La función *sort* está programada en assembler Risc-V en el archivo *sort-rv.s*. Esta función ordena ascendentemente un arreglo *nums* de *n* enteros sin signo usando un algoritmo ridículamente ineficiente. El código equivalente en C está comentado, mostrando la ubicación de las variables en los registros.

El encabezado de la función es: *void sort(unsigned int nums[ ], int n);*

El archivo *sort-rv-**nbits**.s* es una copia de *sort-rv.s*. Modifique la función *sort* en *sort-rv-**nbits**.s* de modo que ordene el arreglo descendentemente según la cantidad de bits en 1 de cada entero. La siguiente tabla muestra el ordenamiento ascendente versus el ordenamiento solicitado:

| *Orden ascendente por magnitud* | *Orden descendente por cantidad de bits en 1* |
|---|---|
| 0b0 | 0b11111111111 |
| 0b10101 | 0b10010010010011111 |
| 0b11111 | 0b1111111101 |
| 0b1111111101 | 0b01111000000000000000000000001111 |
| 0b11111111111 | 0b1010101010101 |
| 0b1010101010101 | 0b11111100000000000000000000000000 |
| 0b10010010010011111 | 0b11111 |
| 0b01111000000000000000000000001111 | 0b11100000000000000000000000000001 |
| 0b10000000000000000000000000000000 | 0b10101 |
| 0b10000000000000000000000000000001 | 0b10000000000000000000000000000001 |
| 0b11100000000000000000000000000001 | 0b10000000000000000000000000000000 |
| 0b11111100000000000000000000000000 | 0b0 |

Los números están en base 2 para que sea más fácil contar los bits en 1. Observe que *0b11111111111* aparece en 1er lugar porque tiene 11 bits en 1, más que todo el resto. Esta tarea se compilará con la opción *-std=c2x* para que acepte las futuras mejoras del lenguaje C, que incluyen el uso de constantes en binario en el formato *0b10110*. ¡Por fin!

## Instrucciones

Baje *t5.zip* de U-cursos y descomprímalo. Contiene el *Makefile* y los programas que necesita para hacer esta tarea. Compile y ejecute con:

*make sort-rv-nbits.run*

El programa no pasa el test de prueba porque ordena ascendentemente por magnitud. Reprograme en assembler la función *sort* en el archivo ***sort-rv-nbits.s***, de manera que ordene de la manera solicitada.

## Restricciones

Ud. solo puede modificar en *sort-rv-nbits.s* el código que compara los elementos consecutivos. Una vez hecha la comparación, la ejecución debe continuar en la etiqueta *.decision* más abajo. Este código está delimitado por un par de comentarios en el archivo. No modifique nada más. Sin esta restricción la tarea sería trivial y por lo tanto sin no la cumple será ***rechazada***.

## Ayuda

- El archivo *sort-c.c* es la versión en C de la función *sort*. Compile y ejecute esta versión con (no pasa el test de prueba):

  *make sort-c.run;*
- Programe primero una versión en C de lo pedido en el archivo *test-sort-c-inbits.c* para probar su algoritmo. Compile y ejecute con:

  *make sort-c-nbits.run*
- Puede obtener ideas de las instrucciones en assembler RiscV que debe usar generando el archivo *sort-c-nbits.s* con:

  *make sort-c-nbits**.s***
- Lance *ddd* para depurar su tarea con el comando:

  *make sort-rv-nbits.ddd* (o *make sort-c-nbits.ddd*)

  Seleccione el menú *View* → *Machine code window* para ver el assembler. Coloque breakpoints en lugares estratégicos con: *break .while_begin*. Lea la guía rápida para usar gdb en:

  *http://www.dcc.uchile.cl/~lmateu/CC4301*
- Conozca la dirección de cada instrucción Risc-V con:

  *make sort-rv-nbits.dump*

  Si una instrucción está en la dirección *1017c*, coloque un breakpoint en esa instrucción ingresando en el panel de comandos de ddd:

  *b *0x1017c*
- Descargue de novedades el *toolchain para Risc-V* y *qemu-riscv32* para poder compilar y ejecutar programas para Risc-V.
- En la clase auxiliar del viernes 22 de octubre se estudió la solución una tarea similar del curso de arquitectura de computadores.

## Entrega

Entregue por medio de U-cursos el archivo ***sort-rv-nbits.s*** con su solución. El comando *make sort-rv-nbits.run* debe felicitarlo, si no su tarea será rechazada. Se descontará medio punto por día de atraso (excluyendo sábados, domingos, festivos o recesos).