Universidad Católica
San Pablo

Práctica N° n

ESCUELA PROFESIONAL: Ciencia de la Computación

SEMESTRE : 2021-02

CURSO : Arquitectura de computadores

DOCENTE : Dr. Yván Jesús Túpac Valdivia

NOMBRES Y APELLIDOS: Fredy Goolberth Quispe Neira

4.2 Pseudo instrucciones

Una "Pseudo instrucción" es una instrucción que tiene la sintaxis de una instrucción MIPS, pero al ensamblarse se convierte en una o varias instrucciones reales MIPS cuyo funcionamiento es el indicado por la pseudo instrucción. Observe y explique por qué el código que se presenta en el archivo `hello.asm` es diferente al del segmento de código ya ensamblado que muestra el emulador.

Porque como se nos explicó, una pseudo-instrucción funciona algo así como una macro ya predefinida para una acción en específico; por ejemplo: la pseudo-instrucción "li" asigna un valor a un registro cualquiera, en este caso ***"li $v0, 4"***, pero en instrucciones miniMIPS, no es posible hacer eso, lo que se puede hacer con miniMIPS es hacer una suma inmediata sin signo "addiu(add immediate unsigned)" con el valor 0, el cual está almacenado por defecto en la posición $0; por ende, la pseudo-instrucción es equivalente a "addiu $v0, $zero, 4".

4.3 Programa Fibonacci.asm

Descargue el archivo `Fibonacci.asm` desde

`https://www.dropbox.com/s/1k77uq3qfsm5bn4/Fibonacci.asm?dl=1`

e insertelo en un nuevo archivo en MARS

- Ejecute `Fibonacci.asm` en el emulador.
- Observe y experimente el funcionamiento del programa. Observe la ventana de datos notando cómo se va modificando los valores del archivo de registros hasta encontrar las respuestas deseadas.
- Revise cada una de las instrucciones y directivas de compilación que aparecen en el programa. Identifique la pseudoinstrucciones en este programa.
- Haga un resumen indicando sus observaciones, incluya un diagrama de flujo y haga el pseudo código de este programita.
- Respuestas a estas tres preguntas deben presentarse en un informe en la siguiente clase/sesión práctica.

Pseudo-instrucciones en el programa:

a) "la $registro, variable": load address, carga inmediata de una variable en memoria al registro
b) "li $registro, word": load immediate, carga inmediata de un word en el registro
Solo ésas son pseudo-instrucciones, las demás: addi, syscall, beq, bltz y j son instrucciones.

Nota: ***Se entregará el informe de acuerdo con lo indicado en la plantilla de trabajo.***

5.1 Algoritmo de ordenación por selección

Implementar y ejecutar en en MARS el algoritmo visto en clase de *Ordenación por Selección*, que utiliza un procedimiento `max` para hallar el valor máximo de un array de números y la posición donde fue encontrado.

1. Verifique el correcto funcionamiento de la ordenación
2. Explique qué ocurre cuando se encuentran números repetidos
3. Elabore un diagrama de flujo y pseudo algoritmo del programa ejecutado.

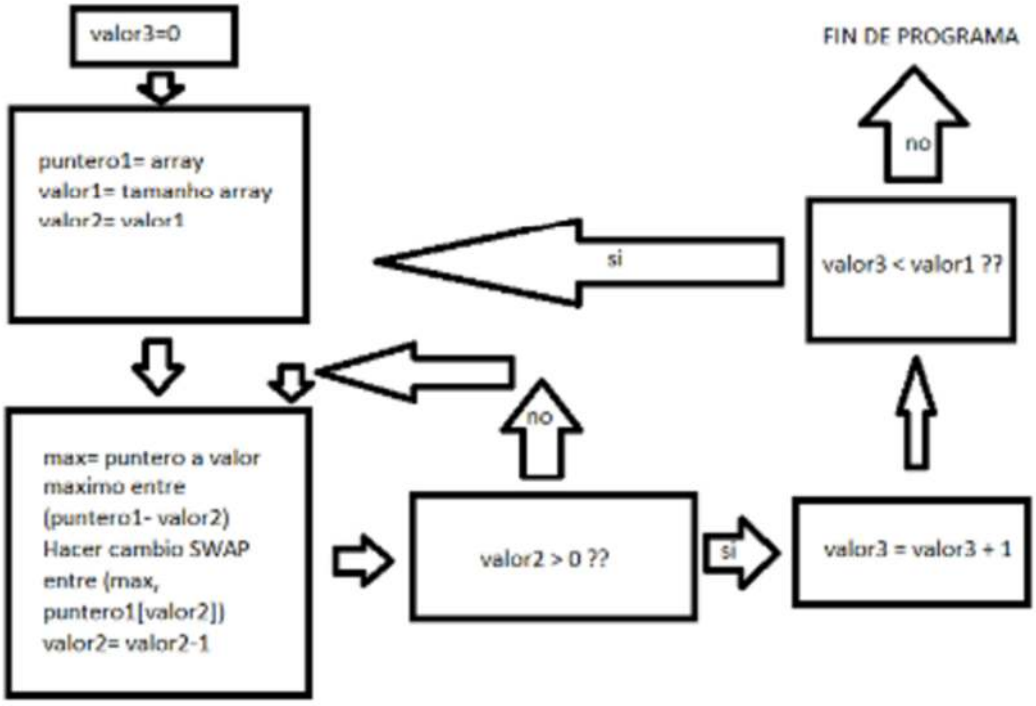
- Cuando se encuentran números repetidos, en el slt se guarda el número 0 en el registro, porque no cumple la condición, al igualarlo con $zero, este se regresa al inicio del bucle obviando las demás instrucciones, que son el cambiar el valor del máximo.
- PseudoAlgoritmo:

```
Valor=0
Mientras que ( valor < tamanhoArray):
        Valor1= tamanhoArray
        Mientras que ( Valor1 > 0):
                Encontrar puntero que apunte el valor máximo desde inicio
                hasta valor1
                Valor1= Valor1 – 1

        Hacer cambio en (valorMáximo, array[valor1])

        Valor= Valor + 1
```

- Diagrama de flujo:



Url al repositorio del código:

https://github.com/2505Fredy/Assembler-ejercicios

En el mencionado repositorio se encuentran los códigos de los archivos descargados, al igual que el código del programa creado. (sort.asm)

6.1 Promedio de números enteros

Copie o descargue el siguiente código al MARS y grabe el programa con el nombre `average.asm`:

- Ejecute el programa y verifique paso a paso cuáles son los registros que se están utilizando (tanto del Procesado Principal como del CoProc01). Preste atencion a las instrucciones que actuan en CoProc01, las de conversio de formato y paso de información entre el Procesador Principal y CoProc01.
- Describa e informe lo ocurrido.

INFORME:

- Primeramente, empieza asignándole el valor 11 (lenght array) al registro $s1
- Le asigna al registro $s0 la dirección en memoria del array.
- Inicializa los registros $t0, $t2 y $t3 en 0.
- Inicia un bucle con una condición de pare, el cual es, si los registros $t2 y $s1 son iguales, finaliza el bucle. Las instrucciones dentro de él son:
  a) Inicializa el registro $t4 asignándole la dirección que $s0 tiene del array más la variable en el registro $t3.
  b) Carga en $t1 la variable de 32-bit que $t4 apunta.
  c) Acumula en $t0 sumando el mismo con la variable que se cargó en $t1.
  d) Aumenta en una unidad el valor del registro $t2, el cual se verifica en la condición de pare.
  e) Asigna a $t3 el valor que $t2 tiene.
  f) En las siguientes instrucciones suma 4 veces el valor que tenía $t3, esto con el fin de avanzar de 4 en 4 bytes, para obtener el siguiente elemento del array.
  g) Se verifica la condición, si ésta cumple termina el bucle; caso contrario, la siguiente instrucción continúa el bucle enviando al PC al inicio del bucle.
- Después de que termine el bucle, todos los elementos están sumados en $t0, la siguiente instrucción copia el valor que éste contiene en $f8, el cual es un registro en punto flotante dentro del coproc1.
- Se hace lo mismo con $s1, el cual tiene la dimensión del array, se copia en $f9.
- Puesto que éstos datos vienen del procesador, éstos están en formato entero, se les hace la conversión a punto flotante teniendo como destino la misma dirección de memoria.
- Se hace la división en flotante de $f8/$f9 y se guarda el resultado en $f12. Esto para realizar la impresión de la variable flotante.
- Se hacen impresiones de cadenas.
- Finalmente se imprime el resultado y termina la ejecución del programa.