Universidad de Alcalá
DEPARTAMENTO DE AUTOMÁTICA
ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

Laboratorio de Arquitectura e Ingeniería de Computadores
PRÁCTICA I

COMPUTADORES SEGMENTADOS (DLX)

OBJETIVO

El objetivo de la práctica es el estudio del funcionamiento de un computador segmentado, analizando la ejecución de programas en un simulador del computador DLX.

INTRODUCCIÓN TEÓRICA.

Vamos a exponer la arquitectura y conjunto de instrucciones del computador DLX. Este computador viene detallado en el libro "Arquitectura de computadores, un enfoque estructurado" de J. L. Hennessy y D. A.Patterson de la editorial McGraw Hill.

Arquitectura del computador DLX.

DLX tiene una sencilla arquitectura de carga-almacenamiento que se eligió basándose en las observaciones sobre las primitivas más frecuentemente usadas en la programación.

Consta de 32 registros de 32 bits de propósito general denominados R0 a R31. El registro R0 siempre tiene el valor 0

Además incorpora un conjunto de registros de punto flotante (FPR) que pueden ser utilizados como registros de simple precisión (32 bits) o agrupados en pares como registros de doble precisión (64bits), estos últimos se denominan F0, F2, ..., F28, F30.

La memoria esta estructurada en bytes con una dirección de 32 bits.

Operaciones.

Hay cuatro clases de instrucciones: Carga y almacenamiento, operaciones de la ALU, saltos y bifurcaciones y operaciones en punto flotante.

Tenemos un único modo de direccionamiento (registro base más desplazamiento de 16 bits con signo).

A continuación se da una relación de las instrucciones disponibles en el DLX y algunos ejemplos de instrucciones concretas con la operación que realizan.

OPERACIONES DE TRANSFERENCIA DE DATOS

Transfiere datos entre registros y memoria, o entre registros enteros y FP o registros especiales; el modo de direccionamiento de memoria es un desplazamiento de 16 bits + contenido de un GPR.

INSTRUCCIÓN	SIGNIFICADO
LB, LBU, SB	Carga byte, carga byte sin signo, almacena byte
LH, LHU, SH	Carga media palabra, carga media palabra sin signo, almacena media palabra.
LW, SW	Carga palabra, almacena palabra (a/desde registros enteros)
LF, LD, SF, SD	Carga punto flotante SP, carga punto flotante DP, almacena punto flotante SP, almacena punto flotante DP.
MOVI2S, MOVS2I	Transfiere desde/a GPR a/desde registro especial
MOVF, MOVD	Copia un registro de punto flotante o un par en DP en otro registro o par
MOVFP2I, MOVI2FP	Transfiere 32 bits desde/a registros FP a/desde registros enteros

OPERACIONES ARITMÉTICAS/LÓGICAS

Operaciones sobre datos enteros o lógicos en GPR/s; la aritmética con signo causa un trap en caso de desbordamiento.

INSTRUCCIÓN	SIGNIFICADO
ADD, ADDI, ADDU, ADDUI	Suma, suma inmediato (todos los inmediatos son de 16 bits); con signo y sin signo
SUB, SUBI, SUBU, SUBUI	Resta, resta inmediata; con signo y sin signo
MULT, MULTU, DIV, DIVU	Multiplica y divide, con signo y sin signo; los operandos deben estar en registros de punto flotante; todas las operaciones tienen valores de 32 bits
AND, ANDI	And, and inmediato
OR, ORI, XOR, XORI	Or, or inmediato, or exclusiva, or exclusiva inmediata
LHI	Carga inmediato superior: carga la mitad superior de registro con inmediato
SLL, SRL, SRA, SLLI, SRLI, SRAI	Desplazamientos: ambos inmediatos (S___I) y forma variable (S___) los desplazamientos son desplazamientos lógicos a la izquierda, lógicos a la derecha, aritméticos a la derecha.
S___, S___I	Inicialización condicional: "___" puede ser LT, GT, LE, GE, EQ, NE.

OPERACIONES DE CONTROL Saltos y bifurcaciones condicionales; relativos al PC o mediante registros	
INSTRUCCIÓN	**SIGNIFICADO**
BEQZ, BNEZ	Salto GPR igual/no igual a cero; desplazamiento de 16 bits desde PC+4
BFPT, BFPF	Test de bit de comparación en el registro de estado FP y salto; desplazamiento de 16 bits desde PC+4
J, JR	Bifurcaciones: desplazamiento de 26 bits desde PC (J) o destino en registro (JR)
JAL, JALR	Bifurcación y enlace: guarda PC+4 en R31, el destino es relativo al PC (JAL) o un registro (JR)
TRAP	Transfiere a sistema operativo a una dirección vectorizada
RFE	Volver al código del usuario desde una excepción; restaurar modo de usuario

OPERACIONES DE PUNTO FLOTANTE Operaciones en punto flotante en formatos DP y SP	
INSTRUCCIÓN	**SIGNIFICADO**
ADDD, ADDF	Suma números DP, SP
SUBD, SUBF	Resta números DP, SP
MULTD, MULTF	Multiplica punto flotante DP, SP
DIVD, DIVF	Divide punto flotante DP, SP
CVTF2D, CVTF2I, CVTD2F, CVTD2I, CVTI2F, CVTI2D	Convierte instrucciones: CVTx2y convierte de tipo x a y
___D, ___F	Compara DP y SP: "___" puede ser LT, GT, LE, EQ, NE; pone bit de comparación en registro de estado FP.

EJEMPLOS DE INSTRUCCIONES		
Instrucción	**Operación**	**Significado**
LW R1,30(R2)	Cargar palabra	$R1 \leftarrow_{32} M[30+R2]$
LW R1,1000(R0)	Cargar palabra	$R1 \leftarrow_{32} M[1000+0]$
LB R1,40(R3)	Cargar byte	$R1 \leftarrow_{32} (M[40+R3]_0)^{24}$ ## M[40+R3]
LBU R1,40(R3)	Cargar byte sin signo	$R1 \leftarrow_{32} 0^{24}$ ## M[40+R3]
LH R1,40(R3)	Cargar media palabra	$R1 \leftarrow_{32} (M[40+R3]_0)^{16}$ ## M[40+R3] ## M[41+R3]
LF F0,50(R3)	Cargar flotante	$F0 \leftarrow_{32} M[50+R3]$
LD F0,50(R2)	Cargar doble	F0 ## $F1 \leftarrow_{64} M[50+R2]$
SW 500(R4),R3	Almacenar palabra	$M[500+R4] \leftarrow_{32} R3$
SF 40(R3),F0	Almacenar flotante	$M[40+R3] \leftarrow_{32} F0$
SD 40(R3),F0	Almacenar doble	$M[40+R3] \leftarrow_{32} F0$; $M[44+R3] \leftarrow_{32} F1$
SH 502(R2),R3	Almacenar media	$M[502+R2] \leftarrow_{16} R3_{16..31}$
SB 41(R3),R2	Almacenar byte	$M[41+R3] \leftarrow_{8} R3_{24..31}$

EJEMPLOS DE INSTRUCCIONES		
Instrucción	**Operación**	**Significado**
J nombre	Bifurcación	PC←nombre; $((PC+4)-2^{15}) \leq nombre < ((PC+4)+2^{15})$
JAL nombre	Bifurcación y enlace	R31←PC+4; PC←nombre; $((PC+4)-2^{15}) \leq nombre < ((PC+4)+2^{15})$
JALR R2	Bifurcación y enlaza registro	R31←PC+4; PC←R2
JR R3	Bifurcación a registro	PC←R3
BEQZ R4, nombre	Salta igual a cero	If (R4==0) PC←nombre; $((PC+4)-2^{15}) \leq nombre < ((PC+4)+2^{15})$
BNEZ R4, nombre	Salta no igual a cero	If (R4!=0) PC←nombre; $((PC+4)-2^{15}) \leq nombre < ((PC+4)+2^{15})$
ADD R1,R2,R3	Suma	R1 ← R2 + R3
ADDI R1,R2,#3	Suma inmediato	R1 ← R2 + 3
LHI R1,#42	Cargar alto inmediato	R1 ← 42 ## 0^{16}
SLL R1,R2,#5	Desplazamiento lógico a la izquierda	R1 ← R2 << 5
SLT R1,R2,R3	Inicializar menor que	If (R2 < R3) R1 ← 1 Else R1 ← 0

Segmentación

En la segmentación del DLX se han definido las siguientes cinco etapas:

IF	búsqueda de la instrucción
ID	decodificación de instrucción y búsqueda de registros
EX	Ejecución de instrucción y cálculo de direcciones efectivas
MEM	Acceso a memoria
WB	posescritura en registro

La segmentación del DLX está dotada de la técnica hardware llamada adelantamiento que permite evitar detenciones en el cauce cuando una operación necesita el resultado de la anterior. Cuando el detención se produce por cargas escalares se puede evitar, en algunos casos, reordenando el código. A continuación se muestra un ejemplo de reordenación de código

El estado del cauce en una secuencia de ejecución de una operación de carga y operación aritmética es la siguiente:

Instrucción	**IF**	**ID**	**EX**	**MEM**	**WB**				
lw r1, a		IF	ID	EX	MEM	WB			
add r3,r2,r1			IF	ID	*detención*	EX	MEM	WB	
Instrucción				IF	*detención*	ID	EX	MEM	WB

La operación de suma debe esperar a que el dato a sea leído de memoria, esta espera se transmite al resto de instrucciones.

En el siguiente programa se producen dos esperas por este motivo.

```
        .data
a:      .word  7
b:      .word  12
c:      .word  30
r:      .word  0

        .text
ini:    lw     r1,a
        lw     r2,b
                       ; espera para lectura de b
        add    r3,r2,r1
        lw     r4,c
                       ;espera para lectura de c
        add    r5,r3,r4
        sw     r,r5
        trap   #6
```

La ejecución de este programa en DLX produce dos ciclos de parada. Estas esperas pueden eliminarse reestructurando el programa de la forma siguiente:

```
        .data
a:      .word  7
b:      .word  12
c:      .word  30
r:      .word  0

        .text
ini:    lw     r1,a
        lw     r2,b
        lw     r4,c
        add    r3,r2,r1
        add    r5,r3,r4
        sw     r,r5
        trap   #6
```

El tratamiento de los saltos en DLX se hace por medio de la predicción de salto no efectivo:

Predicción de salto no efectivo.- Supone que el salto no se va a producir evitando tener que esperar hasta la decisión de salto para cargar la siguiente instrucción. Si la predicción fue errónea anula la instrucción cargada y carga la indicada por el salto.

DESARROLLO DE LA PRACTICA

1.- Realizar un programa para el DLX que sume dos vectores de 10 componentes, sin usar bucles. Analizar los resultados estadísticos y las dependencias que producen detenciones en el cauce.

2.- Modificar el programa anterior para evitar las detenciones, comprobar los resultados.

3.- Repetir la operación usando un bucle para la suma de componentes.

4.- Optimizar el programa del apartado anterior, comprobar resultados.