CSE 2431 **HOMEWORK 2** **SUMMER 2020**

KEY – CLASS

125 points total

Online submission in pdf file due: Friday, May 29th, 11:30 pm

1. CPU Scheduling

Analyze scheduling algorithms for the following five processes given the process CPU burst time, and arrival time.

Processes	CPU Burst	Arrival Time
P_0	14	0
P_1	7	2
P_2	12	4
P_3	4	6
P_4	16	8

a) FIFO/First Come First Served (FCFS) (Arrival times should be considered.)

Gantt chart:

P_0	**P_1**	**P_2**	**P_3**	**P_4**

0 14 21 33 37 53

Wait times per process

P_0: **14 – 0 – 14 = 0**

P_1: **21 – 2 – 7 = 12**

P_2: **33 – 4 – 12 = 17**

P_3: **37 – 6 – 4 = 27**

P_4: **53 – 8 – 16 = 29**

Average wait time = (0 + 12 + 17 + 27 + 29)/5 = <u>85/5</u> = 17

b) (Non-preemptive) Shortest Job First

Gantt chart:

P_0	**P_3**	**P_1**	**P_2**	**P_4**

0 14 18 25 37 53

Wait times per process
P_0: **14 – 0 – 14 = 0**
P_1: **25 – 2 – 7 = 16**
P_2: **37 – 4 – 12 = 21**
P_3: **18 – 6 – 4 = 8**
P_4: **53 – 8 – 16 = 29**

Average wait time = (0 + 16 + 21 + 8 + 29)/5 = 74/5 = 14.8

c) (Preemptive) STCF **ANSWER: There are two possible correct solutions.**

<u>One solution</u>
Gantt chart:

P_0	P_1	P_3	P_0	P_2	P_4

0 2 9 13 25 37 53

Remaining CPU burst time after each run on the CPU (not absolutely necessary to include this, but it is helpful to keep track of which process has the shortest time to completion)

P_0: **14, 12, 0**
P_1: **7, 0**
P_2: **12, 0**
P_3: **4, 0**
P_4: **16,0**

Wait times per process
P_0: **25 – 0 – 14 = 11**
P_1: **9 – 2 – 7 = 0**
P_2: **37 – 4 – 12 = 21**
P_3: **13 – 6 – 4 = 3**
P_4: **53 – 8 – 16 = 29**

Average wait time = (11 + 0 + 21 + 3 + 29)/5 = 64/5 = 12.8

<u>The other solution</u>

Gantt chart:

P_0	P_1	P_3	P_2	P_0	P_4

0 2 9 13 25 37 53

Remaining CPU burst time after each run on the CPU (not absolutely necessary to include this, but it is helpful to keep track of which process has the shortest time to completion)

P_0: **14, 12, 0**
P_1: **7, 0**
P_2: **12, 0**
P_3: **4, 0**
P_4: **16,0**

Wait times per process
P_0: **37 – 0 – 14 = 23**
P_1: **9 – 2 – 7 = 0**
P_2: **25 – 4 – 12 = 9**
P_3: **13 – 6 – 4 = 3**
P_4: **53 – 8 – 16 = 29**

Average wait time = (23 + 0 + 9 + 3 + 29)/5 = 64/5 = 12.8

d) Round Robin Scheduling (Assume that the time quantum is 4 time units). Also assume that if a new process arrives at exactly the same time as the time slice of the executing process expires the executing process is put at the end of the ready queue *after* the arriving process.

Gantt chart

P_0	P_1	P_2	P_0	P_3	P_4	P_1	P_2	P_0	P_4	P_2	P_0	P_4	P_4

0 4 8 12 16 20 24 27 31 35 39 43 45 49 53

Remaining CPU burst times after each run on CPU (Not necessary to include this)
P_0: **14, 10, 6, 2, 0**
P_1: **7, 3, 0**
P_2: **12, 8, 4, 0**
P_3: **4, 0**
P_4: **16, 12, 8, 4, 0**

Wait times per process
P_0: **(45 – 0) – 14 = 31**
P_1: **(27 – 2) – 7 = 18**
P_2: **(43 – 4) – 12 = 27**
P_3: **(20 – 6) – 4 = 10**
P_4: **(53 – 8) – 16 = 29**

Average wait time = (31 + 18 + 27 + 10 + 29)/5 = 115/5 = 23

*Note: Arrival Times should be considered for each of the scheduling algorithms. It's possible that a given algorithm will not need all the information given in the table.

*Note: Arrival Times should be considered for *each* of the scheduling algorithms. It's possible that a given algorithm will not need all the information given in the table.

2. Addressing for Paged Memory

Complete the table below by filling in the empty boxes with the correct value for number of virtual address bits; number of virtual page bits; number of page offset bits; number of virtual pages; and number of bytes in each page.

For page size, express the value in KB, MB, or GB (see note below). For number of virtual pages, write the value as a power of 2.

NOTE: Assume the following values

1 KB = 1024 bytes (2^{10} bytes)
1 MB = 1024 X 1024 bytes (2^{20} bytes)
1 GB = 1024 X 1024 X 1024 bytes (2^{30} bytes)

Virtual Address Bits	Virtual Page Bits	Page Offset Bits	No. Virtual Pages	No. Bytes per Page
32	**20**	12	**2^{20}**	**4 KB**
40	28	**12**	**2^{28}**	**4 KB**
22	**12**	**10**	2^{12}	**1 KB**
30	**19**	**11**	**2^{19}**	2 KB
48	**32**	16	**2^{32}**	**64 KB**
26	**17**	**9**	2^{17}	**½ KB (512 bytes)**

3. Virtual memory

Fill in the boxes in the table below for the approaches to virtual memory discussed in class, with T, if the characteristic applies to the approach, or F if it does not.

Characteristics

A. No mechanism for processes to share a portion of their address spaces is provided.
B. The whole process, or parts of it, can be dynamically relocated by the OS while the process is in execution (that is, between the time it is created and the time it terminates execution).
C. The whole address space of the process must be loaded into memory (but not necessarily in a contiguous sequence of bytes) in order for the process to run.
D. The whole address space for the process must be loaded into a contiguous sequence of bytes in memory in order for the process to run.
E. No MMU hardware is needed to implement the approach (no MMU or registers in an MMU).
F. No general mechanism for protecting processes from each other (i.e., preventing any process from accessing another process's address space) is provided.
G. Each segment of the process must be loaded into a contiguous sequence of bytes in memory.

Approach	Characteristic	A	B	C	D	E	F	G
Static relocation	X	**T**	**F**	**T**	**T**	**T**	**T**	**T**
Dynamic Relocation with Base	X	**T**	**T**	**T**	**T**	**F**	**T**	**T**
Dynamic Relocation with Base + bounds	X	**T**	**T**	**T**	**T**	**F**	**F**	**T**
Segmentation	X	**F**	**T**	**T**	**F**	**F**	**F**	**T**
Paging	X	**F**	**T**	**F**	**F**	**F**	**F**	**F**

4. Virtual Memory – Segmentation

A. Suppose the use of segmentation and a 18 bit virtual address space for a program running as a process with (a maximum of) 4 segments.
B. Here is the segment table

Segment	Base	Bounds	(Protection bits) RW
0	0xa200	0xffff	10
1	0x4400	0x0fff	11
2	0x2300	0x07ff	11
3	0x8200	0x08ff	11

C. Translate the following virtual addresses in the program to the corresponding physical addresses.
   i. ***Show clearly how you calculated the physical address***.
   ii. Say whether the memory reference results in a segmentation fault (invalid memory access), and ***if so, explain why clearly***.

Virtual Addresses

a. Write 0x100e2

Physical address = 0x4400 + 0x00e2 = 0x44e2
No segmentation fault (R is 1, and 0x00e2 < 0x0fff; Not required)

b. Write 0x30a00

Physical address = 0x8200 + 0x0a00 = 0x8c00
Segmentation fault, because 0x0a00 > 0x08ff

c. Write 0x00400

Physical address = 0xa200 + 0x0400 = 0xa600
Segmentation fault, because W is 0

d. Read 0x02bfc

Physical address = 0xa200 + 0x2bfc = 0xcdfc
No segmentation fault (R is 1, and 0x2bfc < 0xffff; Not required)

e. Read 0x20708

Physical address = 0x2300 + 0x0708 = 0x2a08
No segmentation fault (R is 1, and 0x0708 < 0x07ff; Not required)

The homework must be typed, including diagrams.