



HỆ ĐIỀU HÀNH (CO2017)

Assignment 2

Simple operating system

Teacher: Nguyễn Quang Hùng
Teacher Assistant: Hoàng Lê Hải Thanh
Class: L04, Group: 04
Members: Võ Hùng - 2013375
Trần Quốc Thái - 2010616
Đào Đức Thiện - 1713287
Bùi Hoàng Minh - 2010410

Mục lục

1	Scheduler	2
1.1	Implementation	2
1.1.1	Thêm PCB mới vào queue bằng hàm enqueue()	2
1.1.2	Lấy PCB ra khỏi queue bằng hàm dequeue()	2
1.1.3	Lấy PCB của process trong ready_queue bằng hàm get_proc()	2
1.2	Kết quả kiểm thử và biểu đồ Gantt mô tả các quá trình được thực thi bởi CPU	3
1.2.1	Kiểm thử testcase sched_0	3
1.2.2	Kiểm thử testcase sched_1	4
1.2.3	Kiểm thử testcase sched_2	6
1.2.4	Kiểm thử testcase sched_3	8
1.3	Trả lời câu hỏi	10
2	Memory management	10
2.1	Implementation	10
2.1.1	Tìm page table từ segment index bằng hàm get_page_table()	10
2.1.2	Ánh xạ địa chỉ ảo thành địa chỉ vật lý bằng hàm translate()	10
2.1.3	Cấp phát memory bằng hàm alloc_mem()	11
2.1.4	Thu hồi memory bằng hàm free_mem()	13
2.2	Kết quả kiểm thử và giải thích	14
2.2.1	Kiểm thử process m0	14
2.2.2	Kiểm thử process m1	15
2.2.3	Kiểm thử process m2	15
2.2.4	Kiểm thử process m3	16
2.3	Trả lời câu hỏi	16
3	Kết hợp Scheduler và Memory management	17
3.1	Mutex lock	17
3.1.1	Mutex lock trong Scheduler	17
3.1.2	Mutex lock trong Memory management	17
3.2	Kết quả kiểm thử và biểu đồ Gantt cho định thời CPU	18
3.2.1	Kiểm thử testcase os_0	18
3.2.2	Kiểm thử testcase os_1	19
3.2.3	Kiểm thử testcase os_2	22
3.2.4	Kiểm thử testcase os_3	24

1 Scheduler

1.1 Implementation

1.1.1 Thêm PCB mới vào queue bằng hàm enqueue()

- Ý tưởng: Kiểm tra xem hàng đợi đã đầy hay chưa, nếu chưa thì thêm process vào hàng đợi và tăng size của hàng đợi lên 1.

- Code

```
1 void enqueue(struct queue_t * q, struct pcb_t * proc) {
2     /* TODO: put a new process to queue [q] */
3     if (q->size == MAX_QUEUE_SIZE)
4     {
5         printf("Queue is full \n");
6         return;
7     }
8     q->proc[q->size++] = proc;
9 }
```

1.1.2 Lấy PCB ra khỏi queue bằng hàm dequeue()

- Ý tưởng

Bước 1: Kiểm tra xem hàng đợi có rỗng hay không, và in ra thông báo

Bước 2: Duyệt từng process trong queue và so sánh độ ưu tiên

Bước 3: Lấy ra process có độ ưu tiên cao nhất và dời các process phía sau nó lên trước 1 đơn vị.

- Code

```
1 struct pcb_t * dequeue(struct queue_t * q)
2 {
3     /* TODO: return a pcb whose priority is the highest
4     * in the queue [q] and remember to remove it from q
5     */
6     if (q->size == 0)
7     {
8         printf("Queue is empty");
9         return NULL;
10    }
11
12    int i = 0, j;
13    for (j = 1; j < q->size; j++) {
14        if (q->proc[j]->priority < q->proc[i]->priority)
15            i = j;
16    }
17    struct pcb_t * res = q->proc[i];
18    for (j = i+1; j < q->size; j++)
19        q->proc[j-1] = q->proc[j];
20    q->size--;
21    return res;
22 }
```

1.1.3 Lấy PCB của process trong ready_queue bằng hàm get_proc()

- Ý tưởng: Kiểm tra hàng đợi ready_queue đã rỗng hay chưa, nếu rỗng ra sẽ push toàn bộ phần tử trong run_queue vào ready_queue, ngược lại ta sẽ lấy ra process có độ ưu tiên cao nhất và chuyển cho CPU để thực thi.

- Code

```
1 struct pcb_t * get_proc(void) {
2     struct pcb_t * proc = NULL;
3     /*TODO: get a process from [ready_queue]. If ready queue
4     * is empty, push all processes in [run_queue] back to
```

```
5  * [ready_queue] and return the highest priority one.
6  * Remember to use lock to protect the queue.
7  * */
8  pthread_mutex_lock(&queue_lock);
9  if (empty(&ready_queue)) {
10 // move all process is waiting in run_queue back to ready_queue
11 while (!empty(&run_queue)) {
12     enqueue(&ready_queue, dequeue(&run_queue));
13 }
14 }
15
16 if (!empty(&ready_queue)) {
17     proc = dequeue(&ready_queue);
18 }
19 pthread_mutex_unlock(&queue_lock);
20
21 return proc;
22 }
```

1.2 Kết quả kiểm thử và biểu đồ Gantt mô tả các quá trình được thực thi bởi CPU

1.2.1 Kiểm thử testcase sched_0

- Testcase sched_0

```
1 2 1 2
2 0 s0
3 4 s1
```

- Kết quả kiểm thử

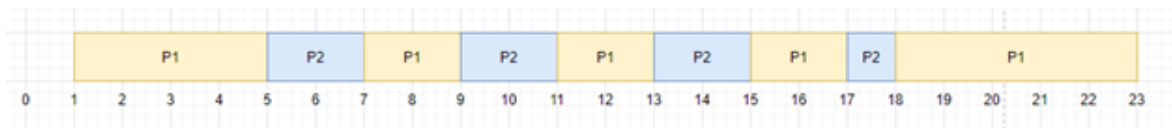
```
1 ./os sched_0
2 Time slot 0
3   Loaded a process at input/proc/s0, PID: 1
4 Time slot 1
5   CPU 0: Dispatched process 1
6 Time slot 2
7 Time slot 3
8   CPU 0: Put process 1 to run queue
9   CPU 0: Dispatched process 1
10 Time slot 4
11   Loaded a process at input/proc/s1, PID: 2
12 Time slot 5
13   CPU 0: Put process 1 to run queue
14   CPU 0: Dispatched process 2
15 Time slot 6
16 Time slot 7
17   CPU 0: Put process 2 to run queue
18   CPU 0: Dispatched process 1
19 Time slot 8
20 Time slot 9
21   CPU 0: Put process 1 to run queue
22   CPU 0: Dispatched process 2
23 Time slot 10
24 Time slot 11
25   CPU 0: Put process 2 to run queue
26   CPU 0: Dispatched process 1
27 Time slot 12
28 Time slot 13
29   CPU 0: Put process 1 to run queue
30   CPU 0: Dispatched process 2
31 Time slot 14
32 Time slot 15
33   CPU 0: Put process 2 to run queue
34   CPU 0: Dispatched process 1
35 Time slot 16
36 Time slot 17
37   CPU 0: Put process 1 to run queue
38   CPU 0: Dispatched process 2
```

```

39 Time slot 18
40 CPU 0: Processed 2 has finished
41 CPU 0: Dispatched process 1
42 Time slot 19
43 Time slot 20
44 CPU 0: Put process 1 to run queue
45 CPU 0: Dispatched process 1
46 Time slot 21
47 Time slot 22
48 CPU 0: Put process 1 to run queue
49 CPU 0: Dispatched process 1
50 Time slot 23
51 CPU 0: Processed 1 has finished
52 CPU 0 stopped
53
54 MEMORY CONTENT:

```

- Biểu đồ Gantt mô tả các quá trình được thực thi bởi CPU



Hình 1: Biểu đồ Gantt cho testcase sched_0

1.2.2 Kiểm thử testcase sched_1

- Testcase sched_1

```

1 2 1 4
2 0 s0
3 4 s1
4 6 s2
5 7 s3

```

- Kết quả kiểm thử

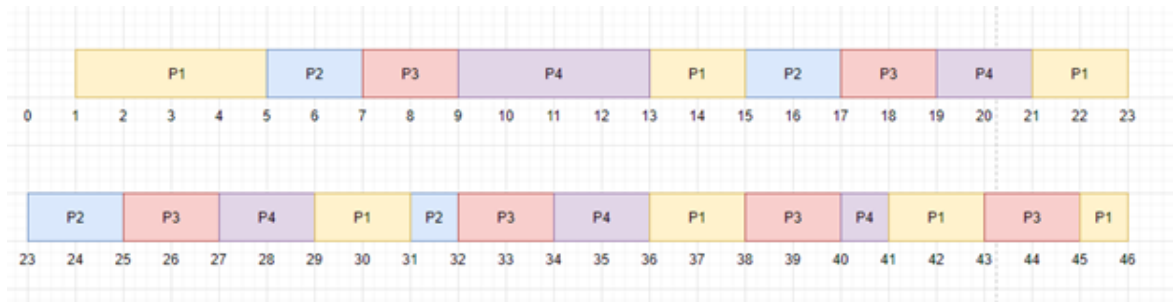
```

1 ./os sched_1
2   Loaded a process at input/proc/s0, PID: 1
3 Time slot 0
4 Time slot 1
5 CPU 0: Dispatched process 1
6 Time slot 2
7 Time slot 3
8 CPU 0: Put process 1 to run queue
9 CPU 0: Dispatched process 1
10 Time slot 4
11   Loaded a process at input/proc/s1, PID: 2
12 Time slot 5
13 CPU 0: Put process 1 to run queue
14 CPU 0: Dispatched process 2
15 Time slot 6
16   Loaded a process at input/proc/s2, PID: 3
17 Time slot 7
18 CPU 0: Put process 2 to run queue
19 CPU 0: Dispatched process 3
20   Loaded a process at input/proc/s3, PID: 4
21 Time slot 8
22 Time slot 9
23 CPU 0: Put process 3 to run queue
24 CPU 0: Dispatched process 4
25 Time slot 10
26 Time slot 11
27 CPU 0: Put process 4 to run queue
28 CPU 0: Dispatched process 4
29 Time slot 12
30 Time slot 13
31 CPU 0: Put process 4 to run queue
32 CPU 0: Dispatched process 1

```

```
33 Time slot 14
34 Time slot 15
35 CPU 0: Put process 1 to run queue
36 CPU 0: Dispatched process 2
37 Time slot 16
38 Time slot 17
39 CPU 0: Put process 2 to run queue
40 CPU 0: Dispatched process 3
41 Time slot 18
42 Time slot 19
43 CPU 0: Put process 3 to run queue
44 CPU 0: Dispatched process 4
45 Time slot 20
46 Time slot 21
47 CPU 0: Put process 4 to run queue
48 CPU 0: Dispatched process 1
49 Time slot 22
50 Time slot 23
51 CPU 0: Put process 1 to run queue
52 CPU 0: Dispatched process 2
53 Time slot 24
54 Time slot 25
55 CPU 0: Put process 2 to run queue
56 CPU 0: Dispatched process 3
57 Time slot 26
58 Time slot 27
59 CPU 0: Put process 3 to run queue
60 CPU 0: Dispatched process 4
61 Time slot 28
62 Time slot 29
63 CPU 0: Put process 4 to run queue
64 CPU 0: Dispatched process 1
65 Time slot 30
66 Time slot 31
67 CPU 0: Put process 1 to run queue
68 CPU 0: Dispatched process 2
69 Time slot 32
70 CPU 0: Processed 2 has finished
71 CPU 0: Dispatched process 3
72 Time slot 33
73 Time slot 34
74 CPU 0: Put process 3 to run queue
75 CPU 0: Dispatched process 4
76 Time slot 35
77 Time slot 36
78 CPU 0: Put process 4 to run queue
79 CPU 0: Dispatched process 1
80 Time slot 37
81 Time slot 38
82 CPU 0: Put process 1 to run queue
83 CPU 0: Dispatched process 3
84 Time slot 39
85 Time slot 40
86 CPU 0: Put process 3 to run queue
87 CPU 0: Dispatched process 4
88 Time slot 41
89 CPU 0: Processed 4 has finished
90 CPU 0: Dispatched process 1
91 Time slot 42
92 Time slot 43
93 CPU 0: Put process 1 to run queue
94 CPU 0: Dispatched process 3
95 Time slot 44
96 Time slot 45
97 CPU 0: Processed 3 has finished
98 CPU 0: Dispatched process 1
99 Time slot 46
100 CPU 0: Processed 1 has finished
101 CPU 0 stopped
102
103 MEMORY CONTENT:
```

- Biểu đồ Gantt mô tả các quá trình được thực thi bởi CPU



Hình 2: Biểu đồ Gantt cho testcase sched_1

1.2.3 Kiểm thử testcase sched_2

- Testcase sched_2

```
1 1 1 4
2 1 s1
3 3 s0
4 5 s3
5 7 s2
```

- Kết quả kiểm thử

```
1 ./os sched_2
2 Time slot 0
3 Time slot 1
4   Loaded a process at input/proc/s1, PID: 1
5 Time slot 2
6   CPU 0: Dispatched process 1
7 Time slot 3
8   CPU 0: Put process 1 to run queue
9   CPU 0: Dispatched process 1
10  Loaded a process at input/proc/s0, PID: 2
11 Time slot 4
12  CPU 0: Put process 1 to run queue
13  CPU 0: Dispatched process 2
14 Time slot 5
15  CPU 0: Put process 2 to run queue
16  CPU 0: Dispatched process 2
17  Loaded a process at input/proc/s3, PID: 3
18 Time slot 6
19  CPU 0: Put process 2 to run queue
20  CPU 0: Dispatched process 3
21 Time slot 7
22  CPU 0: Put process 3 to run queue
23  CPU 0: Dispatched process 1
24  Loaded a process at input/proc/s2, PID: 4
25 Time slot 8
26  CPU 0: Put process 1 to run queue
27  CPU 0: Dispatched process 4
28 Time slot 9
29  CPU 0: Put process 4 to run queue
30  CPU 0: Dispatched process 3
31 Time slot 10
32  CPU 0: Put process 3 to run queue
33  CPU 0: Dispatched process 2
34 Time slot 11
35  CPU 0: Put process 2 to run queue
36  CPU 0: Dispatched process 1
37 Time slot 12
38  CPU 0: Put process 1 to run queue
39  CPU 0: Dispatched process 4
40 Time slot 13
41  CPU 0: Put process 4 to run queue
42  CPU 0: Dispatched process 3
43 Time slot 14
```

```
44 CPU 0: Put process 3 to run queue
45 CPU 0: Dispatched process 2
46 Time slot 15
47 CPU 0: Put process 2 to run queue
48 CPU 0: Dispatched process 1
49 Time slot 16
50 CPU 0: Put process 1 to run queue
51 CPU 0: Dispatched process 4
52 Time slot 17
53 CPU 0: Put process 4 to run queue
54 CPU 0: Dispatched process 3
55 Time slot 18
56 CPU 0: Put process 3 to run queue
57 CPU 0: Dispatched process 2
58 Time slot 19
59 CPU 0: Put process 2 to run queue
60 CPU 0: Dispatched process 1
61 Time slot 20
62 CPU 0: Put process 1 to run queue
63 CPU 0: Dispatched process 4
64 Time slot 21
65 CPU 0: Put process 4 to run queue
66 CPU 0: Dispatched process 3
67 Time slot 22
68 CPU 0: Put process 3 to run queue
69 CPU 0: Dispatched process 2
70 Time slot 23
71 CPU 0: Put process 2 to run queue
72 CPU 0: Dispatched process 1
73 Time slot 24
74 CPU 0: Processed 1 has finished
75 CPU 0: Dispatched process 4
76 Time slot 25
77 CPU 0: Put process 4 to run queue
78 CPU 0: Dispatched process 3
79 Time slot 26
80 CPU 0: Put process 3 to run queue
81 CPU 0: Dispatched process 2
82 Time slot 27
83 CPU 0: Put process 2 to run queue
84 CPU 0: Dispatched process 4
85 Time slot 28
86 CPU 0: Put process 4 to run queue
87 CPU 0: Dispatched process 3
88 Time slot 29
89 CPU 0: Put process 3 to run queue
90 CPU 0: Dispatched process 2
91 Time slot 30
92 CPU 0: Put process 2 to run queue
93 CPU 0: Dispatched process 4
94 Time slot 31
95 CPU 0: Put process 4 to run queue
96 CPU 0: Dispatched process 3
97 Time slot 32
98 CPU 0: Put process 3 to run queue
99 CPU 0: Dispatched process 2
100 Time slot 33
101 CPU 0: Put process 2 to run queue
102 CPU 0: Dispatched process 4
103 Time slot 34
104 CPU 0: Put process 4 to run queue
105 CPU 0: Dispatched process 3
106 Time slot 35
107 CPU 0: Put process 3 to run queue
108 CPU 0: Dispatched process 2
109 Time slot 36
110 CPU 0: Put process 2 to run queue
111 CPU 0: Dispatched process 4
112 Time slot 37
113 CPU 0: Put process 4 to run queue
114 CPU 0: Dispatched process 3
```



```
115 Time slot 38
116 CPU 0: Put process 3 to run queue
117 CPU 0: Dispatched process 2
118 Time slot 39
119 CPU 0: Put process 2 to run queue
120 CPU 0: Dispatched process 4
121 Time slot 40
122 CPU 0: Put process 4 to run queue
123 CPU 0: Dispatched process 3
124 Time slot 41
125 CPU 0: Processed 3 has finished
126 CPU 0: Dispatched process 2
127 Time slot 42
128 CPU 0: Put process 2 to run queue
129 CPU 0: Dispatched process 4
130 Time slot 43
131 CPU 0: Put process 4 to run queue
132 CPU 0: Dispatched process 2
133 Time slot 44
134 CPU 0: Put process 2 to run queue
135 CPU 0: Dispatched process 4
136 Time slot 45
137 CPU 0: Processed 4 has finished
138 CPU 0: Dispatched process 2
139 Time slot 46
140 CPU 0: Put process 2 to run queue
141 CPU 0: Dispatched process 2
142 Time slot 47
143 CPU 0: Processed 2 has finished
144 CPU 0 stopped
145
146 MEMORY CONTENT:
```

1.2.4 Kiểm thử testcase sched_3

- Testcase sched_3

```
1 1 2 3
2 0 s1
3 5 s0
4 15 s2
```

- Kết quả kiểm thử

```
1 ./os sched_3
2 Loaded a process at input/proc/s1, PID: 1
3 Time slot 0
4 Time slot 1
5 CPU 1: Dispatched process 1
6 Time slot 2
7 CPU 1: Put process 1 to run queue
8 CPU 1: Dispatched process 1
9 Time slot 3
10 CPU 1: Put process 1 to run queue
11 CPU 1: Dispatched process 1
12 Time slot 4
13 CPU 1: Put process 1 to run queue
14 CPU 1: Dispatched process 1
15 Time slot 5
16 CPU 1: Put process 1 to run queue
17 CPU 1: Dispatched process 1
18 Loaded a process at input/proc/s0, PID: 2
19 Time slot 6
20 CPU 0: Dispatched process 2
21 CPU 1: Put process 1 to run queue
22 CPU 1: Dispatched process 1
23 Time slot 7
24 CPU 0: Put process 2 to run queue
25 CPU 0: Dispatched process 2
26 CPU 1: Put process 1 to run queue
```

```
27 CPU 1: Dispatched process 1
28 Time slot 8
29 CPU 0: Put process 2 to run queue
30 CPU 0: Dispatched process 2
31 CPU 1: Processed 1 has finished
32 Time slot 9
33 CPU 0: Put process 2 to run queue
34 CPU 0: Dispatched process 2
35 Time slot 10
36 CPU 0: Put process 2 to run queue
37 CPU 0: Dispatched process 2
38 Time slot 11
39 CPU 0: Put process 2 to run queue
40 CPU 0: Dispatched process 2
41 Time slot 12
42 CPU 0: Put process 2 to run queue
43 CPU 0: Dispatched process 2
44 Time slot 13
45 CPU 0: Put process 2 to run queue
46 CPU 0: Dispatched process 2
47 Time slot 14
48 CPU 0: Put process 2 to run queue
49 CPU 0: Dispatched process 2
50 Time slot 15
51 CPU 0: Put process 2 to run queue
52 CPU 0: Dispatched process 2
53 Loaded a process at input/proc/s2, PID: 3
54 Time slot 16
55 CPU 1: Dispatched process 3
56 CPU 0: Put process 2 to run queue
57 CPU 0: Dispatched process 2
58 Time slot 17
59 CPU 1: Put process 3 to run queue
60 CPU 1: Dispatched process 3
61 CPU 0: Put process 2 to run queue
62 CPU 0: Dispatched process 2
63 Time slot 18
64 CPU 1: Put process 3 to run queue
65 CPU 1: Dispatched process 3
66 CPU 0: Put process 2 to run queue
67 CPU 0: Dispatched process 2
68 Time slot 19
69 CPU 1: Put process 3 to run queue
70 CPU 1: Dispatched process 3
71 CPU 0: Put process 2 to run queue
72 CPU 0: Dispatched process 2
73 Time slot 20
74 CPU 1: Put process 3 to run queue
75 CPU 1: Dispatched process 3
76 CPU 0: Put process 2 to run queue
77 CPU 0: Dispatched process 2
78 Time slot 21
79 CPU 1: Put process 3 to run queue
80 CPU 1: Dispatched process 3
81 CPU 0: Processed 2 has finished
82 CPU 0 stopped
83 Time slot 22
84 CPU 1: Put process 3 to run queue
85 CPU 1: Dispatched process 3
86 Time slot 23
87 CPU 1: Put process 3 to run queue
88 CPU 1: Dispatched process 3
89 Time slot 24
90 CPU 1: Put process 3 to run queue
91 CPU 1: Dispatched process 3
92 Time slot 25
93 CPU 1: Put process 3 to run queue
94 CPU 1: Dispatched process 3
95 Time slot 26
96 CPU 1: Put process 3 to run queue
97 CPU 1: Dispatched process 3
```

```
98 Time slot 27
99 CPU 1: Put process 3 to run queue
100 CPU 1: Dispatched process 3
101 Time slot 28
102 CPU 1: Processed 3 has finished
103 CPU 1 stopped
104
105 MEMORY CONTENT:
```

1.3 Trả lời câu hỏi

- **Question:** What is the advantage of using priority feedback queue in comparison with other scheduling algorithms you have learned?
- **Trả lời:**
 - Tận dụng được ưu điểm của giải thuật RoundRobin sử dụng Quantum Time để giới hạn thời gian xử lý 1 process của CPU để thực hiện đồng đều giữa các process, không xảy ra tình trạng 1 process chiếm dụng CPU quá lâu.
 - Vì chỉ có một hàng đợi nên linh hoạt chuyển giữa các process, không để một process nào không được xử lý như điểm bất lợi trong giải thuật MLQS.
 - Sử dụng hàng chờ run_queue để vẫn có thể giữ lại đúng thứ tự ưu tiên của các process nếu các process đó vẫn chưa được xử lý xong trong khoảng Quantum Time đó và được đưa lại vào ready_queue để xử lý lần sau. Đảm bảo được các process có độ ưu tiên cao hơn không được ưu tiên xử lý khi mà các process có độ ưu tiên thấp hơn đứng trước trong hàng đợi ready_queue vẫn chưa được xử lý.

2 Memory management

2.1 Implementation

2.1.1 Tìm page table từ segment index bằng hàm get_page_table()

- Ý tưởng: duyệt qua từng hàng của segment table để kiểm tra segment nào có v_index trùng với segment index của địa chỉ ảo thì trả về page table mà segment đó đang trỏ đến.

- Code

```
1 static struct page_table_t *get_page_table(addr_t index, struct seg_table_t *
  seg_table)
2 {
3     int i;
4     for (i = 0; i < seg_table->size; i++)
5     {
6         //TODO
7         if (index == seg_table->table[i].v_index)
8         {
9             return seg_table->table[i].pages;
10        }
11    }
12    return NULL;
13 }
```

2.1.2 Ánh xạ địa chỉ ảo thành địa chỉ vật lý bằng hàm translate()

- Ý tưởng

Bước 1: Lần lượt tách 5 bits đầu, 5 bits giữa và 10 bits cuối của địa chỉ ảo tương ứng với segment index, page index và offset

Bước 2: Tìm được page table từ segment index

Bước 3: Từ page table, tìm được page có v_index trùng page index

Bước 4: Tạo địa chỉ vật lý bằng cách thay 10 bits đầu trong địa chỉ ảo thành giá trị p_index của page vừa tìm được

- Code

```
1 static int translate(addr_t virtual_addr, addr_t *physical_addr, struct pcb_t *proc)
2 {
3     addr_t offset = get_offset(virtual_addr);
4     addr_t first_lv = get_first_lv(virtual_addr);
5     addr_t second_lv = get_second_lv(virtual_addr);
6
7     struct page_table_t *page_table = NULL;
8     page_table = get_page_table(first_lv, proc->seg_table);
9     if (page_table == NULL)
10    {
11        return 0;
12    }
13    int i;
14    for (i = 0; i < 1 << PAGE_LEN; i++)
15    {
16        if (page_table->table[i].v_index == second_lv)
17        {
18            //TODO
19            *physical_addr = page_table->table[i].p_index * PAGE_SIZE + offset;
20            return 1;
21        }
22    }
23    return 0;
24 }
```

2.1.3 Cấp phát memory bằng hàm alloc_mem()

- Ý tưởng

1. Kiểm tra số pages trống trong vùng nhớ ảo và vùng nhớ vật lý, nếu đủ pages process yêu cầu thì cho giá trị mem_avail là 1
2. Cấp phát memory: khi giá trị mem_avail là 1

Bước 1: Cập nhật giá trị breakpoint của process

Bước 2: Trong physical memory, tìm các page trống và thực hiện cập nhật giá trị proc, index, next cho các page này

Bước 3: Cập nhật segment table và page table

Bước 3: Quay lại bước 2 hoặc thoát nếu đã cập nhật đủ số pages process yêu cầu

- Code

```
1 addr_t alloc_mem(uint32_t size, struct pcb_t *proc)
2 {
3     pthread_mutex_lock(&mem_lock);
4     addr_t ret_mem = 0;
5
6     uint32_t num_pages = ((size % PAGE_SIZE) == 0) ? size / PAGE_SIZE : size /
7     PAGE_SIZE + 1; // Number of pages we will use
8     int mem_avail = 0; // We could allocate new memory region or not?
9
10    //TODO
11    //Kiểm tra page trong trong vùng nhớ ảo và vùng nhớ vật lý
12    int i;
13    int num_avail_pages = 0;
14    for (i = 0; i < NUM_PAGES; i++)
15    { //Check if ram memory space is available
16        if (_mem_stat[i].proc == 0)
17        {
18            num_avail_pages++;
19            if (num_avail_pages == num_pages && proc->bp + num_pages * PAGE_SIZE <=
20            RAM_SIZE)
21            {
22                mem_avail = 1;
23            }
24        }
25    }
26 }
```

```
21     break;
22   }
23 }
24 }
25
26 if (mem_avail)
27 {
28     ret_mem = proc->bp;
29     proc->bp += num_pages * PAGE_SIZE;
30
31     //TODO
32     //Cap nhat breakpointer, _mem_stat, segment table va page table
33     int num_alloc_pages = 0;
34     int pre_index; // index of the previous page in the list
35     addr_t cur_vir_addr;
36     int seg_idx, page_idx;
37     for (i = 0; i < NUM_PAGES; i++)
38     {
39         if (_mem_stat[i].proc == 0)
40         {
41             _mem_stat[i].proc = proc->pid;
42             _mem_stat[i].index = num_alloc_pages;
43
44             if (_mem_stat[i].index != 0)
45                 _mem_stat[pre_index].next = i;
46             pre_index = i;
47
48             int found = 0;
49             struct seg_table_t *seg_table = proc->seg_table;
50             if (seg_table->table[0].pages == NULL)
51                 seg_table->size = 0;
52
53             cur_vir_addr = ret_mem + (num_alloc_pages << OFFSET_LEN);
54
55             seg_idx = get_first_lv(cur_vir_addr);
56             page_idx = get_second_lv(cur_vir_addr);
57             int j;
58             for (j = 0; j < seg_table->size; j++)
59             {
60                 if (seg_table->table[j].v_index == seg_idx)
61                 {
62                     struct page_table_t *cur_page_table = seg_table->table[j].pages;
63
64                     cur_page_table->table[cur_page_table->size].v_index = page_idx;
65                     cur_page_table->table[cur_page_table->size].p_index = i;
66
67                     cur_page_table->size++;
68
69                     found = 1;
70                     break;
71                 }
72             }
73
74             if (!found)
75             { //If not found, add new row into table
76                 seg_table->table[seg_table->size].v_index = seg_idx;
77                 seg_table->table[seg_table->size].pages = (struct page_table_t *)malloc(
sizeof(struct page_table_t));
78
79                 seg_table->table[seg_table->size].pages->table[0].v_index = page_idx;
80                 seg_table->table[seg_table->size].pages->table[0].p_index = i;
81
82                 seg_table->table[seg_table->size].pages->size = 1;
83
84                 seg_table->size++;
85             }
86
87             num_alloc_pages++;
88             if (num_alloc_pages == num_pages)
89             {
90                 _mem_stat[i].next = -1;
```

```
91         break;
92     }
93 }
94 }
95 }
96
97 pthread_mutex_unlock(&mem_lock);
98 return ret_mem;
99 }
```

2.1.4 Thu hồi memory bằng hàm free_mem()

- Ý tưởng

Bước 1: Tìm page table tương ứng với segment index trong địa chỉ ảo

Bước 2: Tìm vị trí page cần xóa trong physical pages bằng địa chỉ vật lý (được ánh xạ từ địa chỉ ảo) và lưu vào biến p_index

Bước 3: Cập nhật page table bằng cách xóa page có v_index tương ứng với page index trong địa chỉ ảo

Bước 4: Nếu page table trống thì giải phóng page table và cập nhật segment table bằng cách xóa segment trở đến page table vừa giải phóng

Bước 5: Cập nhật p_index thành địa chỉ page tiếp theo trong physical pages và quay lại bước 3

- Code

```
1 int free_mem(addr_t address, struct pcb_t *proc)
2 {
3     pthread_mutex_lock(&mem_lock);
4
5     //TODO
6     struct page_table_t *page_table = get_page_table(get_first_lv(address), proc->
7     seg_table);
8
9     int valid = 0;
10    if (page_table != NULL)
11    {
12        int i;
13        for (i = 0; i < page_table->size; i++)
14        {
15            if (page_table->table[i].v_index == get_second_lv(address))
16            {
17                addr_t physical_addr;
18                if (translate(address, &physical_addr, proc))
19                {
20                    int p_index = physical_addr >> OFFSET_LEN;
21                    int num_free_pages = 0;
22                    addr_t cur_vir_addr = (num_free_pages << OFFSET_LEN) + address;
23                    addr_t seg_idx, page_idx;
24                    do
25                    {
26                        _mem_stat[p_index].proc = 0;
27                        int found = 0;
28                        int k;
29                        seg_idx = get_first_lv(cur_vir_addr);
30                        page_idx = get_second_lv(cur_vir_addr);
31                        for (k = 0; k < proc->seg_table->size && !found; k++)
32                        {
33                            if (proc->seg_table->table[k].v_index == seg_idx)
34                            {
35                                int l;
36                                for (l = 0; l < proc->seg_table->table[k].pages->size; l++)
37                                {
38                                    if (proc->seg_table->table[k].pages->table[l].v_index == page_idx)
39                                    {
40                                        int m;
41                                        for (m = l; m < proc->seg_table->table[k].pages->size - 1; m++)
42                                            //Rearrange page table
43                                        }
44                                    }
45                                }
46                            }
47                        }
48                    } while (p_index < proc->seg_table->size);
49                }
50            }
51        }
52    }
53 }
```

```

41         proc->seg_table->table[k].pages->table[m] = proc->seg_table->
table[k].pages->table[m + 1];
42
43         proc->seg_table->table[k].pages->size--;
44         if (proc->seg_table->table[k].pages->size == 0)
45         { //If page empty
46             free(proc->seg_table->table[k].pages);
47             for (m = k; m < proc->seg_table->size - 1; m++) //Rearrange
segment table
48                 proc->seg_table->table[m] = proc->seg_table->table[m + 1];
49                 proc->seg_table->size--;
50             }
51             found = 1;
52             break;
53         }
54     }
55 }
56 }
57
58     p_index = _mem_stat[p_index].next;
59     num_free_pages++;
60 } while (p_index != -1);
61     valid = 1;
62 }
63     break;
64 }
65 }
66 }
67
68     pthread_mutex_unlock(&mem_lock);
69
70     if (!valid)
71         return 1;
72     else
73         return 0;
74 }

```

2.2 Kết quả kiểm thử và giải thích

2.2.1 Kiểm thử process m0

- Process m0

```

1 1 7
2 alloc 13535 0
3 alloc 1568 1
4 free 0
5 alloc 1386 2
6 alloc 4564 4
7 write 102 1 20
8 write 21 2 1000

```

- Kết quả kiểm thử

```

1 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
2 003e8: 15
3 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
4 002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
5 003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
6 004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
7 005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
8 006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
9 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
10 03814: 66
11 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)

```

- Giải thích

- Dòng lệnh 2 và 3 lần lượt cấp phát 14 pages trên bộ nhớ vật lý với địa chỉ page đầu tiên ở register 0 và 2 pages với địa chỉ page đầu tiên ở register 1

- Dòng lệnh 4 thu hồi các pages đã cấp phát có địa chỉ page đầu tiên ở register 0
- Dòng lệnh 5 và 6 tương tự dòng 2 và 3
- Dòng lệnh 7 ghi giá trị $66_{dec}(102_{hex})$ vào địa chỉ của register $1 + 14_{hex}(20_{dec})$, nghĩa là $0x03800 + 14$
- Dòng lệnh 8 tương tự dòng lệnh 7, ghi giá trị $15_{dec}(21_{hex})$ vào địa chỉ của register $2 + 3E8_{hex}(1000_{dec})$, nghĩa là $0x00000 + 3E8$

2.2.2 Kiểm thử process m1

- Process m1

```
1 1 8
2 alloc 13535 0
3 alloc 1568 1
4 free 0
5 alloc 1386 2
6 alloc 4564 4
7 free 2
8 free 4
9 free 1
```

- Kết quả kiểm thử: trống
- Giải thích: tương tự process m1 nhưng lần này thu hồi tất cả các pages đã cấp phát nên kết quả trống

2.2.3 Kiểm thử process m2

- Process m2

```
1 1 9
2 alloc 9000 0
3 alloc 5000 1
4 alloc 1500 2
5 alloc 1600 3
6 write 99 2 100
7 write 103 0 10
8 write 10 3 500
9 alloc 2500 4
10 write 1000 4 1
```

- Kết quả kiểm thử

```
1 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
2 000a: 67
3 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
4 002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
5 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
6 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
7 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
8 006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
9 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
10 008: 02000-023ff - PID: 01 (idx 008, nxt: -01)
11 009: 02400-027ff - PID: 01 (idx 000, nxt: 010)
12 010: 02800-02bff - PID: 01 (idx 001, nxt: 011)
13 011: 02c00-02fff - PID: 01 (idx 002, nxt: 012)
14 012: 03000-033ff - PID: 01 (idx 003, nxt: 013)
15 013: 03400-037ff - PID: 01 (idx 004, nxt: -01)
16 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
17 03864: 63
18 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
19 016: 04000-043ff - PID: 01 (idx 000, nxt: 017)
20 041f4: 0a
21 017: 04400-047ff - PID: 01 (idx 001, nxt: -01)
22 018: 04800-04bff - PID: 01 (idx 000, nxt: 019)
23 04801: ffffffe8
24 019: 04c00-04fff - PID: 01 (idx 001, nxt: 020)
25 020: 05000-053ff - PID: 01 (idx 002, nxt: -01)
```


- Giải thích

- Dòng lệnh 2, 3, 4 và 5 lần lượt cấp phát 8 pages với địa chỉ page đầu lưu vào register 0, 4 pages với địa chỉ page đầu lưu vào register 1, 2 pages với địa chỉ page đầu lưu vào register 2 và 2 pages với địa chỉ page đầu lưu vào register 3.
- Dòng lệnh 6, 7 và 8 lần lượt ghi giá trị $63_{hex}(99_{dec})$ vào địa chỉ $(0x03800 + 64_{hex}(100_{dec}))$, $67_{hex}(103_{dec})$ vào địa chỉ $(0x00000 + a_{hex}(10_{dec}))$, $a_{hex}(10_{dec})$ vào địa chỉ $(0x04000 + 500_{hex}(1f4_{dec}))$
- Dòng lệnh 9 và 10 giải thích tương tự.

2.2.4 Kiểm thử process m3

- Process m3

```
1 1 7
2 alloc 12345 0
3 alloc 500 1
4 alloc 1234 2
5 write 321 0 599
6 write 765 1 544
7 free 2
8 write 42 0 123
```

- Kết quả kiểm thử

```
1 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
2   0007b: 2a
3   00257: 41
4 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
5 002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
6 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
7 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
8 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
9 006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
10 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
11 008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
12 009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
13 010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
14 011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
15 012: 03000-033ff - PID: 01 (idx 012, nxt: -01)
16 013: 03400-037ff - PID: 01 (idx 000, nxt: -01)
17   03620: ffffffff d
```

2.3 Trả lời câu hỏi

- **Question:** What is the advantage and disadvantage of segmentation with paging?

- **Trả lời:**

- **Page**

- * Ưu điểm:

1. Không có phân mảnh ngoài.
2. Không có phân mảnh nội trên hệ điều hành được cập nhật.
3. Các khung (frames) không nhất thiết phải liền nhau.

- * Nhược điểm:

1. Gây ra sự phân mảnh nội trên các hệ thống cũ.
2. Thời gian tra cứu bộ nhớ lâu hơn so với Segmentation (khắc phục bằng bộ nhớ đệm TLB).

- **Segmentation**

- * Ưu điểm:

1. Không có phân mảnh nội.

2. Segment table tiêu thụ ít không gian hơn so với page table.
3. Kích thước phân đoạn trung bình lớn hơn hầu hết các kích thước trang, điều này cho phép các phân đoạn lưu trữ nhiều dữ liệu của quá trình hơn.
4. Chi phí xử lý ít hơn.
5. Đơn giản hơn để chuyển vị trí các phân đoạn so với việc định vị lại các không gian địa chỉ liên tiếp trên đĩa.
6. Segment table nhỏ hơn page table và chiếm ít bộ nhớ hơn.

* Nhược điểm:

1. Kích thước các segment không bằng nhau gây bất lợi trong việc hoán đổi.
2. Việc chuyển Linux sang các kiến trúc khác nhau rất khó xử lý vì nó hỗ trợ rất hạn chế cho việc phân đoạn.
3. Đòi hỏi sự can thiệp của lập trình viên.
4. Quản lý bộ nhớ tốn kém.
5. Chịu sự phân mảnh ngoại nghiêm trọng.

– Segmentation with paging mechanism

* Ưu điểm:

1. Ít tốn bộ nhớ hơn. Không bị phân mảnh ngoại. Tuy nhiên ngày nay, một vài hệ thống có kích thước page table khác với kích thước segment table vẫn bị phân mảnh ngoại.
2. Đơn giản hóa việc phân bổ bộ nhớ.
3. Vì toàn bộ segment không cần phải được hoán đổi nên việc hoán đổi vào bộ nhớ ảo trở nên dễ dàng hơn.

* Nhược điểm:

1. Chưa giải quyết được phân mảnh nội vẫn còn tồn tại trong paging.
2. Cần bổ sung thêm phần cứng.
3. Việc translation trở nên tuần tự hơn làm tăng thời gian truy cập bộ nhớ.
4. Độ phức tạp cao hơn nhiều so với paging.
5. Page table cần được lưu trữ liên tục trong bộ nhớ.

3 Kết hợp Scheduler và Memory management

3.1 Mutex lock

3.1.1 Mutex lock trong Scheduler

CPU thực hiện tranh chấp trong 2 hàng đợi `ready_queue` và `run_queue` để lấy process ra từ 2 hàng đợi này và đưa vào thực thi

- **Tranh chấp ở hàm `get_proc()`:** Khi các CPU đồng thời gọi hàm này sẽ xảy ra tranh chấp process lấy ra để thực thi trên CPU, vì thế để bảo vệ ta dùng mutex lock ở đầu và cuối hàm `get_proc()`.
- **Tranh chấp ở 2 hàm `put_proc()` và `add_proc()`:** Hai hàm này đã được bao sẵn mutex lock ở phần định nghĩa hàm trong source code

3.1.2 Mutex lock trong Memory management

Các CPU sẽ thực hiện tranh chấp nhau chủ yếu ở các phần liên quan đến main memory. Các biến được chia sẻ đó là `_mem_stat` và `_ram`. Các hàm có sử dụng các biến trên là `alloc_mem()`, `free_mem()`, `read_mem()`, `write_mem()`. Cụ thể:

- Ở hàm `alloc_mem()` và `free_mem()`: Hai hàm này thực hiện cung cấp bộ nhớ và giải phóng bộ nhớ cho các process. Việc này phải xảy ra tách rời nhau giữa các CPU khi yêu cầu gọi các hàm này, nếu không sẽ xảy ra hiện tượng xung đột bộ nhớ. Ở hai hàm này sẽ sử dụng mutex lock [`mem_lock`] do file `mem.c` cung cấp để bao đầu và cuối hai hàm.

- Ở hàm `read_mem()` và `write_mem()`: Do việc ghi và đọc lên các memory được thực hiện bởi các CPU khác nhau với các process khác nhau, vì vậy các frame ở main memory mà các CPU truy xuất cũng sẽ khác nhau. Cho nên ở hai hàm này sẽ không xuất hiện hiện tượng xung đột bộ nhớ.

3.2 Kết quả kiểm thử và biểu đồ Gantt cho định thời CPU

3.2.1 Kiểm thử testcase `os_0`

- Testcase `os_0`

```
1 6 2 4
2 0 p0
3 2 p1
```

- Kết quả kiểm thử

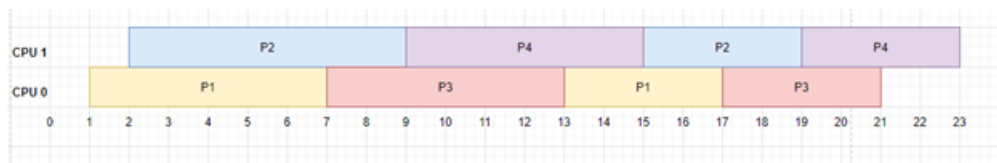
```
1 ./os os_0
2 Time slot 0
3   Loaded a process at input/proc/p0, PID: 1
4 Time slot 1
5   CPU 1: Dispatched process 1
6 Time slot 2
7   Loaded a process at input/proc/p1, PID: 2
8 Time slot 3
9   CPU 1: Dispatched process 2
10  Loaded a process at input/proc/p1, PID: 3
11 Time slot 4
12  Loaded a process at input/proc/p1, PID: 4
13 Time slot 5
14 Time slot 6
15 Time slot 7
16  CPU 0: Put process 1 to run queue
17  CPU 0: Dispatched process 3
18 Time slot 8
19 Time slot 9
20  CPU 1: Put process 2 to run queue
21  CPU 1: Dispatched process 4
22 Time slot 10
23 Time slot 11
24 Time slot 12
25 Time slot 13
26  CPU 0: Put process 3 to run queue
27  CPU 0: Dispatched process 1
28 Time slot 14
29 Time slot 15
30  CPU 1: Put process 4 to run queue
31  CPU 1: Dispatched process 2
32 Time slot 16
33 Time slot 17
34  CPU 0: Processed 1 has finished
35  CPU 0: Dispatched process 3
36 Time slot 18
37 Time slot 19
38  CPU 1: Processed 2 has finished
39  CPU 1: Dispatched process 4
40 Time slot 20
41 Time slot 21
42  CPU 0: Processed 3 has finished
43  CPU 0 stopped
44 Time slot 22
45 Time slot 23
46  CPU 1: Processed 4 has finished
47  CPU 1 stopped
48
49 MEMORY CONTENT:
50 000: 00000-003ff - PID: 02 (idx 000, nxt: 001)
51 001: 00400-007ff - PID: 02 (idx 001, nxt: 007)
52 002: 00800-00bff - PID: 02 (idx 000, nxt: 003)
53 003: 00c00-00fff - PID: 02 (idx 001, nxt: 004)
54 004: 01000-013ff - PID: 02 (idx 002, nxt: 005)
```

```

55 005: 01400-017ff - PID: 02 (idx 003, nxt: -01)
56 006: 01800-01bff - PID: 03 (idx 000, nxt: 011)
57 007: 01c00-01fff - PID: 02 (idx 002, nxt: 008)
58   01de7: 0a
59 008: 02000-023ff - PID: 02 (idx 003, nxt: 009)
60 009: 02400-027ff - PID: 02 (idx 004, nxt: -01)
61 010: 02800-02bff - PID: 01 (idx 000, nxt: -01)
62   02814: 64
63 011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
64 012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
65 013: 03400-037ff - PID: 03 (idx 003, nxt: -01)
66 014: 03800-03bff - PID: 04 (idx 000, nxt: 025)
67 015: 03c00-03fff - PID: 03 (idx 000, nxt: 016)
68 016: 04000-043ff - PID: 03 (idx 001, nxt: 017)
69 017: 04400-047ff - PID: 03 (idx 002, nxt: 018)
70   045e7: 0a
71 018: 04800-04bff - PID: 03 (idx 003, nxt: 019)
72 019: 04c00-04fff - PID: 03 (idx 004, nxt: -01)
73 020: 05000-053ff - PID: 04 (idx 000, nxt: 021)
74 021: 05400-057ff - PID: 04 (idx 001, nxt: 022)
75 022: 05800-05bff - PID: 04 (idx 002, nxt: 023)
76   059e7: 0a
77 023: 05c00-05fff - PID: 04 (idx 003, nxt: 024)
78 024: 06000-063ff - PID: 04 (idx 004, nxt: -01)
79 025: 06400-067ff - PID: 04 (idx 001, nxt: 026)
80 026: 06800-06bff - PID: 04 (idx 002, nxt: 027)
81 027: 06c00-06fff - PID: 04 (idx 003, nxt: -01)

```

- Biểu đồ Gantt cho định thời CPU



Hình 3: Biểu đồ Gantt của testcase os_0

3.2.2 Kiểm thử testcase os_1

- Testcase os_1

```

1 2 4 8
2 1 p0
3 2 s3
4 4 m1
5 6 s2
6 7 m0
7 9 p1
8 11 s0
9 16 s1

```

- Kết quả kiểm thử

```

1 ./os os_1
2 Time slot 0
3 Time slot 1
4   Loaded a process at input/proc/p0, PID: 1
5   CPU 2: Dispatched process 1
6 Time slot 2
7   Loaded a process at input/proc/s3, PID: 2
8 Time slot 3
9   CPU 1: Dispatched process 2
10  CPU 2: Put process 1 to run queue
11  CPU 2: Dispatched process 1
12 Time slot 4
13   Loaded a process at input/proc/m1, PID: 3
14 Time slot 5
15  CPU 0: Dispatched process 3

```

```
16 CPU 1: Put process 2 to run queue
17 CPU 1: Dispatched process 2
18 CPU 2: Put process 1 to run queue
19 CPU 2: Dispatched process 1
20 Time slot 6
21   Loaded a process at input/proc/s2, PID: 4
22 Time slot 7
23 CPU 1: Put process 2 to run queue
24 CPU 1: Dispatched process 2
25 CPU 2: Put process 1 to run queue
26   Loaded a process at input/proc/m0, PID: 5
27 CPU 0: Put process 3 to run queue
28 CPU 0: Dispatched process 5
29 CPU 2: Dispatched process 3
30 CPU 3: Dispatched process 4
31 Time slot 8
32 Time slot 9
33 CPU 1: Put process 2 to run queue
34 CPU 1: Dispatched process 1
35 CPU 0: Put process 5 to run queue
36   Loaded a process at input/proc/p1, PID: 6
37 CPU 0: Put process 3 to run queue
38 CPU 2: Dispatched process 6
39 CPU 2: Put process 4 to run queue
40 CPU 3: Dispatched process 2
41 Time slot 10
42 Time slot 11
43 CPU 2: Put process 6 to run queue
44 CPU 0: Put process 5 to run queue
45 CPU 0: Dispatched process 3
46   Loaded a process at input/proc/s0, PID: 7
47 CPU 2: Dispatched process 5
48 CPU 1: Put process 1 to run queue
49 CPU 1: Dispatched process 7
50 CPU 3: Put process 2 to run queue
51 CPU 3: Dispatched process 4
52 Time slot 12
53 Time slot 13
54 CPU 2: Put process 5 to run queue
55 CPU 0: Put process 3 to run queue
56 CPU 0: Dispatched process 6
57 CPU 1: Put process 7 to run queue
58 CPU 1: Dispatched process 1
59 CPU 2: Dispatched process 3
60 CPU 3: Processed 4 has finished
61 CPU 3: Dispatched process 2
62 Time slot 14
63 Time slot 15
64 CPU 0: Put process 6 to run queue
65 CPU 0: Dispatched process 5
66 CPU 1: Processed 1 has finished
67 CPU 3: Put process 2 to run queue
68 CPU 3: Dispatched process 6
69 CPU 1: Dispatched process 7
70 CPU 2: Processed 3 has finished
71 CPU 2: Dispatched process 4
72 Time slot 16
73 CPU 0: Processed 5 has finished
74 CPU 0: Dispatched process 2
75   Loaded a process at input/proc/s1, PID: 8
76 Time slot 17
77 CPU 0: Processed 2 has finished
78 CPU 0: Dispatched process 8
79 CPU 3: Put process 6 to run queue
80 CPU 3: Dispatched process 6
81 CPU 2: Put process 4 to run queue
82 CPU 2: Dispatched process 4
83 CPU 1: Put process 7 to run queue
84 CPU 1: Dispatched process 7
85 Time slot 18
86 Time slot 19
```

```
87 CPU 0: Put process 8 to run queue
88 CPU 0: Dispatched process 8
89 CPU 1: Put process 7 to run queue
90 CPU 3: Put process 6 to run queue
91 CPU 3: Dispatched process 6
92 CPU 2: Put process 4 to run queue
93 CPU 2: Dispatched process 4
94 CPU 1: Dispatched process 7
95 Time slot 20
96 CPU 3: Processed 6 has finished
97 CPU 3 stopped
98 Time slot 21
99 CPU 0: Put process 8 to run queue
100 CPU 0: Dispatched process 8
101 CPU 1: Put process 7 to run queue
102 CPU 1: Dispatched process 7
103 CPU 2: Put process 4 to run queue
104 CPU 2: Dispatched process 4
105 Time slot 22
106 Time slot 23
107 CPU 0: Put process 8 to run queue
108 CPU 0: Dispatched process 8
109 CPU 1: Put process 7 to run queue
110 CPU 1: Dispatched process 7
111 CPU 2: Processed 4 has finished
112 CPU 2 stopped
113 Time slot 24
114 CPU 0: Processed 8 has finished
115 CPU 0 stopped
116 Time slot 25
117 CPU 1: Put process 7 to run queue
118 CPU 1: Dispatched process 7
119 Time slot 26
120 Time slot 27
121 CPU 1: Put process 7 to run queue
122 CPU 1: Dispatched process 7
123 Time slot 28
124 CPU 1: Processed 7 has finished
125 CPU 1 stopped
126
127 MEMORY CONTENT:
128 000: 00000-003ff - PID: 06 (idx 000, nxt: 001)
129 001: 00400-007ff - PID: 06 (idx 001, nxt: 031)
130 006: 01800-01bff - PID: 06 (idx 000, nxt: 009)
131 007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
132 01fe8: 15
133 008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
134 009: 02400-027ff - PID: 06 (idx 001, nxt: 010)
135 010: 02800-02bff - PID: 06 (idx 002, nxt: 011)
136 011: 02c00-02fff - PID: 06 (idx 003, nxt: -01)
137 021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
138 05414: 64
139 024: 06000-063ff - PID: 05 (idx 000, nxt: 025)
140 06014: 66
141 025: 06400-067ff - PID: 05 (idx 001, nxt: -01)
142 031: 07c00-07fff - PID: 06 (idx 002, nxt: 032)
143 07de7: 0a
144 032: 08000-083ff - PID: 06 (idx 003, nxt: 033)
145 033: 08400-087ff - PID: 06 (idx 004, nxt: -01)
146 049: 0c400-0c7ff - PID: 05 (idx 000, nxt: 050)
147 050: 0c800-0cbff - PID: 05 (idx 001, nxt: 051)
148 051: 0cc00-0cfff - PID: 05 (idx 002, nxt: 052)
149 052: 0d000-0d3ff - PID: 05 (idx 003, nxt: 053)
150 053: 0d400-0d7ff - PID: 05 (idx 004, nxt: -01)
```

- Biểu đồ Gantt cho định thời CPU



Hình 4: Biểu đồ Gantt của testcase os_1

3.2.3 Kiểm thử testcase os_2

- Testcase os_2

```

1 4 3 7
2 1 s0
3 2 p0
4 4 m0
5 8 s1
6 16 m1
7 32 p1
8 34 s3

```

- Kết quả kiểm thử

```

1 ./os os_2
2 Time slot 0
3 Time slot 1
4   Loaded a process at input/proc/s0, PID: 1
5 Time slot 2
6   CPU 2: Dispatched process 1
7   Loaded a process at input/proc/p0, PID: 2
8 Time slot 3
9   CPU 1: Dispatched process 2
10 Time slot 4
11   Loaded a process at input/proc/m0, PID: 3
12 Time slot 5
13   CPU 0: Dispatched process 3
14 Time slot 6
15   CPU 2: Put process 1 to run queue
16   CPU 2: Dispatched process 1
17 Time slot 7
18   CPU 1: Put process 2 to run queue
19   CPU 1: Dispatched process 2
20 Time slot 8
21   Loaded a process at input/proc/s1, PID: 4
22 Time slot 9
23   CPU 0: Put process 3 to run queue
24   CPU 0: Dispatched process 4
25 Time slot 10
26   CPU 2: Put process 1 to run queue
27   CPU 2: Dispatched process 3
28 Time slot 11
29   CPU 1: Put process 2 to run queue
30   CPU 1: Dispatched process 1
31 Time slot 12
32 Time slot 13
33   CPU 2: Processed 3 has finished
34   CPU 2: Dispatched process 2
35   CPU 0: Put process 4 to run queue
36   CPU 0: Dispatched process 4
37 Time slot 14
38 Time slot 15
39   CPU 2: Processed 2 has finished
40   CPU 1: Put process 1 to run queue
41   CPU 1: Dispatched process 1
42 Time slot 16
43   CPU 0: Processed 4 has finished
44   Loaded a process at input/proc/m1, PID: 5

```

```
45 Time slot 17
46 CPU 2: Dispatched process 5
47 Time slot 18
48 CPU 1: Processed 1 has finished
49 Time slot 19
50 Time slot 20
51 Time slot 21
52 CPU 2: Put process 5 to run queue
53 CPU 2: Dispatched process 5
54 Time slot 22
55 Time slot 23
56 Time slot 24
57 Time slot 25
58 CPU 2: Processed 5 has finished
59 Time slot 26
60 Time slot 27
61 Time slot 28
62 Time slot 29
63 Time slot 30
64 Time slot 31
65 Loaded a process at input/proc/p1, PID: 6
66 CPU 2: Dispatched process 6
67 Time slot 32
68 Time slot 33
69 Loaded a process at input/proc/s3, PID: 7
70 CPU 1: Dispatched process 7
71 Time slot 34
72 CPU 0 stopped
73 Time slot 35
74 CPU 2: Put process 6 to run queue
75 CPU 2: Dispatched process 6
76 Time slot 36
77 Time slot 37
78 CPU 1: Put process 7 to run queue
79 CPU 1: Dispatched process 7
80 Time slot 38
81 Time slot 39
82 CPU 2: Put process 6 to run queue
83 CPU 2: Dispatched process 6
84 Time slot 40
85 Time slot 41
86 CPU 2: Processed 6 has finished
87 CPU 2 stopped
88 CPU 1: Put process 7 to run queue
89 CPU 1: Dispatched process 7
90 Time slot 42
91 Time slot 43
92 Time slot 44
93 CPU 1: Processed 7 has finished
94 CPU 1 stopped
95
96 MEMORY CONTENT:
97 000: 00000-003ff - PID: 03 (idx 000, nxt: 001)
98 00014: 66
99 001: 00400-007ff - PID: 03 (idx 001, nxt: -01)
100 002: 00800-00bff - PID: 06 (idx 000, nxt: 003)
101 003: 00c00-00fff - PID: 06 (idx 001, nxt: 004)
102 004: 01000-013ff - PID: 06 (idx 002, nxt: 005)
103 011e7: 0a
104 005: 01400-017ff - PID: 06 (idx 003, nxt: 006)
105 006: 01800-01bff - PID: 06 (idx 004, nxt: -01)
106 007: 01c00-01fff - PID: 02 (idx 000, nxt: -01)
107 01c14: 64
108 008: 02000-023ff - PID: 03 (idx 000, nxt: 009)
109 023e8: 15
110 009: 02400-027ff - PID: 03 (idx 001, nxt: -01)
111 010: 02800-02bff - PID: 03 (idx 000, nxt: 011)
112 011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
113 012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
114 013: 03400-037ff - PID: 03 (idx 003, nxt: 014)
115 014: 03800-03bff - PID: 03 (idx 004, nxt: -01)
```



```
116 019: 04c00-04fff - PID: 06 (idx 000, nxt: 020)
117 020: 05000-053ff - PID: 06 (idx 001, nxt: 021)
118 021: 05400-057ff - PID: 06 (idx 002, nxt: 022)
119 022: 05800-05bff - PID: 06 (idx 003, nxt: -01)
```

3.2.4 Kiểm thử testcase os_3

- Testcase os_3

```
1 1 5 6
2 0 p1
3 3 s1
4 6 m1
5 9 s2
6 12 m0
7 15 p0
```

- Kết quả kiểm thử

```
1 ./os os_3
2   Loaded a process at input/proc/p1, PID: 1
3   Time slot   0
4   Time slot   1
5     CPU 4: Dispatched process 1
6   Time slot   2
7     CPU 4: Put process 1 to run queue
8     CPU 4: Dispatched process 1
9   Time slot   3
10    CPU 4: Put process 1 to run queue
11    CPU 4: Dispatched process 1
12    Loaded a process at input/proc/s1, PID: 2
13   Time slot   4
14    CPU 3: Dispatched process 2
15    CPU 4: Put process 1 to run queue
16    CPU 4: Dispatched process 1
17   Time slot   5
18    CPU 3: Put process 2 to run queue
19    CPU 3: Dispatched process 2
20    CPU 4: Put process 1 to run queue
21    CPU 4: Dispatched process 1
22   Time slot   6
23    CPU 3: Put process 2 to run queue
24    CPU 3: Dispatched process 2
25    CPU 4: Put process 1 to run queue
26    CPU 4: Dispatched process 1
27    Loaded a process at input/proc/m1, PID: 3
28   Time slot   7
29    CPU 2: Dispatched process 3
30    CPU 3: Put process 2 to run queue
31    CPU 3: Dispatched process 2
32    CPU 4: Put process 1 to run queue
33    CPU 4: Dispatched process 1
34   Time slot   8
35    CPU 2: Put process 3 to run queue
36    CPU 2: Dispatched process 3
37    CPU 3: Put process 2 to run queue
38    CPU 3: Dispatched process 2
39    CPU 4: Put process 1 to run queue
40    CPU 4: Dispatched process 1
41   Time slot   9
42    CPU 2: Put process 3 to run queue
43    CPU 2: Dispatched process 3
44    CPU 3: Put process 2 to run queue
45    CPU 3: Dispatched process 2
46    CPU 4: Put process 1 to run queue
47    CPU 4: Dispatched process 1
48    Loaded a process at input/proc/s2, PID: 4
49   Time slot  10
50    CPU 1: Dispatched process 4
51    CPU 2: Put process 3 to run queue
```

```
52 CPU 2: Dispatched process 3
53 CPU 3: Put process 2 to run queue
54 CPU 3: Dispatched process 2
55 CPU 4: Put process 1 to run queue
56 CPU 4: Dispatched process 1
57 Time slot 11
58 CPU 1: Put process 4 to run queue
59 CPU 1: Dispatched process 4
60 CPU 2: Put process 3 to run queue
61 CPU 2: Dispatched process 3
62 CPU 3: Processed 2 has finished
63 CPU 4: Processed 1 has finished
64 Time slot 12
65 CPU 1: Put process 4 to run queue
66 CPU 1: Dispatched process 4
67 CPU 2: Put process 3 to run queue
68 CPU 2: Dispatched process 3
69 Loaded a process at input/proc/m0, PID: 5
70 Time slot 13
71 CPU 1: Put process 4 to run queue
72 CPU 1: Dispatched process 4
73 CPU 0: Dispatched process 5
74 CPU 2: Put process 3 to run queue
75 CPU 2: Dispatched process 3
76 Time slot 14
77 CPU 1: Put process 4 to run queue
78 CPU 1: Dispatched process 4
79 CPU 0: Put process 5 to run queue
80 CPU 0: Dispatched process 5
81 CPU 2: Put process 3 to run queue
82 CPU 2: Dispatched process 3
83 Time slot 15
84 CPU 1: Put process 4 to run queue
85 CPU 1: Dispatched process 4
86 CPU 0: Put process 5 to run queue
87 CPU 0: Dispatched process 5
88 CPU 2: Processed 3 has finished
89 Loaded a process at input/proc/p0, PID: 6
90 Time slot 16
91 CPU 1: Put process 4 to run queue
92 CPU 1: Dispatched process 4
93 CPU 0: Put process 5 to run queue
94 CPU 0: Dispatched process 5
95 CPU 2: Dispatched process 6
96 Time slot 17
97 CPU 1: Put process 4 to run queue
98 CPU 1: Dispatched process 4
99 CPU 0: Put process 5 to run queue
100 CPU 0: Dispatched process 5
101 CPU 2: Put process 6 to run queue
102 CPU 2: Dispatched process 6
103 CPU 3 stopped
104 CPU 4 stopped
105 Time slot 18
106 CPU 1: Put process 4 to run queue
107 CPU 1: Dispatched process 4
108 CPU 0: Put process 5 to run queue
109 CPU 0: Dispatched process 5
110 CPU 2: Put process 6 to run queue
111 CPU 2: Dispatched process 6
112 Time slot 19
113 CPU 1: Put process 4 to run queue
114 CPU 1: Dispatched process 4
115 CPU 0: Put process 5 to run queue
116 CPU 0: Dispatched process 5
117 CPU 2: Put process 6 to run queue
118 CPU 2: Dispatched process 6
119 Time slot 20
120 CPU 1: Put process 4 to run queue
121 CPU 1: Dispatched process 4
122 CPU 0: Processed 5 has finished
```

```
123 CPU 0 stopped
124 CPU 2: Put process 6 to run queue
125 CPU 2: Dispatched process 6
126 Time slot 21
127 CPU 1: Put process 4 to run queue
128 CPU 1: Dispatched process 4
129 CPU 2: Put process 6 to run queue
130 CPU 2: Dispatched process 6
131 Time slot 22
132 CPU 1: Processed 4 has finished
133 CPU 1 stopped
134 CPU 2: Put process 6 to run queue
135 CPU 2: Dispatched process 6
136 Time slot 23
137 CPU 2: Put process 6 to run queue
138 CPU 2: Dispatched process 6
139 Time slot 24
140 CPU 2: Put process 6 to run queue
141 CPU 2: Dispatched process 6
142 Time slot 25
143 CPU 2: Put process 6 to run queue
144 CPU 2: Dispatched process 6
145 Time slot 26
146 CPU 2: Processed 6 has finished
147 CPU 2 stopped
148
149 MEMORY CONTENT:
150 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
151 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
152 002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
153 009e7: 0a
154 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
155 004: 01000-013ff - PID: 01 (idx 004, nxt: -01)
156 005: 01400-017ff - PID: 05 (idx 000, nxt: 006)
157 017e8: 15
158 006: 01800-01bff - PID: 05 (idx 001, nxt: -01)
159 007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
160 01c14: 66
161 008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
162 009: 02400-027ff - PID: 05 (idx 000, nxt: 010)
163 010: 02800-02bff - PID: 05 (idx 001, nxt: 011)
164 011: 02c00-02fff - PID: 05 (idx 002, nxt: 012)
165 012: 03000-033ff - PID: 05 (idx 003, nxt: 013)
166 013: 03400-037ff - PID: 05 (idx 004, nxt: -01)
167 014: 03800-03bff - PID: 06 (idx 000, nxt: -01)
168 03814: 64
169 025: 06400-067ff - PID: 01 (idx 000, nxt: 026)
170 026: 06800-06bff - PID: 01 (idx 001, nxt: 027)
171 027: 06c00-06fff - PID: 01 (idx 002, nxt: 028)
172 028: 07000-073ff - PID: 01 (idx 003, nxt: -01)
```