Proyecto: Máquina Virtual

22 de Abril del 2015

1 Objectivo del Proyecto.

La finalidad de este proyecto es que implementen un simulador de una arquitectura de computadora hipotética especificada en este documento.

Para poder implementar este simulador es necesario que conozcan las diferentes partes que lo componen y posteriormente lean la especificación dada de la máquina para implementarla fidedignamente. En este caso, van a implementar un intérprete del código máquina de la máquina virtual especificada. Para facilitar la programación de dicha máquina se les proporcionará un compilador de un lenguaje ensamblador similar a MIPS al código máquina del simulador que deben implementar.

1.1 Componentes del simulador

La implemetación de su máquina virtual debe tener los siguientes componentes:

1. Registros. Su máquina virtual constará de 14 registros. Más adelante se detalla qué debera hacer cada uno.

2. Memoria Primaria. Deberán simular de algún modo la memoria primaria de la máquina virtual (lo que sería la RAM de sus equipos). Recuerden, la memoria es simplemente un arreglo de bytes que almacenan datos, por lo cual debeŕan tratarla como tal.

3. Unidad Aritmético-Lógica y Unidad de Control. El nucleo principal de su máquina. Decodificará y ejecutará las instrucciones dadas en la especificación de la máquina virtual.

4. Llamada al sistema. La implemetacion de esta maquian virtual sólo dispondrá de 2 tipos de syscalls: imprimer y leer de consola, esto para comunicarse con el usuario.

5. Manejo de errores. En caso de que haya un fallo en la máquina (causado por el programador), en caso de un error fatal, detener la ejecución de manera controlada. Para reportar los errores deberán detener la ejecución de su máquina virtual y devolver el control al sistema operativo devolviendo

un código de error y además almacenar un volcado de memoria en un archivo. Se penalizará a las implemetacioones que salgan sin códigos de error correctos o que devuelvan errores del runtime de C (sementation fault, etc).

2 Forma de calificar

Tendrán que hacer una breve presentación de su proyecto de no más de 10 minutos, donde expliquen de manera general como lo implementaron, con que retos tuvieron que enfrentarse y que muestran un ejemplo de su simulador con un programa hecho por ustedes compilado para el mismo.

Para calificar el proyecto se utilizará el paquete de pruebas incluidos en el proyecto, su calificacion sera el porcentaje de casos de prueba satisfactorios contra el número total de casos de prueba.

De cualquier modo su proyecto será inspeccionado y posiblemente haga pruebas de forma manuales para buscar algún tipo de anomalía. Esto con la finalidad de evitar prácticas duplicadas o personas que no hayan hecho el proyecto (i. e. que alguien mas se los haya hecho). En caso de detectar dicha situación, se anula en su totalidad el proyecto.

3 Especificación de la máquina virutal

Su máquina virtual constará de:

1. 20 opcodes. 4 aritmética entera, 4 de aritmética de punto flotante, 4 operaciones de bits, 4 operaciones de memoria, 4 operaciones de salto de instrucción y la instrucción de llamada al sistema.

2. 14 registros. 8 Registros de propósito general, 2 registros de argumentos para llamada al sistema, 1 registro de retorno de datos de llamada al sistema, 1 registro de contador de programa, un registro de apuntador a la pila de memoria y un registro para apuntar a direccion de regreso(para implementar funciones).

3. 9 syscalls. 4 Instrucciones para leer de la consola , 4 para escribir y uno para terminar el programa.

3.1 Registros

La máquina contará con 14 registros de 32 bits cuyo uso se especificar a continuación:

Registros	Descripción
0,...,7	Registros de propósito general0x
8,9	Argumentos para llamada al sistema
10	Retorno de datos de llamda al sistema
11	Dirección de retorno
12	Contador de programa
13	Apuntador a la pila de memoria

3.2 Códigos de operación (opcodes)

Su máquina virutal deberá implementar los siguientes códigos de operación:

Operación	Código (en hex)	Duración (ciclos)	Descripción
add	0x0	3	Suma entera (con signo)
sub	0x1	4	Diferencia entera (con signo)
mul	0x2	10	Producto entero (con signo)
div	0x3	11	Cociente entero (con signo)
fadd	0x4	4	Suma flotante
fsub	0x5	5	Diferencia flotante
fmul	0x6	9	Producto flotante
fdiv	0x7	10	Cociente flotante
and	0x8	1	Operador de bits AND
or	0x9	1	Operador de bits OR
xor	0xA	1	Operador de bits XOR
not	0xB	1	Operador de bits NOT
lb	0xC	500	Cargar Byte
lw	0xD	1500	Cargar palabra (4 bytes)
sb	0xE	700	Guardar Byte
sw	0xF	2100	Guardar palabra (4 bytes)
li	0x10	1500	Cargar valor constante
b	0x11	1	Salto incondicional
beqz	0x12	4	Salto si es igual a cero
bltz	0x13	5	Slato si es menor que cero
syscall	0x14	50	Llamad al sistema

3.3 Llamadas al sistema (syscalls)

La forma de operar de las llamadas al sistema es igual que en MIPS: cargan un código de llamada al sistema en un registro, un argumento en uno o mas registros y el sistema devolverá algún datos en un tercer registro de retorno. Las convenciones de entrada y salida de datos de las llamadas del sistema serán:

1. Registro para codigo de llamda: 8
2. Registro para pasar agumento a la llamda: 9
3. Registro donde se reciben datos de la llamada: 10

Los códigos de llamada al sistema serán los siguientes:

- Códigos para leer de consola

Código	Significado	Retorno (en registro 10)
0x0	Leer entero	Entero leido
0x1	Leer caracter	Caracter leido
0x2	Leer flotante	Numero leido
0x3	Leer cadena	Numero de caracteres leidos. Se debe especificar como en el registro 9 la direccion

- Codigos para escribir en consola

Código	Significado	Argumento (en registro 9)
0x4	Escribir entero	Entero a escribir
0x5	Escribir caracter	Caracter a escribir
0x6	Escribir flotante	Numero a escribir
0x7	Escribir cadena	Dirección de memoria donde empezar a imprimr. La cadena debe estar delimit

- Salir del programa

Código	Significado	Argumento (en registro 9)
0x8	Salir del programa	No usado

3.4 Codificación de instrucciones

Las instrucciones de la máquina virtual seŕan de 32 bits (4 bytes) y estarán codificadas de la siguiente manera:

0	8	16	24
OpCode	Dr	Op1	Op2

Donde OpCode represental al código de operación, dr represeta el número de registro donde guardar el resultado, y Op1 y Op2 representa los números de registro de los operandos.

El caso de las instrucciones BEQZ y BLTZ la direccion de saltos estára en dr y el registro a evaluar será Op2.

Para los intrucciiones que sólo tiene un operando (carga y almacenamientos de memoria, saltos de instrucciones y el operador NOT) sólo se toma en cuenta como operando el valor almacenado en el campo Op2.

En caso de la operacion li es especial, ya que es una instruccion que tiene un tamaño de 6 bytes y su decodificación va como sigue:

0	8	16	24	32	40
0x10	rd	Cons.	de	32	bits

Esto complicará un poco la implemetanción de la máquina, sin embaro, no debe ser un obstáculo muy complicado de superar.

3.5 Pila de memoria

El stack pointer (registro 14) siempre empezara aputando al último byte de la memoria primaria. Los programas siempre se cargarán en lo primeros bytes de la

memoria, y el apuntador apunta del hacia el incion. Eso para evitar que los datos en la pila de memoria sobreescriban el código del programa en ejecucion. Por lo tanto, si el programor desea realizar una operacion push deberá decrementar el stack pointer y al realizar un pop incremetarlo (como en MIPS).

3.6 Códigos de error

En caso de ocurrir un error fatal, su máquina virtual deberá finalizar su ejecución devolviendo un código de error que especifica la falla ocurrida. Adicionalmente, deberán guardar un volcado de la memoria primaria en un archivo para propósitos de depuración. Si quieren devolver mensajes de error, lo deber´an hacer por medio del flujo de datos de error (stderr) y no en stdout ya que eso invalidará la prueba de su programa. Deben devolver el control al sistema operativo devolviendo un exit code como aparece en la siguiente tabla:

Código de Error	Significado
1	División entre cero
2	Dirección de memoria inválida
3	Memoria agotada
4	Número de rigistro inválido
5	Código de operacion inválido
6	Código de llamada a sistema inválido

4 Requerimientos del simulador

El simulador que implementen, además de poder ejecutra programas escritor en el lenguaje de máquina especificado en la sección anterior, deberá de implementar las siguientes crracterísticas:

4.1 Argumentos de línea de comando

La invocación de su máquina virtual deberá ser de la forma:

```
$ ./myvm -m 65536 helloworld.bin
```

Donde el argumento -m representa el tamaño de la memoria pricipal (medida en bytes), y el archivo helloworld.bin represetan un programa escrito en el lenguaje máquina que interpeta su simulador.

4.2 Ejecución

En caso de que la ejecución del programa sea satisfactoria, su simulador debe imprimer al final un número entero positivo que representa el tiempo de ejecución del programa medido en ciclos de reloj. Ejemplo:

```
$ ./myvm -m 1048578 helloworld.bin
Hello World!
6950
```

Para ello deberán de llevar la cuenta de ciclos de reloj de cada instrucción que ejecuta el programa e imprimir la suama al final de la ejecución. La duración de ciclos de reloj de cada instrucción está especificada en la tabla de operaciones que deben implementar.

Si el programa produce un error fatal en su máquina virtual, la ejecución se deberá de finalizar inmediatamente y debeŕan guardar un volcado de la memoria en el archivo dumpfile.bin No deben imprimer el número de ciclos de reloj en este caso, deben salir inmediatamente.

5 Compilador

Para facilitar la programación de su simulador, se les proporcionar´a un compilador de lenguaje ensamblador que podrán usar para producir código máquina de la máquina virtual que deben implementar. La semántica del compilador es exactamente igual que los códigos de operación de su máquina virtual, y la sintaxis es muy similar a la del intérprete de SPIM.

5.1 Comentarios

Para poner comentarios en su programa, se utiliza el caracter ;. Cualquier cosa que se encuentre después de este caracter se ignora.

5.2 Palabras reservadas

5.2.1 Instrucciones

Todas las instrucciones se delimitan por el caracter de nueva línea. Las siguientes palabras reservadas representan las instrucciones de la máquina virtual.

add, sub, mul, div, fadd, fsub, fmul, fdiv, or, and, xor, not, lb, sb, lw, sw, beqz, bltz, b, li, syscall

5.2.2 Insutrucciones adicionales

Para facilitar el trabajo del programador, el comiplador ofrece 1 instruccion adicional:

- mov (move). Para asignar el valor de un registro en otro. Funciona igual que en SPIM.

5.2.3 Registros

Los indentificadores de registro, al igual que en SPIM utilizarán el símbolo $. No se utilizará el número de registro com en el código máquina, sino un nombre más descriptivo que se muestra en la tabla a continuación:

Nombre	Número de Registro	Descripción
$r0,...,$r7	0,...,7	Registros de propósito general
$a0, $a1	8,9	Registros de argumentos en llamada al sistema
$s0	10	Registro de retorno para llamada al sistema
$ra	11	Registro para almacenar direcciones de retorno
$pc	12	Contador de programa
$sp	13	Apuntador al tope de la pila de memoria

5.2.4 Definiciones

Los macros para la definicion de datos son:

- .text Delimita el inicio del código de su programa
- .ascii

  Sirve para representar arreglos de bytes, por lo tanto, si quieren reservar un arreglo de enteros, recuerden mutiplicar el tamaño de arreglo por 4 su sintaxis es:

  .ascii ID INT

  Donde ID representa un identificador del arreglo e INT una literal entera que representa el tamaño en bytes

- .asciiz

  Representa cadenas de caracteres. A las cadenas definidas con .asciiz se les agrega aumaticamente el caracter nulo al final. Su sintaxis es:

  .asciiz STR ID

  Donde STR representa una literal de cadena e ID representa un identificador para la cadena. Las literalees de la cadena son del estilo de C. Se soportan sólo las secuencias de control \[bnft] para Backspace, Newline, Line Feed y Tab.

El delimitador .text es obligatorio en cualuquier programa para especificar en dónde inicia el código del mismo. Todos los demás macros son opcionales. Todas las definiciones de variables deberan hacerse antes del token .text

5.2.5 Etiquetas

Para definir subrutinas se puede hacer uso de etiquetado al igual que en SPIM utilizando el caracter : para especificar una subrutina. Todo programa debe tener una etiqueta main la cual representa el punto de arranque del programa.

5.2.6 Funciones no soportadas del intérprete SPIM

- No se soporta manejo de direcciones de memoria con la sitaxis Offset ($registro) por lo cual, toda la aritmética de direcciones la deberán de hacer manual.

- No se soporta el uso del frame pointer.
- No se soporta la instrucción jump and link. Deberán guardar la pista de la dirección de retorno en la pila de memoria de manera manual en caso de implementar funciones.
- No se permite usar identificadores para ninguna funcion instrucción que no sea li. Por lo tanto, siempre que se quiera un valor de variable (o identificador) se debe primero usar li par cargar en un registro.

5.2.7 Programas de ejemplo

El clasico Hello World:

```
; Programa que imprime
; La cadena de texto Hello World
.asciiz Hello Word! hwstr       ;definimos la cadena a imprimir
.text                           ;inicial el codigo de programa
main:
li $a0, 7                       ;Guardamos en a0 el codigo de servicio de
                                ;impresion en consola
li $a1, hwstr                   ;Pasamos la direccion de memoria de la
                                ;cadena a imprimir
syscall                         ;llamada al sistema
li $a0, 8
syscall                         ;llama al sistema para salir del programa
```

5.3 Instalación

Para poder instalar el compialdor, deberán desempacar el comprimido donde viene el codigo fuiente y compilarlo ingresando en el subdirectorio src y tecleando lo siguiente:

```
$make
```

Al final de este proceso deberán de obtener un archivo ejecutable llamdo sasm. La sintaxis del ensamblador es:

```
$sasm <archivo fuente> <binaro destino>
```

El compilador está totalmente contenido de dicho ejecutable y lo pueden mover a donde necesiten para poder compilar sus programas.

6 Paquete de pruebas

En el subdirectorio test se encuentra los archivos de prueba, tanto los programas escritos en ensamblador con terminacion .s y los que tiene el codigo maquina generador de dichos programas con terminacion .bin

Los programas de prueba llamdos err1, err2, err3, err6 como su nombre lo indican deben detenerse con el tipo de error correspondiente.

Los programas de prueba branch (prueba los saltos), flt (prueba operaciones con punto flotante), int (prueba operaciones con enteros), log (prueba operaciones logicas) y mem (prueba sw, lw, etc) tienen aparte un archivo con el nombre respectivo y terminacion .res dentro de este archivo esta lo que su simulador debe imprimir en la consola para que este sea correcto.

Por ultimo el programa de prueba syscall, pide al usuario los datos mismo que deben ser impresos inmediatamente despues para que esta prueba sea correcta.

7 Dudas y preguntas

Cualquier duda que tenga respecto al proyecto no duden en preguntar por correo, entre más oportunos sean menos probabilidad habrá de que se lleven sorpresas o que no puedan implementar el proyecto