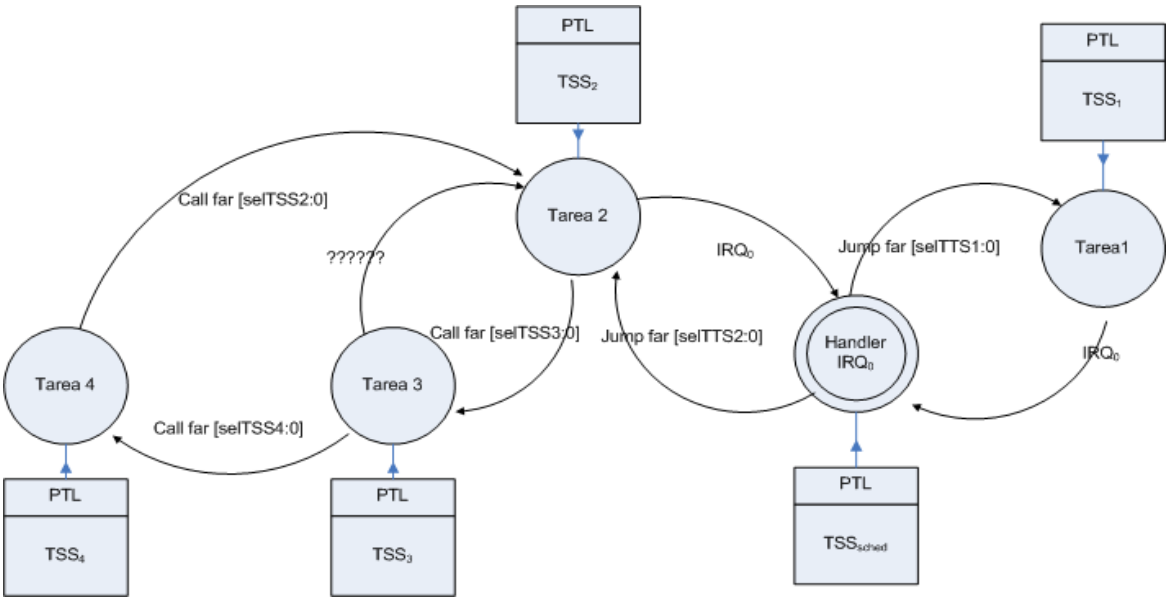
Universidad Tecnológica Nacional **Técnicas Digitales III**
Facultad Regional Buenos Aires 20 de Julio de 2010
Departamento de Ingeniería Electrónica Examen Final

Apellido y Nombres	Legajo	Calificación

Teórico (45 minutos máx.)

1. Explicar que función cumplen el bit ***Busy***, el campo ***Previous Task Link***, y el flag **NT**. Indique Además en que estructura o registro se encuentra cada uno. Analizar el estado o valor de **B**, **PTL**, y **NT** en el siguiente gráfico, para cada una de las tareas y en cada una de las transiciones. La lista de tareas del scheduler es: selTSS1, selTSS2, NULL.



Indique como se implementa la transición de Tarea3 a Tarea2. Indique que ocurre como consecuencia de la transición Tarea4 a Tarea2. ¿Es necesario efectuar alguna operación previa a la instrucción que implementa esta última transición?.

2. Un procesador genera la siguiente dirección lineal: 0x8Bc170FC. Por los pines de Address sale el número 0x324E70FC. Se pide:

   a) Calcular el contenido del descriptor de la página de 4 K de memoria física.

   b) Calcular el número de ese descriptor dentro de tabla de páginas.

   c) Calcular el número de descriptor de la página de memoria física que contiene la Tabla de Página en donde está el descriptor de la página de 4K direccionada.

   d) Sabiendo que la estructura de tablas de páginas se ha diseñado a partir de la dirección 0x8000 (en donde está el DTP), y que las Tablas de Páginas se ponen a continuación ordenadas de acuerdo con el número de su descriptor en la DTP calcular:

      i. El contenido del descriptor de la página que contiene la tabla de página en donde está el descriptor citado en (a)

      ii. El contenido de CR3.

Apellido y Nombres	Legajo	Calificación

e) Graficar el sistema de paginación con los resultados calculados indicando el proceso de traducción completo

3. TLB: Describir el funcionamiento. Indique en base a esto que valor se almacenaría de acuerdo con los datos del ejercicio anterior.

4. Ejecución fuera de orden. Dibuje el esquema con las partes constitutivas y explique como se consigue ejecutar en diferente orden al flujo de instrucciones y como se reordenan los resultados. Traducción a micro operaciones. Composición de la Unidad de Ejecución(diferentes tipos de puertos de ejecución). ¿Cuál es la diferencia entre un procesador con un pipeline de ejecución, y uno que implemente un motor de ejecución fuera de orden?

Ejercicio 2

Escribir un programa que escuche conexiones por el port 23667. Cada vez que se le requera una conexión creraá una instancia child para atenderla. El padre controlará que nunca haya activos mas de n childs, siendo n el 1er. argumento que se le haya pasado por línea de comandos en el momento de su ejecución.
El child será un proveedor de tráfico streaming de video.
Origen: carpeta /media/video/
Allí se encuentr auna colección de archivos en formato avi.
Cada child al ser creado interrogará al cliente destino por el port 23667 con un comando “NAME”. El cliente remoto debe devolver el nombre de un archivo .avi. El child lo busca y si no existe devuelve “FILE NOT FOUND”, y a continuación “NAME” Luego de tres intentos aborta la comunicación y termina su ejecución.
Si lo encuentra lo abre y lo transmite por el port udp 44688 a razón de un frame por vez. Finalizado cierra la conexión y termina.
No deben quedar instancias zombies.
Recursos adicionales: librería de video: Funciones a invocar y tipos de datos:

```
typedef struct CvCapture;
/*  Estructura interna para obtener los datos de captura de un archivo, o
dispositivo de video. */

cvCapture* cvCaptureFromFile (const char * filenane);
/* asigna e inicializa la estructura CvCapture para leer la secuencia de vídeo
desde el archivo especificado.*/

IplImage* cvQueryFrame(CvCapture* capture);
/*La función cvQueryFrame toma un cuadro de una cámara o un archivo de vídeo, lo
descomprime y lo devuelve. En el caso de un error, el valor devuelto es NULL.
Debe transmitirse la estrcutura completa por cada cuadro.*/

void cvReleaseCapture(CvCapture** capture)¶
/* Libera la estructura CvCapture.*/

typedef struct _IplImage
```

Apellido y Nombres	Legajo	Calificación

```
{
    int  nSize;   // sizeof(IplImage)
    int  ID;      // Version, always equals 0
    int  nChannels;     //Nºof channels. Support 1-4 channels.
    int  alphaChannel;  //Ignored
    int  depth;         //See comment below
    char colorModel[4]; //Ignored
    char channelSeq[4]; //Ignored
    int  dataOrder;     //0=IPL_DATA_ORDER_PIXEL: interleaved color channels, 1:
                            separate color channels
    int  origin;  //0:top-left origin, 1:bottom-left origin (Windows bitmap style)
    int  align;   //Ignore
    int  width;   //Image width in pixels
    int  height;  //Image height in pixels
    struct _IplROI *roi; // Region Of Interest (ROI). If not NULL, only this image
                            region will be processed
    struct _IplImage *maskROI; // Must be NULL
    void  *imageId;           // Must be NULL
    struct _IplTileInfo *tileInfo; // Must be NULL
    int  imageSize;     // Image data size in bytes
    char *imageData;    // A pointer to the aligned image data
    int  widthStep;     // The size of an aligned image row, in bytes
    int  BorderMode[4]; //Ignored
    int  BorderConst[4]; //Ignored
    char *imageDataOrigin; // Pointer to the origin of the image data (not
                            necessarily aligned). This is used for image
                            deallocation
}
IplImage;

/* Channel depth in bits + the optional sign bit ( IPL_DEPTH_SIGN ). The supported
depths are:
            IPL_DEPTH_8U
                Unsigned 8-bit integer
            IPL_DEPTH_8S
                Signed 8-bit integer
            IPL_DEPTH_16U
                Unsigned 16-bit integer
            IPL_DEPTH_16S
                Signed 16-bit integer
            IPL_DEPTH_32S
                Signed 32-bit integer
            IPL_DEPTH_32F
                Single-precision floating point
            IPL_DEPTH_64F
                Double-precision floating point
*/
```