**Operating Systems**

Spring, 2019

School of Software, CAU

**Project #2**

**- Implementation of Memory Management Schemes –**

20151523 정모세

20155760장혁진

20154652이동재

요구사항 (1) :

Pintos 프로젝트 디렉터리 내 ‘threads/palloc.c’등의 소스코드를 분석하고, 현재 First Fit으로 구현된 연속된 페이지 할당 정책에 Next Fit, Best Fit, 그리고 Buddy System 을 추가한다. 이는 User Pool과 Kernel Pool에 동시에 적용될 수 있어야 한다.

쓰레드의 palloc.c 파일을 분석했을 때 page를 allocate시 palloc\_get\_multiple를 이용하여 할당 받는 다는 것을 확인하였고, 코드를 따라 확인했을 때

palloc\_get\_multiple -> bitmap\_scan\_and\_flip->bitmap\_set\_multiple과 bitmap\_scan 으로 나뉜다.

1. bitmap\_scan 비트맵을 검색하는 부분을 확인해야한다는것을 확인 후

palloc.c 파일에

Fisrt Fit, Next Fit, Best Fit, Buddy 프로그램을 선택할 수 있는 pallocator을 만들어

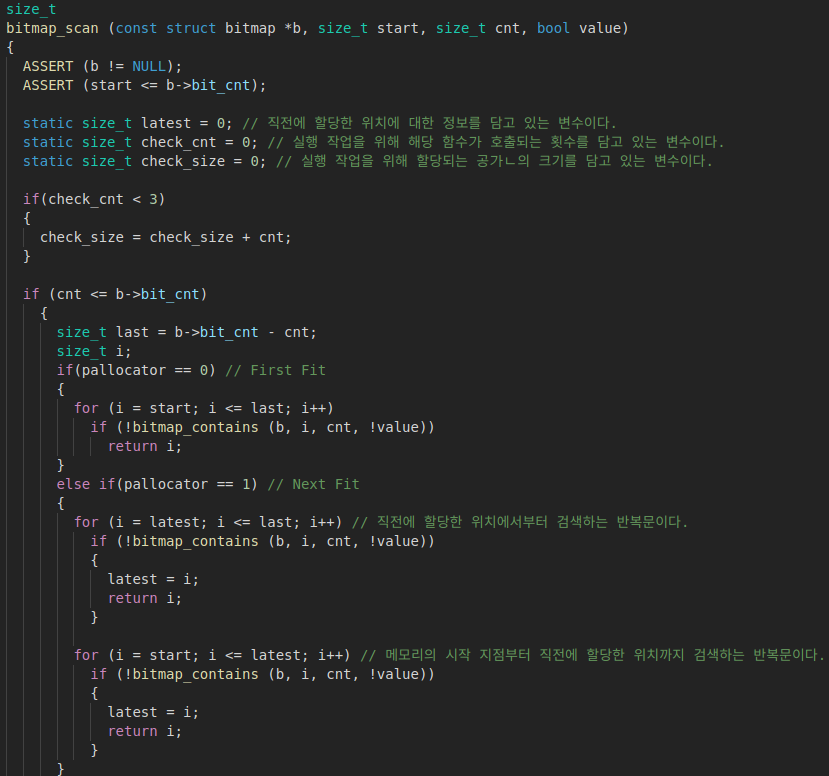
pallocator =0 -> Fisrt Fit,

pallocator =1 -> Next Fit

pallocator =2 -> Best Fit,

pallocator =3 -> Buddy 를 선택할 수있게 하였다.





bitmap\_scan 부분에서 First FIt으로 구현되어있던 부분을

0~3 까지 4가지로 if문을 걸어 주어 각각 구현을 하였다.

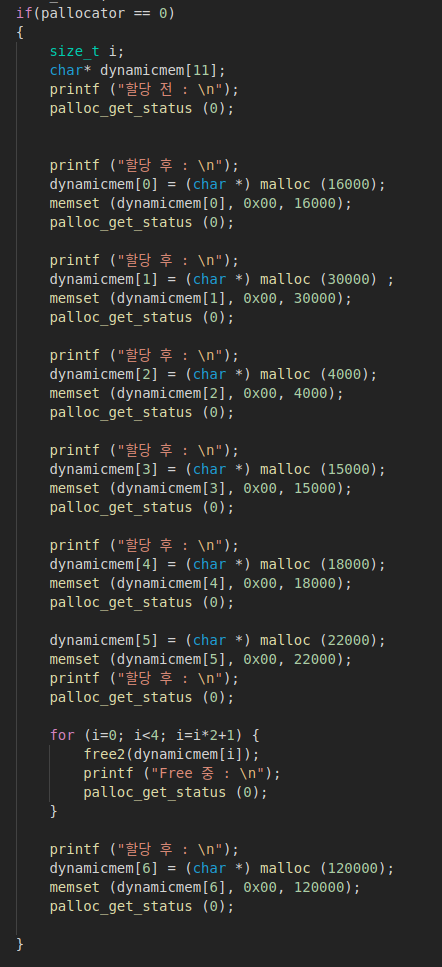
**<Test 방법>**

**cd vm/ -> ./clear.sh -> cd build -> ../../utils/pintos -q test**

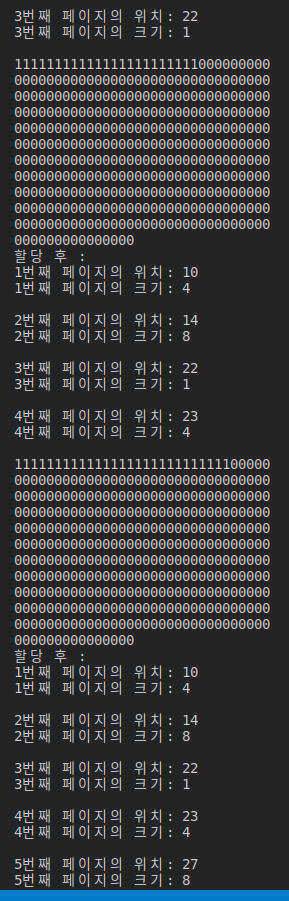
1. **First Fit의 실행**

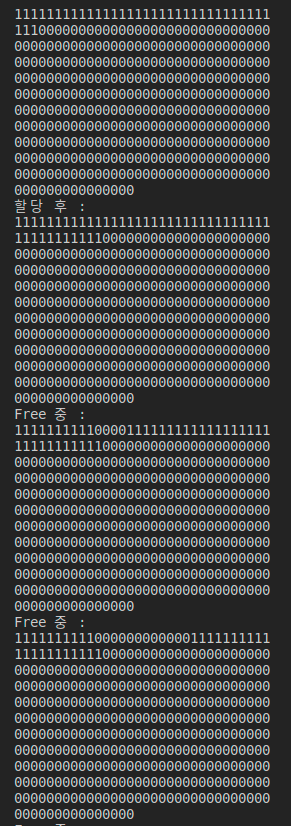
**6번의 malloc으로 할당 후 3번의 Free를 한 다음**

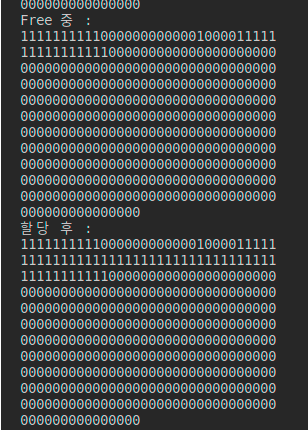
**다시 2번의 malloc 메모리할당으로 어떻게 page가 할당되는지 확인한다.**



**First Fit의 실행 결과 :**









1. **Next Fit의 구현 사진과 실행**

**First Fit과의 같은 방법으로 6번의 malloc으로 할당 후 3번의 Free를 한 다음**

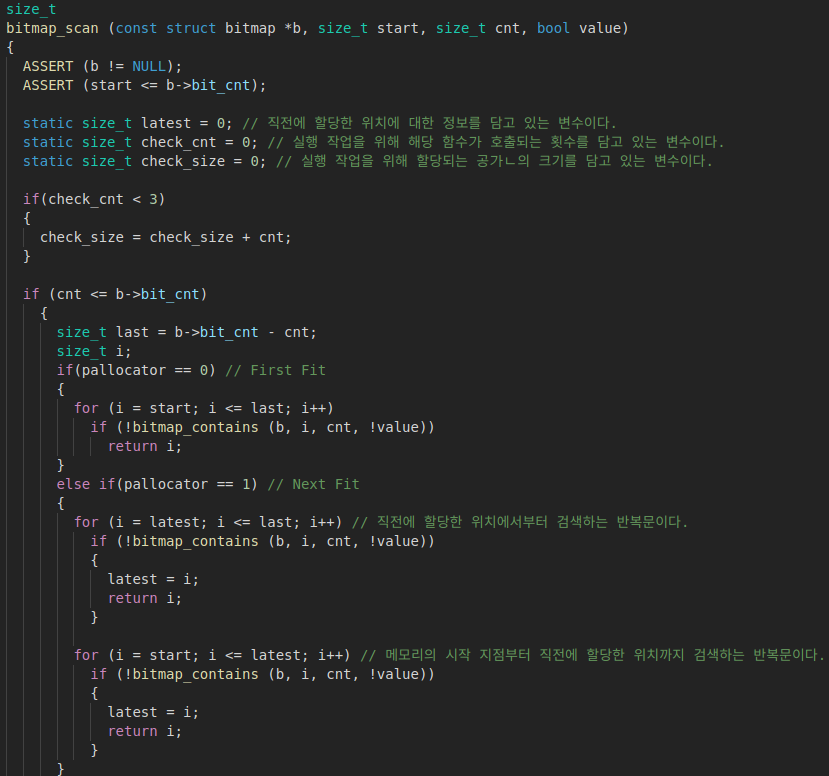
**다시 2번의 malloc 메모리할당으로 어떻게 page가 할당되는지 확인한다.**

**Lastest의 변수의 선언으로 현재 page할당 된 위치(Index)로 기억함으로써**

**기본 Fistfit이 start->last 까지 한번에 iteration을 도는것을**

**lastest->last까지 iteration 1번**

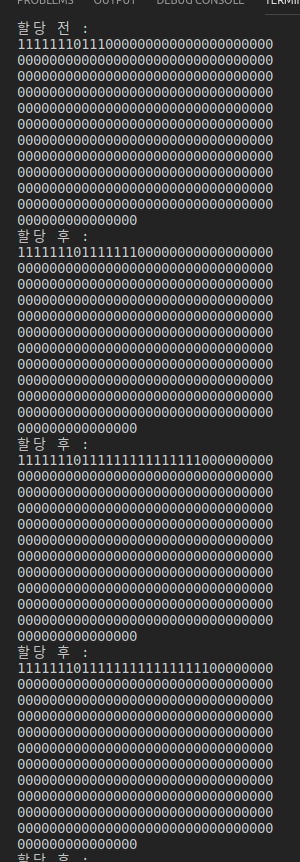
**start->lastest까지 iteration 1번을 도는 방식으로 나누어 구현하였다.**

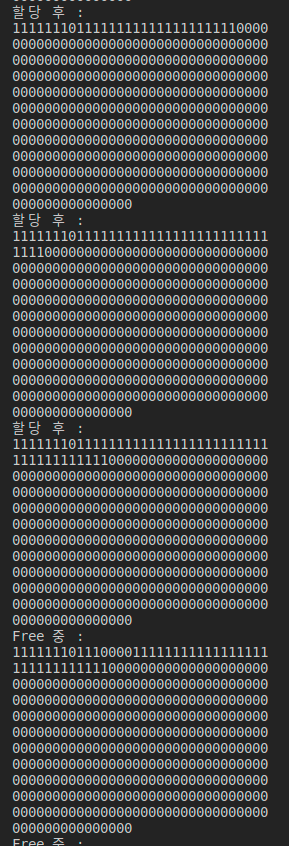


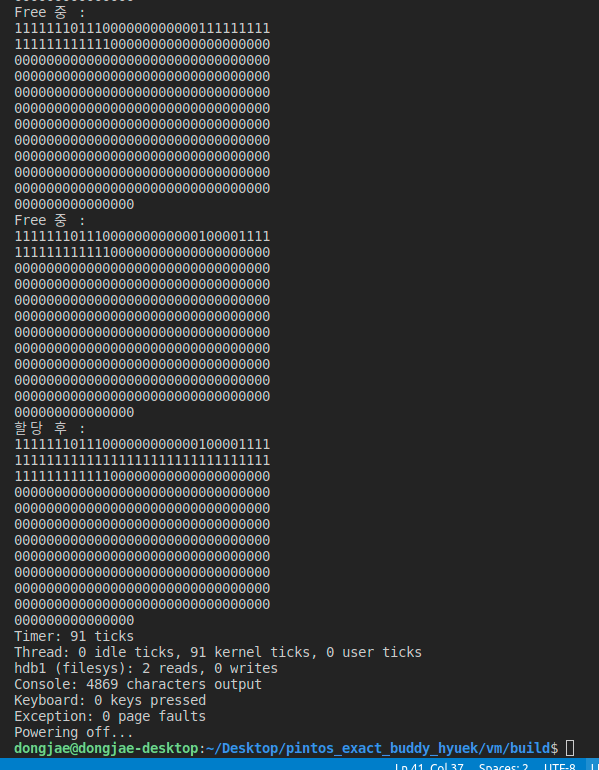


**Nextfit 실행 결과 :**

**Free후 할당을 시작할때 그때의 위치(index)를 기억해서 그 뒤부터 이어서 page를 할당하고 있다.**

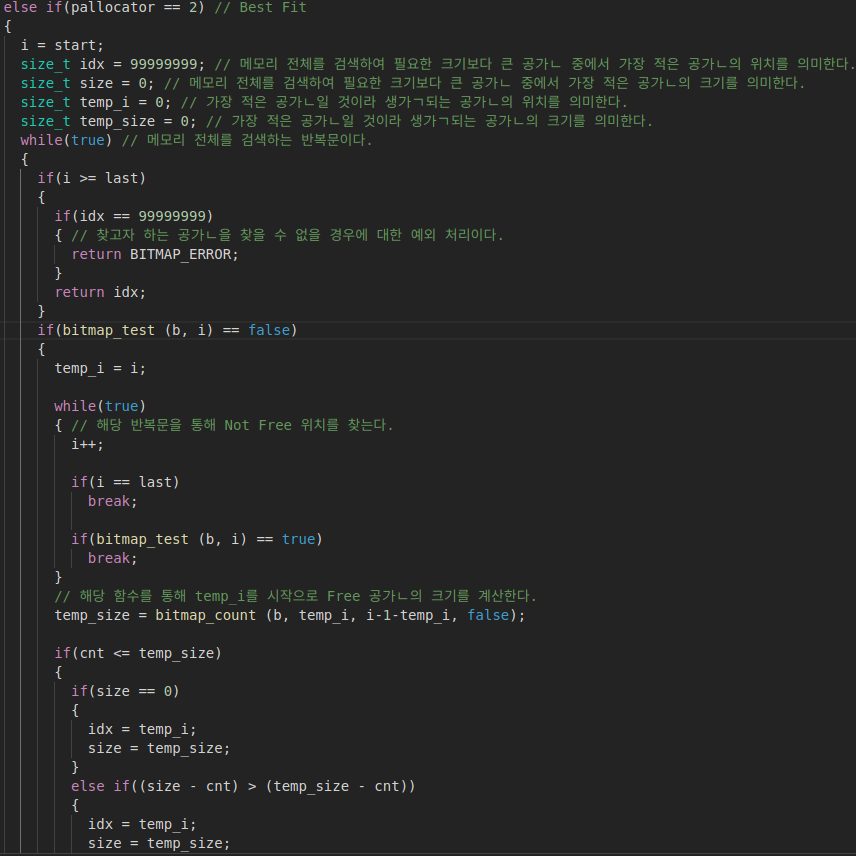




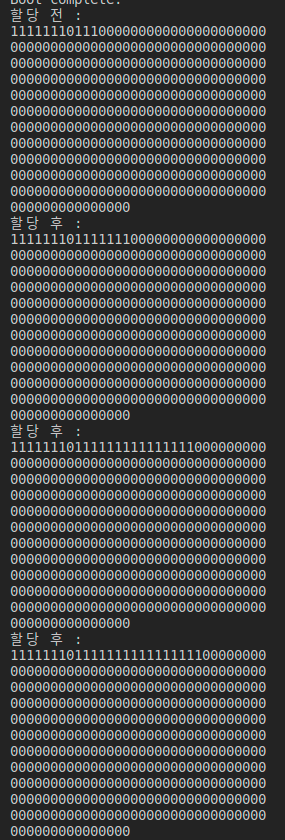


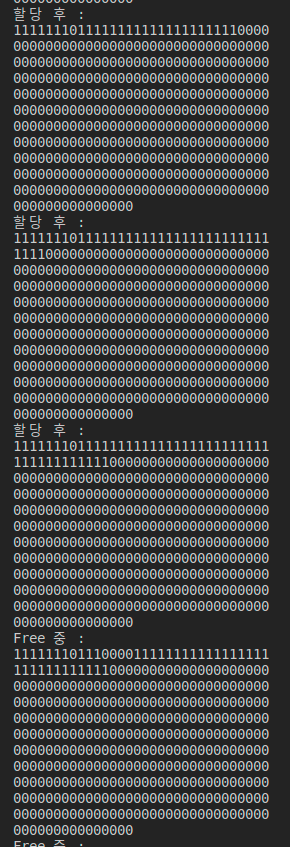
1. **Best Fit의 구현 사진과 실행**

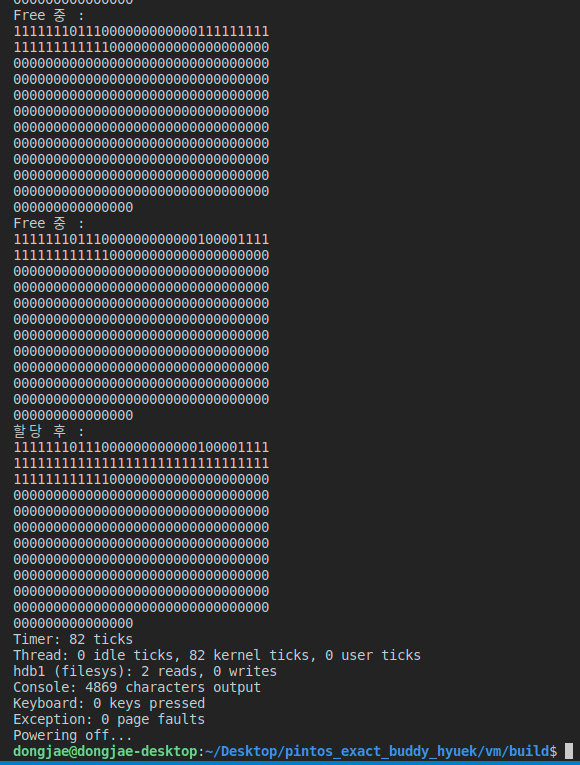
**Best Fit은 First FIt처럼 처음처럼 돌지만 위치를 찾았을 때 바로 return하지말고 temp 장소를 만들어서 가장 작은 위치를 return하는 방식을 사용한다.**



**Best 실행 결과 : f**

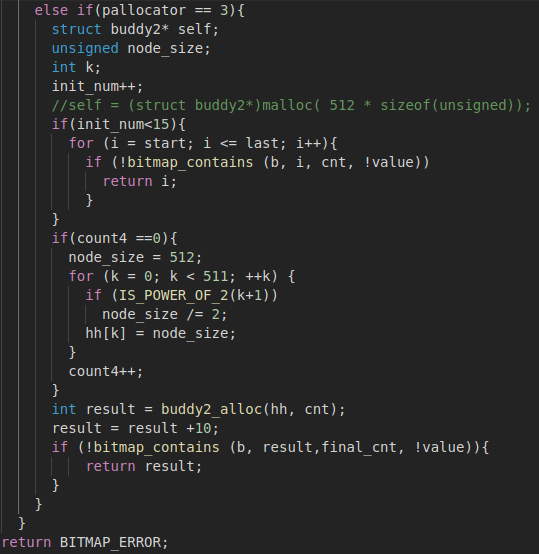






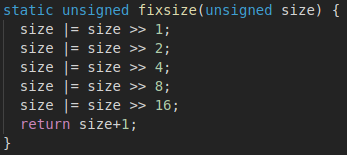
**buddy system**

**pallocator = 3 일 때 buddy algorithm start**

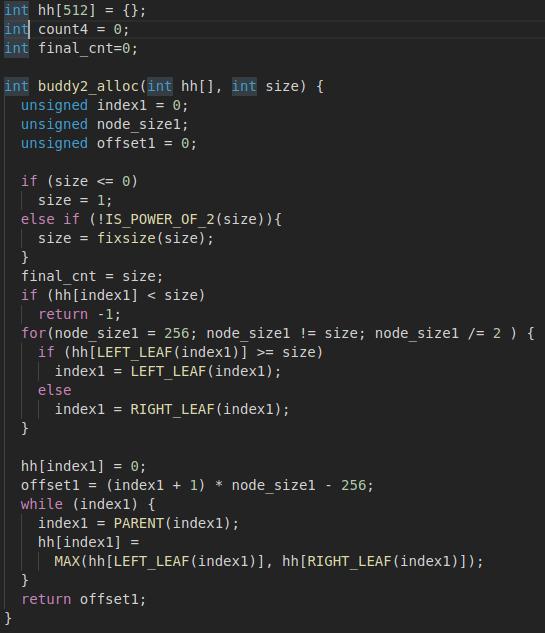


final\_cnt = 할당할 크기

result = 할당할 location

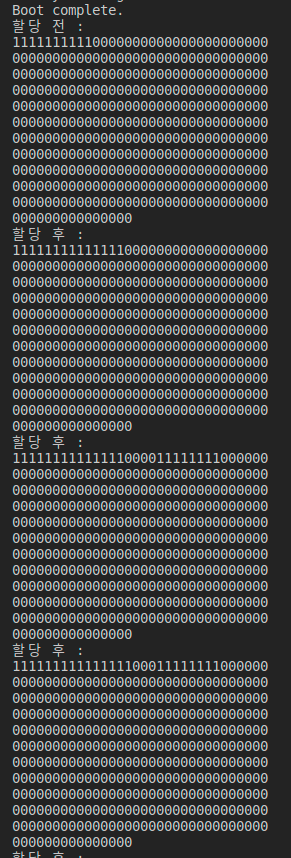


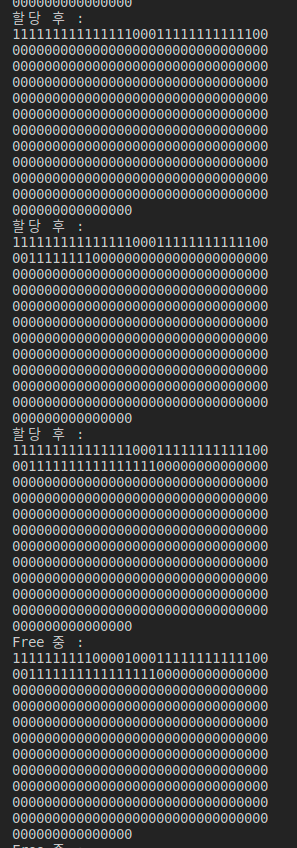
**fixsize()** : malloc size >> **2의 거듭제곱 change**

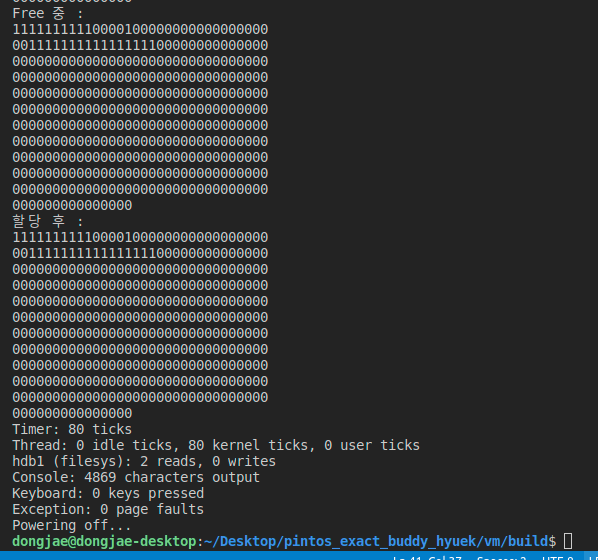


**buddy2\_alloc() : find**

**Buddy System 실행 결과 :**







**요구사항 2: Page Allocation 정책의 성능 평가**

우선 각 정책의 시간을 비교해보면,

First fit->67tick

Next fit -> 91tick

Best fit->82tick

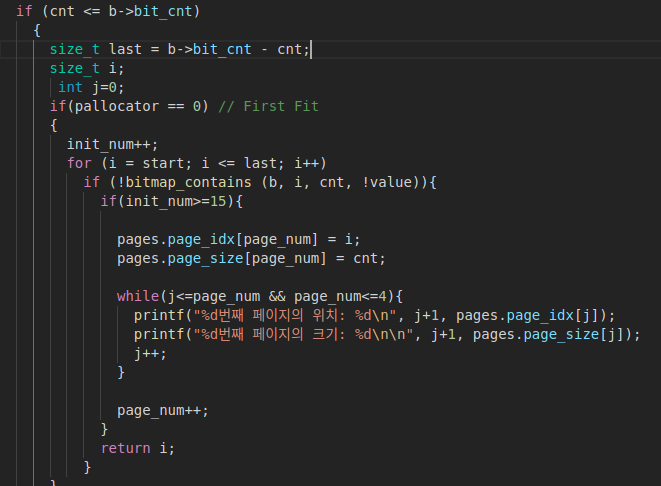
buddy System->80tick

의 성능을 보였다.

하지만 페이지 free후 다시 할당을 할 때,

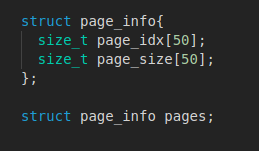
**first fit**에서 가장 메모리 낭비를 덜 하였다.

**요구사항 3: Frame Table 구현**

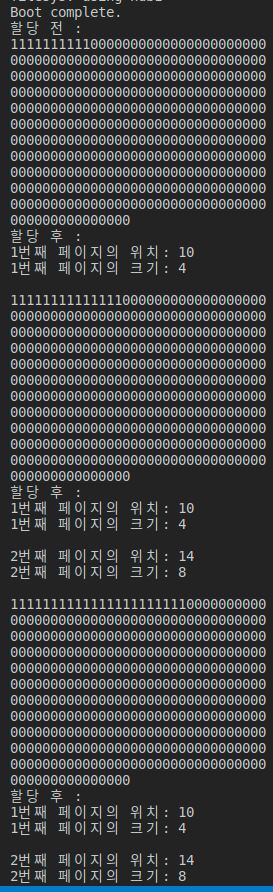


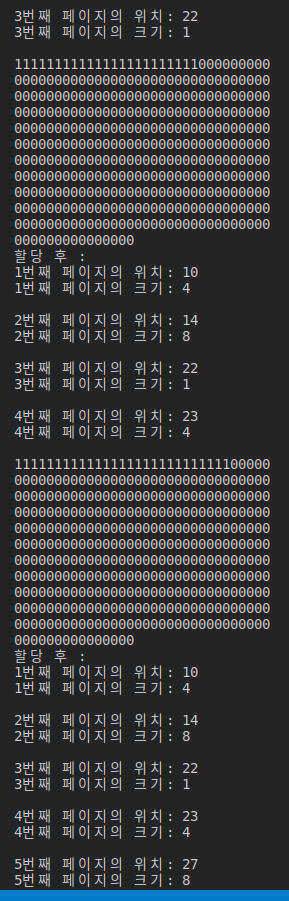
위의 코드와 같이, 요구사항 3: Page Table 구현을 만족하기 위해

bitmap.c 파일에서



와 같은 방식으로 페이지들의 위치값과 크기를 가지고 있는 구조체를 만들어서 Page Table을 만들었다. 구조체가 올바르게 페이지의 element를 가지고 있는 지 확인하기 위하여, firtst fit을 실행할 때 페이지를 5번 할당하는 동안 페이지 정보 또한 Page Table내에 올바르게 저장하는 지 확인해 보았고,





다음과 같이 올바르게 할당이 되는 것을 확인 할 수 있다.