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

Operating System (CO2017)

Assignment

Simple Operating System

Instructor: Mr. Nguyễn Lê Duy Lai, PhD
Students: Nguyễn Viết Hòa - 2052486
Phạm Châu Thanh - 2052254
Lưu Quốc Vinh - 2052319

HO CHI MINH CITY, APRIL 2022

Contents

1 Members list & Workload

No.	Full name	Student ID	Duty in the assignment	Percentage of work
1	Nguyễn Viết Hoà	2052486	- Implement function `get_page_table()`, `translate()` and function `alloc_mem()` (in mem.c) - In charge of the presentation slides	33.3%
2	Phạm Châu Thanh	2052254	- Implement function `get_proc(void)` (in sched.c) and function `free_mem()` (in mem.c). - In charge of the majority of the report (subsection 2.2)	33.3%
3	Lưu Quốc Vinh	2052319	- Team leader. - Implement function `enqueue()` and `dequeue()` (in queue.c). - Answer two questions from the assignment specification file	33.3%

2 Scheduler

2.1 Questions

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

- No starvation: Because each process can only be executed during a time slice, the `ready_queue` will eventually be empty and all process will be executed. Therefore, there is no starvation.
- Fast response time: Each process can only be executed in a time slice. Therefore we do not need to wait too long for a certain process in the `ready_queue` to be executed even though it may have low priority.
- Priority oriented: Processes with higher priority get executed first, while ones with lower priority get executed later.

2.2 Demonstration and explanation

2.2.1 Test `sched_0`

2.2.1.1 Input

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

In this test, time slice is **2**, the number of CPU is **1** and the number of processes is **2**. The first process arrives at time **0**, has **15** tasks with priority of **12**. The second task arrives at time **4**, has **7** tasks with priority of **20**.

2.2.1.2 Output and Gantt chart

At time 0, process 1 is loaded into `ready_queue` and start running from time 1 to 2, then `ready_queue` is now empty so process 1 is pushed back to `ready_queue` from `run_queue` and run again. At time 4, process 2 is loaded into `ready_queue` and run from time 5 to 6.

Now at time 7, the `ready_queue` is once again empty so all processes in `run_queue` are pushed back to `ready_queue`. At this moment, since both processes are in `ready_queue` and process 2 has higher priority than process 1, so it is executed first. Process 2 runs from time 7 to 8, then process 1 runs from time 9 to 10. This sequence repeats until both processes have done their executions.

```
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
        Loaded a process at input/proc/s1, PID: 2
Time slot   4
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
```

(a)

```
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
```

(b)

Figure 1: Output of the execution of test `sched_0`

Time slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
PID		1		1		2		2		1		2		1		2	1		1		1		1	

Figure 2: Gantt chart of test `sched_0`

2.2.2 Test `sched_1`

2.2.2.1 Input

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

In this test, time slice is **2**, the number of CPU is **1** and the number of processes is **4**. The first process arrives at time **0**, has **15** tasks with priority of **12**. The second task arrives at time **4**, has **7** tasks with priority of **20**. The third process arrives at time **6**, has **12** tasks with priority of **20**. The fourth task arrives at time **7**, has **11** tasks with priority of **7**.

2.2.2.2 Output and Gantt chart

At time 0, process 1 is loaded into `ready_queue` and start running from time 1 to 2, then `ready_queue` is now empty so process 1 is pushed back to `ready_queue` from `run_queue` and run again. At time 4, process 2 is loaded into `ready_queue` and run from time 5 to 6. At time 6, process 3 is loaded into `ready_queue` and at time 7, process 4 is loaded into it. At this time, process 3 and process 4 are both in `ready_queue`, but process 3 runs because it has higher priority than process 4. After process 3 finishes this time slice after time 8, process 4 is executed from time 9 to time 10.

Now at time 11, the `ready_queue` is once again empty, so all processes are pushed back to `ready_queue` from `run_queue`. At this moment, since every process is in `ready_queue`, process 2 gets to be executed first since it has the highest priority among the remaining processes (priority 20, same with that of process 3 but process 2 runs first because process 2 arrives before process 4). Process 2 runs from time 11 to 12 and then go to `run_queue`. After that, with the same priority comparing procedure, process 3 runs from time 13 to 14, process 1 runs from time 15 to 16 and process 4 runs from time 17 to 18. This sequence repeats until every process finishes its execution.

```
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
        Loaded a process at input/proc/s1, PID: 2
Time slot   4
Time slot   5
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
        Loaded a process at input/proc/s2, PID: 3
Time slot   6
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
```

(a)

```
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
```

(b)

```
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
        CPU 0: Processed  4 has finished
        CPU 0: Dispatched process  1
Time slot  46
        CPU 0: Processed  1 has finished
        CPU 0 stopped
```

(c)

Figure 3: Output of the execution of test `sched_1`

Time slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
PID		1	1	2	3	4	2	3	1	4	2	3	1	4	2	3	1	4	3	1	4	3	1	4	2	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	4	1			

Figure 4: Gantt chart of test `sched_1`

3 Memory Management

3.1 Questions

What is the advantage and disadvantage of segmentation with paging?

- Advantage:
  1. The page table doesn't grow very large compared to just using one-level paging.
  2. Reduces external fragmentation.
  3. Fewer entry in the segment tables compared to two-level paging.
- Disadvantage:
  1. Still can have internal fragmentation.
  2. Extra level of paging causes delay.
  3. High level of complexity.

3.2 Demonstration and explanation

3.2.1 Test `m_0`

3.2.1.1 Input

Test `m_0` is given with the following input:

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

As it can be seen from the input text, this process has 7 tasks with the priority of 1.

3.2.1.2 Output and explanation

This program simulates an 1-megabyte RAM with 20-bit address code whose last 10 bits is the offset. Therefore, the size of the page is $2^{10} = 1024$ bytes, and the number of physical pages is $2^{20}/2^{10} = 2^{10} = 1024$ pages. The status of RAM after each task is described as follow:

1. `alloc 13535 0`: allocating $13535/2^{10} \approx 13.22 \rightarrow 14$ pages from `000` to `013`. Register 0 stores the address of the first byte of this memory region, which is `0x00000`.

```
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)
```

2. `alloc 1568 1`: allocating $1568/2^{10} = 1.53125 \rightarrow 2$ pages from `014` to `015`. Register 1 stores the address of the first byte of this memory region, which is `0x03800`.

```
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)
```

3. `free 0`: releasing the memory region whose first byte address register 0 holds. Pages that this memory region occupied (which range from `000` to `013`) are now free and can be chosen to be reallocated in next tasks.

```
014:  03800-03bff - PID: 01 (idx 000, nxt:  015)
015:  03c00-03fff - PID: 01 (idx 001, nxt:  -01)
```

4. `alloc 1386 2`: allocating $1386/2^{10} \approx 1.354 \rightarrow 2$ pages from `000` to `001` (since these two pages were released in the previous task). Register 2 stores the address of the first byte of this memory region, which is `0x00000`.

```
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)
```

5. `alloc 4564 4`: allocating $4564/2^{10} \approx 4.457 \rightarrow 5$ pages from `002` to `006` (these pages were released in task 3). Register 4 stores the address of the first byte of this memory region,

which is 0x00800.

```
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)
```

6. `write 102 1 20`: write 102_{10} (66_{16}) to address `0x03800 + 0x14 = 0x03814` (0x03800 is the address stored in register 1, 0x14 is the hexadecimal representation of 20).

```
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)

  03814:  66
015:  03c00-03fff - PID: 01 (idx 001, nxt:  -01)
```

7. `write 21 2 1000`: write 21_{10} (15_{16}) to address `0x00000 + 0x3e8 = 0x003e8` (0x00000 is the address stored in register 2, 0x3e8 is the hexadecimal representation of 1000).

```
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)
```

The RAM status after task 7 is also the final outcome of this test.

3.2.2 Test m_1

3.2.2.1 Input

Test m_1 is given with the following input:

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

As it can be seen from the input text, this process has 8 tasks with the priority of 1.

3.2.2.2 Output and explanation

The status of RAM after each task is described as follow:

1. `alloc 13535 0`: allocating $13535/2^{10} \approx 13.22 \rightarrow 14$ pages from 000 to 013. Register 0 stores the address of the first byte of this memory region, which is 0x00000.

```
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)
```

2. `alloc 1568 1`: allocating $1568/2^{10} = 1.53125 \rightarrow 2$ pages from 014 to 015. Register 1 stores the address of the first byte of this memory region, which is 0x03800.

```
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)
```

3. `free 0`: releasing the memory region whose first byte address register 0 holds. Pages that this memory region occupied (which range from `000` to `013`) are now free and can be chosen to be reallocated in next tasks.

```
014:  03800-03bff - PID: 01 (idx 000, nxt:  015)
015:  03c00-03fff - PID: 01 (idx 001, nxt:  -01)
```

4. `alloc 1386 2`: allocating $1386/2^{10} \approx 1.354 \rightarrow 2$ pages from `000` to `001` (since these two pages were released in the previous task). Register 2 stores the address of the first byte of this memory region, which is `0x00000`.

```
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)
```

5. `alloc 4564 4`: allocating $4564/2^{10} \approx 4.457 \rightarrow 5$ pages from `002` to `006` (these pages were released in task 3). Register 4 stores the address of the first byte of this memory region, which is `0x00800`.

```
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)
```

6. `free 2`: releasing the memory region whose first byte address register 2 holds. Register 2 holds the address 00000. Pages that this memory region occupied (which range from `000` to `001`) are now free and can be chosen to be reallocated in next tasks.

```
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)
```

7. `free 4`: releasing the memory region whose first byte address register 4 holds. Pages that this memory region occupied (which range from `002` to `006`) are now free and can be chosen to be reallocated in next tasks.

   ```
   014:  03800-03bff - PID: 01 (idx 000, nxt:  015)
   015:  03c00-03fff - PID: 01 (idx 001, nxt:  -01)
   ```

8. `free 1`: releasing the memory region whose first byte address register 1 holds. Pages that this memory region occupied (which range from `014` to `015`) are now free. At this moment, no pages are occupied.

After task 8, since there is no occupied pages left, the program prints nothing to the terminal.

4 Put It All Together

4.1 Test os_0

4.1.1 Input

Test os_0 is given with the following input:

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

In this test, time slice is **6** with the number of CPU is **2**. There are **4** processes:

1. The first process arrives at time **0**, has **10** tasks with priority of **1** (PID: 1).
2. The second process arrives at time **2**, has **10** tasks with priority of **1** (PID: 2).
3. The third process arrives at time **2**, has **10** tasks with priority of **1** (PID: 3).
4. The fourth process arrives at time **2**, has **10** tasks with priority of **1** (PID: 4).

4.1.2 Scheduler output, Gantt chart and memory status

```
Time slot   0
        Loaded a process at input/proc/p0, PID: 1
Time slot   1
        CPU 1: Dispatched process  1
        Loaded a process at input/proc/p1, PID: 2
Time slot   2
        CPU 0: Dispatched process  2
Time slot   3
        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
        CPU 1: Put process  1 to run queue
        CPU 1: Dispatched process  3
Time slot   7
Time slot   8
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  4
Time slot   9
Time slot  10
Time slot  11
```

(a)

```
Time slot  12
        CPU 1: Put process  3 to run queue
        CPU 1: Dispatched process  1
Time slot  13
Time slot  14
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot  15
Time slot  16
        CPU 1: Processed  1 has finished
Time slot  17
        CPU 1: Dispatched process  3
Time slot  18
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  4
Time slot  19
Time slot  20
        CPU 1: Processed  3 has finished
        CPU 1 stopped
Time slot  21
Time slot  22
        CPU 0: Processed  4 has finished
        CPU 0 stopped
```

(b)

Figure 5: Scheduler outcome of test os_0

Time slot	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			2						4						2				4				
CPU 1	1						3						1					3					

Figure 6: Gantt chart of test `os_0`

```
MEMORY CONTENT:
000: 00000-003ff - PID: 03 (idx 000, nxt: 001)
001: 00400-007ff - PID: 03 (idx 001, nxt: 002)
002: 00800-00bff - PID: 03 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 03 (idx 003, nxt: -01)
004: 01000-013ff - PID: 04 (idx 000, nxt: 005)
005: 01400-017ff - PID: 04 (idx 001, nxt: 006)
        01414: 64
006: 01800-01bff - PID: 04 (idx 002, nxt: 012)
007: 01c00-01fff - PID: 02 (idx 000, nxt: 008)
008: 02000-023ff - PID: 02 (idx 001, nxt: 009)
009: 02400-027ff - PID: 02 (idx 002, nxt: 010)
        025e7: 0a
010: 02800-02bff - PID: 02 (idx 003, nxt: 011)
011: 02c00-02fff - PID: 02 (idx 004, nxt: -01)
012: 03000-033ff - PID: 04 (idx 003, nxt: -01)
014: 03800-03bff - PID: 03 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 03 (idx 001, nxt: 016)
016: 04000-043ff - PID: 03 (idx 002, nxt: 017)
        041e7: 0a
017: 04400-047ff - PID: 03 (idx 003, nxt: 018)
018: 04800-04bff - PID: 03 (idx 004, nxt: -01)
023: 05c00-05fff - PID: 02 (idx 000, nxt: 024)
024: 06000-063ff - PID: 02 (idx 001, nxt: 025)
025: 06400-067ff - PID: 02 (idx 002, nxt: 026)
026: 06800-06bff - PID: 02 (idx 003, nxt: -01)
047: 0bc00-0bfff - PID: 01 (idx 000, nxt: -01)
057: 0e400-0e7ff - PID: 04 (idx 000, nxt: 058)
058: 0e800-0ebff - PID: 04 (idx 001, nxt: 059)
059: 0ec00-0efff - PID: 04 (idx 002, nxt: 060)
        0ede7: 0a
060: 0f000-0f3ff - PID: 04 (idx 003, nxt: 061)
061: 0f400-0f7ff - PID: 04 (idx 004, nxt: -01)
```

Figure 7: Memory status after the execution of test `os_0`

4.2 Test os_1

4.2.1 Input

Test os_1 is given with the following input:

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

In this test, time slice is **2** with the number of CPU is **4**. There are **8** processes:

1. The first process arrives at time **1**, has **10** tasks with priority of **1** (PID: 1).
2. The second process arrives at time **2**, has **11** tasks with priority of **7** (PID: 2).
3. The third process arrives at time **4**, has **8** tasks with priority of **1** (PID: 3).
4. The fourth process arrives at time **6**, has **12** tasks with priority of **20** (PID: 4).
5. The fifth process arrives at time **7**, has **7** tasks with priority of **1** (PID: 5).
6. The sixth process arrives at time **9**, has **10** tasks with priority of **1** (PID: 6).
7. The seventh process arrives at time **11**, has **15** tasks with priority of **12** (PID: 7).
8. The eighth process arrives at time **16**, has **7** tasks with priority of **20** (PID: 8).

4.2.2 Scheduler output, Gantt chart and memory status

```
Time slot   0
        Loaded a process at input/proc/p0, PID: 1
        CPU 2: Dispatched process  1
Time slot   1
Time slot   2
        Loaded a process at input/proc/s3, PID: 2
        CPU 3: Dispatched process  2
Time slot   3
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  1
        Loaded a process at input/proc/m1, PID: 3
        CPU 1: Dispatched process  3
Time slot   4
        CPU 3: Put process  2 to run queue
Time slot   5
        CPU 3: Dispatched process  2
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  1
        Loaded a process at input/proc/s2, PID: 4
        CPU 1: Put process  3 to run queue
        CPU 1: Dispatched process  3
        CPU 0: Dispatched process  4
Time slot   6
        CPU 3: Put process  2 to run queue
Time slot   7
        CPU 3: Dispatched process  2
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  1
        Loaded a process at input/proc/m0, PID: 5
        CPU 1: Put process  3 to run queue
        CPU 1: Dispatched process  5
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
```

(a)

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

(b)

```
Time slot  15
        Loaded a process at input/proc/s1, PID: 8
        CPU 2: Processed  2 has finished
Time slot  16
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  6
        CPU 2: Dispatched process  8
        CPU 1: Put process  4 to run queue
        CPU 1: Dispatched process  4
        CPU 3: Put process  5 to run queue
        CPU 3: Dispatched process  7
Time slot  17
        CPU 2: Put process  8 to run queue
        CPU 0: Put process  6 to run queue
Time slot  18
        CPU 0: Dispatched process  5
        CPU 1: Put process  4 to run queue
        CPU 1: Dispatched process  4
        CPU 2: Dispatched process  8
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  6
Time slot  19
        CPU 0: Processed  5 has finished
        CPU 0: Dispatched process  7
        CPU 2: Put process  8 to run queue
Time slot  20
        CPU 2: Dispatched process  8
        CPU 1: Processed  4 has finished
        CPU 1 stopped
        CPU 3: Put process  6 to run queue
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
Time slot  21
        CPU 3: Dispatched process  6
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
```

(c)

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

(d)

Figure 8: Scheduler outcome of test `os_1`

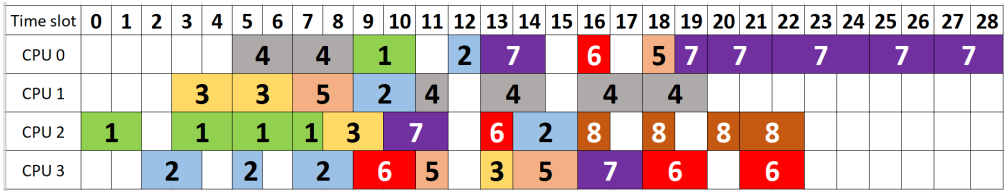

Figure 9: Gantt chart of test `os_1`

```
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)
       01414: 64
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)
021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
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 000, nxt: 032)
032: 08000-083ff - PID: 06 (idx 001, nxt: 033)
033: 08400-087ff - PID: 06 (idx 002, nxt: 034)
       085e7: 0a
034: 08800-08bff - PID: 06 (idx 003, nxt: 035)
035: 08c00-08fff - PID: 06 (idx 004, nxt: -01)
```

Figure 10: Memory status after the execution of test `os_1`