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 de 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. Tercera Fase de MLMV

En esta tercera etapa del proyecto deberán implementar la máquina virtual MLMV que ejecuta programas en lenguaje de máquina siguiendo el ciclo de ejecución antes mencionado. El desarrollo de su máquina virtual lo harán sobre el archivo MLMV.s que usarán como esqueleto. Sobre este archivo deberán agregar toda la funcionalidad necesaria para simular la ejecución de un programa ensamblado. Su programa deberá estar bien estructurado usando funciones y cumpliendo con las convenciones de responsabilidad compartida en cuanto al uso de los registros.

Su aplicación debe leer cada una de las palabras de un programa ensamblado y almacenarlas en la dirección de memoria **programa**. Deberá solicitarle al usuario el nombre del archivo que contiene el programa ensamblado (Primera Fase del proyecto).

Una vez cargado el programa en memoria, se inicia la ejecución del mismo a partir de la dirección contenida en **contador.** Esta dirección debe actualizarse en cada ciclo de **Fetch** con la dirección de la próxima instrucción a ejecutar.

En la dirección **coop** se encuentra almacenada la estructura de datos que contiene información de los códigos de operación de la máquina virtual. Esta tabla es accedida por medio del código de operación [0 ... 63] y contiene la siguiente información:

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

  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 extensión si hace uso de códigos de extensión. 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 extensión 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)

Los registros de la máquina virtual estarán implementados en memoria a partir de la dirección **registros**. A partir de esta dirección se reservan 128 bytes que se usarán para salvar el contenido de los 32 registros de la máquina virtual (32 * 4 = 128). Por ejemplo si el programa a ejecutar usa el registro $16, el contenido de dicho registros se encuentra en la dirección registros + 64

Cualquier acceso que el programa haga a la memoria estática, deberá transformarse en una dirección relativa a la etiqueta **memoria.** Es decir si se ejecuta la instrucción lw $10, 4($11), la dirección del segundo operando se transforma en memoria+4+ $10.

Si el acceso a memoria usa el registro $29 como en la instrucción lw $5, 0($29) entonces el cálculo de la direccion será pila+0+29. Recuerden que el $29 es el stack pointer.

La máquina virtual ejecuta el programa hasta que consiga la instrucción 0x00000000 que indica fin del programa. Una vez finalizada la ejecución deberán mostrar como salida el contenido de los 32 registros de la máquina virtual en hexademical

Reserva del espacio de memoria incluido en MLMV.s

```
.align 2
registros:.space 128          # espacio reservado para los registros de la maquina virtual

memoria:.space 1200           # espacio reservado para la memoria estatica de la maquina
virtual

fin_pila:.space 396           # espacio para implementar la pila de la maquina virtual
pila:.word 0                  # primera palabra disponible de la pila. La pila crece de
```

```
                              # direcciones altas a bajas

contador:.word programa       # Registro especial Contador de Programa. Contiene la
                              # direccion de la proxima instruccion a ser ejecutada en la
                              # Maquina Virtual

programa:.space 800           # espacio reservado para almacenar el codigo ensamblado del
                              # programa a ser ejecutado por la maquina virtual
```

Entrega:

La entrega de esta fase del proyecto es el día viernes 10 de enero de 2014 hasta las 11.59pm. Deben subir su programa como **G??-fase3.s** donde ?? corresponde a su número de grupo en Aula Virtual

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.