Proyecto 1. MIPS Ligero Máquina Virtual

1. Objetivo General

Desarrollar una máquina virtual (MIPS Ligero Máquina Virtual) capaz de ejecutar programas escritos con un conjunto reducido de instrucciones MIPS (MIPSLigero). Esta máquina virtual será desarrollada en lenguaje ensamblador MIPS.

Los programas que se ejecutarán en MLMV deben ser ensamblados en MIPSLigero word-code que corresponde al formato binario de las instrucciones que la máquina virtual es capaz de entender y ejecutar. Cada word-code de 32 bits contiene una instrucción que especifica el código de la operación a llevar a cabo y los operandos sobre los cuales actúa siguiendo los formatos de instrucción que serán especificados más adelante.

La MLMV tiene un conjunto reducido instrucciones para:

- Carga y almacenamiento en memoria
- Operaciones Aritméticas
- Transferencia de Control (saltos)

Esta máquina virtual deberá ejecutar cada instrucción de un programa ensamblado almacenado en memoria siguiendo el ciclo de ejecución.

2. Ciclo de Ejecución

Se define como el ciclo de ejecución de un procesador a los pasos internos que sigue para ejecutar una instrucción. El número de pasos y duración de este ciclo varían de procesador a procesador y depende de su arquitectura. A modo de simplificación para el desarrollo de nuestra máquina virtual, el ciclo de ejecución de las instrucciones de MIPSLigero constarán de cinco etapas:

- Fetch (F),
- Decodificación y búsqueda de registros (D),
- Ejecución (E)
- Acceso a memoria (Mem)
- Escritura de resultados (W)

2.1 Fetch

En esta fase el procesador obtiene de la memoria la siguiente instrucción a ejecutar . El procesador recibe los bytes de la instrucción y los almacena en el registro de instrucciones (IR) para proceder a su decodificación. Al mismo tiempo que se obtienen los bytes de la instrucción se incrementa el contador de programa. Como todas las instrucciones en MIPSligero son de 32 bits, el contador de programa se incrementa en 4.

2.2 Decodificación

En el proceso de decodificación se extrae el código de operación. Dependiendo del código de operación se procede a obtener el resto de los elementos de la instrucción tomando en cuenta los formatos de instrucción de MIPSLigero. Al terminar esta fase ya se sabe con exactitud qué operación se deberá realizar y la forma en que se obtendrán los operandos.

Formatos de Instrucción

Tipo R

(shamt: *shift amount* en instrucciones de desplazamiento)

Cód. Op.	Registro fuente 1	Registro fuente 2	Registro destino		Funct
xxxxxx	rs	rt	rd	shamt	funct
6	5	5	5	5	6
31-26	25-21	20-16	15-11	10-6	5-0

Tipo I

(carga o almacenamiento, ramificación condicional, operaciones con inmediatos)

Cód. Op.	Registro base	Registro destino	Desplazamiento
xxxxxx	rs	rt	Offset
6	5	5	16
31-26	25-21	20-16	15-0

Tipo J

(salto incondicional)

Cód. Op.	Dirección destino
xxxxxx	target
6	26
31-26	25-0

2.3 Ejecución

Una vez obtenidos los operandos, en esta fase se realizan los cálculos aritméticos/lógicos especificados con la operación. Si se trata de una operación de transferencia de control, se lleva a cabo la comparación de los operandos y el cálculo de la dirección de salto (actualización del PC en caso de cumplirse la condición). Para el caso de las operaciones de transferencia se calcula la dirección del operando en memoria.

2.4 Memoria

Para el caso de las operaciones de transferencia se accede a memoria tomando en cuenta la dirección del operando calculada en la etapa anterior.

2.5 Escritura de resultados

Una vez terminada la operación aritmético/lógica codificada en la instrucción, el procesador guarda el resultado en el registro destino especificado en la instrucción

Al terminar esta fase, el procesador tiene en el contador de programa la dirección de memoria en la que está almacenada la siguiente instrucción. La siguiente fase de fetch comienza la ejecución de una nueva instrucción.

3. Primera Fase de MLMV

Lectura y Carga en memoria del programa ensamblado

3.1 Objetivos Específicos

- Adquirir destrezas en programación con el lenguaje ensamblador MIPS
- Adquirir destrezas en el uso de llamadas al sistema para la lectura de archivos
- Adquirir destrezas en la manipulación de bits a nivel de lenguaje ensamblador

3.2 Detalles de Implementación

Para esta primera fase se desea que implemente un programa en MIPS para la lectura de un archivo que contiene un programa ensamblado en MIPSLigero y que posteriormente será ejecutado por la máquina virtual a desarrollar a lo largo de la asignatura.

MIPS ofrece una serie de llamadas al sistema para la Entrada y Salida a través de archivos.

Llamada	$v0	Argumentos	Resultado
open file	13	$a0 = address of null-terminated string containing filename $a1 = flags $a2 = mode	$v0 contains file descriptor (negative if error).
read from file	14	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read	$v0 contains number of characters read (0 if end-of-file, negative if error).
write to file	15	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write	$v0 contains number of characters written (negative if error).
close file	16	$a0 = file descriptor	

La llamada al sistema 14 (read from file) permite la lectura de un cierto número de caracteres (especificados en el registro $a2).
Un programa ensamblado en Hexadecimal luce de la siguiente forma:

```
2402000d
3c011001
34240000
24050000
24060000
0000000c
0002b021
2402000e
00162021
3c011001
```

En nuestra asignación el programa ensamblado se encontrará almacenado en el archivo ***programa.txt***

Su aplicación debe leer cada una de las palabras del programa ensamblado y almacenarlas a partir de la dirección de memoria identificada con la etiqueta programa.

La dificultad de la asignación se encuentra en que para leer cada una de las líneas del archivo deben leer 10 caracteres: 8 caracteres + 2 que corresponden al salto de línea (archivo creado en Windows). Deben tener en cuenta que cada caracter tiene una representación en ASCII ('0' = 48, '1' = 49, '2' = 50, etc. 'a' = 97, 'b' = 98, c = '99', d = '100', e = '101', f = '102'), y cuando por ejemplo se lee la línea 00162021, ésta se almacenará en 10 bytes de la siguiente forma

0016			
36	31	30	30

2021			
31	32	30	32

salto de línea			
00	00	0a	0d

Su programa deberá tomar la información almacenada en las dos primeras palabras que se muestran arriba y obtener como resultado 00162021 y almacenar este valor a partir de la dirección de memoria programa. Este proceso debe repetirse hasta el final del archivo. La última línea del programa contendrá 00000000

Como resultado debe imprimir por consola el programa en hexadecimal que fue cargado en memoria.

4. Segunda Fase de MLMV

Búsqueda de Instrucciones y Decodificación

4.1 Objetivos Específicos

- Adquirir destrezas en programación con el lenguaje ensamblador MIPS
- Adquirir destrezas en la manipulación de estructuras de datos
- Adquirir destrezas en la manipulación de bits a nivel de lenguaje ensamblador

4.2 Detalles de Implementación

Para esta segunda fase se desea que implemente un programa en MIPS que cargue en memoria un programa ensamblado en MIPS Ligero (fase1), y des-ensamble cada una de las instrucciones indicando la operación y los operandos sobre los cuales se lleva a cabo dicha operación.

Por ejemplo para el siguiente programa :

```
81090006
2129ffd0
000a5100
01495021
```

La salida a obtener es:

```
81090006          lb $9 $8 6
2129ffd0          addi $9 $9 -48
000a5100          sll $10 $10 4
01495021          addu $10 $10 $9
```

Recomendaciones para facilitar la implementación:
Es conveniente disponer de una tabla en el área de datos. Esta tabla será indexada usando el código de operación [0 ... 63] y contiene la siguiente información:

- Tipo de formato (2 bytes):
  - Formato-R = 0,
  - Formato-I = 1,
  - Formato-J = 2
- Campo (2bytes) que indica si dicho código de operación hace uso de la expansión de código funct
  - 0 = No usa el campo,
  - 1 = Si usa el campo

  Esto es necesario para el formato tipo R. Para el resto de los formatos no es necesario pero de igual forma se mantiene el campo en la tabla.
- Apuntador a la cadena de caracteres terminada con el caracter NULL que contiene el nombre de la operación asociada con el código de operación (4 bytes)
- Apuntador a la tabla auxiliar con los códigos de expansión si hace uso de códigos de expansión. En caso contrario contiene la dirección de la

función que deberá ejecutar la Máquina Virtual para llevar a cabo la operación especificada (4 bytes)

La tabla para los códigos de expansión contiene la siguiente información

- Apuntador a la cadena de caracteres terminada con el caracter NULL que contiene el nombre de la operación asociada con el código de operación (4 bytes)
- Dirección de la función que deberá ejecutar la Máquina Virtual para llevar a cabo la operación especificada (4 bytes)

```
tabla_coop:   .half 0  1                          # coop 0
              .word 0
              .word _tabla0
              .word 0 0 0                         # coop 1
              .half 3  0                          # coop 2
              .word __j
              .word _j
              .half 3  0                          # coop 3
              .word __jal
              .word _jal
              .half 1  0                          # coop 4
              .word __beq
              .word _beq

_tabla0:
              .word __sll          # código 0
              .word _sll
              .word 0 0            # código 1
              .word __srl          # código 2
              .word _srl
              .word 0 0            # código 3
              .word 0 0            # código 4
              .word 0 0            # código 5
              .word 0 0            # código 6
              .word 0 0            # código 7
              .word __jr           # código 8
              .word _jr


__j:          .asciiz "j"
__jal:        .asciiz "jal"
__beq:        .asciiz "beq"
__sll:        .asciiz "sll"
__srl:        .asciiz "srl"
```

__jr: .asciiz "jr"

Entrega:

La entrega de esta fase del proyecto es el día domingo 15 de Diciembre hasta las 11.59pm. Deben subir su programa **fase2-carnet.s** en Aula Virtual, para lo cual deberán crear el directorio **Proyecto1** dentro de la carpeta documentos de su grupo. Note que debe estar inscrito en algún grupo en aula Virtual para poder optar a esta opción.

El código debe incluir la identificación del Grupo y de sus integrantes, además debe contener una cantidad adecuada de comentarios incluyendo la planificación de los registros usados.