Universidad del Valle de Guatemala
Organización de computadoras y assembler

Proyecto Assembler

Walter Saldaña 19897
Carlos Alberto Raxtum 19721
Eduardo Ramirez 19946

Primera Fase:

1.Cronograma de trabajo con tareas, objetivos significativos a lograr en cada etapa (hitos), responsable asignado de cada tarea

Objetivo Principal del Proyecto

Con los conceptos aprendidos en clase, el objetivo de este proyecto es diseñar un programa básico en lenguaje ensamblador del procesador ARM mixto con C que implemente el juego Word Zapper.

Objetivo a lograr en la primera fase:

Cómo principales objetivos de la fase 1 del proyecto son:

- Organización del grupo para la realzar el proyecto
- Definir lo que se va a estar trabajando por fase
- Definir responsabilidades por integrante
- Definir el algoritmo que se va a implementar incluyendo descripción de subrutinas y registros en el proyecto.

Nombre del Integrante	Tarea realizada
Carlos Alberto Ráxtum Ramos	Elaboración del diagrama de flujo Descripción del objetivo principal y primera fase, propuestas de subrutinas en el catálogo subrutina retirar
Eduardo Ramirez	Identificación de las subrutinas y funcionamiento 1. subrutina obtener elemento
Walter Danilo Saldaña	Identificación de las subrutinas y funcionamiento, utilización de los registros en las subrutinas identificadas. 1. Lógica general del juego 2. Subrutina sustituir

Sesiones realizadas mayo 2020

Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
3	**4**	**5**	**6**	**7**	**8** Organización general de grupo estableciendo la	**9** Distribución de tareas para la primera entrega

					forma de trabajar para la elaboración del proyecto	
10 Elaboración del diagrama de flujo	**11** Plantear y describir las subrutinas que tenemos pensadas para el proyecto	**12**	**13**	**14**	**15**	**16** Encabezado, descripción de secciones y comentarios (documentación)
17 Captura de datos	**18** Banner (preliminar, con o sin ASCII art)	**19**	**20**	**21** Revisar funcionalidad de las subrutinas	**22**	**23**
24	**25** Revisar funcionalidad de las subrutinas	**26**	**27**	**28** Elaboración del manual de usuario y conclusiones finales sobre el proyecto	**29** Realizar pruebas al proyecto para verificar si cumple con los requisitos	**30**

2. Descripción del uso de los registros en cada subrutina

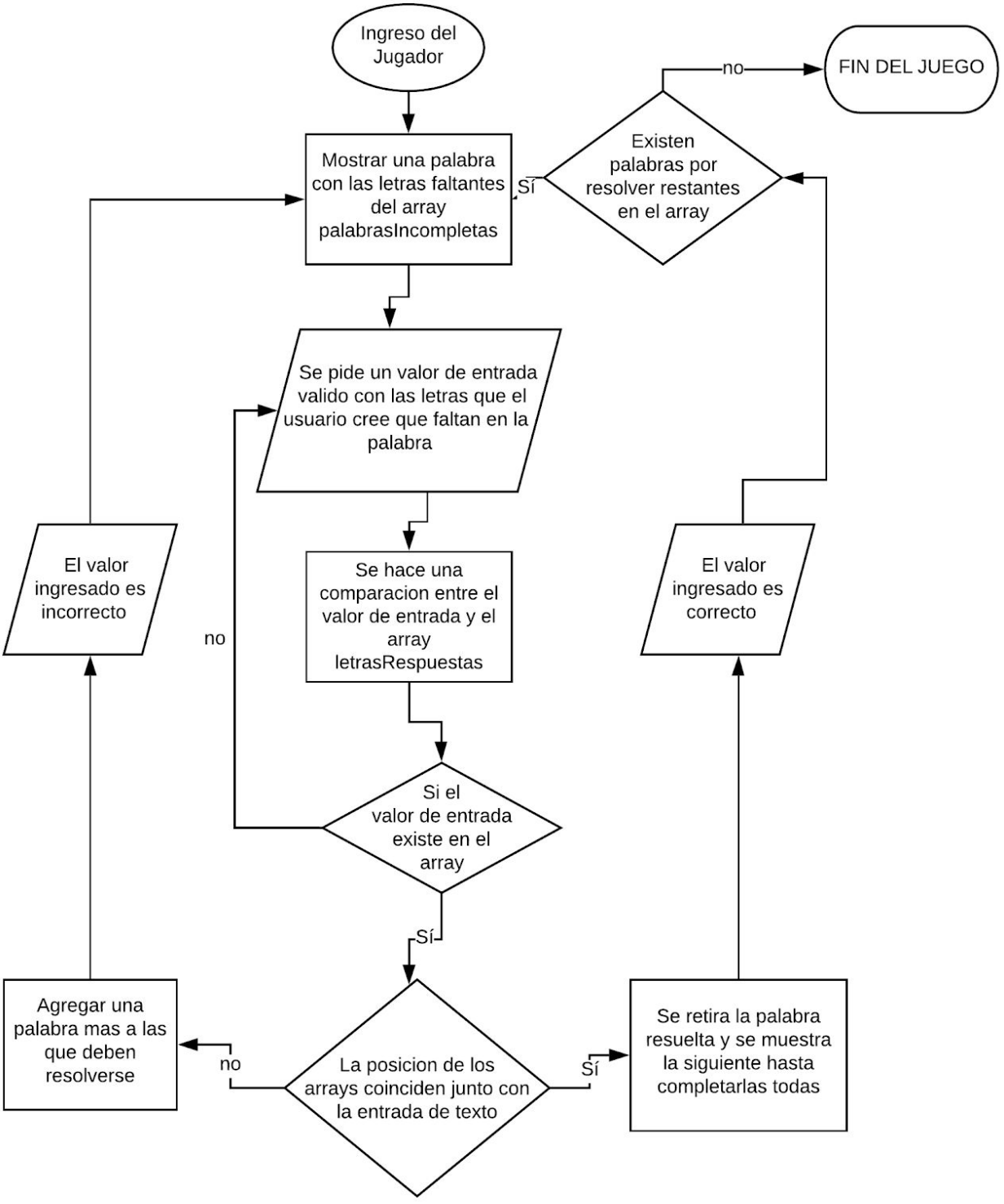
a. **obtener_elemento:**
   - Entradas:
     - R0: Dirección de memoria al array
     - R1: Posición del elemento en el array solicitado
     - R2: Cantidad de elementos del array solicitados.
   - Salida:

        - R0: Elemento en la posición seleccionada del array seleccionado concatenado a la n cantidad de elementos solicitados que le siguen.

b. **retirar:**
    - Entradas:
        - R0: Dirección de memoria del array
        - R1: Posición del elemento a retirar del array.
    - Salida:
        - R0: Valor del elemento en la posición seleccionada antes de ser retirado del arreglo.

c. **sustituir:**
    - Entradas:
        - R0: Dirección a memoria de la palabra a operar.
        - R1: Carácter que se quiere reemplazar.
        - R2 Carácter por el cual se quiere reemplazar.
        - R3: Número de caracteres de la palabra.
    - Salidas: No tiene salidas.

3. Diagrama de flujo: hecho en una herramienta de software, ordenado y con la simbología correcta.



4. Catálogo de funciones y subrutinas: nombre, entradas y salidas. Explicación de uso.

a. **obtener_elemento:** Manejo de arreglo con colección de datos separados uniformemente. Cada elemento del array debe de tener la misma longitud, si no se deben de llenar los espacios faltantes con "?" y se devolverá todo carácter diferente de este. Se usará para obtener elementos del array de palabras y de respuestas.
   - Entradas: R0, R1, R2
   - Salida: R0

   La entrada R0 determina cual es la dirección base del array. En R1 se indicará cual es la posición ordinaria del elemento en el array que se desea obtener empezando en 0. Y en R2 se indica cuántos elementos se quiere obtener. Luego, la rutina carga en R0 el valor del elemento del array en la posición seleccionada, concatenada a los siguientes valores el número de veces según se haya indicado en R2.

   Ejemplo de uso:

   Considere el arreglo:

   ARRAY .asciz "Hola??????", "Mensaje???", "De prueba?", "Subrutina?"

   Si se llama la subrutina de la siguiente forma:

   ldr r0, =ARRAY

   mov r1, #1

   mov r2, #2

   bl obtener_elemento

   Entonces devolverá en R0 el siguiente valor:

   "MensajeDe prueba"

b. **retirar:** Eliminar un elemento de un array y correr sus elementos para llenar el vacío. Los elementos del array deben de estar separados uniformemente.
   - Entradas: R0, R1
   - Salida: R0

   La entrada R0 está destinada para la dirección base del array. El registro R1 se utiliza para indicar la posición del elemento que se quiere retirar del arreglo. Por último la rutina elimina el elemento deseado pero antes devuelve el valor de ese elemento en R0.

   Ejemplo de uso:

Considere el arreglo:

ARRAY .asciz “Hola??????”, “Mensaje???”, “De prueba?”, “Subrutina?”

Si se llama la subrutina de la siguiente forma:

ldr r0, =ARRAY

mov r1, #1

bl retirar

Entonces devolverá en R0 el siguiente valor:

“Mensaje”

Y el array quedará de la siguiente manera

ARRAY .asciz “Hola??????”, “De prueba?”, “Subrutina?”

c. **sustituir:** Sustituye determinado carácter de una palabra en memoria por el que defina el usuario. Solo sustituirá el primero que encuentre.
   - Entradas: R0, R1, R2
   - Salidas: -

   La entrada R0 indica la dirección en memoria de la palabra a la cual se le quiere realizar la operación. R1 es donde se le indica cual es el carácter que se quiere reemplazar por el carácter que se guarda en R2. El resultado lo guarda directamente en la memoria de la palabra original (sobreescribe la palabra).

   Ejemplo de uso:

   Considere la palabra:

   MSG .asciz “Hola Mundo”

   Si se llama la subrutina de la siguiente forma:

   ldr r0, =MSG

   mov r1, #111 @ascii de “o”

   mov r2, #49\@ascii de “1”

   Entonces se almacenará en la dirección =MSG el siguiente contenido:

   “H1la Mund1”

5. Manual de Usuario:

1. El juego requiere de 1 jugador. Consiste en “destruir” las palabras que son enviadas por la computadora en forma secuencial.
2. Al iniciar el juego se le mostrará una palabra con letras faltantes, para llenar una palabra indique el número correspondiente de la misma que se encuentra en paréntesis y seguidamente (sin espacio) la letra faltante.
3. Si el jugador ingresa correctamente las letras faltantes de las palabras alcanzará el objetivo y tendrá menos palabras por resolver para ganar el juego.
4. Si no adivina correctamente las letras, las palabras irán aumentando y si no logra adivinar ninguna palabra de la lista y el acumulado es igual al total de palabras el juego se dará por perdido.

6. Conclusiones:

- Para la elaboración del proyecto fue fundamental aplicar todos los conceptos vistos en clase, de esa forma pudimos ser eficientes y prácticos para programar nuestro proyecto.
- El uso de los registros es muy delicado, ya que se pueden almacenar valores que no queremos en el espacio que deseamos por lo cual, hay que estar constantemente probando el programa para evitar errores más grandes.
- El uso de las subrutinas en nuestro programa fue esencial ya que estas realizan una tarea específica y es relativamente independiente del resto del código. Esto nos ayudó a tener un programa más organizado y a la vez lo hace más entendible. Otra de las ventajas fue que nos permite la reutilización del código para que este no fuera tan extenso.

7. Bibliografía:

1. Ferrari, Adam; Batson, Alan; Lack, Mike; Jones, Anita (2018-11-19) [Spring 2006]. Evans, David (ed.). *"x86 Assembly Guide*". Computer Science CS216: Program and Data Representation. Recuperado de: http://www.cs.virginia.edu/~evans/cs216/guides/x86.html
2. Norton, Peter; Socha, John (1986). *Peter Norton's Assembly Language Book for the IBM PC*. New York, USA: Brady Books.