Laboratorio de Electrónica Digital II
Práctica 4: Generación de Valores de 64-bits de la 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 9, 2024

Fecha de entrega:	del 23 al 25 de abril de 2024
Medio de entrega:	https://virtualingenieriaudea.co/
Sustentación:	Horario de laboratorio
Valor Práctica:	10% (15%) 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, de manera similar a como se realizó en la práctica anterior. 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, sabiendo que los valores de la serie a generar serán de 64-bits.

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. Extender los Pseudocódigos o diagramas de flujo que se elaboraron para la práctica anterior, con el propósito de cumplir los nuevos requerimientos dados en esta guía.

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

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

4 Especificaciones

El programa a desarrollar deberá generar una secuencia de N valores pertenecientes a la serie Fibonacci en función de las posiciones especificadas por el usuario, de la misma forma que se generaron en la práctica anterior, teniendo en cuenta las siguientes definiciones:

(i) Número de valores de la secuencia a generar: N, en el rango: $10 \leq N \leq 50$.
(ii) Lista de N posiciones a generar: POS, cada valor de la lista en el rango $1 \leq VAL \leq 90$.
(iii) ORDER: valor 0 para indicar ascendente o valor 1 para indicar descendente.
(iv) Los N valores a generar son de 64-bits sin signo, por lo que deberán ser almacenados en dos posiciones diferentes y contiguas de memoria (la parte menos significativa en la menor de las direcciones y la parte más significativa en la mayor).

Adicionalmente, se deberán tener en cuenta los siguientes requerimientos para la escritura del código ensamblador:

(i) Deberá existir una función **Fibonacci** encargada de generar los primeros M elementos de la serie. Será llamada mediante la instrucción en ensamblador **BL** (Branch and Link). Recibirá dos parámetros en los registros **R0** y **R1**: el valor **M** y la dirección a partir de la cual se almacenará la lista de valores M de la serie de Fibonacci. No retornará valores.

(ii) Deberá existir una función **Sort** encargada de ordenar, ascendente o descendentemente, L valores en memoria que se encuentran ubicados a partir de una posición dada. Será llamada mediante la instrucción en ensamblador **BL**. Recibirá tres parámetros en los registros **R0**, **R1** y **R2**: el valor **L**, la dirección donde se encuentra el primer valor a ordenar y el tipo de ordenamiento (ORDER). No retornará valores. Tenga en cuenta que los datos a organizar estarán consecutivos en memoria.

(iii) Deberá existir una función **Add64** encargada de realizar la operación R = A + B, donde A, B y R son valores de 64-bits sin signo. Será llamada mediante la instrucción en ensamblador **BL**. Recibirá dos parámetros en los registros R1:R0 y R3:R2: el valor de **A** en los registros **R1(MSB):R0(LSB)** y el valor de **B** en los registros **R3(MSB):R2(LSB)**. Retornará un valor de 64-bits correspondiente al valor de **R** en los registros **R1(MSB):R0(LSB)**. Esta función deberá ser llamada desde la función **Fibonacci** para calcular el siguiente valor de la serie.

(iv) Para todas las funciones, se deberá emplear la metodología de *Stack* para mantener el valor de los registros que deben preservarse entre llamados.

La forma como se manipularán los datos de entrada y salida es similar a como se hizo en la práctica anterior, incluída la validación de valores de entrada. El código a desarrollar se basará en la plantilla que se indica en la Fig. 1 (muy similar a la de la práctica anterior).

El grupo de trabajo deberá guardar su programa en un archivo de nombre **program_v2_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.

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

El nombre del archivo comprimido deberá tener el siguiente formato:
p4_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: *p4_plazas_garcia_m9-12.zip.*

```
.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, 85, 49, 12, 38
ORDER:          .dc.l   1

SortedValues:   .ds.l   MAXN*2
```

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

6 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento con cumplimiento de requerimientos en código (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.

Tenga en cuenta que esta práctica tendrá un valor de 15% si el grupo de trabajo no tiene lista la práctica anterior en el momento pactado para la entrega, pero puede mostrar un avance significativo en la misma (por lo menos un 50% de desarrollo); en caso contrario, ambas prácticas serán evaluadas por aparte, teniendo esta práctica un valor de 10%.

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`