OS Team Project

최종 보고서

7팀

20192613 이동현(팀장)

20192634 정민교

20192612 이건우

20180759 박천명

목차

*코드는 **github**에 있으므로 따로 표기 안함*

github 주소 : https://github.com/SSUminiOS/A7teamOS

1.개요

1.1 프로젝트 개요 및 목표

MiniOS를 사용해 팀별로 자유롭게 OS관련 개념이 들어간 주제를 선정하여 그에 맞는 시스템을 구현하고 분석한다.

1.2 주제 선정

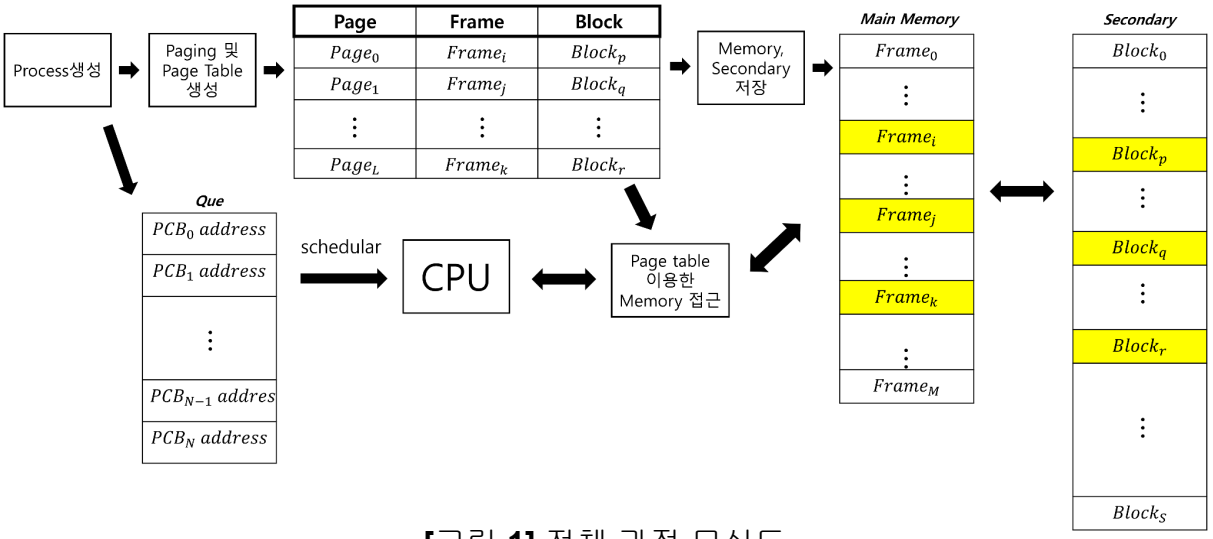
우리팀은 이번 운영체제 프로젝트의 주제로 프로세스 생성부터 메모리 및 보조 저장 장치에 할당하고, CPU가 이를 직접 Running 시키고, 이후 메모리에서 해제 시키는 전체적인 일련의 과정을 다뤘다. 물론, 운영체제라는 이름 하에 선택할 수 있는 좀더 고도화되거나 난이도가 높은 주제들도 많다고 생각한다. 하지만 우리팀은 이번 운영체제 과목에서 얻어가야할 핵심적인 부분이 어느 한부분에 치우친 난해한 개념 보다는 '운영체제' 라는 것의 의미와 운용방식을 넓은 시야를 가지고 큰틀에서 이해하는 것이 맞다고 생각했다. 따라서 이번 프로젝트에서는 앞서 말한대로 가장 핵심적인 개념들을 사용해서 실제 운영되는 '순서' 및 '흐름'에 집중하여 전체적인 과정을 다루기로 결정했다. 이를 통해 수업 시간에 배운 운영체제의 핵심 개념들을 좀 더 포괄적이고 종합적으로 이해하고 실습해볼 수 있는 기회라고 생각했다.

1.3 수행 일정

팀원	박천명	이건우	이동현	정민교
역할분담	- 프로세스 생성 - 물리 메모리 주소 할당 - Paging 구현	- 프로세스 생성 - 프로세스별 가상 메모리 주소 할당 - PCB구현	- 프로세스 생성 - page table 생성 - round robin scheduler 구현	- 프로세스 생성 - 메인 메모리 생성 - File system 구현
모임일자	발표내용			
2024-04-18	- 공부 및 과제 내용 복습 - Minios 관련 주제 토의 - 회의 일정 조율			
2024-04-23	- 7주차 과제 내용 복습 - Minios 관련 주제 구체화 - 리눅스 관련 명령어 구체적으로 공부			
2024-04-30	- 프로젝트 주제 선정 Pixel 기반 경계 검출 프로그램			
2024-05-09	- 프로젝트 주제에 대한 재토의 - 프로젝트 주제 최종 후보 선정 및 구체화			
2024-05-13/14	- 프로젝트 주제 최종 선정 및 구체화 Minios 에서 Paging 기법 구현			
2024-05-18/21	- Paging 구현 Main Memory, Virtual Memory, Page Table 구현			
2024-05-25/28	- Process 당 PCB 구현 - Round Robin Scheduler 구현			
2024-06-01	- Header file(.h)과 Source file(.c) 분리 - Process 개수 증가 및 메모리 크기 확정 - 출력(결과)값 선정			
2024-06-08/11	- File System (Secondary Storage) 구현 - Scheduler 에서 Arrival Time 설정			

2. 수행과정

본격적인 프로젝트에 들어가기 앞서서, 우리는 주제로 선정한 프로세스 생성부터 시작해서 메모리 및 보조저장장치에 저장시키고 scheduler을 통해 실행 시킨 후, 메모리 해제를 시키는 일련의 과정을 구현하고자 하였다. 우선적으로 실제 구현해야할 전체 과정을 순서에 맞게 [그림 1]과 같이 모식도로 표현해보았다.



[그림 1] 전체 과정 모식도

모식도를 토대로 MiniOS Kernel안에 해당 역할을 담는 함수들을 구현하여 프로젝트를 진행하기로 하였다. 이때, 기존 MiniOS구조에 착안하여 구현할 수 많은 함수들의 정의와 선언부를 모아놓을 헤더파일(basic_include.h)과 C파일(basic_include.c)을 운용하였다. 그리고 헤더파일 안에 기본적으로 정해놓고 시작해야하는 시스템 파라미터를 정의해놓았고 해당 값은 [표 1]과 같다.

MiniOS 시스템을 전체적으로 구현함에 있어서 크게 Paging과 Scheduler 2파트로 분류하였고 그 안에서 세부적으로 모식도 순서에 맞게 과정을 진행해 나아갔다. 최종 시스템은 이전 방식과 동일하게 텍스트 명령어를 받아서 이에 해당하는 기능을 실행시키도록 구현하였다.

종류	값
Secondary Storage Size	8192 Byte
Main Memory Size	4096 Byte
Virtual Memory Size	4096 Byte
Page Size(=Frame, Block Size)	128 Byte
Queue Size	10 개
Time Quantum	1 초

[표 1] 시스템 파라미터 정의 값

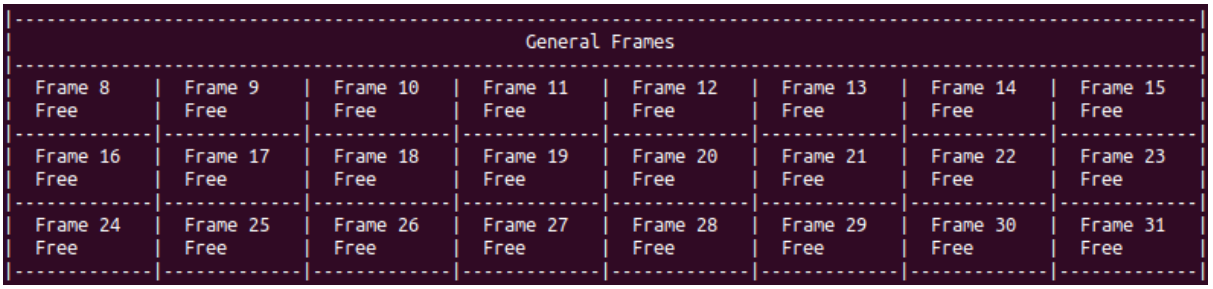
2.1. Paging

2.1.1 Virtual Memory, Physical Memory, Secondary Storage

CPU가 프로그램을 실행하게 되면 프로그램, Process에 필요한 메모리 주소 공간이 Virtual Memory 형태로 생성이 된다. 각 프로세스마다 독립적인 Virtual Memory가 생성이 되는 데 이는 Process간의 Memory 충돌을 방지한다.

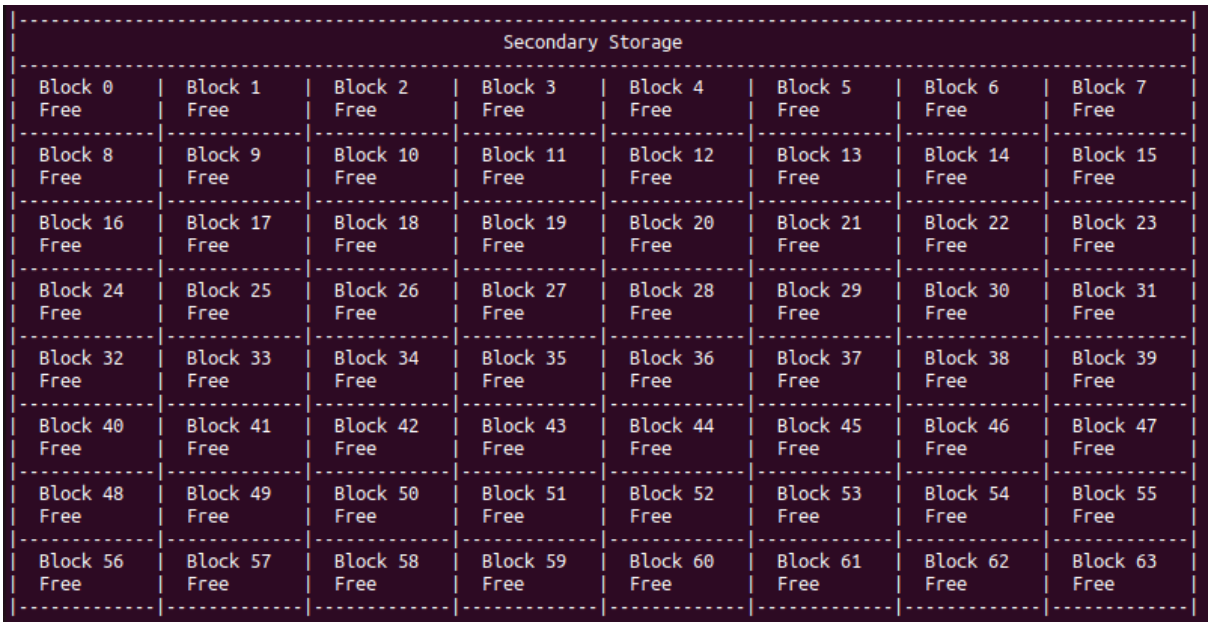
Physical Memory는 Main Memory이며 현재 작업중인 데이터를 저장하고 빠르게 접근할 수 있는 휘발성 메모리이다. Secondary Storage는 보조 기억 장치라고도 하며 데이터를 오래 저장하기 위한 비휘발성 메모리이다.

[표 1]에서 볼 수 있듯이 본 MiniOS에서는 Physical Memory의 크기를 4096 Byte, Virtual Memory을 크기를 4096 Byte. Secondary Storage의 크기를 8192 Byte로 설정하여 진행하였다. MiniOS를 실행하게 되면 모두 비어있는 상태에서 시작하여 Process에 의해 생성되는 Page를 Main Memory의 Frame으로, Secondary Storage의 Block으로 할당하도록 구현하였다.



General Frames							
Frame 8 Free	Frame 9 Free	Frame 10 Free	Frame 11 Free	Frame 12 Free	Frame 13 Free	Frame 14 Free	Frame 15 Free
Frame 16 Free	Frame 17 Free	Frame 18 Free	Frame 19 Free	Frame 20 Free	Frame 21 Free	Frame 22 Free	Frame 23 Free
Frame 24 Free	Frame 25 Free	Frame 26 Free	Frame 27 Free	Frame 28 Free	Frame 29 Free	Frame 30 Free	Frame 31 Free

[그림 2] Main Memory



Secondary Storage							
Block 0 Free	Block 1 Free	Block 2 Free	Block 3 Free	Block 4 Free	Block 5 Free	Block 6 Free	Block 7 Free
Block 8 Free	Block 9 Free	Block 10 Free	Block 11 Free	Block 12 Free	Block 13 Free	Block 14 Free	Block 15 Free
Block 16 Free	Block 17 Free	Block 18 Free	Block 19 Free	Block 20 Free	Block 21 Free	Block 22 Free	Block 23 Free
Block 24 Free	Block 25 Free	Block 26 Free	Block 27 Free	Block 28 Free	Block 29 Free	Block 30 Free	Block 31 Free
Block 32 Free	Block 33 Free	Block 34 Free	Block 35 Free	Block 36 Free	Block 37 Free	Block 38 Free	Block 39 Free
Block 40 Free	Block 41 Free	Block 42 Free	Block 43 Free	Block 44 Free	Block 45 Free	Block 46 Free	Block 47 Free
Block 48 Free	Block 49 Free	Block 50 Free	Block 51 Free	Block 52 Free	Block 53 Free	Block 54 Free	Block 55 Free
Block 56 Free	Block 57 Free	Block 58 Free	Block 59 Free	Block 60 Free	Block 61 Free	Block 62 Free	Block 63 Free

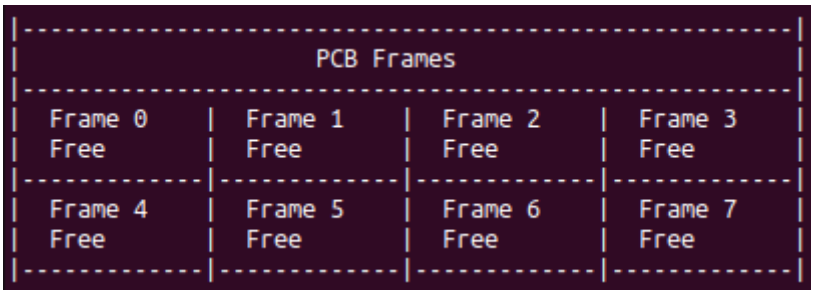
[그림 3] Secondary Storage

[그림 2]와 [그림 3]에서 볼 수 있듯이 MiniOS가 처음 켜지게 되면 모두 비어 있는 상태 (Free) 상태로 설정이 되어있다.

2.1.2 Process 및 PCB 생성

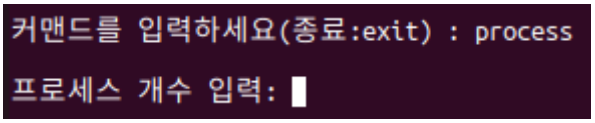
Process는 현재 실행 중인 프로그램을 의미하며 운영체제에서 다양한 작업을 수행하는 기본적인 단위이다. 사용자가 프로그램을 실행하거나 시스템이 특정 작업을 실행하게 되면 Process가 생성된다. 그렇기에 본 프로젝트의 실행을 시작할 때 가장 먼저 Process가 생성이 되도록 MiniOS를 구현했으며 프로세스의 개수를 입력 받아서 실행이 된다. Process가 생성이 될 때 각각의 Process마다 Process ID인 PID가 할당이 되며 PCB(Process Control Block)도 같이 생성이 된다. 본 MiniOS에서는 PCB 하나의 크기를 Frame(Page)의 크기와 동일한 128 Byte로 설정을 하고 진행을

하였다. PCB는 Main Memory의 OS, Kernel 부분에 저장되는 데 우선 최대 8개의 PCB가 들어갈 수 있도록 8개의 Frame을 할당했으며 이를 [그림 4]에서 확인할 수 있다. 아직 어떠한 Process도 생성이 되지 않았기 때문에 모두 Free 상태이다.



[그림 **4] PCB**를 저장하는 **Main Memory**의 **8**개의 **Frame**

다음으로, [그림 5]와 같이 'process'라는 Command를 입력하고 프로세스의 개수를 입력하게 되면 구현한 MiniOS가 실행된다. 본 보고서에서는 프로세스의 개수를 6개로 설정하여 서술한다.



[그림 **5] Process** 개수 입력 받기

2.1.3 Page 할당

Process의 개수를 입력 받게 되면, 각 Process는 Page를 생성한다. 본 MiniOs에서는 각 Process가 1개에서 4개의 Page를 생성도록 구현하였다. 하나의 Page의 크기는 [표 1]에 나와있듯이 128 Byte이다.

각각의 Process는 자신만의 가상 메모리(Virtual Memory)를 갖고 있기 때문에, 각 Process의 가상 메모리에 Page 0부터 최대 Page 3까지 할당이 되도록 설계를 하였다.

2.1.4 Page Table

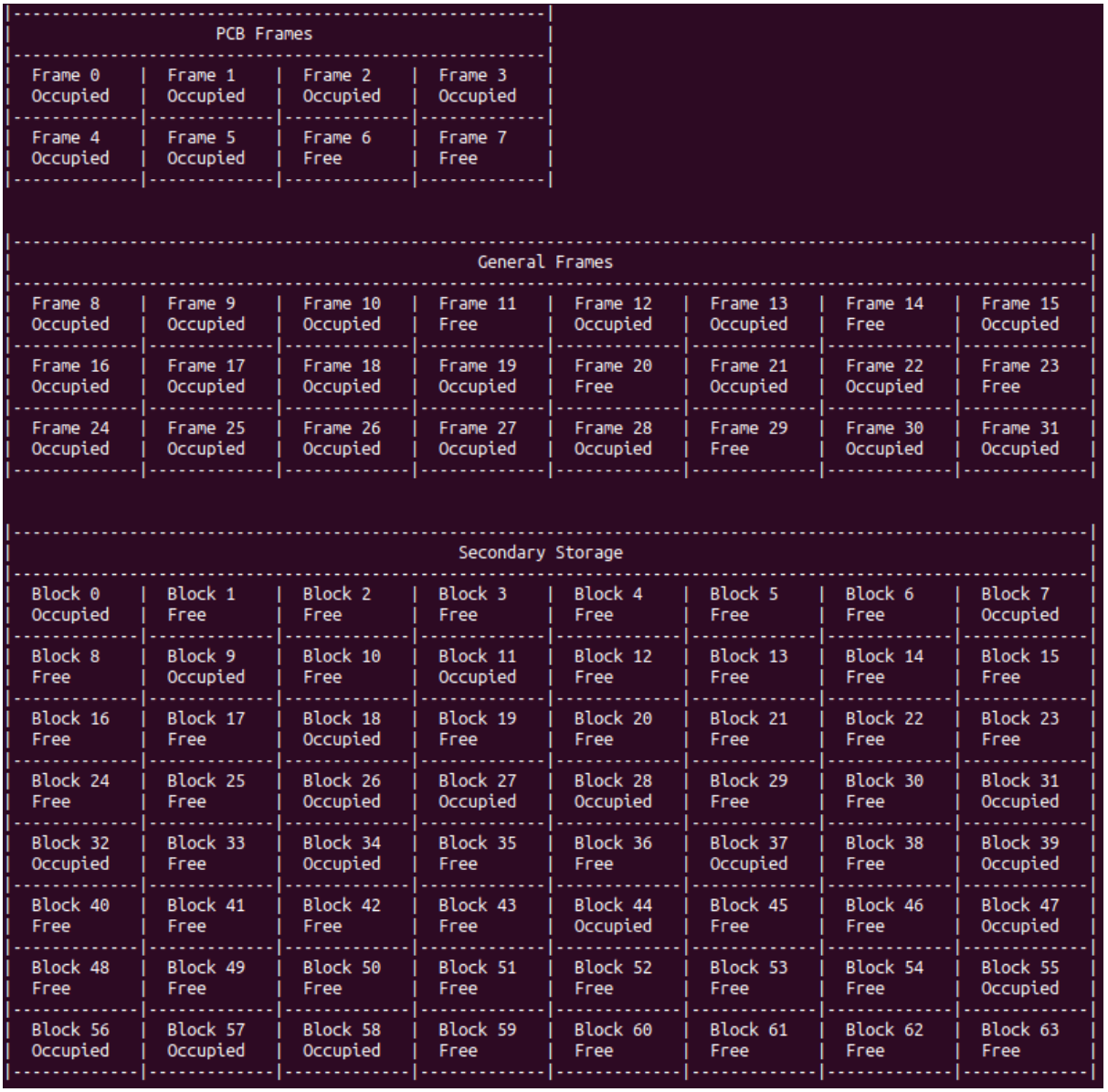
각 Process마다 Page를 할당한 후에는 Physical Memory (Main Memory)에 이를 할당해주어야 하며 Process마다의 Page Table도 생성된다. Page Table은 Page ID, Frame Number, Block Number, Valid Bit, In_Secondary로 구성이 되게끔 만들었다.

Page ID는 Process마다의 Virtual Memory에 저장된 Page마다의 번호이다. Frame Number는 Main Memory의 어떤 Frame에 할당이 되었는 지, Block Number는 Secondary Storage의 어떤 Block에 할당이 되었는 지를 의미한다. Valid Bit은 원하는 Page, Frame이 Main Memory에 있으면 1, 없으면 0이 되고 마지막으로 In_Secondary는 Secondary Storage에 있는 지 여부를 나타내며 있으면 1이 된다.

```
Page Table for Process 1:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |           27 |           18 |     1 |            1
              1 |           31 |           58 |     1 |            1

Page Table for Process 2:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |           30 |           56 |     1 |            1
              1 |           17 |           11 |     1 |            1
              2 |           26 |           31 |     1 |            1

Page Table for Process 3:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |           16 |           44 |     1 |            1
              1 |           28 |           55 |     1 |            1
              2 |           21 |           26 |     1 |            1
              3 |           10 |           57 |     1 |            1

Page Table for Process 4:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |            8 |           37 |     1 |            1
              1 |           18 |            9 |     1 |            1
              2 |           25 |           28 |     1 |            1
              3 |           15 |           27 |     1 |            1

Page Table for Process 5:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |           24 |           39 |     1 |            1
              1 |           12 |            0 |     1 |            1

Page Table for Process 6:
Logical Page ID | Frame Number | Block Number | Valid | In_Secondary
-------------------------------------------------------------------
              0 |           13 |           47 |     1 |            1
              1 |            9 |            7 |     1 |            1
              2 |           19 |           34 |     1 |            1
              3 |           22 |           32 |     1 |            1
```

[그림 6] Page Table

[그림 6]에서는 각 Process마다 Page Table이 존재하며, 각 Page가 서로 다른 Frame(in Main Memory)에, 서로 다른 Block(in Secondary Storage)에 할당되는 것을 확인할 수 있다. 모든 Page가 할당이 되었으니 Valid Bit는 1이 되며, In_Secondary Bit도 1이 된다.



PCB Frames			
Frame 0 Occupied	Frame 1 Occupied	Frame 2 Occupied	Frame 3 Occupied
Frame 4 Occupied	Frame 5 Occupied	Frame 6 Free	Frame 7 Free

General Frames							
Frame 8 Occupied	Frame 9 Occupied	Frame 10 Occupied	Frame 11 Free	Frame 12 Occupied	Frame 13 Occupied	Frame 14 Free	Frame 15 Occupied
Frame 16 Occupied	Frame 17 Occupied	Frame 18 Occupied	Frame 19 Occupied	Frame 20 Free	Frame 21 Occupied	Frame 22 Occupied	Frame 23 Free
Frame 24 Occupied	Frame 25 Occupied	Frame 26 Occupied	Frame 27 Occupied	Frame 28 Occupied	Frame 29 Free	Frame 30 Occupied	Frame 31 Occupied

Secondary Storage							
Block 0 Occupied	Block 1 Free	Block 2 Free	Block 3 Free	Block 4 Free	Block 5 Free	Block 6 Free	Block 7 Occupied
Block 8 Free	Block 9 Occupied	Block 10 Free	Block 11 Occupied	Block 12 Free	Block 13 Free	Block 14 Free	Block 15 Free
Block 16 Free	Block 17 Free	Block 18 Occupied	Block 19 Free	Block 20 Free	Block 21 Free	Block 22 Free	Block 23 Free
Block 24 Free	Block 25 Free	Block 26 Occupied	Block 27 Occupied	Block 28 Occupied	Block 29 Free	Block 30 Free	Block 31 Occupied
Block 32 Occupied	Block 33 Free	Block 34 Occupied	Block 35 Free	Block 36 Free	Block 37 Occupied	Block 38 Free	Block 39 Occupied
Block 40 Free	Block 41 Free	Block 42 Free	Block 43 Free	Block 44 Occupied	Block 45 Free	Block 46 Free	Block 47 Occupied
Block 48 Free	Block 49 Free	Block 50 Free	Block 51 Free	Block 52 Free	Block 53 Free	Block 54 Free	Block 55 Occupied
Block 56 Occupied	Block 57 Occupied	Block 58 Occupied	Block 59 Free	Block 60 Free	Block 61 Free	Block 62 Free	Block 63 Free

[그림 7] Main Memory(for PCB and Pages)와 Secondary Storage

[그림 7]에서 볼 수 있듯이 6개의 Process가 실행 되었으므로 6개의 PCB가 존재하며 Main Memory의 OS, Kernel 부분(Frame 0~5)까지 저장이 된 것을 확인할 수 있다. 또한 각 Process에서 생성된 Page가 Main Memory의 서로 다른 Frame들과 Secondary Storage의 서로 다른 Block으로 들어가 저장이 된다. 이를 통해 모든

Process의 Page들이 Main Memory와 Secondary Storage에 잘 할당이 된 것을 볼 수 있으며 앞의 [그림 6]의 Page Table의 정보와 일치하기에 Page Table 역시 성공적으로 생성된 것을 알 수 있다.

2.2. Scheduler

2.2.1 Round Robin Scheduler Background

Round Robin Scheduling은 컴퓨터 시스템의 프로세스 스케줄링 기법 중 하나로, 주로 시분할 시스템에서 사용되며 모든 프로세스가 CPU 시간을 공정하게 배분받을 수 있도록 설계되었다.

☐ Time Quantum

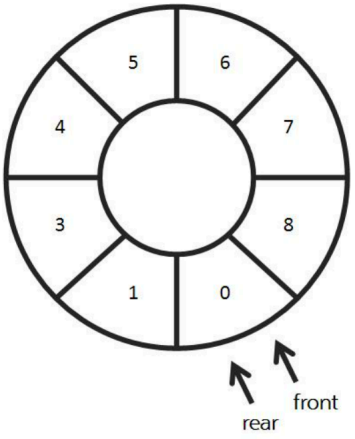
Time Quantum은 각 프로세스가 CPU를 사용할 수 있는 최대 시간을 의미한다. 이 Time Quantum이 지나게 되면, CPU의 Preemptive가 발생하게 되며 다음 프로세스에게 순번을 넘겨주게 된다. 이 단위를 짧게 할 경우 Context Switch가 빈번해져 오버헤드가 커질 가능성이 있다.

☐ Queue

모든 Ready 상태에 있는 프로세스는 FIFO방식으로 Queue에 저장되게 된다. 만약 Time Quantum으로 인해 다시 Ready Queue로 넘어가게 될 경우, 맨 뒤에서 순번을 기다리게 된다. Round Robin Scheduling에서는 Arrival Time까지 고려하여 Time Quantum으로 인해 Process가 끝난 직후, Arrival Time에 의해 도착한 Process를 확인하고 도착하였을 시 동일하게 Ready Queue에 Process를 넣게 된다.

2.2.2 Round Robin Scheduler 구현

우선 첫번 째로 Round Robin Scheduling의 구현을 위해서 우리는 Circular Queue를 사용하였다. Round Robin은 Context Switch가 잦기 때문에 Queue의 크기가 정해져 있을 경우 많은 Context Switch를 감당하지 못한다고 생각하였다.

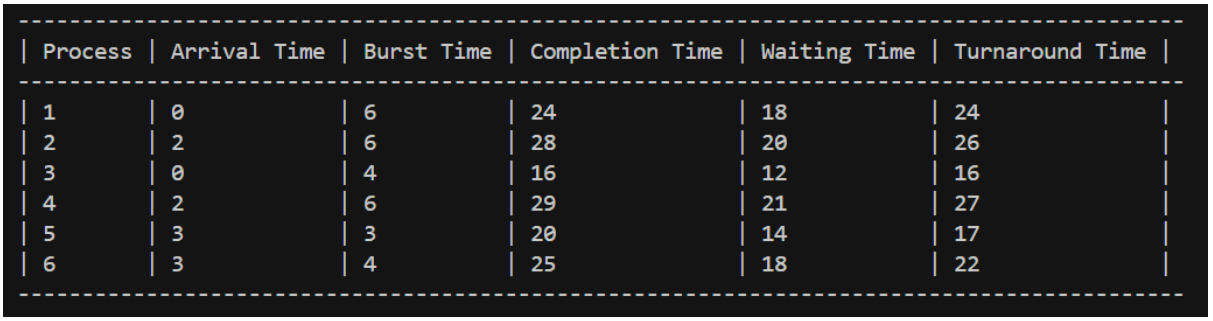


[그림 **8] Circular Queue** 구조

Circular Queue Struct의 구성은 item[Queue_Size], front, rear로 구성되어 있다. 초기화를 진행할 때, front와 rear는 0으로 초기화를 하고 item의 경우 PCB의 pid가 들어가게 되는데, 우리는 Process를 fork()를 진행하지 않은 임의의 Process를 생성하기 때문에 pid가 0부터 생기게 된다. 따라서 0으로 초기화 하는 것이 아닌 -1로 초기화를 진행한다.

앞서 Process Count를 통해서 생성하고자 하는 Process의 개수를 설정하였다. Process Count를 remaining_processes 변수에 넣어 남은 Process가 0보다 클 경우 Loop를 반복한다. 반복문이 실행되면 Current Time과 Arrival Time이 0으로 같은 Process부터 Ready Queue에 넣게 된다. Current Time은 각 Loop가 끝날 때 마다 1씩 증가하게 된다. 이제 Ready Queue에 존재하는 순서대로 Dequeue를 하게 된다. Dequeue를 하여 해당 Process의 pid를 가져오고 그를 통해서 PCB 정보를 가져오게 된다. 만약 Queue에 아무것도 존재하지 않을 경우 Current Time을 증가 시키고 다음 번으로 넘어간다. 임시의 Process변수를 선언하여 Queue에서 나온 pid번호를 이용하여 해당 번호의 Process를 넣어준다. 이때 Process의 Remain Burst Time이 0보다 큰 경우에만 CPU의 Running State로 들어가게 된다. Process가 Running State에 있는 만큼 즉 Time Quantum만큼 Remain Burst Time에서 빼준다. 그러던 중 Process가 종료될 경우 Remain Processes를 1 감소 시킨 뒤 메모리를 풀어준다.

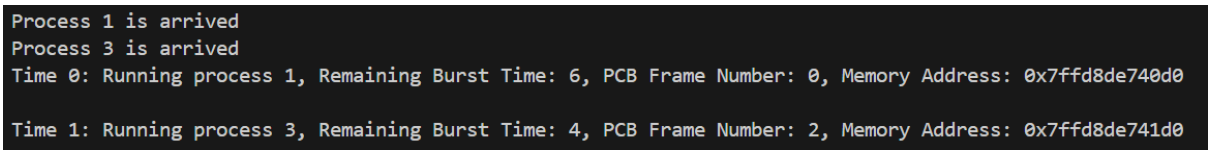
2.2.3 Scheduler 실행결과



Process	Arrival Time	Burst Time	Completion Time	Waiting Time	Turnaround Time
1	0	6	24	18	24
2	2	6	28	20	26
3	0	4	16	12	16
4	2	6	29	21	27
5	3	3	20	14	17
6	3	4	25	18	22

[그림 9] Round Robin Process Info

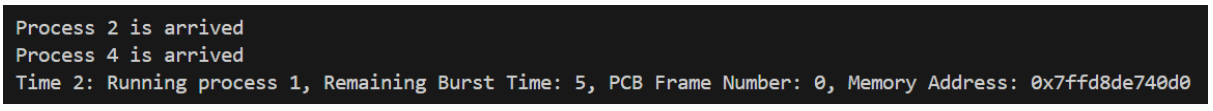
[그림 9]는 생성된 프로세스들의 정보를 보여주고 있다. 이러한 표를 보고 실행결과를 확인해 보겠다. 먼저 Arrival Time이 0일때 Process 1과 Process 3이 도착하였다. Time =1일 때는 도착하는 Process가 없기 때문에 Time 0에서 Time 1까지는 Process 1과 3을 반복하고 Q에 다시 들어갈 것이다.



```
Process 1 is arrived
Process 3 is arrived
Time 0: Running process 1, Remaining Burst Time: 6, PCB Frame Number: 0, Memory Address: 0x7ffd8de740d0

Time 1: Running process 3, Remaining Burst Time: 4, PCB Frame Number: 2, Memory Address: 0x7ffd8de741d0
```

[그림 10] Progress 1

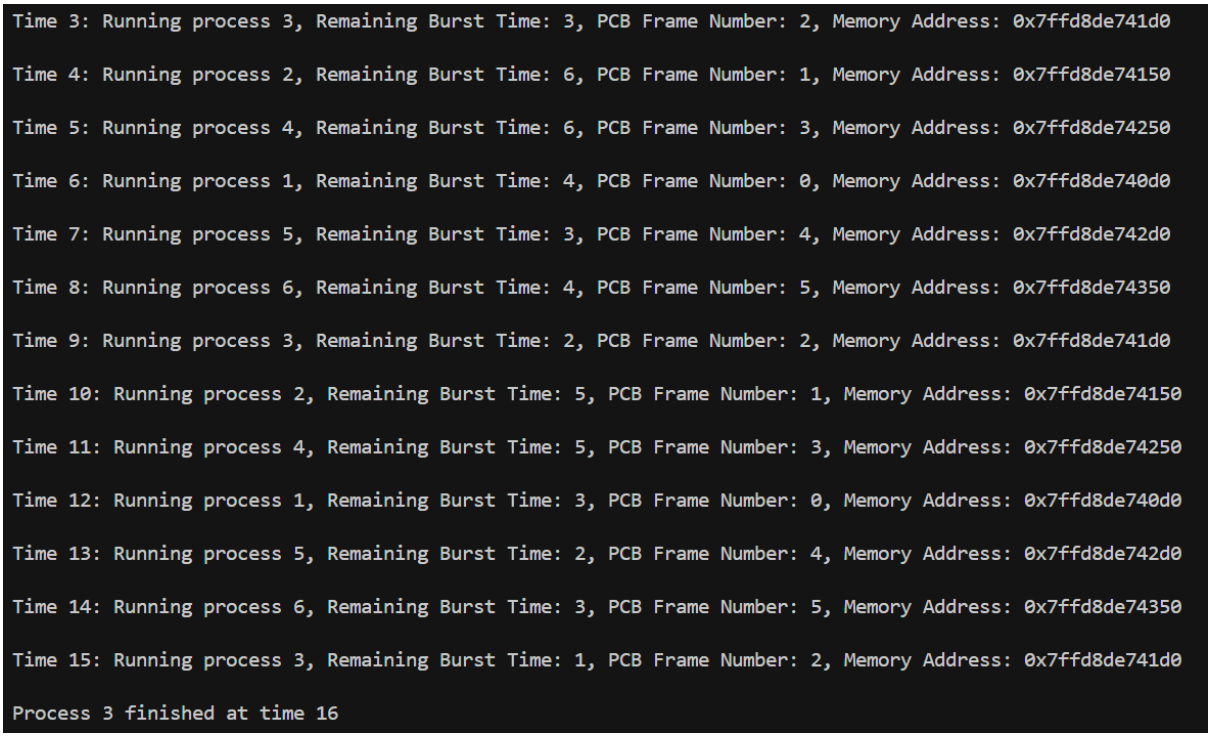
[그림 10]은 Process 1과 Process 3이 도착 후 Running State를 거친 것을 확인할 수 있다. 현재 Queue에는 {1 , 3} 순서대로 들어가 있다. Time 2에서 Process 2와 Process 4가 순서대로 들어오게 되는데, 그러면 현재 Queue에는 {1 ,3, 2, 4}가 들어 있다.



```
Process 2 is arrived
Process 4 is arrived
Time 2: Running process 1, Remaining Burst Time: 5, PCB Frame Number: 0, Memory Address: 0x7ffd8de740d0
```

[그림 11] Progress 2

[그림 11]은 Process 2와 Process 4가 도착한 뒤 Time 2에 Queue의 가장 앞에 있는 1이 실행된 것을 볼 수 있다. 이제 Time 3에서는 남은 Process들이 도착하기 때문에 방금 실행된 Process 1이 맨 뒤로 가고 그 뒤로 Process 5와 6이 들어오므로 Ready Queue에는 {3,2,4,1,5,6}의 순서로 저장이 되어있다.



[그림 12] Progress 3

[그림 12]는 앞서 Ready Queue의 순서인 {3,2,4,1,5,6}의 순서대로 Process가 돌아가는 것을 확인할 수 있다. 이때 Time 15를 기점으로 Process 3이 종료되었기 때문에 Queue에 들어가지 않게 되고 {2,4,1,5,6}의 순서로 진행이 된다.

```
Time 16: Running process 2, Remaining Burst Time: 4, PCB Frame Number: 1, Memory Address: 0x7ffd8de74150
Time 17: Running process 4, Remaining Burst Time: 4, PCB Frame Number: 3, Memory Address: 0x7ffd8de74250
Time 18: Running process 1, Remaining Burst Time: 2, PCB Frame Number: 0, Memory Address: 0x7ffd8de740d0
Time 19: Running process 5, Remaining Burst Time: 1, PCB Frame Number: 4, Memory Address: 0x7ffd8de742d0
Process 5 finished at time 20
Time 20: Running process 6, Remaining Burst Time: 2, PCB Frame Number: 5, Memory Address: 0x7ffd8de74350
Time 21: Running process 2, Remaining Burst Time: 3, PCB Frame Number: 1, Memory Address: 0x7ffd8de74150
Time 22: Running process 4, Remaining Burst Time: 3, PCB Frame Number: 3, Memory Address: 0x7ffd8de74250
Time 23: Running process 1, Remaining Burst Time: 1, PCB Frame Number: 0, Memory Address: 0x7ffd8de740d0
Process 1 finished at time 24
Time 24: Running process 6, Remaining Burst Time: 1, PCB Frame Number: 5, Memory Address: 0x7ffd8de74350
Process 6 finished at time 25
Time 25: Running process 2, Remaining Burst Time: 2, PCB Frame Number: 1, Memory Address: 0x7ffd8de74150
Time 26: Running process 4, Remaining Burst Time: 2, PCB Frame Number: 3, Memory Address: 0x7ffd8de74250
Time 27: Running process 2, Remaining Burst Time: 1, PCB Frame Number: 1, Memory Address: 0x7ffd8de74150
Process 2 finished at time 28
Time 28: Running process 4, Remaining Burst Time: 1, PCB Frame Number: 3, Memory Address: 0x7ffd8de74250
Process 4 finished at time 29
```

[그림 13] Progress 4

Time 16부터는 앞선 언급 처럼 {2,4,1,5,6}의 순서로 진행되다가 Process 5가 끝난 시점에서 Process 5가 종료되어 Queue에서 제외되게 된다. 다음 순서로 {6,2,4,1}의 순으로 진행되게 되고, Process 1이 종료되어 {6,2,4} 순서가 되고, Process 6이 종료되어 {2,4}가 남아 서로 반복하면서 진행되다 두개 모두 종료되는 모습을 확인할 수 있다. 이로써, Round Robin 알고리즘이 정상적으로 구현된 것을 확인할 수 있다.

다시 [그림 9]를 확인하면 각각의 Process의 Completion Time이 일치하는 것을 볼 수 있다.

3. 결론

최종적으로 우리팀은 앞서 프로젝트 주제로 삼았던 전체적인 과정에 대해 단계단계씩 진행해서 구현한 기능들을 하나의 시스템으로 묶을 수 있었다. 따라서 우리팀은 프로세스를 생성하는 과정부터 Paging 작업을 통해 Page Table을 생성하고 이를 토대로 메모리 및 보조저장장치에 할당하는 작업을 거쳐 Scheduler을 통한 CPU Scheduling 까지 한 시스템 내에서 이뤄지도록 구현할 수 있었다. 이에 대한 전체적인 결과는 [그림 14] 와 같다.

[그림 **14]** 전체 결과

```
Process 1: Allocated PCB to PCB Frame 0 at Memory Address 0x7ffe52b75c90

Process 2: Allocated PCB to PCB Frame 1 at Memory Address 0x7ffe52b75d10

Process 3: Allocated PCB to PCB Frame 2 at Memory Address 0x7ffe52b75d90

Process 4: Allocated PCB to PCB Frame 3 at Memory Address 0x7ffe52b75e10

Process 5: Allocated PCB to PCB Frame 4 at Memory Address 0x7ffe52b75e90

Process 6: Allocated PCB to PCB Frame 5 at Memory Address 0x7ffe52b75f10

Available frames: 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Process 1: Allocated Page 0 to Frame 16 at Memory Address 0x7ffe52b76490
Process 1(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 1: Allocated Page 0 to Secondary Storage Block 4

Available frames: 8 9 10 11 12 13 14 15 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Process 1: Allocated Page 1 to Frame 23 at Memory Address 0x7ffe52b76810
Process 1(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 1: Allocated Page 1 to Secondary Storage Block 53

Available frames: 8 9 10 11 12 13 14 15 17 18 19 20 21 22 24 25 26 27 28 29 30 31

Process 1: Allocated Page 2 to Frame 20 at Memory Address 0x7ffe52b76690
Process 1(Page ID 2): Virtual Address [0x0100 - 0x017f]
Process 1: Allocated Page 2 to Secondary Storage Block 12

Available frames: 8 9 10 11 12 13 14 15 17 18 19 21 22 24 25 26 27 28 29 30 31

Process 2: Allocated Page 0 to Frame 27 at Memory Address 0x7ffe52b76a10
Process 2(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 2: Allocated Page 0 to Secondary Storage Block 22

Available frames: 8 9 10 11 12 13 14 15 17 18 19 21 22 24 25 26 28 29 30 31

Process 2: Allocated Page 1 to Frame 14 at Memory Address 0x7ffe52b76390
Process 2(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 2: Allocated Page 1 to Secondary Storage Block 52

Available frames: 8 9 10 11 12 13 15 17 18 19 21 22 24 25 26 28 29 30 31

Process 2: Allocated Page 2 to Frame 12 at Memory Address 0x7ffe52b76290
Process 2(Page ID 2): Virtual Address [0x0100 - 0x017f]
Process 2: Allocated Page 2 to Secondary Storage Block 40

Available frames: 8 9 10 11 13 15 17 18 19 21 22 24 25 26 28 29 30 31

Process 2: Allocated Page 3 to Frame 9 at Memory Address 0x7ffe52b76110
Process 2(Page ID 3): Virtual Address [0x0180 - 0x01ff]
Process 2: Allocated Page 3 to Secondary Storage Block 56

Available frames: 8 10 11 13 15 17 18 19 21 22 24 25 26 28 29 30 31

Process 3: Allocated Page 0 to Frame 10 at Memory Address 0x7ffe52b76190
Process 3(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 3: Allocated Page 0 to Secondary Storage Block 3

Available frames: 8 11 13 15 17 18 19 21 22 24 25 26 28 29 30 31

Process 3: Allocated Page 1 to Frame 24 at Memory Address 0x7ffe52b76890
Process 3(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 3: Allocated Page 1 to Secondary Storage Block 2

Available frames: 8 11 13 15 17 18 19 21 22 25 26 28 29 30 31

Process 3: Allocated Page 2 to Frame 15 at Memory Address 0x7ffe52b76410
Process 3(Page ID 2): Virtual Address [0x0100 - 0x017f]
Process 3: Allocated Page 2 to Secondary Storage Block 28

Available frames: 8 11 13 17 18 19 21 22 25 26 28 29 30 31

Process 3: Allocated Page 3 to Frame 18 at Memory Address 0x7ffe52b76590
Process 3(Page ID 3): Virtual Address [0x0180 - 0x01ff]
Process 3: Allocated Page 3 to Secondary Storage Block 31

Available frames: 8 11 13 17 19 21 22 25 26 28 29 30 31

Process 4: Allocated Page 0 to Frame 31 at Memory Address 0x7ffe52b76c10
Process 4(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 4: Allocated Page 0 to Secondary Storage Block 15

Available frames: 8 11 13 17 19 21 22 25 26 28 29 30

Process 4: Allocated Page 1 to Frame 26 at Memory Address 0x7ffe52b76990
Process 4(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 4: Allocated Page 1 to Secondary Storage Block 50

Available frames: 8 11 13 17 19 21 22 25 28 29 30
```

```
Process 4: Allocated Page 2 to Frame 21 at Memory Address 0x7ffe52b76710
Process 4(Page ID 2): Virtual Address [0x0100 - 0x017f]
Process 4: Allocated Page 2 to Secondary Storage Block 0

Available frames: 8 11 13 17 19 22 25 28 29 30

Process 4: Allocated Page 3 to Frame 29 at Memory Address 0x7ffe52b76b10
Process 4(Page ID 3): Virtual Address [0x0180 - 0x01ff]
Process 4: Allocated Page 3 to Secondary Storage Block 41

Available frames: 8 11 13 17 19 22 25 28 30

Process 5: Allocated Page 0 to Frame 13 at Memory Address 0x7ffe52b76310
Process 5(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 5: Allocated Page 0 to Secondary Storage Block 1

Available frames: 8 11 17 19 22 25 28 30

Process 5: Allocated Page 1 to Frame 8 at Memory Address 0x7ffe52b76090
Process 5(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 5: Allocated Page 1 to Secondary Storage Block 39

Available frames: 11 17 19 22 25 28 30

Process 6: Allocated Page 0 to Frame 22 at Memory Address 0x7ffe52b76790
Process 6(Page ID 0): Virtual Address [0x0000 - 0x007f]
Process 6: Allocated Page 0 to Secondary Storage Block 26

Available frames: 11 17 19 25 28 30

Process 6: Allocated Page 1 to Frame 17 at Memory Address 0x7ffe52b76510
Process 6(Page ID 1): Virtual Address [0x0080 - 0x00ff]
Process 6: Allocated Page 1 to Secondary Storage Block 35

Available frames: 11 19 25 28 30

Process 6: Allocated Page 2 to Frame 19 at Memory Address 0x7ffe52b76610
Process 6(Page ID 2): Virtual Address [0x0100 - 0x017f]
Process 6: Allocated Page 2 to Secondary Storage Block 18

Available frames: 11 25 28 30

Process 6: Allocated Page 3 to Frame 11 at Memory Address 0x7ffe52b76210
Process 6(Page ID 3): Virtual Address [0x0180 - 0x01ff]
Process 6: Allocated Page 3 to Secondary Storage Block 37

|-------------------------------------------------|
|                   PCB Frames                    |
|-------------------------------------------------|
| Frame 0    | Frame 1    | Frame 2    | Frame 3   |
| Occupied   | Occupied   | Occupied   | Occupied  |
|------------|------------|------------|-----------|
| Frame 4    | Frame 5    | Frame 6    | Frame 7   |
| Occupied   | Occupied   | Free       | Free      |
|------------|------------|------------|-----------|


|-----------------------------------------------------------------------------------------------|
|                                        General Frames                                         |
|-----------------------------------------------------------------------------------------------|
| Frame 8    | Frame 9    | Frame 10   | Frame 11   | Frame 12   | Frame 13   | Frame 14   | Frame 15   |
| Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   |
|------------|------------|------------|------------|------------|------------|------------|------------|
| Frame 16   | Frame 17   | Frame 18   | Frame 19   | Frame 20   | Frame 21   | Frame 22   | Frame 23   |
| Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   | Occupied   |
|------------|------------|------------|------------|------------|------------|------------|------------|
| Frame 24   | Frame 25   | Frame 26   | Frame 27   | Frame 28   | Frame 29   | Frame 30   | Frame 31   |
| Occupied   | Free       | Occupied   | Occupied   | Free       | Occupied   | Free       | Occupied   |
|------------|------------|------------|------------|------------|------------|------------|------------|
```

Secondary Storage

Block 0 Occupied	Block 1 Occupied	Block 2 Occupied	Block 3 Occupied	Block 4 Occupied	Block 5 Free	Block 6 Free	Block 7 Free
Block 8 Free	Block 9 Free	Block 10 Free	Block 11 Free	Block 12 Occupied	Block 13 Free	Block 14 Free	Block 15 Occupied
Block 16 Free	Block 17 Free	Block 18 Occupied	Block 19 Free	Block 20 Free	Block 21 Free	Block 22 Occupied	Block 23 Free
Block 24 Free	Block 25 Free	Block 26 Occupied	Block 27 Free	Block 28 Occupied	Block 29 Free	Block 30 Free	Block 31 Occupied
Block 32 Free	Block 33 Free	Block 34 Free	Block 35 Occupied	Block 36 Free	Block 37 Occupied	Block 38 Free	Block 39 Occupied
Block 40 Occupied	Block 41 Occupied	Block 42 Free	Block 43 Free	Block 44 Free	Block 45 Free	Block 46 Free	Block 47 Free
Block 48 Free	Block 49 Free	Block 50 Occupied	Block 51 Free	Block 52 Occupied	Block 53 Occupied	Block 54 Free	Block 55 Free
Block 56 Occupied	Block 57 Free	Block 58 Free	Block 59 Free	Block 60 Free	Block 61 Free	Block 62 Free	Block 63 Free

Page Table for Process 1:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	16	4	1	1
1	23	53	1	1
2	20	12	1	1

Page Table for Process 2:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	27	22	1	1
1	14	52	1	1
2	12	40	1	1
3	9	56	1	1

Page Table for Process 3:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	10	3	1	1
1	24	2	1	1
2	15	28	1	1
3	18	31	1	1

Page Table for Process 4:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	31	15	1	1
1	26	50	1	1
2	21	0	1	1
3	29	41	1	1

Page Table for Process 5:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	13	1	1	1
1	8	39	1	1

Page Table for Process 6:

Logical Page ID	Frame Number	Block Number	Valid	In_Secondary
0	22	26	1	1
1	17	35	1	1
2	19	18	1	1
3	11	37	1	1

```
Queue is empty
Process 2 is arrived
Process 3 is arrived
Process 6 is arrived
Time 0: Running process 2, Remaining Burst Time: 7, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Process 5 is arrived
Time 1: Running process 3, Remaining Burst Time: 5, PCB Frame Number: 2, Memory Address: 0x7ffe52b75d90

Process 4 is arrived
Time 2: Running process 6, Remaining Burst Time: 4, PCB Frame Number: 5, Memory Address: 0x7ffe52b75f10

Process 1 is arrived
Time 3: Running process 2, Remaining Burst Time: 6, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Time 4: Running process 5, Remaining Burst Time: 5, PCB Frame Number: 4, Memory Address: 0x7ffe52b75e90

Time 5: Running process 3, Remaining Burst Time: 4, PCB Frame Number: 2, Memory Address: 0x7ffe52b75d90

Time 6: Running process 4, Remaining Burst Time: 2, PCB Frame Number: 3, Memory Address: 0x7ffe52b75e10

Time 7: Running process 6, Remaining Burst Time: 3, PCB Frame Number: 5, Memory Address: 0x7ffe52b75f10

Time 8: Running process 1, Remaining Burst Time: 5, PCB Frame Number: 0, Memory Address: 0x7ffe52b75c90

Time 9: Running process 2, Remaining Burst Time: 5, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Time 10: Running process 5, Remaining Burst Time: 4, PCB Frame Number: 4, Memory Address: 0x7ffe52b75e90

Time 11: Running process 3, Remaining Burst Time: 3, PCB Frame Number: 2, Memory Address: 0x7ffe52b75d90

Time 12: Running process 4, Remaining Burst Time: 1, PCB Frame Number: 3, Memory Address: 0x7ffe52b75e10

Process 4 finished at time 13

Time 13: Running process 6, Remaining Burst Time: 2, PCB Frame Number: 5, Memory Address: 0x7ffe52b75f10

Time 14: Running process 1, Remaining Burst Time: 4, PCB Frame Number: 0, Memory Address: 0x7ffe52b75c90

Time 15: Running process 2, Remaining Burst Time: 4, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Time 16: Running process 5, Remaining Burst Time: 3, PCB Frame Number: 4, Memory Address: 0x7ffe52b75e90

Time 17: Running process 3, Remaining Burst Time: 2, PCB Frame Number: 2, Memory Address: 0x7ffe52b75d90

Time 18: Running process 6, Remaining Burst Time: 1, PCB Frame Number: 5, Memory Address: 0x7ffe52b75f10

Process 6 finished at time 19

Time 19: Running process 1, Remaining Burst Time: 3, PCB Frame Number: 0, Memory Address: 0x7ffe52b75c90

Time 20: Running process 2, Remaining Burst Time: 3, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Time 21: Running process 5, Remaining Burst Time: 2, PCB Frame Number: 4, Memory Address: 0x7ffe52b75e90

Time 22: Running process 3, Remaining Burst Time: 1, PCB Frame Number: 2, Memory Address: 0x7ffe52b75d90

Process 3 finished at time 23

Time 23: Running process 1, Remaining Burst Time: 2, PCB Frame Number: 0, Memory Address: 0x7ffe52b75c90

Time 24: Running process 2, Remaining Burst Time: 2, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Time 25: Running process 5, Remaining Burst Time: 1, PCB Frame Number: 4, Memory Address: 0x7ffe52b75e90

Process 5 finished at time 26

Time 26: Running process 1, Remaining Burst Time: 1, PCB Frame Number: 0, Memory Address: 0x7ffe52b75c90

Process 1 finished at time 27

Time 27: Running process 2, Remaining Burst Time: 1, PCB Frame Number: 1, Memory Address: 0x7ffe52b75d10

Process 2 finished at time 28


-------------------------------------------------------------------------------
| Process | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time |
-------------------------------------------------------------------------------
| 1       | 3            | 5          | 27              | 19           | 24              |
| 2       | 0            | 7          | 28              | 21           | 28              |
| 3       | 0            | 5          | 23              | 18           | 23              |
| 4       | 2            | 2          | 13              | 9            | 11              |
| 5       | 1            | 5          | 26              | 20           | 25              |
| 6       | 0            | 4          | 19              | 15           | 19              |
-------------------------------------------------------------------------------

커맨드를 입력하세요(종료:exit) : exit
[MiniOS SSU] System Shutdown.......
```

이렇게 지금까지 모든 구현한 모든 기능들을 모아 우리 팀만의 MiniOS를 갖춘 시스템을 구현할 수 있었고 의도한 대로 성공적으로 작동함을 알 수 있었다.

본 프로젝트를 통해 운영체제의 핵심 기능인 프로세스 생성, 메모리 및 보조 저장 장치 할당, CPU 스케줄링 과정등의 흐름과 전체적인 시스템의 흐름을 명확하게 파악할 수 있었다. 또 이론수업에서는 알지 못했던 실제 구현 과정에서의 어려움과 이를 해결하기 위해 여러가지 방법들을 시도하고 결국 해냄으로써 자신감과 끈기를 얻을 수 있었고, 실용적으로 많은 경험을 쌓을 수 있는 값진 기회였다고 생각한다. 앞으로도 이러한 경험과 자신감 및 끈기를 바탕으로 닥쳐올 일에 대해 유연하게 헤쳐 나아갈 수 있다는 생각이 들었고, 앞으로의 학업, 일, 연구를 넘어서 삶에까지 큰 영향을 미칠 밑거름이 될것이라고 생각한다.