Universdad Simón Bolívar
Departamento de Computacón y TI
Organizacón del Computador

Proyecto
Máquina Virtual MIPS-USB

El objetivo de este proyecto es desarrollar una máquina virtual sobre MARS capaz de ejecutar programas ensamblados para la arquitectura MIPS-USB. A continuación se presenta una descripción de la arquitectura.

Arquitectura del conjunto de Instrucciones

MIPS-USB es un procesador de 32-bits RISC con 16 registros de uso general de 32 bits cada uno ($0 a $15). Posee además un registro de uso especial conocido como Contador de Programa (CP) que contiene la dirección de la próxima instrucción a ejecutar. Todas las instrucciones son de 32 bits y posee 2 formatos de instrucción: Tipo-R y Tipo-I.

Tipo-R

OP	Rd	Ra	Rb	No usado
8 bits	4 bits	4 bits	4 bits	12 bits

Tipo-I

OP	Rd	Ra	Inmediato
8 bits	4 bits	4 bits	16 bits

En el formato Tipo-R, **Ra** y **Rb** indican el número de los registros fuentes, y **Rd** contiene el número del registro que se usa como destino de la operación. Para el formato Tipo-I, **Ra** es el número del registro fuente y **Rd** es el registro destino. La constante inmediata de 16 bits corresponde a un número en complemento a dos.

Conjunto de Instrucciones

	Tipo	Significado	Codificación				
`add Rd, Ra, Rb`	R	`Rd= Ra + Rb`	`OP= 0x00`	`Rd`	`Ra`	`Rb`	`No usado`
`sub Rd, Ra, Rb`	R	`Rd= Ra - Rb`	`OP= 0x01`	`Rd`	`Ra`	`Rb`	`No usado`
`slt Rd, Ra, Rb`	R	`Rd= 1 si (Ra < Rb),` `0 en caso contrario`	`OP= 0x02`	`Rd`	`Ra`	`Rb`	`No usado`
`and Rd, Ra, Rb`	R	`Rd= Ra & Rb`	`OP= 0x03`	`Rd`	`Ra`	`Rb`	`No usado`
`or Rd, Ra, Rb`	R	`Rd= Ra | Rb`	`OP= 0x04`	`Rd`	`Ra`	`Rb`	`No usado`
`xor Rd, Ra, Rb`	R	`Rd= Ra ^ Rb`	`OP= 0x05`	`Rd`	`Ra`	`Rb`	`No usado`
`addi Rd, Ra, Inm`	I	`Rd= Ra + Inm`	`OP= 0x50`	`Rd`	`Ra`	`Inm`	
`slti Rd, Ra, Inm`	I	`Rd= 1 si (Ra < Inm),` `0 en caso contrario`	`OP= 0x51`	`Rd`	`Ra`	`Inm`	
`andi Rd, Ra, Inm`	I	`Rd= Ra & Inm`	`OP= 0x52`	`Rd`	`Ra`	`Inm`	
`ori Rd, Ra, Inm`	I	`Rd= Ra | Inm`	`OP= 0x53`	`Rd`	`Ra`	`Inm`	
`xori Rd, Ra, Inm`	I	`Rd= Ra ^ Inm`	`OP= 0x54`	`Rd`	`Ra`	`Inm`	
`sll Rd, Ra, Inm`	I	`Rd= Ra << Inm`	`OP= 0x55`	`Rd`	`Ra`	`Inm (5 bits)`	
`srl Rd, Ra, Inm`	I	`Rd= Ra >> Inm`	`OP= 0x56`	`Rd`	`Ra`	`Inm (5 bits)`	
`sra Rd, Ra, Inm`	I	`Rd= Ra >> Inm`	`OP= 0x57`	`Rd`	`Ra`	`Inm (5 bits)`	
`ror Rd, Ra, Inm`	I	`Rd= Ra >> Inm Rotate right`	`OP= 0x58`	`Rd`	`Ra`	`Inm (5 bits)`	
`lui Rd, Inm`	I	`Rd= Inm << 16`	`OP= 0x59`	`Rd`	`0`	`Inm`	
`li Rd, Inm`	I	`Rd= Inm`	`OP= 0x5A`	`Rd`	`0`	`Inm`	
`lw Rd, Inm(Ra)`	I	`Rd= Mem[Ra + Inm]`	`OP= 0x60`	`Rd`	`Ra`	`Inm`	
`sw Rd, Inm(Ra)`	I	`Mem[Ra + Inm]= Rd`	`OP= 0x61`	`Rd`	`Ra`	`Inm`	
`beq Rd, Ra`	I	`Si Rd==Ra PC=PC+Inm`	`OP= 0x62`	`Rd`	`Ra`	`Inm`	
`bne Rd, Ra`	I	`Si Rd!=Ra PC=PC+Inm`	`OP= 0x63`	`Rd`	`Ra`	`Inm`	
`Halt`	R	`Finaliza ejecución`	`OP= 0xFF`				

La arquitectura posee un espacio de direcciones de 2^{16} Bytes para las instrucciones (segmento de texto direccionado por el CP) y un espacio de direcciones para los datos también de 2^{16} Bytes (segmento de datos).

Modos de Direccionamiento

- **Desplazamiento relativo al CP**
  Utilizado para las instrucciones de salto. La dirección de salto se calcula de la siguiente forma:
  CP = CP + (extensión-signo(Inm) << 2)
- **Desplazamiento con registro base**
  Utilizado en las instrucciones lw y sw. La dirección del operando se calcula como:
  (Ra) + extensión-signo(Inm)
- **Modo Registro**
- **Modo Inmediato**

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 consta de las siguientes etapas:

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

Fetch

En esta fase el procesador obtiene del segmento de texto la siguiente instrucción a ejecutar. El procesador recibe los bytes de la instrucción 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 MIPS-USB son de 32 bits, el contador de programa se incrementa en 4.

Decodificación

En el proceso de decodificación se extrae el código de operación (OP). Dependiendo de este código se procede a obtener el resto de los elementos de la instrucción tomando en cuenta los formatos de instrucción (Tipo-R y Tipo-I). Al terminar esta fase ya se sabe con exactitud qué operación se deberá realizar y la forma en que se obtendrán los operandos.

Ejecución

Una vez obtenidos los operandos, en esta fase se realizan los cálculos aritméticos/lógicos especificados en 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 CP en caso de cumplirse la condición). Para el caso de las operaciones de transferencia se calcula la dirección del operando en memoria.

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.

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 y repite el ciclo de ejecución.

Primera Entrega

Lectura y Carga en memoria de un programa ensamblado y decodificación parcial de las instrucciones

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

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 para MIPS-USB y que posteriormente será ejecutado por la máquina virtual a desarrollar a lo largo del taller.

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 un archivo cuyo nombre proveerá el usuario a tempo de ejecución.

Su aplicación debe leer cada una de las líneas del programa ensamblado y almacenarlas en memoria. La dificultad de la asignación se encuentra en que para leer cada una de las líneas del archivo deberán leer 10 caracteres: 8 caracteres que corresponden a un dígito hexadecimal mas 2 que corresponden al salto de línea (para el caso de archivos creados en Windows, verificar el caso Mac). 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, ' A' =65, etc.), y cuando por ejemplo se lee la línea 00162021, ésta se almacenará en 10 bytes de la siguiente forma (little endian)

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, transformarlo al entero 00162021 y almacenar este nuevo valor en el segmento de texto de su máquina virtual. Este proceso debe repetirse hasta el final del archivo.
´

Además de implementar la carga inicial del programa en el segmento de texto de su máquina virtual deberán implementar la decodficación de las instrucciones. Para esta primera entrega no verificarán a qué tipo de formato pertenece cada una de las instrucciones. Deberán mostrar para cada nstrucción su decodficacón como Tipo-R y Tipo I.

Ejemplo:

```
0x 00FAB000
0x 03985600
```

La salida de su programa será

```
0x 00FAB000
     OP = 00
     Rd = F
     Ra = A
     Rb = B   o  Inm = B000

0x 03985600
     OP = 03
     Rd = 9
     Ra = 8
     Rb = 5   o  Inm = 5600
```

Segunda Entrega

En esta segunda etapa del proyecto deberán implementar la máquina virtual MIPS-USB capaz de ejecutar programas escritos en lenguaje de máquina siguiendo el ciclo de ejecución antes mencionado.

En esta entrega deberán:

- reservar espacio para el segmento de texto en donde se carga el programa a ejecutar,
- actualizar el contenido del CP con la dirección de la primera instrucción a ejecutar
- reservar espacio para el segmento de datos
- reservar espacio para los 16 registros de MIPS-USB
- implementar cada una de las operaciones del conjunto de instrucciones
- a medida que el procesador virtual ejecuta una instrucción deben mostrar el "des-ensamblaje" de la misma, es decir 0x00543000 add $5 $4 $3
- Al final de la ejecución, deben mostrar el contenido de cada uno de los 16 registros de MIPS-USB

En esta entrega se evaluará el diseño de la estructura de datos utilizada para el manejo de las distintas operaciones del conjunto de instrucciones, eficiencia en cuanto a los accesos a memoria, estructuración del código utilizando funciones, uso de las convenciones de responsabilidad compartida y documentación del código.

Fecha de entrega: Viernes de la semana 9 (6 de junio) 23:55