BIRZEIT UNIVERSITY

College of Engineering and Technology

Computer Science Department

Each question 20 pts. Total 5*20 = 100

Name ---------- Number ----------

Comp431 Midterm Exam Spring 15/16

20 10

[1] (a) Circle True or False. Explain your answer in less the 3 lines. 1 pt for T, F; 1.5 for explanation

- Each thread has its own stack. (T) F

Each thread is a process (LWP) which has its own Registers set and stack, it shares with its peer threads computer ressources

- Starvation implies deadlock. T (F)

In starvation the system is running normally but some processes are starving (waiting very long time). But in deadlock, the system is blocked and processes are running

- Shortest Job First (SJF) is the "**optimal**" scheduling algorithm, but it is generally not implemented directly, due to excessive context switching overhead. T (F)

SJF is not implemented because there is no way to decide the next cpu burst duration.

- Using a smaller page size decreases the size of the page table. T (F)

smaller page size increases the size of the page table.

10 **(b)** In a paging system, the page table is **kept in memory** with memory access 100 nans, How long the CPU needs to access an instruction?

we need two memory accesses.

4 $2 * 100 = 200$ nans

If **associative registers** are added to the system with lookup time t = 5 nans and the EAT for the instruction becomes 109 nans. Compute the hit ratio **h**.

$$EAT = (1-h) * (200+5) + h * (100+5)$$

$$109 = 205 - 205h + 105h$$

6 $$109 = 205 - 100h$$

$$100h = 96$$

$$h = 0.96$$

23 **[2] (a)** Briefly explain **deadlock prevention**?

Deadlock prevention is to make sure at least one of the 4 necessary conditions do not hold.

10 (1) Mutual Exclusion. some resources are mutually exclusive by nature & can't be shared (printer, tape)

(2) Hold & wait: Give the process all its resources when it has none when the resources are available.

(3) No preemption: If the process request a resource which is not available, it must release all instances of the resource

(4) Circular wait: Apply some order of the resources, the process can request the resources in increasing order only.

(b) 4 processes P_1, P_2, P_3, P_4 and 3 resource types R_1, R_2, R_3. **All the available** resources instances in the system are (3, 2, 2) units. A snap shot of the system looks like:

P_1 holds one unit of R_1 and request one unit of R_2.

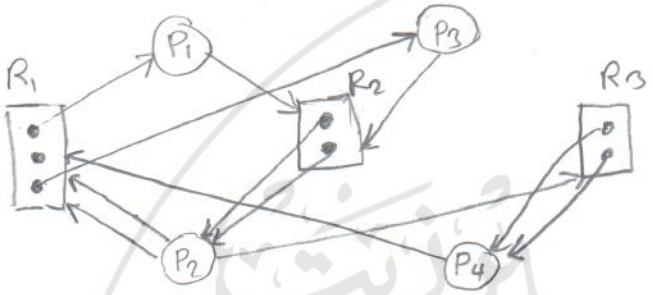
P_2 holds 2 unit of R_2 and request 2 units of R_1 and request one unit of R_3 .

P_3 holds one unit of R_1 and request one unit of R_2.

P_4 holds 2 unit of R_3 and request one unit of R_1.

13

- Compute the available resources vector after allocation.
- Compute the matrix Needs.
- Is the system is safe. Is there a deadlock. Show your work.



- Available Ressources (1, 0, 0)

matrix Allocation

	R_1	R_2	R_3
P_1	1	0	0
P_2	0	2	0
P_3	1	0	0
P_4	0	0	2

- matrix Needs :

	R_1	R_2	R_3
P_1	0	1	0
P_2	2	0	1
P_3	0	1	0
P_4	1	0	0

- No, it is not safe. There is no safe sequence.

Available

1	0	0
1	0	2

$< P_4 ,$

[3] (a) What is the difference between the followings, give examples:

- **Best fit** and **best available fit** in dynamic regions.

Best fit: Finds the smallest region to fit the job only.

Best available fit: If the smallest region that fits the job is not available, finds any smallest region that fits

- **Internal** and **external** fragmentation in fixed regions.

Internal fragmentation: The remaing unused space in the allocated fixed Region.

External fragmentation: The unused fixed region which small that do not fit any job at this minute.

(b) Given the following ready queue.

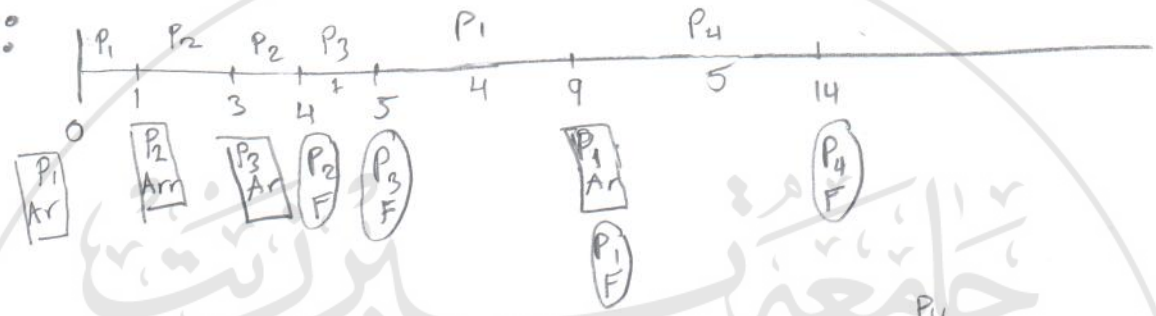
10

process	Arrival Time	Burst Time
p1	0	5
p2	1	3
p3	3	1
p4	9	5

Compute the average **turnaround time** for the SRTF and RR with Q=1.

SRTF:

6

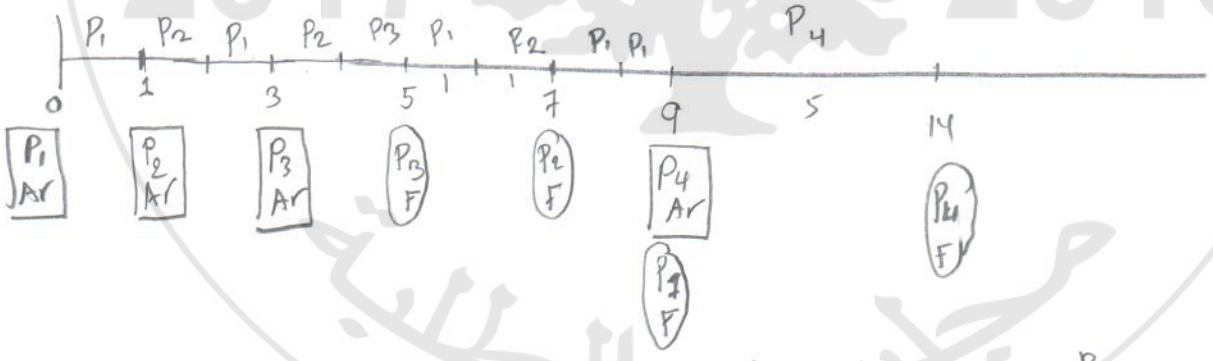


Average Turnaround Time $= \frac{P_1 + P_2 + P_3 + P_4}{4} = \frac{(9-0) + (4-1) + (5-3) + (14-9)}{4}$

$= \frac{9+3+2+5}{4} = \frac{19}{4}$

5

RR, Q=1:



Average Turn-around Time $= \frac{P_1 + P_2 + P_3 + P_4}{4} = \frac{(9-0) + (7-1) + (5-3) + (14-9)}{4}$

$= \frac{9+6+2+5}{4} = \frac{22}{4}$

[4] **(a)** Write down the code for ***wait(S)***, declare any data you need. Show with an example why ***wait(S)*** must be executed **atomically**.

- int S=1;
  wait(S) : while (S<=0)
  do-nothing;
  S=S-1;

- If wait(S) is not executed atomically, say P_1 & P_2 executes wait(S) concurrently, that is, say, P_1 executes S=S-1, P_2 executes S=S-1 concurrently, then both enter their critical region, which is a violation of mutual execlusion condition

(b) Given the following:

```
boolean Flag[2];
Flag[0]=Flag[1]=F;
```

```
Process(pi)
Repeat
  While ( Flag[(i+1) %2] == T)
      Do-nothing;
  Flag[i]=T;
  Critical—section;
  Flag[i]=F;
  Remainder section;
Forever
```

Do you think the above code solves the **critical section** problem requirements for **two** processes. Explain.

No, will not solve the problem.

1- Mutual Exclusion - not ok, Both execute the while statements & enter their critical section.

2- Progress - ok. No process will wait if critical sector empty

3. Bounded waiting - ok. (one go each wait)

P d

[5] **(a)** If the LA is 24 bits long and given the LA = 0000 0000 0100 0001 0000 1010 , and page size = 8192 word, and given the page table , **compute**:

- **p** and **d** without using the / and **%** operations.

0	10
1	150
P → 2	110
3	10
.	.
.	.
	100

P = 0000 0000 010 = 2

d = 0 0001 0000 1010 =

$2^8 + 2^3 + 2^1 = 256 + 8 + 2$

$= 266$

- **PA**

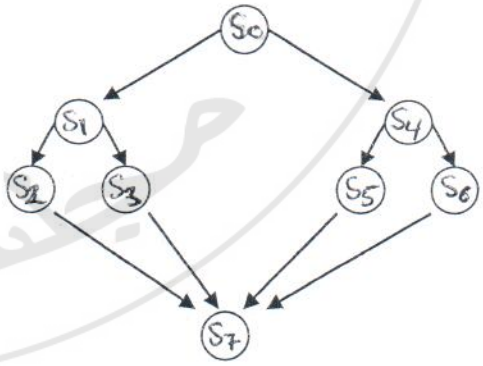
PA = PageSize × F + d

$= 2 * 110 + 266$

$= 901386$

(b) Given the precedence graph:

Write an equivalent code using **parbegin** & **parend**




```
BEGIN
  S0;
  Parbegin
    BEGIN
      S1;
      Parbegin
        S2;
        S3;
      Parend
    END
    BEGIN
      S4;
      Parbegin
        S5;
        S6;
      Parend;
    END
  Parend
END S7
```