Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

Objetivo

El objetivo de este taller de programación es afianzar los conocimientos en el lenguaje ensamblador bajo la arquitectura ARM, así como aprender algunas instrucciones básicas del lenguaje y la escritura de programas simples con aplicaciones prácticas.

Material y Equipo Requerido

- Computador Personal
- Emulador VisUAL 1.0 / 2.0 (o cualquier otro compatible con microarquitectura ARM v7 o superior)

Parte 1: Comprobación

La parte de comprobación requiere la implementación de los ejemplos mostrados, para luego **analizar** su funcionamiento mediante una **descripción** en su informe de laboratorio. La descripción debe mostrar lo sucedido en los registros, memoria RAM y símbolos de acuerdo a la ejecución de las instrucciones. Si desea puede añadir un diagrama de flujo y otros elementos cuando lo considere necesario. Siempre tenga presente: ¿Es coherente el resultado mostrado? ¿se esperaba ese comportamiento?. **Deje un registro escrito explícito y detallado por cada ejercicio planteado en esta sección. Realice comentarios sobre el código que se comprueba para conocer su funcionamiento.**

- **Introducción**

  Las instrucciones de ensamblador en ARM pueden dividirse en tres conjuntos diferentes: **Las instrucciones de procesamiento de datos** manipulan los datos dentro de los registros. Estos pueden ser aritméticos (suma, resta, multiplicación, división), lógica (operaciones booleanas), relacional (comparación de dos valores) o instrucciones de movimiento. **Las instrucciones de acceso a la memoria** mueven los datos hacia y desde la memoria principal. Ya que todas las demás operaciones sólo funcionan con valores constantes inmediatos (codificados en la propia instrucción) o con valores de los registros, las instrucciones de carga/almacenamiento son necesarias para hacer frente a todos los conjuntos de datos. Por último, **las instrucciones de ramificación (branching)** cambian el flujo de control, modificando el valor del Program Counter (R15). Se necesitan para implementar declaraciones condicionales, bucles y llamados a subrutinas externas.

  Actividad: Realice una consulta sobre el subconjunto de instrucciones THUMB de ARM. ¿En qué consiste el subconjunto de Instrucciones THUMB? ¿cómo se diferencias del set de instrucciones completo de ARM? ¿como es el manejo de registros y memoria cuando el procesador está en modo THUMB? ¿En qué versiones y microarquitecturas de ARM se implementa THUMB?

- **Instrucciones de procesamiento de datos**

  Las instrucciones de procesamiento de datos ARM permiten al programador realizar operaciones aritméticas y lógicas sobre los valores de los datos en los registros. El resto de las instrucciones sólo mueven los datos y controlan la secuencia de ejecución del programa, por lo que las instrucciones de procesamiento de datos son las únicas que modifican los valores de los datos. La mayoría de las instrucciones de procesamiento de datos pueden procesar uno de sus operandos utilizando el desplazador. Si se utiliza el sufijo 'S' en una instrucción de procesamiento de datos, se actualizan las banderas en la ALU. Las operaciones de movimiento y lógica actualizan la bandera de acarreo C, la bandera negación N y la bandera cero Z. La bandera de acarreo (C) se establece como resultado del corrimiento cuando se desplaza el último bit. El indicador negativo (N) se fija en el bit 31 del resultado. El indicador de cero (Z) se fija si el resultado es cero.

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

Instrucción de movimiento (MOV)

Dentro del procesador, los datos residen en los registros. MOV es la instrucción básica que se mueve los datos constantes en el registro o mueve esos datos de un registro a otro. En el conjunto de instrucciones THUMB, la instrucción MOVT mueve el valor inmediato de 16 bits a la media palabra superior (bits 16 a 31) y la media palabra inferior permanece inalterada. Ejecute las siguientes instrucciones en su emulador y observe los resultados (recuerde que el resultado y análisis de todas las actividades deben dejarse registradas de manera detallada en su informe):

```
        MOV     R0, #0xFF
        MOV     R1, #-0x281
        MOV     R4, #0xD364

        MOVT    R4, #0xFFB1

        MOV     R5, R4
```

- **Instrucciones de Desplazamiento y Rotación**

Las instrucciones de Desplazamiento y Rotación se utilizan para cambiar la posición de los valores de los bits en un registro. En este manual se tratan diferentes variantes de estas instrucciones.

Desplazamiento lógico a la derecha (LSR)

Esta instrucción es similar a la división sin signo por 2^n donde n especifica el número de desplazamientos. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

```
        MOV     R0, #0x06
        MOV     R3, #0x200
        MOV     R5, #0x9B1D
STOP
        LSR     R4, R3, #6
        LSR     R2, R3, R0

        LSR     R6, R5, #5

        LSRS    R7, R5, #5

        B       STOP
        END
```

Desplazamiento aritmético a la derecha (ASR)

Esta instrucción es similar a la división con signo por 2^n donde n especifica el número de desplazamientos. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

```
        MOV     R0, #0x07
        MOV     R3, #-0x669
        MOV     R5, #0xD634
        MOVT    R5, #0xFFB1

STOP
        ASR     R4, R3, #5
        ASR     R2, R3, R0

        ASR     R6, R5, #5

        ASRS    R7, R5, #5

        B       STOP
        END
```

Desplazamiento lógico a la izquierda (LSL)

La instrucción del cambio lógico a la izquierda funciona bien para los números con y sin signo. Esta instrucción es sinónimo de multiplicar por 2^n donde n especifica el número de desplazamientos. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

```
        MOV     R7, #0xC
        MOV     R0, #-0x281
        MOV     R2, #0x25C3

        MOV     R4, #0x593C

        MOVT    R4, #0xA377


STOP
        LSL     R1, R0, #3
        LSL     R3, R2, R7

        LSL     R5, R4, #10

        LSLS    R6, R4, #10

        B       STOP

        END
```

Rotación a la derecha (ROR)

Las instrucciones de rotación no desechan ningún bit del registro. En su lugar, los valores de los bits se eliminan de un extremo del registro y se insertan en el otro extremo. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

```
        MOV     R0, #0x9
        MOV     R1, #0x25C3

        MOV     R3, #0x73A2

        MOVT    R3, #0x5D81


STOP
        ROR     R2, R1, #24
        ROR     R7, R3, R0

        ROR     R4, R3, #10

        RORS    R5, R3, #10

        B       STOP
        END
```

Rotación a la derecha extendida (RRX)

RRX es una operación ROR con una diferencia crucial. Rota el número a la derecha en un lugar, pero el bit original 31 se llena con el valor de la bandera de acarreo y el bit original 0 se mueve a la bandera de acarreo. Esto permite una rotación de 33 bits usando tanto el registro como la bandera de acarreo. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

```
        MOV     R3, #0xD129

        MOVT    R3, #0xF29A


STOP
        RRX     R4, R3
        RRXS    R6, R3

        B       STOP
        END
```

- **Instrucciones de Acceso a Memoria**

ARM es una arquitectura de carga/almacenamiento diseñada mediante RTL (Register Transfer Level). No soporta operaciones de procesamiento de datos de **memoria a memoria**. Todas las operaciones de procesamiento de datos se ejecutan en los registros, es decir, los valores de datos necesarios para una operación deben ser trasladados a los registros antes de utilizarlos. Es necesario tener instrucciones que interactúen con la memoria para mover los datos de la memoria a los registros y viceversa. ARM tiene tres conjuntos de instrucciones que interactúan con la memoria principal.

1. Transferencia de datos de un solo registro (LDR/STR)
2. Transferencia de datos en bloque (LDM/STM)
3. Intercambio único de datos (SWP)

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

En esta guia de laboratorio, discutiremos las instrucciones de transferencia de datos de un solo registro y los diferentes modos de direccionamiento que se pueden utilizar para mover el contenido de la memoria a los registros y viceversa. y viceversa.

Transferencia de datos de registro único

Las instrucciones básicas de transferencia de datos soportadas por el ensamblador ARM pueden cargar y almacenar diferentes tamaño de los datos desde y hacia la memoria respectivamente. **Nota**: Se muestra el código completo con las declarativas THUMB, AREA y otras que están presentes en el microprocesador real. Si usa VisUAL solo es necesario dejar el código de las instrucciones. Probablemente necesite remapear las instrucciones según el mapa de memoria de VisUAL, el cual puede encontrar en https://salmanarif.bitbucket.io/visual/memory_map.html. Para la declaración de variables use DCD (modo word) o DCB (modo byte) en vez de SPACE. Por ejemplo: `data DCD 2` (en vez de `data SPACE 2`). Ejecute las siguientes instrucciones en su emulador y observe los resultados (haga especial énfasis en el pointer information y la visualización de memoria al ejecutar la instrucción **STR**):

```
        THUMB           ; Marks the THUMB mode of operation

;****** Data Variables are declared in DATA AREA ******;

        AREA    MyData, DATA, READWRITE

X       SPACE   2       ; 2 bytes for variable X are reserved
                        ; in memory at 0x20000000
data    SPACE   2       ; 2 bytes for variable data is declared in memory

;***** The user code (program) is placed in CODE AREA *****;

        AREA    |.text|, CODE, READONLY, ALIGN=2

        ENTRY           ; ENTRY marks the starting point
                        ; of the code execution
        EXPORT __main

__main
; ***** User code starts from the next line *****;

      MOV R0, #0x2D
      MOV R1, #0x40

      LDR R2, =data     ; Load the address of data variable in R2
      ADD R0, R0, R1
      STR R0, [R2]      ; Store the contents of R0 at address loaded in R2
                        ; Observe memory window at address specified in R2
STOP
      B   STOP          ; Infinite loop to STOP
      ALIGN
      END               ; End of the program, matched with ENTRY keyword
```

La instrucción LDR también puede ser usada para cargar cualquier constante en los registros. Cualquier constante numérica de 32 bits puede ser construida en una sola instrucción. Se utiliza para generar

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

constantes que están fuera del alcance de las instrucciones MOV y MVN. Ejecute las siguientes instrucciones en su emulador y observe los resultados (haga especial énfasis en el pointer information y la visualización de memoria al ejecutar la instrucción **STR**):

```
        THUMB           ; Marks the THUMB mode of operation
;****** Data Variables are declared in DATA AREA ******
        AREA    MyData, DATA, READWRITE
Result  SPACE 4         ; 4 bytes reserved for variable Result
;***** The user code (program) is placed in CODE AREA *****;
        AREA    |.text|, CODE, READONLY, ALIGN=2
        ENTRY           ; ENTRY marks the starting point of
                        ; the code execution
        EXPORT __main
__main
; ***** User code starts from the next line *****;
        LDR  R1, =0x458C3   ; Loads the value 0x458C3 in R1
                            ; which is out of range of MOV instruction
        LDR  R2, =0xA8261
        LDR  R4, =Result    ; Load the address of variable
                            ; result in R4
        MOV  R0, #0
        MUL  R3, R1, R2
        LSRS R3, #9         ; Logical shift right with flags update
        STR  R3, [R4]       ; Store the contents of R3 to memory
                            ; address already loaded in R4
STOP
        B   STOP
        END
```

Estas instrucciones proporcionan la forma más flexible de transferir datos individuales entre un registro ARM y la memoria. El elemento de datos puede ser un byte, una media palabra de 16 bits, una palabra de 32 bits o una palabra doble de 64 bits. Debe añadirse un modificador opcional para acceder al tipo de datos apropiado. La siguiente tabla muestra los tipos de datos disponibles y sus rangos.

type	Data Type	Range
	32-bit word	0 to $2^{32}-1$ or -2^{31} to $2^{31}-1$
B	unsigned Byte	0 to $2^{8}-1$
SB	Signed Byte	-2^{7} to $2^{7}-1$
H	unsigned Halfword	0 to $2^{16}-1$
SH	Signed Halfword	-2^{15} to $2^{15}-1$
D	Double word	0 to $2^{64}-1$ (two registers used)

En el siguiente ejemplo, un dato de tipo Byte es usado. Ejecute las siguientes instrucciones en su emulador y observe los resultados:

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

```
        THUMB           ; Marks the THUMB mode of operation

;****** Data Variables are declared in DATA AREA ******;

        AREA    MyData, DATA, READWRITE

High    DCB     0       ; 1 byte reserved for variable High
Low     DCB     0       ; 1 byte reserved for variable Low
Result  DCB     0       ; 1 byte reserved for variable Result
Result1 DCD     0       ; 4 bytes reserved for variable Result1

;***** The user code (program) is placed in CODE AREA *****;

        AREA    |.text|, CODE, READONLY, ALIGN=2

        ENTRY           ; ENTRY marks the starting point of
                        ; the code execution
        EXPORT __main
__main
; ***** User code starts from the next line *****;

      LDR  R0, =High      ; R0 =
      LDR  R1, =Low       ; R1 =
      LDR  R2, =Result    ; R2 =
      LDR  R5, =Result1   ; R5 =

      MOV  R3, #0x5
      MOV  R4, #0x8

      LSL  R3, #4         ; Shift the lower nibble to upper nibble of
                          ; the least significant byte

      STRB R3, [R0]       ; Observe memory window at the address
                          ; stored in R0

      STRB R4, [R1]       ; Observe memory window at the address
                          ; stored in R1

      ORR  R3, R3, R4     ; Set lower four bits of R3 equal to
                          ; lower four bits of R4

      STRB R3, [R2]       ; Observe memory window at the address
                          ; stored in R2

      STR  R3, [R5]       ; Takes 32 bits to store the result

      ALIGN
STOP
        B    STOP
      END
```

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

Modos de direccionamiento de Memoria

El conjunto de instrucciones de ARM proporciona diferentes modos de direccionamiento de la memoria. Estos modos incorporan uno de los métodos de indexación: preindexar con escritura, preindexar y postindexar. Consulte las diapositivas de clase para recapitular acerca de los modos de direccionamiento y las instrucciones (ISA). Ejecute las siguientes instrucciones en su emulador y observe los resultados

```
        THUMB

;****** Data Variables are declared in DATA AREA ******;

        AREA    MyData, DATA, READWRITE

data    SPACE  4      ; 4 bytes reserved for variable data

        AREA    |.text|, CODE, READONLY, ALIGN=2

        ENTRY          ; ENTRY marks the starting point of
                       ; the code execution
        EXPORT __main

__main
; ***** User code starts from the next line *****;

        MOV R0, #0x4A8
        MOV R1, #0x761
        MOV R4, #0x8D
        MOV R3, #0x0C

        SUB R1, R1, R0

; ********** Offset Addressing **********

        LDR R2, =data       ; Load the address of data in R2

        STR R1, [R2]        ; Store the 32-bit contents of R1 register
                            ; at the address loaded in R2. The contents
                            ; of R2  does not change.

        STR R1, [R2, #4]    ; A word in register R1 is stored at
                            ; memory address calculated by adding the
                            ; offset to R2. Contents of R2 does not change.

        STR R1, [R2, R3]    ; Store the contents of R1 at memory address
                            ; calculated by adding value in the base
                            ; register R2 to the value in the register R3.
                            ; Both R2 and R3 remain unchanged.
```

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

```
        STR R1, [R2, R3, LSL #2]
        ; Store the 32-bit value in R1 to the memory address calculated by
        ; adding the address in base register R2 to the value obtained
        ; by shifting the contents of R3 towards left by 2 bits. R2
        ; and R3 remain unchanged.

; ********** Post-index Addressing **********

        LDR R2, =data       ; Load the address of data in R2

        STR R1, [R2], #12
        ; Contents of R1 will be stored in memory at the address loaded in
        ; R2. R1 is updated with new address after adding the constant,
        ; specified, to it.

        STR R4, [R2], #24

; ********** Pre-index Addressing **********

        LDR R2, =data       ; Load the address of data in R2

        STR R1, [R2, #32]!
        ; 32-bit word in R1 will be stored in memory at an address
        ; calculated by adding 32 to the address loaded in R2.
        ; The contents of R2 also get updated with this new address

STOP
        B   STOP
        END
```

Parte 2: Ejercicios

1. Escriba un programa en ensamblador ARM para mover el byte superior del registro R2 y ponerlo en la parte inferior del registro R3.
2. Asuma R1 = 0x12AD. Aplique LSR en este valor y guarde el resultado en R2. Considere que el desplazamiento aumenta como 1,3,12.
3. Asuma R3 = 0x253B11. Aplique ASR en este valor y guarde el resultado en R4. Considere que el desplazamiento aumenta como 1,5,15.
4. El registro R3 contiene el valor negativo decimal -456. Aplique las instrucciones LSR y ASR y describa el resultado. ¿Qué instrucción da el resultado correcto y por qué?. Escriba su resultado en ambos casos y explique.
5. El registro R1 tiene el valor signado -123 y el registro R2 tiene el valor no signado 123. Aplique la operación LSL en ambos registros y observe el resultado.
6. ¿Cuál es el resultado neto si un patrón de 32 bits es desplazado 2 posiciones lógicamente a la izquierda y luego desplazado 2 posiciones lógicamente a la derecha? Compruébelo con código.
7. Escriba un programa en ensamblador ARM para realizar la función de Valor absoluto. El registro R0 contendrá el valor inicial y R1 contendrá el valor absoluto.
8. Escriba un programa en ensamblador ARM para realizar el ordenamiento ascendente y descendente (seleccionando la función a través de un valor cargado a registro) de 50 números cargados en memoria RAM. El algoritmo de ordenamiento puede ser simple, bubble sort, merge sort, heapsort, o quicksort.
9. Escriba un programa en ensamblador ARM que genere todas las combinaciones posibles de una lista dada de números. Escriba los resultados en RAM.
10. Escriba un programa en ensamblador ARM para buscar un elemento en un árbol de búsqueda binario. Usted puede definir los parámetros que sean más apropiados para su solución.

Laboratorio 2 - Microprocesadores.

Tema: Fundamentos de Programación en ASM para ARM

Parte 3: Aplicación

Escriba un programa en ensamblador ARM para realizar un cifrado de información ROT13. Puede usar strings o codificar la información únicamente a modo numérico. Brinde un código que permita la resolución del mismo. Provea un diagrama de flujo que permita la verificación teórica. **Puntos extra**: uso de strings (cadenas de texto).

Entregables:

1. Se deben adjuntar los códigos de ARM ASM en carpetas o archivos separados y organizados con el nombre del punto correspondiente, **tanto de la parte de comprobación, como de los ejercicios extra y la aplicación planteada**. Ejemplo: Ejercicio_1.s. Los códigos deben ir comentados. Dentro de cada programa d**ebe incluir el nombre del estudiante y código**. Agréguelo como un comentario al inicio de cada programa.

2. Realice un informe de laboratorio, formato IEEE, que contenga las siguientes partes: Título, resumen, introducción, marco teórico, objetivo, materiales, desarrollo (la resolución de cada uno de los puntos de esta práctica, complementando con análisis de resultados e imágenes de los programas, donde se vea su nombre en cada una de ellas), conclusiones y bibliografía en citación formato IEEE.

3. Guarde este archivo únicamente en formato PDF como InformARM2_ApellidoNombre_Codigo.pdf.

4. Envíe por medio del aula virtual UPTC un solo único archivo comprimido, formato ZIP que contenga los programas de cada una de los ejercicios y el informe generado en formato PDF.

Nota: el desarrollo de esta práctica se puede hacer en equipo de máximo 2 personas.