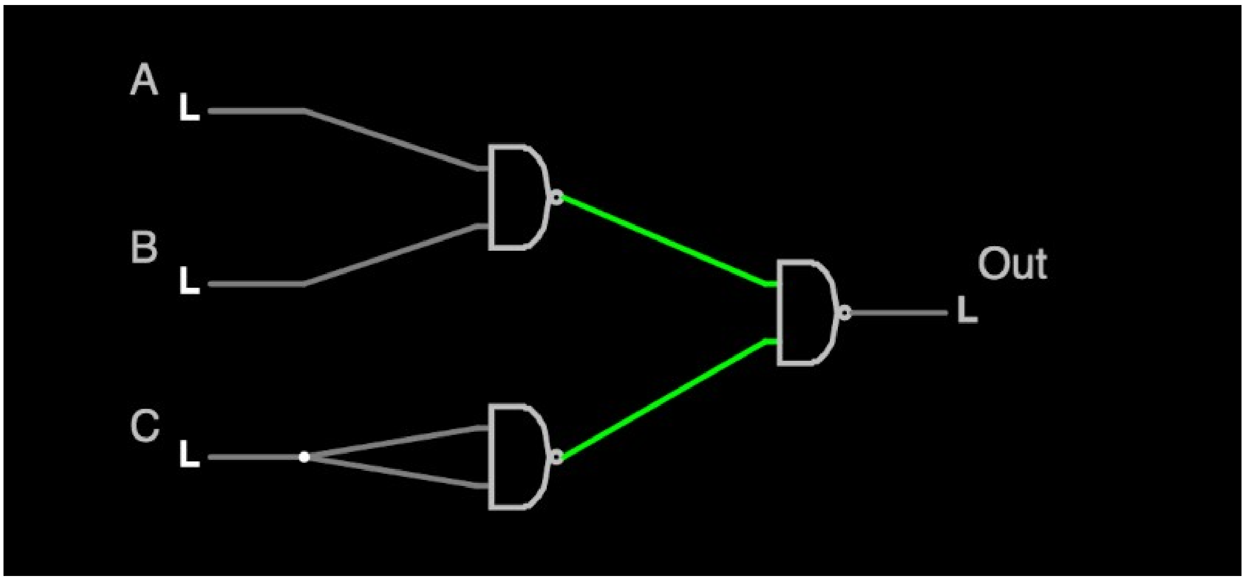
Daniel Gooden
30/07/2024
COSC240

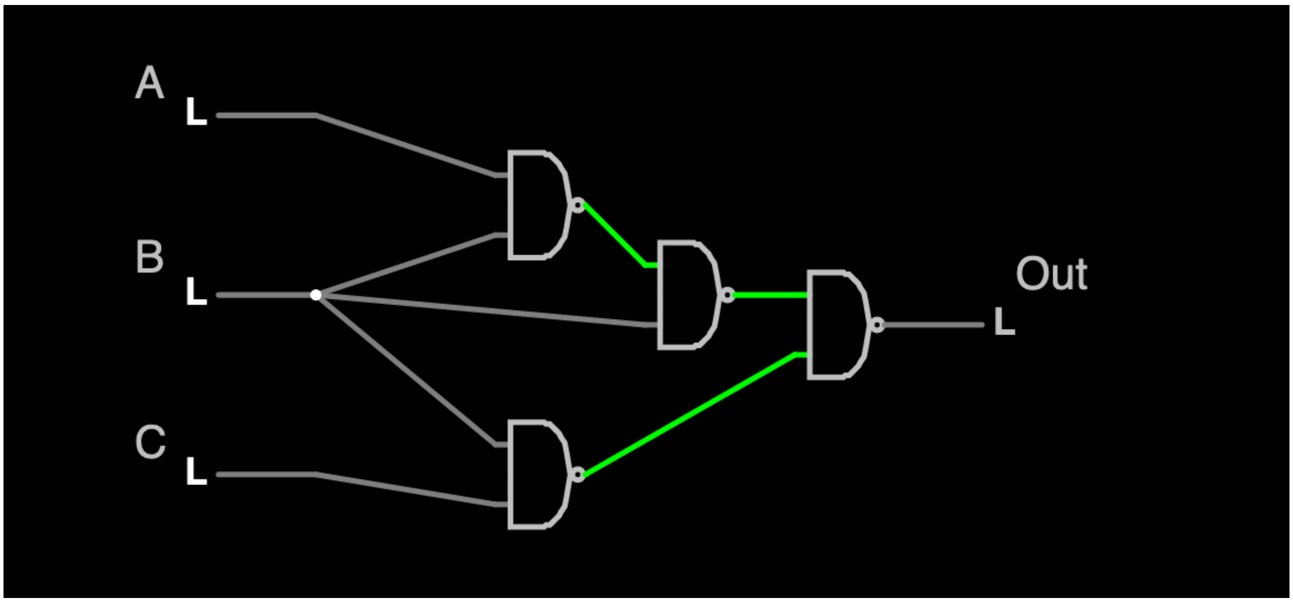
Assessment 1

Theory Assignment

1. Write a truth table for the circuits:



A	**B**	**C**	**Out**
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	1
1	1	0	1
0	1	1	1
1	0	1	1
1	1	1	1



A	B	C	Out
0	0	0	0
1	0	0	0
0	1	0	1
0	0	1	0
1	1	0	0
0	1	1	1
1	0	1	0
1	1	1	1

2. Using the 7 state process model introduced in lecture, describe each state and the transitions between them. Explain how this model developed from the simple FCFS model implemented on batch systems.

   The 7 state process model is comprised of the following states:

   1. **New:** the initial state of a process as it enters the system.
   2. **Ready:** the process is loaded into main memory and is ready to execute
   3. **Running:** the process that is currently being executed by the CPU
   4. **Blocked:** the process is waiting on an event
   5. **Ready/Suspend:** the process is in secondary memory and ready to execute once it is loaded back into main memory
   6. **Blocked/Suspend:** The process is in secondary memory and is waiting for an event to occur
   7. **Exit:** The process has finished execution and is being removed from the system

Transitions Between States:

- **New → Ready:** If main memory is available, the process moves to the Ready state
- **New → Ready/Suspend:** If main memory is not available, the process moves to Ready/Suspend
- **Ready/Suspend → Ready:** When main memory becomes available, the process moves to the Ready state
- **Ready → Running:** If operating system scheduler selects the process to execute, moving it into the Running state
- **Running → Ready:** If the process is interrupted (e.g., time slice expiration), it moves back to the Ready state.
- **Running → Blocked:** If the process needs to wait for an event (e.g., I/O operation), it moves to the Blocked state.
- **Blocked → Ready:** Once the event the process was waiting for occurs, it moves to the Ready state.
- **Ready → Ready/Suspend:** If the system needs to free up main memory, the process can be suspended and moved to secondary memory.
- **Blocked → Blocked/Suspend:** If the system needs to free up main memory, the process in the Blocked state can be suspended and moved to secondary memory.
- **Blocked/Suspend → Blocked:** When main memory becomes available, the process moves back to the Blocked state if it is still waiting for the event.
- **Blocked/Suspend → Ready/Suspend:** If the event occurs while the process is in secondary memory, it moves to the Ready/Suspend state.
- **Running → Exit:** If the process completes its execution, it moves to the Exit state.

The multi-state model was developed over many iterations. Early multi-programming systems introduced “Running” and “Not Running” so the CPU could switch processes. This evolved into the 5 state model which added a “Blocked” state to handle processes waiting on events, as we well as “New” for an entry state and “Exit” for a exiting state so that processes could be audited. This gave a marked improvement to CPU utilization as “Blocked” processes would no longer be given time on the CPU while they wait for their event to unblock. The 7 state model further refined this by allowing for “Suspended” states that can pull processes into secondary memory to free up main main memory. This improved memory management so that the CPU could handle processes that take up more memory than is available on main memory.

3. Five batch jobs, A through E, arrive in alphabetical order at a computer at almost the same time (i.e. they all arrive before the next scheduling event occurs). They have estimated running times of 100, 300, 400, 200, and 500 ms, respectively. Using diagrams, show how the process scheduling algorithms below would schedule these jobs, and calculate the mean process turnaround time (assuming no process switching overhead):

1. FCFS – First Come First Serve

0	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**	**1100**	**1200**	**1300**	**1400**	**T/A**
A															100
	B	B	B												400
				C	C	C	C								800
								D	D						1000
										E	E	E	E	E	1500

Mean process turnaround time is 760 ms

2. SPN – Shortest Process Next

0	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**	**1100**	**1200**	**1300**	**1400**	**T/A**
A															100
			B	B	B										600
						C	C	C	C						1000
	D	D													300
										E	E	E	E	E	1500

Mean process turnaround time is 700 ms

3. Round Robin with a quantum of 100ms

0	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**	**1100**	**1200**	**1300**	**1400**	**T/A**
A															100
	B				B				B						1000
		C				C				C		C			1300
			D				D								800
				E				E			E		E	E	1500

Mean process turnaround time is 940 ms

4. Round Robin with a quantum of 200ms

0	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**	**1100**	**1200**	**1300**	**1400**	**T/A**
A															100
	B	B							B						1000
			C	C						C	C				1200
					D	D									700
							E	E				E	E	E	1500

Mean processes turnaround time is 900 ms

4. Does the following pseudo-code solve the mutual exclusion problem for two processes (0 and 1)? Why or why not?

```
flag = {false, false}
process p(i):
      while (flag[1-i]) {
            // do nothing
      }
      flag[i] = true
      while (flag[1-i] {
            // do nothing
      }
      <critical section>
      flag[i] = false
```

The pseudo-code does address mutual exclusion where it will not allow both processes to enter a critical section simultaneously however there is potential for deadlock. Mutual exclusion is achieved when the program blocks multiple processes from accessing the same critical section at the same time. Deadlock is when two processes are stuck and unable to proceed. In this code snippet this can happen if both processes are false and try to run their process at the same time. They both succeed the first blocking loop as both flags are false and then set their flags to true at the same time. They will both block as both flags are set to true leading to indefinite postponement of entering the critical region due to a circular wait condition. Each process is waiting for the other to flag itself false before entering the critical section, yet they themselves are true which will result in a circular wait condition. Therefore, this pseudo-code does not effectively solve the mutual exclusion problem due to the potential for deadlock.

5. Suppose a 20MB file is stored on a disk on the same track (track 20) in consecutive sectors and the disk arm is currently situated over track 28. Assume that moving the arm from one cylinder to the next takes 2ms, it takes 10ms for the sector where the file begins to rotate under the disk head, and the drive has a reading rate of 100MB/s. If there are no other disk requests, how long will it take to retrieve this file from the disk?

Track Movement Time: (28-20) = 8 * 2ms = 16ms
Rotational Delay: 10ms
Reading Time: 20MB/100MB/s = 0.2s = 200ms
Total Time: 16ms + 10ms + 200 ms = 226ms
In total the time to read will be: **226ms.**