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

NAME: Yifan Yao.740

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

NOTE: ***Please keep a copy of your homework to study for the exam.***

You should not discuss the details of any problem, or how to approach the solution, with other students. If you have questions of this type, you should post a question on Piazza for the instructor. If your post shows part of your solution, or what you think the solution might be, you should leave your post private, which is what it will be by default.

The homework must be typed, including diagrams (for Gantt diagrams, you can use a table with 1 row and the needed number of columns).

NOTE: You should type your name above, enter your answers in this document, save it, and export to pdf to submit it on Carmen. Only pdf submissions will be accepted.

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

For each of the following scheduling algorithms draw a Gantt diagram (you can use a one row table with an appropriate number of columns), and also calculate clearly (show how you do the calculation), (a) the total wait time for each process, and (b) the average wait time.

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

Gantt chart (you can add columns if needed; make other Gantt charts below this way)

P0	P1	P2	P3	P4	
0	14	21	33	37	53

1. Total wait time for each process (Completion Time - Arrival Time - Burst Time)
    a. P0: 14 - 0 - 14 = 0
    b. P1: (14 + 7) - 2 - 7 = 12
    c. P2: (14 + 7 + 12) - 4 - 12 = 17
    d. P3: (14 + 7 + 12 + 4) - 6 - 4 = 27
    e. P4: (14 + 7 + 12 + 4 + 16) - 8 - 16 = 29
2. Average wait times
    a. (0 + 12 + 17 + 27 + 29) / 5 = 17

b) (Non-preemptive) Shortest Job First

Gantt chart (you can add columns if needed; make other Gantt charts below this way)

P0	P3	P1	P2	P4	
0	14	18	25	37	53

1. Total wait time for each process (Completion Time - Arrival Time - Burst Time)
    a. P0: 14 - 0 - 14 = 0
    b. P3: (14 + 4) - 6 - 4 = 8
    c. P1: (14 + 4 + 7) - 2 - 7 = 16
    d. P2: (14 + 4 + 7 + 12) - 4 - 12 = 21
    e. P4: (14 + 4 + 7 + 12 + 16) - 8 - 16 = 29
2. Average wait times
    a. (0 + 8 + 16 + 21 + 29) / 5 = 14.8

c) (Preemptive) STCF

Gantt chart (you can add columns if needed; make other Gantt charts below this way)

P0	P1	P3	P0	P2	P4
0	2	9	13	23	45

- P0 runs first since it arrived at 0
- P0 run 2, P1 arrives, P1 (7) < P0 (14 - 2 = 12)
- P1 run 7 and complete, P2 arrives, P3 arrives, P4 arrives, P3 (4) < P0 (12) = P2 (12) < P4 (16)
- P3 run 4 and complete, P0 (12) = P2 (12) < P4 (16)
- P0 run 12 and complete, P2 (12) < P4 (16)

- P2 run 12 and complete, P4 (16)
- P4 run 16 and complete

1. Total wait time for each process (Completion Time - Arrival Time - Burst Time)
    a. P0: (2 + 7 + 4 + 12) - 0 - 14 = 11
    b. P1: (2 + 7) - 2 - 7 = 0
    c. P2: (2 + 7 + 4 + 12 + 12) - 4 - 12 = 21
    d. P3: (2 + 7 + 4) - 6 - 4 = 3
    e. P4: (2 + 7 + 4 + 12 + 12 + 16) - 8 - 16 = 29
2. Average wait times
    a. (11 + 0 + 21 + 3 + 29) / 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 (you can add columns if needed; make other Gantt charts below this way)

P0	P1	P2	P0	P3	P4	P1	P2	P0	P4	P2	P0	P4	
0	4	8	12	16	20	24	27	31	35	39	43	45	53

- P0 runs first since it arrived at 0
- P0 run 4 (10 left), P1 arrives, P2 arrives
- P1 run 4 (3 left), P2 run 4 (8 left), P0 run 4 (6 left), P3 arrives, P4 arrives
- P3 run 4 (0 left), P4 run 4 (12 left), P1 run 3 (0 left), P2 run 4 (4 left), P0 run 4 (2 left)
- P4 run 4 (8 left), P2 run 4 (0 left), P0 run 2 (0 left)
- P4 run 4 (4 left), P4 run 4 (0 left)

1. Total wait time for each process (Completion Time - Arrival Time - Burst Time)
    a. P0: 45 - 0 - 14 = 31
    b. P1: 27 - 2 - 7 = 18
    c. P2: 43 - 4 - 12 = 27
    d. P3: 20 - 6 - 4 = 10
    e. P4: 53 - 8 - 16 = 29
2. Average wait times
    a. (31 + 18 + 27 + 10 + 29) / 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.

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^{11}	2 KB
48	32	16	2^{32}	64 KB
26	17	9	2^{17}	512 B

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	F	F	F	T	T	T	F
Dynamic Relocation with Base	X	T	T	F	T	F	T	F
Dynamic Relocation with Base + bounds	X	T	T	F	T	F	F	T
Segmentation	X	T	T	T	F	F	F	T
Paging	X	F	T	T	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
   a. Seg: 1
   b. 0x00e2 < 0x0fff (Valid)
   c. Base + Offset = 0x4400 + 0x00e2 = 0x44e2
   d. Protection bit: 11, Read is permitted
   e. Valid Access
b. Write 0x30a00
   a. Seg: 3
   b. 0x0a00 > 0x08ff (Invalid)
   c. Invalid Access
c. Write 0x00400
   a. Seg: 0
   b. 0x0400 < 0xffff (Valid)
   c. Base + Offset = 0xa200 + 0x0400 = 0xa600
   d. Protection bit: 10, Read is permitted
   e. Invalid Access
d. Read 0x02bfc
   a. Seg: 0
   b. 0x2bcf < 0xffff (Valid)

    c. Base + Offset = 0xa200 + 0x2bcf = 0xcdcf
    d. Protection bit: 10, Read is permitted
    e. Valid Access
e. Read 0x20708
    a. Seg: 2
    b. 0x0708 < 0x07ff (Valid)
    c. Base + Offset = 0x2300 + 0x0708 = 0x2a08
    d. Protection bit: 11, Read, write is permitted
    e. Valid Access

<u>The homework must be typed, including diagrams.</u>