IS-603 Arquitectura de Computadoras

I Parcial - Instrucciones de Transformación de datos

IS-UNAH

I PAC 2024

Contenido

Los números en el Computador

Puntos Clave

1. Los números (y toda la información) en el computador se codifica en binario
2. Hasta ahora hemos trabajado con números codificados en binario natural
3. Para representar enteros se utiliza una representación de Complemento a 2: Ca_2

Complemento a 2: Ca_2

1. Se utiliza para representar números enteros, en binario
2. Dos números operados en Ca_2 darán como el resultado correcto codificado en Complemento a 2
3. **Aplicamos la operación de Ca_2 a los números negativos, utilizando un bit para representar signo**
4. Los números positivos se representan en binario natural, teniendo cuidado de utilizar el MSB como signo.
5. Si tenemos un número entero, el bit más significativo representa el signo.
   - 5.1 Si el MSB es 0, el número es 0 o positivo
   - 5.2 Si el MSB es 1, el número es negativo

Bits de acarreo

1. Sirve para indicar si el resultado obtenido no es correcto, o
2. para permitir sumas o restas de diversas palabras
3. En el caso de la suma, el resultado es correcto si el bit de acarreo es 0.
4. En el caro de la resta, el resultado es correcto si el bit de acarreo es 1

Bit de desbordamiento

Cuando se opera números en Ca_2 el bit de desbordamiento indica si el resultado obtenido es correcto o no. Depende de:

1. La operación realizada: suma o resta.
2. Del signo de los números con los que se está operando
3. Del signo esperado del resultado

Contenedor de k bits

Para operar números en Ca_2 se debe tener en cuenta establecer el tamaño del contenedor del número binario. Este número se llamará k, y $k = 8, 16, 32$ bits.

Una vez definido k podemos definir el rango de enteros que se puede codificar, utilizando la siguiente expresión

$$n \in [-2^{k-1}, 2^{k-1} - 1]$$

Suma binaria

La suma binaria sigue las siguientes reglas:

1. 0 + 0 = 0
2. 0 + 1 = 1
3. 1 + 0 = 1
4. 1 + 1 = 10 (0, acarreo=1)

Suma binaria

Encontrar el resultado de sumar los dos valores:

0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1

Suma binaria: resultado

Acarreo	0	0	1	1	1	1	1		
Binario1	0	0	0	0	1	1	1	1	+
Binario2	0	0	0	1	0	0	0	1	
Result.	0	0	1	0	0	0	0	0	

Aplicar complemento a 2

Encontrar la representación de Ca2[a]

1 0 1 1 1 0 0 1

[a] $k = 8$ bits. El número representado es negativo, su MSB=1. Al aplicar Ca2 a un número negativo, obtenemos el valor absoluto del número. ¿Cuál es el valor absoluto del número dado?

Aplicar complemento a 2

Primero encontramos el Ca1[b]:

Binario	1	0	1	1	1	0	0	1
Ca1	0	1	0	0	0	1	1	0

Después sumamos 1 al Ca1, para obtener Ca2: [c]

Acarreo	0	0	0	0	0	0	0	
Ca1	0	1	0	0	0	1	1	0
$+1_2$	0	0	0	0	0	0	0	1
Ca2	0	1	0	0	0	1	1	1

+

[b] Ca1 es equivalente a NOT (binario)

[c] Los ceros en el acarreo se colocan con fines demostrativos, podemos omitirlos.

Aplicar complemento a 2

Encontrar la representación de Ca2[d]

-37

[d] $k = 8$ bits.

Aplicar complemento a 2

Econtramos el Ca1 del valor absoluto del número:

Abs(−37)	37							
Binario	0	0	1	0	0	1	0	1
Ca1	1	1	0	1	1	0	1	0

Después sumamos 1 al Ca1, para obtener Ca2: [e]

Acarreo	0	0	0	0	0	0	0	
Ca1	1	1	0	1	1	0	1	0
$+1_2$	0	0	0	0	0	0	0	1
Ca2	1	1	0	1	1	0	1	1

+

[e] Los ceros en el acarreo se colocan con fines demostrativos, podemos omitirlos.

Realizar las siguientes operaciones

Todos los operandos son números enteros y $k = 8$ bits. Dar el resultado según se indica después del signo de igualdad [f].

1. 45 + 1110 0000 = 0b
2. 1100 0110 + 1110 0101 = (dec)
3. −63−18 = 0b
4. 0010 1101 + 0010 0000 + 0001 0001 = 0b
5. 1100 0001 + 1010 1011 = (dec)

[f] Al finalizar el ejercicio 5, ir a Diapositiva 7

Extensión de signo

Un número entero (Ca2) representado con k=8 bits, puede representarse con 16 y 32 bits con solo fijarnos en el bit de signo.

Así, el número 1111 1000 se puede representar en 16 bits extendiendo el bit de signo (i.e. repitiendo) hasta completar los 16 bits:

1111 1111 1111 1111 **1**111 1000 [g]

[g] ¿Qué número (decimal) está representado en el ejemplo? ¿Cuál sería su representación en hexadecimal? ¿Cuál sería la representación con 32 bits?

Banco de Registros de ARM

Banco de Registros

1. Del 1 al 12 son registros de propósito general
2. Del 13 al 15 son de propósito específico.
3. Del 1 al 7 son registros altos, usables para los programas.
4. Del 8 al 15 son registros bajos, usables por instrucciones específicas.
5. Hay un registro no visible al programador que es el registro de estado.

Registro de estado

Indica el estado actual del procesador, empleando para ellos los indicadores de condición.

- **N** Se activa cuando el resultado de la operación realizada es negativo.
- **Z** Se activa cuando el resultado de la operación realizada es cero.
- **C** Se activa cuando el resultado de la operación produce un acarreo.
- **V** Se activa cuando el resultado de la operación produce un desbordamiento.

Operaciones Aritméticas

Suma y Resta

- **mov** rd, #Inm8 : $rd \leftarrow Inm8$ [h]
- **add** rd, rs, rn : $rd \leftarrow rs + rn$
- **sub** rd, rs, rn : $rd \leftarrow rs - rn$
- **add** rd, rs, #Inm3 : $rd \leftarrow rs + Inm3$
- **sub** rd, rs, #Inm3 : $rd \leftarrow rs - Inm3$
- **add** rd, #Inm8 : $rd \leftarrow rd + Inm8$
- **sub** rd, #Inm8 : $rd \leftarrow rd - Inm8$
- **cmp** rs, rn : $rs - rn$ [i]

[h] mov es una operación de carga, pero se ve en este capítulo para simplificar los ejercicios.

[i] Observe que no guarda resultado pero sí activa los indicadores de condición

Operaciones Lógicas

Operaciones Lógicas

1. **and** rd, rs: $rd \leftarrow rd \ AND \ rs$
2. **tst** rn, rm: $rn \ AND \ rm$[j]
3. **orr** rd, rs: $rd \leftarrow rd \ OR \ rs$
4. **eor** rd, rs: $rd \leftarrow rd \ EOR \ rs$ [k]
5. **mvn** rd, rs: $rd \leftarrow \ NOT \ rs$

[j] No guarda el resultado

[k] Operación XOR: Exclusive OR: EOR

Operaciones de Desplazamiento

Desplazamiento aritmético a la derecha

Las operaciones de desplazamiento aritmético conservan el signo del contenido del registro. Una utiliza dato inmediato, y la otra utiliza el contenido de un registro.

- **asr** `rd, rm, #Shift:` $rd \leftarrow rm >>_a Shift$
- **asr** `rd, rs:` $rd \leftarrow rd >>_a rs$

Desplazamiento lógico a la derecha

Las operaciones de desplazamiento lógico no conservan el signo del contenido del registro. Una utiliza dato inmediato, y la otra utiliza el contenido de un registro.

- **lsr** rd, rm, #Shift: $rd \leftarrow rm >>_l Shift$
- **lsr** rd, rs: $rd \leftarrow rd >>_l rs$

Desplazamiento lógico a la izquierda

- **lsl** `rd, rm, #Shift:` $rd \leftarrow rm << Shift$
- **lsl** `rd, rs:` $rd \leftarrow rd << rs$

Ejercicios

Analizar las instrucciones del siguiente código:

```
1           .text
2  main:    mov r2, #250
3           ldr r0, =0xffffff41
4           mov r1, #4
5           mov r3, #0b1101101
6           mov r4, #0x7
7           add r5, r3, r4
8           asr r0, r1
9           lsr r0, r1, #0b10
10 stop:    wfi
```

Ejercicios propuestos

Realice los siguientes ejercicios, con el simulador. Antes de escribir el código, analice los enunciados para ver el formato y tipo de las instrucciones. Es recomendable que haga la prueba de escritorio de cada ejercicio.

1. 3.15
2. 3.16
3. 3.17
4. 3.19
5. 3.20