 데이터가 직접적으로 들어있지 않다. '명령어' 형태로 존재하고 있을 것이다.)
- 실행파일 (프로그램)은 크기가 작다. (몇 KB 밖에 안된다.  
동적 데이터를 가지고 있지 않기 때문인 것 같다.)
- 어떤 실행파일이 실행되면, 'process image (4GB)'가  
디스크 내 가상메모리 공간에 생성된다.  
[ 디스크 내 가상메모리 공간에  
process image가 저장된다.]
- process image는 code section, heap section, stack section,  
pdata section으로 나누어져 있다.
- 실행파일에 있었던 윈도(명령어, 정적 변수, 전역변수)들이 Process Image의  
직접한 세션에 저장된다. (정적 변수는 해당 변수를 생성하는 명령어가 실행된 후에,  
그리하여 Process Image의 heap 영역에 저장된다.)
- Process Image 내에 저장되어 있는 윈도를 중, 실행할 때 있어야 현재  
필요한 윈도들만 physical Memory로 이동된다.  
[ demand loading Page. ]





## $\langle$ goal of fs $\rangle$

- store files in disk efficiently
- solution :
  - file =  $\sum$  Block
  - disk =  $\sum$  Block
  - store file blocks in disk blocks
  - remember block location in inode  
 $\rightarrow i\text{-block[ ]}$

file not  
가지고 있는 것

- 3 problems of paging
- 1. process too big
- 2. page table too big
- 3. address mapping getting to slow.

⟨Solution for "process too big"⟩

- process =  $\sum$  page
- store only active pages in memory frame.
- remember frame location  
~~\* in map table~~

" " " " "

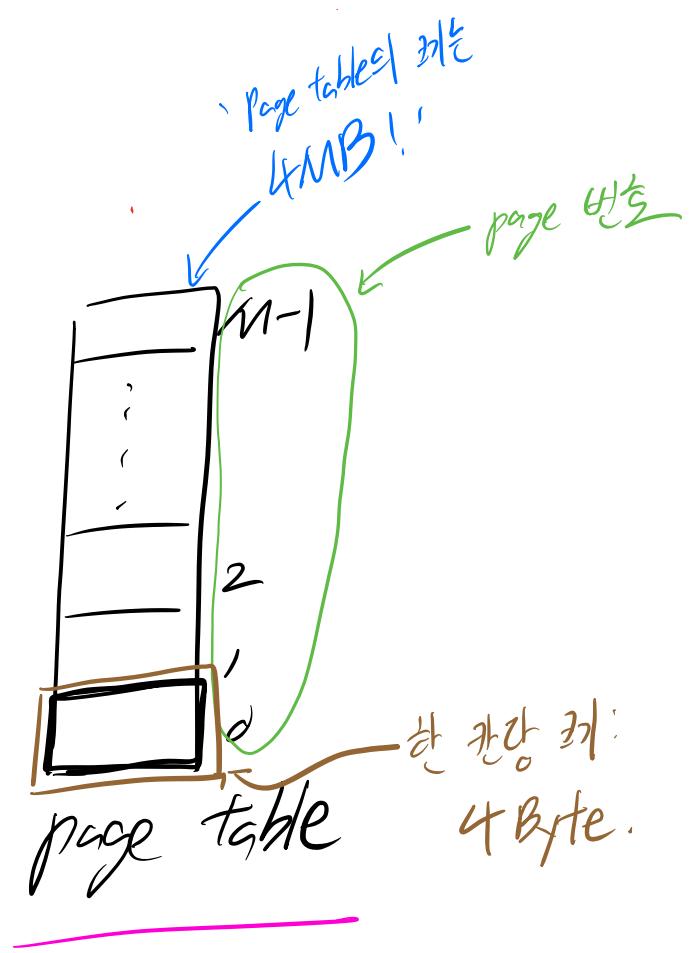
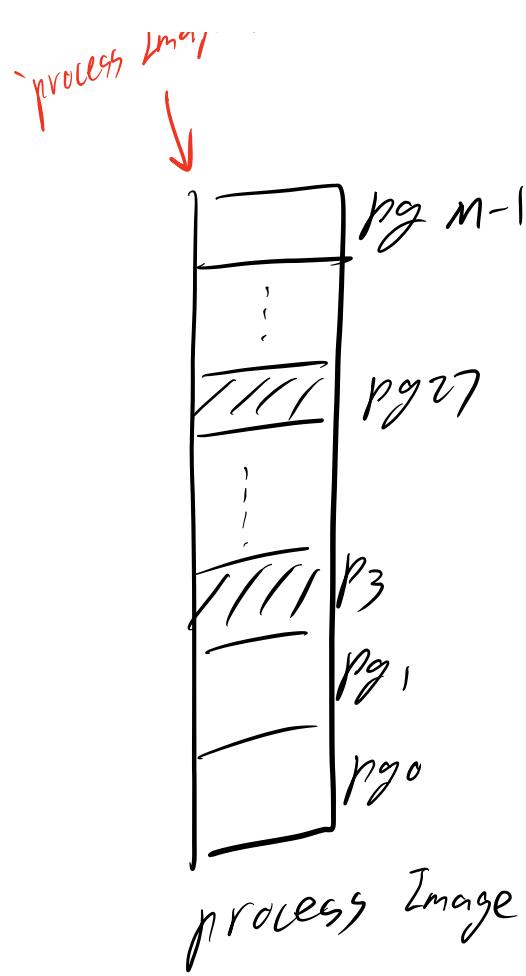
## < Solution for page table too big >

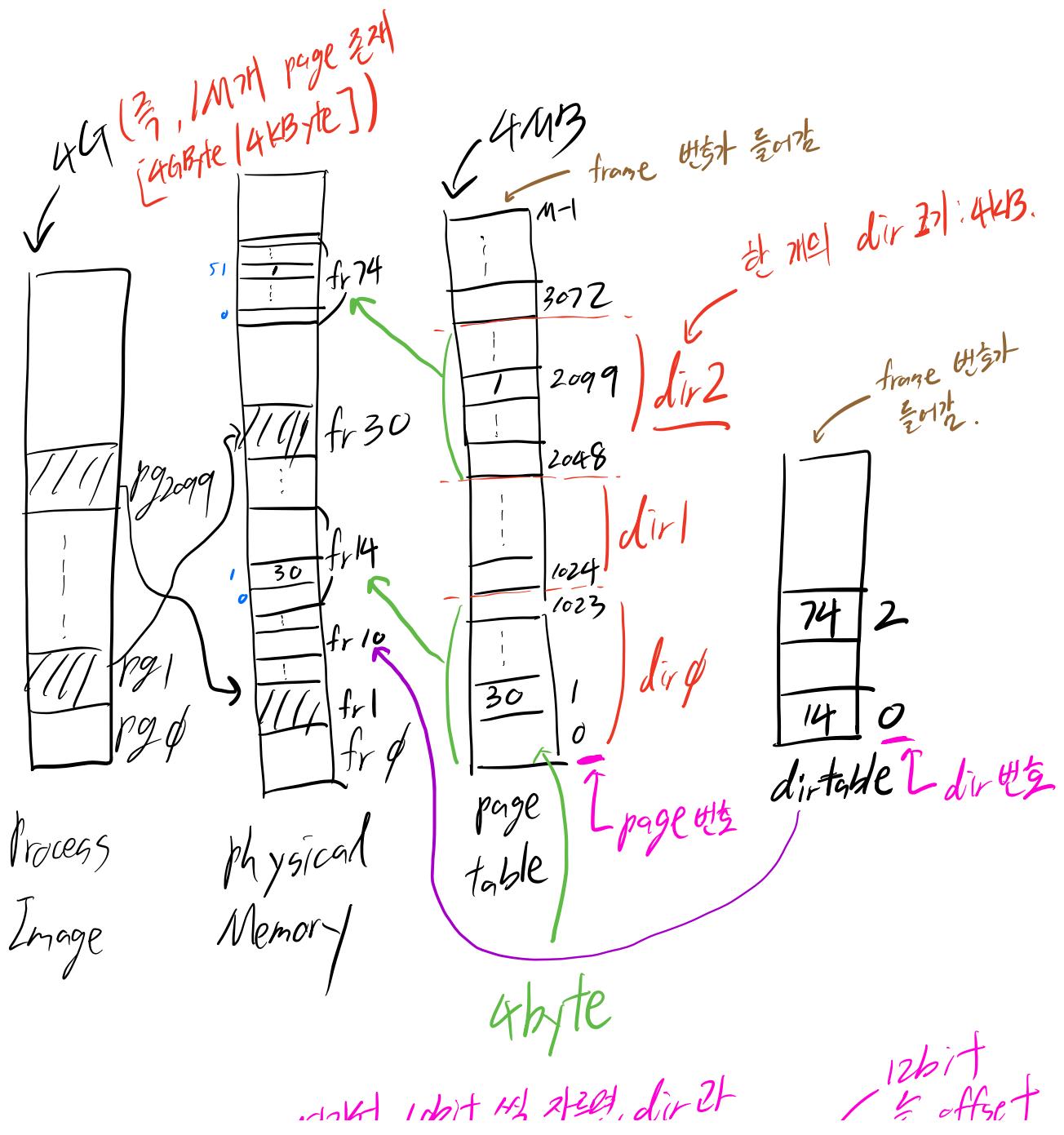
- ~~page table =  $\sum \underline{\text{dir}}$~~  active 디렉토리를  
가지다가 physical  
store only active dirs. in Memory on  
mem frame
- remember fr location in dir  
table

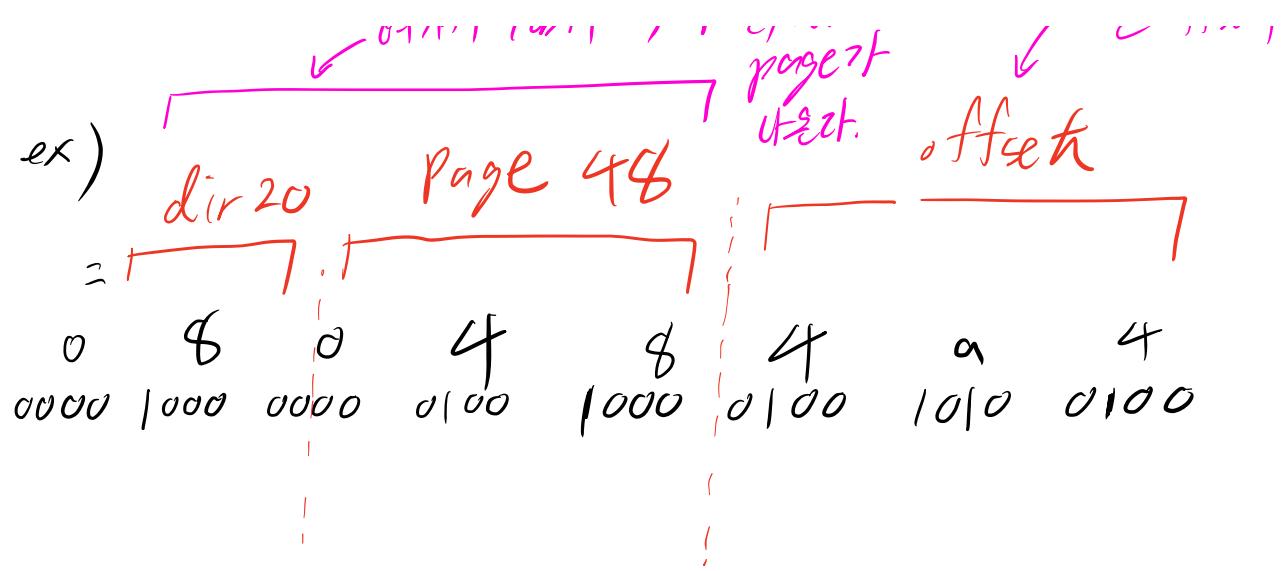
[ ]

[ 해당 dir table'가 physical memory on  
넣어야 함.]

→ next 3개는 4GB!  $\Rightarrow$  4GB를 32비트 차지하는 32(4KBByte)로  
4GB, Process Image를 대체하는  
32비트 차지하는 1MB (1백화)







offset: distance