VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE & ENGINEERING

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

Ho Chi Minh City, 11/2021

Mục lục

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

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

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

```
struct pcb_t * dequeue(struct queue_t * q)
{
    /* TODO: return a pcb whose prioprity is the highest
   * in the queue [q] and remember to remove it from q
   * */
    if (q->size == 0)
    {
        printf("Queue is empty");
    return NULL;
  }

    int i = 0, j;
  for (j = 1; j < q->size; j++) {
    if (q->proc[j]->priority < q->proc[i]->priority)
      i = j;
  }
  struct pcb_t * res = q->proc[i];
  for (j = i+1; j < q->size; j++)
    q->proc[j-1] = q->proc[j];
  q->size--;
  return res;
}
```

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

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

```
    * [ready_queue] and return the highest priority one.
    * Remember to use lock to protect the queue.
    * */
     pthread_mutex_lock(&queue_lock);
   if (empty(&ready_queue)) {
   // move all process is waiting in run_queue back to ready_queue
     while (!empty(&run_queue)) {
       enqueue(&ready_queue, dequeue(&run_queue));
     }
   }

   if (!empty(&ready_queue)) {
     proc = dequeue(&ready_queue);
   }
   pthread_mutex_unlock(&queue_lock);

   return proc;
}
```

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

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

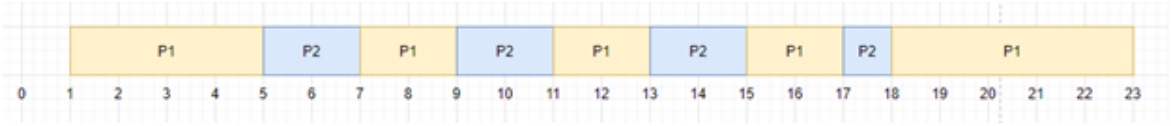
- Kết quả kiểm thử

```
./os sched_0
Time slot   0
  Loaded a process at input/proc/s0, PID: 1
Time slot   1
  CPU 0: Dispatched process  1
Time slot   2
Time slot   3
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  1
Time slot   4
  Loaded a process at input/proc/s1, PID: 2
Time slot   5
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   6
Time slot   7
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot   8
Time slot   9
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot  10
Time slot  11
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  12
Time slot  13
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot  14
Time slot  15
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  16
Time slot  17
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
```

```
Time slot   18
  CPU 0: Processed  2 has finished
  CPU 0: Dispatched process  1
Time slot   19
Time slot   20
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  1
Time slot   21
Time slot   22
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  1
Time slot   23
  CPU 0: Processed  1 has finished
  CPU 0 stopped

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

```
2 1 4
0 s0
4 s1
6 s2
7 s3
```

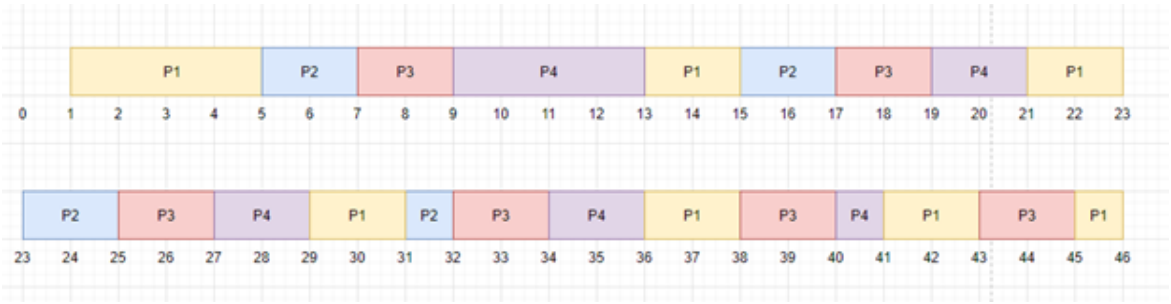
- Kết quả kiểm thử

```
./os sched_1
    Loaded a process at input/proc/s0, PID: 1
Time slot   0
Time slot   1
  CPU 0: Dispatched process  1
Time slot   2
Time slot   3
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  1
Time slot   4
   Loaded a process at input/proc/s1, PID: 2
Time slot   5
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   6
   Loaded a process at input/proc/s2, PID: 3
Time slot   7
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  3
   Loaded a process at input/proc/s3, PID: 4
Time slot   8
Time slot   9
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot   10
Time slot   11
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  4
Time slot   12
Time slot   13
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  1
```

```
Time slot   14
Time slot   15
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   16
Time slot   17
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  3
Time slot   18
Time slot   19
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot   20
Time slot   21
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  1
Time slot   22
Time slot   23
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   24
Time slot   25
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  3
Time slot   26
Time slot   27
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot   28
Time slot   29
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  1
Time slot   30
Time slot   31
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   32
  CPU 0: Processed  2 has finished
  CPU 0: Dispatched process  3
Time slot   33
Time slot   34
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot   35
Time slot   36
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  1
Time slot   37
Time slot   38
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  3
Time slot   39
Time slot   40
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot   41
  CPU 0: Processed  4 has finished
  CPU 0: Dispatched process  1
Time slot   42
Time slot   43
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  3
Time slot   44
Time slot   45
  CPU 0: Processed  3 has finished
  CPU 0: Dispatched process  1
Time slot   46
  CPU 0: Processed  1 has finished
  CPU 0 stopped

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 schede_2

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

- Kết quả kiểm thử

```
./os sched_2
Time slot   0
Time slot   1
  Loaded a process at input/proc/s1, PID: 1
Time slot   2
  CPU 0: Dispatched process  1
Time slot   3
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  1
  Loaded a process at input/proc/s0, PID: 2
Time slot   4
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  2
Time slot   5
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
  Loaded a process at input/proc/s3, PID: 3
Time slot   6
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  3
Time slot   7
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  1
  Loaded a process at input/proc/s2, PID: 4
Time slot   8
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  4
Time slot   9
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  10
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  11
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  12
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  4
Time slot  13
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  14
```

```
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  15
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  16
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  4
Time slot  17
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  18
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  19
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  20
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  4
Time slot  21
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  22
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  23
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  1
Time slot  24
  CPU 0: Processed  1 has finished
  CPU 0: Dispatched process  4
Time slot  25
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  26
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  27
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot  28
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  29
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  30
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot  31
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  32
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  33
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot  34
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot  35
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot  36
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot  37
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
```

```
Time slot   38
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  2
Time slot   39
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot   40
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  3
Time slot   41
  CPU 0: Processed  3 has finished
  CPU 0: Dispatched process  2
Time slot   42
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot   43
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  2
Time slot   44
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot   45
  CPU 0: Processed  4 has finished
  CPU 0: Dispatched process  2
Time slot   46
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot   47
  CPU 0: Processed  2 has finished
  CPU 0 stopped

MEMORY CONTENT:
```

1.2.4 Kiểm thử testcase sched_3

- Testcase schede_3

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

- Kết quả kiểm thử

```
./os sched_3
  Loaded a process at input/proc/s1, PID: 1
Time slot   0
Time slot   1
  CPU 1: Dispatched process  1
Time slot   2
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
Time slot   3
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
Time slot   4
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
Time slot   5
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
  Loaded a process at input/proc/s0, PID: 2
Time slot   6
  CPU 0: Dispatched process  2
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
Time slot   7
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
  CPU 1: Put process  1 to run queue
```

```
  CPU 1: Dispatched process  1
Time slot   8
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
  CPU 1: Processed  1 has finished
Time slot   9
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  10
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  11
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  12
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  13
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  14
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  15
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
  Loaded a process at input/proc/s2, PID: 3
Time slot  16
  CPU 1: Dispatched process  3
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  17
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  18
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  19
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  20
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  2
Time slot  21
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
  CPU 0: Processed  2 has finished
  CPU 0 stopped
Time slot  22
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
Time slot  23
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
Time slot  24
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
Time slot  25
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
Time slot  26
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
```

```
Time slot   27
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  3
Time slot   28
  CPU 1: Processed  3 has finished
  CPU 1 stopped

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

```
static struct page_table_t *get_page_table(addr_t index, struct seg_table_t *
    seg_table)
{
  int i;
  for (i = 0; i < seg_table->size; i++)
  {
    //TODO
    if (index == seg_table->table[i].v_index)
    {
      return seg_table->table[i].pages;
    }
  }
  return NULL;
}
```

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

```
static int translate(addr_t virtual_addr, addr_t *physical_addr, struct pcb_t *proc)
{
  addr_t offset = get_offset(virtual_addr);
  addr_t first_lv = get_first_lv(virtual_addr);
  addr_t second_lv = get_second_lv(virtual_addr);

  struct page_table_t *page_table = NULL;
  page_table = get_page_table(first_lv, proc->seg_table);
  if (page_table == NULL)
  {
    return 0;
  }
  int i;
  for (i = 0; i < 1 << PAGE_LEN; i++)
  {
    if (page_table->table[i].v_index == second_lv)
    {
      //TODO
      *physical_addr = page_table->table[i].p_index * PAGE_SIZE + offset;
      return 1;
    }
  }
  return 0;
}
```

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 breakpointer 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 càu

- Code

```
addr_t alloc_mem(uint32_t size, struct pcb_t *proc)
{
  pthread_mutex_lock(&mem_lock);
  addr_t ret_mem = 0;

  uint32_t num_pages = ((size % PAGE_SIZE) == 0) ? size / PAGE_SIZE : size /
    PAGE_SIZE + 1; // Number of pages we will use
  int mem_avail = 0; // We could allocate new memory region or not?

  //TODO
  //Kiem tra page trong trong vung nho ao va vung nho vat ly
  int i;
  int num_avail_pages = 0;
  for (i = 0; i < NUM_PAGES; i++)
  { //Check if ram memory space is avaiable
    if (_mem_stat[i].proc == 0)
    {
      num_avail_pages++;
      if (num_avail_pages == num_pages && proc->bp + num_pages * PAGE_SIZE <=
    RAM_SIZE)
      {
        mem_avail = 1;
```

```
        break;
      }
    }
  }

  if (mem_avail)
  {
    ret_mem = proc->bp;
    proc->bp += num_pages * PAGE_SIZE;

    //TODO
    //Cap nhat breakpointer, _mem_stat, segment table va page table
    int num_alloc_pages = 0;
    int pre_index; // index of the previous page in the list
    addr_t cur_vir_addr;
    int seg_idx, page_idx;
    for (i = 0; i < NUM_PAGES; i++)
    {
      if (_mem_stat[i].proc == 0)
      {
        _mem_stat[i].proc = proc->pid;
        _mem_stat[i].index = num_alloc_pages;

        if (_mem_stat[i].index != 0)
          _mem_stat[pre_index].next = i;
        pre_index = i;

        int found = 0;
        struct seg_table_t *seg_table = proc->seg_table;
        if (seg_table->table[0].pages == NULL)
          seg_table->size = 0;

        cur_vir_addr = ret_mem + (num_alloc_pages << OFFSET_LEN);

        seg_idx = get_first_lv(cur_vir_addr);
        page_idx = get_second_lv(cur_vir_addr);
        int j;
        for (j = 0; j < seg_table->size; j++)
        {
          if (seg_table->table[j].v_index == seg_idx)
          {
            struct page_table_t *cur_page_table = seg_table->table[j].pages;

            cur_page_table->table[cur_page_table->size].v_index = page_idx;
            cur_page_table->table[cur_page_table->size].p_index = i;

            cur_page_table->size++;

            found = 1;
            break;
          }
        }

        if (!found)
        { //If not found, add new row into table
          seg_table->table[seg_table->size].v_index = seg_idx;
          seg_table->table[seg_table->size].pages = (struct page_table_t *)malloc(
    sizeof(struct page_table_t));

          seg_table->table[seg_table->size].pages->table[0].v_index = page_idx;
          seg_table->table[seg_table->size].pages->table[0].p_index = i;

          seg_table->table[seg_table->size].pages->size = 1;

          seg_table->size++;
        }

        num_alloc_pages++;
        if (num_alloc_pages == num_pages)
        {
          _mem_stat[i].next = -1;
```

```
            break;
          }
        }
      }
    }

    pthread_mutex_unlock(&mem_lock);
    return ret_mem;
}
```

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 trỏ 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

```
int free_mem(addr_t address, struct pcb_t *proc)
{
  pthread_mutex_lock(&mem_lock);

    //TODO
  struct page_table_t *page_table = get_page_table(get_first_lv(address), proc->
    seg_table);

  int valid = 0;
  if (page_table != NULL)
  {
    int i;
    for (i = 0; i < page_table->size; i++)
    {
      if (page_table->table[i].v_index == get_second_lv(address))
      {
        addr_t physical_addr;
        if (translate(address, &physical_addr, proc))
        {
          int p_index = physical_addr >> OFFSET_LEN;
          int num_free_pages = 0;
          addr_t cur_vir_addr = (num_free_pages << OFFSET_LEN) + address;
          addr_t seg_idx, page_idx;
          do
          {
            _mem_stat[p_index].proc = 0;
            int found = 0;
            int k;
            seg_idx = get_first_lv(cur_vir_addr);
            page_idx = get_second_lv(cur_vir_addr);
            for (k = 0; k < proc->seg_table->size && !found; k++)
            {
              if (proc->seg_table->table[k].v_index == seg_idx)
              {
                int l;
                for (l = 0; l < proc->seg_table->table[k].pages->size; l++)
                {
                  if (proc->seg_table->table[k].pages->table[l].v_index == page_idx)
                  {
                    int m;
                    for (m = l; m < proc->seg_table->table[k].pages->size - 1; m++)
  //Rearrange page table
```

```
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 7
alloc 13535 0
alloc 1568 1
free 0
alloc 1386 2
alloc 4564 4
write 102 1 20
write 21 2 1000
```

- Kết quả kiểm thử

```
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
  003e8: 15
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
  03814: 66
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 8
alloc 13535 0
alloc 1568 1
free 0
alloc 1386 2
alloc 4564 4
free 2
free 4
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 9
alloc 9000 0
alloc 5000 1
alloc 1500 2
alloc 1600 3
write 99 2 100
write 103 0 10
write 10 3 500
alloc 2500 4
write 1000 4 1
```

- Kết quả kiểm thử

```
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
  0000a: 67
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: -01)
009: 02400-027ff - PID: 01 (idx 000, nxt: 010)
010: 02800-02bff - PID: 01 (idx 001, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 002, nxt: 012)
012: 03000-033ff - PID: 01 (idx 003, nxt: 013)
013: 03400-037ff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
  03864: 63
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
016: 04000-043ff - PID: 01 (idx 000, nxt: 017)
  041f4: 0a
017: 04400-047ff - PID: 01 (idx 001, nxt: -01)
018: 04800-04bff - PID: 01 (idx 000, nxt: 019)
  04801: ffffffe8
019: 04c00-04fff - PID: 01 (idx 001, nxt: 020)
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 7
alloc 12345 0
alloc 500 1
alloc 1234 2
write 321 0 599
write 765 1 544
free 2
write 42 0 123
```

- Kết quả kiểm thử

```
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
  0007b: 2a
  00257: 41
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: -01)
013: 03400-037ff - PID: 01 (idx 000, nxt: -01)
  03620: fffffffd
```

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 ngoại.
      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 liền kề 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 cấp 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 sủ dụng các biến trên là alloc_mem(), free|mem(), read_nem(), 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

```
6 2 4
0 p0
2 p1
```

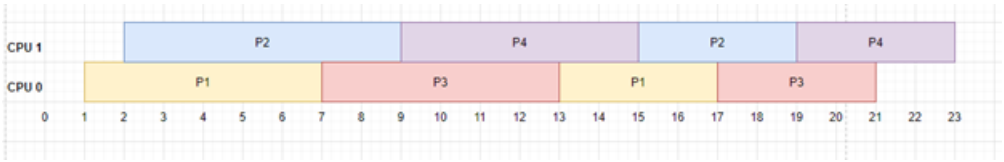
- Kết quả kiểm thử

```
./os os_0
Time slot   0
  Loaded a process at input/proc/p0, PID: 1
Time slot   1
  CPU 1: Dispatched process  1
Time slot   2
  Loaded a process at input/proc/p1, PID: 2
Time slot   3
  CPU 1: Dispatched process  2
  Loaded a process at input/proc/p1, PID: 3
Time slot   4
  Loaded a process at input/proc/p1, PID: 4
Time slot   5
Time slot   6
Time slot   7
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  3
Time slot   8
Time slot   9
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  4
Time slot  10
Time slot  11
Time slot  12
Time slot  13
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  1
Time slot  14
Time slot  15
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  2
Time slot  16
Time slot  17
  CPU 0: Processed  1 has finished
  CPU 0: Dispatched process  3
Time slot  18
Time slot  19
  CPU 1: Processed  2 has finished
  CPU 1: Dispatched process  4
Time slot  20
Time slot  21
  CPU 0: Processed  3 has finished
  CPU 0 stopped
Time slot  22
Time slot  23
  CPU 1: Processed  4 has finished
  CPU 1 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 02 (idx 000, nxt: 001)
001: 00400-007ff - PID: 02 (idx 001, nxt: 007)
002: 00800-00bff - PID: 02 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 02 (idx 001, nxt: 004)
004: 01000-013ff - PID: 02 (idx 002, nxt: 005)
```

```
005: 01400-017ff - PID: 02 (idx 003, nxt: -01)
006: 01800-01bff - PID: 03 (idx 000, nxt: 011)
007: 01c00-01fff - PID: 02 (idx 002, nxt: 008)
  01de7: 0a
008: 02000-023ff - PID: 02 (idx 003, nxt: 009)
009: 02400-027ff - PID: 02 (idx 004, nxt: -01)
010: 02800-02bff - PID: 01 (idx 000, nxt: -01)
  02814: 64
011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
013: 03400-037ff - PID: 03 (idx 003, nxt: -01)
014: 03800-03bff - PID: 04 (idx 000, nxt: 025)
015: 03c00-03fff - PID: 03 (idx 000, nxt: 016)
016: 04000-043ff - PID: 03 (idx 001, nxt: 017)
017: 04400-047ff - PID: 03 (idx 002, nxt: 018)
  045e7: 0a
018: 04800-04bff - PID: 03 (idx 003, nxt: 019)
019: 04c00-04fff - PID: 03 (idx 004, nxt: -01)
020: 05000-053ff - PID: 04 (idx 000, nxt: 021)
021: 05400-057ff - PID: 04 (idx 001, nxt: 022)
022: 05800-05bff - PID: 04 (idx 002, nxt: 023)
  059e7: 0a
023: 05c00-05fff - PID: 04 (idx 003, nxt: 024)
024: 06000-063ff - PID: 04 (idx 004, nxt: -01)
025: 06400-067ff - PID: 04 (idx 001, nxt: 026)
026: 06800-06bff - PID: 04 (idx 002, nxt: 027)
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

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

- Kết quả kiểm thử

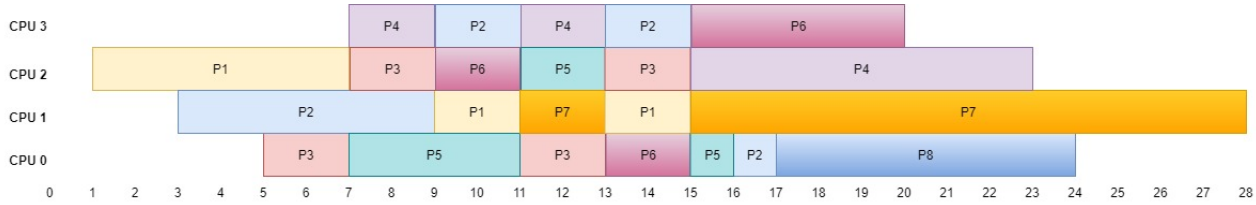
```
./os os_1
Time slot   0
Time slot   1
  Loaded a process at input/proc/p0, PID: 1
  CPU 2: Dispatched process  1
Time slot   2
  Loaded a process at input/proc/s3, PID: 2
Time slot   3
  CPU 1: Dispatched process  2
  CPU 2: Put process  1 to run queue
  CPU 2: Dispatched process  1
Time slot   4
  Loaded a process at input/proc/m1, PID: 3
Time slot   5
  CPU 0: Dispatched process  3
```

```
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  2
  CPU 2: Put process  1 to run queue
  CPU 2: Dispatched process  1
Time slot   6
   Loaded a process at input/proc/s2, PID: 4
Time slot   7
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  2
  CPU 2: Put process  1 to run queue
   Loaded a process at input/proc/m0, PID: 5
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Dispatched process  3
  CPU 3: Dispatched process  4
Time slot   8
Time slot   9
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  1
  CPU 0: Put process  5 to run queue
   Loaded a process at input/proc/p1, PID: 6
  CPU 0: Put process  3 to run queue
  CPU 2: Dispatched process  6
  CPU 2: Put process  4 to run queue
  CPU 3: Dispatched process  2
Time slot  10
Time slot  11
  CPU 2: Put process  6 to run queue
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  3
   Loaded a process at input/proc/s0, PID: 7
  CPU 2: Dispatched process  5
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  7
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  4
Time slot  12
Time slot  13
  CPU 2: Put process  5 to run queue
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  6
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  1
  CPU 2: Dispatched process  3
  CPU 3: Processed  4 has finished
  CPU 3: Dispatched process  2
Time slot  14
Time slot  15
  CPU 0: Put process  6 to run queue
  CPU 0: Dispatched process  5
  CPU 1: Processed  1 has finished
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  6
  CPU 1: Dispatched process  7
  CPU 2: Processed  3 has finished
  CPU 2: Dispatched process  4
Time slot  16
  CPU 0: Processed  5 has finished
  CPU 0: Dispatched process  2
   Loaded a process at input/proc/s1, PID: 8
Time slot  17
  CPU 0: Processed  2 has finished
  CPU 0: Dispatched process  8
  CPU 3: Put process  6 to run queue
  CPU 3: Dispatched process  6
  CPU 2: Put process  4 to run queue
  CPU 2: Dispatched process  4
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
Time slot  18
Time slot  19
```

```
  CPU 0: Put process  8 to run queue
  CPU 0: Dispatched process  8
  CPU 1: Put process  7 to run queue
  CPU 3: Put process  6 to run queue
  CPU 3: Dispatched process  6
  CPU 2: Put process  4 to run queue
  CPU 2: Dispatched process  4
  CPU 1: Dispatched process  7
Time slot  20
  CPU 3: Processed  6 has finished
  CPU 3 stopped
Time slot  21
  CPU 0: Put process  8 to run queue
  CPU 0: Dispatched process  8
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
  CPU 2: Put process  4 to run queue
  CPU 2: Dispatched process  4
Time slot  22
Time slot  23
  CPU 0: Put process  8 to run queue
  CPU 0: Dispatched process  8
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
  CPU 2: Processed  4 has finished
  CPU 2 stopped
Time slot  24
  CPU 0: Processed  8 has finished
  CPU 0 stopped
Time slot  25
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
Time slot  26
Time slot  27
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
Time slot 28
  CPU 1: Processed  7 has finished
  CPU 1 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 06 (idx 000, nxt: 001)
001: 00400-007ff - PID: 06 (idx 001, nxt: 031)
006: 01800-01bff - PID: 06 (idx 000, nxt: 009)
007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
  01fe8: 15
008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
009: 02400-027ff - PID: 06 (idx 001, nxt: 010)
010: 02800-02bff - PID: 06 (idx 002, nxt: 011)
011: 02c00-02fff - PID: 06 (idx 003, nxt: -01)
021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
  05414: 64
024: 06000-063ff - PID: 05 (idx 000, nxt: 025)
  06014: 66
025: 06400-067ff - PID: 05 (idx 001, nxt: -01)
031: 07c00-07fff - PID: 06 (idx 002, nxt: 032)
  07de7: 0a
032: 08000-083ff - PID: 06 (idx 003, nxt: 033)
033: 08400-087ff - PID: 06 (idx 004, nxt: -01)
049: 0c400-0c7ff - PID: 05 (idx 000, nxt: 050)
050: 0c800-0cbff - PID: 05 (idx 001, nxt: 051)
051: 0cc00-0cfff - PID: 05 (idx 002, nxt: 052)
052: 0d000-0d3ff - PID: 05 (idx 003, nxt: 053)
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

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

- Kết quả kiểm thử

```
./os os_2
Time slot   0
Time slot   1
  Loaded a process at input/proc/s0, PID: 1
Time slot   2
  CPU 2: Dispatched process  1
  Loaded a process at input/proc/p0, PID: 2
Time slot   3
  CPU 1: Dispatched process  2
Time slot   4
  Loaded a process at input/proc/m0, PID: 3
Time slot   5
  CPU 0: Dispatched process  3
Time slot   6
  CPU 2: Put process  1 to run queue
  CPU 2: Dispatched process  1
Time slot   7
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  2
Time slot   8
  Loaded a process at input/proc/s1, PID: 4
Time slot   9
  CPU 0: Put process  3 to run queue
  CPU 0: Dispatched process  4
Time slot  10
  CPU 2: Put process  1 to run queue
  CPU 2: Dispatched process  3
Time slot  11
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  1
Time slot  12
Time slot  13
  CPU 2: Processed  3 has finished
  CPU 2: Dispatched process  2
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  4
Time slot  14
Time slot  15
  CPU 2: Processed  2 has finished
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  1
Time slot  16
  CPU 0: Processed  4 has finished
  Loaded a process at input/proc/m1, PID: 5
```

```
Time slot   17
  CPU 2: Dispatched process  5
Time slot   18
  CPU 1: Processed  1 has finished
Time slot   19
Time slot   20
Time slot   21
  CPU 2: Put process  5 to run queue
  CPU 2: Dispatched process  5
Time slot   22
Time slot   23
Time slot   24
Time slot   25
  CPU 2: Processed  5 has finished
Time slot   26
Time slot   27
Time slot   28
Time slot   29
Time slot   30
Time slot   31
   Loaded a process at input/proc/p1, PID: 6
  CPU 2: Dispatched process  6
Time slot   32
Time slot   33
   Loaded a process at input/proc/s3, PID: 7
  CPU 1: Dispatched process  7
Time slot   34
  CPU 0 stopped
Time slot   35
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot   36
Time slot   37
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
Time slot   38
Time slot   39
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot   40
Time slot   41
  CPU 2: Processed  6 has finished
  CPU 2 stopped
  CPU 1: Put process  7 to run queue
  CPU 1: Dispatched process  7
Time slot   42
Time slot   43
Time slot   44
  CPU 1: Processed  7 has finished
  CPU 1 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 03 (idx 000, nxt: 001)
   00014: 66
001: 00400-007ff - PID: 03 (idx 001, nxt: -01)
002: 00800-00bff - PID: 06 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 06 (idx 001, nxt: 004)
004: 01000-013ff - PID: 06 (idx 002, nxt: 005)
   011e7: 0a
005: 01400-017ff - PID: 06 (idx 003, nxt: 006)
006: 01800-01bff - PID: 06 (idx 004, nxt: -01)
007: 01c00-01fff - PID: 02 (idx 000, nxt: -01)
   01c14: 64
008: 02000-023ff - PID: 03 (idx 000, nxt: 009)
   023e8: 15
009: 02400-027ff - PID: 03 (idx 001, nxt: -01)
010: 02800-02bff - PID: 03 (idx 000, nxt: 011)
011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
013: 03400-037ff - PID: 03 (idx 003, nxt: 014)
014: 03800-03bff - PID: 03 (idx 004, nxt: -01)
```

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

3.2.4 Kiểm thử testcase os_3

- Testcase os_3

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

- Kết quả kiểm thử

```
./os os_3
  Loaded a process at input/proc/p1, PID: 1
Time slot   0
Time slot   1
  CPU 4: Dispatched process  1
Time slot   2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot   3
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
  Loaded a process at input/proc/s1, PID: 2
Time slot   4
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot   5
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot   6
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
  Loaded a process at input/proc/m1, PID: 3
Time slot   7
  CPU 2: Dispatched process  3
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot   8
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot   9
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
  Loaded a process at input/proc/s2, PID: 4
Time slot  10
  CPU 1: Dispatched process  4
  CPU 2: Put process  3 to run queue
```

```
  CPU 2: Dispatched process  3
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  2
  CPU 4: Put process  1 to run queue
  CPU 4: Dispatched process  1
Time slot  11
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
  CPU 3: Processed  2 has finished
  CPU 4: Processed  1 has finished
Time slot  12
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
  Loaded a process at input/proc/m0, PID: 5
Time slot  13
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Dispatched process  5
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
Time slot  14
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Put process  3 to run queue
  CPU 2: Dispatched process  3
Time slot  15
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Processed  3 has finished
  Loaded a process at input/proc/p0, PID: 6
Time slot  16
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Dispatched process  6
Time slot  17
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
  CPU 3 stopped
  CPU 4 stopped
Time slot  18
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  19
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  5
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  20
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 0: Processed  5 has finished
```

```
  CPU 0 stopped
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  21
  CPU 1: Put process  4 to run queue
  CPU 1: Dispatched process  4
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  22
  CPU 1: Processed  4 has finished
  CPU 1 stopped
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  23
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  24
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  25
  CPU 2: Put process  6 to run queue
  CPU 2: Dispatched process  6
Time slot  26
  CPU 2: Processed  6 has finished
  CPU 2 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
  009e7: 0a
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: -01)
005: 01400-017ff - PID: 05 (idx 000, nxt: 006)
  017e8: 15
006: 01800-01bff - PID: 05 (idx 001, nxt: -01)
007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
  01c14: 66
008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
009: 02400-027ff - PID: 05 (idx 000, nxt: 010)
010: 02800-02bff - PID: 05 (idx 001, nxt: 011)
011: 02c00-02fff - PID: 05 (idx 002, nxt: 012)
012: 03000-033ff - PID: 05 (idx 003, nxt: 013)
013: 03400-037ff - PID: 05 (idx 004, nxt: -01)
014: 03800-03bff - PID: 06 (idx 000, nxt: -01)
  03814: 64
025: 06400-067ff - PID: 01 (idx 000, nxt: 026)
026: 06800-06bff - PID: 01 (idx 001, nxt: 027)
027: 06c00-06fff - PID: 01 (idx 002, nxt: 028)
028: 07000-073ff - PID: 01 (idx 003, nxt: -01)
```