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 el siguiente fragmento de código, respetando la ABI presentada en clase. Deberá incluir en la resolución un diagrama del stack frame resultante. Asumir que `strlen()` ya fue implementada y puede invocarse directamente por `strrev()`.

```
char *strrev(char *p)
{
    char *s = p;
    char *e = p + strlen(p) - 1;
    while (s < e)
    {
        char a = *s;
        *s = *e;
        *e = a;
        s++, e--;
    }
    return p;
}
```

2. Resolver los siguientes problemas de Virtual Memory

   a) Un CPU genera direcciones virtuales de 32 bits, tiene páginas de 4KB, y una TLB que cuenta con 128 entradas y es 4WSA. ¿Cuál es el tamaño mínimo de los tags en la TLB?

   b) Para una computadora con MMU se tiene la tabla de páginas jerárquica de 2 niveles completa en memoria, y todas las páginas están residentes en memoria (todas las PTEs tienen R=1). El tiempo de acceso a memoria es de 100ns. Toma 1ns el acceso y resolución de la TLB, y en caso de TLB miss se suman los tiempos de acceso a memoria. Si el hit ratio de la TLB es 0.6, ¿cuál es el tiempo efectivo de acceso a memoria visto desde el CPU? (asumir que no cuenta con otros caches).

   c) Un procesador cuenta con una MMU que utiliza tablas de páginas de 2 niveles, almacenadas en memoria principal. Tanto las direcciones virtuales como físicas son de 32 bits. Los 10 bits más significativos de la virtual address corresponden al índice del primer nivel de tablas de páginas, y los siguientes 10 bits corresponden al índice del segundo nivel de tablas de páginas. Las PTEs en ambos niveles son de 4 bytes. Un proceso dispone sólo de las siguientes páginas en su espacio de direcciones virtuales: 2 páginas de código contiguas partiendo de la dirección `0x00040000`, 512 páginas de datos contiguas partiendo de `0x00700000`, y una página de stack partiendo de `0x7ffff000`. ¿Cuál es la cantidad de memoria necesaria para almacenar estas tablas de páginas? Elabore su respuesta.

3. Se tiene una computadora con un cache L1 split, donde ambos L1I y L1D son direct mapped de 4KB y 16 bytes por línea.

```
.text
.align 4
.globl strcpy
.ent strcpy
strcpy:
        subu  s0, s0, 8
        .cprestore 4
        sw    fp, 0(sp)
        move  fp, sp
        sw    a0, 8(fp)
        sw    a1, 12(fp)
```

```
        move  v0, a0
    loop:
        lb    t1, 0(a0)
        sb    t1, 0(a1)
        beq   t1, $zero, end
        addiu a1, a1, 1
        addiu a2, a2, 1
        j     loop
    end:
        lw    fp, 0(sp)
        lw    gp, 4(sp)
        jr    ra
.end strcpy
```

Cuando se llama a `strcpy()`, `$a0` apunta a un string C de 1024 bytes de longitud (sin contar el 0 final con el que suma 1025 bytes en total), y `$a1` apunta a un buffer de tamaño suficiente para albergar al string apuntado por `$a0`. Al entrar en `strcpy()`, se tiene que `$a0` = `0x00804000`, `$a1` = `0x00823000` y `$sp` = `0x7fffffff8`. Desde que el procesador comienza a ejecutar `strcpy()` hasta que finaliza, y asumiendo que ambos caches están vacíos:

a) Estimar el miss rate de L1I (asumiendo que `strcpy()` es una dirección alineada a múltiplo de 16).

b) Estimar el miss rate de L1D, si éste es write back - write allocate.

c) Estimar el miss rate de L1D, si éste es write through - write no allocate.

d) Encontrar el miss rate de L1D, si éste es del mismo tamaño pero 2WSA LRU y write back - write allocate.

4. 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?