TP N° 6 – Comprensión del ISA de MIPS. Programación en lenguaje ensamblador MIPS con MIPSX

Ejercicio 0. Explique con sus palabras por qué un diseño segmentado es clasificado como “paralelismo a nivel de instrucciones”. Tambien, detalle el motivo (o los motivos) por el cual, este diseño, mejora el rendimiento de la máquina. En su explicación utilice la ecuación de tiempo de ejecución, indique variables que se mejoran, por qué y/o cómo es que mejoran (la explicación de por qué y/o cómo es lo más importante), metodología del reloj, etc.

Ejercicio 1. Cree un programa que dado un número flotante de precisión simple extraiga los valores de signo, exponente y mantisa, guardando 1 si es positivo y -1 si es negativo en la variable `signo`, guardando el exponente traducido de notación en exceso a complemento a dos (es decir, habiéndose restado 127) en la variable `exponente`, y la mantisa (sin modificar, poner en cero todos los bits que no pertenezcan a la mantisa) en la variable `mantisa`.

```
.data
    memoria:
    flotante:
        .float 3.14
    signo:
        .byte 0 # -1 si es negativo, 1 si es positivo
    exponente:
        .byte 0 # En C2
    mantisa:
        .word 0 # Sin modificar del punto flotante,
                # es decir sin agregar el cero.
```

a. Cargue el programa anterior en **MIPSX** y, observando el código desensamblado, identifique cuales de las instrucciones de su programa son pseudo instrucciones y a que instrucciones máquina fueron traducidas.
b. Analizando el volcado de memoria, identifique la codificación expresada en hexadecimal de dos instrucciones aritméticas, una de salto, y dos de carga y almacenamiento ¿Cuál es la dirección del último byte del segmento de texto?
c. ¿Cuál es la dirección de memoria de los datos `flotante`, `signo` y `mantisa`? ¿Cuántos bytes se desperdician por alineamiento?

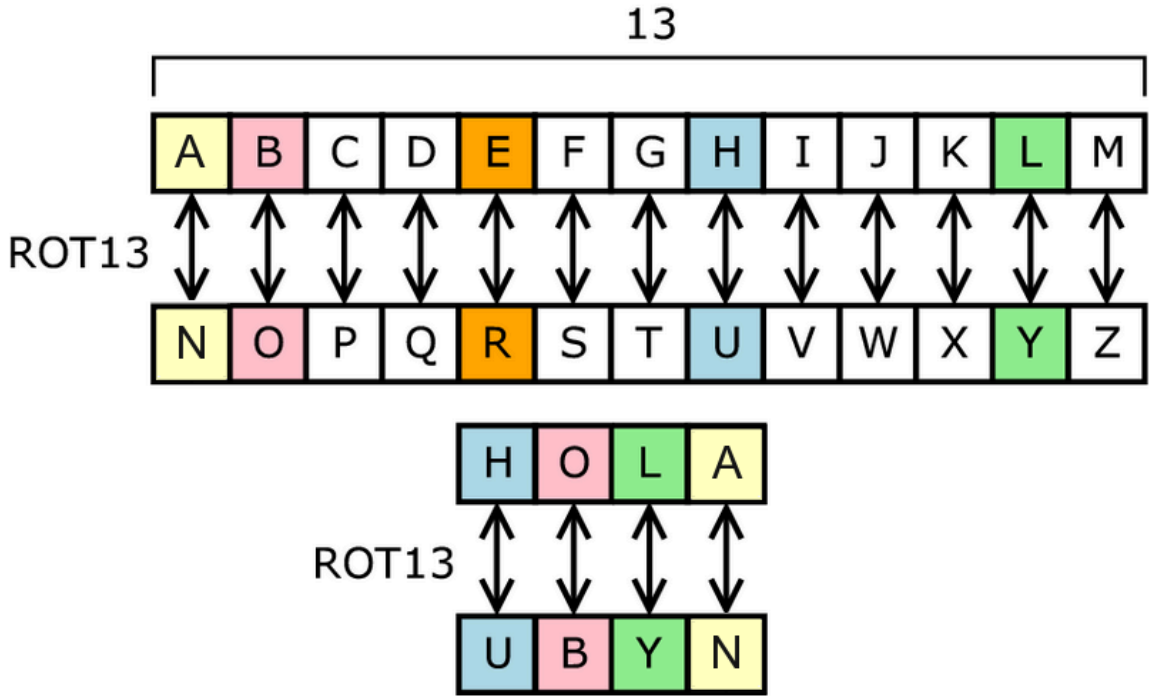
Ejercicio 2. Dado el siguiente segmento de datos donde `Imagen` representa un a imagen de 9 pixeles con el formato **RGB** donde cada byte es la intensidad de color que compone cada pixel (reprecentada como un entero sin signo donde cero indica minima intencidad y 255 maxima intencidad). Realizar un programa que haga una conversión de colores a escala de grises, guardando el resultado en `destino` (la imagen destino también debe ser almacenada en formato **RGB**, con las tres componentes almacenando el mismo valor).

Para pasar a escala de grises deben realizar el promedio entre las tres componentes que forman el píxel,

donde cada componente es un entero sin signo que va de `0..255`.

```
.data
    memoria:
    Imagen:
        .byte 123,0,33,44,55,2,56,78,66
        .byte 0,33,44,55,2,56,78,66,123
        .byte 33,44,55,2,56,78,66,23,55
    destino: .space 27
```

Ejercicio 3. Cree un programa que aplique el cifrado `ROT13` a una cadena de texto `ASCII`. Este cifrado sustituye las letras de la A a la M por aquellas de la N a la Z, y viceversa.



Ejercicio 4. Desarrolle un programa calculadora. Este programa lee una cadena de texto de 3 letras, donde se encuentran dos operandos de un dígito y una operación, como se presenta en el segmento de datos de ejemplo, en la etiqueta **`calcular`**.

```
.data

memoria:
calcular:  .ascii "3+6"
resultado_ascii:  .byte 0
resultado: .byte 0
```

La calculadora soporta la operación de suma o resta.
El programa debe:

1. Convertir los valores ascii a su valor numérico.

2. Decodificar la operación a realizar, que está también en representación de código de caracteres ascii, entre los operandos.
3. Realizar la operación.
4. Almacenar el resultado numérico en la posición de memoria **`resultado`**.
   El resultado también deberá almacenarse en la posición de memoria **`resultado_ascii`**, pero en representación de código de caracteres ascii.

El resultado posible es de un sólo dígito, en el rango de 0 a 9. Si el resultado excede este rango de representación coloque en **`resultado_ascii`** la letra "**`N`**".