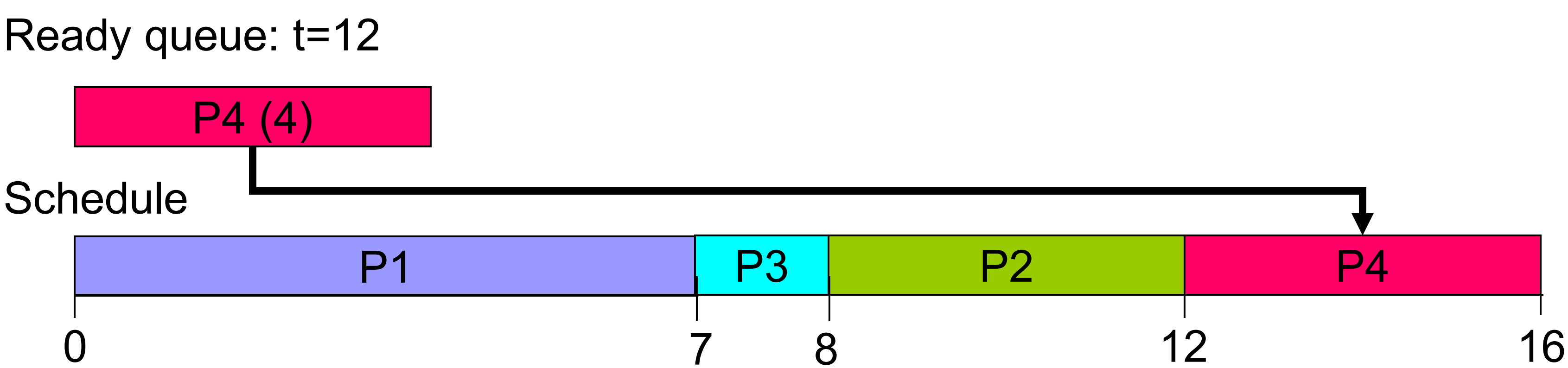
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
- A process with shortest burst length gets the CPU first
- SJF provides the minimum average waiting time (optimal!)
- Two schemes
  - **Non-preemptive** – once CPU given to a process, it cannot be preempted until its completion
  - **Preemptive** – if a new process arrives with shorter burst length, preemption happens

Non-Preemptive SJF Example

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Wait time = completion time – arrival time – run time (burst time)

AWT = [(7-0-7)+(12-2-4)+(8-4-1)+(16-5-4)]/4 = (0+6+3+7)/4 = 4

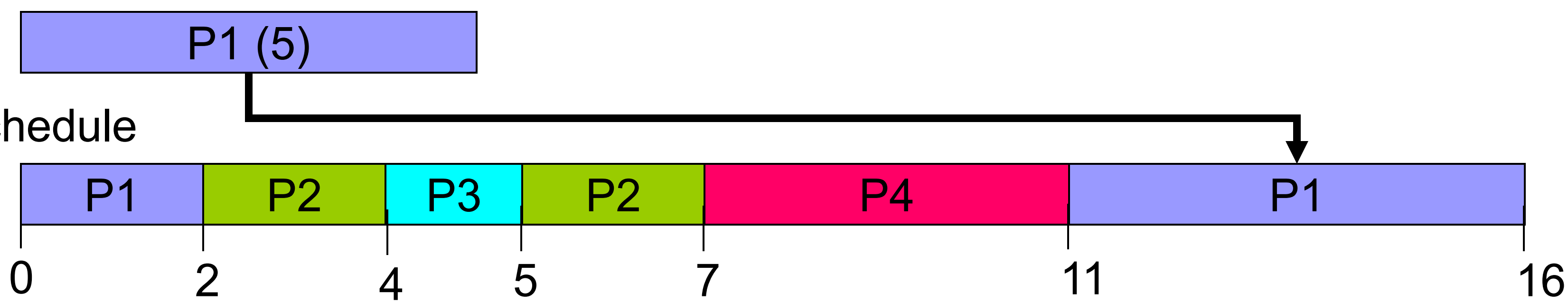
Response Time: P1=0, P2=6, P3=3, P4=7

Preemptive SJF Example

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Ready queue: t=11

Schedule



Wait time = completion time – arrival time – run time (burst time)

AWT = [(16-0-7)+(7-2-4)+(5-4-1)+(11-5-4)]/4 = (9+1+0+2)/4 = 3

Response Time: P1=0, P2=0, P3=0, P4=2

Approximate Shortest-Job-First (SJF)

- SJF difficulty: no way to know length of the next CPU burst
- **Approximate SJF**: the next burst can be predicted as an **exponential average** of the measured length of previous CPU bursts

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

new one

history

Commonly,

$\alpha = 1/2$

$$= \alpha t_n + (1-\alpha)\alpha t_{n-1} + (1-\alpha)^2 \alpha t_{n-2} + \ldots$$

$$= (\frac{1}{2})t_n + (\frac{1}{2})^2 t_{n-1} + (\frac{1}{2})^3 t_{n-2} + \ldots$$