Apellido y Nombres	Legajo	Calificación

Teórico

1.Explique que función cumplen el bit ***Busy***, el campo ***Previous Task Link***, y el flag **NT**. Enumere que instrucciones y que tipos de conmutación de tareas modifican a cada uno.

2.Ejecución fuera de orden. Dibuje el esquema con las partes constitutivas, explique como se consigue ejecutar en diferente orden al flujo de instrucciones y como se reordenan los resultados. ¿Cuál es el concepto/mecanismo que permite su implementación? ¿Cuál es la diferencia entre un procesador con un pipeline de ejecución, y uno que implemente un motor de ejecución fuera de orden?

3.Explique al menos 2 motivos por los que se puede generar una excepción de fallo de página, las consecuencias de que eso ocurra para el programa en ejecución, y los pasos a realizar para solucionarla (si fuera posible).

4.Qué es un proceso en Linux? Qué es un thread? Cómo se crean, cómo los identifica el sistema y qué características tiene cada uno?

Practico

1)

a. Una aplicación en Linux debe recibir datos por un socket (**connfd**, ya creado y conectado) y datos por una FIFO (**fifofd**, ya creada y abierta). Los datos recibidos serán procesados por las funciones **Process_Socket(char *buf)** y **Process_FIFO(char *buf)**. Los datos pueden llegar de forma espontanea por cualquiera de las dos alternativas, y la aplicación debe responder de forma inmediata en el orden correspondiente. Escriba el código de la implementación.

b. Puede usar el mismo esquema que desarrolló anteriormente, si se reemplaza la FIFO por una Message Queue? Realice las modificaciones necesarias para implementar el cambio.

2)

Codifique un servidor concurrente que actue de front-end, aceptando conexiones TCP por el puerto 2010, y que debe generar un thread por cada cliente que se conecte. El thread recibirá 200 bytes del cliente, y los enviará por una message queue a un proceso batch, usando mensajes de tipo 2. El máximo número de conexiones simultaneas debe limitarse a 20 threads.