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 número de bits en 1* |
|---|---|
| 0b0 | 0b11111111111 |
| 0b10101 | 0b10010010010010011111 |
| 0b11111 | 0b1111111101 |
| 0b1111111101 | 0b01111000000000000000000000001111 |
| 0b11111111111 | 0b1010101010101 |
| 0b1010101010101 | 0b11111100000000000000000000000000 |
| 0b10010010010010011111 | 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 archivos que necesita para hacer esta tarea. Ejecute el comando *make* sin parámetros para recibir instrucciones sobre la ubicación de los archivos que debe modificar y cómo compilar, ejecutar y depurar. En particular lea los tips para la depuración y la solución de problemas.

## Restricciones

No hay restricciones para la programación de *sort-c-nbits.c*. Para la programación de *sort-rv-nbits.s* **Ud. solo puede modificar el código que compara los elementos consecutivos**. Está claramente delimitado en el archivo original. No modifique nada más. Sin esta restricción la tarea sería trivial. **Además para hacer la comparación *no puede* invocar otras funciones**. Una vez hecha la comparación, la ejecución debe continuar en la etiqueta *.decision* y el resultado de la comparación debe quedar en el registro *t1*. Si *t1>0*, los números en *p[0]* y *p[1]* están desordenados y por lo tanto se intercambiarán. Si *t1≤0* no se intercambiarán.

## Ayuda

- Modifique en *sort-c-nbits.c* la función *sort* de manera que se comparen los números por la cantidad de bits en 1. Haga un esfuerzo en llegar a la función más pequeña, porque así será menor la cantidad de líneas en assembler que deberá programar y depurar. Pruebe *sort-c-nbits.c* ejecutando el comando *make sort-c-nbits.run* (*make sort-c-nbits.ddd* para depurar)
- Una vez que pase exitosamente la prueba de *sort-c-nbits.c*, ejecute el comando *make sort-c-nbits.s* para compilar a assembler. Estudie en el archivo *sort-c-nbits.s* la traducción a assembler Risc-V de la función que compara los números. Use un código similar para comparar los número en la zona delimitada en *sort-rv-nbits.s* y así completar su tarea. Recuerde: sin invocar otras funciones. Pruebe su tarea con: *make sort-rv-nbits.run* (*make sort-rv-nbits.ddd* para depurar)
- Revise si el toolchain para compilar y ejecutar programas para Risc-V está instalado en */opt/riscv*. Si no encuentra ese directorio, descargue el archivo *riscv.tgz* (*riscv-arm.tgz* para Arm) de esta carpeta de google drive e instale el *toolchain* para RiscV con estos comandos:

```
cd ... directorio en donde descargó riscv.tgz ...
sudo bash
cat riscv.tgz | ( cd / ; tar zxvf - )
```

- En la clase auxiliar del viernes 9 de mayo se estudió la solución de una tarea similar de un semestre pasado.

## Entrega

Entregue por medio de U-cursos el archivo ***nbits.zip*** generado con el comando *make zip*. Este comando verifica que su tarea funcione correctamente y luego genera *nbits.zip* con los archivos *sort-c-nbits.c*, *sort-rv-nbits.s* y *resultados.txt* con la ejecución de la tarea. **Recuerde descargar de u-cursos lo que entregó, descargar nuevamente los archivos adjuntos** y vuelva a probar la tarea tal cual como la entregó. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificacion. Su tarea debe ordenar correctamente, si no será rechazada. Se descontará medio punto por día de atraso (excluyendo sábados, domingos, festivos o vacaciones).