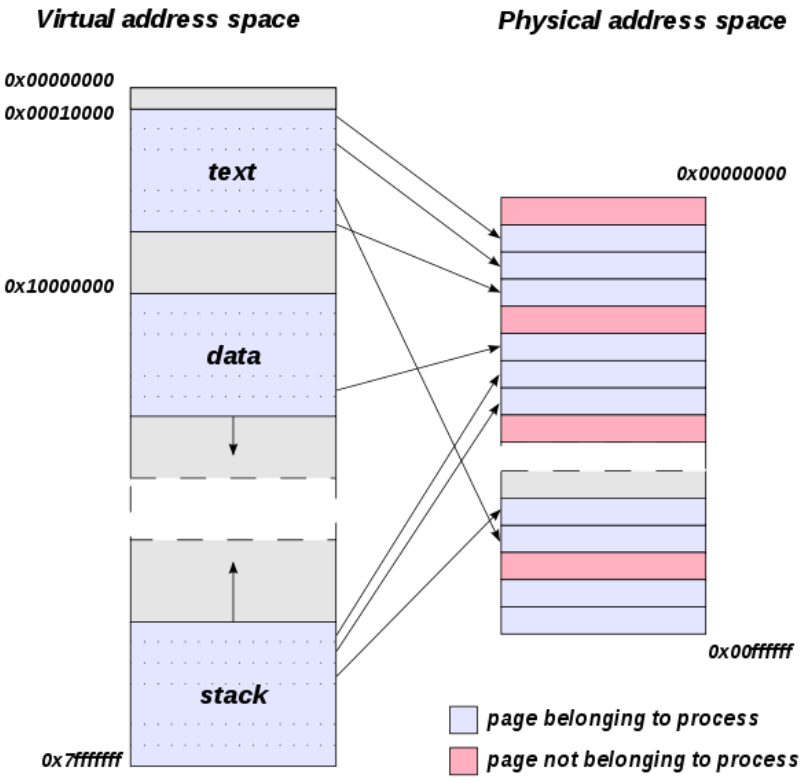
2019년 1학기 운영체제 과제 #3 (50점)

과제 개요

운영 체제는 실제 메모리의 용량보다 큰 가상 주소 공간을 제공하여, 실행되는 프로세스들이 물리적으로 존재하는 것보다 많은 메모리를 참조할 수 있게 한다. 이러한 기법을 가상 메모리(Virtual Memory)라고 한다. 이번 과제에서는 이러한 가상 메모리에서 주로 사용되는 메모리 관리 기법인 페이징(Paging)과 버디 시스템(Buddy System)에 대해 알아보고, 단순한 형태의 시뮬레이터를 제작할 것이다.



1 과제 목적

✓ 가상 메모리의 구조와 동작에 대해 이해한다.

✓ 메모리 관리 기법인 페이징과 버디 시스템을 이해한다.

2 제출 기한

✓ 2019년 6월 17일(월) 오후 6:00 까지

3 제출 방법

✓ 과제는 **보고서 과제**와 **실습 과제**로 이루어져 있다.

✓ 보고서 출력본: 제 4공학관 8층 D814호 앞 과제 제출함

✓ 보고서 파일 및 실습 과제 소스 파일: 과목 홈페이지 과제 제출 게시판

✓ 보고서 파일은 과제 게시판에도 제출하고 출력본을 연구실에도 제출해야 함

4 제한 사항

✓ 운영체제는 리눅스를 사용한다. 리눅스 종류와 버전은 제한이 없다.

- 채점 서버 OS: Ubuntu 16.04
- 채점 서버 컴파일러: gcc 6.4.0 / clang 3.8.0
- UTF-8 (한글을 사용할 경우 euc-kr을 사용하지 말고 locale을 ko.utf-8로 사용)
- 위에서 언급한 환경이 아닌 다른 환경을 사용하여 동작하지 않는 경우 큰 감점

✓ 프로그래밍 언어는 C 또는 C++을 사용한다.

- 과제의 입력과 출력은 파일 입출력이 아닌 표준 입출력으로 한다.
- C 표준 라이브러리 또는 C++ 표준 라이브러리는 자유롭게 사용 가능하나 그 외의 외부 라이브러리의 사용은 금지한다.

✓ 실습 과제 프로그램 구조

- 제출물의 압축을 해제하면 생성되는 폴더 내에 Makefile과 구현한 프로그램의 소스 코드가 존재해야 한다.
- make 명령을 실행하면 소스 코드가 컴파일 되어 project3이라는 이름의 실행 가능한 바이너리가 생성되어야 한다. 터미널에서 ./project3 명령으로 프로그램이 실행된다.
- 컴파일 시 필요한 옵션(라이브러리, C++17 등)은 Makefile에 반드시 기재하여 채점 서버에서 컴파일이 되도록 해야 한다. 채점 서버에서 컴파일이 되지 않으면 매우 크게 감점을 당할 수 있다.
- make clean 명령을 실행하면 make를 통해 생성된 빌드 결과물을 삭제해야 한다.
- 이 구조를 따르지 않는 경우 크게 감점이 된다.

✓ 1인 1프로젝트이며 각자의 환경에서 작업한다.

5 유의 사항

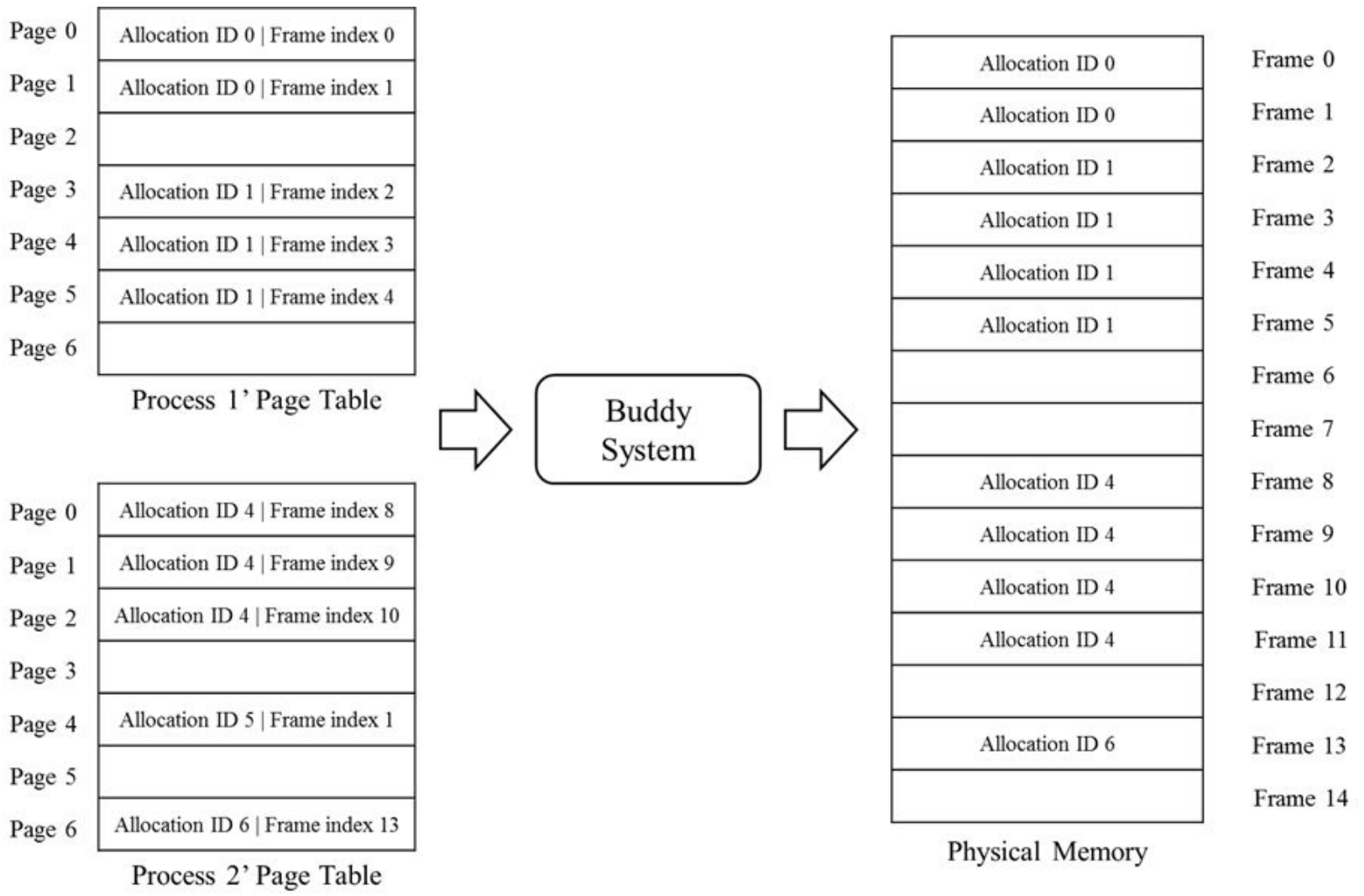
- ✓ **적절한 사유 없이 Delay 될 경우 절대 받지 않음**
- ✓ 그림, 코드 조각을 포함한 모든 참고 자료는 내용은 반드시 출처를 명시 (출처 명시 안했을 시 감점)
- ✓ 타 수강생의 보고서 및 과제를 카피 또는 수정하여 제출하였을 경우 모두 0점 처리
- ✓ 과제 제출 방법, 제출 파일 생성 방법과 프로그램 구조(특히 프로그램의 이름)를 지키지 않을 경우 감점 또는 0점 처리
- ✓ 제출한 소스에는 반드시 주석을 달 것, 주석의 내용이 불분명하거나 없을 경우 감점 처리
- ✓ 제출한 과제가 컴파일이 되지 않는 경우 감점 또는 0점 처리
- ✓ 과제에 대한 문의 사항이 있을 경우 연구실 웹 사이트의 게시판으로 문의
- ✓ **이 과제 명세에 대한 내용을 웹 사이트에 배포하거나 질문한 것이 발각되는 경우 0점 처리**

과제 세부 사항

과제는 실습 과제와 보고서 과제로 이루어져 있다. 페이징(Paging)과 버디 시스템(Buddy System)에 대한 설명은 강의 노트를 참고한다.

실습 과제

- 전체 구현 내용
  - 버디 시스템과 페이징이 결합된 가상의 시스템을 시뮬레이션 한다.
    - 입력으로 프로세스가 사용하는 메모리 연산(Memory Operation)이 들어오고 명령어 한개 단위로 연산을 처리하는 과정을 출력하는 시스템이다
  - 본 시스템의 전체 구성은 아래 그림과 같다.



  - 프로세스는 독자적인 2048byte(2K) 크기의 가상 메모리(Virtual Memory) 공간을 갖는다.
  - 본 시스템은 1024byte(1K) 크기의 물리 메모리(Physical Memory)가 존재한다.
  - 프로세스의 가상 메모리는 32byte의 페이지 단위로 나뉘어 관리가 되며, 물리 메모리 또한 32byte 프레임 단위로 관리된다.
    - 페이지 테이블(Page Table)은 각 페이지가 물리 메모리에 할당될 시 갖게 되는 Frame index, Valid bit를 갖는다.
    - 가상 메모리 할당의 경우 요청하는 크기가 들어가는 첫 번째 공간으로 할당한다.
    - 가상 메모리는 하위 주소부터 메모리를 채워나간다.
  - 프로세스는 페이지 테이블 할당(Page Table Allocation), 메모리 접근(Memory Access) 두 가지 연산을 수행할 수 있다.
    - 페이지 테이블 할당(Page Table Allocation) : 프로세스가 원하는 연속된 페이지 개수 만큼 페이지 테이블에 메모리 할당(Memory Allocation)을 수행한다

- 프로세스는 원하는 페이지의 개수 만큼을 페이지 테이블에 할당하고, 이를 Allocation ID를 통해 관리한다.
- 예를 들어 위의 그림에서 Process 2는 3개의 페이지를 Allocation ID 4로 정하고 관리한다. Allocation ID는 시스템에서 중복되지않는 유일한 값이다.
- 페이지 테이블 할당 연산의 경우, 물리 메모리에 프레임을 할당하지 않고 페이지 테이블에만 할당한다.(아래의 예제 참고)

■ 메모리 접근(Memory Access) : 프로세스가 이전에 할당한 페이지와 그것에 연결된 프레임에 접근(Access)한다. 이 때 Allocation ID를 통해 접근한다.

- 페이지에 할당된 프레임이 없는 경우에는 버디 시스템을 사용하여 할당해야 할 프레임의 수를 계산한 뒤, 물리 메모리 상에서 연속된 프레임을 할당한다. 이 때 버디 시스템은 하위 주소부터 메모리를 채워나간다.
  - 프로세스가 메모리 접근 연산을 수행하면, 요청한 페이지들은 버디 시스템에 의해 할당해야 할 페이지의 개수를 계산한 뒤 해당 페이지들을 가상 메모리 상에 해당 크기만큼 할당 가능한 연속된 공간에 순차적으로 할당된다.
- 버디 시스템에 의해 프로세스로부터 요청된 페이지들을 물리 메모리에 할당할 때, 물리 메모리에 할당 가능한 공간이 없다면 바로 아래에 서술되어있는 **페이지 교체 알고리즘**에 따라 기존에 할당되어 있는 프레임을 해제한다. 프로그램의 입력으로 어떤 페이지 교체 알고리즘을 사용할 지 번호(0, 1, 2, 3, 4, 5)로 주어지며, 해당 번호가 무엇을 의미하는지는 아래의 설명에 적혀있는 바와 같다.

■ 페이지 교체 알고리즘(Page Replacement Algorithms)

- 알고리즘 공통 조건
  - 프레임을 해제할 때에는 같은 Allocation ID를 갖는 연속된 프레임을 모두 해제한다.
  - 요청된 연속된 페이지들이 할당될 수 있는 물리 메모리 내의 가용 공간이 확보될 때까지, 페이지 교체 알고리즘에 의한 해제 과정을 반복한 후 할당한다.
  - 요청된 페이지들이 물리 메모리에 할당되면, 해당 Frame index가 프로세스의 페이지 테이블에 기록된다. 페이지 테이블은 페이지 교체 알고리즘에 의해 해제된 프레임에 대한 사항을 업데이트 한다.
  - 만약 알고리즘에 의해 선택될 수 있는 프레임이 두개 이상이라면, Allocation ID가 작은 것부터 선택해 이를 해제후 교체한다.
  - 각 알고리즘에 대한 자세한 설명은 강의노트(L9-virtualmemory pg..17~)를 참고하도록 한다.
- FIFO 알고리즘 (0번)
- LRU 알고리즘 (1번)
- Sampled LRU 알고리즘 (2번)
  - Time interval은 8번의 명령어 입력으로 한다.
  - Reference byte는 강의노트에 있는 것처럼 1 byte(8 bits)로 한다.
- LFU 알고리즘 (3번)
- MFU 알고리즘 (4번)
- Optimal 알고리즘 (5번)
  - 강의노트에 서술되어 있는 대로 "앞으로 가장 오랫동안 사용되지 않을 페이지 (the one that will not be used for the longest period of time)"를 선택해서 교체한다. 이는 모든 경우의 수를 탐색해 얻은 page fault rate가 최소가 되는 해와 다를 수 있다.

- **입력 파일 구성**
  - 과제 2의 program4와 같이 표준 입력을 통해 입력받는다.
    - Ex. ./project3 < input.txt
  - 첫번째 줄에는 어떤 **페이지 교체 알고리즘**을 사용할 것인지가 주어진다 (0, 1, 2, 3, 4, 5)
  - 두번째 줄에는 시뮬레이션에서 사용될 프로세스의 수 P가 주어진다. (P<5)
  - 세번째 줄에는 명령어 개수 N이 주어진다.
  - 네번째 줄부터 N+3번째 줄까지 입력되는 명령어는 다음과 같다.
    - [PID]tab[Function]tab[Allocation ID]tab[Demand Page 개수]
      - Function : 1 = 페이지 테이블 할당, 0 = 메모리 접근
      - PID는 0~(P-1) 까지 중 하나가 입력된다.
      - Allocation ID : Page Table 할당 시 연속된 Block임을 나타내기 위한 ID
        - Allocation ID는 0 이상의 정수이며, 중복되지 않는다.
      - Demand Page 개수 : Allocation 하고자 하는 연속된 Page 수

○ 프로세스의 수 P는 입력에 따라 달라질 수 있다. LRU 알고리즘을 사용한 2개의 프로세스에 대한 입력 예시는 다음과 같다.

```
1         // 어떤 페이지 교체 알고리즘을 사용할 것인지 (0, 1, 2, 3, 4, 5)
2         // 프로세스 수
20        // 입력되는 명령어 개수
0     1     1     16          // PID 0에서 16개의 Page를 Allocation 요청
0     1     2     12
0     1     3     22
0     1     4     14
1     1     5     11
1     1     6     9
1     1     7     20
1     1     8     10
1     1     9     14
0     0     2                 // PID 0에서 Allocation ID가 2인 메모리에 접근
1     0     5
1     0     6
1     0     5
0     0     2
0     0     3
0     0     4
1     0     6
1     0     8
0     0     4
0     0     2
```

- 할당되지 않은 Allocation ID에 대한 접근은 없다고 가정한다.
- 할당 가능한 연속된 페이지가 없는 경우 또는 가상 메모리 전체 크기보다 큰 메모리를 요청한 경우에는 무시한다.

- **출력 형식**
  - 표준 출력을 사용하여 **물리 메모리의 할당, 접근 기록**을 출력한다. 위의 입력 명령어 한개가 처리될 때마다 메모리 정보를 출력한다.
  - 예시 출력 파일을 참고하여 아래에 자세히 서술된 출력 포맷대로 출력을 한다. 양식을 지키지 않으면 채점이 불가능할 수 있음을 유의한다.
  - 출력 파일의 맨 마지막 줄에는 시뮬레이션에서 발생한 **페이지 폴트의 개수**를 출력한다.
  - 다음은 위 예제 입력에 대한 시뮬레이션 결과이다.
    - AID는 Allocation ID를 의미한다. Valid bit은 해당 페이지가 물리 메모리에 존재하는지를 나타낸다.
    - 페이지 테이블과 물리 메모리를 출력할때 왼쪽을 하위 주소로 표현한다.

```
* Input : Pid [0] Function [ALLOCATION] Alloc ID [1] Page Num[16]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [0] Function [ALLOCATION] Alloc ID [2] Page Num[12]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [0] Function [ALLOCATION] Alloc ID [3] Page Num[22]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|33--|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|00--|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [0] Function [ALLOCATION] Alloc ID [4] Page Num[14]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [1] Function [ALLOCATION] Alloc ID [5] Page Num[11]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
```

```
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|555-|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|000-|----|----|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [1] Function [ALLOCATION] Alloc ID [6] Page Num[9]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|

* Input : Pid [1] Function [ALLOCATION] Alloc ID [7] Page Num[20]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|

* Input : Pid [1] Function [ALLOCATION] Alloc ID [8] Page Num[10]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|88--|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|00--|----|----|----|

* Input : Pid [1] Function [ALLOCATION] Alloc ID [9] Page Num[14]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [0] Function [ACCESS] Alloc ID [2] Page Num[12]
>> Physical Memory :          |2222|2222|2222|2222|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [1] Function [ACCESS] Alloc ID [5] Page Num[11]
>> Physical Memory :          |2222|2222|2222|2222|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1110|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [1] Function [ACCESS] Alloc ID [6] Page Num[9]
>> Physical Memory :          |6666|6666|6666|6666|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [1] Function [ACCESS] Alloc ID [5] Page Num[11]
>> Physical Memory :          |6666|6666|6666|6666|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1110|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [0] Function [ACCESS] Alloc ID [2] Page Num[12]
>> Physical Memory :          |2222|2222|2222|2222|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1110|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [0] Function [ACCESS] Alloc ID [3] Page Num[22]
>> Physical Memory :          |3333|3333|3333|3333|3333|3333|3333|3333|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|1111|1111|1111|1111|1111|1100|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|

* Input : Pid [0] Function [ACCESS] Alloc ID [4] Page Num[14]
>> Physical Memory :          |4444|4444|4444|4444|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0011|1111|1111|1111|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
```

```
* Input : Pid [1] Function [ACCESS] Alloc ID [6] Page Num[9]
>> Physical Memory :          ¦4444¦4444¦4444¦4444¦6666¦6666¦6666¦6666¦
>> pid(0) Page Table(AID) :   ¦1111¦1111¦1111¦1111¦2222¦2222¦2222¦3333¦3333¦3333¦3333¦3333¦3344¦4444¦4444¦4444¦
>> pid(0) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0011¦1111¦1111¦1111¦
>> pid(1) Page Table(AID) :   ¦5555¦5555¦5556¦6666¦6666¦7777¦7777¦7777¦7777¦7777¦8888¦8888¦8899¦9999¦9999¦9999¦
>> pid(1) Page Table(Valid) : ¦0000¦0000¦0001¦1111¦1111¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦

* Input : Pid [1] Function [ACCESS] Alloc ID [8] Page Num[10]
>> Physical Memory :          ¦8888¦8888¦8888¦8888¦6666¦6666¦6666¦6666¦
>> pid(0) Page Table(AID) :   ¦1111¦1111¦1111¦1111¦2222¦2222¦2222¦3333¦3333¦3333¦3333¦3333¦3344¦4444¦4444¦4444¦
>> pid(0) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦
>> pid(1) Page Table(AID) :   ¦5555¦5555¦5556¦6666¦6666¦7777¦7777¦7777¦7777¦7777¦8888¦8888¦8899¦9999¦9999¦9999¦
>> pid(1) Page Table(Valid) : ¦0000¦0000¦0001¦1111¦1111¦0000¦0000¦0000¦0000¦0000¦1111¦1111¦1100¦0000¦0000¦0000¦

* Input : Pid [0] Function [ACCESS] Alloc ID [4] Page Num[14]
>> Physical Memory :          ¦8888¦8888¦8888¦8888¦4444¦4444¦4444¦4444¦
>> pid(0) Page Table(AID) :   ¦1111¦1111¦1111¦1111¦2222¦2222¦2222¦3333¦3333¦3333¦3333¦3333¦3344¦4444¦4444¦4444¦
>> pid(0) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0011¦1111¦1111¦1111¦
>> pid(1) Page Table(AID) :   ¦5555¦5555¦5556¦6666¦6666¦7777¦7777¦7777¦7777¦7777¦8888¦8888¦8899¦9999¦9999¦9999¦
>> pid(1) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦1111¦1111¦1100¦0000¦0000¦0000¦

* Input : Pid [0] Function [ACCESS] Alloc ID [2] Page Num[12]
>> Physical Memory :          ¦2222¦2222¦2222¦2222¦4444¦4444¦4444¦4444¦
>> pid(0) Page Table(AID) :   ¦1111¦1111¦1111¦1111¦2222¦2222¦2222¦3333¦3333¦3333¦3333¦3333¦3344¦4444¦4444¦4444¦
>> pid(0) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦1111¦1111¦1111¦0000¦0000¦0000¦0000¦0000¦0011¦1111¦1111¦1111¦
>> pid(1) Page Table(AID) :   ¦5555¦5555¦5556¦6666¦6666¦7777¦7777¦7777¦7777¦7777¦8888¦8888¦8899¦9999¦9999¦9999¦
>> pid(1) Page Table(Valid) : ¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦0000¦

page fault = 10
```

- 입력의 각 줄을 실행 후 결과를 위와 같이 출력한다. **채점을 위해 출력 형태는 아래의 printf문을 반드시 사용하여 형태를 맞춘다. 아래 printf 문을 사용하지 않아 출력 형태가 맞지 않을 경우 감점.**
- 첫번째 줄 : 입력 파일의 각 줄에 대한 정보
  - 출력 포맷 :

    ```
    printf("* Input : Pid [%d] Function [%s] Alloc ID [%d] Page Num[%d]\n", 각 Parameter..);
    ```
  - 각 Parameter는 입력 파일의 각 줄을 읽어서 실행결과 화면과 같이 출력한다.
- 두번째 줄 : 현재 물리 메모리 현황
  - 출력 포맷 :

    ```
    printf("%-30s", ">> Physical Memory : "); printf(현재 Physical Memory 현황 출력);
    ```
  - 현재 물리 메모리 현황 : 각 현황을 32 byte 단위로 출력한다. 각 숫자는 해당 Frame에 할당된 Page의 Allocation ID를 나타낸다. 만약 해당 Frame에 할당된 Page가 없다면 “-“를 출력, 128bytes당 하나의 “|”를 출력하여 눈으로 인지할 수 있도록 한다.
- 그 다음 줄부터는 프로세스별로 두 줄씩 출력한다. 프로세스의 번호가 작은 것부터 출력한다.
- 프로세스의 첫번째 줄 : 현재 페이지 테이블의 현황(Allocation ID 별)
  - 출력 포맷 :

    ```
    printf(">> pid(%d) %-20s",pid, "Page Table(AID) : "); printf(현재 Page Table 현황 출력);
    ```
  - Allocation ID에 따른 페이지 테이블 현황 : 물리 메모리의 현황을 출력 형태와 같이 32 byte 단위로 출력한다. 각 숫자는 해당 Page가 할당되어 있다면 할당된 Allocation ID를 뜻한다.
- 프로세스의 두번째 줄 : 현재 페이지 테이블의 현황(물리 메모리 할당 여부 별)
  - 출력 포맷 :

    ```
    printf(">> pid(%d) %-20s",pid, "Page Table(Valid) : "); printf(현재 Page Table 현황 출력);
    ```
  - 물리 메모리 할당 여부에 따른 페이지 테이블 현황 : 세번째 줄 출력 형태와 같이 출력하되, 각 숫자는 할당된 각 페이지가 실제 물리 메모리에 할당되어 있는지에 대한 여부를 나타낸다. 해당 페이지가 물리 메모리에 할당되어 있다면 1, 아니면 0을 출력한다.
- 위의 출력 이후 한줄 개행 문자 `printf("\n");`을 추가로 출력하여 다음 입력 줄 처리 결과와 구분될 수 있도록 한다.
- 프로그램의 맨 마지막줄: 시뮬레이션 과정에서 발생한 총 페이지 폴트의 수
  - 출력 포맷 :

    ```
    printf("page fault = %d\n", page_fault);
    ```

보고서 과제

보고서는 자유 형식으로 작성하되, 아래의 내용을 반드시 포함하도록 한다.

- ✓ 작성한 프로그램의 동작 과정과 구현 방법
  - 소스 코드 줄별 설명보다는 전체적인 동작 방식에 대한 설명
  - 각 알고리즘에 대하여 순서도나 블록 다이어그램 등 도식화 자료를 반드시 이용할 것
- ✓ 개발 환경 명시
  - uname -a 실행 결과
  - 사용한 컴파일러 버전, CPU, 메모리 정보 등
- ✓ 결과 화면 스크린샷과 그것에 대한 토의 내용
  - 자신이 만든 3가지 이상의 다른 input 파일과 그에 따른 5가지 알고리즘의 page fault rate 비교
  - input 파일 스크린샷, 그래프 또는 도표등을 반드시 활용하여 결과 분석을 성의있게 할 것
- ✓ 과제 수행 시 겪었던 어려움과 해결 방법

보고서 작성 관련 유의사항은 다음과 같다.

- ✓ 자신의 소스 코드 전체를 첨부하지 말 것 (소스 코드 의 특정 부분을 설명하기 위한 일부 첨부는 허용)
- ✓ 그림을 비롯한 참조 내용은 반드시 출처를 명시할 것
- ✓ 항목을 분명하게 나누어 명료하게 서술하는 것이 좋다

과제 제출 방법

보고서 pdf 파일(hw3_[학번].pdf)과 소스 압축 파일(hw3_[학번].tar)을 제출한다.

보고서 파일(hw3_[학번].pdf)

PDF 파일로 변환하여 제출한다. 파일의 이름은 반드시 hw3_[학번].pdf로 한다.

소스 압축 파일(hw3_[학번].tar)

반드시 다음 명령어를 사용해서 압축 파일을 생성한다. **다른 형식의 압축 또는 이중 압축은 절대 허용하지 않으며 보고서 파일을 함께 압축해서는 안된다.**

```
# 본인이 작업하는 디렉토리가 hw3라고 가정하자.
# 작업하는 디렉토리가 hw3이라는 것은 hw3 폴더 바로 안에 Makefile이 있음을 의미한다.
# 본인의 학번이 2019123123 이라고 가정하면, 아래의 일련의 명령어를 수행한 이 후에 hw3_2019123123.tar을 제출한다.
# 아래의 일련의 명령어를 수행한 이 후에 hw3_2019123123.tar을 제출한다.
# 이는 압축을 해제하였을 때 본인의 학번으로 된 폴더 하나가 나타나야 하고, 그 폴더 바로 아래에 Makefile이 있어야 함을 의미한다.
$ cd ~/hw3/..
$ mv hw3 2019123123
$ tar cvf hw3_2019123123.tar 2019123123
```

참고 사이트

https://en.wikipedia.org/wiki/Virtual_memory

https://en.wikipedia.org/wiki/Paging

https://en.wikipedia.org/wiki/Buddy_memory_allocation

http://egloos.zum.com/sweeper/v/2988689