INGENIERÍATÉCNICA EN INFORMÁTICA DE GESTIÓN. CURSO 2001-2002

LABORATORIO DE ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

Sesión 10
Instrucciones aritmético-lógicas y desplazamientos

INSTRUCCIONES ARITMÉTICAS, LÓGICAS Y DE DESPLAZAMIENTO

En esta sesión se presentan las instrucciones que permiten realizar operaciones aritméticas, lógicas y de desplazamiento de datos. Las instrucciones aritméticas están constituidas por las operaciones de suma y resta (add, addu, addi, addiu, sub, subu) y las operaciones de multiplicación y división (mult, multu, div, divu). La diferencia entre las instrucciones acabadas o no acabadas en u (por ejemplo entre add y addu) está en que las primeras provocan una excepción de desbordamiento si el resultado de la operación no es representable en 32 bits y las segundas no tienen en cuenta el desbordamiento. Recordar que el rango de representación de los números enteros con signo de 32 bits en complemento a 2 va desde –2.147.483.648 a 2.147.483.647 (0x80000000 a 0x7fffffff).

Dentro del grupo de instrucciones que permiten realizar operaciones lógicas están: suma lógica (or y ori), producto lógico (and y andi) y la or exclusiva (xor y xori).

Finalmente, se presentan las instrucciones de desplazamiento aritmético y lógico (sra, sll, srl).

Operaciones aritméticas con datos inmediatos

Crea un fichero con el siguiente código:

```
            .data    #zona de datos
            #Máx. Positivo representable 0x7FFFFFFF
numero:     .word 2147483647
            .text    #zona de instrucciones
main:       lw     $t0, numero($0)
            addiu  $t1, $t0, 1
```

Descripción:

La instrucción “addiu” es una instrucción de suma con un dato inmediato y con detección de desbordamiento.

Borra los valores de la memoria, carga el fichero y ejecútalo paso a paso.

Cuestión 1: Localiza el resultado de la suma efectuada. Comprueba el resultado.

Cambia la instrucción “addiu” por la instrucción “addi”.

Borra los valores de la memoria, carga el fichero y ejecútalo paso a paso.

Cuestión 2: ¿Qué ha ocurrido al efectuar el cambio? ¿Por qué?

Operaciones aritméticas con datos en memoria

Crea un fichero con el siguiente código:

```
                .data
numero1:    .word 0x80000000 #max. Negativo represent.
numero2:    .word 1
numero3:    .word 1
            .text
main:
            lw    $t0,numero1($0)
            lw    $t1,numero2($0)
            subu  $t0,$t0,$t1
            lw    $t1,numero3($0)
            subu  $t0,$t0,$t1
            sw    $t0,numero1($0)
```

Borra los valores de la memoria, carga el fichero y ejecútalo paso a paso.

Cuestión 3: ¿Qué hace el programa anterior? ¿Qué resultado se almacena en numero1? ¿Es correcto?

Cuestión 4: ¿Se producirá algún cambio si las instrucciones “subu” son sustituidas por instrucciones “sub”? ¿Por qué?

Multiplicación y división con datos en memoria

Crea un fichero con el siguiente código:

```
            .data
numero1:    .word 0x7FFFFFFF #Máx. Positivo represent.
numero2:    .word 16
            .space      8
            .text
main:       lw $t0,numero1($0)
            lw $t1,numero2($0)
            mult $t0,$t1      # multiplica los dos números
            mfhi $t0
            mflo $t1
            sw $t0,numero2+4($0) #32 bits más peso
            sw $t1,numero2+8($0) #32 bits menos peso
```

Descripción:

El código realiza la multiplicación de dos números, almacenando el resultado de la multiplicación a continuación de los dos multiplicandos.

Borra los valores de la memoria, carga el fichero y ejecútalo.

Cuestion 5: ¿Qué resultado se obtiene después de realizar la operación? ¿Por qué se almacena en dos palabras de memoria?

Cuestión.6: Modifica los datos anteriores para que numero1 y numero2 sean 10 y 3, respectivamente. Escribe el código que divida numero1 entre numero2 (dividendo y divisor, respectivamente) y coloque el cociente y el resto a continuación de dichos números.

Operaciones lógicas

Crea un fichero con el siguiente código:

```
            .data
numero:     .word 0x3ff41
            .space      4
            .text
main:       lw $t0,numero($0)
            andi $t1,$t0,0xfffe
            # 0xffe en binario es 0...0111111111111110
            sw $t1,numero+4($0)
```

Descripción:

El código anterior pone los 16 bits más significativos y el bit 0 de numero a 0, y almacena el resultado en la siguiente palabra (numero+4). La instrucción "andi" realiza el producto lógico, bit a bit, entre el dato contenido en un registro y un dato inmediato. El resultado obtenido es que aquellos bits que en el dato inmediato están a 1 no se modifican con respecto al contenido original del registro, es decir, los 16 bits de menor peso excepto el bit 0. Por otro lado, todos aquellos bits que en el dato inmediato están a 0, en el resultado también están a 0, es decir, los 16 bits de mayor peso y el bit 0. Los 16 bits de mayor peso del resultado se ponen a 0 puesto que, aunque el dato inmediato en este caso es de 16 bits, a la hora de realizar la operación el procesador trabaja con un dato de 32 bits.

Borra los valores de la memoria, carga el fichero y ejecútalo. Comprueba el resultado.

Cuestión 7: Modifica el código para obtener, en la siguiente palabra, que los 16 bits más significativos de numero permanezcan tal cual, y que los 16 bits menos significativos queden a cero, excepto el bit cero que también debe quedar como estaba.

Operaciones de desplazamiento

Crea un fichero con el siguiente código:

```
            .data
numero:     .word 0xffffff41
            .text
main:       lw $t0,numero($0)
            sra $t1,$t0,4
```

Descripción:

El código anterior desplaza el valor de numero cuatro bits a la derecha, rellenando el hueco generado a su izquierda con el bit del signo.

Borra los valores de la memoria, carga el fichero y ejecútalo.

Cuestión 8: ¿Para qué se ha rellenado numero con el bit del signo?

Cuestión 9: ¿Qué ocurre si se sustituye la instrucción "sra" por "srl"?

Cuestión 10: Modifica el código para desplazar el contenido de numero 2 bits a la izquierda.

Cuestión 11: ¿Qué operación aritmética acabamos de realizar?

Cuestión 12: Volviendo al código que iniciaba este apartado, indica qué operación aritmética tiene como consecuencia la ejecución de dicho código.

Problemas propuestos

1. Diseña un programa ensamblador que defina el vector de enteros de dos elementos V=(10,20) en la memoria de datos a partir de la dirección 0x10000000 y almacene su suma a partir de la dirección donde acaba el vector.
2. Diseña un programa ensamblador que divida los enteros 18,-1215 almacenados a partir de la dirección 0x10000000 entre el número 5 y que a partir de la dirección 0x10010000 almacene el cociente de dichas divisiones.
3. Pon a cero los bits 3,7,9 del entero 0xabcd12bd almacenado en memoria a partir de la dirección 0x10000000, sin modificar el resto.
4. Cambia el valor de los bits 3,7,9 del entero 0xff0f1235 almacenado en memoria a partir de la dirección 0x10000000, sin modificar el resto.
5. Multiplica el número 0x1237, almacenado en memoria a partir de la dirección 0x10000000, por 32 (2^5) sin utilizar las instrucciones de multiplicación ni las pseudoinstrucciones de multiplicación.