

!Programa principal

.begin

```
.macro push arg
    add %r14,-4,%r14
    st arg,%r14
.endmacro
```

```
.macro pop arg
    ld %r14,arg
    add %r14,4,%r14
.endmacro
```

.extern comparar

push %r15! Como hay calls en el programa que lo sobrescriben, hay que guardar el r15 (de donde me llamaron) para despues salir al final de éste

```
                sethi 1111111110101010000000,%r1 !0xFFAA0000 -> r1
loop:    ld %r1, %r2                ! Dato completo -> r2
                sra %r2,16,%r3      !1er nro (16 bits mas sign.)-> %r3
                sll %r2,16,%r4      !paso intermedio para extender el signo
                sra %r4,16,%r4      !2do nro a r4

                subcc %r4,%r3,%r0    !los comparo
                be loop
                ba distintos
```

distintos: push %r3

```
                push %r4
                call comparar
```

```
                pop %r5
                subcc %r5,1,%r0      !me fijo si devolvio 1 o 0
                be fin               !si devolvio 1 termina el programa
                ba loop
```

```
fin:    pop %r15                !restauro el registro con la dir desde donde llamaron al
main    jmp! %r15+4, %r0 !retorno a donde me llamaron
```

.end

!subrutina

.begin

!definir de vuelta los macros

.extern comparar

```
                pop %r31!2do nro a r31
                pop %r30!1er nro a r30
```

```
                srl %r31,15,%r31  !agarro el bit de signo, el mas significativo
                srl %r30,15,%r30  !
```

```

        subcc %r30, %r31,%r0      !comparo
        be devolver1             !si los dos son 1 o los dos son 0, la resta da 0

        push %r0 !devuelvo 0
        ba fin

devolver1:      add %r0,1,%r31
                push %r31

fin: jmpl %r15+4,%r0

.end

```