ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Report: Assignment 2 - Operating Systems

Topic

Simple Operating System

Lecturer: Nguyễn Hoài Nam
Group 3 Students:
Nguyễn Long Kim - 1812742
Nguyễn Duy Kiên - 1812704
Nguyễn Văn Khang - 1812554
Nguyễn Anh Khoa - 1812649

TP. HỒ CHÍ MINH, THÁNG 6/2020

Task assingment

Members:	Task :	Due date:	Accomplished date:
Nguyễn Duy Kiên	• Implement functions: enqueue() and dequeue() in queue.c • Draw Gantt chart for sched_0 • Answer the question in section Scheduler	24/6/20220	23/6/2020
Nguyễn Văn Khang	• Implement functions: get_proc() in sched.c • Draw Gantt chart for sched_1 • Answer the question in section Scheduler	24/6/20220	23/6/2020
Nguyễn Anh Khoa	• Implement functions: translate() and get_page_table() in mem.c • Show status of RAM after each allocation and deallocation function call in m0 • Answer the question in section Memory Management	24/6/20220	23/6/2020
Nguyễn Long Kim	• Implement functions: alloc_mem() and free_mem() in mem.c • Show status of RAM after each allocation and deallocation function call in m1 • Synchronization for [_ram] access in mem.c • Write report	24/6/20220	23/6/2020

Contents

1 Scheduler

1.1 Question & Answer

Question: What is the advantage of using priority feedback queue in comparison with other scheduling algorithms you have learned?
Answer:

- Thời gian đợi trung bình giảm so với giải thuật First Come First Served
- CPU chạy các process theo Round-Robin "style" nên đảm bảo tính công về thời gian thực thi của các process. Không có process nào được chạy quá thời gian mà OS cho phép, từ đó dẫn đến tránh được tình trạng trì hoãn vô hạn định xảy ra ở một số giải thuật định thời khác như: Priority scheduling hay Multilevel Queue Scheduling
- Sử dụng hai hàng đợi *ready_queue* và *run_queue*, các process được chuyển qua lại giữa hai hàng đợi này đến khi các process được hoàn tất, tăng thời gian đáp ứng cho các process do các process có độ ưu tiên thấp đến sau vẫn có thể được thực thi trước các process có độ ưu tiên cao hơn (vì có thể lúc này process có độ ưu tiên cao hơn đang ở *run_queue*).

1.2 Gantt chart

Requirement: Draw Gantt diagram describing how processes are executed by the CPU

1.2.1 Test sched_0

- **sched_0 log**

```
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  2
Time slot   8
Time slot   9
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  1
Time slot  10
Time slot  11
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot  12
Time slot  13
        CPU 0: Put process  2 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
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  1
Time slot  17
Time slot  18
        CPU 0: Put process  1 to run queue
        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:
```

- **Gantt chart**

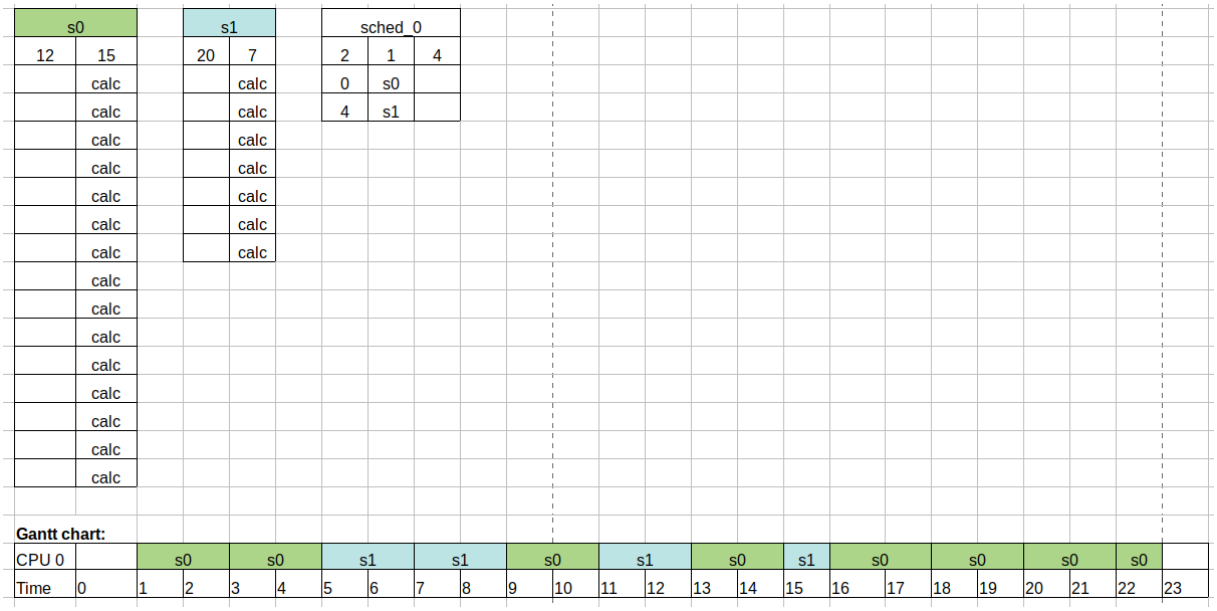

Figure 1: Gantt chart CPU thực thi các process trong sched_0

1.2.2 Test sched_1

- **sched_1 log**

```
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
         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  2
Time slot  12
Time slot  13
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot  14
Time slot  15
        CPU 0: Put process  3 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  4
Time slot  18
Time slot  19
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot  20
Time slot  21
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot  22
Time slot  23
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot  24
Time slot  25
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot  26
Time slot  27
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot  28
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  3
Time slot  29
Time slot  30
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot  31
Time slot  32
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot  33
Time slot  34
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot  35
Time slot  36
        CPU 0: Put process  3 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  4
Time slot  39
Time slot  40
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot  41
Time slot  42
        CPU 0: Processed  3 has finished
        CPU 0: Dispatched process  1
Time slot  43
Time slot  44
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot  45
```

```
96          CPU 0: Processed  4 has finished
97          CPU 0: Dispatched process  1
98  Time slot  46
99          CPU 0: Processed  1 has finished
100         CPU 0 stopped
101
102 MEMORY CONTENT:
```

- **Gantt chart**

s0		s1		s2		s3	
12	15	20	7	20	12	7	11
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc		calc		calc		calc
	calc				calc		calc
	calc				calc		calc
	calc				calc		calc
	calc				calc		calc
	calc				calc		calc
	calc						
	calc						
	calc						

sched_1		
2	1	4
0	s0	
4	s1	
6	s2	
7	s3	

Gantt chart:

CPU 0		s0		s0		s1		s2		s3		s1		s2		s0		s3		s1		s2	
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

CPU 0	s0		s3		s1	s2		s0		s3		s2		s0		s3		s2		s0		s3	s0	
Time	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	42	44	45	46

Figure 2: Gantt chart CPU thực thi các process trong sched_1

2 Memory management

2.1 Question & Answer

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

- **Advantage:**
  - Tiết kiệm bộ nhớ, sử dụng bộ nhớ hiệu quả, khắc phục việc kích thước bảng phân trang quá lớn
  - Khắc phục được phân mảnh ngoại
- **Disadvantage:**
  - Chưa khắc phục được hiện tượng phân mảnh nội

2.2 Show status of RAM

Requirement: Show the status of RAM after each memory allocation and deallocation function call

2.2.1 Test $m0$

- **Below is content of file** $m0$

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

- **Status of RAM after each allocation and deallocation function call in** $m0$

```
alloc 13535 0:
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)
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: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
================================================
alloc 1568 1:
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)
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: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
```

```
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
free 0:
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
alloc 1386 2:
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
alloc 4564 4:
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
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)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
```

2.2.2 Test $m1$

- **Below is content of file $m1$**

```
1 8
alloc 13535 0
alloc 1568 1
free 0
alloc 1386 2
alloc 4564 4
free 2
free 4
free 1
```

- **Status of RAM after each allocation and deallocation function call in $m1$**

```
alloc 13535 0:
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)
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: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
================================================
alloc 1568 1:
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)
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: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
free 0:
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
alloc 1386 2:
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
alloc 4564 4
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
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)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
free 2:
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)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
free 4:
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
================================================
free 1:
```

3 Put it all together

3.1 Synchronization

Trong file *mem.c*, hai hàm *read_mem*() và *write_mem*() truy cập vào [*_ram*] có thể xảy ra bất đồng bộ nên ta cần thêm một *mutex_lock*: *ram_lock* để giải quyết, khi đó hai hàm trên được chỉnh lại như sau:

- **Hàm read_mem()**

```
int read_mem(addr_t address, struct pcb_t * proc, BYTE * data) {
  addr_t physical_addr;
  if (translate(address, &physical_addr, proc)) {
    pthread_mutex_lock(&ram_lock); // add
    *data = _ram[physical_addr];
    pthread_mutex_unlock(&ram_lock); // add
    return 0;
  }else{
    return 1;
  }
}
```

- **Hàm write_mem()**

```
int write_mem(addr_t address, struct pcb_t * proc, BYTE data) {
```

```
  addr_t physical_addr;
  if (translate(address, &physical_addr, proc)) {
    pthread_mutex_lock(&ram_lock); // add
    _ram[physical_addr] = data;
    pthread_mutex_unlock(&ram_lock); // add
    return 0;
  }else{
    return 1;
  }
}
```

3.2 Result of simulation

- **os_0 log**

```
Time slot   0
        Loaded a process at input/proc/p0, PID: 1
Time slot   1
        CPU 0: Dispatched process  1
Time slot   2
        Loaded a process at input/proc/p1, PID: 2
Time slot   3
        CPU 1: Dispatched process  2
Time slot   4
Time slot   5
Time slot   6
Time slot   7
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   8
Time slot   9
        CPU 1: Put process  2 to run queue
        CPU 1: Dispatched process  2
Time slot  10
Time slot  11
Time slot  12
        CPU 0: Processed  1 has finished
        CPU 0 stopped
Time slot  13
        CPU 1: Processed  2 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)
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
015: 03c00-03fff - PID: 02 (idx 000, nxt: 016)
016: 04000-043ff - PID: 02 (idx 001, nxt: 017)
017: 04400-047ff - PID: 02 (idx 002, nxt: 018)
018: 04800-04bff - PID: 02 (idx 003, nxt: -01)
```

- **os_1 log**

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

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

```
        CPU 1: Dispatched process  6
        CPU 3: Dispatched process  7
        CPU 0: Processed  1 has finished
        CPU 0: Dispatched process  5
Time slot  18
        CPU 2: Put process  4 to run queue
        CPU 0: Processed  5 has finished
        CPU 0: Dispatched process  4
Time slot  19
        CPU 2: Dispatched process  8
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
Time slot  20
        CPU 1: Put process  6 to run queue
        CPU 1: Dispatched process  6
        CPU 2: Put process  8 to run queue
        CPU 0: Processed  4 has finished
        CPU 0 stopped
Time slot  21
        CPU 2: Dispatched process  8
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
        CPU 1: Put process  6 to run queue
        CPU 1: Dispatched process  6
Time slot  22
Time slot  23
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
Time slot  24
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
        CPU 1: Processed  6 has finished
        CPU 1 stopped
        CPU 2: Processed  8 has finished
        CPU 2 stopped
Time slot  25
Time slot  26
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
Time slot  27
        CPU 3: Processed  7 has finished
        CPU 3 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 05 (idx 000, nxt: 001)
        003e8: 15
001: 00400-007ff - PID: 05 (idx 001, nxt: -01)
002: 00800-00bff - PID: 05 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 05 (idx 001, nxt: 004)
004: 01000-013ff - PID: 05 (idx 002, nxt: 005)
005: 01400-017ff - PID: 05 (idx 003, nxt: 006)
006: 01800-01bff - PID: 05 (idx 004, nxt: -01)
011: 02c00-02fff - PID: 06 (idx 000, nxt: 012)
012: 03000-033ff - PID: 06 (idx 001, nxt: 013)
013: 03400-037ff - PID: 06 (idx 002, nxt: 014)
014: 03800-03bff - PID: 06 (idx 003, nxt: -01)
019: 04c00-04fff - PID: 01 (idx 000, nxt: -01)
        04c14: 64
024: 06000-063ff - PID: 05 (idx 000, nxt: 025)
        06014: 66
025: 06400-067ff - PID: 05 (idx 001, nxt: -01)
026: 06800-06bff - PID: 06 (idx 000, nxt: 027)
027: 06c00-06fff - PID: 06 (idx 001, nxt: 028)
028: 07000-073ff - PID: 06 (idx 002, nxt: 029)
        071e7: 0a
029: 07400-077ff - PID: 06 (idx 003, nxt: 030)
030: 07800-07bff - PID: 06 (idx 004, nxt: -01)
```