Laboratorio de Electrónica Digital II
Práctica No. 4: Ordenamiento de Números Enteros con Signo Empleando Lenguaje Ensamblador para ARM

Profesores
Luis Fernando Castaño L. (luis.castanol@udea.edu.co)
Luis Germán García M. (german.garcia@udea.edu.co)

Septiembre 26, 2023

Fecha de entrega:	Del 3 al 6 de Octubre de 2023
Medio de entrega:	https://virtualingenieriaudea.co/
Sustentación:	Horario de Laboratorio
Valor Práctica:	6% del curso

1 Introducción

En esta práctica de laboratorio, el grupo de estudiantes hará uso del lenguaje ensamblador para procesadores ARM de 32-bits, con el propósito de desarrollar un programa que sea capaz de ordenar números enteros con signo. Los datos de entrada al programa estarán ubicados en la memoria en posiciones específicas. Los números ordenados deberán ser escritos a la memoria sin sobreescribir los datos de entrada. El funcionamiento del programa se comprobará mediante el uso del simulador CPUlator.

2 Objetivo de la Práctica

Desarrollar un programa para el procesador ARM, empleando lenguaje ensamblador, que ordene un conjunto de números enteros con signo ubicados en la memoria de acuerdo con ciertos criterios y realizar la correcta simulación para comprobar su funcionamiento.

3 Procedimiento

Para el correcto diseño e implementación de la práctica, es necesario leer completamente esta guía. Se sugiere seguir el procedimiento indicado a continuación:

a. Comprender la arquitectura del conjunto de instrucciones del procesador ARM.

b. Simular los programas de ejemplo disponibles en la página del curso, empleando el simulador del procesador ARM sugerido en esta guía, y comprender su funcionamiento.

c. Elaborar los Pseudocódigos o diagramas de flujo para el programa en ensamblador a desarrollar, el cual debe llevar a cabo la funcionalidad que se indicará más adelante.

d. Desarrollar el programa solicitado empleando lenguaje ensamblador para el procesador ARM basado en los Pseudocódigos o diagramas de flujo realizados previamente.

e. Simular el programa desarrollado para el procesador ARM y verificar el correcto funcionamiento del mismo. Llevar a cabo las correcciones pertinentes, si fuese el caso.

f. Enviar el código fuente de su programa en lenguaje ensamblador para el procesador ARM, junto con un breve reporte, antes de la fecha límite.

g. Sustentar el diseño en el horario de laboratorio correspondiente.

4 Especificaciones

El programa a desarrollar debe estar en capacidad de realizar cualquiera de los ordenamientos indicados en Tab. 1 sobre un conjunto N de valores numéricos.

El valor de N, así como los N valores numéricos a ordenar, estarán ubicados en la memoria. El tipo de ordenamiento a realizar cuando el programa se ejecute en el simulador será determinada a partir de un valor ubicado también en la memoria, denominado OP. Los rangos de valores para los datos que serán entregados al programa son los siguientes:

(i) N: valor entero en el rango $2 \leq N \leq 200$.
(ii) Datos: valores enteros en el rango $-2^{31}(-2147483648) \leq VAL \leq 2^{31} - 1(+2147483647)$.
(iii) OP: valor entero en el rango $1 \leq OP \leq 2$.

Grupos de trabajo	Tipos de ordenamiento
1, 4, 7	1) Ordenamiento ascendente 2) Ordenamiento descendente de números múltiplos de 2
2, 5, 8	1) Ordenamiento descendente 2) Ordenamiento ascendente de números múltiplos de 4
3, 6, 9	1) Ordenamiento descendente de números múltiplos de 4 2) Ordenamiento ascendente de números múltiplos de 2

Tab. 1: Tipos de ordenamiento a realizar

Todos los valores antes mencionados se cargarán, junto con las instrucciones del programa, en el momento en que inicie la simulación. Durante la ejecución, el programa almacenará en memoria los valores ordenados, siempre que estos cumplan con el criterio dado (e.j., múltiplo de 2). En la siguiente sección se dará a conocer la forma de especificar tanto los valores de entrada, como la posición de memoria donde se ubicarán los valores ordenados.

Finalmente, tenga en cuenta que un número entero A se puede considerar múltiplo de un número entero B sí existe un número entero C, distinto de cero, tal que satisfaga la ecuación: A = B * C. Algunas definiciones indican que C debería ser un número entero natural (número entero no negativo); sin embargo, para esta práctica, consideraremos a C solamente un número entero.

5 Diseño y Simulación

El código a desarrollar se basará en la plantilla que se indica en la Fig. 1. Para los datos que se leen desde la memoria tenga en cuenta que:

(i) N se encuentra ubicado en la dirección dada por la etiqueta **N**.

(ii) El primer valor de los datos se encuentra ubicado en la dirección dada por la etiqueta **Data**, el segundo en **Data + 4**, el tercero en **Data + 8** y así sucesivamente.

(iii) OP se encuentra ubicado en la dirección dada por la etiqueta **OP**.

Por otro lado los datos, resultado del ordenamiento, se escribirán en la memoria así: el primer dato de la lista de valores ordenados según el tipo de ordenamiento seleccionado se almacenará en la dirección de memoria dada por la etiqueta **SortedData**, el segundo en **SortedData + 4**, el tercero en **SortedData + 8** y así sucesivamente.

Adicionalmente, tenga en cuenta que la arquitectura ARM que se está estudiando en el curso (ARMv4), así como la versión disponible en el simulador sugerido (ARMv7), no dispone de instrucciones de división (UDIV, SDIV). El uso de subrutinas para tener un código modular es opcional en esta práctica.

```
.global _start
    .equ    MAXN,   200
    .text
_start:
    /* Inicio de Programa:
     * Inicialización de registros y lectura de valores requeridos desde la memoria
     */

    /* Cuerpo del programa:
     * Código principal (*opcional* y subrutinas y llamados a subrutinas desde el código
     * principal) para realizar la operación dada por OP sobre los datos en memoria
    */

    /* Fin de Programa:
     * Bucle infinito para evitar la búsqueda de nuevas instrucciones
     */
finish:
    b finish

    .data
N:          .dc.l   12
Data:       .dc.l   1,15,-79,35,16,-564,8542,-89542,12021,54215,12,-35
OP:         .dc.l   1

SortedData: .ds.l   MAXN
```

Fig. 1: Plantilla para el código a desarrollar

El grupo de trabajo deberá guardar su programa en un archivo de nombre **program_xy.asm**, donde **x** es la letra inicial del primer apellido del integrante 1 y **y** es la letra inicial del primer apellido del integrante 2. Para la simulación, utilice el simulador dado en las referencias (al final de este documento) y compruebe el funcionamiento del programa con sus propios valores de prueba.

6 Entrega

El grupo de trabajo deberá escribir un breve reporte en formato IEEE que contenga los siguientes elementos (ver guía para reportes en la página del curso):

a. **Abstract**: resumen del diseño e implementación de la práctica.

b. **Pseudocódigo**: Pseudocódigo o diagrama de flujo del programa realizado en lenguaje ensamblador.

c. **Conclusiones**: dos o tres conclusiones sobre el trabajo realizado por el grupo de trabajo. Indicar el tiempo que les tomó realizar la práctica en las conclusiones.

Crear un archivo comprimido que incluya el reporte y los archivos importantes de su código en ensamblador como se describe a continuación:

a. **Reporte**: archivo con extensión .pdf

b. **Archivos esenciales**: archivo con extensión .asm

El nombre del archivo comprimido deberá tener el siguiente formato:
p4_primerapellidointegrante1_primerapellidointegrante2_horariolaboratorio.zip.
Ejemplo: si el primer apellido de ambos integrantes es **Castano** y **Garcia**, respectivamente, y el laboratorio es el Martes 9-12, entonces el archivo debe ser nombrado: *p4_castano_garcia_m9-12.zip*.

7 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento (50%), sustentación (40%) y reporte (10%). La nota del funcionamiento se asigna por igual a todos los integrantes del grupo de trabajo (máximo dos personas por equipo), mientras que la nota de sustentación es individual. En caso un estudiante obtenga una nota inferior a 3.0 en la sustentación, la nota final de la práctica para el estudiante en mención será la que obtuvo en la sustentación, es decir, no se tendrá en cuenta el funcionamiento en el cálculo.

Cada grupo de trabajo deberá sustentar la práctica en un tiempo de 15 minutos, 8 minutos para revisar la implementación y 7 minutos para preguntas. Es importante tener abierto el CPUlator con el programa listo para cuando el profesor llegue a su puesto de trabajo. No habrá tiempo para hacer correcciones de último momento.

8 Referencias

a. Harris, Sarah and Harris, David. Digital Design and Computer Architecture: ARM Edition. Morgan Kaufmann Publishers Inc. 2015. ISBN: 0128000562

b. CPUlator Computer System Simulator
`https://cpulator.01xz.net/?sys=arm`