**Bitácora de Trabajo - 8/11/2024**

**Actividades Realizadas:**

1. **Inicio del Proyecto:**
   * Se estableció la estructura inicial del simulador, incluyendo los módulos principales:
     + main.py: Coordinador principal.
     + pipeline.py: Lógica del pipeline.
     + visualization.py: Representación gráfica con Pygame.
   * Se diseñó el flujo general del pipeline (IF, ID, EX, MEM, WB).
2. **Desarrollo Básico:**
   * Se implementaron las primeras funciones en pipeline.py:
     + Inicialización del pipeline (initialize\_pipeline).
     + Simulación de un ciclo (execute\_cycle) para las instrucciones básicas.
   * En visualization.py, se creó la base de la visualización, mostrando el pipeline, los registros y la memoria en una ventana.
3. **Pruebas Iniciales:**
   * Se cargó un programa con una instrucción ADD básica.
   * Se verificó la actualización de los registros y la memoria tras la ejecución.

**Resultados:**

* Se estableció un pipeline funcional básico con visualización gráfica.
* Se logró simular el flujo de una instrucción ADD en todas las etapas del pipeline.

**Siguientes Pasos:**

* Ajustar la lógica para manejar mejor la sincronización entre etapas.
* Introducir colores dinámicos y más instrucciones en el pipeline.

**9/11/2024**

**Actividades Realizadas:**

1. **Corrección de Errores:**
   * Se solucionó el problema donde IF no mostraba correctamente las instrucciones, introduciendo un retardo en la transferencia a ID.
   * Se corrigieron errores de sintaxis en los mensajes de depuración.
2. **Mejoras en Visualización:**
   * Se añadieron colores dinámicos para etapas activas y vacías.
   * Se reorganizó la visualización de los registros y la memoria en cuadrículas claras.
   * Se agregó un encabezado mostrando ciclo, tiempo y valor del PC.
3. **Reestructuración del Flujo:**
   * Se mejoró main.py para soportar simulaciones en consola o visualizaciones en tiempo real.
4. **Sincronización de Pipeline:**
   * Se corrigió el problema donde dos etapas del pipeline se mostraban como activas simultáneamente. Ahora solo una etapa está activa a la vez.
   * Se agregó lógica para limpiar cada etapa del pipeline después de mover una instrucción, asegurando una transición clara y precisa entre las etapas.

**Resultados:**

* Pipeline sincronizado y funcional, mostrando correctamente el estado en cada etapa.
* Interfaz gráfica clara y en sincronía con la simulación textual.

**Siguientes Pasos:**

* Implementar más instrucciones (SUB, LOAD, STORE) y manejar dependencias de datos.
* Documentar y optimizar el código.

**Bitácora de Trabajo - 10/11/2024**

**Actividades Realizadas:**

1. **Implementación de Nuevas Instrucciones:**
   * Se añadieron las instrucciones **SUB** (resta) y **MUL** (multiplicación) al pipeline.
   * Ajustes realizados en las siguientes áreas:
     + **pipeline.py**: Extensión de la lógica de execute\_cycle para manejar las nuevas operaciones.
     + **registers.py**: Inicialización de valores específicos en los registros para probar SUB y MUL.
     + **main.py**: Inclusión de botones interactivos para las nuevas instrucciones en la interfaz gráfica.
2. **Manejo de Segmentación:**
   * Se detectó un problema inherente a la segmentación del pipeline que causaba conflictos al sobrescribir registros intermedios.
   * Ajustes realizados para garantizar que las instrucciones procesen los valores correctos en cada etapa.
3. **Pruebas y Validaciones:**
   * Verificación de la ejecución correcta de SUB y MUL en todas las etapas del pipeline.
   * Validación de resultados en los registros tras la ejecución:
     + SUB: R4 = R5 - R6.
     + MUL: R7 = R0 \* R1.
   * Confirmación de que los resultados eran correctos en un entorno segmentado.
4. **Actualización de Visualización:**
   * Sincronización mejorada entre la ejecución y la representación gráfica.
   * Inclusión de colores dinámicos para las nuevas instrucciones SUB y MUL.

**Resultados:**

* El simulador ahora soporta tres instrucciones aritméticas: **ADD**, **SUB**, y **MUL**.
* Visualización gráfica sincronizada y representativa del estado del pipeline en tiempo real.
* Resolución de problemas de segmentación, garantizando resultados precisos en un entorno segmentado.

**Siguientes Pasos:**

* Implementar instrucciones de acceso a memoria (LOAD, STORE).
* Introducir manejo de dependencias de datos y riesgos de pipeline.
* Continuar documentando y optimizando el código base para futuras extensiones.