Universidad Simón Bolívar.
Departamento de Computación
CI-3815 Organización del Computador.
Sep-Dic 2014.

Proyecto # 1 (8 %)

El objetivo de esta tarea es que el estudiante aprenda a utilizar el simulador MARS y el lenguaje ensamblador MIPS.

1 Parte 1: Ensamblador MIPS y simulador MARS

Obtenga el simulador MARS de la página de Moodle o de la dirección Web: http://courses.missouristate.edu/KenVollmar/MARS/index.htm. Una vez que la herramienta MARS esté instalada, será posible abrir la ventana estándar del Ambiente de Desarrollo Integrado (Integrated Development Enviroment, IDE) de MARS.
Abra el archivo proy1.asm seleccionando **File->Open** en la parte superior de la ventana del MARS. Esto hará que se vea en la ventana Edit de MARS el programa o seleccione **File-> New** si decide escribir el programa.
Analice el archivo y note que el editor de texto del MARS mostrará automáticamente el número de las líneas de código del archivo fuente en lenguaje ensamblador.
Si el archivo no tiene nombre debe colocarle uno que sea de su interés antes de poder ensamblarlo. Seleccione **Run->Assemble** del menú de la parte superior de MARS y note el cambio en la apariencia de MARS.
Corrija cualquier error a fin de obtener un código ensamblado exitoso.

1.1 Preguntas

1.1.1 Antes de ejecutar el código

1. Estudiar las directivas e instrucciones que aparecen en el código.¿ Cuántas directivas tiene el código?.

2. Seleccione la pestaña llamada Register en la ventana ubicada en la derecha del MARS y note que la mayoría de los registros están en cero excepto Stack Pointer ($sp), el Global Pointer ($gp) y el Program Counter ($pc).¿Cuáles son sus valores?

MIPS Register	gp(hexadecimal)	sp(hexadecimal)	pc(hexadecimal)
Value			

3. Rellene una tabla como la que se muestra a continuación, indicando las etiquetas del programa.¿ Cuáles son los nombres y direcciones (hexadecimal) de las etiquetas del segmento de programa(Text Segment) y cuáles la del segmento datos(Data Segment)?.

Text Segment		Data Segment	
Name	Address	Name	Address

4. ¿Número de instrucciones escritas del programa y número de instrucciones generadas por MARS?

5. Identificar la instrucción número 24 del programa y diga en cuántas instrucciones de lenguaje máquina se transformó?.

6. ¿Cuántos bytes de memoria ocupan los datos del programa?.

7. ¿Cuál es el contenido de la dirección de memoria 0x10010018?

Caracter								
Hexadecimal								

8. Ver la ventana del segmento de texto (Text Segment). Observar que hay cinco columnas. En la cuarta (Basic) aparece un código algo diferente al que hemos escrito. El código que hemos cargado de un archivo está en ensamblador con simbolos (etiquetas), escrito por el programador. El ensamblador lo convierte en un código ensamblador sin simbolos y sin pseudo-instrucciones, apto para ser directamente convertido en código de máquina, que está descrito en la tercera columna (Code). La segunda columna informa de las direcciones de memoria donde se encuentra cada una de las instrucciones.
   Identifique la instrucción número 31 de su programa. Escriba los contenidos de las columnas Address, Code y Basic.

Address(hexadecimal)	Code(hexadecimal)	Basic(ensamblador)

   Haga la conversión de la columna Code a Binario.

Hexade																															
Binary																															

1.1.2 Después de ejecutar el código

Seleccione **Run->Go** desde el Menú de la parte superior de MARS y observe los cambios

1. Estudiar cómo se cargan en registros las distintas posiciones de un arreglo de datos que está en memoria. Qué operaciones aritméticas son necesarias realizar (y porqué). Observar los valores que han tomado algunos de ellos. Para explicar esos valores, observar las operaciones descritas por las instrucciones de nuestro código. ¿Cuántos registros han cambiado de valor?. Rellene una tabla como la que se muestra indicando los cambios que han ocurrido en dichos registros

Register		Value	
Name	Number	Hexadecimal	Decimal

2. La memoria de datos ha sufrido algún cambio?. Muestre el estado del conjunto de direcciones de memoria.

Memory Address	Contenido
0x10010000	
0x10010004	
0x10010008	
0x1001000c	
0x10010010	
0x10010014	
0x10010018	
0x1001001c	
0x10010020	

3. Describa lo que hace el programa

4. ¿En que dirección se detiene el programa?

1.1.3 Modificacion del código

1. Reeemplace en el código las instrucciones 28,31 y 34 (mul $s5,$s4,$s6) por sll $s5,$s4,2 y vuelva a ensamblar el programa y vea que pasa. Ejecute de nuevo el programa. ¿Que ha cambiado?.

2. Puede identificar si alguna instrucción sobra o no cumple con alguna función?. Identifiquela y diga que hace?.

2 Parte 2 Servicios de Entrada y Salida

MARS ofrece una forma de interactuar con el usuario recibiendo datos de entrada y/o devolviendo datos de salida. Estas entradas y salidas, así como otros servicios, son ofrecidos a través de llamadas al sistema operativo, usando la instrucción **syscall**. Ahora, cómo es posible decirle al sistema operativo cuál servicio queremos utilizar?. La respuesta es simple. Antes de usar la instrucción **syscall**, es necesario cargar en diferentes registros los parámetros requeridos por el sistema operativo a fin de saber qué servicio ejecutar. Observe el menú de ayuda del MARS para conocer todos los servicios disponibles y los parámetros requeridos.
Abra el archivo proy12.asm seleccionando **File->Open** en la parte superior de la ventana del MARS. Esto hará que se vea en la ventana Edit de MARS el programa o seleccione **File-> New** si decide escribir el programa.
Analice el código para conocer como se solicitan los diferentes servicios.

2.1 Ensamble y Ejecute

1. Ensamble y ejecute el programa paso a paso (**Run->Step**). Asegurece que la opción "popup dialog for input syscalls" del **Settings** en el menú principal, esté seleccionado. Fijese en la ventana de mensajes,la pestaña **Run I/O** y escriba las salidas que se van obteniendo.

2. Diga como cambia la memoria de datos durante la ejecución.

Memory Address	Contenido antes ejecutar	Contenido despues ejecutar
0x10010000		
0x10010004		
0x10010008		
0x1001000c		
0x10010010		
0x10010014		
0x10010018		
0x1001001c		
0x10010020		
0x10010024		
0x10010028		
0x10010034		
0x10010038		

3. Modifique el código cambiando la instrucción li $a1,9 por li $a1,28. y vuelva a ejecutar el programa e introduzca un string de al menos 28 caracteres. Diga como cambia la memoria de datos.

Memory Address	Contenido antes ejecutar	Contenido despues ejecutar
0x10010000		
0x10010004		
0x10010008		
0x1001000c		
0x10010010		
0x10010014		
0x10010018		
0x1001001c		
0x10010020		
0x10010024		
0x10010028		
0x10010034		
0x10010038		

Fecha de entrega: Viernes 17-10-2014. Debe, en horas de clase, entregar el documento impreso con el proyecto resuelto, y el día anterior hacer submit del archivo a Moodle.