Comparación de números enteros y los flags del microprocesador

Para determinar si un número es mayor o menor que otro tenemos varias cuestiones involucradas.

1. En ARC (filosofía RISC) no existe una instrucción específica para comparar números => restamos ambos números con SUBCC y miramos como quedan los flags
2. El análisis de flags es diferente si los valores son interpretados como enteros sin signo o como enteros con signo (complemento a 2)
3. Con cuatro flags debo encontrar criterios para distinguir: igual, distinto, mayor, menor, mayor o igual, menor o igual. Y en cada caso criterios para números con signo y números sin signo
4. Luego de restar números sin signo el análisis debería basarse en el concepto de Borrow. Sin embargo, los microprocesadores no tienen implementado un bit de borrow. La información de borrow es representada en el bit Carry que todos los microprocesadores sí poseen. Por lo tanto el bit C debe representar ambas cosas y depende del procesador el modo en que lo hace. En la arquitectura ARC (así como en SPARC), luego de la operación subcc el bit C toma el valor de borrow.

Código assembler para comparar dos números

La idea es (1°) restar ambos números y a continuación (2°) analizar los flags
Para analizar los flags el procesador cuenta con instrucciones específicas: los saltos condicionales (branch).
Dado que el análisis de flags es diferente si los números se interpretan como signados o no-signados, existen saltos condicionales para cada caso.

Tras un código del tipo

```
subcc  %r11, %r10, %r0
gbe   LabelX
```

el procesador produce o no el salto a la dirección LabelX en acuerdo con el contenido de los registros, el programador se limita a elegir el tipo de salto condicional adecuado, el procesador realiza automáticamente el análisis de los flags. Este análisis forma parte del proceso de ejecutar la instrucción de assembler.

Criterios generales para el análisis de los flags

Números sin signo:

1 . "Igualdad": ocurre cuando Z=1

2. "Mayor que": Si el primer argumento es mayor que el segundo su resta no produce un borrow y por lo tanto el procesador ARC pone un 0 en el carry. Sin embargo, la condición C=0 no asegura que el primer argumento sea mayor que el segundo: ambos podrían ser iguales dando también en este caso C=0. Por lo tanto, la condición que debe darse para que el primer argumento sea mayor que el segundo es que el carry este en 0 (descarto que sea menor) y también que el flag Z esté en 0 (descarto que sean iguales). La instrucción bgu, "branch if greater unsigned" da un 1 en caso de que ambas condiciones se verifiquen, por lo tanto puede definirse como f(C,Z)= Not(C or Z)

3. Las demás condiciones pueden definirse sobre la base de un razonamiento similar al anterior y son detalladas en la tabla más adelante.

Números con signo:

1 . "Igualdad": ocurre cuando Z=1

2. "Menor que": Si el primer argumento es menor entonces el resultado debe dar negativo (N=1). Esta es la condición básica, sin embargo podría darse que el resultado no sea representable en el sistema numérico (V=1) y solo en ese caso el resultado sería positivo. Las posibilidades de la condición "menor que" son: (resultado negativo y no hubo overflow) o (resultado positivo y hubo overflow). Nunca N=1 y V=1 ni tampoco nunca N=0 y V=0. Esto puede ser representado mediante f(N,V)=(N xor V)

3. Las demás condiciones pueden definirse sobre la base de un razonamiento similar al anterior y son detalladas en la tabla siguiente.

Instrucciones de Assembler ARC

	Branch	**Descripción**	**Función lógica**	
Con o sin signo	BE	Es igual	Z	Z=1
	BNE	Es distinto	Not(Z)	Z=0
Números sin signo				
	BGU	Es mayor	Not(C or Z)	
	BCC	Es mayor o igual	Not(C)	C =0
	BCS	Es menor	C	C=1
	BLEU	Es menor o igual	C or Z	
Números con signo	BG	Es mayor	not(Z or (N xor V))	
	BGE	Es mayor o igual	not(N xor V)	N = V
	BL	Es menor	N xor V	N ≠ V
	BLE	Es menor o igual	Z or (N xor V)	