Arquitectura de Computadores

Tercer curso

Itinerario de Ingeniería de Computadores

@Iván Castilla Rodríguez 2021-2022

SIMDE: Simulador de Planificación Dinámica y Estática

Un Simulador Para el Apoyo Docente en la Enseñanza de las Arquitecturas ILP con Planificación Dinámica y Estática

@Iván Castilla Rodríguez 2021-2022

- Simuladores como **herramienta docente**
  - Visión más práctica de los conceptos
  - Posibilidad de experimentación
  - Aprendizaje interactivo
- Simuladores existentes no satisfactorios
  - Muy específicos o emuladores (TSM320C6xxx)
  - Muy complejos y poco ilustrativos (SatSIM)
  - No funcionan (Midas)
  - No hay simuladores “duales” Superescalar-VLIW

¿Qué es SIMDE?

- Simulador arquitecturas **ILP**: Superescalar/VLIW
- Estructura común
- Control Superescalar basado en **Tomasulo**
- Estructura VLIW muy **simple**

¿Qué no es SIMDE?

- **NO** realista
- **Objetivo**: Ilustrar conceptos (NO realizar experimentos)
- **NO** es un **Compilador VLIW**

Estructura de la máquina: características comunes

- Diseñadas para trabajar con **palabras de 32 bits**, tanto con enteros como con flotantes.
- Juego de instrucciones basado en **MIPS IV**
- **Monoprograma**.
- Las instrucciones ocupan **1 palabra** de memoria
  - Se avanza sumando 1 al PC

- **Memoria**
  - Tamaño fijo: 1024 palabras de 32 bits
  - Programa completamente contenido en caché de instrucciones (no hay fallos de caché de instrucciones)
  - Fallos de Caché de datos aleatorios (controlados por porcentaje)

- Registros de propósito general (**GPR**)
  - Banco de 64 registros de 32 bits.
  - Se denotan R0, R1, R2, ..., R63.
  - El valor de R0 se mantiene siempre a 0.

- Registros de punto flotante (**FPR**)
  - Banco de 64 registros de 32 bits (precisión simple).
  - Se denotan F0, F1, F2, ..., F63.

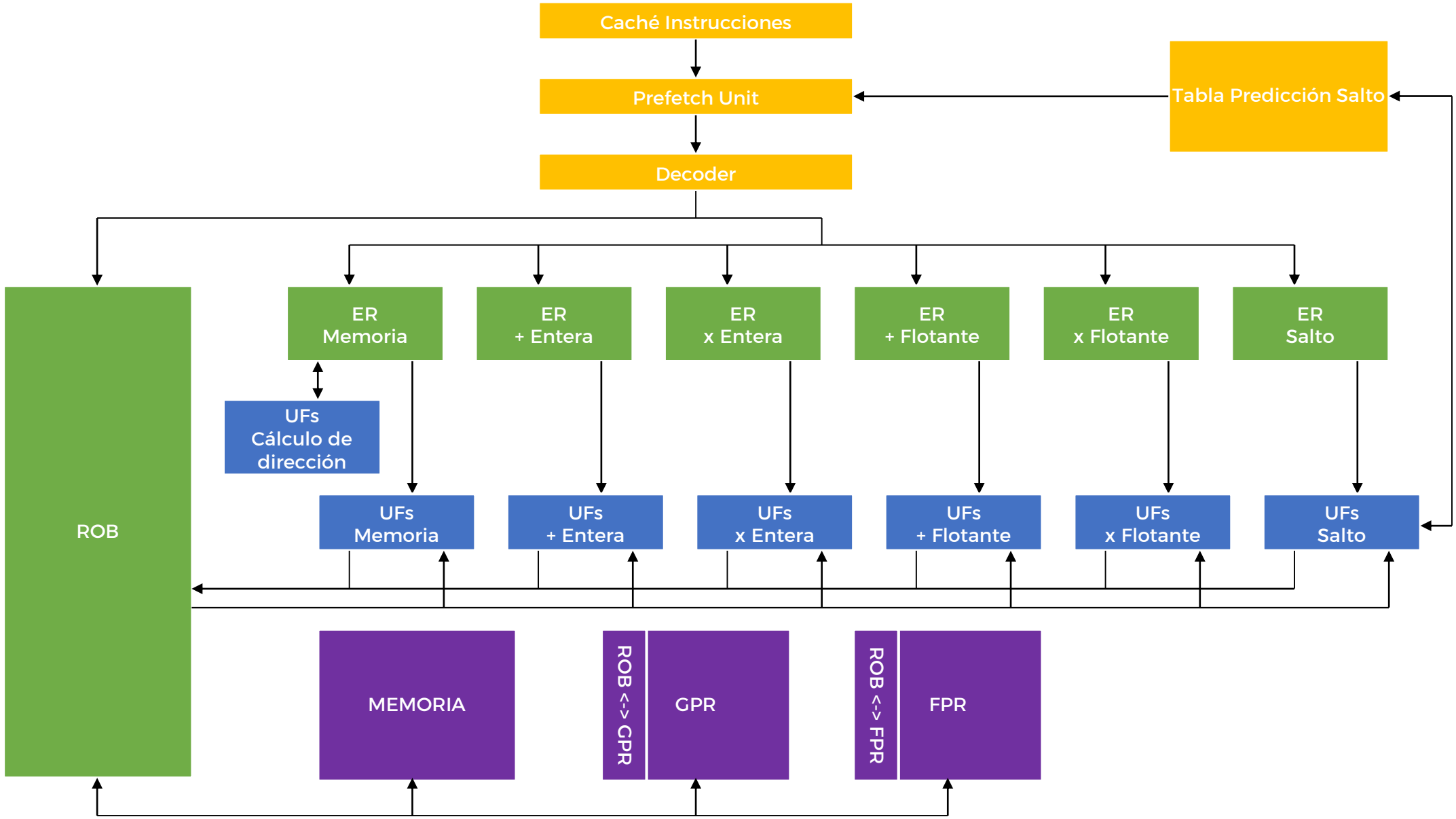
Estructura de la máquina: elementos comunes

- Unidades Funcionales (**UF**):
  - Segmentadas
  - Etapas de 1 ciclo
  - Latencia = Nº de etapas
  - Tipos:
    - **ALU entera**: ADDI, ADD, SUB, OR, AND, XOR, NOR, SLLV, SRLV
    - **Multiplicación entera**: MULT
    - **Suma flotante**: ADDF, SUBF
    - **Multiplicación flotante**: MULTF
    - **Memoria**: LW, LF, SW, SF
    - **Salto**: BNE, BEQ, BGT

Estructura de la máquina: superescalar

- Control de la máquina basado en el algoritmo de **Tomasulo**
- **Grado de emisión** parametrizable.
- **Predicción dinámica de salto** controlada con una tabla de 2 bits
- **Ejecución especulativa**

Estructura de la máquina: superescalar



Funcionalidades: superescalar

- Cargar un único **programa secuencial** cada vez, base para la ejecución Superescalar.
- Modificar los **parámetros** de las máquinas.
- Añadidas herramientas para facilitar la creación, carga desde fichero y modificación de este código:
  - Coloreado de bloques básicos
  - Chequeo automático del código

Funcionalidades: superescalar

- Modificar el contenido de la **memoria** y **registros**.
- Realizar **simulaciones** continuas o paso a paso de la máquina Superescalar, permitiendo el uso de **breakpoints** para detener la ejecución en un punto del código determinado.

SIMDEweb: Funcionalidades

- Versión **nueva** basada en **tecnología web**

- **Mejora** a SIMDE pero está **incompleta**
  - Permite avanzar y retroceder en la traza de ejecución
  - El ensamblador da mejores mensajes de error sobre el código
  - Permite ejecución por lotes

- Dos formas de acceder:
  - Última versión: http://simde.iaas.ull.es/
  - Solo superescalar: https://etsiiull.github.io/SIMDE/

Probando el programa

Supón que tienes un vector de 16 elementos (punto flotante) a partir de la posición 50 de memoria, y quieres sumar a cada elemento una cantidad fija (en la posición de memoria 40).

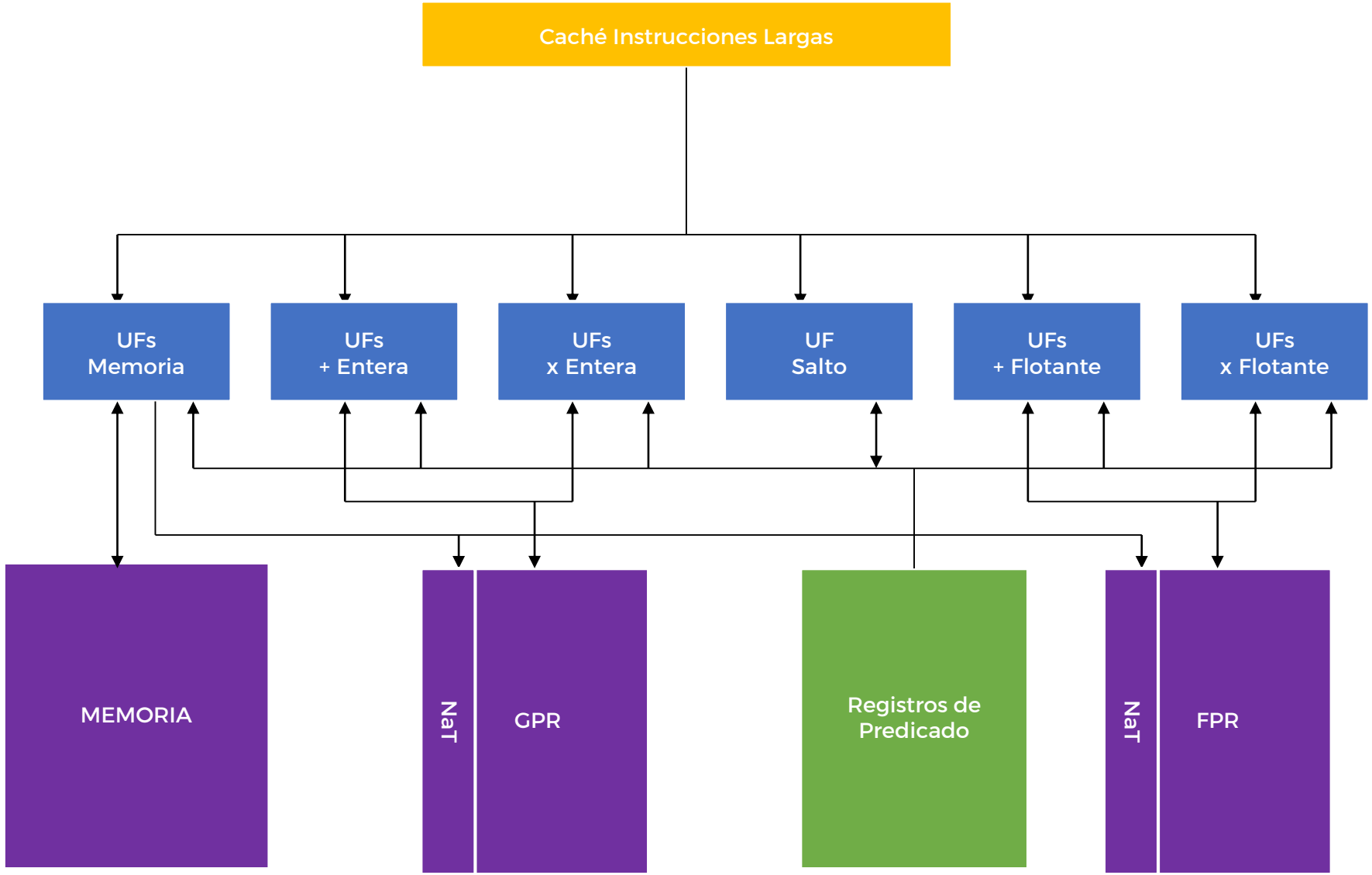
El resultado se coloca a partir de la posición 70 de memoria.

Codifica el programa que realiza esa operación usando el repertorio de SIMDE. Prueba que el programa compile con el mismo SIMDE.

Estructura de la máquina: VLIW

- Hardware extremadamente **simple**
- **1 instrucción larga por ciclo**
- Instrucción larga totalmente **horizontal** (tantas operaciones y de igual tipo que las UF existentes).
- 1 sólo **salto** por instrucción
- Emplea **predicación** (64 registros).
- Los fallos de caché detienen la ejecución (**Bits NaT**)

Estructura de la máquina: VLIW



- Registros de propósito general (**GPR**) y de punto flotante (**FPR**)
  - Aparte de las características comunes, la lectura se realiza en la primera mitad del ciclo y la escritura en la segunda mitad. De esta manera se evitan los riesgos WAR.

Funcionalidades: VLIW

- Cargar un único **programa secuencial** cada vez, que sirve de base para diseñar un código VLIW.
- Modificar los **parámetros** de las máquinas.
- Ofrece un entorno para hacer el trabajo del compilador de una máquina VLIW al crear un **código de instrucciones largas**.
- Añadidas herramientas para facilitar la creación, carga desde fichero y modificación de este código:
  - Coloreado de bloques básicos
  - Chequeo automático del código

Funcionalidades: VLIW

- Modificar el contenido de la **memoria** y **registros**.
- Realizar **simulaciones** continuas o paso a paso de la máquina VLIW, permitiendo el uso de **breakpoints** para detener la ejecución en un punto del código determinado.

- 2 trabajos diferenciados:
  - **Optimización del código secuencial** (desenrollado de bucles, software-pipelining...). Se realiza en el fichero secuencial.
  - **Planificación de las operaciones secuenciales** en las instrucciones largas VLIW. Se realiza con el simulador.

- Tener en cuenta dependencias **RAW**
- Aprovechar **configuración de los registros**
- Fijar correctamente los **destinos de los saltos**
- Las **instrucciones largas** se ejecutan completas
- **Repetir** operaciones si es necesario
- Usar registros de **predicado**
- **Chequeo** automático de código