Parte a.- Un grupo de unos en un entero *x* es una secuencia consecutiva de bits en uno en *x* delimitada a la izquierda por 0 o el inicio de *x*, y a la derecha por 0 o el final de *x*. El número de grupos de unos de 0b**111**00**1111**00001 es 3. Programe en el archivo *groups.c* la función *groups* que calcula el número de grupos de unos de *x*. El encabezado de la función es: *int groups(unsigned int x).*

Parte b.- La función *sort* está programada en C y en assembler Risc-V en los archivos *sort-c.c* y *sort-rv.s* respectivamente. 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-groups.s* es una copia de *sort-rv.s*. Modifique la función *sort* en *sort-rv-groups.s* de modo que se ordene el arreglo ascendentemente según el número de grupos de unos de cada elemento de *nums*. El siguiente ejemplo muestra el ordenamiento ascendente normal versus el ordenamiento solicitado.

Orden ascendente normal	*Orden solicitado*
0b0	0b0
0b**10**10**10**10**10**10**10**10**10**10**10**	0b**1111**0000**1111**000000000000
0b**11**00**11**0001**1**00**111111**0**11**	0b**11**00**11**0001**1**00**111111**0**11**
0b**1111**0000**1111**000000000000	0b**10**10**10**10**10**10**10**10**10**10**10**

Los números están en base 2 para que sea más fácil contar los grupos de unos. En la columna derecha, el número de grupos de unos es respectivamente 0, 2, 5 y 10. En negritas aparecen los grupos de unos. 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

Ud. solo puede modificar en *sort-rv-groups.s* 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. Está prohibido retornar de la función en su propio código. 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

- **Haga un esfuerzo en llegar a la función *groups* más pequeña, porque así será menor la cantidad de líneas en assembler que deberá programar**. Use los operadores de bits. Pruebe con *make groups.run*. Depure con *make groups.ddd*.
- Invoque el comando *make groups.s*. Estudie la función *groups* en assembler en el archivo *groups.s*. Ahora programe la versión en assembler del nuevo *sort* del archivo *sort-rv-groups.s*. Use instrucciones similares a las de *groups.s*, pero deberá usar etiquetas y registros distintos. Recuerde que no puede llamar a funciones. Pruebe con *make sort-rv-groups.run*. Depure con *make sort-rv-groups.ddd*.
- Revise si el toolchain para compilar y ejecutar programas para RiscV 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:

```
sudo bash
cd /
tar zxvf /...ruta.../riscv.tgz
```

- En la clase auxiliar del viernes 14 de octubre se estudió la solución de una tarea similar de un semestre pasado.

Entrega

Entregue por medio de U-cursos el archivo ***sort-groups.zip*** generado con el comando *make zip*. Este comando verifica que su tarea funcione correctamente y luego genera *sort-groups.zip* con los archivos *groups.c*, *sort-rv-groups**.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).