Para todos los problemas del enunciado, si es necesario hacer suposiciones, deberán estar indicadas claramente, por escrito. Las mismas, serán consideradas parte integral de tu respuesta en el momento de la corrección.

1. Pasar a assembly MIPS, usando la convención de llamadas a funciones implementada en los trabajos prácticos:

```
/*
 * conversion functions between pg_wchar and multibyte streams.
 * Tatsuo Ishii
 * src/backend/utils/mb/wchar.c
 *
 */

/*
 * The pg_wchar type
 */
typedef unsigned int pg_wchar;

/*
 * conversion to pg_wchar is done by "table driven."
 * to add an encoding support, define mb2wchar_with_len(), mblen(), dsplen()
 * for the particular encoding. Note that if the encoding is only
 * supported in the client, you don't need to define
 * mb2wchar_with_len() function (SJIS is the case).
 *
 * These functions generally assume that their input is validly formed.
 * The "verifier" functions, further down in the file, have to be more
 * paranoid.  We expect that mblen() does not need to examine more than
 * the first byte of the character to discover the correct length.
 *
 * Note: for the display output of psql to work properly, the return values
 * of the dsplen functions must conform to the Unicode standard. In particular
 * the NUL character is zero width and control characters are generally
 * width -1. It is recommended that non-ASCII encodings refer their ASCII
 * subset to the ASCII routines to ensure consistency.
 */

/*
 * SQL/ASCII
 */
static int
pg_ascii2wchar_with_len(const unsigned char *from, pg_wchar *to, int len)
{
        int                     cnt = 0;

        while (len > 0 && *from)
        {
                *to++ = *from++;
                len--;
                cnt++;
        }
        *to = 0;
        return cnt;
}
```

2. En una máquina sin pipeline se corre un programa en el cual el 10 % de las instrucciones ejecutadas son divisiones. Para este procesador, todas las instrucciones se ejecutan en 1 ciclo, excepto las divisiones, que se ejecutan en 50 ciclos.

   a) ¿Cuál es el CPI de dicho programa en este procesador?

   b) ¿Qué porcentaje de tiempo se pasa ejecutando divisiones?

c) ¿Cuál sería el speedup si duplicáramos la velocidad de división?

3. Considerar el siguiente segmento de código C:

```
char A[8192];
for (j=0; j<100000; j++)
        for (i=0; i<Y; i=i+X)
                A[i] = A[i] + 1;
```

Asumir que sólo los accesos al array `A` van al caché de datos (las otras variables se mantienen en registros).

Para este código, cuál sería el *hit-rate* ...

a) ... con un caché de datos de 1KB y líneas de 16 bytes, direct-mapped, para los siguientes valores de X e Y:

	X=2	X=4	X=32
Y=128			
Y=2048			
Y=1536*			

*Nota: $1536 = 1024 + 512$

b) ... con un caché de datos de 1KB y líneas de 16 bytes, fully-associative, para los siguientes valores de X e Y:

	X=8	X=64
Y=2048		
Y=8192		

4. Sea un sistema de memoria virtual que permite alojar páginas de 4KB. El sistema usa 44 bits efectivos para describir direcciones virtuales, y 40 para direcciones físicas. En este ejercicio, vamos a constrastar aspectos del modelo de organización jerárquico de tres niveles con aquel en donde la tabla de paginación es lineal. En ambos casos, las PTEs son de 8 bytes. Ver figura 1.

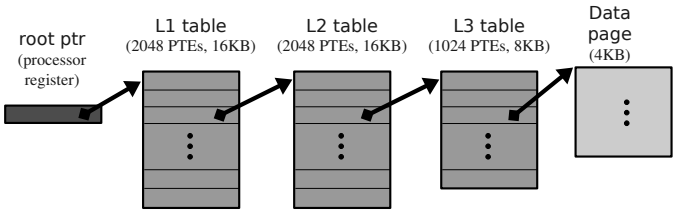

Figura 1: jerarquía de traducción de direcciones.

El procesador dispone de un TLB de datos con 64 entradas. Al ocurrir un TLB miss, las tablas de traducción son recorridas por hardware, a fin de recargar el TLB. La política de reemplazo es FIFO.

En este ejercicio, evaluaremos la ejecución y utilización de memoria del siguiente programa; el mismo es usado para sumar dos arreglos, y almacenar el resultado en un tercero:

```
uint8_t A[1048576]; /* 1MB array */
uint8_t B[1048576]; /* 1MB array */
uint8_t C[1048576]; /* 1MB array */
for(int i = 0; i < 1048576; ++i)
        C[i] = A[i] + B[i];
```

Suponemos, además, que los arreglos `A`, `B`, y `C` están ubicados en una región contigua de memoria física. Consideraremos, además, que los arreglos son mapeados usando 768 páginas de 4KB (es decir, cada arreglo usa 256 páginas).

En las preguntas que siguen, suponer que el programa es el único proceso en el sistema, e ignorar cualquier *overhead* asociado a la ejecución de las instrucciones, o con el sistema operativo. Suponer que los arreglos están alineados de tal forma, que minimizan el número de PTEs involucradas.

a) El particionado de la dirección virtual, en el caso del modelo jerárquico de 3 niveles es: 11 bits (L1 index), 11 bits (L2 index), 10 bits (L3 index), 12 bits (page offset). Ver figura 2.

Figura 2: composición de las direcciones virtuales en un modelo de traducción jerárquico de 3 niveles.

Mostrar, en forma similar, cómo sería el particionado de una dirección virtual en el modelo de traducción lineal. Incluir el nombre y longitud en bits de todos los campos involucrados.

b) Definimos el overhead de traducción (PTO), como el cociente entre (numerador) la cantidad de memoria física usada para almacenar las tablas de paginación, y (denominador) la cantidad de memoria física dedicada a páginas de datos.

Para el programa dado anteriormente, ¿cuál es el PTO_{hier}? ¿cuál es el PTO_{linear}?

c) Definimos el overhead de fragmentación (PFO), como el cociente entre (numerador) la cantidad de memoria física dedicada a páginas de datos, pero que nunca es accedida; y (denominador) cantidad de memoria física alocada a páginas de datos, accedida.

Para el programa visto, ¿cuánto valen PFO_{hier} y PFO_{linear}?

d) Consideremos ahora la ejecución del programa, suponiendo que la TLB está inicialmente vacía. Para cada uno de los casos (i.e. jerarquía de 3 niveles vs. lineal), ¿cuántos TLB misses ocurren, y cuántos accesos a memoria (PTEs) son necesarios para recargar la TLB?

e) ¿Cuál de los siguientes números es el mejor candidato para estimar (orden de magnitud) el speedup? $SU = \{1{,}01; 10; 1000; 1000000\}$. Elegir uno, explicando brevemente tu respuesta. Tomar el caso del modelo jerárquico como referencia.