#Midterm Solutions#

(1) **Operating System**: The software that acts as an intermediary between computer user and computer hardware
→ Software that manages the computer-hardware
Ex: UNIX, Windows, MAC OS, Linux,

Pipeline: mechanism for inter-process communication using message passing.
Ex:

(2) a) A multi-threades process has two program counters per thread. F
b) The short-term scheduler controls the degree of programming F
c) Interrupt-driven I/O provides better performance when moving large amounts of data than DMA. F
d) $@ symbol in a makefile represents the left side of the : symbol. F?
e) A child process can only be an orphan process while its parent can be either orphan or a zombie process. F
f) There must be a space character in the beginning of any command in a Makefile. T
g) When using fork system call, parent and child process have the same address space. T
h) With NUMA, some parts of memory may take longer to access than other parts. T
i) CD-R and DVD-R are examples for WORM devices. T
j) CPU registers has faster access time than any other device including CPU cache memory T
k) Privilege escalation allows user to change file access permittan. T
l) Emulation used when source CPU type different from target type. T
m) Operations in Message Passing architecture is faster than shared memory architectures F

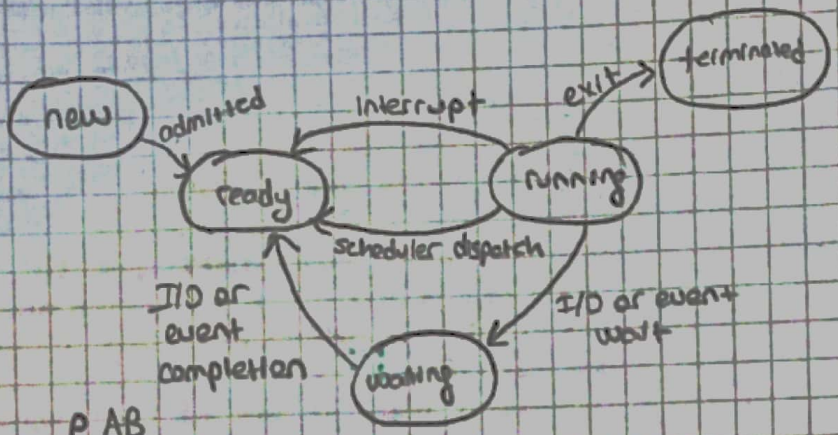
(3) a) A trap or exception is a software-generated interrupt caused either by an error or a user request.
b) When an interrupt occurs, the OS preserves the state of the CPU by storing every registers and the program counter.
c) The load instruction moves a byte or word from main memory to an internal register within the CPU while the store instruction moves the content of a register to main memory.
d) The sequence of steps that the CPU follows to process instructions is called as [redacted] instruction cycle.
e) In a multiprocessor environment all CPUs must have the most recent value in their cache which is known as cache coherence.
f) In the context of Cloud Computing, Google Docs is an example for SaaS
g) One of the example shell program in a Linux/UNIX systems is Bash Shell.
h) One method for system call parameter passing is to use registers.
i) PID value of 1 is assigned to the [redacted] init process on Linux systems.

(4) core dump = Failure of an application can generate a core dump capturing memory of process.

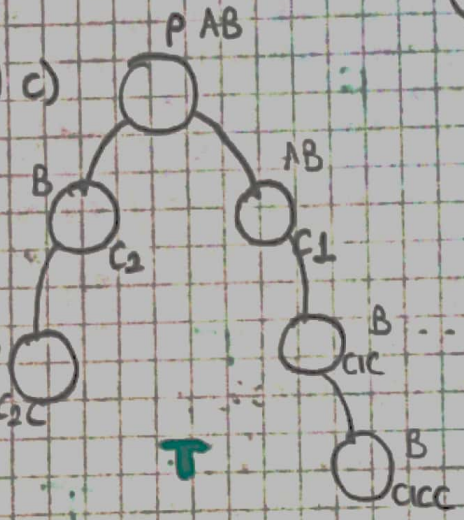
crash dump = OS failure can generate a crash dump containing kernel memory.

(5)

(6)



(7) c)



a) A: 2 times
b) B: 6 times

(8) Shared variables ⇒ int x = 2;
int y = 0;

Process A

```
while (x == 2);
printf("E");
y = 1;
x = 0;
printf("M");
y = 1;
```

Process B

```
printf("L");
x = 1;
while (y == 0);
printf("A");
```

(9)

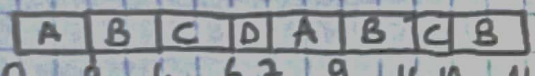
```
P1:
  T1;
  signal(S);
P2:
  wait(S);
  T2;
```

#Final Exam#

①

Pid	Submission Time (Arrival Time)	Required Time (Burst)
A	0	4
B	1	6
C	3	3
D	4	1

a) Round-Robin. Quantum of 2. Response, waiting and TAT for each process? → preemptive



Waiting Times
A= 0+5=5
B= 1+5+1=7
C= 1+5=6
D= 2

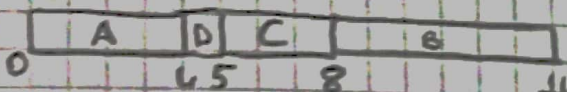
Response Times (ilk kez kacta calistı - Arrival)
A=0
B=1
C=1
D=2

Completion Times
A=9
B=14
C=12
D=7

Turn-Around-Times
A=9
B=13
C=9
D=3

- Turn around time = Required Time + Waiting Time
- Completion = Kacta bitti? Time
- Response = First Time Executed − Arrival Time

b) Shortest-Job First? → Can be both



Waiting Times	Response Time	TAT	Completion Time
A=0	A=0	A=4	A=4
B=7	B=7	B=13	B=14
C=2	C=2	C=5	C=8
D=0	D=0	D=1	D=5

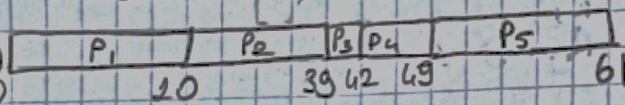
c) First Come First Serve → Non preemptive

Waiting Times	Response Time	TAT	Completion Time
A: 0	A: 0	A: 4	A: 4
B: 3	B: 3	B: 9	B: 10
C: 7	C: 7	C: 10	C: 13
D: 9	D: 9	D: 10	D: 14

(2) a) FCFS
b) SJF
c) RR, q=5

Process	Burst
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

a)



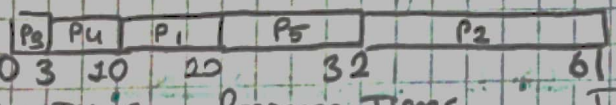
Waiting Times
$P_1 = 0$
$P_2 = 10$
$P_3 = 39$
$P_4 = 42$
$P_5 = 49$

Response Times
$P_1 = 0$
$P_2 = 10$
$P_3 = 39$
$P_4 = 42$
$P_5 = 49$

TAT
$P_1 = 10$
$P_2 = 39$
$P_3 = 42$
$P_4 = 49$
$P_5 = 61$

Completion Times
$P_1 = 10$
$P_2 = 39$
$P_3 = 42$
$P_4 = 49$
$P_5 = 61$

b)



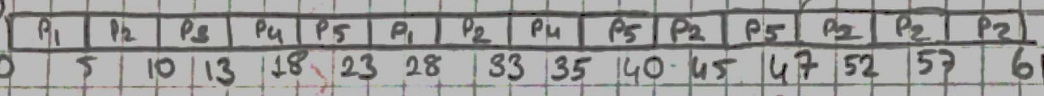
Waiting Times
$P_1 = 10$
$P_2 = 32$
$P_3 = 0$
$P_4 = 3$
$P_5 = 20$

Response Times
$P_1 = 10$
$P_2 = 32$
$P_3 = 0$
$P_4 = 3$
$P_5 = 20$

TAT
$P_1 = 20$
$P_2 = 61$
$P_3 = 3$
$P_4 = 10$
$P_5 = 32$

Completion Times
$P_1 = 20$
$P_2 = 61$
$P_3 = 3$
$P_4 = 10$
$P_5 = 32$

c)



Waiting Times
$P_1 = 0 + 18 = 18$
$P_2 = 5 + 18 + 7 + 2 = 32$
$P_3 = 10$
$P_4 = 13 + 15 = 28$
$P_5 = 18 + 12 + 5 = 35$

Response Times
$P_1 = 0$
$P_2 = 5$
$P_3 = 10$
$P_4 = 13$
$P_5 = 18$

TAT
$P_1 = 28$
$P_2 = 61$
$P_3 = 13$
$P_4 = 35$
$P_5 = 47$

Completion Times
$P_1 = 28$
$P_2 = 61$
$P_3 = 13$
$P_4 = 35$
$P_5 = 47$

(3) Effective Access Time

α : %90
ε : 10 ns for TLB
250 ns for memory access

$\alpha = 0.9$
$\varepsilon = 10$ ns
$m = 250$ ns

$$EAT = \alpha(\varepsilon + m) + (1 - \alpha)(\varepsilon + m + m)$$

$$0.9(260) + (0.1)(270)$$

$$= 285 \text{ ns}$$

④

Max

	P_1	P_2	P_3	P_4
R_1	1	6	3	4
R_2	2	1	1	2
R_3	2	3	4	2

Allocated

	P_1	P_2	P_3	P_4
R_1	1	6	2	0
R_2	0	1	1	0
R_3	0	2	1	2

Available

R_1	0
R_2	1
R_3	1

Total

R_1	9
R_2	3
R_3	6

a) Need matrix?
b) Is the system safe?

a)

Need

	P_1	P_2	P_3	P_4
R_1	0	0	1	4
R_2	2	0	0	2
R_3	2	1	3	0

b)

Available

	R_1	R_2	R_3
→	0	1	1
P_2	6	2	3
P_3	8	3	4
P_4	8	3	6
P_1	9	3	6

$<P_2, P_3, P_4, P_1>$

∴ Safe.

⑤

Allocation

	A	B	C	D	E
P_0	2	0	0	1	0
P_1	0	1	2	1	1
P_2	2	1	0	3	1
P_3	1	3	1	0	2
P_4	1	4	3	2	2

Max

	A	B	C	D	E
P_0	4	1	1	2	3
P_1	3	2	5	2	3
P_2	2	3	1	7	3
P_3	1	4	2	4	4
P_4	3	6	6	5	4

Available

	A	B	C	D	E
	3	3	3	1	2
P_1	3	4	5	2	3
P_0	5	4	5	3	3
P_4	6	8	8	5	5
P_2	8	9	8	8	6
P_3	9	12	9	8	8

a) Total matrix?) How many resource are there for each resource type in this system?
b) Is this system safe?
c) If a request from P_4 arrives for (0,0,2,0,1) can the request be granted immediately?

Need

	A	B	C	D	E
P_0	2	2	1	1	3
P_1	3	1	3	1	2
P_2	0	2	1	4	2
P_3	0	1	1	4	2
P_4	2	2	3	3	2

a) 9, 12, 9, 8, 8 for A, B, C, D, E respectively.

b) Yes. $<P_1, P_0, P_4, P_2, P_3>$ is safe sequence.

c) Request ≤ need
request ≤ allocation → available resource is 5, 4, 5, 3, 3 at request time. It can easily granted

allocation += request
need -= request
?

(6) Prioritized round robin.

a) Suppose the scheduler gives higher priority to processes that have larger quanta.
Is starvation possible in the system?

→ No. Although priority scheduling may cause starvation, we have a round-robin. So when higher priority processes are done with q or new processes added, they will be added to the end of the ready queue. (If process has remaining time, of course.) So at some point, lower priority will be executed.

b) Suppose instead that scheduler gives higher priority to processes that have smaller quanta. Is starvation possible in the system?

→ No. The same reason above. Each process is given quanta for execution or fixed time to execute.

(7)

1-) Only one writer writes at a time.
2-) While writing, reading is not allowed.
3-) While reading, writing is not allowed.
4-) Up to 5 readers can read at the same time.

Writer Process

```
wait (wlf);
    writing is performed
signal (wlf);
```

Reader Process

```
wait(mutex)
read count ++;
if readcount=1 then wait (wlf)
 signal (mutex);
 reading is performed
 wait (mutex)
    readcount --;
 if readcount=0 then signal (wlf);
 signal (mutex);
```

```
wait(mutex);
read count ++;
if readcount=1
   wait (wlf);
{if readcount<=5}
   signal(mutex)
{if mutex=1}
   wait(mutex)
readcount--;
 if (readcount=0)
   signal (wlf)
signal (mutex)
```

(8) Female - 3 toilet
Male - 2 toilet

2 Male toilet occurs.
Erkek yoksa kadın kullanabiliyor.
Kadın yoksa erkek " .

```
wait(semaphore)
signal(semaphore)
intSem(semaphore.value);
getValue(value)
```

```
while (true){
  wait(female);
    femalecount++
    if(femalecount == -1)
     wait(male)
     signal(female);
    wait(female)
     femalecount--
      if(female cont==0)
          signal(male);
           signal(female);
```