Report

2016112083 김연웅

개발환경 명시

uname -a 실행결과

- **uname - a 실행결과**

```
eric@ubuntu:~$ uname -a
Linux ubuntu 5.10.19-2016112083 #1 SMP Thu Mar 11 09:06:07 PST 2021 x86_64 x86_6
4 x86_64 GNU/Linux
eric@ubuntu:~$
```

- **사용한 컴파일러 버젼**

```
eric@ubuntu:~$ g++ --version
g++ (Ubuntu 7.5.0-6ubuntu2) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

eric@ubuntu:~$
```

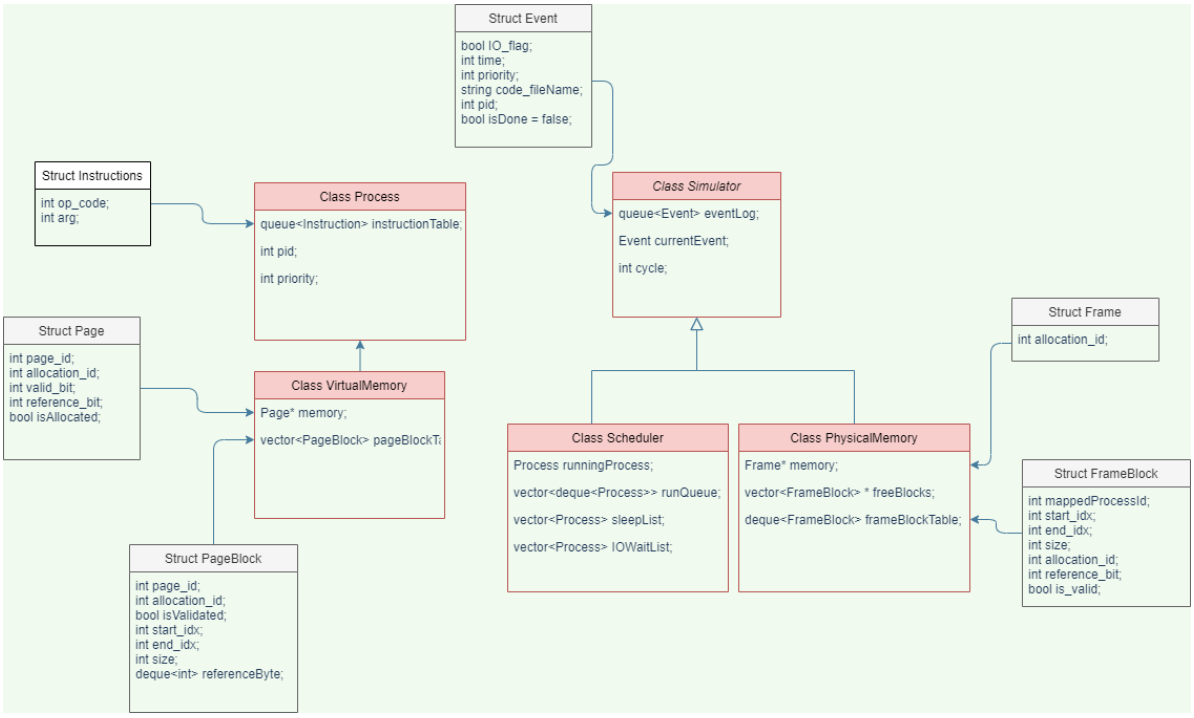
CPU 6코어, 메모리 4기가바이트

```
eric@ubuntu:~$ cat /proc/meminfo | grep 'MemTotal'
MemTotal:        3995220 kB
```

```
eric@ubuntu:~$ grep -c processor /proc/cpuinfo
6
```

작성한 프로그램의 동작 과정과 구현 방법

사용한 클래스와 구조체 설명



프로젝트에서 사용한 클래스와 구조체의 대략적인 개념도이다. (위 개념도에는 일부 중요한 멤버들만 표기했다.)

Class Process

Instruction은 과제의 설명대로 opcode와 arg로 나누어 하나의 구조체로 표현했다. Process 클래스는 해당 Process객체의 code를 전부 읽고 Instruction 구조체 형식으로 모든 line을 instructionTable에 저장한다. 또한 pid와 priority라는 고유하고 불변하는 값을 지닌다. 각 프로세스는 VirtualMemory 객체를 한개씩 지닌다.

Class VirtualMemory

가상 메모리를 나타내는 클래스이다. 각 프로세스마다 한개 객체씩 생성되어 프로세스의 멤버로 존재한다. 먼저 가상 메모리 공간은 Page 구조체를 정의하고 이들로 이루어진 배열로 표현냈다. Page 구조체는 page_id, allocation_id, valid_bit, reference_bit 등과 같은 pageTable에 존재하는 값들을 지닌다. PageBlock은 memory allocation시 할당된 크기만큼의 블록을 나타내는 구조체이다.해당 페이지 블록의 정보를 지니고 있으며 가상 메모리 공간 내에서 시작주소와 끝주소 그리고 크기를 저장한다. sampled page replace 알고리즘의 경우, reference byte정보도 지닌다. 각 가상메모리에서 할당된 PageBlock은 vector pageBlockTable에 저장되고 관리된다.

Class PhysicalMemory

물리 메모리를 나타내는 클래스이다. 시스템에 한개 존재하므로 Simulator의 멤버로 하나의 객체만 존재한다. 가상 메모리와 마찬가지로 메모리 공간을 Frame 구조체를 선언, Frame로 이루어진 배열로 표현했다. 물리 메모리는 가상메모리의 page table과 같은 정보가 따로 없으므로 단순히 allocation id만 있으면 된다. 또한 마찬가지로, FrameBlock이라는 구조체로 할당된 물리메모리 공간의 덩어리를 표현했다. 시작주소, 끝주소, 크기 그리고 할당된 FrameBlock과 맵핑되는 프로세스의 아이디를 저장하였다. 할당된 frameBlock들은 vector frameBlockTable에 저장되고 관리된다.

Class Scheduler

스케줄러 역할을 수행하는 클래스이다. 시스템에 한개의 객체만 simulator의 멤버로 존재한다. 9개의 runningQueue가 있으며 sleepList,IOwaitList로 프로세스들의 스케줄링을 관리한다. 또한 현재 작업중인 프로세스를 runningProcess로 지니고 있다.

Class Simulator

과제의 시스템을 표현하는 클래스이다. Input file의 이벤트들을 Event 구조체로 정의했고 input file의 모든 이벤트들을 구조체로 변환해 eventLog에 저장한다. cycle 을 저장하며 simulator의 실질적인 실행, 관리 종료를 담당한다.

전체적인 실행 흐름

1. **main.cpp**

프로그램의 메인 함수는 main.cpp에 존재한다. main함수에서는 주어진 argument 값들을 읽어 -dir option과 -page option을 저장한다. 이후 인풋 파일의 모든 라인을 읽고 메모리 설정 값들을 저장하고, 모든 이벤트들을 Event구조체로 저장한다. 앞서 모든 저장한 값들을 토대로 Simulator 객체를 생성하고 simulator을 실행시킨다.

2. **simulator.simulate()**

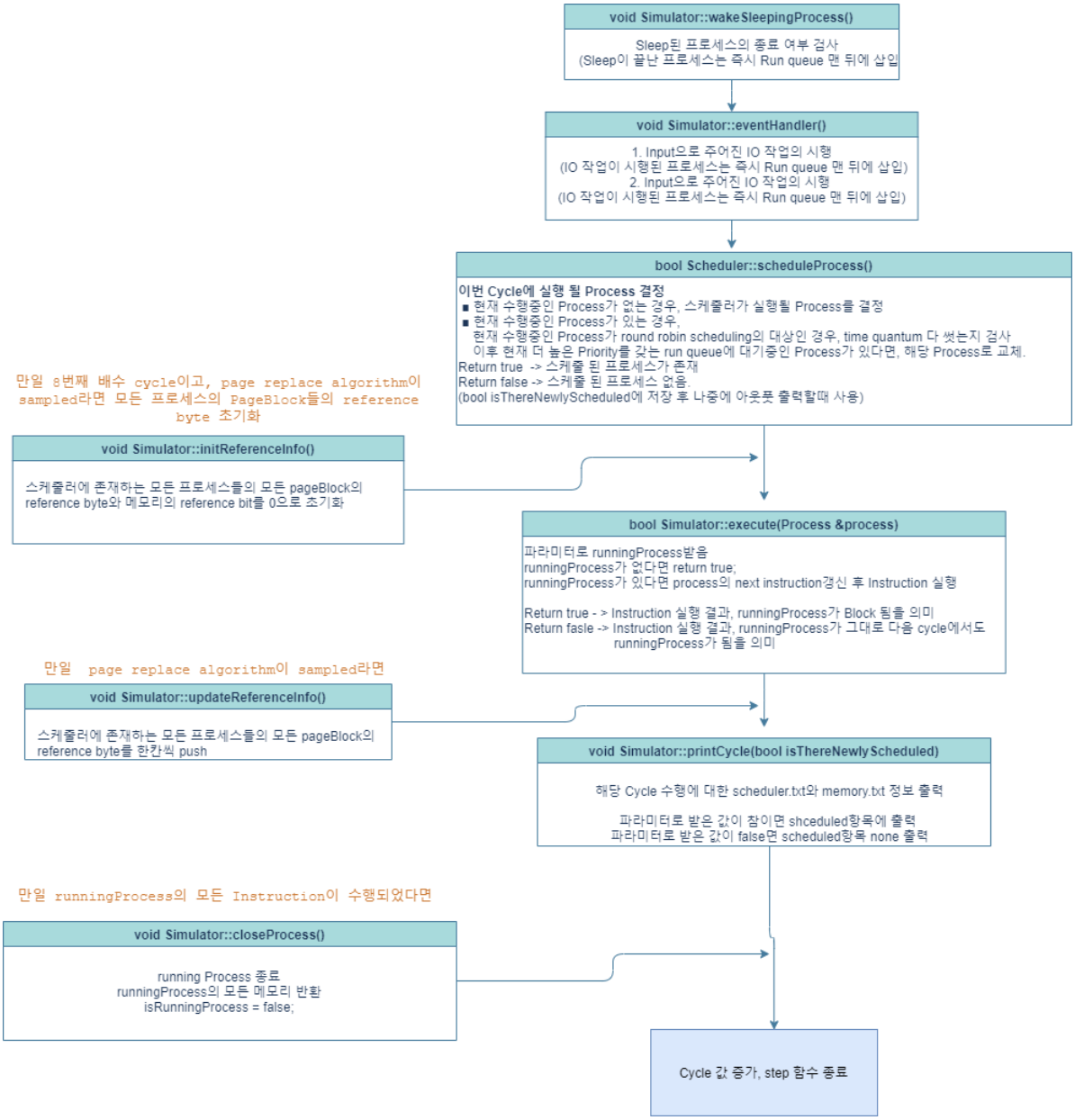
```
void Simulator::simulate() {
    /// trigger method for simulating system

    step();
    while(!checkTerminateCondition()){
        step();
    }
    fprintf(mem,"page fault = %d\n", numPageFault);
}
```

simulate method는 각 cycle의 동작을 step함수로 처리하며 매 step 함수를 반복문으로 실행한다. 종료 검사에 해당되면 pageFault 값을 출력 후 종료한다. 종료 검사는 checkTerminateCondition()함수로 수행하며 이 함수는 scheduler내의 모든 Queue, IOWaitList, SleepList에 남아있는 프로세스가 있는지 검사한다.

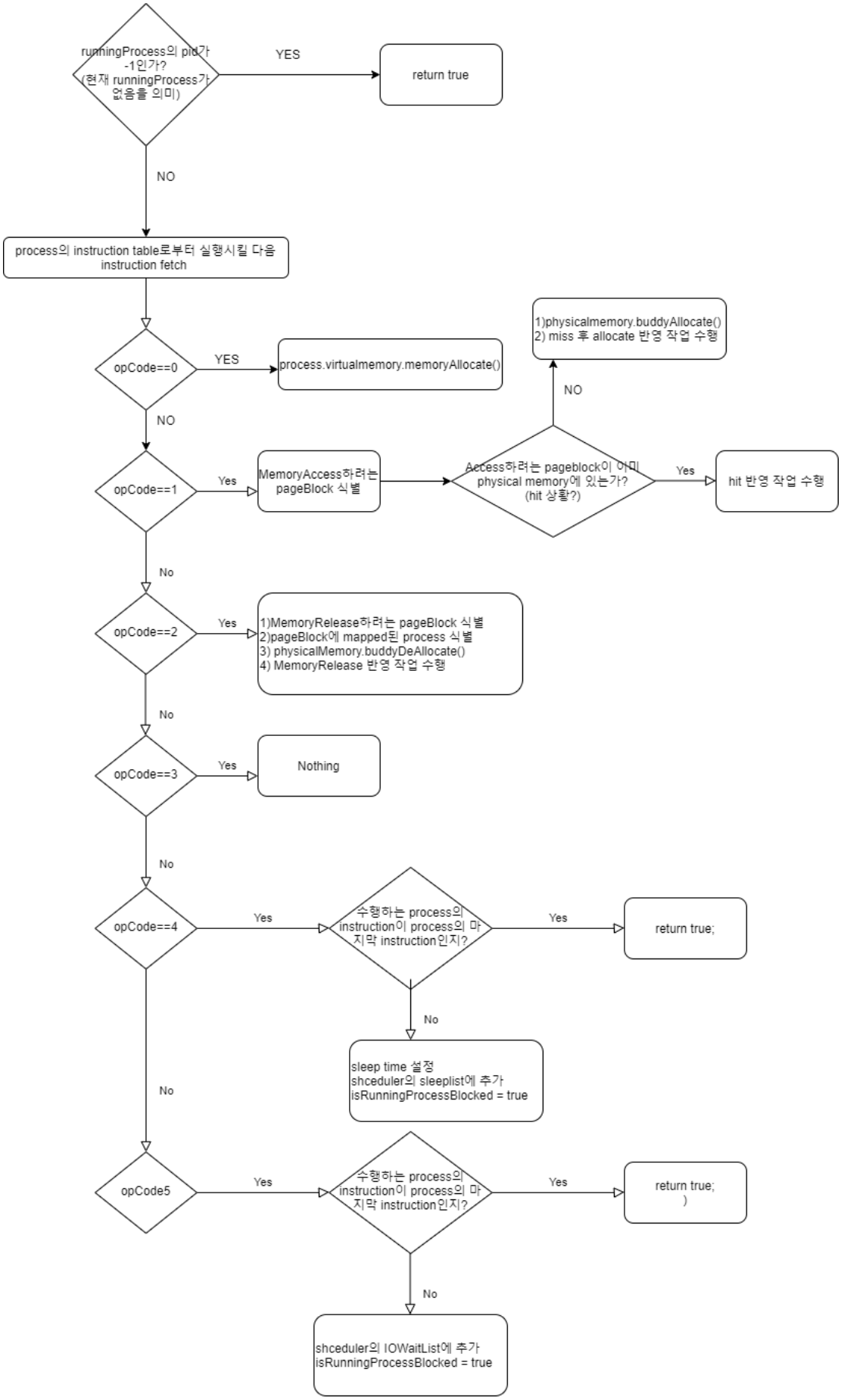
3. **Simulator.step()**

시뮬레이터의 각 cycle의 동작을 의미한다. step() 메소드의 흐름도는 다음과 같다.



Process의 Instruction 수행 구현 방법

Process의 Instruction 수행은 위 step 함수의 흐름도에서 bool Simulator::execute(Process &process)에서 행해진다. 파라미터로 실행시킬, 스케줄러에 의해 스케줄된 runningProcess를 받는다.다음은 execute 메소드의 흐름도이다.


runningProcess의 pid가 -1인가? (현재 runningProcess가 없음을 의미)
YES
return true
NO
process의 instruction table로부터 실행시킬 다음 instruction fetch
opCode==0
YES
process.virtualmemory.memoryAllocate()
NO
opCode==1
Yes
MemoryAccess하려는 pageBlock 식별
Access하려는 pageblock이 이미 physical memory에 있는가? (hit 상황?)
Yes
hit 반영 작업 수행
NO
1)physicalmemory.buddyAllocate()
2) miss 후 allocate 반영 작업 수행
No
opCode==2
Yes
1)MemoryRelease하려는 pageBlock 식별
2)pageBlock에 mapped된 process 식별
3) physicalMemory.buddyDeAllocate()
4) MemoryRelease 반영 작업 수행
No
opCode==3
Yes
Nothing
No
opCode==4
Yes
수행하는 process의 instruction이 process의 마지막 instruction인지?
Yes
return true;
No
sleep time 설정
shceduler의 sleeplist에 추가
isRunningProcessBlocked = true
No
opCode5
Yes
수행하는 process의 instruction이 process의 마지막 instruction인지?
Yes
return true;
)
No
shceduler의 IOWaitList에 추가
isRunningProcessBlocked = true
이후 공통적으로,
실행한 process가 round robin scheduling의 대상이라면 -> time quantum - 1
return isRunningProcessBlocked

return value인 isRunningProcessBlocked는 해당 cycle의 instruction 수행 시 running Process가 block되는 상황(sleep, IOWait, 혹은 프로세스가 이제 종료될 예정) 임을 simulator의 step 함수에 알려주게된다.

opcode 1, memoryAccess와 opcode 2 memoryRelease의 맨 마지막 "반영작업"이라 함은 각각의 instruction수행의 결과를 virtualmemory의 memory에 page Table bits 정보들을 업데이트 한다는 의미이다. 반영작업은 pageReplace Algorithm에 따라 각각 상이하다.

버디 시스템 구현 방법

버디 시스템으로 physical memory를 할당, 해제, 그리고 page replacement의 구현은 PhysicalMemory.cpp에서 구현했다. 버디 시스템의 구현에 해당하는 소스 코드는 다음 사이트의 도움을 받아 작성했다.

https://www.geeksforgeeks.org/buddy-memory-allocation-program-set-2-deallocation/

먼저, physicalMemory class에는 vector * freeBlocks가 존재한다. freeBlocks array는 시스템의 물리메모리에 사용할 수 있는 빈 FrameBlock들을 담고 있다. freeBlocks의 각 인덱스에는 벡터가 존재한다. 각 인덱스는 해당하는 인덱스의 벡터가 담고 있는 빈 frameBlock 공간의 크기(2^인덱스)를 의미한다. 예를 들어 freeBlocks[1]에 있는 벡터에는 크기가 2^0인 free FrameBlock들이 담기고 freeBlocks[5]에 있는 벡터에는 크기가 2^5인 free FrameBlock이 담긴다.

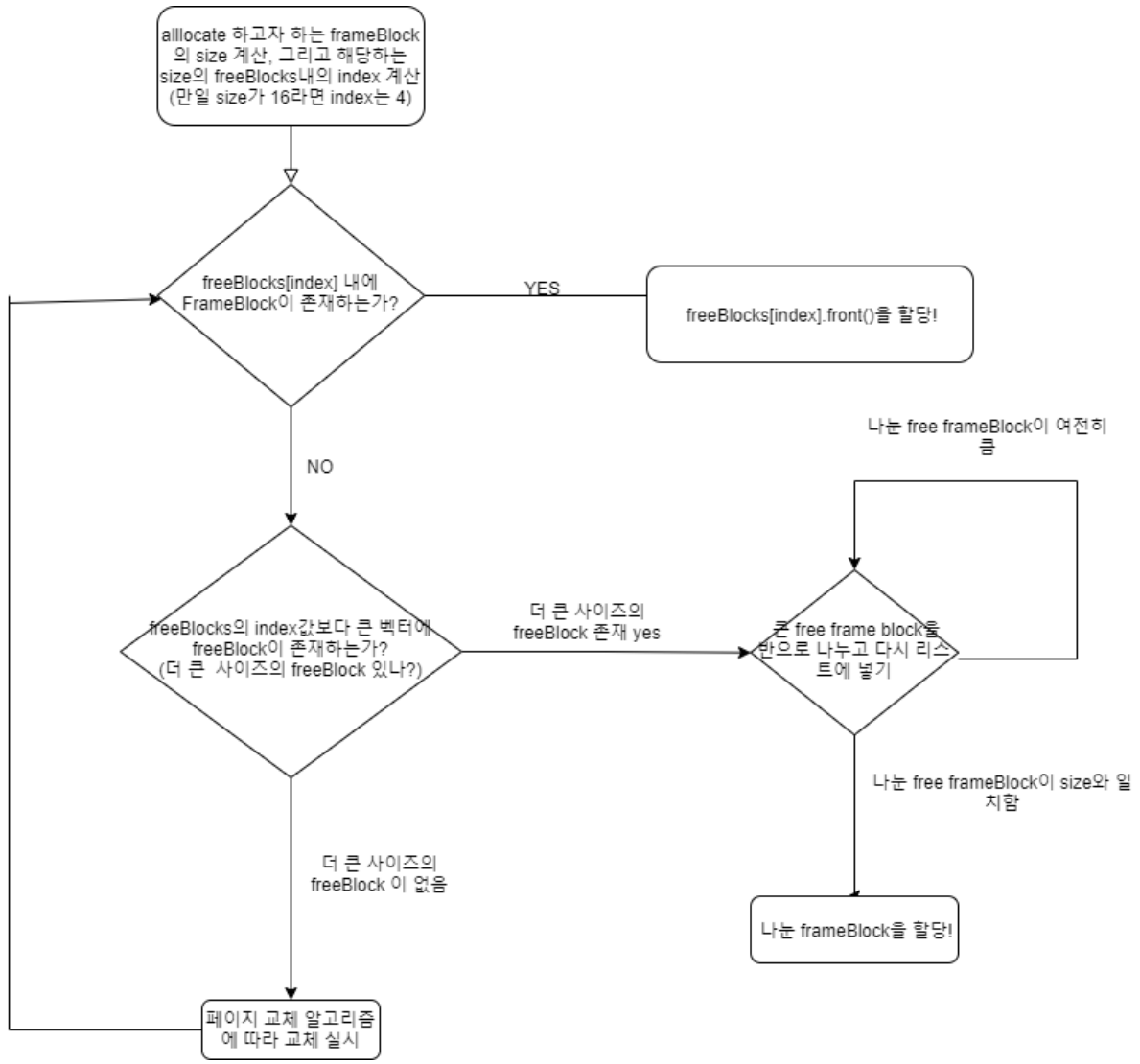
다음은 과제 명세서의 예시 설정대로 생성된 physical memory의 freeBlocks의 초기값이다. vector 내의 FrameBlock은 편의상 시작 주소값과 끝 주소 값으로 나타냈다

인덱스	0	1	2	3	4	5	6
vector	{}	{}	{}	{}	{}	{}	{(0,63)}

PhysicalMemory 클래스는 메소드로 FrameBlock buddyAllocate(Process& requestingProcess, PageBlock& accessingPageBlock, Scheduler &scheduler) 과 void buddyDeAllocate(Process& requestingProcess, FrameBlock victim) 을 가진다. 각각 물리메모리에 메모리 할당과 메모리 해제를 담당하는 메소드들이다.

다음은 memory allocate을 수행하는 메소드는 FrameBlock buddyAllocate(Process& requestingProcess, PageBlock& accessingPageBlock, Scheduler &scheduler) 이다. allocate가 완료된 FrameBlock을 return한다. 파라미터로는 allocate을 하려는 Pageblock, 그리고 그 pageBlock을 가지고 있는 Process, 그리고 scheduler을 받는다.

buddyAllcoate메소드의 실행 흐름은 다음과 같다.



다음은 freeBlock 자료구조가 allocate 과정에서 어떤 흐름을 가지는지, 과제 스펙의 예시 7cycle~8cycle을 들어 설명하겠다.

```
[7 Cycle] Input: Pid[1] Function[IOWAIT]
>> Physical Memory:          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(PID):   |0000|0000|0000|0000|1111|1111|1111|2222|2222|2222|2222|2222|2233|3333|3333|3333|
>> pid(0) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid): |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[8 Cycle] Input: Pid[0] Function[ACCESS] Page ID[1] Page Num[12]
>> Physical Memory:          |0000|0000|0000|0000|----|----|----|----|
>> pid(0) Page Table(PID):   |0000|0000|0000|0000|1111|1111|1111|2222|2222|2222|2222|2222|2233|3333|3333|3333|
>> pid(0) Page Table(AID):   |----|----|----|----|0000|0000|0000|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid): |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

7cycle

아직 allocated 된 frameBlock이 없으므로 freeBlocks의 상태는 초기상태와 같다.

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{}	{0,32}

8 cycle

먼저 allocate하고자 하는 buddy의 size는 16, index는 4이다. 첫째로, freeBlocks[4]의 벡터에 frameBlock이 있는지 확인하지만 없으므로 split을 진행한다. 이후 freeblocks[4]의 첫번째 frameBlock을 할당해준다

초기상태

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{}	{(0,32)}

split진행후

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(0,15), (16,32)}	{}

할당 진행 후

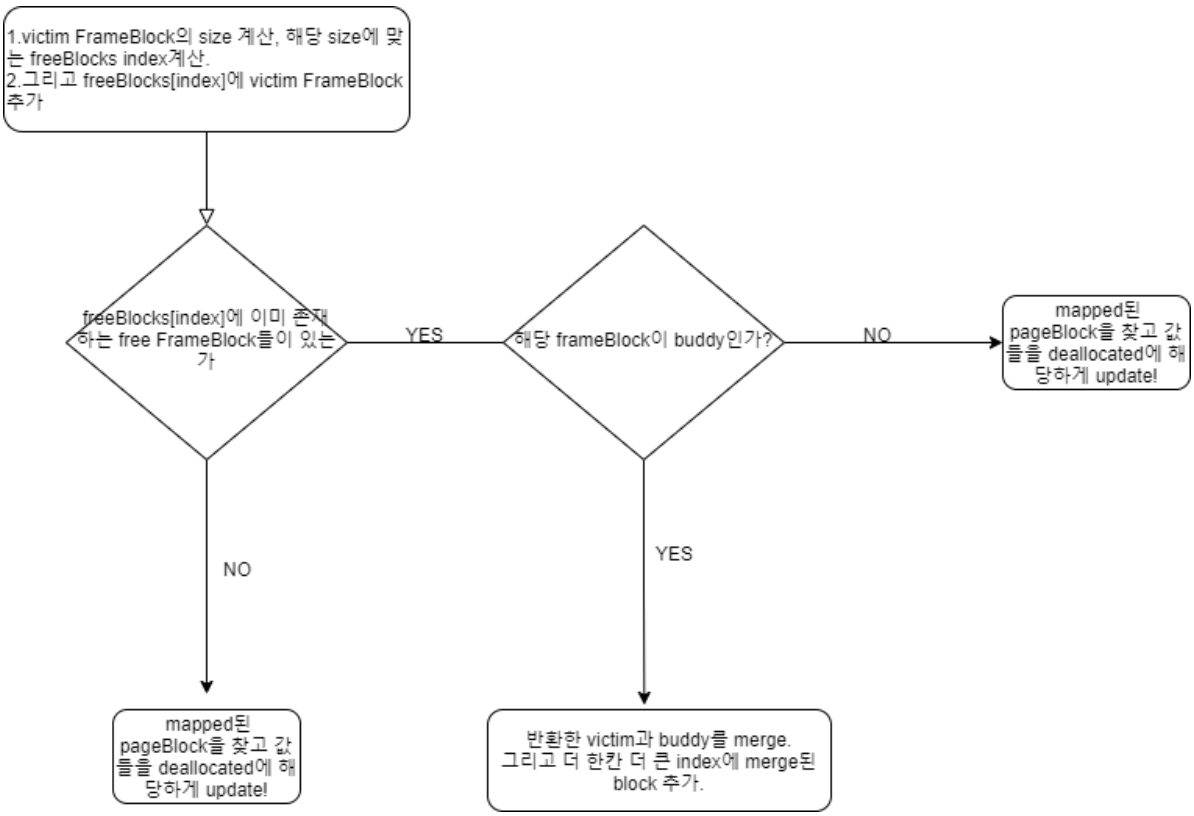
인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(16,32)}	{}

할당을 위해 선택된 frameBlock -> (0 , 16)

위 예시와 같은 흐름으로 할당이 진행된다. 이후 선택된 frameBlock에 부가적인 정보와 스케줄러로부터 mapped된 process의 mapped된 pageBlock을 찾고 해당 pageBlock에도 부가적인 정보를 업데이트 해줌으로써 allocation과정이 끝나게된다.

다음은 물리 메모리의 frameBlock을 해제하는 메소드인 void PhysicalMemory::buddyDeAllocate(Process& requestingProcess, FrameBlock victim)의 흐름이다. 이 메소드는 해제하는 대상(victim) frameBlock과 victim 이 mapped된 Process 객체를 파라미터로 받는다

buddyDeAllcoate메소드의 실행 흐름은 다음과 같다.



다음은 freeBlock 자료구조가 buddy deallocate 과정에서 어떤 흐름을 가지는지, 앞선 과제 스펙의 예시 7cycle~8cycle에 이어 9~10 cycle를 들어 설명하겠다.

```
[9 Cycle] Input: Pid[0] Function[ACCESS] Page ID[1] Page Num[12]
>> Physical Memory:          |0000|0000|0000|0000|----|----|----|----|
>> pid(0) Page Table(PID):   |0000|0000|0000|0000|1111|1111|1111|2222|2222|2222|2222|2222|2233|3333|3333|3333|
>> pid(0) Page Table(AID):   |----|----|----|----|0000|0000|0000|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid): |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[10 Cycle] Input: Pid[0] Function[ACCESS] Page ID[2] Page Num[22]
>> Physical Memory:          |1111|1111|1111|1111|1111|1111|1111|1111|
>> pid(0) Page Table(PID):   |0000|0000|0000|0000|1111|1111|1111|2222|2222|2222|2222|2222|2233|3333|3333|3333|
>> pid(0) Page Table(AID):   |----|----|----|----|0000|0000|0000|1111|1111|1111|1111|1111|11--|----|----|----|
>> pid(0) Page Table(Valid): |0000|0000|0000|0000|0000|0000|0000|1111|1111|1111|1111|1111|1100|0000|0000|0000|
>> pid(0) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(PID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(AID):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(Valid): |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(2) Page Table(Ref):   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

9 Cycle

이미 allocate된 page를 다시 access하라는 Instruction이 수행되었으므로 Hit 상태이다. hit인 경우 추가적인 메모리 작업은 없다.

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(16,32)}	{}

10 Cycle

size가 22인 pageBlock을 물리메모리에 할당해야하는 Instruction이다. 이 경우 memory allocate의 과정에서 남은 공간이 충분치 않으므로 page replace가 필요하므로 buddyDeAllocate함수가 call되게 된다.

초기상태

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(16,32)}	{}

victim FrameBlock 선정-> (0 , 15) buddy deAllocate 실시

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(0,15),(16,32)}	{}

반환된 victim의 buddy가 이미 freeBlocks[5]에 존재 -> merge후 이동

인덱스	0	1	2	3	4	5
vector	{}	{}	{}	{}	{(0,32)}	{}

위 예시와 같은 흐름으로 해제가 진행된다. 이후 victimize된 frameBlock에 부가적인 정보와 mapped된 process의 mapped된 pageBlock을 찾고 해당 pageBlock에도 부가적인 정보를 업데이트 해줌으로써 할당 해제 과정이 끝나게된다.

Page Replacement Algorithm 구현 방법

Page Replace algorithm은 물리메모리 할당 진행 중, 물리 메모리에 남은 충분한 공간이 없는 경우, 이미 할당된 물리메모리의 FrameBlock들 중 어떤 FrameBlock을 victim으로 선정해 deAllocate할지, victimize 선택의 기준을 뜻한다. 따라서 앞선 buddyAllocate의 흐름도에서 두개의 조건을 지나친 후 마지막 단계에서 실행된다.((1)freeBlocks[index] 내에 FrameBlock이 존재하는가? (2) freeBlocks의 index 값보다 큰 벡터에 freeBlock이 존재하는가? 이후에 (3)pageReplace 진행 그리고 다시 전체 과정(1)~(3) 반복.)

- LRU Algorithm

  기본 LRU 알고리즘에서 victim 대상 선정은 간단하다. physicalMemory의 frameBlockTable중 가장 마지막에 위치한 frameBlock을 선택하면 된다. frameBlockTable은 stack의 구조로 유지되기 때문이다. 항상 최근에 memory access가 발생한 frameBlock들이 위로 올라오도록 프로그램 실행 내내 관리된다.

  frameBlockTable에는 항상 present한 frameBlock들로만 구성되어 있다. 달리 말해서, 물리 메모리 공간으로부터 deAllocate된 frameBlock은 삭제된다.

- Sampled LRU Algorithm

frameBlock table에 있는 frameBlock들 중, 가장 작은 reference byte값을 지닌 frameBlock을 victim으로 선정한다. 이때, 만일 reference byte 의 값이 동점이라면, allocation id가 작은 값이 victim으로 선정된다.(이를 위해 먼저 frameBlockTable의 allocation id 기준 정렬을 수행했다.)

우선 reference byte는 모든 프로세스의 가상메모리의 pageBlock들마다 존재한다. 따라서 frameBlock Table을 정렬한 후, 모든 frameBlock들 마다 loop를 돌며 각 frameBlock의 mappedProcessId 값을 통해 스케줄러에 존재하는 mapped process를 찾는다. 이후 해당하는 mapped process의 모든 pageBlock들을 다시 검색하며 allocation id가 동일한 pageBlock을 선출한다. 해당하는 pageBlock이 찾아졌다면 reference byte의 값을 10진수로 계산하여 referByteValue[]라는 int형 배열에 값을 저장한다.

referByteValues 배열에 모든 frameBlock에 해당하는 10진수 reference byte값이 저장되었다면 가장 작은 값을 찾아낸다. 이때 값이 같은 frameBlock들이 존재해도 allocation id 순으로 정렬을 해 놓은 상태로 진행했으므로, 과제의 조건에 부합되게 선출된다.

기본 LRU와 마찬가지로 frameBlockTable에는 항상 present한 frameBlock들로만 구성되어 있다. 달리 말해서, 물리 메모리 공간으로부터 deAllocate된 frameBlock은 삭제된다.

- Clock Algorithm

  clock algorithm은 시스템에 전역변수인 clock pointer을 두어 page replace의 상황, victimize 상황이 도래하면 clock pointer가 가리키는 frameBlockTable의 인덱스부터 시작해서 allocation id 순서대로 reference bit를 검사한다. 이때 reference bit가 0이면 victimize의 대상으로 선출하며 1인 경우 reference bit을 0으로 설정하는 second chance를 준다.

  의외로 간단한 구현이었으나 clock pointer을 정확하게 원하는대로 가리키게 하는 것이 구현에 있어 중요했던 부분이었다.

  특히, 다음과 같은 특수한 상황들의 경우가 그러하다.

  상황:

  1. 물리 메모리에 있는 AID 목록: 0, 1, 3
  2. 페이지 교체가 일어나 AID 3이 메모리에서 빠짐 (0, 1)
  3. AID 4가 물리 메모리에 들어옴 (0, 1, 4)

  이상황에서 page replace가 다시 발생하면 다음 clock pointer는 과제 스펙의 설명대로, 4를 가리키게끔 구현했다.

프로그램의 동작 검증

작성한 프로그램의 스케줄링 기능, 그리고 페이지 교체 정책 옵션에 따른 메모리 관리 기능을 검증하기 위해 총 3가지 인풋 테스트 케이스를 준비해서 검증했다 .

테스트 케이스 1

테스트케이스 1번은 스케줄링 기능 검증에 목적을 두고 테스트 케이스를 디자인했다. 편의를 위해 메모리 관련 Instruction은 일체 사용하지 않고 오로지 opcode 3~5번 (NON_MEMORY, Sleep, INPUT) Instruction으로 구성한 프로세스들로만 테스트를 진행했다.

사용한 인풋 파일 구성과 설명

input.txt

```
6 2048 1024 32
0 program0 5
0 program1 5
0 program2 5
30 program4 2
30 program3 1
45 INPUT 4
```

- round robin scheduling 의 대상인 priority 5인 program 0, program1, program2
- FCFS 의 대상인 priority 1, 2 인 program3, program 4
- 프로세스가 생성되는 순으로 pid가 부여되므로, program0의 pid는 0, program1의 pid는 1, program2의 pid는 2, **program3의 pid는 4, program4의 pid는 3**가 부여될 것임.

program0 program1 program2

```
30
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
```

program0, program1, program2는 모두 동일함. - 30개의 opcode 3의 instruction으로 구성됨

program3

```
13
3 0
3 0
3 0
3 0
```

```
3 0
3 0
3 0
3 0
5 0
3 0
3 0
3 0
3 0
4 100
```

line 9에 IO-Wait Instruction 존재
마지막 라인에 sleep Instruction 존재, 하지만 프로세스의 마지막 라인이므로 실제 스케줄링의 sleep queue에는 register되지 않을 것임.

program4

```
13
3 0
4 4
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
5 0
```

line 2에 sleep Instruction 존재
마지막 라인 IOwait instruction 존재, 하지만 프로세스의 마지막 라인이므로 실제 스케줄링의 sleep queue에는 register되지 않을 것임.

테스트 결과

(메모리 관련 Instruction이 전혀 없기에 page option은 default("lru")로 시행)

scheduler.txt

0 cycle

```
[0 Cycle] Scheduled Process: 0 program0 (priority 5)
Running Process: Process#0(5) running code program0 line 1(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
```

```
SleepList: Empty
IOWait List: Empty
```

0 cycle의 input.txt의 event가 잘 처리되었음을 확인.

9~10 cycle

```
[9 Cycle] Scheduled Process: None
Running Process: Process#0(5) running code program0 line 10(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty

[10 Cycle] Scheduled Process: 1 program1 (priority 5)
Running Process: Process#1(5) running code program1 line 1(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

program0의 time quantum을 다 소진해서 10cycle 시점에 다음 순서인 program1로 scheduling되는 것을 확인

program 0은 다시 runQueue로 재등록되어 대기중인것을 확인.

19~20 cycle

```
[19 Cycle] Scheduled Process: None
Running Process: Process#1(5) running code program1 line 10(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
```

```
IOWait List: Empty

[20 Cycle] Scheduled Process: 2 program2 (priority 5)
Running Process: Process#2(5) running code program2 line 1(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 0(program0) 1(program1)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

9~10 cycle과 같은 이유로 같은 동작을 함을 확인

30 cycle

```
[30 Cycle] Scheduled Process: 4 program3 (priority 1)
Running Process: Process#4(1) running code program3 line 1(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: 3(program4)
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 0(program0) 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

input.txt에 의해 program3와 program4가 생성됨을 확인.
우선순위가 높은 program3가 실행됨을 확인
program3의 pid는 4임을 확인 (inputs.txt에서 program4가 먼저 시스템에 등록됨, program4의 pid는 3)

38~41cycle

```
[38 Cycle] Scheduled Process: None
Running Process: Process#4(1) running code program3 line 9(op 5, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: 3(program4)
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 0(program0) 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: 4(program3)
```

```
[39 Cycle] Scheduled Process: 3 program4 (priority 2)
Running Process: Process#3(2) running code program4 line 1(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 0(program0) 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: 4(program3)

[40 Cycle] Scheduled Process: None
Running Process: Process#3(2) running code program4 line 2(op 4, arg 4)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 0(program0) 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: 3(program4)
IOWait List: 4(program3)
[41 Cycle] Scheduled Process: 0 program0 (priority 5)
Running Process: Process#0(5) running code program0 line 11(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: 3(program4)
IOWait List: 4(program3)
```

38 사이클에서 IO wait가 정상적으로 동작.(program3은 45 cycle에 input event를 실행하고 다시 스케줄 될 예정)
39 사이클에서 다음 우선순위인 program4가 정상적으로 스케줄된 후 실행
40 사이클에서 sleep Instruction 정상적으로 실행. (program4는 40~43사이클까지 자고 44 사이클에 다시 스케줄 될 예정)
41 사이클에서 다음 우선순위인 program0이 정상적으로 스케줄된 후 실행 재게

44 cycle

```
[44 Cycle] Scheduled Process: 3 program4 (priority 2)
Running Process: Process#3(2) running code program4 line 3(op 3, arg 0)
RunQueue 0: Empty
```

```
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: 4(program3)
```

44사이클에 sleep에서 일어난 program4 실행 재개

45 cycle

```
[45 Cycle] Scheduled Process: 4 program3 (priority 1)
Running Process: Process#4(1) running code program3 line 10(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: 3(program4)
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

45 cycle에서 program3 이 IO event를 받고 러닝 큐에 다시 재등록 된 후 우선순위가 앞서므로 스케줄되고 실행

49 cycle

```
[49 Cycle] Scheduled Process: None
Running Process: Process#4(1) running code program3 line 14(op 4, arg 100)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: 3(program4)
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

49 사이클에서 프로그램3의 마지막 라인 실행.
sleep Instruction이지만 sleep List에 추가하지 않음.

60cycle

```
[60 Cycle] Scheduled Process: None
Running Process: Process#3(2) running code program4 line 14(op 5, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: 1(program1) 2(program2) 0(program0)
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

60 cycle에 program4의 마지막 라인 실행

IO wait Instruction이지만 waitList에 넣지 않음.

117 cycle

```
[117 Cycle] Scheduled Process: None
Running Process: Process#0(5) running code program0 line 30(op 3, arg 0)
RunQueue 0: Empty
RunQueue 1: Empty
RunQueue 2: Empty
RunQueue 3: Empty
RunQueue 4: Empty
RunQueue 5: Empty
RunQueue 6: Empty
RunQueue 7: Empty
RunQueue 8: Empty
RunQueue 9: Empty
SleepList: Empty
IOWait List: Empty
```

117 cycle을 마무리로 전체 시뮬레이션 종료.

프로세스의 총 instruction 개수

(program0~program2) 30 * 3 = 90
program3~ program4 = 14 * 2 = 28

90 + 28 = 118개 0~117과 맞음을 확인.

테스트 케이스 2

테스트케이스 2번은 sampled algorithm 을 검증하려는 목적을 두고 테스트 케이스를 디자인했다. 실행 option은 sampled이다.

시용한 인풋 파일 구성과 설명

input.txt

```
4 2048 1024 32
0 program0 1
0 program1 3
16 INPUT 0
16 INPUT 1
```

program0

```
21
0 5
0 5
0 5
0 5
1 3
1 2
1 1
1 0
1 1
1 2
1 0
1 3
5 0
1 2
4 8
1 3
4 3
1 1
0 5
1 4
1 2
```

program1

```
11
0 4
1 0
5 0
3 3
0 5
0 5
1 1
1 2
4 2
1 0
3 0
```

테스트 결과

~7 cycle

```
[7 Cycle] Input : Pid[0] Function[ACCESS] Page ID[0] Page Num[5]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

program 0의 모든 pageBlock들이 물리메모리에 할당된 상태.

8 cycle

```
[8 Cycle] Input : Pid[0] Function[ACCESS] Page ID[1] Page Num[5]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0111|1100|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

8사이클이 되면서 모든 reference byte와 reference bit가 초기화된다. 이때 page id가 1인(allocation id 2) pageblock은 다시 access했으므로 referencebit가 1임을 확인.

<실행 후 reference byte 상황>

```
program0 pageID0 <1 0 0 0 0 0 0 0 >
program0 pageID1 <1 0 0 0 0 0 0 0 >
program0 pageID2 <1 0 0 0 0 0 0 0 >
program0 pageID3 <1 0 0 0 0 0 0 0 >
```

9~14 cycle

```
[9 Cycle] Input : Pid[0] Function[ACCESS] Page ID[2] Page Num[5]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[10 Cycle] Input : Pid[0] Function[ACCESS] Page ID[0] Page Num[5]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[11 Cycle] Input : Pid[0] Function[ACCESS] Page ID[3] Page Num[5]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

```
[12 Cycle] Input : Pid[0] Function[IOWAIT]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[13 Cycle] Input : Pid[1] Function[ALLOCATION] Page ID[0] Page Num[4]
>> Physical Memory:           |0000|0000|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[14 Cycle] Input : Pid[1] Function[ACCESS] Page ID[0] Page Num[4]
>> Physical Memory:           |4444|----|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |1111|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

13 cycle실행 이후 reference byte 상황

```
program0 pageID0 <1 0 0 0 0 0 0 0 >
program0 pageID1 <1 0 0 0 0 0 0 0 >
program0 pageID2 <1 0 0 0 0 0 0 0 >
program0 pageID3 <1 0 0 0 0 0 0 0 >
```

14 cycle에서 page 교체가 필요! -> 모든 reference byte가 동점 -> allocation id가 0 인 pageBlock인 program0의 pageID 3이 victimize

이때 valid bit과 reference bit도 바뀌는 것을 확인

16 cycle

```
[15 Cycle] Input : Pid[1] Function[IOWAIT]
>> Physical Memory:           |4444|----|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |1111|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[16 Cycle] Input : Pid[0] Function[ACCESS] Page ID[2] Page Num[5]
>> Physical Memory:           |4444|----|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

16 cycle이므로 reference byte update.
실행 Instruction은 program0의 pageID 2를 접근. -> program0 pageID2는 reference bit가 1임을 확인.

16cycle 실행이후 reference byte 상태

```
program0 pageID0 <1 1 0 0 0 0 0 0 >
program0 pageID1 <1 1 0 0 0 0 0 0 >
program0 pageID2 <1 1 0 0 0 0 0 0 >
program0 pageID3 <0 1 0 0 0 0 0 0 >
program1 pageID0 <1 0 0 0 0 0 0 0 >
```

20~21 cycle

```
[20 Cycle] Input : Pid[1] Function[ALLOCATION] Page ID[2] Page Num[5]
>> Physical Memory:           |4444|----|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[21 Cycle] Input : Pid[1] Function[ACCESS] Page ID[1] Page Num[5]
>> Physical Memory:           |5555|5555|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5---|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|1111|1000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|1111|1000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

21 cycle에 page 교체 필요!

이때 20cycle까지의 reference byte상태를 보았을때, program1 pageID0이 가장 작은 reference Byte 값을 지니므로 victimize

22 cycle

```
[22 Cycle] Input : Pid[1] Function[ACCESS] Page ID[2] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
```

페이지 교체 발생. allocation id 가 5인 program1의 pageID 1이 교체됨.

23~24 cycle

```
[23 Cycle] Input : Pid[1] Function[SLEEP]
>> Physical Memory:           |6666|6666|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|

[24 Cycle] Input : Function[NO-OP]
>> Physical Memory:           |6666|6666|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

23 이후 24 시작 update!

```
program0 pageID0 <0 1 1 0 0 0 0 0 >
program0 pageID1 <0 1 1 0 0 0 0 0 >
program0 pageID2 <1 1 1 0 0 0 0 0 >
program0 pageID3 <0 0 1 0 0 0 0 0 >
program1 pageID0 <0 1 0 0 0 0 0 0 >
program1 pageID1 <0 0 0 0 0 0 0 0 >
program1 pageID2 <1 0 0 0 0 0 0 0 >
```

25~26 cycle

```
[25 Cycle] Input : Pid[0] Function[ACCESS] Page ID[3] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|0000|0000|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0001|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[26 Cycle] Input : Pid[0] Function[SLEEP]
>> Physical Memory:           |6666|6666|1111|1111|0000|0000|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0001|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

25 cycle 에서 page replacement 발생

이때 program0-pageID 0, progrm0-pageID 1 동점 - > allocation ID가작은 program0-pageID1이 victimize.

26~27 cycle

```
[26 Cycle] Input : Pid[0] Function[SLEEP]
>> Physical Memory:           |6666|6666|1111|1111|0000|0000|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1111|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0001|1111|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[27 Cycle] Input : Pid[1] Function[ACCESS] Page ID[0] Page Num[4]
>> Physical Memory:           |6666|6666|1111|1111|4444|----|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |1111|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

27 cycle page Replacement 발생. 가장 최근에 access되고 아직 reference byte를 갱신 안했으므로, allocation id 0인 program0-pageID3이 교체된 것을 확인.

28~29 cycle

```
[28 Cycle] Input : Pid[1] Function[NON-MEMORY]
>> Physical Memory:           |6666|6666|1111|1111|4444|----|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |1111|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[29 Cycle] Input : Pid[0] Function[ACCESS] Page ID[1] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0111|1100|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

29 cycle page Replacement 발생.

바로 전 교체와 같은 이유로 교체 발생.

30~31 cycle

```
[30 Cycle] Input : Pid[0] Function[ALLOCATION] Page ID[4] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|2222|2222|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|4444|4---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1110|0000|0000|0---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0111|1100|0000|0000|0000|0---|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[31 Cycle] Input : Pid[0] Function[ACCESS] Page ID[4] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|7777|7777|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|4444|4---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|7777|7---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1110|0000|1111|1---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|1111|1---|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

같은 이유로 31 cycle에서 교체

32~34 cycle

```
[32 Cycle] Input : Pid[0] Function[ACCESS] Page ID[2] Page Num[5]
>> Physical Memory:           |6666|6666|1111|1111|7777|7777|3333|3333|
>> pid(0) Page Table(PID):    |0000|0111|1122|2223|3333|4444|4---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID):    |3333|3222|2211|1110|0000|7777|7---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1000|0011|1110|0000|1111|1---|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Ref):    |0000|0000|0011|1110|0000|0000|0---|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[33 Cycle] Input : Pid[1] Function[NON-MEMORY]
>> Physical Memory:           |6666|6666|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|

[34 Cycle] Input : Pid[1] Function[NON-MEMORY]
>> Physical Memory:           |6666|6666|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|1111|1222|22--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|5555|5666|66--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0111|11--|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|00--|----|----|----|----|----|----|----|----|----|----|----|----|
```

31 cycle 이후 32 cycle 시작 시 reference byte update.

```
program0 pageID0 <0 0 1 1 0 0 0 0 >
program0 pageID1 <0 0 1 1 0 0 0 0 >
program0 pageID2 <0 1 1 1 0 0 0 0 >
program0 pageID3 <0 0 0 1 0 0 0 0 >
program0 pageID4 <1 0 0 0 0 0 0 0 >
program1 pageID0 <0 0 1 0 0 0 0 0 >
program1 pageID1 <0 0 0 0 0 0 0 0 >
program1 pageID2 <0 1 0 0 0 0 0 0 >
```

32 사이클을 마무리로 program0 종료 메모리 반환이 잘 일어남.

테스트 케이스 3

테스트케이스 3번은 clock algorithm 을 검증하려는 목적을 두고 테스트 케이스를 디자인했다. 실행 option은 clock이다.

사용한 인풋 파일 구성과 설명

input.txt

```
2 2048 1024 32
0 program0 1
0 program1 3
```

program0

```
12
0 16
0 16
0 16
0 16
1 0
1 1
4 3
1 2
1 3
1 1
4 5
3 0
```

program1

```
7
0 16
0 16
1 0
1 1
3 0
4 2
2 1
```

테스트 결과

5 cycle

```
[5 Cycle] Input : Pid[0] Function[ACCESS] Page ID[1] Page Num[16]
>> Physical Memory:           |0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
```

5 cycle까지의 실행 결과.

6~8 cycle

```
[6 Cycle] Input : Pid[0] Function[SLEEP]
>> Physical Memory:           |0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

[7 Cycle] Input : Pid[1] Function[ALLOCATION] Page ID[0] Page Num[16]
>> Physical Memory:           |0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|

[8 Cycle] Input : Pid[1] Function[ALLOCATION] Page ID[1] Page Num[16]
>> Physical Memory:           |0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(VALID):  |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
```

9 cycle

```
[9 Cycle] Input : Pid[0] Function[ACCESS] Page ID[2] Page Num[16]
>> Physical Memory:           |2222|2222|2222|2222|1111|1111|1111|1111|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|----|----|----|----|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|1111|1111|1111|1111|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|

[10 Cycle] Input : Pid[0] Function[ACCESS] Page ID[3] Page Num[16]
>> Physical Memory:           |2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|1111|1111|1111|1111|1111|1111|1111|1111|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|1111|1111|1111|1111|1111|1111|1111|1111|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
```

9 cycle page replace 필요!

이때 가리키는 clock pointer 는 0이다. allocation id 0과 1 모두 reference bit가 1이므로 순회 후 결국 allocation id가 0 인 frame이 victimize된다. 교체 이후 clock pointer는 1이다.

10 cycle page replace 필요!

clock pointer는 1이며 reference bit가 0이므로 allocation ID가 1 인 frame이 victimize된다. 교체 이후 clock pointer는 2이다.

11 cycle

```
[11 Cycle] Input : Pid[0] Function[ACCESS] Page ID[1] Page Num[16]
>> Physical Memory:           |1111|1111|1111|1111|3333|3333|3333|3333|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
```

페이지 교체 필요! -> clock pointer 2 이고 allocation id 2의 reference bit가 1, allocation id3의 reference bit 도 1이므로 순회 후 다시 allocation id 2가 선택됨.

12 ~13 cycle

```
[12 Cycle] Input : Pid[0] Function[SLEEP]
>> Physical Memory:           |1111|1111|1111|1111|3333|3333|3333|3333|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|1111|1111|1111|1111|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
```

```
[13 Cycle] Input : Pid[1] Function[ACCESS] Page ID[0] Page Num[16]
>> Physical Memory:           |1111|1111|1111|1111|4444|4444|4444|4444|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |1111|1111|1111|1111|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |1111|1111|1111|1111|0000|0000|0000|0000|----|----|----|----|----|----|----|----|
```

13 cycle 페이지 교체 필요.

clock pointer 3이므로 allocation id 3의 reference bit가 0이므로 페이지 교체 실시. 교체 후 clock ponter 4

14 ~15 cycle

```
[14 Cycle] Input : Pid[1] Function[ACCESS] Page ID[1] Page Num[16]
>> Physical Memory:           |1111|1111|1111|1111|5555|5555|5555|5555|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|

[15 Cycle] Input : Pid[1] Function[NON-MEMORY]
>> Physical Memory:           |1111|1111|1111|1111|5555|5555|5555|5555|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
```

14 cycle 교체 발생

clock pointer 4, allocation id 4의 reference bit 가 1, allocation id 1 의 reference bit가 다시 1, 순회 후 allcation id 4를 교체

15~18 cylcle

```
[15 Cycle] Input : Pid[1] Function[NON-MEMORY]
>> Physical Memory:           |1111|1111|1111|1111|5555|5555|5555|5555|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|

[16 Cycle] Input : Pid[1] Function[SLEEP]
>> Physical Memory:           |1111|1111|1111|1111|5555|5555|5555|5555|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
```

```
[17 Cycle] Input : Pid[0] Function[NON-MEMORY]
>> Physical Memory:           |1111|1111|1111|1111|5555|5555|5555|5555|
>> pid(0) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(AID):    |0000|0000|0000|0000|1111|1111|1111|1111|2222|2222|2222|2222|3333|3333|3333|3333|
>> pid(0) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(0) Page Table(Ref):    |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|

[18 Cycle] Input : Function[NO-OP]
>> Physical Memory:           |----|----|----|----|5555|5555|5555|5555|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|5555|5555|5555|5555|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|1111|1111|1111|1111|----|----|----|----|----|----|----|----|
```

18 cycle program0 종료 후 메모리 반환

19 cycle

```
[19 Cycle] Input : Pid[1] Function[RELEASE] Page ID[1] Page Num[16]
>> Physical Memory:           |----|----|----|----|----|----|----|----|
>> pid(1) Page Table(PID):    |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID):    |4444|4444|4444|4444|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(VALID):  |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Ref):    |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
```

메모리 release 명령어 실행.

스케줄러의 한계

구현한 스케줄러는 multi level queue scheduling 으로 동작하며 0~ 9까지의 priority를 에 해당하는 running Queue가 존재한다. 이때 priority는 fixed되어있는 것이 한계점이라고 생각된다. 각 priority마다 priority가 높은 순서부터 낮은 순서까지 점차적으로 긴 time slice를 부여하고, time slice내에 해결하지 못한 프로세스들은 다음 priority의 큐로 넘어가게 하는 식으로 구현을 한다면 좀더 효율적인 스케줄

러 시스템이 될 것이다. 이때 마지막 큐는 더이상 내려갈 priority가 없으므로 round robin으로 처리하게 하며 밑 우선순위의 레디 큐가 실행되기 위해서는 그 위 우선순위의 모든 큐들이 비어있어야하는 조건을 부여해야한다. time slice를 주어 다음 priority로 프로세스를 넘기는 위 같은 스케줄러는, io bound process는 비교적 높은 우선순위에서 우선적으로 처리 될 것이며 cpu bound process는 아래 우선순위로 넘어가게 될 것이므로 io bound와 cpu bound process들을 균형있게 처리할 수 있는 스케줄러가 될 것이다.

각 페이지 교체 알고리즘의 성능 비교 분석

페이지 교체 알고리즘의 성능을 비교 분석하기 위해 테스트 케이스를 만들어 각 알고리즘 옵션별로 실행한 후 발생한 page fault의 수를 기록했다.

사용한 테스트 케이스 파일들.

inputs.txt

```
1 2048 1024 32
0 program0 5
```

편의를 위해 오로지 하나의 프로세스로 진행.

```
210
0 4
0 4
0 4
0 4
0 4
0 6
0 6
0 6
0 6
0 13
1 6
1 7
.... 이하 랜덤
```

총 10개의 pageblock을 process에 할당했다. 4 사이즈 5개, 6사이즈 4개 13사이즈 1개이다.
이후 200개의 opcode 1번 Instructon 을 주었으며 Instruction의 arg 는 0~9까지의 random 숫자들로 구성해 테스트를 진행했다.

200 Memory Access Instruction Test

테스트는 위와 동일한 형식의 테스트 케이스로, program 0의 instruction들의 arg만 랜덤화해서 총 3번 실시했다. 결과는 다음과 같다.

	LRU	**SAMPLED**	**CLOCK**
trial#1	115	112	119
trial#2	111	112	111
trail#3	108	106	116

전혀 유의미한 상관관계나 성능상 차이를 찾아 볼 수 없었다. 그래서 이번엔 input size를 1000개 (opcode 1 Instruction 수를 1000개) 로 늘려 다시 실험했다.

1000 Memory Access Instruction Test

	LRU	SAMPLED	CLOCK
trial#1	562	569	567
trial#2	562	566	560
trail#3	571	568	571

마찬가지로 유의미한 상관관계나 성능상 차이를 찾아 볼 수 없었다. 이번에는 pageBlock의 수와 크기가 너무 작아서 유의미한 분석결과를 못 얻는다고 생각하여 pageBlock의 수와 크기를 조정했다. 이번에는 총 32개의 동일한 크기의 pageBlock 을 할당했고 다시 1000개의 instruction 수를 사용하여 테스트를 진행했다.

새로운 program0

```
1032
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
0 2
1 15
1 31
1 4
1 12
```

```
1 2
1 6
1 19
....   이하 랜덤
```

op code가 1 이고 arg 가 0~31까지의 random한 수로 주어진 instruction이 이후 1000개까지 이어진다.

1000 Memory Access Instruction, 32 pageBlocks Test

	LRU	SAMPLED	CLOCK
trial#1	494	528	529
trial#2	492	530	517
trail#3	499	508	505

테스트 결과, 유의미하다고는 할 수 없지만, 전체적으로 LRU algorithm이 더 낮은 수의 page fault를 기록했다.

모든 memory access instruction의 arg가 랜덤이라는 점을 감안하면 내가 행했던 테스트들의 code들은 **전혀 temporal, spacial locality들을 갖춘 코드**들이 아니었다. 비록 테스트 케이스가 충분하지 않아 검증되진 않았지만 내 테스트의 결과로 알 수 있는 점은 code의 locality를 갖추지 않는 조건, 즉 완벽한 random memory access의 상황에서는 비록 효과는 미약하지만 lru가 가장 좋은 성능을 내는 알고리즘이라는 사실이다.

과제 수행시 겪었던 어려움과 해결방법

- "sampled" 알고리즘을 잘못 이해해서 처음에 구현을 잘못했었다. 매 사이클마다 reference byte를 push하고, 8cycle 마다 모든 reference bit와 reference byte를 모두 0으로 초기화한다고 이해했었지만, 이내 q&a의 부가 설명을 보고 잘못됨을 파악, 수정했다.
- Memory allocation의 buddy system을 구현하는데 어려움을 겪었다. 특히 buddy Deallocation하는 부분은 알고리즘적인 구현이 떠오르지 않아 고생했었지만 구글링의 도움을 받아 해결했다.(https://www.geeksforgeeks.org/buddy-memory-allocation-program-set-2-deallocation/) 사이트의 코드를 참고하여 과제에 맞게 재귀적 호출, 자료구조 수정 등등 부가적인 작업을 통해 buddy system을 구현해냈다.
- 보고서 과제에서, 내가 직접 고안한 input test case 3개를 검증하라는 부분을 작성할때 어려움을 겪었다. 동작을 검증하기 위해 테스트 케이스를 많이 고민했다.
- 프로그램의 규모가 크다 보니, 예상치 못한 오작동이 계속 발생했다. 오류가 계속 발생하다 보니(메모리 구현 관련) 구현한 프로그램에 확신이 들지 않아 불안했다. 후에, memory access instruction의 arg에 랜덤한 숫자를 부여하여 수많은 테스트 케이스를 진행해 보니, 발견 못한 에러를 많이 발견하게 되었고 프로그램의 디버깅 작업을 수월하게 진행할 수 있었다.
- 클래스, 구조체 등과 같은 자료구조를 좀 더 효율적으로 설계할 수 있었는데 그러지 못했던 것 같아 아쉬웠다. 큰 규모의 프로그램을 작성할 때, 우선 신중히 구현할 내용에 대해 학습하고 정리하는 시간을 가진 후에 효율적으로 자료구조들을 설계해야한다는 것을 깨달았다.