.UBAfiuba
FACULTAD DE INGENIERÍA

Intérprete de Máquina Ideal RISC

Una mirada interactiva a la arquitectura de una computadora en funcionamiento

El intérprete de una máquina ideal fue desarrollado como Trabajo Profesional de Ingeniería en Informática en el año 2024 por los estudiantes Balmaceda, Fernando; Bocaccio, Agustina; Grati, Lucas y Pinto, Nicolás, con la tutoría del Prof. Arturo Carlos Servetto.

Especificaciones de la Máquina Ideal

Memoria principal con 256 celdas de 1 byte con direcciones representadas en patrones hexadecimales desde el 00 al FF.

CPU con 16 registros de propósito general de 1 byte, numerados de 0 a F con patrones hexadecimales.

Contador de programa de 1 byte y registro de instrucción de 2 bytes.

Valores enteros representados en complemento a 2 en 8 bits, y fraccionarios representados en punto flotante en 8 bits con desvío (bias) 4 y primer bit significativo oculto.

Entrada/salida en correspondencia con la memoria:

- Puerto de entrada en celda FE
- Puerto de salida en celda FF

Instrucciones

Las letras R, S y T representan números de registros en hexadecimal (0 a F).

Las letras XY representan dos dígitos hexadecimales que configuran bien una dirección de la memora RAM (entre 00 y FF) o bien un patrón particular (valor inmediato).

Código	Operandos	Descripción
1	RXY	Cargar en el registro R el contenido de la celda con dirección XY
2	RXY	Cargar en el registro R el valor XY
3	RXY	Almacenar el contenido del registro R en la celda con dirección XY
4	0RS	Copiar el contenido del registro R en el registro S
5	RST	Sumar en complemento a 2 los contenidos de los registros S y T y dejar el resultado en R
6	RST	Sumar en punto flotante los contenidos de los registros S y T y dejar el resultado en R

Código	Operandos	Descripción
7	RST	Disyunción (OR) de los contenidos de los registros S y T con resultado en registro R
8	RST	Conjunción (AND) de los contenidos de los registros S y T con resultado en registro R
9	RST	Disyunción exc. (XOR) de los contenidos de los registros S y T con resultado en registro R
A	R0X	Rotar a derecha el contenido del registro R, X veces
B	RXY	Saltar a la instrucción con dirección XY si el contenido del registro R es igual al del reg. 0
C	000	Parar la ejecución

Operaciones lógicas

Combinan dos cadenas de bits para producir otra cadena como resultado, todas del tamaño de registros de propósito general.

Generalmente se utilizan para aislar o alterar parte de la cadena de bits contenida en un registro, configurando un patrón en otro registro denominado **máscara**. Por ejemplo:

- Para conocer o cambiar el signo de un número en complemento a 2
- Para conocer o cambiar el signo de un número en punto flotante
- Para aislar el exponente en exceso o la mantisa de un número en punto flotante

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

Ejemplos

Conocer el signo de un número en complemento a 2 de 8 bits:

```
     bbbbbbbb      00001010 (10)       11110110 (-10)
AND  10000000      10000000            10000000
     b0000000      00000000 (0 ≡ +)    10000000 (128 ≡ -)
```

Cambiar el signo de un número en complemento a 2 de 8 bits:

1. Se invierten los bits de la cadena
2. Se suma 1 a la cadena invertida

```
    00001010 ( 10)  11110110 (-10)
XOR 11111111        11111111
    11110101        00001001
  + 00000001 (  1)  00000001 (  1)
    11110110 (-10)  00001010 ( 10)
```

Conocer el signo de un número en punto flotante de 8 bits:

```
     seeemmmm       01101010 (6.5)[1]     11101010 (-6.5)
AND  10000000       10000000              10000000
     s0000000       00000000 (0 ≡ +)      10000000 (128 ≡ -)
```

Cambiar el signo de un número en punto flotante de 8 bits:

```
    01101010            11101010
XOR 10000000            10000000
    11101010            01101010
```

Aislar el exponente en exceso de 4 de un número en punto flotante de 8 bits:

```
     seeemmmm       11101010
AND  01110000       01110000
     0eee0000       01100000
```

Aislar la mantisa de un número en punto flotante de 8 bits:

```
     seeemmmm       01101010
AND  00001111       00001111
     0000mmmm       00001010
```

Rotación a derecha

Sirve como base para otras operaciones de corrimiento lógico (alineamiento de patrones, convirtiendo en ceros a los bits que rotan) o aritmético (divisiones o multiplicaciones por potencias de 2, convirtiendo en bits iguales que al del signo a los que rotan a derecha, o en ceros a los que rotan a izquierda para multiplicaciones, pero con riesgo de *overflow*).

Patrón de 8 bits en un registro	**Rotar 1 posición a derecha**	**Rotar 1 posición a izq. (8-1 a der.)**
11110000	01111000	01111000

Observar que, como regla general, rotar x posiciones a izquierda equivale a rotar 8-x posiciones a derecha.

[1] $01101010_{float} \equiv ((-1)^0 * \mathbf{1}.1010 * 10^{110-100})_2 = (1.1010 * 10^{10})_2 = 110.1_2 \equiv 6.5_{10}$

Ejemplos

- Para justificar a derecha exponentes en exceso de números en punto flotante de 8 bits, el patrón 0bbb0000 (previamente aislado) se rota o corre a derecha 4 bits con igual resultado: 00000bbb
- Para multiplicar un número en complemento a 2 de 8 bits por una potencia de 2, se corre a izquierda el patrón tantos bits cuantos indique el exponente de la potencia:

  $00000111 \times 10^{10} = 00011100$ $(7 \times 4 = 28)$
  $11111001 \times 10^{10} = 11100100$ $(-7 \times 4 = -28)$

- Para dividir un número en complemento a 2 de 8 bits por una potencia de 2, se corre aritméticamente a derecha el patrón tantos bits cuantos indique el exponente de la potencia:

  $00010100 / 10^{10} = 00000101$ $(20 / 4 = 5)$
  $11101100 / 10^{10} = 11111011$ $(-20 / 4 = -5)$
  $11111001 / 10^{1} = 11111100$ $(-7 / 2 = -4)$ ← es división entera