

## 4\_ خوارزمية الجدولة الدائرية Round Robin Scheduling Algorithm

نرمز لها اختصاراً ب (RR) بحيث أن كل إجرائية تحصل على مقدار صغير من زمن المعالج CPU Time (كمية زمن نرمز لها ب  $q=quantum$ ) عادة ما يكون هذا الزمن يتراوح بين 10ms إلى 100ms.

وبعد أن يمكّن من الزمن قدره  $q$ . على بقاء الإجرائية في الـ CPU يتم استيقاف (طرد) الإجرائية الموجودة في الـ CPU وإضافتها إلى نهاية رتل الجاهزية end of ready queue.

\* إن كان لدينا  $n$  إجرائية في رتل الجاهزية وكانت كمية الزمن تساوي  $q$  وبالتالي :

نتيجة 1 : كل إجرائية سوف تأخذ  $\frac{1}{n}$  من الزمن الكلي للمعالج CPU وفي كل مرة تحصل على زمن قدره  $q$  ثانية على الأكثر أي أنها قد تنتهي خلال زمن قدره  $q$  أو قد تنتهي في زمن أقل منه.

نتيجة 2 : كل إجرائية تنتظر على الأكثر زمن قدره  $(n-1)q$  لكي تحصل على زمن تنفيذ قدره  $q$  مرة أخرى.

إنما خوارزمية بسيطة simple ، سهلة التحقيق implement ، ولا يحصل فيها حرمان أبداً أي starvation-free حيث أن كل الإجرائيات يتم معاملتها بشكل عادل لكي يتشاركون المعالج فيما بينهم

### ملاحظات حول Round Robin

- تشبه خوارزمية الـ FCFS (القادم أولاً يخدم أولاً) ، لكن تم إضافة الاستباقية إليها (الشفعية) وهذا يسمح للنظام بالتبديل بين الإجرائيات كل كمية من الزمن قدرها  $q$  (شريحة زمن time slice).
- يتم معاملة رتل الجاهزية على أنه رتل دائري cyclic بحيث أن النظام يمر على كل الإجرائيات بشكل دائري بدءاً من أول إجرائية وصولاً إلى آخر إجرائية عائداً إلى بداية الدائرة .
- معامل رتل الجاهزية على أنه FIFO (الداخل أولاً خارج أولاً) أي أن الإجرائيات الجديدة تضاف إلى نهاية الرتل (ذيل = tail) بحيث أن المدول يختار الإجرائية من بداية الرتل ليبدأ مؤقت timer لمدة من الزمن قدرها  $q$  وبعدها يقوم بإحداث مقاطعة interrupt لإدخال إجرائية جديدة .
- عند كل عملية طرد للإجرائية من الـ CPU تحدث عملية ابدال السياق Context Switch.

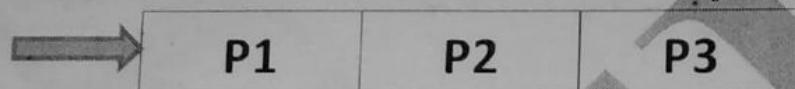
خوارزمية preemptive Round Robin بالمطلق

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

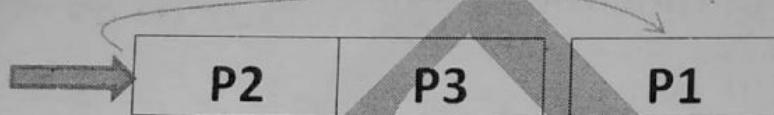
مثال : لتكن لدينا الإجراءات المبينة في الجدول التالي (ولتكن  $q=4 \text{ ms}$ ) :

Process	Burst Time
$P_1$	24
$P_2$	3
$P_3$	3

أولاً يكون رتل الجاهزية بالشكل التالي :



يتم أولاً تنفيذ  $P_1$  لمدة  $4 \text{ ms}$  ثم تخرج من ال CPU لتضاف إلى نهاية رتل الجاهزية أي :



لتأتي بعدها  $P_2$  لمدة  $4 \text{ ms}$  لكنها تنتهي قبل ذلك وبالتالي تخرج قبل انتهاء ال

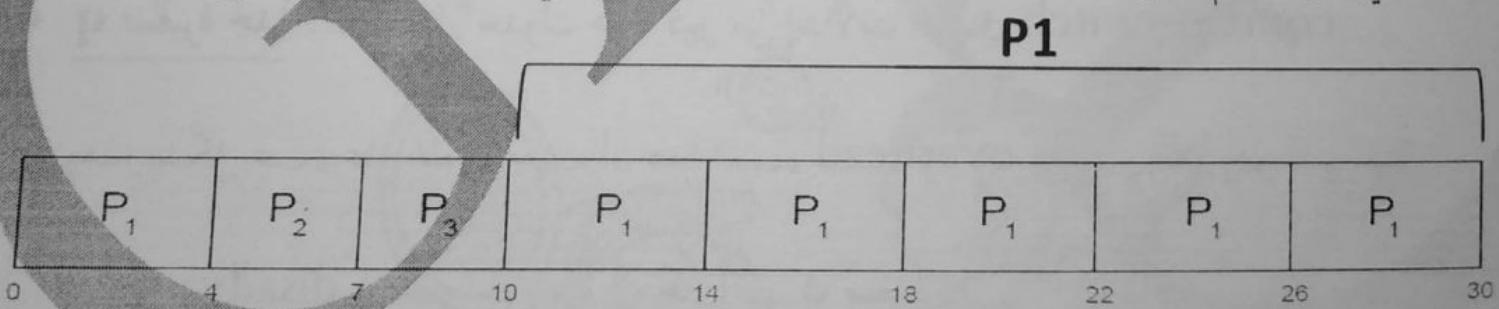
أي يبقى في رتل الجاهزية الإجراءات :



الآن يتم تنفيذ  $P_2$  لمدة  $4 \text{ ms}$  ولكنها أيضاً تنتهي خلال  $3 \text{ ms}$  أي في زمن أقل من الزمن المعطى لها وبالتالي

تخرج قبل انتهاء المدة لتدخل الإجرائية  $P_3$  لمدة  $4 \text{ ms}$  ثم تخرج ثم تعود لتدخل مجدداً لأن الإجرائية الوحيدة

المتبقية وبالتالي سوف يتم تكرار تنفيذها لمدة  $4 \text{ ms}$  متقطعة إلى أن تنتهي بالكامل وبالتالي يصبح مخطط غانت:



بالتالي نعود للقاعدة السابقة في تحديد زمن الانتظار لكل إجرائية بشكل دقيق :

$$P_1 \Rightarrow 10 - 4 = 6$$

$$P_2 \Rightarrow 4$$

$$P_3 \Rightarrow 7$$

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)



بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg} = \frac{6+4+7}{3} = \frac{17}{3} = 5.66$$

لاحظ عدد مرات خروج ودخول الإجرائية P1 وهذا يعني حدوث ابتدال السياق context switch في كل مرة تخرج فيها الإجرائية من الـ CPU ((بالتالي فإنها تضيع زمن وهو زمن ابتدال السياق)).

نستنتج أن :

خوارزمية RR تأخذ زمن لازم لكل إجرائية turnaround time كبير مقارنة مع خوارزمية SJF حيث أنها تبدل الإجرائية نفسها أكثر من مرة وبشكل كبير حتى تنتهي ، لكن !!! تعتبر خوارزمية RR تستجيب بشكل أكبر من خوارزمية SJF (Better response).

العلاقة بين كمية الزمن q وعدد مرات ابتدال السياق context switch

# بشكل عام يجب أن يكون  $q$  أكبر من زمن ابتدال السياق بشكل مناسب #

أداء خوارزمية RR يعتمد بشكل أساسى على كمية الزمن  $q$  :

•  $q$  كبيرة جداً  $\leftarrow$  تصبح RR كأنها FCFS أي أن القادم أولاً يخدم أولاً وبالتالي لم يستفاد شيئاً.

•  $q$  صغيرة جداً  $\leftarrow$  وبالتالي حدوث عدد كبير من إبدالات السياق ... context switch

وهذا يشكل عبء على المعالج حيث أن هذا العباء overhead هو كبير جداً وهو أهم سبيعة

في هذه الخوارزمية في حال كون  $q$  صغيرة . disadvantage

ملاحظة : عندما نقول FCFS فإننا نقصد تماماً FIFO حيث أن :

FIFO = First In First Out

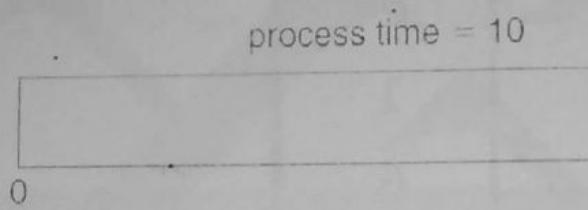
## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

مثال يبيّن أهمية كمية الزمن  $q$  في الخوارزمية RR :

ليكن لدينا إجرائية واحدة تحتاج إلى 10 وحدات زمنية (بعض النظر عن مقدار هذه الوحدة)

حالة 1 : إذا كان  $q=12 > 10$  time units بال التالي فإن الإجرائية ستنتهي في أقل من شريحة واحدة

أي أنه لن يحدث إنزال سلسلة ويكون لدينا الشكل التالي :



quantum

12

context switches

0

حالة 2 : إذا كان  $q=6 < 10$  بال التالي فإن الإجرائية ستحتاج إلى شريحتي زمن  $q=2$  أي سيتم إجراء تبديل

السياق مرة واحدة كما في الشكل التالي :



quantum

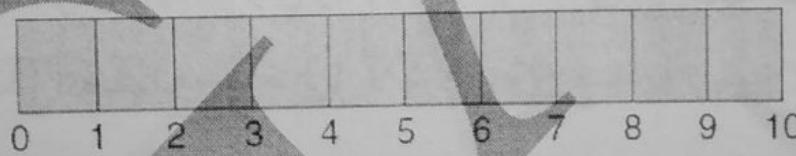
6

context switches

1

حالة 3 : إذا كان  $q=1 < 10$  بال التالي فإن الإجرائية تحتاج إلى 10 شرائح زمن  $q=10$  وبالتالي سيتم إجراء تبديل

السياق 9 مرات كما في الشكل التالي :

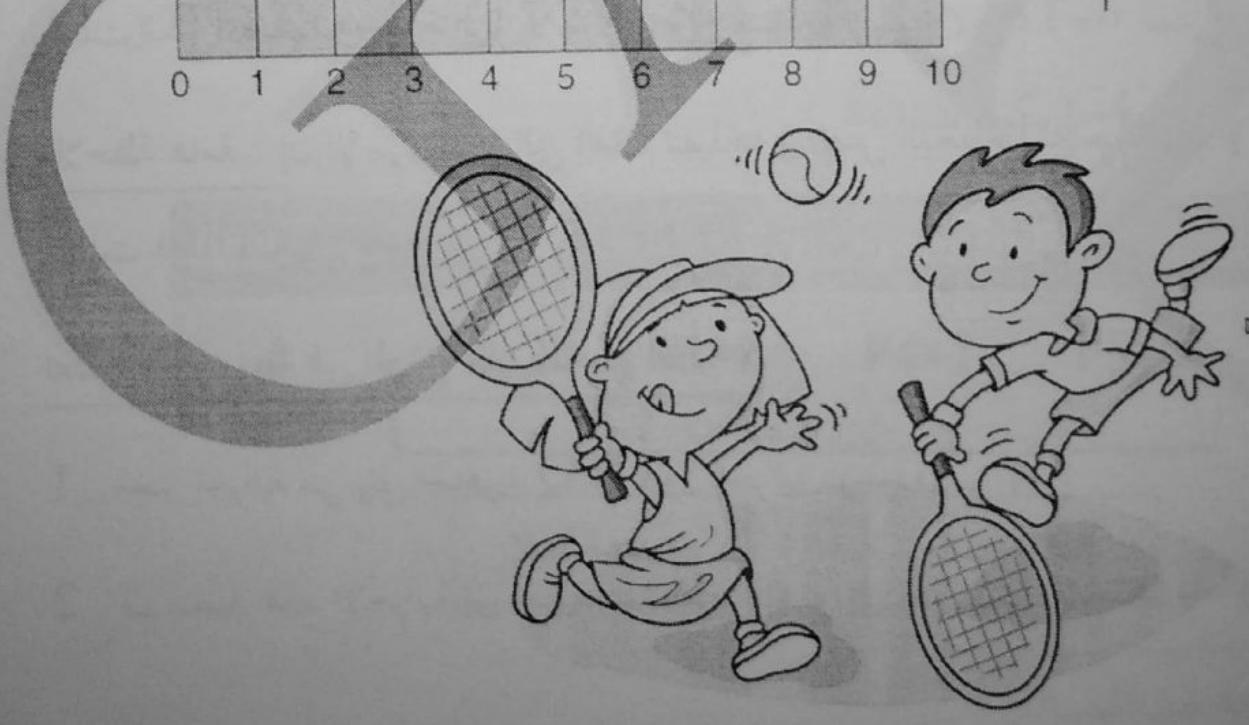


quantum

1

context

9



مثال إضافي على خوارزمية RR : ليكن لدينا مخطط الإجرائيات التالي (بفرض  $q=2 \text{ ms}$ )

Process No	Arrival Time (AT)	Burst Time (BT)
1	0	4
2	1	5
3	2	2
4	3	1
5	4	6
6	6	3

المطلوب : ارسم مخطط غانت ، واحسب الزمن الكلي TAT وزمن الانتظار WT لكل إجرائية .

الحل : سوف نقوم برسم جدول يعبر عن رتل الجاهزية ready queue بحيث يحوي على عمودين ، الأول هو الزمن  $t$  (بدءاً من الثانية 0) مقسم بحسب شريحة الزمن  $q$  أي  $2 \text{ ms}$  (ميلي ثانية = ثانية).

الثاني هو الإجرائيات الموجودة في الرتل مرتبة حسب الإجرائية التي أتت أولاً ، ومن أجل كل إجرائية تنفذ في المعالج نعود ونضعها في نهاية الرتل وهكذا بشكل دائري إلى أن تنتهي كل الإجرائيات .

نلاحظ أن كل إجرائية لها زمن وصول محدد ، وبالتالي من أجل كل لحظة  $t$  يجب أن نعلم أي إجرائية وصلت لكي نضعها في نهاية الرتل .

ملاحظة هامة : إن الإجرائية  $p_i$  التي انتهت تنفيذها في نفس اللحظة  $t$  للإجرائية  $p_j$  أو الإجرائيات التي قد وصلت فإننا نضعها في نهاية الرتل (تكون  $p_i$  في آخر الرتل) .

تعتمد الخوارزمية في العمل على الخطة التالية :

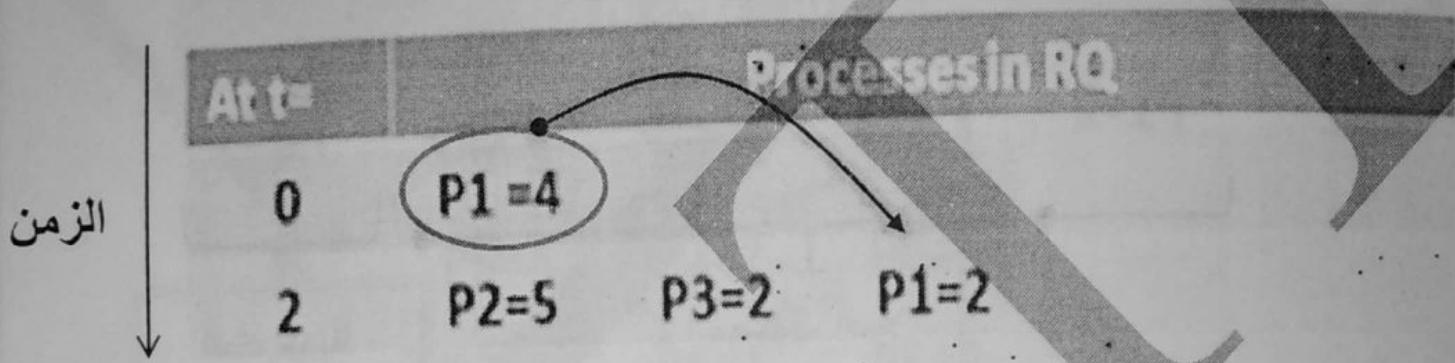
- 1 - اختار إجرائية من رتل الجاهزية  $RQ$  (أول إجرائية موجودة) .
- 2 - قم بتنفيذ هذه الإجرائية إلى حين انتهاء الزمن  $q$  أو إلى حين انتهاء تنفيذها بشكل كامل .

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

لبدأ العمل من اللحظة  $t=0$

في اللحظة  $t=0$  تأتي الإجرائية  $P1$  ذات الرشقة  $BT=4$  وبالتالي تنفذ وتحرج في الزمن  $2$  لتصبح في نهاية رتل الجاهزية (آخر الرتل).

في اللحظة  $t=1$  تكون قد أتت الإجرائية  $P2$  ذات الرشقة  $BT=5$  ، وبعدها في اللحظة  $t=2$  تأتي الإجرائية  $P3$  ذات الرشقة  $BT=2$  ولا ننسى أن نضع الإجرائية  $P1$  التي تبقى منها  $2$   $BT=2$  في نهاية المترال وبالتالي من أجل الثانية  $t=0$  و  $t=2$  يكون لدينا :



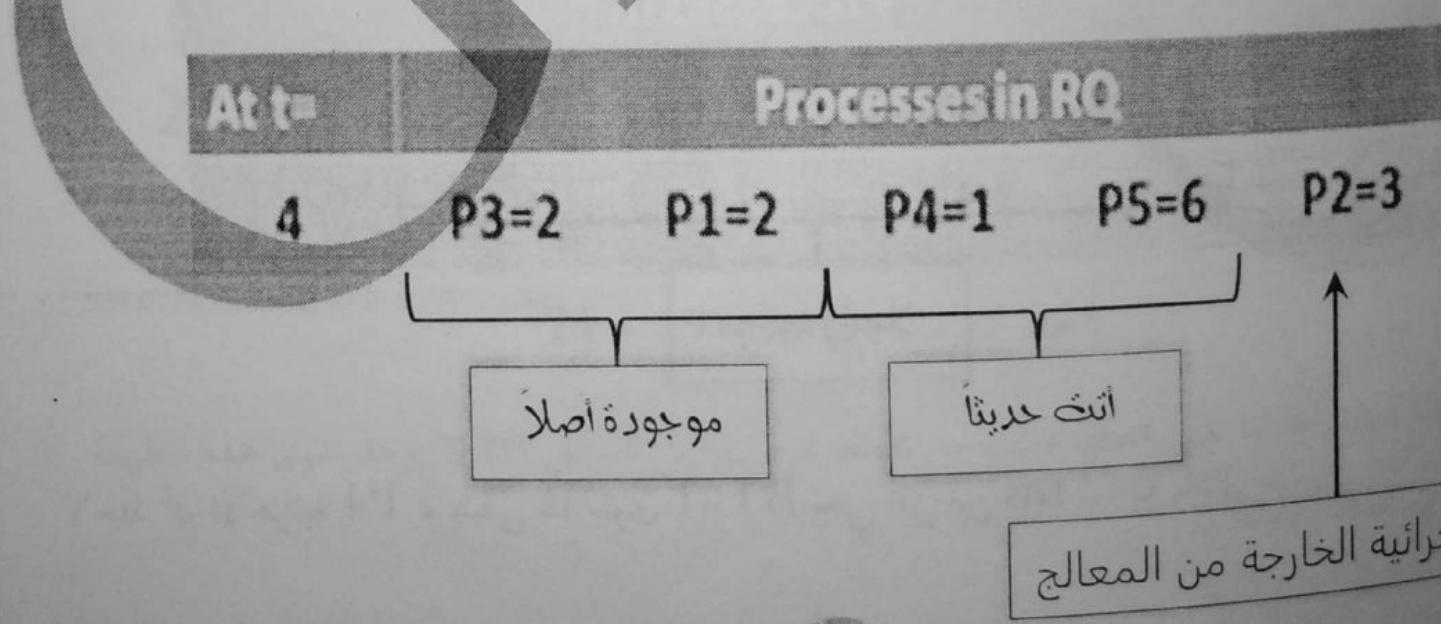
بعد ذلك ستنتهي الإجرائية  $P2$  وتحرج في نهاية رتل الجاهزية (آخر الرتل).

الآن لا ننسى أن نضع الإجرائيات الموجودة سابقاً  $P3$  و  $P1$  في بداية الرتل ومن أجل كل إجرائية

تأتي حديثاً نضعها في النهاية بعد كل الإجرائيات الموجودة أصلاً ...

بعد ذلك في اللحظة  $t=3$  تكون قد أتت الإجرائية  $P4$  ذات الرشقة  $BT=1$  ، وبعدها في اللحظة  $t=4$  تأتي الإجرائية  $P5$  ذات الرشقة  $BT=6$  ولا ننسى أن نضع الإجرائية  $P2$  التي تبقى منها  $3$   $BT=3$  في

نهاية الرتل ... وبالتالي من أجل  $t=4$  يكون لدينا :

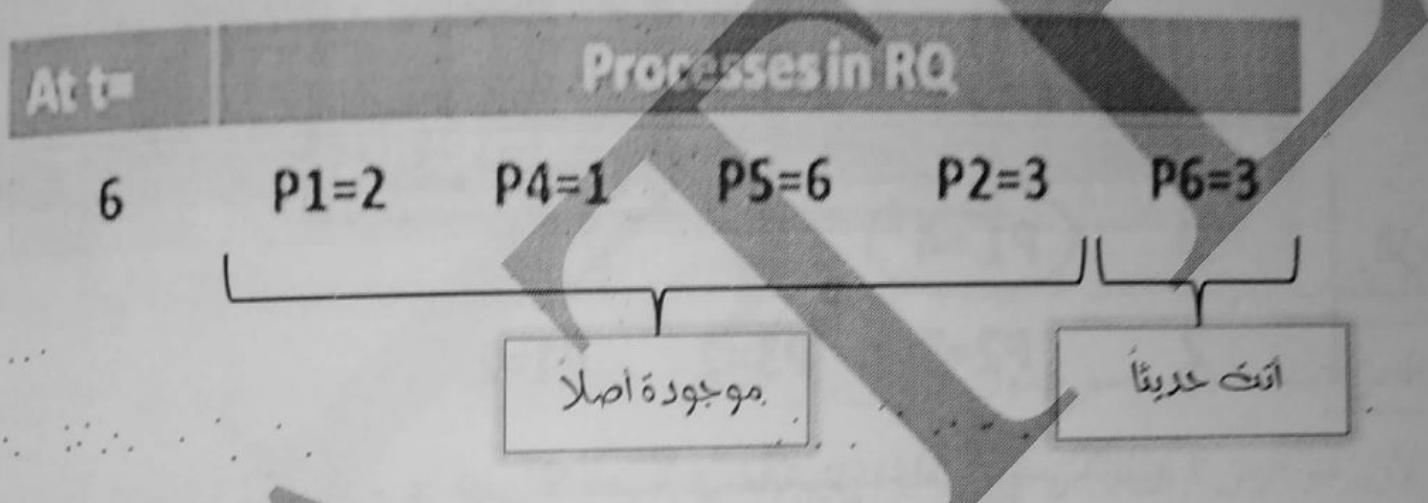


يأكل سيد الماء P3 وتخرج في الزمن 6 ليخرج بحائياً من رتل الجاهزية RQ (لماذا ٦٦٦).

لذلك، إن نتائج الامتحانات الموجودة أصلًا أي  $P_1, P_4, P_5, P_2$  بالترتيب ...

في تلك الحضرة، (6) كانت الإخبارية P6 ذات الرئشة BT=3 بالثانية تضعها في آخر الزريل... لا تحظى أله لا

يُوجَد إِيجَارٌ مُتَّسِعٌ لِكُوْنِهِ فِي تَهَايَةِ الْفَرْتِلِ إِذَا الْإِجَارَى P6 هِي الْأَخِيرَةِ وَيَكُونُ لِدِينَا :

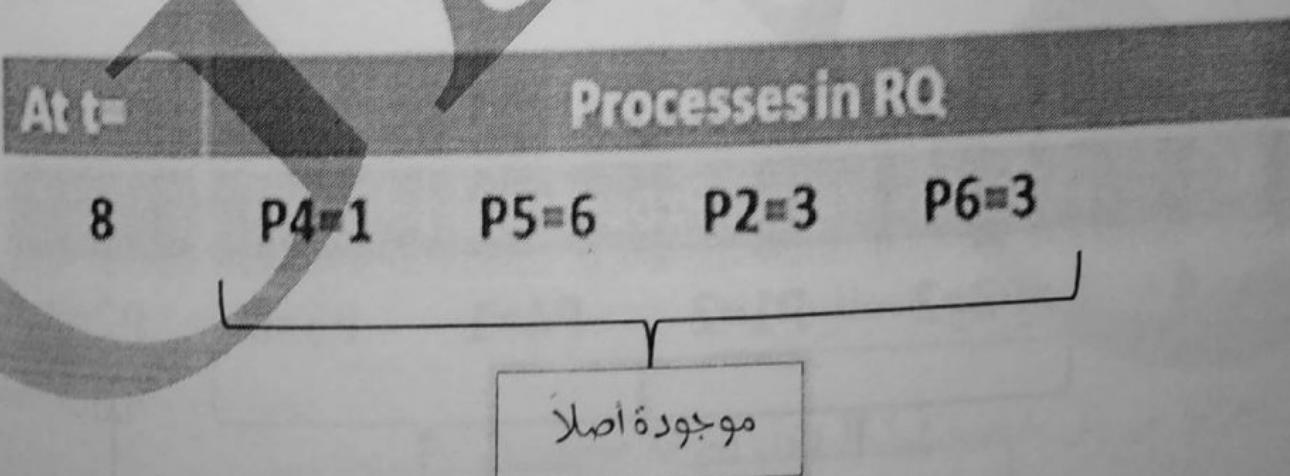


بالتالي تنفذ الإجرائية P1 ونخرج في الزمن 8 لخروج نهائياً من رتل الجاهزية RQ

الآن نلاحظ أننا في اللحظة  $t=8$  لم يعد لدينا أي إجراءات قد تأتي حديثاً ... وبالتالي فإن الإجراءات

الموجودة في الرتل  $RQ$  سوف يتناقص عددها تدريجياً مع مرور الوقت ☺ .

الآن ستخرج الإجرائية P1 وتبقى الإجرائيات الموجودة كما هي :



لاحظ أن الإجرائية P4 لم يتبقى لها سوى  $BT=1$  وهي أقل من  $q=2 \text{ ms}$  وبالتالي ...

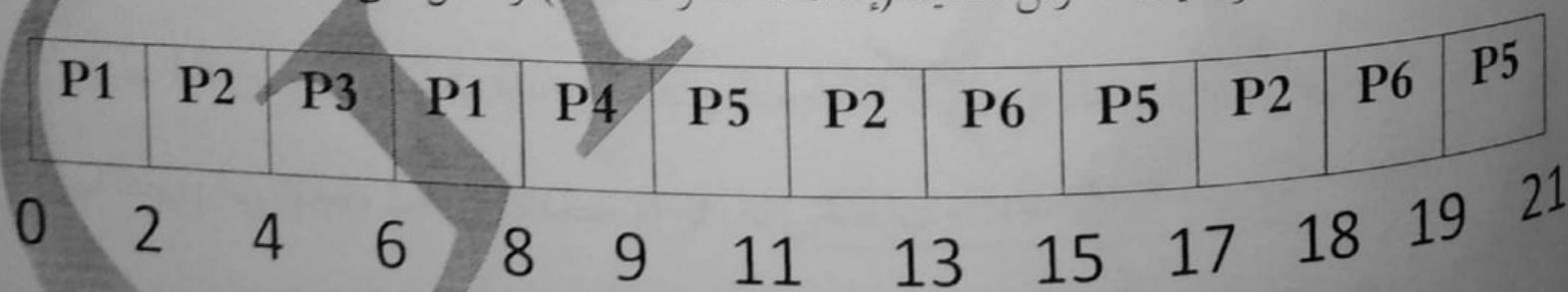
ستنتهي إلى اللحظة  $t=9$  ويكون لدينا :

At t =	Processes in RQ			
9	$P_5=6$	$P_2=3$	$P_6=3$	

الآن أصبح لدينا نفس المثال الموجود في المحاضرة السادسة (صفحة 25) وتكون تتمة الجدول كما يلي :

At t =	Processes in RQ			
11	$P_2=3$	$P_6=3$	$P_5=4$	
13	$P_6=3$	$P_5=4$	$P_2=1$	
15	$P_5=4$	$P_2=1$	$P_6=1$	
17	$P_2=1$	$P_6=1$	$P_5=2$	
18	$P_6=1$	$P_5=2$		
19	$P_5=2$			

الآن بأخذ العمود الثاني من الجدول بأكمله نحصل على مخطط غانت المنشود ، بحيث تكون نقطة البداية هي زمن البدء أي  $t=0$  ونضيف لها زمن التنفيذ (إما 2 ms أو 1 ms) ونحصل على :



لاحظ أن الإجرائية P4 تقدّمت فقط ثانية واحدة من 8 إلى 9

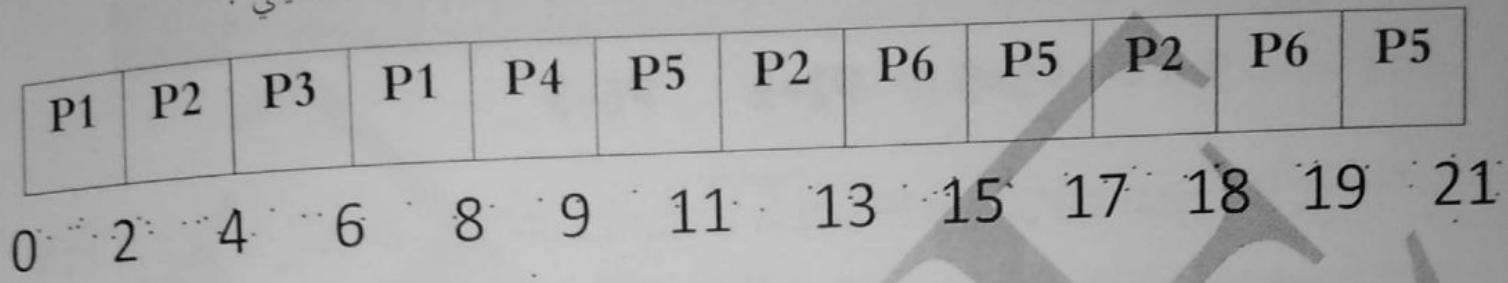
لاحظ أن زمن الانتهاء  $21 = 19 + 2$

هل فهمت كيف حصلنا على مخطط غانت من مخطط الزمن والرتب السابق؟ لا يوجد سوى هذه الطريقة  
لا تفكّر في غيرها أرجوكم ...

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

الآن سوف نحسب زمن الانتظار WT و زمن الدورة الكلية TAT لكل إجرائية ...

سوف نعتمد على الجدول الأساسي (الجدول المعطى) بالإضافة إلى مخطط غانت كما يلي :



Process No	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)	Turn-Around Time (TAT)	Waiting Time (WT)
1	0	4	8	8	4
2	1	5	18	17	12
3	2	2	6	4	2
4	3	1	9	6	5
5	4	6	21	17	11
6	6	3	19	13	10

حساب زمن الانتظار سوف نأخذ مثال عن الإجرائية P1 :

لاحظ أن P1 وصلت في اللحظة  $t=0$  ونفدت في اللحظة  $t=4$  ms مباشرةً وبالتالي لم تنتظر شيء ...

لأنها انتظرت من اللحظة  $t=2$  إلى اللحظة  $t=6$  وبالتالي انتظرت  $4$  ms.

لاحظ أيضاً أن الإجرائية P3 وصلت في اللحظة  $t=2$  ونفدت في اللحظة  $t=4$  وبالتالي انتظرت  $2$  ms.

لاحظ أنه يوجد علاقة أخرى نستطيع منها الحصول على الزمن TAT :

$$TAT = CT - AT$$

العلاقة الأسهل هي :

$$TAT = BT + WT$$

حيث أن CT هي زمن الانتهاء ، AT هي زمن الوصول .

## ٥- خوارزمية الجدولة وفق أرطال متعددة المستويات : Multilevel Queue

في هذه الخوارزمية يتم تقسيم رتل الظاهرة إلى أرطال متفصلة separate queues ready queue إلى أرطال أساسية يتم التقسيم أغلب الأحيان إلى رتلين أساسين هما :

### ١- الرتل الأمامي foreground queue

وهي يتم وضع الإجراءات التفاعلية interactive (إجراءات تتطلب دخلاً من المستخدم) في هذه الرتل.

### ٢- الرتل الخلفي background queue

وهي يتم وضع الإجراءات الدفعية batch (إجراءات تتطلب معالجة في الأغلب)

ملاحظة : أي إجرائية هي حتماً permanently موجودة في أحدى الأرطال .

بحيث أن كل رتل لديه خوارزمية الجدولة الخاصة به :

Foreground queue  $\Rightarrow$  RR Algorithm (الدائري)

Background queue  $\Rightarrow$  FCFS Algorithm (القادم أولًا يخدم أولًا)

بالإضافة إلى أنه يوجد عدة خوارزميات جدولة بين الأرطال نفسها ذكر منها :

### ١- الجدولة ثابتة الأولوية الشفعية fixed-priority preemptive scheduling

أي أنه يوجد أولوية خاصة لكل ثابتة لا تتغير أبداً الدهر ...

فمثلاً يمكن أن يكون لدينا ثلاثة أرطال  $q_1, q_2, q_3$

يمكناً أن نعطي الرتل  $q_1$  الأولوية 0 والرتل  $q_2$  الأولوية 1 والرتل  $q_3$  الأولوية 2

بالتالي الإجراءات الموجودة في  $q_1$  لها أولوية أعلى من كل الإجراءات الموجودة في  $q_2$  و  $q_3$  .

أي أنه يتم تخدم Serve كل الإجراءات الموجودة في الرتل الأمامي foreground أولاً ثم بعدها يتم تخدم الإجراءات الموجودة في الرتل الخلفي background

بال التالي قد يحدث هنا !!!

ماذا قد يحدث هنا ???

إن لم تستطع الإجابة أصلح  
بإعادة دراسة المحاضرة جيداً

لنأخذ مثلاً واقعاً على رتل جاهزيات مقسم إلى 5 أرطال تدرج في الأولوية بشكل تناظري :

highest priority

إجراءات النظام

system processes

إجراءات تفاعلية

interactive processes

إجراءات التحرير التفاعلية

interactive editing processes

إجراءات دفعية

batch processes

إجراءات طلاب (مستخدمين)

student processes

lowest priority

نستنتج من هذه الأرطال ما يلي:

- أن كل رتل له أولوية مطلقة على الأرطال الأدنى منه أولوية.
- لا يمكن تنفيذ إجرائية من رتل الإجراءات الدفعية batch مثلاً، ما لم تكن أرطال إجراءات النظام والإجراءات التفاعلية وإجراءات التحرير التفاعلية فارغة كلها.
- إذا دخلت إجرائية تحرير تفاعلية في رتل الماجاهزية في أثناء تنفيذ إجرائية دفعية فإن الإجرائية الدفعية التي تفقد بحث أن تتوقف Stop بحق الشفعة (What The Hell???) .

## 2\_ الجدولة بقطع الوقت : Time Slice

كل رتل يأخذ مدة محددة من زمن المعالج بحيث خلال هذا الوقت يستطيع كل رتل أن يجدول إجرائياته بالطريقة التي يريدها .

مثال : يمكن أن نعطي 80% للرجل الأمامي و 20% للرجل الخلفي أي أن :

80%  $\Rightarrow$  foreground in RR

20%  $\Rightarrow$  background in FCFS

ملاحظة : كل إجرائية موجودة في رتل ما فهي تبقى في هذا الرتل ، لا يمكننا تغييره أبداً.

## نظم تشغيل المعاشرة السادسة (خوارزميات الجدولة 2)



### 6\_ الجدولة وفق أرطال متعددة المستويات بتغذية راجعة Multilevel Feedback Queue

لـ Multilevel Queue لكن هنا يمكن للإجرائية أن تنتقل من رتل إلى آخر ☺

حيث أنه يتم تقسيم الإجرائيات تبعاً لطول رشقات الـ CPU ، وبالتالي الإجرائية التي تتطلب زمن معالجة كبير (طولة رشقة كبيرة) سوف يتم نقلها إلى رتل آخر أقل أولوية .

بالإضافة إلى أن الإجرائية التي تتطلب وقتاً طويلاً في الرتل منخفض الأولوية lower-priority queue وهذا النمط من التعمير يمنع قد يتم نقلها إلى رتل أعلى أولوية higher-priority queue .

حصول الحرمان starvation (تحذرنا عنه سابقاً).

يتم تعريف المجدول متعدد الأرطال ذو التغذية الراجعة بتحديد الوسطاء parameters التالية :

- عدد الأرطال number of queues
- خوارزمية الجدولة لكل رتل scheduling algorithms for each queue
- الطريقة التي يتم فيها اختيار إجرائية لكي يتم ترقيتها upgrade
- الطريقة التي يتم فيها اختيار إجرائية لكي يتم تخفيضها demote
- الطريقة التي يتم فيها اختيار الرتل الذي ستتدخله إجرائية إن كانت تحتاج للخدمة needs service

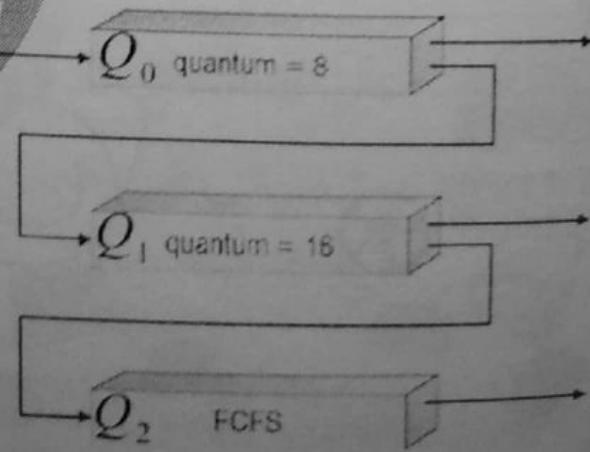
مثال : لأخذ النموذج التالي :

لدينا ثلاثة أرطال وهي  $Q_0$  و  $Q_1$  و  $Q_2$  بحيث أن :

$$Q_0 \Rightarrow RR (q = 8)$$

$$Q_1 \Rightarrow RR (q = 16)$$

$$Q_2 \Rightarrow FCFS$$



## سـم سـيل المـحـاـصـرـه السـادـسـه (خـواـرـزـمـيـات)

أولاً : من ناحية الأولوية بين الأرتال :

الرتل  $Q_0 <> Q_1 <> Q_2$

أي أنه سيتم تنفيذ كل الإجرائيات الموجودة في الرتل  $Q_0$  وحين يكون  $Q_0$  حالياً (empty)

يتم تنفيذ كل الإجرائيات الموجودة في  $Q_1$

و حين يكون كلاً من  $Q_0$  و  $Q_1$  فارغين معاً يتم تنفيذ كل إجرائيات الموجودة في  $Q_2$ .

ثانياً : من ناحية التبديل بين الأرتال :

كل إجرائية جديدة (عمل job) تدخل إلى الرتل  $Q_0$  يتم إعطاؤها 8 ms لكي تنهي عملها ،

وإن مضت الـ 8 ms ولم تنتهي الإجرائية بعملها وبالتالي تنقل إلى ذيل (tail) الرتل  $Q_1$

الآن إنما كان  $Q_0$  فارغاً وبالتالي يتم إعطاء الإجرائية الموجودة في رأس (head) الرتل  $Q_1$  16 ms لتنهي عملها ، وإن لم تنتهي عملها في هذه المدة وبالتالي يتم نقلها إلى ذيل  $Q_2$  لمعاملة بطريقة FCFS

**نتيجة :** هذه الخوارزمية من أكثر الخوارزميات تعقيداً لأنها يجب معرفة كل الوسطاء السابقة .

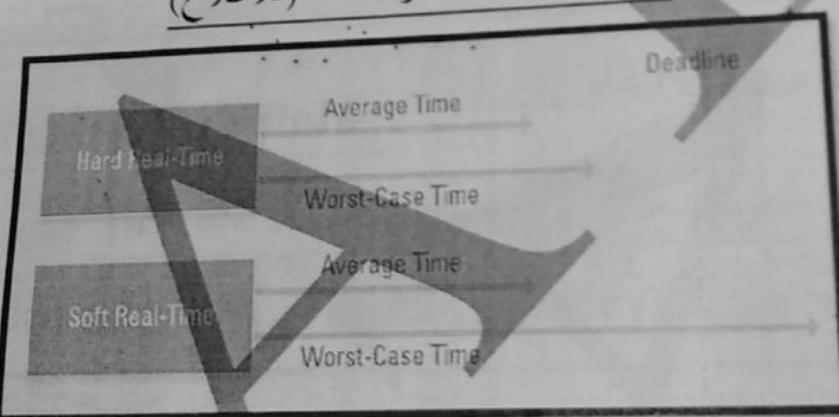
ملاحظة : لدينا خوارزمية MLQ (متعددة المستويات العادية) ولدينا خوارزمية MLFQ (متعددة المستويات بتغذية راجعة) حيث أنه يوجد مثال عن خوارزمية MLQ في الصفحة (18)



## جدولة المعالج في نظم الزمن الحقيقي

نميز نوعين من أنظمة الزمن الحقيقي :

- نظم الزمن الحقيقي البرمجية soft real-time systems : لا يوجد ضمان للإجراة الحرجة critical:real-time process no guarantee .  
سيتم جدولتها scheduled لكي تتم المعالجة .
- نظم الزمن الحقيقي العتادية hard real-time systems : كل مهمة يجب أن تخدم في وقتها المقطوع deadline (أقصى حد مسموح) وهو زمن محدد من أجل كل إجرائية نرمز له d  
مقارنة بين soft و hard (للاطلاع)



مثال عن الجدولة في نظم الزمن الحقيقي :

الجدولة وفق أقرب موعد نهائي EDF=Earliest Deadline First

يتم إعطاء الأولويات للإجراءات حسب الموعد النهائي (deadline) لكل إجرائية بحيث :

الموعد النهائي للإجراء أقرب (earlier)  $\Leftarrow$  يتم إعطاء أولوية أعلى (higher priority)

والعكس صحيح أي أنه :

الموعد النهائي أبعد (later)  $\Leftarrow$  يتم إعطاء أولوية أقل (lower priority)

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

تعلمنا إلى الآن ... كيفية جدولة وحدة المعالجة المركزية في

### النظم ذات المعالج الواحد

الآن سوف نتعلم كيف يمكننا الجدولة في حال وجود أكثر من معالج ...

### جدولة المعالجات المتعددة

تصبح مهمة جدولة وحدة المعالجة أمراً أكثر تعقيداً more complex عندما يكون هناك أكثر من معالج .

هناك شيء اسمه Multiprogramming وهو يعني البرمجة المتعددة (تعرفنا عليها سابقاً)

الآن يوجد ما يسمى بـ Multiprocessing وهو يعني المعالجة المتعددة (أكثر من معالج)

### المعالجة المتعددة : MultiProcessing

يعني وجود وحدة معالجة مركبة central processing unit أو أكثر ، ضمن نفس النظام الحاسوبي .

ملاحظة هامة : يجب التمييز بين

النظام الحاسوبي computer system وبين نظام التشغيل operating system

• نميز نوعين من المعالجة المتعددة MultiProcessing

معالجة متعددة غير متوازنة (AMP) Asymmetric MultiProcessing

معالجة متعددة متوازنة (SMP) Symmetric MultiProcessing

## (AMP) Asymmetric Multiprocessing

أولاً : المعالجة المتعددة غير المتناظرة **Multiprocessing** لا يتم معاملة كل المعالجات all CPUs بالتساوي equal ، بحيث أنه يوجد معالج وحيد only one (المعالج الرئيس master processor) يتحكم بالمعالجات الباقيه ويقوم بكل مهام tasks نظام التشغيل ، وهو الذي يتحكم بكل قرارات الجدولة scheduling decisions ، وهو الوحيد الذي يستطيع الوصول إلى كل بنى بيانات النظام system data structures لخفيف alleviating الحاجة إلى تبادل البيانات data sharing بين المعالجات .

## (SMP) Symmetric MultiProcessing

كل معالج يملك جدولته الخاصة self-scheduling ، أي أنها أمام حالتين :

1\_ كل الإجراءات موجودة في رتل جاهزيات مشترك common ready queue

بال التالي يتوجب على نظام التشغيل OS أن يتأكد أن معالجين لن يختاروا نفس الإجرائية same process

2\_ كل معالج لديه رتل جاهزيات خاص به private ready queue

يقوى أحد المعالجات خاماً (غير نشط) idle وبالتالي تحتاج إلى موازنة الحمل load balancing .

في حالة SMP تحتاج إلى توافر عمل لكل معالج all CPUs loaded لكي يحافظ على كفاءة العمل

efficiency ، وهذه بعض الطرق للقيام بذلك :

- موازنة الحمل Load Balancing : تقوم بالحفاظ على عبء العمل workload موزعاً بالتساوي

- دفع الإجراءات Push migration : يوجد مهمة task تقوم بتفحص حمل check load

كل معالج بشكل دوري periodic ، وإن وجد معالج عليه حمل زائد CPU overloaded يتم نقل

الإجراءات إلى معالج آخر أقل حملاً .

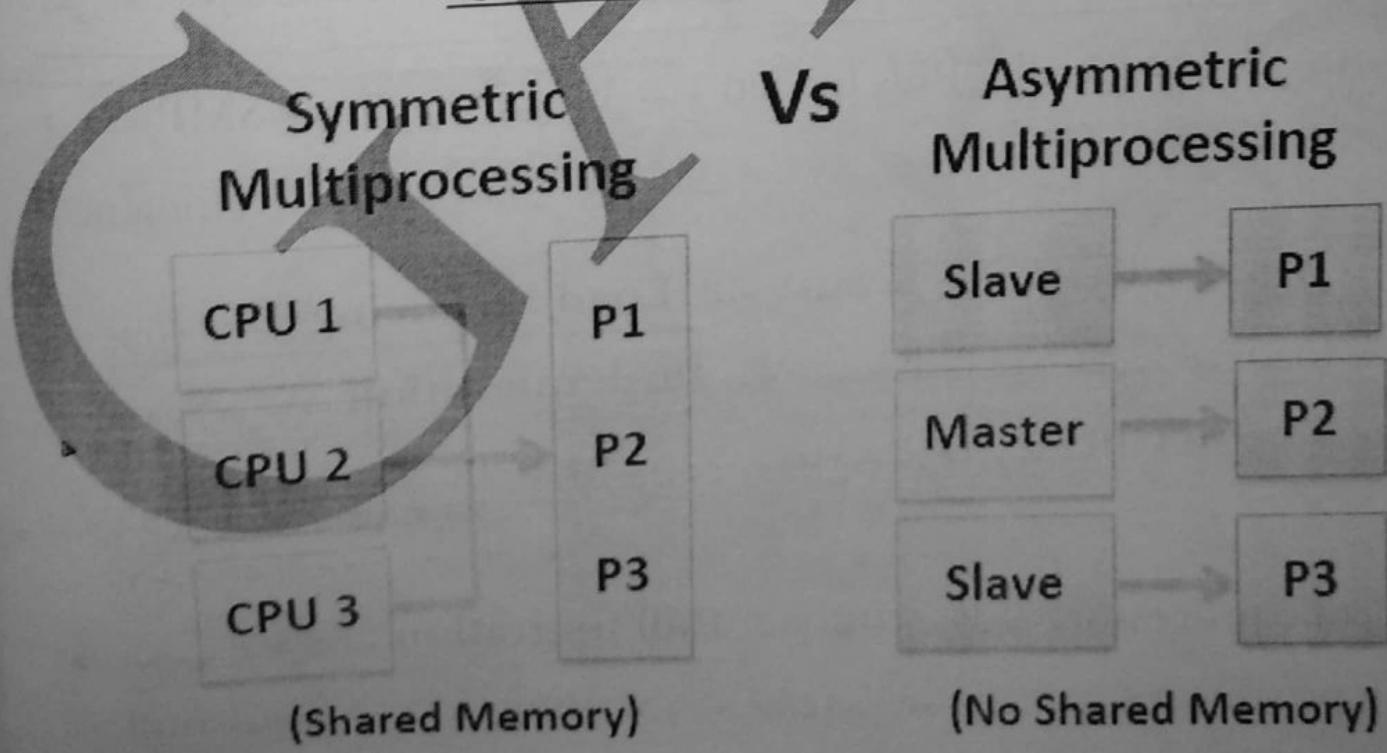
- سحب الإجراءات Pull migration : تقوم المعالجات الخاملة idle processors بسحب

- pull المهام المنتظرة waiting tasks (الإجراءات) من المعالجات المشغولة busy processor

## مقارنة بين SMP و AMP (ملخص)

المعالجة المتعددة غير المتناظرة AMP	المعالجة المتعددة المتناظرة SMP
لا تعامل بشكل متساوي	كل المعالجات تعامل بشكل متساوي
مهام النظام يقوم بها المعالج الرئيسي master	كل معالج يقوم بمهامه الخاصة individual
لا يوجد تواصل بين المعالجات لأنه يتم التحكم بها من قبل المعالج الرئيسي master	تواصل المعالجات مع بعضها عن طريق الذاكرة المشتركة shared memory
الإجرائية إما أن تكون master (تابعة للرئيس) أو أن تكون slave (تابعة للمعالجات الأخرى)	يتم أحد الإجرائية من رتل الجاهزية ready queue وكل الإجرائيات متشابهة
رخيصة وشغالتها فاضية cheaper	مكلفة وغالبة الثمن high cost
من السهل تصميمها easy to design	من الصعب تصميمها complex to design

## مقارنة بين AMP و SMP (للاطلاع)



## مثال إضافي على خوارزمية MultiLevel Queue (صعب) :

سوف نقوم بتطبيق خوارزمية الجدولة MLQ بحيث يوجد لدينا 2 رتل ، الرتل الأول Q1 له أولوية أعلى من الرتل الثاني Q2 ، يتم استخدام خوارزمية الجدولة Round Robin في الرتل Q1 و Q2 أيضاً بحيث أن بكل رتل له طول الشريحة الزمنية الخاص به  $T_{q1} = 4 \text{ ms}$  ،  $T_{q2} = 3 \text{ ms}$  .

ولدينا جدول الإجراءات التالي :

Process	Burst Time	Arrival Time	Priority queue
P1	10	0	2
P2	7	3	1
P3	6	4	2
P4	5	12	1
P5	8	18	1

المطلوب رسم مخطط غانت gant chart وحساب :

- زمن الاستجابة الوسطي AVG RT
- زمن الانتظار الوسطي AVG WT
- زمن الدورة الكلية الوسطي AVG TAT

ملاحظة هامة : صحيح أنه يوجد لدينا 2 رتل ، لكن لدينا معالج CPU واحد وبالتالي بحسب أولوية الرتل

سوف نقوم بالتنفيذ ضمن المعالج ، مع ملاحظة أن كل رتل

يطبق خوارزمية RR خاصة به وبالتالي أصبحنا نطبق

خوارزمية ضمن خوارزمية لحل هذه المشكلة سندخل في

معامرة شديدة وسنحتاج إلى فين وجاك للقيام بذلك .

## نظم سعيل المحاضره السادسه (حواجز مباب الجداوله 2)

سوف نتعرف على مفهوم جديد وهو  $RT = \text{Response Time}$

وهو الزمن الفاصل بين وصول الإجرائية  $AT$  و بدء تنفيذها لأول مرة .

أي أن  $RT = WT + \text{Remaining}$  لأن  $WT$  هو حتماً أصغر من  $RT$

زمن الانتظار = زمن الاستجابة (حتى بدأت أول تنفيذ) + مجموع أزمنة الانتظار حتى انتهت بالكامل

الآن لدينا  $t=0$  باللحظة :

لدينا الإجرائية  $P1$  قد وصلت في اللحظة  $t=0$  وهي تتبع للرتب  $Q2$  (أقل أولوية و  $q=3\text{ms}$ )

بالتالي سوف تتفقد على الأكثر  $3\text{ ms}$  ثم تخرج (طول رشقتها  $BT=10$ ) وبالتالي سيتبقى فيها  $7$

<u><math>Q1</math></u>			
<u><math>Q2</math></u>	...	...	$P1=7$

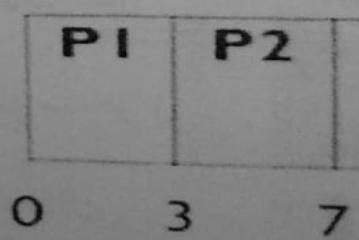
الآن نحن في اللحظة  $t=3$  ونلاحظ أن الإجرائية  $P2$  قد وصلت الآن وهي تتبع للرتب  $Q1$  (أعلى أولوية و  $q=4\text{ ms}$ )

لكن في اللحظة  $t=4$  تصل الإجرائية  $P3$  (تتبع للرتب  $Q2$  و  $BT=6$ ) لكن هذا لا يؤثر (لماذا؟؟؟).

لاحظ أن الإجرائية  $P3$  سوف توضع في نهاية الرتب  $Q2$  وليس في بدايتها !!!

<u><math>Q1</math></u>		$P2=7$	
<u><math>Q2</math></u>		$P1=7$	$P3=6$

بالتالي سوف تتفقد الإجرائية  $P2$  على الأكثر  $4\text{ ms}$  ( $BT=7$ ) ثم تخرج وبالتالي يتبقى فيها  $3$

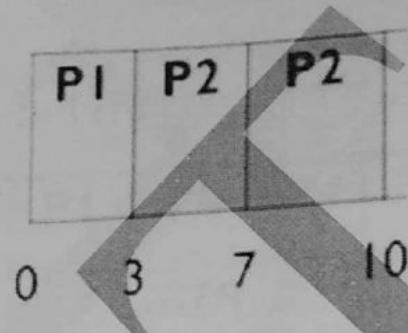


## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

<u>Q1</u>	P2=3	
<u>Q2</u>	P1=7	P3=6

الآن نحن في اللحظة  $t=7$

ونلاحظ أنه لدينا إجرائيتين P1, P3 (ضمن الرتل Q2) والإجرائية P2 (ضمن الرتل 1 Q1) وبالتالي سوف تعود وتنفذ الإجرائية P2 زمناً قدره فقط 3 (لماذا ؟؟؟).



الآن الإجرائية P2 انهت تنفيذ كل رشقاتها CPU BURST

الآن نحن في اللحظة  $t=10$  ولم يأتي أحد بعد ...

<u>Q1</u>		
<u>Q2</u>	P1=7	P3=6

نلاحظ أن الرتل Q1 فارغ ، لكن الرتل Q2 يحوي في بدايته الإجرائية  $P1=7$

لاحظ أن الرتل Q2 شريحة الزمن فيه  $q=3ms$  وبالتالي من المفترض أن تنفذ الإجرائية P1 زمناً قدره 3ms لكن نلاحظ أنه في اللحظة 12 (بعد ثانيتين من بدء التنفيذ) أتت الإجرائية P4 (BT=5) التي تتبع للرتل Q1 ذو الأولوية العليا ...

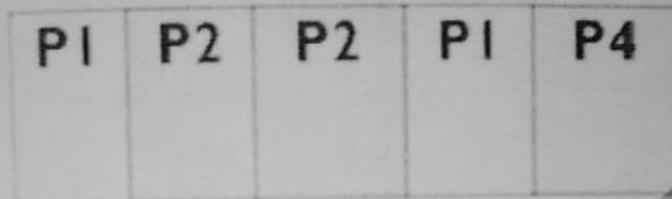
بالتالي سوف تخرج الإجرائية P1 (يتبقي فيها  $5 - 3 = 2ms$ ) ولا تنسى وضعها في آخر الرتل !!!

<u>Q1</u>	P4=5	
<u>Q2</u>	P3=6	P1=5

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)



وتوضع بدلاً منها الإجرائية P4 (BT=5) لتنفذ زمناً قدره 4 ms.



0 3 7 10 12 16 ...

الآن نحن في اللحظة  $t=16$

سوف تخرج الإجرائية P4 (يتبقى فيها  $1 \text{ ms}$ )

<u>Q1</u>	P4=1	
<u>Q2</u>	P3=6	P1=5

لكن في الرتل Q1 لا يوجد سوى الإجرائية P4 (بعض النظر عن الرتل Q2)

بالتالي سوف توضع الإجرائية P4 محدداً لزمن قدره 1ms (لم يبقى غيره أصلاء).

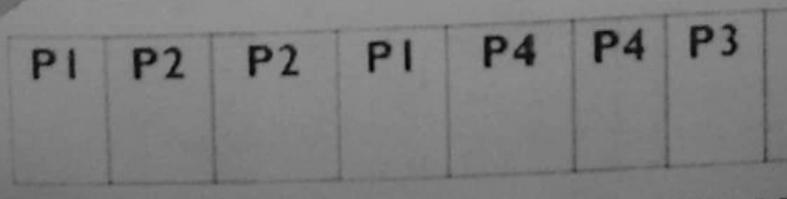
<u>Q1</u>		
<u>Q2</u>	P3=6	P1=5

الآن نحن في اللحظة  $t=17$  ولم يأتي أحد جديد ...

بالتالي سوف تنفذ الإجرائية P3 زمناً قدره على الأكثر 3 ms لكن !!!

في اللحظة  $t=18$  تأتي الإجرائية P5 التي تتبع للرتل Q1 ومتلک طول رشقة (BT=8)

بالتالي سوف تخرج الإجرائية P3 (نفذت فقط 1 ms) وتوضع في نهاية رتلها :



0 3 7 10 12 16 17 18

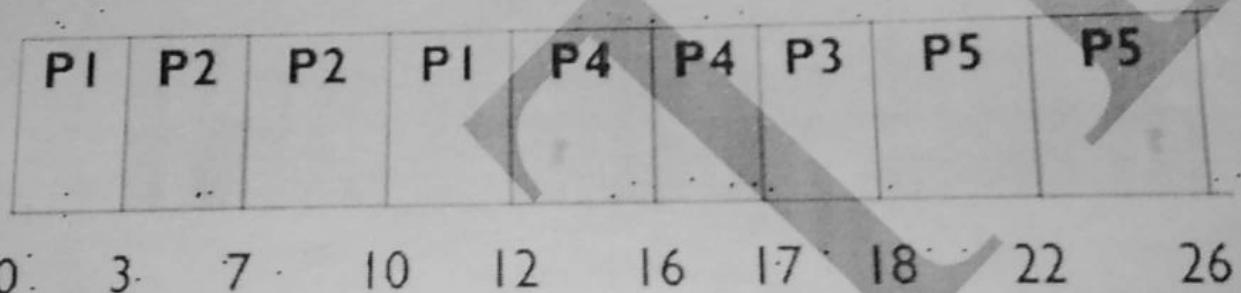
## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)

<u>Q1</u>	P5=8	
<u>Q2</u>	P1=5	P3=5

الآن سوف تنفذ الإجرائية P5 زمناً قدره 4 ms على الأكثر

<u>Q1</u>	P5=4	
<u>-Q2</u>	P1=5	P3=5

وسوف تعود وتنفذ من جديد حتى تنتهي تماماً أي يصبح لدينا :



<u>Q1</u>		
<u>Q2</u>	P1=5	P3=5

الآن سوف تنفذ الإجرائية P1 زمناً قدره 3 ms على الأكثر ولا ننسى أن نضعها في نهاية الرتل

<u>Q1</u>		
<u>Q2</u>	P3=5	P1=2

الآن سوف تنفذ الإجرائية P3 زمناً قدره 3 ms على الأكثر ولا ننسى أن نضعها في نهاية الرتل

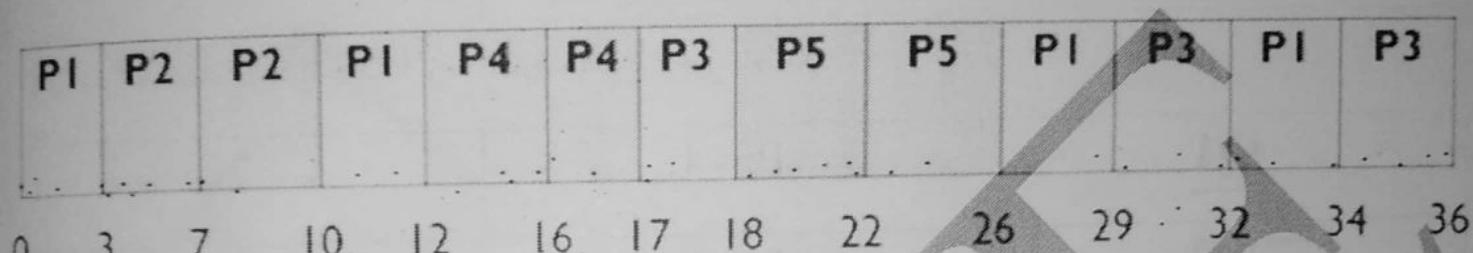
<u>Q1</u>		
<u>Q2</u>	P1=2	P3=2

## نظم تشغيل المحاضرة السادسة (خوارزميات الجدولة 2)



الآن نحن في اللحظة 32 ... سوف تنفذ الإجرائية P1 زمناً قدره 2 ms فقط

ويتبقى لدينا الإجرائية P3 أيضاً ستنتهي زمناً قدره 2 ms فقط وبالتالي يصبح لدينا :



بالعودة إلى الجدول الأساسي نلاحظ أن كل إجرائية عندما وصلت بدأت التنفيذ مباشرةً أي أن  $RT=0$

إلا الإجرائية P3 وصلت في اللحظة  $t=4$  ولم تبدأ التنفيذ إلا في اللحظة  $t=17$  وبالتالي  $RT=13$

إن الأزمنة WT و TT تعلم كيف يتم حسابهم وإيجاد المتوسط الحسابي avg

Process	Burst Time	Arrival Time	Priority queue	RT	WT	TT
P1	10	0	2	0	24	34
P2	7	3	1	0	0	7
P3	6	4	2	13	26	32
P4	5	12	1	0	0	5
P5	8	18	1	0	0	8
			AVG	2.6	10	17.2

Special thanks to Ahmad Mokayes

انتهت المحاضرة

بالتوفيق ☺

