SISTEM OPERASI

Arif Wicaksono Septyanto, M.Kom

Email : arif.wicaksono@lecturer.itk.ac.id

Penjadwalan Proses

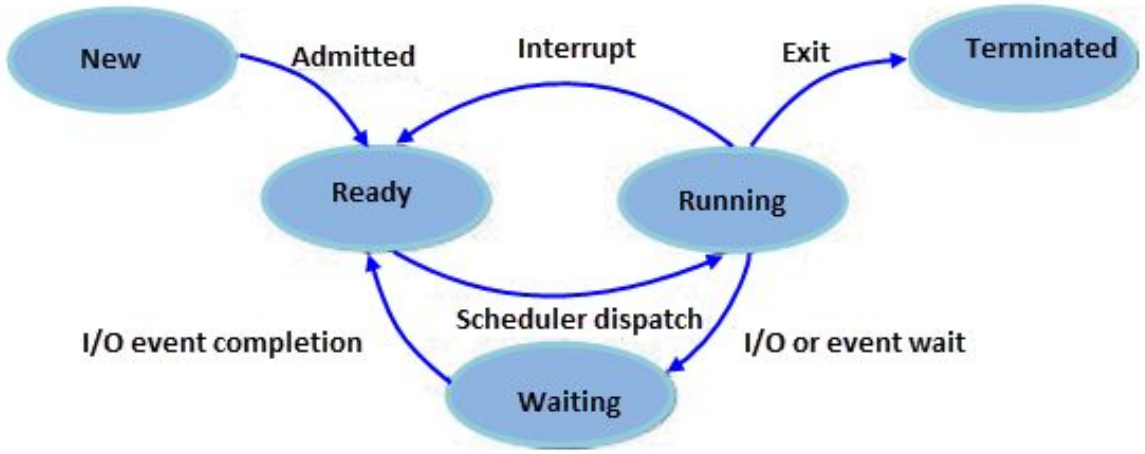
Penjadwalan Proses

Kumpulan kebijaksanaan dan mekanisme di sebuah sistem operasi yang berkaitan dengan urutan kerja dalam memutuskan proses mana yang akan dijalankan dahulu, kapan berjalan atau berapa lama proses itu berjalan

Kriteria Penjadwalan dalam Sistem Operasi

1. **Throughput**
   - Jumlah proses yang selesai dieksekusi dalam periode waktu tertentu.
2. **Turnaround Time**
   - Total waktu yang dibutuhkan proses dari awal hingga selesai.
   - turnaround time = waktu eksekusi + waktu tunggu
3. **Response Time**
   - Waktu yang diperlukan untuk mulai merespon proses setelah permintaan diajukan.
4. **Waiting Time**
   - Waktu proses menunggu dalam antrian sebelum mendapatkan CPU.
5. **CPU Utilization**
   - Persentase waktu CPU digunakan untuk menjalankan proses dibandingkan waktu idle.
6. **Fairness**
   - CPU dialokasikan secara adil di antara semua proses.
7. **Priority**
   - Proses dengan prioritas lebih tinggi dijalankan lebih dulu.

Penjadwalan Proses dalam Sistem Operasi



1. Start (Mulai) = Tahap awal ketika sebuah proses baru saja dibuat atau dimulai.
2. Ready (Siap) = Proses menunggu giliran untuk menggunakan CPU. Proses berada di sini setelah dibuat atau ketika dihentikan sementara oleh penjadwal.
3. Running (Berjalan) = Proses sedang berjalan dan instruksinya sedang dieksekusi oleh CPU.
4. Waiting (Menunggu) = Proses menunggu sumber daya, seperti input pengguna atau file, sebelum bisa dilanjutkan.
5. Terminated (Berhenti) = Proses selesai atau dihentikan, dan menunggu dihapus dari memori.

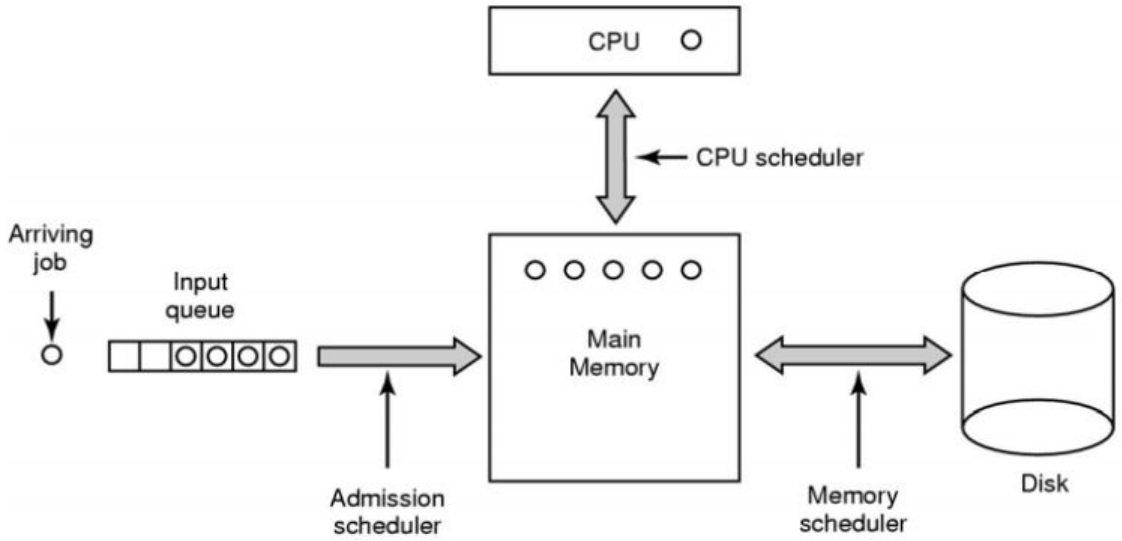
Tujuan

Meminimalisir waktu menganggur atau (Idle) Time CPU, untuk seluruh proses yang ada di memori yang akan dikerjakan di CPU perlu dijadwalkan sehingga waktu tunggu atau waiting time tiap-tiap proses di ready queue / pada antrian tidak terlalu lama.

Tipe Penjadwalan Berdasarkan Jangka Waktu

1. **Penjadwalan Jangka Pendek (Short-Term Scheduling)**
   - Menentukan proses mana yang akan dieksekusi oleh CPU dalam waktu dekat.
   - Dilakukan oleh CPU Scheduler.
   - Sering terjadi dan berlangsung sangat cepat.
   - **Contoh:** Round Robin, Shortest Job First (SJF).
2. **Penjadwalan Jangka Menengah (Medium-Term Scheduling)**
   - Menunda atau menangguhkan proses (swapping) dari memori utama ke disk untuk mengurangi beban memori.
   - Dilakukan oleh Memory Manager.
   - Tujuannya adalah untuk mengoptimalkan penggunaan memori dan mendukung multitasking.
   - **Contoh:** Suspended Process Management, Swapping.
3. **Penjadwalan Jangka Panjang (Long-Term Scheduling)**
   - Mengontrol jumlah proses yang dimasukkan ke dalam memori utama untuk dieksekusi.
   - Dilakukan oleh Long-Term Scheduler (Job Scheduler).
   - Bertujuan untuk menjaga keseimbangan antara proses yang berjalan dan yang menunggu.
   - **Contoh:** Pengaturan antrian untuk proses batch, penerimaan proses baru ke dalam sistem.
   - Batch biasanya merupakan proses-proses dengan penggunaan sumber daya yang intensif (perhitungan data statistik, backup data, perangkat I/O)

Penjadwalan Proses



Tiga Level Penjadwalan

Konsep Penjadwalan dalam Sistem Operasi

1. **Arriving Job**
   - Proses atau pekerjaan baru yang masuk ke sistem dan siap diproses.
2. **Input Queue**
   - Antrian untuk proses-proses baru yang menunggu alokasi sumber daya sebelum dieksekusi.
3. **Admission Scheduler**
   - Komponen yang menerima arriving jobs dan memutuskan untuk memasukkannya ke antrian siap (ready queue).
4. **CPU Scheduler**
   - Menentukan proses mana yang akan dijalankan oleh CPU selanjutnya dan mengelola antrian siap.
5. **Memory Scheduler**
   - Mengelola alokasi memori untuk proses dan mengatur swapping antara memori dan disk jika diperlukan.

Scheduling Algorithm

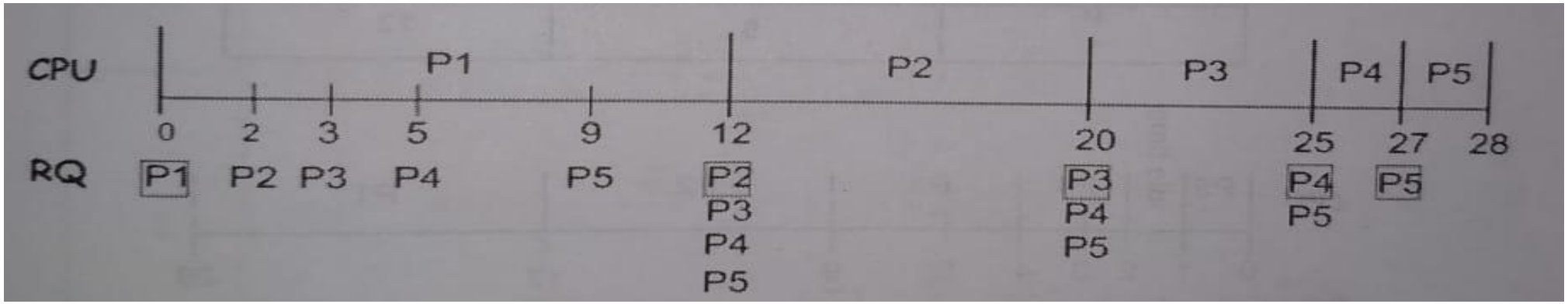
- Proses yang belum mendapatkan jatah alokasi dari CPU akan mengantri di *ready queue*.
- Algoritma Penjadwalan diperlukan untuk mengatur giliran proses-proses tersebut.
- Algoritma-algoritma penjadwalan :
  - ✔ FCFS (First Come First Serve).
  - ✔ SJF (Sortest Job First).
  - ✔ Priority Scheduling.
  - ✔ Round Robin.

FCFS Algorithm (1)

- Penjadwalan ini merupakan penjadwalan Non Preemptive.
- Dalam penjadwalan FCFS (First Come First Serves) :
  - ✔ Proses yang pertama kali minta jatah waktu untuk menggunakan CPU akan dilayani terlebih dahulu.
  - ✔ Kalau ada proses tiba pada waktu yang sama, maka pelayanan mereka dilaksanakan melalui urutan mereka dalam antrean.
  - ✔ Proses di antrean belakang harus menunggu sampai semua proses di depannya selesai.

FCFS Algorithm (2)

Proses- proses yang akan dikerjakan oleh CPU sebagai berikut:



Proses	Arrival Time (AT)	Burst Time (BT)
P1	0	12
P2	2	8
P3	3	5
P4	5	2
P5	9	1

Waktu tunggu setiap proses :

P1 = 0 = 0
P2 = 12-2 = 10
P3 = 20-3 = 17
P4 = 25-5 = 20
P5 = 27-9 = 18
==========
Jumlah = 65

Jadi rata-rata waktu tunggu setiap proses : Avg 65/5 = 13

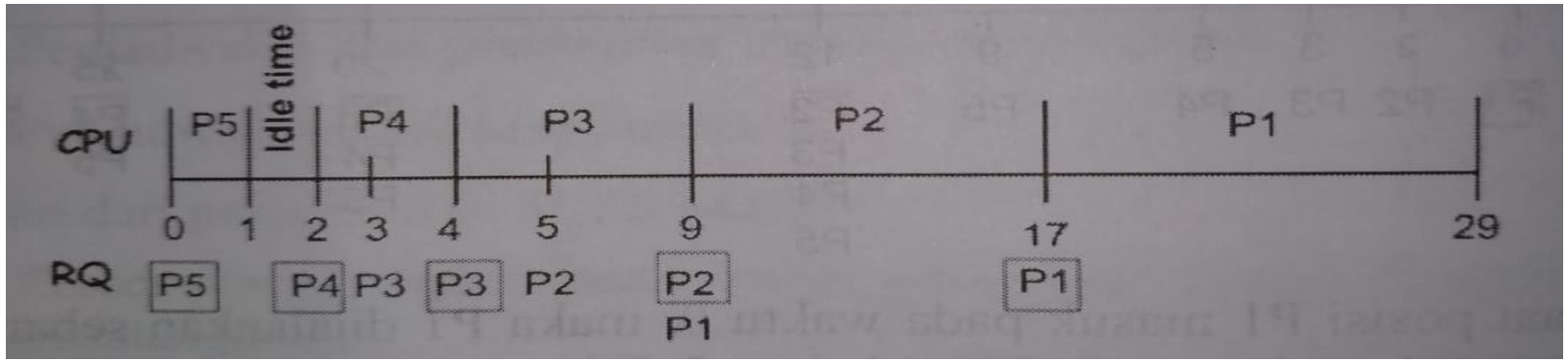
FCFS Algorithm (2)

- **Contoh soal 1:**
  Jika diketahui terdapat 5 macam antrian proses, yaitu A-B-C-D-E dengan waktu kedatangan semuanya 0. Lama proses berturut-turut antara lain: 5-2-6-8-3
  - ✔ Pertanyaan:
    - Kapan dimulainya eksekusi dari tiap-tiap antrian proses tsb?
    - Kapan selesai eksekusinya?
    - Hitung Turn Arround Time (TA)-nya?
    - Berapa rata-rata TA?

- Rumus
  - ✔ TA = Waktu Tunggu + Lama Eksekusi
  - ✔ Rata-rata TA = ∑TA / ∑Job
  - ✔ Waktu Tunggu = Mulai Eksekusi – Waktu Tiba

FCFS Algorithm (3)

Proses- proses yang akan dikerjakan oleh CPU sebagai berikut:



Proses	Arrival Time (AT)	Burst Time (BT)
P5	0	1
P4	2	2
P3	3	5
P2	5	8
P1	9	12

FCFS Algorithm (3)

- **Contoh soal 1:**
  Jika diketahui terdapat 5 macam antrian proses, yaitu A-B-C-D-E dengan waktu kedatangan semuanya 0. Lama proses berturut-turut antara lain: 5-2-6-8-3
  - ✔ Pertanyaan:
    - Kapan dimulainya eksekusi dari tiap-tiap antrian proses tsb?
    - Kapan selesai eksekusinya?
    - Hitung Turn Arround Time (TA)-nya?
    - Berapa rata-rata TA?

- Rumus
  - ✔ TA = Waktu Tunggu + Lama Eksekusi
  - ✔ Rata-rata TA = ∑TA / ∑Job
  - ✔ Waktu Tunggu = Mulai Eksekusi – Waktu Tiba

FCFS Algorithm (4)

Jawaban:

Nama Proses	Waktu Tiba	Lama Eksekusi
A	0	5
B	0	2
C	0	6
D	0	8
E	0	3

FCFS Algorithm (5)

Nama Proses	Waktu Tiba	Lama Eksekusi	Mulai Eksekusi	Waktu Tunggu	Selesai Eksekusi	TA
A	0	5	0	0	5	5
B	0	2	5	5	7	7
C	0	6	7	7	13	13
D	0	8	13	13	21	21
E	0	3	21	21	24	24
					Rata-rata TA = 14	

FCFS Algorithm (6)

- **Contoh Soal 2:**
  Jika diketahui terdapat 5 macam antrian proses, yaitu A-B-C-D-E dengan waktu kedatangan semuanya 0-1-2-2-5 Lama proses berturut-turut antara lain: 5-2-6-8-3
  - ✔ Pertanyaan:
    - Kapan dimulainya eksekusi dari tiap-tiap antrian proses tsb?
    - Kapan selesai eksekusinya?
    - Hitung Turn Arround Time (TA)-nya?
    - Berata rerata TA?

- Rumus
  - ✔ TA = Waktu Tunggu + Lama Eksekusi
  - ✔ Rerata TA = ∑TA / ∑Job
  - ✔ Waktu Tunggu = Mulai Eksekusi – Waktu Tiba

FCFS Algorithm (7)

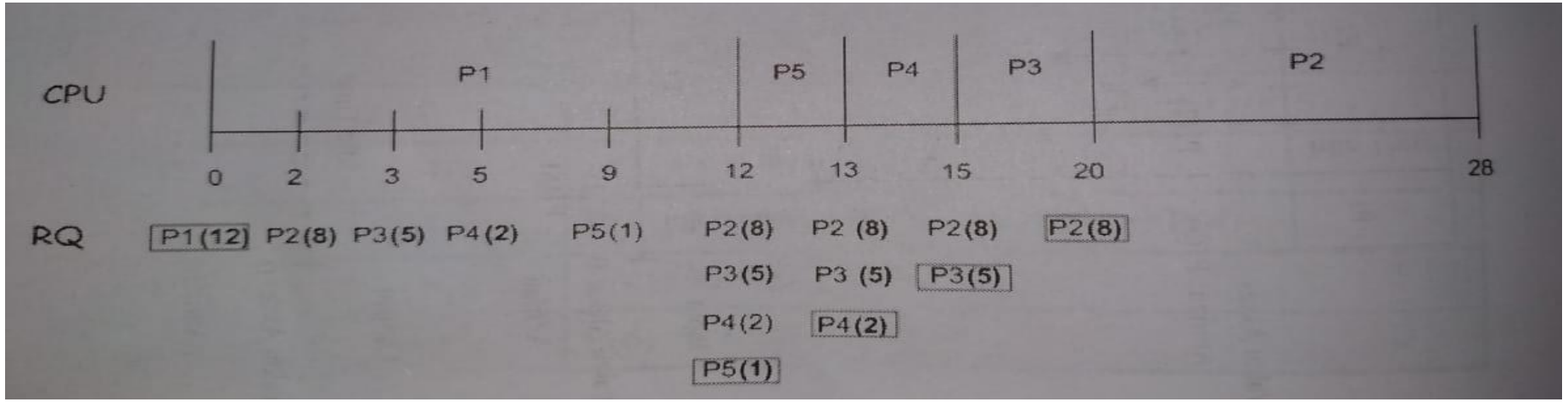
Nama Proses	Waktu Tiba	Lama Eksekusi	Mulai Eksekusi	Seleksi Eksekusi	Waktu Tunggu	TA
A	0	5	0	5	0	5
B	1	2	5	7	4	6
C	2	6	7	13	5	11
D	2	8	13	21	11	19
E	5	3	21	24	16	19
					Rerata TA = 12	

SHORTEST JOB FIRST SCHEDULING (SJF)

- Pada penjadwalan SJF, proses yang memiliki CPU burst paling kecil dilayani terlebih dahulu. Terdapat dua skema :
  - **Non preemptive**, bila CPU diberikan pada proses, maka tidak bisa ditunda sampai CPU burst selesai.
  - **Preemptive**, jika proses baru datang dengan panjang CPU burst lebih pendek dari sisa waktu proses yang saat itu sedang dieksekusi, proses ini ditunda dan diganti dengan proses baru. Skema ini disebut dengan Shortest-Remaining Time-First (SRTF).

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh 1. SJF Non-Preemptive



Proses	Arrival Time (AT)	Burst Time (BT)
P1	0	12
P2	2	8
P3	3	5
P4	5	2
P5	9	1

Waktu tunggu setiap proses :
P1 = 0 = 0
P2 = 20-2 = 18
P3 = 15-3 = 12
P4 = 13-5 = 8
P5 = 12-9 = 3
==========
Jumlah = 41

Jadi rata-rata waktu tunggu setiap proses : Avg 41/5 = 8,2 satuan waktu

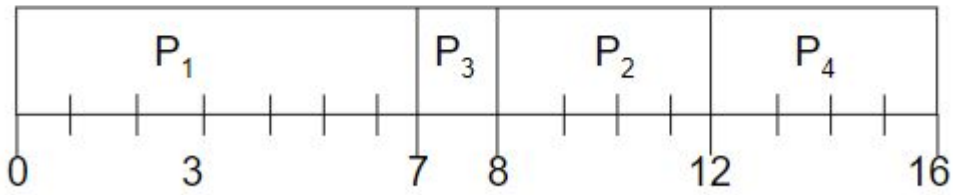
SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh 2. SJF Non-Preemptive

Contoh dari Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

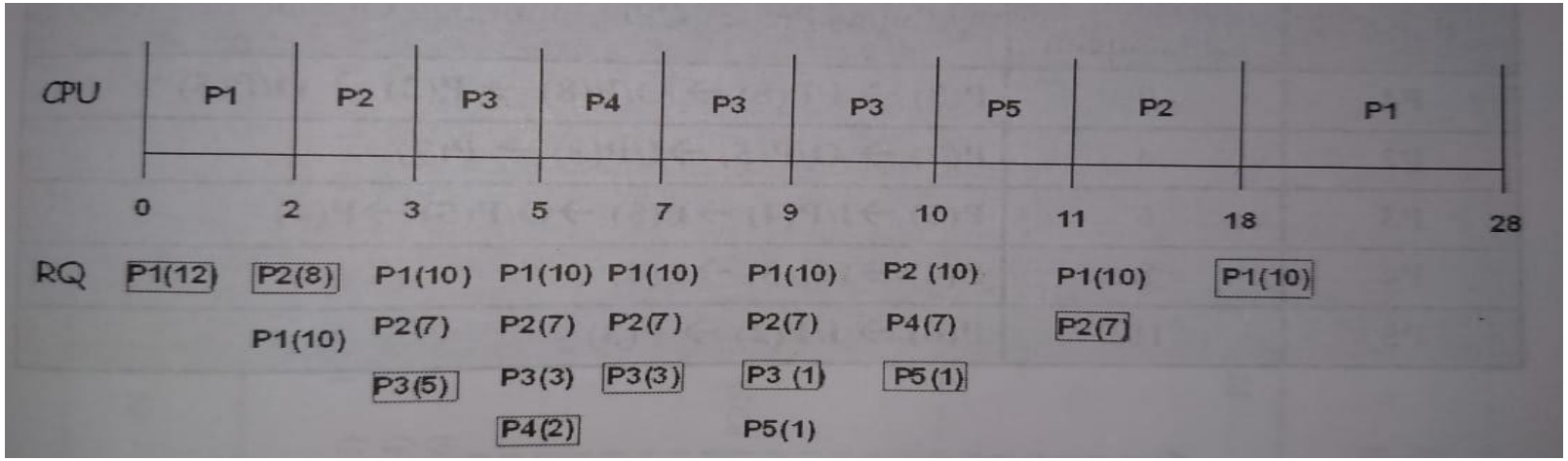


- Waktu tunggu rata-rata = (0 + 6 + 3 + 7)/4 = 4

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh 1. SJF Preemptive

Proses	Arrival Time (AT)	Burst Time (BT)
P1	0	12
P2	2	8
P3	3	5
P4	5	2
P5	9	1



Waktu tunggu setiap proses :
P1 = 0 + (18-2) = 16 (karena saat memasuki P1, P2 kembali menunggu)
P2 = (11-3) = 9
P3 = (7-5)= 2
P4 = 0 (5-5) = 0
P5 = 10-9 = 1
==========
Jumlah = 28

Jadi rata-rata waktu tunggu setiap proses : Avg 28/5 = 5,6 satuan waktu

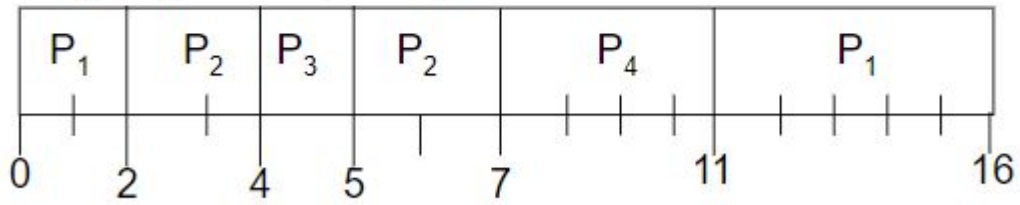
SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh 2. SJF Preemptive

Contoh dari Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh Soal 1:

Nama Proses	Waktu Tiba	Lama Eksekusi
A	0	10
B	0	5
C	0	7
D	0	1
E	0	3

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh Soal 1:

Nama Proses	Waktu Tiba	Lama Eksekusi
D	0	1
E	0	3
B	0	5
C	0	7
A	0	10

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh Soal 1:

Nama proses	Waktu tiba	Lama eksekusi	Mulai eksekusi	Selesai eksekusi	TA
D	0	1	0	1	1
E	0	3	1	4	4
B	0	5	4	9	9
C	0	7	9	16	16
A	0	10	16	26	26
				Rata2 TA = 11,2	

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh Soal 2:

Nama Proses	Lama Eksekusi	Waktu Tiba
D	1	0
E	3	2
B	5	5
C	7	7
A	10	9

SHORTEST JOB FIRST SCHEDULING (SJF)

Contoh Soal 2:

Nama Proses	Waktu Tiba	Lama Eksekusi	Mulai Eksekusi	Selesai Eksekusi	Waktu Tunggu	TA
D	0	1	0	1	0	1
E	2	3	2	5	0	3
B	5	5	5	10	0	5
C	7	7	10	17	3	10
A	9	10	17	27	8	18
					Rata-rata = 7,4	

PRIORITY SCHEDULING

- Merupakan algoritma yang mendahulukan proses yang memiliki prioritas tertinggi.
- Prioritas proses ditentukan berdasar:
  - ✔ Time limit
  - ✔ Memory requirement
  - ✔ File access
  - ✔ Perbandingan antara burst proses dengan CPU
  - ✔ Tingkat kepentinagn proses
- Priority scheduling dapat dijalankan secara preemptive dan non-preemptive.
  - ✔ Preemptive => jika ada proses yang baru datang memiliki prioritas lebih tinggi dari proses yang sedang berjalan, maka proses yang sedang berjalan tsb dihentikan, lalu CPU dialihkan untuk proses yang baru datang tersebut.
  - ✔ Non preemptive => proses yang baru datang tidak dapat menganggu proses yang sedang berjalan, tapi hanya diletakkan di depan queue

PRIORITY SCHEDULING

- Kelemahan PS adalah terjadinya infinite blocking **(starvation),** yaitu suatu proses dengan prioritas yang rendah memiliki kemungkinan tidak pernah dieksekusi jika terdapat proses lain yang memiliki prioritas lebih tinggi.
- Solusi dari starvation adalah **aging**, yaitu meningkatkan prioritas dari setiap proses yang menunggu dalam queue secara bertahap.
- Contoh : setiap 10 menit, prioritas dari masing-masing proses yang menunggu dalam queue dinaikkan 1 tingkat.
- Maka proses yang memiliki prioritas 127, setidaknya dalam 21 jam 20 menit, proses tsb akan memiliki prioritas 0, yaitu prioritas yang tertinggi

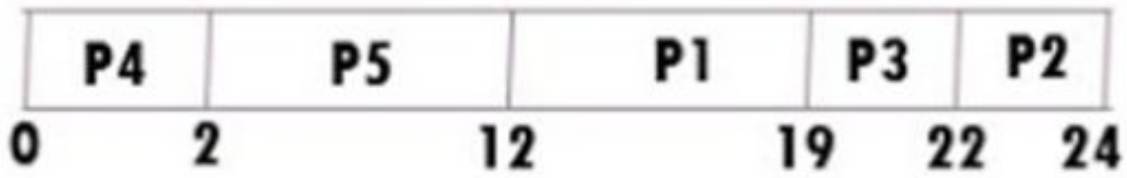
PRIORITY SCHEDULING

Contoh diketahui 5 proses dengan urutan proses sbb:

Proses	Burst Time (ms)	Prioritas
P1	7	3
P2	2	4
P3	3	3
P4	2	1
P5	10	2

PRIORITY SCHEDULING

- Gant chart



- Waiting time

Proses	Waiting Time (ms)
P1	12
P2	22
P3	19
P4	0
P5	2

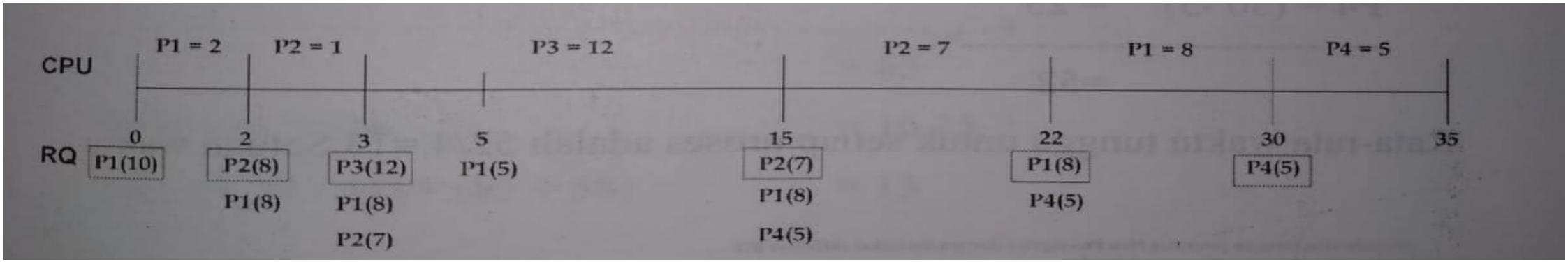
$$\text{AWT} = \frac{12 + 22 + 19 + 0 + 2}{5}$$

$$\text{AWT} = \frac{55}{5}$$

$$\text{AWT} = 11 \text{ ms}$$

PRIORITY SCHEDULING

Penjadwalan dengan prioritas Preemptive berdasarkan Prioritas size yang lebih besar.



Proses	arrival time	Burst time	Size (kb)
P1	0	10	100kb
P2	2	8	150kb
P3	3	12	175kb
P4	5	5	100kb

Waktu tunggu setiap proses :

P1 = 0 + (22-2) = 20

P2 = 15-3 = 12

P3 = 3-3 = 0

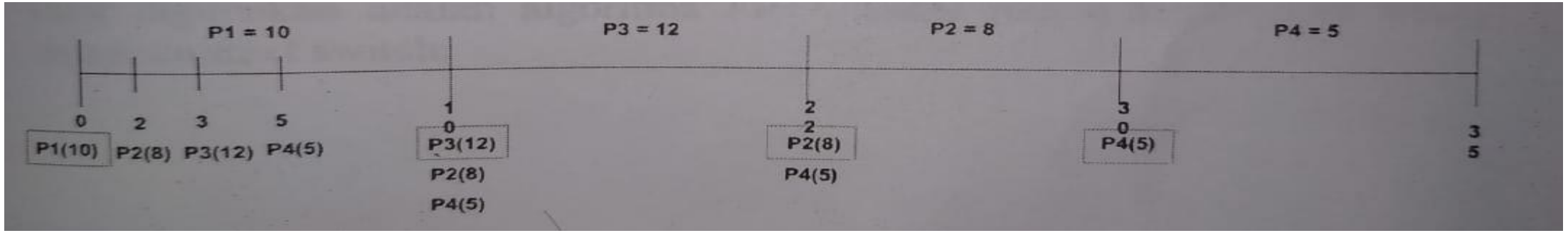
P4 = 30-5 = 25

==========

Jumlah = 57/4 = 14,25

PRIORITY SCHEDULING

Penjadwalan dengan prioritas Non preemptive berdasarkan Prioritas size yang lebih besar.



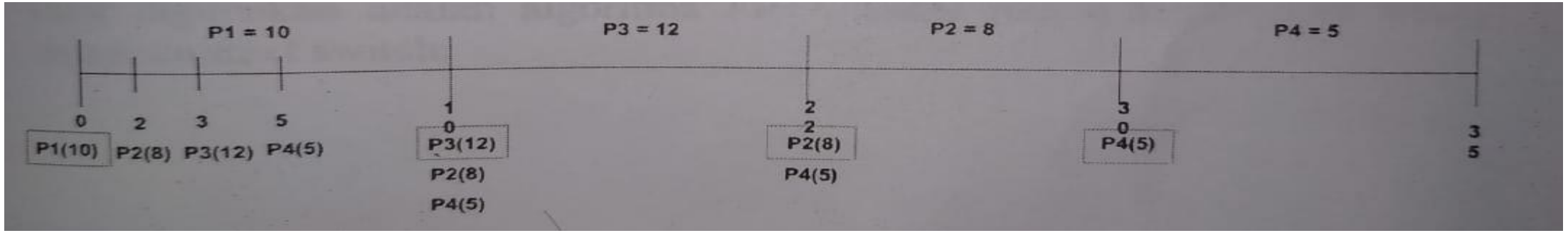
Proses	arrival time	Burst time	Size (kb)
P1	0	10	100kb
P2	2	8	150kb
P3	3	12	175kb
P4	5	5	125kb

Waktu tunggu setiap proses :

P1 = 0 = 0
P2 = 22-2 = 20
P3 = 10-3 = 7
P4 = 30-5 = 25
==========
Jumlah = 52/4 = 13

PRIORITY SCHEDULING

Penjadwalan dengan prioritas Non preemptive berdasarkan Prioritas size yang lebih besar.



Proses	arrival time	Burst time	Size (kb)
P1	0	10	100kb
P2	2	8	150kb
P3	3	12	175kb
P4	5	5	125kb

Waktu tunggu setiap proses :

P1 = 0 = 0

P2 = 22-2 = 20

P3 = 10-3 = 7

P4 = 30-5 = 25

==========

Jumlah = 52/4 = 13

ROUND ROBIN

- Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar time quantum. Jika time quantum- nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya.
- Proses ini cukup adil, karena tidak ada proses yang diprioritaskan.
- Semua proses mendapat jatah waktu yang sama dari CPU yaitu 1/n, dan tidak akan menunggu lebih lama dari (n-1)q; dimana q adalah lama 1 quantum.
- Algoritma RR sepenuhnya bergantung besarnya time quantum (TQ).
- Jika TQ terlalu besar, algoritma ini akan sama saja dengan algoritma FCFS
- Jika TQ terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu yang terbuang

ROUND ROBIN

- Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar time quantum. Jika time quantum- nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya.
- Proses ini cukup adil, karena tidak ada proses yang diprioritaskan.
- Semua proses mendapat jatah waktu yang sama dari CPU yaitu 1/n, dan tidak akan menunggu lebih lama dari (n-1)q; dimana q adalah lama 1 quantum.
- Algoritma RR sepenuhnya bergantung besarnya time quantum (TQ).
- Jika TQ terlalu besar, algoritma ini akan sama saja dengan algoritma FCFS
- Jika TQ terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu yang terbuang

PERMASALAHAN ALGORITMA RR

- Permasalahan utamanya adalah menentukan besarnya TQ. Jika TQ yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 quantum.
- Akibatnya akan terjadi banyak switch, padahal CPU memerlukan waktu untuk beralih dari satu proses ke proses yang lain (= context switches time)
- Sebaliknya, jika TQ yang ditentukan terlalu besar, algoritma RR akan berjalan seperti FCFS.
- TQ ideal adalah jika 80% dari total proses memiliki CPU burst time yang lebih kecil dari 1 TQ

PERMASALAHAN ALGORITMA RR

Diketahui 3 proses sbb:

Proses	Burst Time (ms)
P1	23
P2	6
P3	6

Gantt chart

Burst Time

Proses	Burst Time (ms)
P1	0+(9-3)+(18-12)=12
P2	3+(12-6) =9
P3	6+(15-9) =12

$$AWT=\frac{12+9+12}{3}$$
$$=\frac{33}{3}$$
$$=11 \text{ ms}$$

THANKS!

arif.wicaksono@lecturer.itk.ac.id
+62 852 1308 1309