Laboratorio de Electrónica Digital II
Práctica 3: Generación de Serie Fibonacci empleando Lenguaje Ensamblador de ARM

Profesores
María Katherine Plazas O. (maria.plazas@udea.edu.co)
Luis Germán García M. (german.garcia@udea.edu.co)

Abril 2, 2024

Fecha de entrega:	del 9 al 11 de abril de 2024
Medio de entrega:	https://virtualingenieriaudea.co/
Sustentación:	Horario de laboratorio
Valor Práctica:	5% 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 genere y manipule un conjunto de datos dados unos criterios. Los datos de entrada al programa estarán ubicados en la memoria en posiciones específicas. Los datos a generar deberán ser escritos en 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 genere los valores de las posiciones especificadas de la serie Fibonacci y los escriba en la memoria en determinado orden.

3 Procedimiento

Para el correcto diseño e implementación del programa en ensamblador, se sugiere realizar el siguiente procedimiento:

a. Leer esta guía varias veces para comprender el programa a desarrollar, identificando así los requerimientos que deberán cumplirse el día de la sustentación del trabajo.

b. Comprender la arquitectura del conjunto de instrucciones del procesador ARM dada en clase y en el texto guía *Digital Design And Computer Architecture - ARMEdition.*

c. 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.

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

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

f. 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.

g. Enviar el código fuente del programa en lenguaje ensamblador para el procesador ARM, junto con los Pseudocódigos o diagramas de flujo, antes de la fecha límite.

h. Sustentar el desarrollo en el horario de laboratorio correspondiente.

4 Especificaciones

El programa deberá generar una secuencia de N valores pertenecientes a la serie Fibonacci en función de las posiciones especificadas por el usuario. Estos valores serán escritos en la memoria en orden ascendente o descendente, según la selección del usuario. El usuario deberá especificar la cantidad de valores a generar de la secuencia (N), las posiciones dentro de la serie a generar (POS) y el orden en el que se escribirán los valores en la memoria (ORDER). La serie Fibonacci es una secuencia en la que cada número es la suma de los dos números anteriores (0, 1, 1, 2, 3, 5, 8, 13, 21, y así sucesivamente). En caso se solicite generar las posiciones 2, 9 y 5, el programa deberá generar y escribir en la memoria los valores 1, 3, 21 respectivamente, si el orden seleccionado es

ascendente. En caso que el orden seleccionado sea descendente, los valores a escribir en la memoria serán 21, 3 y 1.

El valor de N, así como las N posiciones de la serie y el orden de escritura, tienen los siguientes rangos de valores:

(i) N: valor entero en el rango $5 \leq N \leq 30$.
(ii) POS: lista de valores enteros en el rango $1 \leq VAL \leq 40$.
(iii) ORDER: valor 0 para indicar ascendente o valor 1 para indicar descendente.

Todos los valores antes mencionados se cargarán, junto con las instrucciones del programa, en el momento en que inicie la simulación. Por lo tanto, no es necesario solicitar dichos valores al usuario (de alguna manera, los datos del usuario ya fueron llevados a la memoria). Durante la ejecución, el programa almacenará en memoria los valores ordenados a partir de la posición de memoria dada más adelante. Tenga en cuenta que si alguno de los valores de entrada no se encuentra dentro del rango dado, el programa deberá abortar la ejecución.

El código a desarrollar se basará en la plantilla que se indica en la Fig. 1 (Los valores presentes en la plantilla son solo de ejemplo). 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 la lista de posiciones se encuentra ubicada en la dirección dada por la etiqueta **POS**, el segundo en **POS + 4**, el tercero en **POS + 8** y así sucesivamente.
(iii) ORDER se encuentra ubicado en la dirección dada por la etiqueta **ORDER**.

Por otro lado, los valores de la secuencia generados y ordenados se escribirán en la memoria así: el primer dato de la lista de valores se almacenará en la dirección de memoria dada por la etiqueta **SortedValues**, el segundo en **SortedValues + 4**, el tercero en **SortedValues + 8** y así sucesivamente. En caso el programa haya abortado la ejecución debido a un error en el rango de los datos de entrada, se deberá escribir el valor 0xA5A5A5A5, en la posición de memoria dada por la etiqueta **SortedValues**.

El grupo de trabajo deberá guardar su programa en un archivo de nombre **program_xy.s**, 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.

5 Entrega

El grupo de trabajo deberá crear un archivo comprimido que incluya:

a. **Algoritmos**: archivo con extensión .pdf donde se incluye los Pseudocódigos o diagramas de flujo del programa realizado. La organización será tenida en cuenta en la calificación.

```
.global _start
    .equ    MAXN,   30
    .text
_start:
    /* Inicio de Programa:
     * Inicialización de registros y lectura de valores requeridos desde la memoria
     */

    /* Cuerpo del programa:
     * Código principal para generar los valores solicitados de la serie Fibonacci
     */

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

    .data
    /* Constantes y variables propias:
     * Utilice esta zona para declarar sus constantes y variables requeridas
     */

    /* Constantes y variables dadas por el profesor:
     * Esta zona contiene la definición de N, POS, ORDER y SortedValues
     */
N:              .dc.l   5
POS:            .dc.l   2, 9, 35, 4, 16
ORDER:          .dc.l   1

SortedValues:   .ds.l   MAXN
```

Fig. 1: Plantilla para el código a desarrollar (Valores de ejemplo)

b. **Programa en Ensamblador**: archivo con extensión .s.

El nombre del archivo comprimido deberá tener el siguiente formato:
p3_primerapellidointegrante1_primerapellidointegrante2_horariolaboratorio.zip.
Ejemplo: si el primer apellido de ambos integrantes es **Plazas** y **Garcia**, respectivamente, y el laboratorio es el Martes 9-12, entonces el archivo debe ser nombrado: *p3_plazas_garcia_m9-12.zip*.

6 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento (40%), sustentación (40%) y algoritmos / código fuente (20%). 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. La sustentación podrá realizarse de tres maneras posibles:

a. Solicitud de cambios al código por parte del profesor de laboratorio.

b. Un par de preguntas orales que pongan a prueba los conocimientos del estudiante sobre el desarrollo de la práctica.

c. Una presentación corta a todo el grupo de laboratorio.

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. Finalmente, en caso no se realice entrega de los Pseudocódigos o diagramas de flujo, la nota asignada a la práctica de laboratorio será de 0.0 así se haya sustentado.

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 programa en CPUlator cuando el profesor llegue a su puesto de trabajo. No habrá tiempo para hacer correcciones de último momento.

7 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`