Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI-0114 Fundamentos de Arquitectura de Computadoras

Profesor: Luis Araya Carballo

Estudiantes:

- Amber Villarreal Campos - C28481
- Liqing Yosery Zheng Lu - C38680

Proyecto Final: Programación en ensamblador MIPS

Índice

Descripción 2
Consideraciones técnicas: 2
Restricciones del programa 2
Ejecución del programa 2
Resultados 3
Funciones Desarrolladas 3
1. MANEJO_TURNOS 3
2. TURNO_HUMANO 3
3. ERROR_FUERA_DE_RANGO 4
4. TURNO_MAQUINA 4
5. TIRADA_X 5
6. TIRADA_RANDOM 5
7. MOVIMIENTO_GANADOR 5
8. GANADOR_HUMANO 5
9. GANADOR_MAQUINA 6
10. separate1 y separate2 6
11. setContador y setTurnos 6
12. initial 6
13. contadorInitial y contadorInterface 7
14. turnosInitial y turnos 7
15. evaluar 7
16. Números (0 a 9) 7
Diagrama de Flujo 8

Descripción

"Llegar a 100" es un juego de dos jugadores, en donde en cada turno, se le puede sumar al contador un número del 1 al 10. El ganador es el jugador que, en su turno, logra llegar a 100.

En el caso de este programa, el jugador competirá contra una máquina, la cual estará programada con la intención de ganar la mayoría de partidas posibles.

Consideraciones técnicas:

- El programa será desarrollado para que el usuario juegue contra una máquina.
- El usuario tomará el primer turno.
- Solo hay una partida conocida en la que la máquina no gana: entradas para que el contador del usuario dé, en orden = (1, 12, 23, 34, 45, 56, 67, 78, 89, 100).
- Se desarrollará una interfaz gráfica, en la cual se mostrará en una ventana el estado actual del contador y el número de turnos que ha durado la partida.
- El programa se terminará de ejecutar una vez el contador llegue a 100.
- Si se le ingresa un carácter diferente a entero, ocurrirá un error de syscall y el programa terminará de ejecutarse.

Restricciones del programa

- Para la interfaz gráfica se debe tomar en cuenta las restricciones que se pueden enfrentar al programar en ensamblador MIPS.
- La máquina estará equipada con una lógica implementada para calcular el movimiento ganador de manera que siempre gane.

Ejecución del programa

La descripción detallada para la ejecución del programa se encuentra en el documento *'README.md'* presente en el repositorio realizado para la documentación del progreso del proyecto, disponible en el link: *https://github.com/cerillo2808/Llegar_a_100*

Resultados

Imprime de quién es el turno al principio de cada uno. Si es el usuario, le pide que ingrese un número del 1 al 10. En caso de que se le ingrese un número fuera del rango pedido, se imprime un mensaje pidiendo otro número para continuar con la jugada. Una vez ingresado un número válido, se imprime un mensaje de confirmación sobre el número que ya se logró sumar al contador, además del estado del contador actual, estos dos contadores van a ser presentados en la interfaz gráfica del programa.

En el turno de la máquina se imprime el número que se jugó y el estado del contador actual. Cuando el contador llegue a 100, el programa imprimirá una felicitación al usuario si este gana y la pantalla se tornará verde, de lo contrario, imprime un mensaje indicando que ganó la máquina y la pantalla se tornará roja.

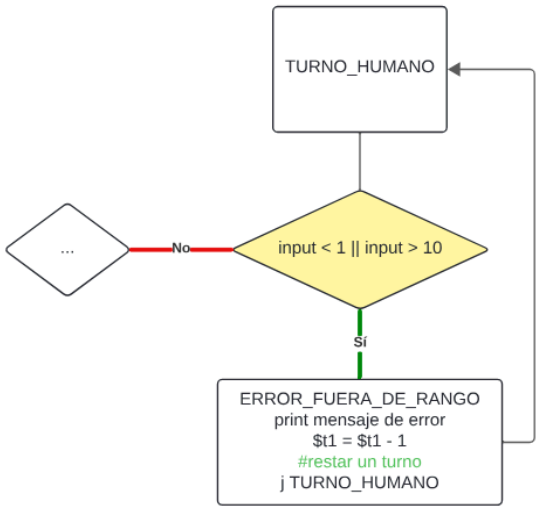
Funciones Desarrolladas

Aquellas funciones que no presentan un diagrama de flujo se encuentran desarrolladas en el diagrama de flujo del programa principal.

1. MANEJO_TURNOS:
    a. Parámetros: Recibe la cantidad de turnos, guardado en el registro $t1. El registro $t9 para el manejo de la interfaz gráfica y $t5 para más adelante.
    b. Funcionalidad: Inicia guardando el valor 0 en $t5 y haciendo un branch a *separate1* si .word 'impresion' en $t9 es igual a 0, en caso de no serlo continua a la siguiente instrucción convirtiendo 'impresion' a 0 a través de $t0. Si el turno es par, salta a *TURNO_HUMANO*, de lo contrario, salta a *TURNO_MAQUINA.*
2. TURNO_HUMANO:
    a. Parámetros: Recibe la cantidad de turnos, guardado en el registro $t1, y el contador guardado en $a1.

b. Funcionalidad: Pide que se ingrese un número del 1 al 10. Si el número ingresado no cumple con el rango, salta a la función *ERROR_FUERA_DE_RANGO*. Le suma al contador $a1 el número ingresado y a la cantidad de turnos le suma 1. Salta a *separate1*.

3. ERROR_FUERA_DE_RANGO:
   a. Diagrama:



   b. Parámetros: Recibe la cantidad de turnos guardado en le registro $t1.
   c. Funcionalidad: Imprime el mensaje de error, le resta 1 a la cantidad de turnos y salta a *TURNO_HUMANO*.

4. TURNO_MAQUINA:
   a. Parámetros: Recibe la cantidad de turnos, guardado en el registro $t1, y el contador guardado en $a1.
   b. Funcionalidad: Le suma 1 a la cantidad de turnos. Si el contador actual es igual a algún número crítico (1, 12, 23, 34, 45, 56, 67, 78, 89), salta a *TIRADA_RANDOM*. Si el contador actual es menor o igual a 90, salta a *MOVIMIENTO_GANADOR*. De lo contrario, salta a *TIRADA_X*, siendo X el número crítico más cercano.

5. TIRADA_X:
    a. Parámetros: Recibe el contador guardado en $a1.
    b. Funcionalidad: Encuentra el número que se le debe sumar al contador para que sea igual a X restando X con el contador actual. Imprime en pantalla el número encontrado y se lo suma al contador. El contador termina siendo igual a X.
    c. Observación: Hay 8 métodos de *TIRADA_X*, siendo las X los números 12, 23, 34, 45, 56, 67, 78, y 89.
6. TIRADA_RANDOM:
    a. Parámetros: Recibe el contador guardado en $a1.
    b. Funcionalidad: Genera un número random del 0 al 9. Le suma 1 al número random generado para que esté dentro del rango deseado. Imprime en pantalla el número random y se lo suma al contador.
7. MOVIMIENTO_GANADOR:
    a. Parámetros: Recibe el contador guardado en $a1.
    b. Funcionalidad: Encuentra el número que se le debe sumar al contador para que sea igual a 100 restando 100 con el contador actual. Imprime en pantalla el número encontrado y se lo suma al contador. El contador termina siendo igual a 100.
8. GANADOR_HUMANO:
    a. Parámetros: No recibe nada.
    b. Funcionalidad: Imprime el mensaje de ganador humano, continúa a *loopWin*, un for donde se torna la pantalla de Bitmap Display verde y termina la ejecución del programa.

9. GANADOR_MAQUINA:
    a. Parámetros: No recibe nada.
    b. Funcionalidad: Imprime el mensaje de ganador máquina continúa a *loopLoose*, un for donde se torna la pantalla de Bitmap Display roja y termina la ejecución del programa.
10. separate1 y separate2:
    a. Parámetros: Registros $a1 y $t1 utilizados para el contador y turnos de la partida, registro $t9 para el manejo de la interfaz gráfica y $t7 con el valor 10.
    b. Funcionalidad: En el caso de *separate1* convierte a $t9 en 1 y lo guarda en .word 'impresion', divide $a1 entre $t7, guarda en el registro lo el cociente y en el registro hi el residuo, importante para la impresión de los contadores en Bitmap Display más adelante, guarda el valor de lo en $t8. Si es *separate1*, salta a *initial*, si es *separate2*, salta a *turnosInitial*.
11. setContador y setTurnos:
    a. Parámetros: Registros $t4 para desplazarse en la pantalla de Bitmap Display y $t8 para manejo de contador (en $a1) y turnos (en $t1).
    b. Funcionalidad: Para ambos guarda el valor en el registro hi en $t8. Si es *setContador*, salta a *contadorInterface*, si es *setTurnos*, salta a *turnosturnos.*
12. initial:
    a. Parámetros: $t9 para cambiar el colo, $t4 para desplazarse.
    b. Funcionalidad: Initial guarda por $t9 el color a imprimir en la primera mitad de BitmapDisplay (el contador), continúa al for *loop* donde imprime dicho color hasta cubrir la primera mitad, hace branch a *loop2* donde se guarda el segundo color a imprimir en la segunda mitad de la pantalla (los turnos) y salta a *contadorInitial* al terminar.

13. contadorInitial y contadorInterface:
    a. Parámetros: Registro $t4 para desplazarse en la pantalla, $t9 para guardar el color a imprimir, $t8 para saber cuál número se debe imprimir.
    b. Funcionalidad: *contadorInitial* establece el bit particular donde empezar a imprimir el primer número. *contadorInterface* establece el color en $t9 y hace branch al número a imprimir según $t8.
14. turnosInitial y turnos:
    a. Parámetros: Registro $t4 para desplazarse en la pantalla, $t9 para guardar el color a imprimir, $t8 para saber cuál número se debe imprimir.
    b. Funcionalidad: *turnosInitital* establece el bit particular donde empezar a imprimir el primer número. *turnos* establece el color en $t9 y hace branch al número a imprimir según $t8.
15. evaluar:
    a. Parámetros: Registro $t5 que mantiene la cuenta de la cantidad de números que han sido impresos .
    b. Funcionalidad: Guarda en $t5 el valor de $t5 más 1, si $t5 es igual a 1, hace branch a *setContador*, si es igual a 2, hace branch a *separate2*, si es igual a 3, hace branch a *setTurnos*, si es igual a 4, regresa a *MANEJO_TURNOS*.
16. Números (0 a 9):
    a. Parámetros: Directivo .space ‘display’, registros $t4 y $t6 utilizados para desplazarse alrededor de la pantalla de Bitmap Display y $t9 para pintar la posición deseada.
    b. Funcionalidad: Imprimir en Bitmap Display el estado actual del contador y el número de turnos que lleva la partida. Todos saltan a *evaluar*.

Diagrama de Flujo: