República Bolivariana de Venezuela

Universidad de Carabobo

Facultad de Ciencias y Tecnología

Departamento de Computación

Asignatura: Arquitectura del Computador

Alumno:

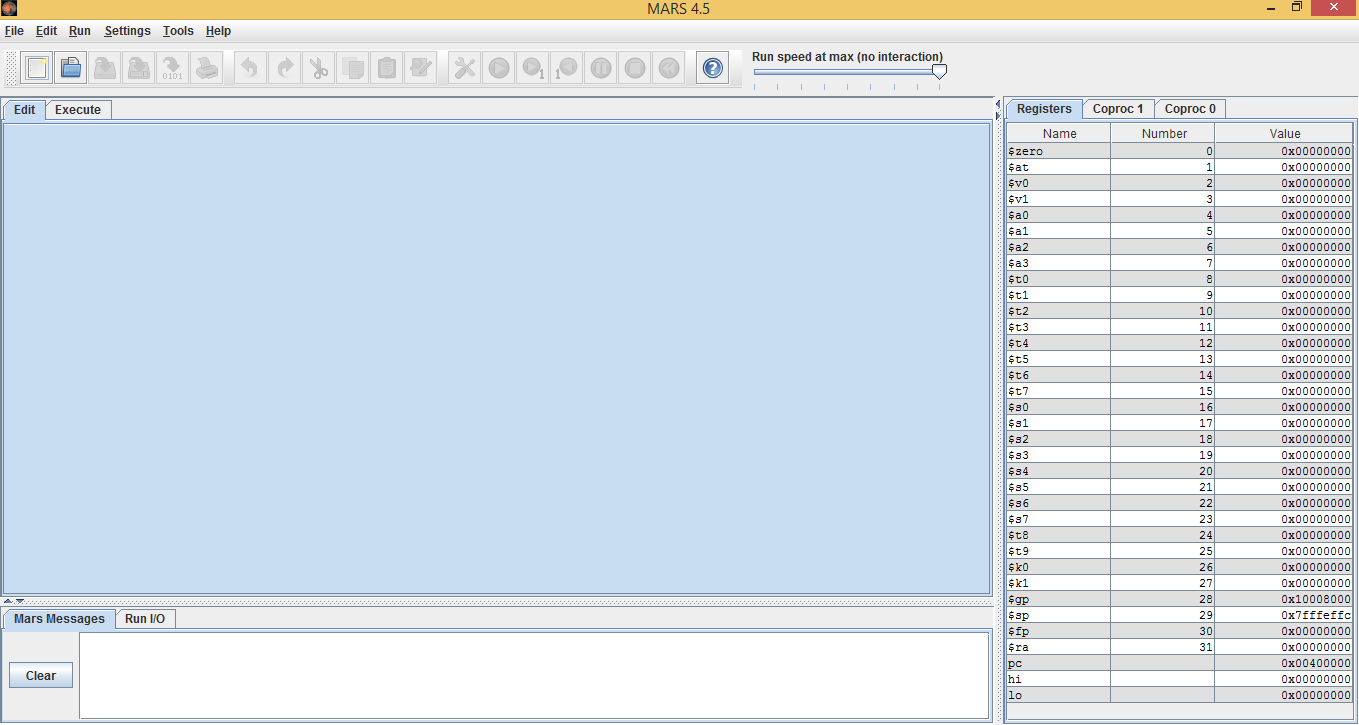
* Hector Abreu C.I. 30.000.231

Informe Laboratorio 1

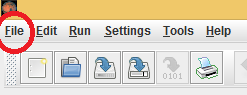
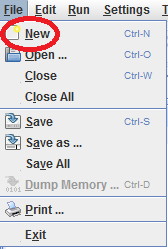
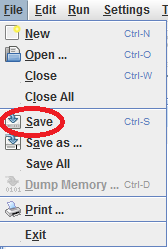
Configuración del Sector (ambiente de trabajo)

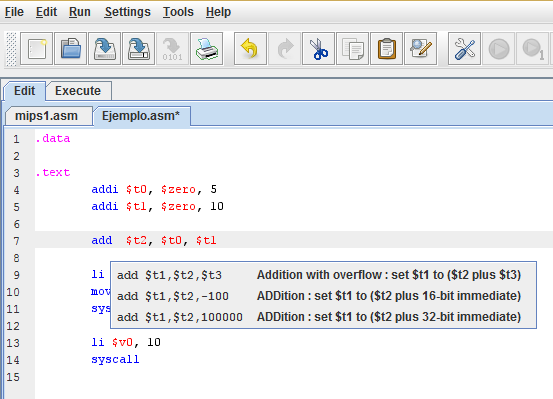
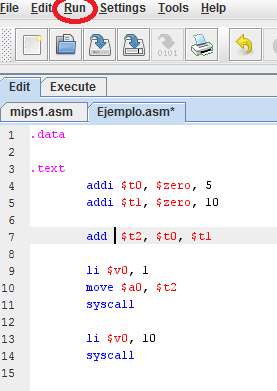
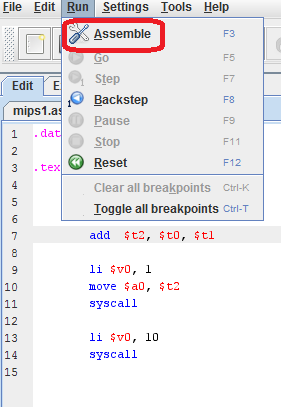
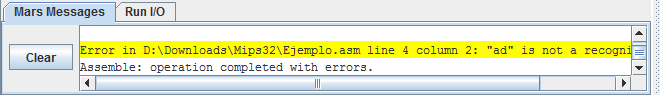
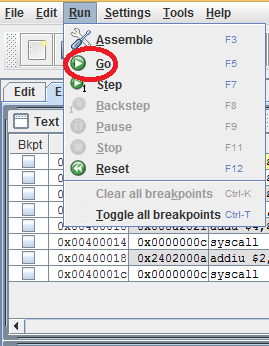
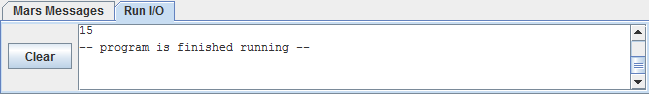
Mars (MIPS Assembler and Runtime Simulator) es un entorno de desarrollo integrado (IDE) para programar en lenguaje ensamblador MIPS. Iniciamos con la instalación del programa.

1. Descargar Mars: Lo primero que se debe hacer es descargar el archivo ejecutable de Mars desde su página oficial: <http://courses.missouristate.edu/KenVollmar/mars/>. Ahí se encontrará la versión más reciente del programa.
2. Instalar Mars: Una vez que se haya descargado el archivo, se procede a abrirlo y seguir las instrucciones para instalar Mars en su computadora. Es un proceso muy sencillo y no debería ocurrir problemas.
3. Abrir Mars: Una vez que se haya instalado Mars, se abre y se verá su entorno de desarrollo. En la ventana principal, encontramos una barra de menú y varias pestañas. A la derecha podemos observar los diferentes registros que funcionan como variables y que al apuntarles cada uno muestra una pequeña descripción de su uso.



1. Crear un nuevo archivo: Para comenzar a programar, debe crear un nuevo archivo. Hacemos clic en "File" en la barra de menú y selecciona "New". Luego, se realiza clic en “save” y dele un nombre a su archivo y guárdelo en una dirección deseada.

1. Escriba el código: Ahora puede comenzar a escribir tu código en el editor de texto de Mars. A medida que se escriba, el programa mostrará sugerencias de sintaxis y errores. 
2. Compila el código: Una vez que haya terminado de escribir su código, debe compilarlo para asegurarse de que no haya errores. Se realiza clic en "Run" en la barra de menú y selecciona "Assemble". Si hay errores, Mars los mostrará en la ventana inferior.   Ejemplo de error: 
3. Ejecución del código: Si el código compila sin errores, puede ejecutarlo haciendo clic en "Run" y seleccionando "Go". Verá el resultado de su programa en la ventana inferior.  

Funcionamiento de los algoritmos y comparación de instrucciones

Utilizadas en estos con respecto a las utilizadas en el libro

Fibonacci:

La secuencia de Fibonacci consiste en la suma de los dos números anteriores para conformar el número actual, en el ejemplo que se ve en la página con el nombre de **Fibonacci.asm** se lleva a cabo el mismo concepto con ciertos detalles, en este caso utilizaremos un arreglo en el cual se van a almacenar (en este caso) 19 números de la secuencia, por lo cual se declara tanto el arreglo como la cantidad de número como variables globales en “.data” para luego ser cargados en registros en “.text” y poder operar, siendo el resultado del algoritmo impreso por pantalla mostrando los 19 primeros números de la secuencia de Fibonacci. También se observa el algoritmo que permite que el usuario ingrese la cantidad de números a imprimir, pero se mantiene comentado.

Introducción a nuevas instrucciones (Fibonacci):

La instrucción nueva principalmente sería bgtz (Branch if greater than zero) la cual como su nombre lo indica, realiza es un salto condicional que compara un registro con el valor 0, y en caso de el registro ser mayor, se realiza el salto.

Luego con funciones similares se encuentran bgt (Branch if greater than) y blt (Branch if less than), las cuales se encargan de comprobar que el número ingresado por el usuario sea válido, en el caso del algoritmo que permita al usuario ingresar la cantidad de números de la secuencia a leer.

Row-Major y Column-Major:

Los algoritmos **row-major.asm** y **column-major.asm** funcionan de manera similar en cuanto a lo que hacen, recorren una matriz cuadrada de 16x16 rellenándola con el valor de un contador inicializado en 0 y que aumenta en cada iteración, resultando en una matriz de números que van desde el 0 hasta el 255, la principal diferencia es que row-major.asm rellena la matriz por filas y column-major.asm rellena la matriz por columnas.

Introducción a nuevas instrucciones (Row-Major y Column-Major):

La nueva instrucción que se nos presenta vendría a ser mult (multiplication), la cual realiza una multiplicación de dos números de 32 bits y almacena el resultado en un registro de 64 bits, que es el resultado de la unión de dos registros especiales llamados “HI” y “LO”.

Luego estaría la instrucción mflo (move from LO register), la cual copia el valor almacenado en el registro LO y lo ingresa en el registro destino.

Funcionamiento de Acumulador de una suma recursiva:

El código implementa un programa en MIPS 32 que solicita al usuario que ingrese un número, lo suma recursivamente hasta llegar a 1 y luego imprime el resultado.

Primero, se define el mensaje a mostrar al usuario y el mensaje de resultado. Luego, se inicia el programa principal "main".

Dentro de "main", se utiliza la llamada al sistema para imprimir el mensaje "Ingrese un numero:" y luego se utiliza la llamada al sistema para leer un número ingresado por el usuario.

Después, se mueve el valor leído del registro $v0 al registro $a0 y se salta a la función "sum".

En "sum", se compara si el valor en $a0 es menor que 1 y, si es así, se salta a "sum\_exit". Si no, se suma el valor en $a0 al valor en $a1 y se resta 1 al valor en $a0 antes de saltar de nuevo a "sum".

Cuando se alcanza la condición de salida en "sum\_exit", se mueve el valor en $a1 al registro $t0 y se salta de vuelta a "main".

En "main", se imprime el mensaje de resultado y se mueve el valor en $t0 al registro $a0 antes de imprimirlo. Finalmente, se llama a la función de salida del programa.

Funcionamiento de Método de ordenamiento BubbleSort:

El código implementa el algoritmo de ordenamiento burbuja en MIPS 32 para ordenar un array de números desordenados. Primero, se declara el array y su longitud, así como algunas cadenas de caracteres para imprimir mensajes. Luego, se cargan los valores iniciales en los registros correspondientes y se inicia un loop principal.

Dentro del loop principal, se inicializa el índice del primer elemento a comparar y se inicia un loop interno que compara pares de elementos adyacentes en el array y los intercambia si están en el orden incorrecto. Luego, se avanza al siguiente par de elementos y se repite el proceso hasta llegar al final del array.

Una vez que se ha completado una pasada completa por el array sin realizar ningún intercambio, se incrementa el índice del primer elemento a comparar y se vuelve a empezar el loop interno. Esto se repite hasta que se ha completado una pasada completa por el array sin realizar ningún intercambio, lo que significa que el array está ahora ordenado en orden ascendente.

Finalmente, se utiliza un loop para imprimir los elementos del array ordenado, separados por comas, y se imprime un mensaje de finalización antes de salir del programa.