Diego Hurtado

P3] a) Clasico mono → este nucleo ya tiene sus interrupciones inhibidos dentro, por lo que no tenemos problemos

Moderno mono → Como es mono-core, no van a haber otros procesos paralelos, por lo que utilizando reccion criticos para inibir los interrupciones debiera funcionar bien

Moderno multi → Para asegurar la exclusion mutua en este se tiene que utilizar las herramientas de sincronizacion clasicas, pues no se tiene que detener todo el nucleo. Solo basta inhibir durante reccion criticos

Diego Hurtado

P3J b)

(i) Para obtener un buen tiempo de respuesta utilizaría RR (Round - Robin), ya que en este caso, el usuario va recibiendo reñales a medida que el proceso avanza. RR minimiza este tiempo en el que el usuario ve el primer resultado desplegado

(ii) Para minimizar el tiempo de despacho utilizaría SJF (el trabajo mas corto primero). Si bien no se sabe cuál va a ser el mas corto, esta estrategia tiene todo un sistema y algebra que ayudan en eso, entregando así los resultados de los tareas mas cortas en primer lugar, reduciendo el tiempo de despacho

Diego Arévalo

P3]

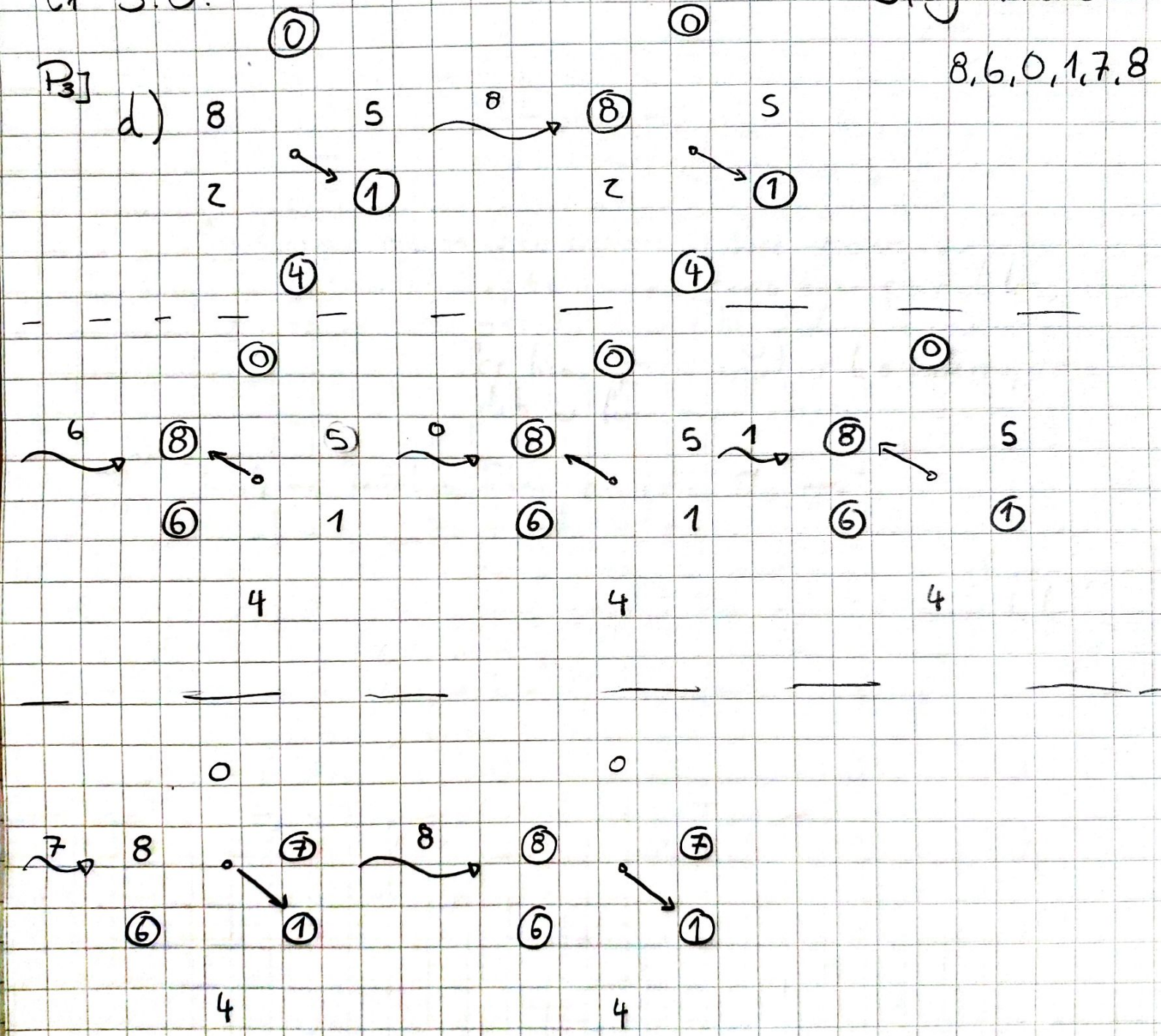
c) Al ser solo un proceso, no ocurre thrashing, lo que al utilizar working set solo le estamos sumando el costo de calcular el WS. Por esto es que tiene mucho mas sentido utilizar la estrategia del reloj, que no calcula WS y en este caso no va a tener problemas de thrashing (se da con multiples procesos)

C1 S.O.

Diego Hurtado

8, 6, 0, 1, 7, 8

P3]



P3

Diego Hurtado

e) Como no hay MMU, no hay proteccion de procesos.

Si el desarrollador edita, compila y ejecuta y todo al mismo tiempo, nada asegura que el editor, se este trabajando sobre la memoria que se quiere trabajar.

Como no van a haber paginas virtuales, podrian haber muchas incoherencias entre los procesos, informacion cruzada sin proteccion

Como consecuencia podria pasar que un proceso modifique los espacios de otro, entregando resultados impredecibles o que simplemente se caigan.