

ARQUITECTURA DE SISTEMAS DISTRIBUIDOS. 3º GTI

PRÁCTICA 3A (DIG). PLANIFICACIÓN DINÁMICA Y PROCESADORES SUPERESCALARES: BUCLES VECTORIZABLES.

OBJETIVOS.

En esta práctica se trata de estudiar el algoritmo de Tomasulo y algunos ejemplos simples de bucles vectorizables para distintos grados de superescalaridad m .

Se manejará un simulador visual desarrollado como un proyecto Fin de carrera dentro del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Sevilla: SUPERTOMASIM (SUPERescalarTOMasuloSIMulator). Éste simula un procesador didáctico superescalar y superencadenado de un modelo de RISC típico de 32 bits (llamado DLX), que dispone de “scheduling” dinámico (algoritmo de Tomasulo). Además son configurables tanto el grado de superescalaridad como los recursos del procesador: F.U. (Functional Units), CDB (Common Data Bus), RS (estaciones de reserva), etc.

PREPARACIÓN.

Fichero de código para preparar.

Trabajaremos primero con un bucle simple

; Cabecera con definición de variables e inicialización de registros

.data ; las siguientes etiquetas sirven para obtener la dirección de comienzo de vectores o variables:

arrayx: .float 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25

arrayy: .float 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27

a: .float 3 ; Constante a . Además sirve de dirección donde acaba el vector y.

.text

addi r1,r0, arrayx ;Puntero a Arrayx

addi r2,r0, arrayy ;Puntero a Arrayy

addi r3,r0, a ;Puntero a variable a

lf f11, 0(r3) ; carga el valor de a en f11

; Bucle

bucle: lf f5, 0(r1) ; carga elem del vector x

lf f6, 0(r2) ; carga elem del vector y

multf f5, f11, f5

addf f5, f6, f5

sf 0(r2), f5

addi r1, r1, 4

addi r2, r2, 4

slti r3, r2, a

bnez r3, **bucle**

Piense y resuelva las siguientes preguntas sobre el código dado. Anótelas en la tabla de resultados:

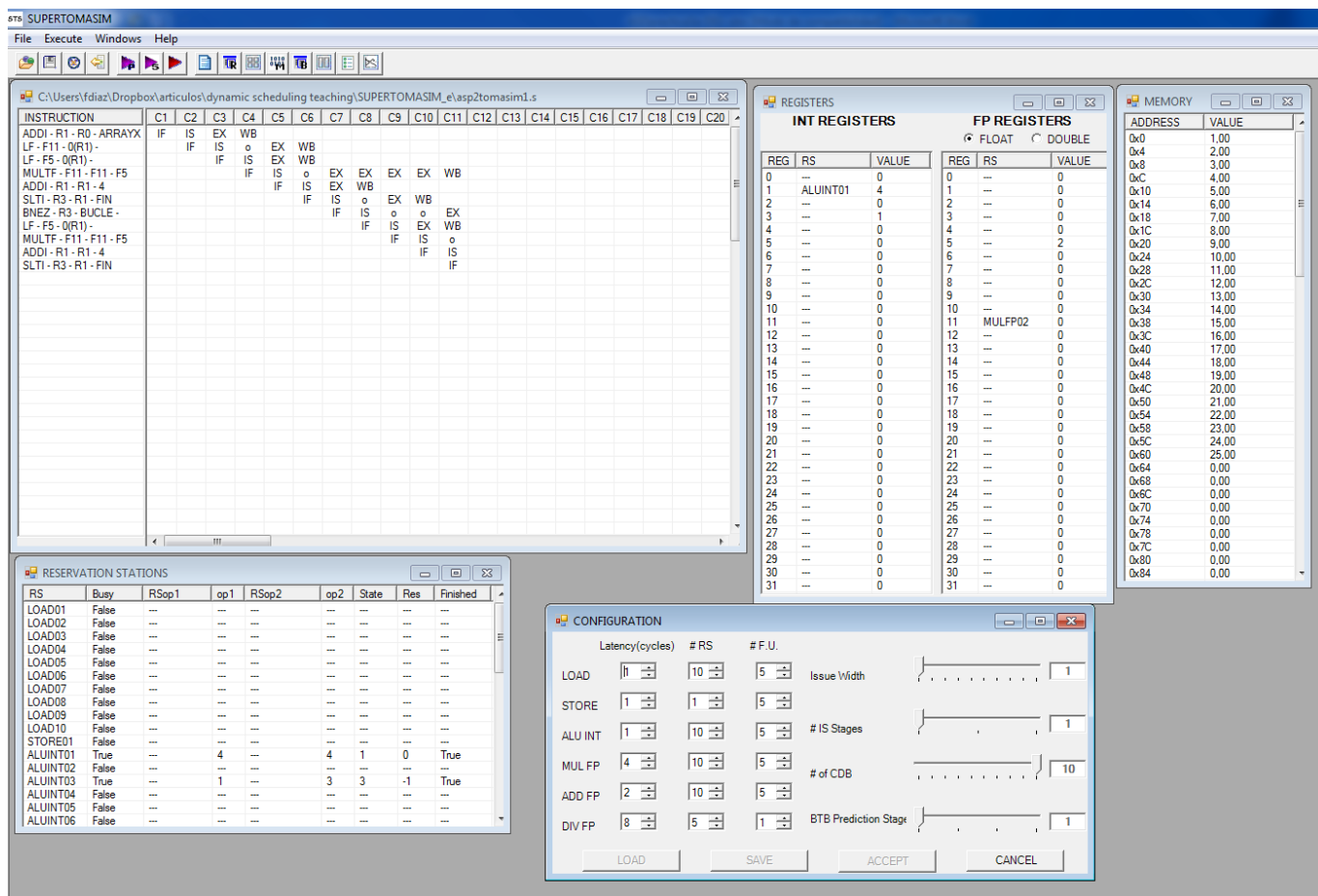
- Dibuje las dependencias reales dentro de una iteración
- Tradúzcalo a lenguaje C o similar.
- ¿Se puede desenrollar? ¿Se pueden computar en paralelo todos los elementos de los vectores x, y (es decir, es vectorizable)? Razonar.

REALIZACIÓN DE LA PRÁCTICA.

Descripción del simulador SUPERTOMASIM.

Este simulador admite cualquier código RISC (del llamado procesador DLX). Como se ve en la siguiente figura, muestra el cronograma que se está ejecutando y el estado completo (RS, registros, memoria, etc.) del procesador simulado.

Las etiquetas (nombre del registro interno tras el renombrado) se llaman igual que las RS (LOAD01, LOAD02,... , ALUINT01, ALUINT02, etc.). Los registros y RS que esperan por una etiqueta (por existir una dependencia real RAW) muestran tal nombre en lugar del valor (por ejemplo en los registros R1, F11 por las RS ALUINT01, MULFP02 de la figura).



Además permite configurar en el menú *windows->configuration* (ver cuadro de diálogo inferior derecho de la imagen):

- Duración de cada F.U. (Latency)
- Número de RS (estaciones de reserva) (#RS)
- Número de F.U. (#F.U.)
- m = Grado de Superescalaridad (Issue Width)
- Grado de superencadenamiento de la fase IS (IS Stages)
- Número de CDB (# of CDB)
- Fase en la que se realiza la predicción de la BTB (BTB Prediction Stage)

El **cronograma y configuración se pueden salvar** en ficheros aparte (el cronograma tiene cada fase separada por tabuladores para que se pueda copiar fácilmente por ejemplo en una hoja de Excel).

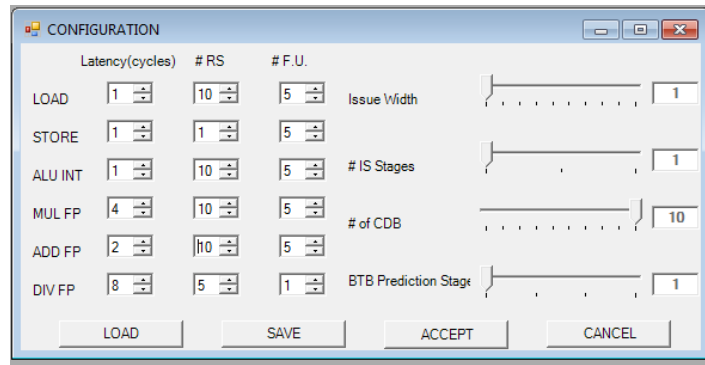
Ejecutar el Simulador SUPERTOMASIM. El simulador es la primera parte de un proyecto de simulación de procesadores superescalares, y algunos detalles de la interfaz de usuario están aún incompletos, son poco amigables o no detectan ciertos errores de entrada. Por tanto, se van a dar los ficheros de ensamblador en el laboratorio para evitar que salgan ciertos errores.

Algunas consideraciones sobre la arquitectura que simula SUPERTOMASIM son:

- **Cada bloqueo estructural se muestra repitiendo la fase** que no puede avanzar. Por ejemplo: IF IF IF IS significa que han habido dos ciclos de bloqueo porque IS no pudo ejecutarse (según notación de clase: IF - - IS).
- **La BTB** predice saltos como siempre tomados (se podría cambiar esto usando un fichero llamado *BTB.txt*)
- Para **cambiar cualquier parámetro de la configuración** (cuadro de diálogo de configuración: *windows->configuration*) o para **cambiar de código simulado**, debe primero resetear el simulador con la opción: *File->Reset Sim*. Se pierde evidentemente todo el cronograma anterior.


1. Para bucle vectorizable (RISC ESCALAR)

La configuración que vamos a usar es la siguiente. Utilice *windows->configuration* y luego **LOAD**. Cargue el fichero *asdtomasim1.xml* (si estuviera deshabilitado la carga, resetear el simulador con la opción: *File->Reset Sim*). Para esta prueba 1, se han puesto muchos CDBs y muchas F.U. para que no haya esperas por estos problemas estructurales. Además, hay bastantes R.S. excepto de Store (sólo hay 1 de Store, con objeto de que aparezcan bloqueos totales más adelante).



Ahora cargue el código *asdtomasim1.s*.

Se pide:

- d) Ejecute **paso a paso** (tecla F7) o de 5 en 5 ciclos (tecla F8) hasta que la primera Store se haya emitido (haya hecho la fase IS). Si es necesario, mueva la barra de Scroll del cronograma para poder ver los últimos ciclos. Dibuje el contenido de las R.S. (que están ocupadas) de ADDF, MULF, SF justo cuando la primera Store se haya emitido (haya hecho la fase IS). Explique por qué las R.S. tienen este contenido.
- e) Ejecute la primera iteración **paso a paso** (tecla F7) o de 5 en 5 ciclos (tecla F8), o de 20 en 20 (icono ). Mueva Scroll del cronograma para poder ver los últimos ciclos. Copie el cronograma de la primera iteración y compruebe que las dependencias reales están retrasando las fases EX (dibuje con flechas de WB a EX los bypasses a través del CDB)
- f) ¿Cuál es la cadena crítica de ejecución (cadena de RAWs más larga) en una única iteración y cuantos ciclos dura?
- g) Ejecute más iteraciones hasta que se compruebe que se ha llegado a un “estacionario” del bucle, es decir, que todas las iteraciones tienen el mismo cronograma. ¿Cuánto vale el CPIreal del bucle en el estacionario? NOTA: para medir ciclos reales, cuente desde la fase IS del primer LF de una iteración, hasta la fase IS del primer LF de la siguiente iteración.

2. Para bucle vectorizable (RISC SUPERESCALAR)

Resetear el simulador con la opción: *File-> Reset Sim*.

Modifique la configuración para que sea superescalar de grado 3 (Issue Width=3)

Cargue el código *asdtomasim1.s*. Ejecute al menos 4 iteraciones.

Se pide:

- h) Hallar CPI ideal, real y CPI bloqueo para las iteraciones 3 y 4. Recuerde: para medir ciclos reales, cuente desde la fase IS del primer LF de iteración 3, hasta la fase IS del primer LF de iteración 5. El CPIbloqueo puede calcularlo restando CPIreal-CPIideal o contando los bloqueos (esto puede resultar más difícil).
- i) Resetear el simulador con la opción: *File-> Reset Sim* y repita lo mismo para superescalar de grado 6 (Issue Width=6). Hallar CPI ideal, real y CPI bloqueo para las iteraciones 3 y 4.
- j) ¿Por qué el CPI real es el mismo para altos grados de superescalaridad (Issue Width)?
- k) Resetear el simulador con la opción: *File-> Reset Sim*, y aumente el número de R.S. de Store (y tal vez de otras F.U.) hasta que no haya bloqueos por agotamiento de R.S. ¿Cree que el número máximo de R.S. de Store consumidos u ocupados (sin que haya bloqueos) es proporcional al grado de superescalaridad (Issue Width)? ¿Por qué?

APÉNDICE: SIMULADOR SUPERTOMASIM

Otros pormenores del simulador SUPERTOMASIM son:

- **El número máximo de RS de cada tipo es 150. El número máximo de F.U.de cada tipo es 10.** Podría interrumpirse la simulación si se superan estos límites (no se chequean estos rangos en la configuración).
- La configuración se puede salvaren un fichero (con la **opción GUARDAR/SAVE**) y luego recuperarla (**opción CARGAR/LOAD**).

Cuando escriba código DLX, debe tener en cuenta los siguientes pormenores:

- **No puede haber una línea en blanco al final del fichero.**
- No distingue entre mayúsculas y minúsculas.
- Detrás de la **directiva .text** (cuando se escriba el código), **no se pueden poner líneas en blanco**, ni líneas que comiencen por “;” (es decir, comentarios que ocupen toda la línea).
- Las **etiquetas de salto deben empezar en el primer carácter de la línea y no pueden estar solas en una línea**, sino que detrás del “:” debe ir una instrucción válida; por ejemplo, así:
`bucle: lbu r3, 0(r1) ; Carga Arrayx[i]`
- Los registros flotantes de doble precisión se llaman d0, d1, d2, etc. (no son válidos los f0, f2, etc.).

APELLIDOS:

NOMBRE:

PREPARACIÓN

a)	
b)	
c)	

TABLA DE RESULTADOS

d)	
----	--

e)	
f)	
g)	
h)	
i)	
j)	
k)	