INFORME TÉCNICO PARCIAL #2 ORGANIZACIÓN DE COMPUTADORES

1. **Integrantes:**
   - Simón Mazo Gómez
   - Andrés Vélez Álvarez
   - Sebastián Salazar Henao

2. **Descripción del algoritmo elegido y ¿por qué?**

   El algoritmo elegido implementa el cálculo de la parte entera de la raíz cuadrada de un número en el lenguaje ensamblador de la máquina Hack. Lo que hace es ir probando valores de un contador (i) y, para cada uno, calcular su cuadrado mediante sumas repetidas, ya que en esta arquitectura no existe una instrucción directa de multiplicación. Después, el valor obtenido (sqr) se compara con el número original (num). Si el cuadrado es mayor que el número, el proceso se detiene y la respuesta final es el valor de i-1, el cual representa la raíz cuadrada entera.

   La elección de este algoritmo se hizo basada en la búsqueda de un código y procedimiento que cumpliera con algunos parámetros tales como el manejo inteligente de la memoria, un procedimiento que estuviera cimentado solo en operaciones básicas y que tuviera sentido y utilidad básica.

3. **Diseño gráfico de las iniciales: ¿cómo las dibujaron? ¿qué efectos usaron?**
   El diseño gráfico de las iniciales se elaboró en el emulador virtual de **nand2tetris**, utilizando el **Bitmap Editor**. Primero se diseñó cada letra por separado, pero trabajando ambas tipografías simultáneamente: una en mayúscula y otra en minúscula, ubicadas una debajo de la otra. Para garantizar una buena visibilidad en pantalla, se definió una **altura de 48 píxeles**, mientras que el **ancho** se ajustó según la forma de cada letra, sin superar también los 48 píxeles.

   Las tipografías se eligieron a partir de modelos existentes en el sitio **Dafont**, seleccionando estilos de tipo *bitmap* que facilitaran su reproducción manual, píxel por píxel.

   - **Mayúsculas:** Ari-w9500
   - **Minúsculas:** Kiwisoda

4. **Dificultades técnicas y cómo las superaron.**
    - En el algoritmo de la raíz cuadrada, la comparación entre el cuadrado i*i se estaba haciendo algo complicada de implementar por el tipo de lógica a la que estamos acostumbrados. La solución se hizo a través de la resta, pues un numero a es mayor que b, si a-b>0.

    - Para implementar el dibujo de las iniciales, enfrentamos la incertidumbre de si era posible pintar las 3 letras simultáneamente o si debíamos hacerlo secuencialmente. Después de probar diferentes enfoques, determinamos que la manera óptima era crear funciones modulares para cada letra (**draw_A**, **draw_S**, **draw_B**) e invocarlas secuencialmente - primero **draw_A**, luego **draw_S** y finalmente **draw_B.** Esta arquitectura modular nos permitió reutilizar las mismas funciones cuando el usuario presiona una tecla específica para mostrar una letra individual, evitando duplicación de código.

    - Otro problema era cómo pintar las letras separadas sin que se superpusieran. Lo solucionamos con un desplazamiento de 5 palabras de memoria (80 píxeles), así la letra A empieza en la palabra 0, la letra S en la palabra 5 y la letra B en la palabra 10. Esto se realiza sumando el offset a la dirección base de SCREEN."

5. **Distribución del trabajo y aprendizajes clave.**
    - Simón Mazo: Algoritmo y documentación de la raíz.
    - Sebastián Salazar: Diseño de iniciales y documentación.
    - Andres Velez: Implementación de iniciales en pantalla y documentación.

    En cuanto a aprendizajes clave tenemos:

    - Representación binaria de la información y direccionamiento directo de memoria.
    - Implementación de operaciones aritméticas básicas a partir de sumas y restas.
    - Diseño de rutinas gráficas de bajo nivel en un entorno sin soporte de gráficos vectoriales.
    - Coordinación y trabajo colaborativo en entornos técnicos con múltiples responsabilidades.
    - Importancia de la modularidad y la legibilidad, incluso en lenguajes de bajo nivel.