Bitácora de Proyecto Grupal 2

Gabriel Guzman Rojas
Sebastián Manyusly Chen Cerdas
Sebastián Hernandez Bonilla
Jeremy Serracin Oporta

Fundamentos de Arquitectura de Computadores
Instituto Tecnológico de Costa Rica

1. 1/11/2024

Se realizó una reunión virtual para discutir acerca de este proyecto, ya que como teniamos el examen del curso el cual era necesario para realizar el procesador porque en este se encontraba el pipeline, por lo que nos iba a permitir entenderlo mejor estudiandolo. En esta reunion repartimos todo, y decidimos que Sebastian Chen y Jeremy se iban a encargar de realizar el procesador basico, mientras que Gabriel y Sebastian Hernandez se iban a encargar del hazard y el manejo de los riesgos.

2. 7/11/2024

Jeremy comenzo a realizar el pipeline, ya que de los 3 era el que ya no tenia proyectos pendientes, y cuando Sebastian Chen terminara el proyecto en el que estaba trabajando, iba a empezar con Jeremy a realizar el pipeline.

3. 10/11/2024

Jeremy termina el procesador con pipeline, este realiza todas las instrucciones de forma correcta en cada una de las 5 etapas del procesador, queda preparado para agregar instrucciones y crear la unidad de hazards

4. 13/11/2024

Chen añade 9 instrucciones más al procesador que se ejecutan de forma correcta, desde operaciones de lógica y aritmética, a instrucciones de carga y lectura desde memoria, todas permiten lectura de registros e inmediatos.

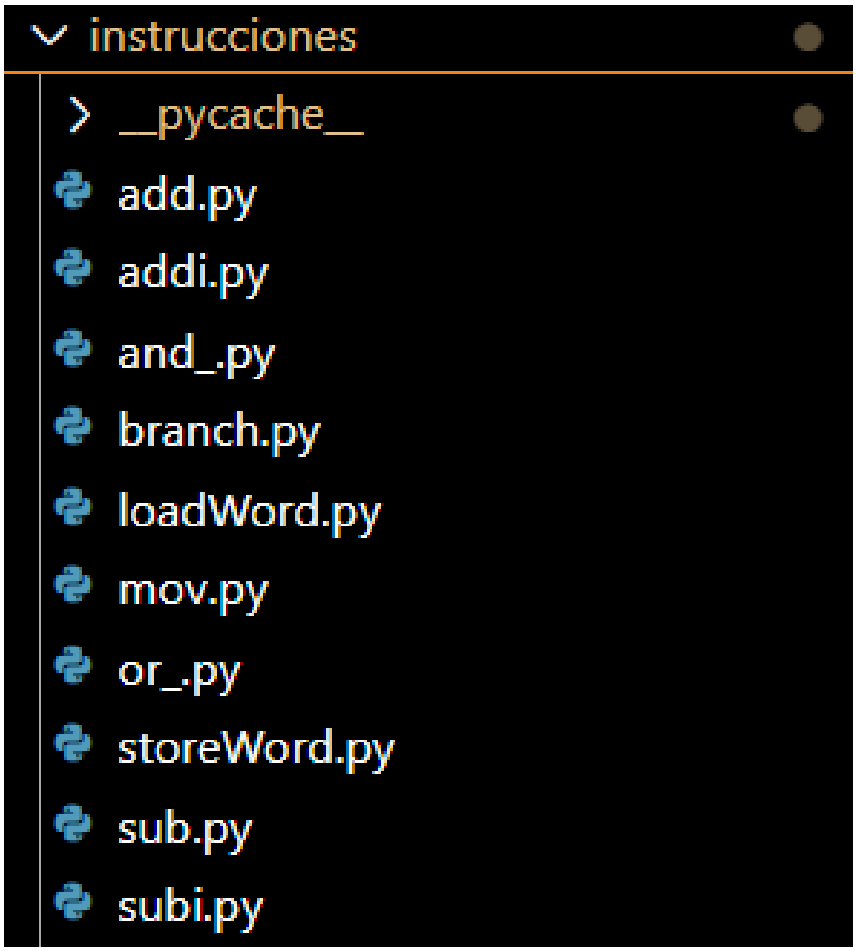

Figura 1. instrucciones de los procesadores.

5. 16/11/2024

Este día se implementó la instrucción branch, que era la única instrucción que todavía no se había programado, pero que era necesaria porque se ocupaba para realizar la predicción de saltos más adelante. Esta instrucción está diseñada para realizar branch equal, ya que es el único salto que implementamos en este caso. De la misma manera, Sebastián Hernández comenzó a trabajar en el procesador con predicción de saltos.

6. 17/11/2024

Este día, Sebastián Hernández terminó el procesador con predicción de saltos. La implementación fue sencilla porque realmente la ciencia no es tanta, por lo que no se tuvo que cambiar nada del procesador. Esto se debe a que el salto realiza una predicción, y solo hay dos formas de saltar: una de ellas tiene la penalidad de miss prediction, mientras que la otra opción simplemente realiza un flushed.

Ese mismo día, Gabriel Guzmán comenzó a trabajar en el procesador con forwarding. A diferencia de la predicción de salto, el forwarding, debido a la estructura de nuestro procesador, era un poco más complicado de implementar.

7. 18/11/2024

Chen crea una version basica de la interfaz gráfica, siendo una clase e instanciando en cada procesador para mostrar sus métricas, en la interfaz se puede ver cada componente de hardware utilizado por el pipeline

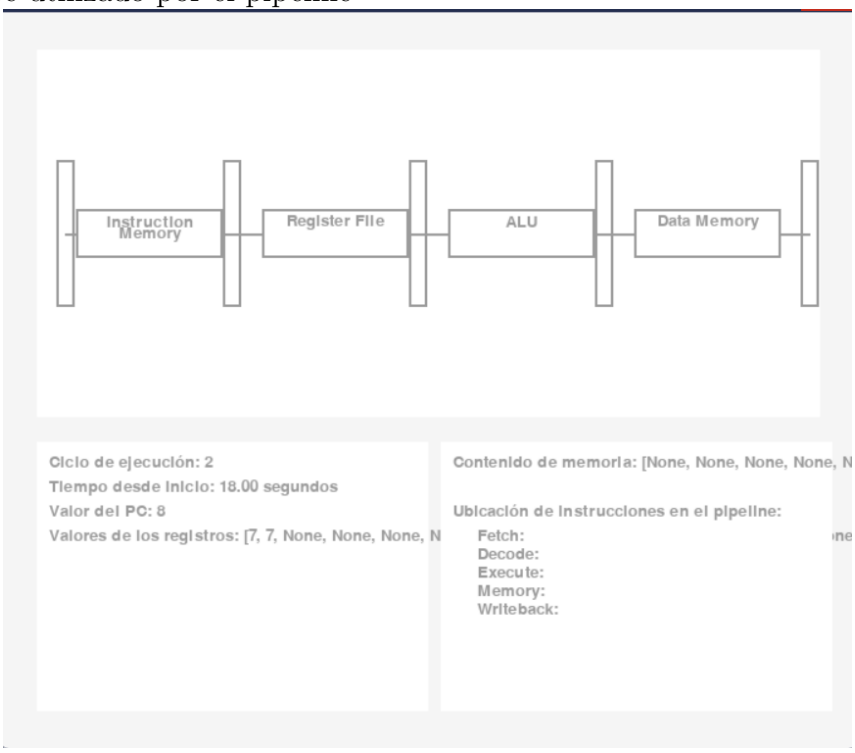

Figura 2. interfaz basica.

8. 19/11/2024

Este día se terminó el forwarding de Gabriel Guzmán, el cual consiste en un forwarding simple, pero funcional. Este procesador tomó más tiempo debido a la estructura del procesador en sí y a cómo se utilizaban los datos en cada etapa. Siguiendo los principios básicos del forwarding, los datos se procesan y, si la instrucción proviene de la siguiente etapa, los datos se recuperan del resultado de la ALU, que en este caso se obtiene al final de la etapa execute. De la misma manera, en el caso de ser una instrucción que proviene de la tercera etapa, se envían desde memoria, y después de esa etapa, el dato ya se escribió, por lo que no es necesario aplicar más forwarding en este caso.

9. 22/11/2024

Jeremy arregla algunos errores específicos que ocurren en el branch prediction, ya que se ejecutaba en la etapa incorrecta, despues se hacen merges de las ramas de desarrollo para volver a la rama principal del pipeline.

10. 23/11/2024

Jeremy une los codigos del Branch Prediction con el de Forwarding y crea el procesador con Full Hazards que integra toda la unidad de riesgos al procesador, con este quedan listo los 4 procesadores que se piden en la especificación.

```python
class ProcesadorFullHazard:
    def __init__(self, interval=1):
        self.PC = 0
        self.IM = memoriaInstrucciones()
        self.regIM = Registro()
        self.RF = archivoRegistros()
        self.regRF = Registro()
        self.ALU = ALU()
        self.regALU = Registro()
        self.DM = memoriaDatos()
        self.regDM = Registro()
        self.hazard_control = HazardControl(self)
        self.branch_predictor = BranchPredictor(default_prediction=True)
```

Figura 3. Procesador con full hazards integrado.

11. 23/11/2024

Chen y Sebastian trabajan en la interfaz gráfica, se integra en cada uno de los procesadores y se conectan las actualizaciones entre el procesador y la interfaz, ademas se agrega una interfaz para elegir el procesador a ejecutar, corriendo 2 simultaneamente y sus interfaces.

Figura 4. interfaz de seleccion.

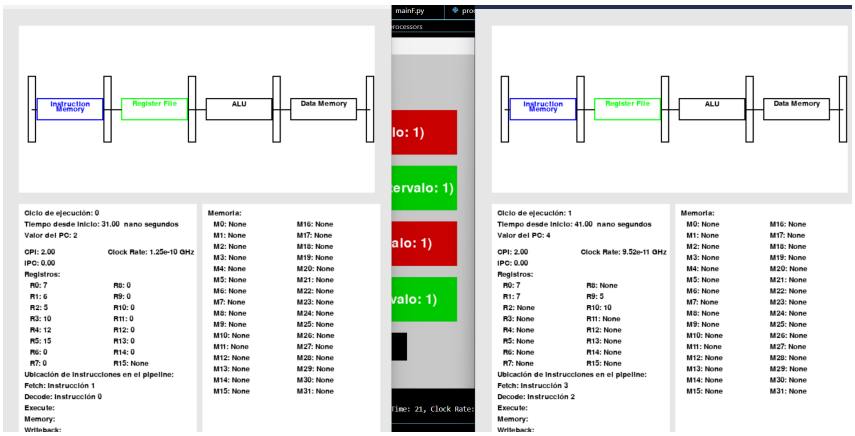

Figura 5. interfaz de cada procesador.

12. 24/11/2024

Sebastian ajusta las ultimas métricas a mostrar de la interfaz, como el ipc, cpi y el clock rate según cada procesador.