MEMORIA

ESTRUCTURA DE COMPUTADORES

Robert Alex Nicolae Ristici c200106
Julia Serrano Arrese c200119
2021-2022

Identificador de grupo de alumnos: estru

ÍNDICE

ÍNDICE	2
INTRODUCCIÓN	3
ESTRUCTURA DEL PROYECTO	4
HISTORIAL DE DESARROLLO	5
JUEGO DE ENSAYO	7
OBSERVACIONES FINALES Y COMENTARIOS	10

INTRODUCCIÓN

El proyecto consiste en la programación de un conjunto de rutinas que realizan la compresión y descompresión de un texto definido en memoria mediante una versión simplificada de uno de los compresores sin pérdidas habituales, del tipo de zip, gzip.

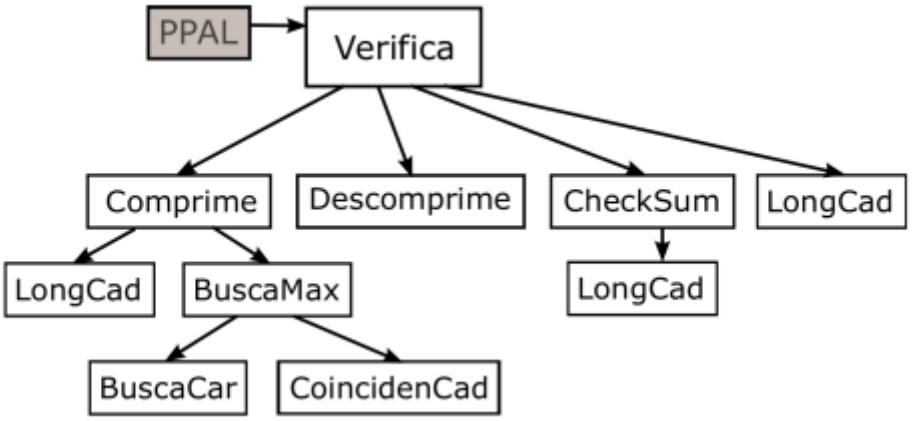
El algoritmo de compresión sin pérdida es aquel que tiene como objetivo representar cierta cantidad de información utilizando u ocupando un espacio menor, siendo posible una reconstrucción exacta de los datos originales.

El objetivo final del compresor será obtener un resultado idéntico tras el proceso de compresión-descompresión.

El texto a comprimir estará almacenado en memoria y no en un archivo, ya que el objetivo del proyecto es conocer las posibilidades del procesador y no el uso de periféricos.

El proyecto consta de una serie de subrutinas que permitirán aplicar una compresión y descompresión a un texto almacenado en memoria. La compresión del texto almacenado en una zona de memoria Z1 dejará su resultado en otra zona Z1c y la descompresión del contenido de la zona Z1c depositará el texto obtenido en una tercera zona, Z2, de tal modo que el contenido de Z1 y Z2 deberán ser idénticos

ESTRUCTURA DEL PROYECTO



El proyecto está compuesto por 8 subrutinas que se relacionan tal como se indica en la figura anterior.

Las subrutinas a desarrollar son:

1. LongCad (cadena)

La función devuelve la longitud de la cadena proporcionada en su único parámetro y sin incluir el carácter terminador 0x00.

2. BuscaCar(C, ref, from, to)

La función devuelve la distancia en caracteres desde el comienzo de la cadena ref hasta la posición en que se localiza el carácter C.

3. CoincidenCad (cadena1, cadena2)

   La función devuelve la longitud de la zona de caracteres coincidentes entre ambas cadenas.

4. BuscaMax(ref, max, jj)

La función devuelve la longitud del tramo coincidente más largo entre cualquier subcadena de ref que comience antes de Dir(ref[max]) y la subcadena que comienza precisamente en esa dirección Dir(ref[max]).

5. CheckSum(texto)

La función calcula la suma de las palabras de 32 bits que componen el texto proporcionado como parámetro, interpretando dicho texto como un vector de enteros sin signo de 32 bits y conservando únicamente los 32 bits menos significativos de esa suma

6. Comprime(texto, comprdo)

La función comprime el texto original que está almacenado a partir de texto y deja el resultado a partir de comprdo

7. Descomprime(com, desc)

La función realiza la descompresión del texto que está almacenado a partir de com (en el formato comprimido de este proyecto) y deja el resultado a partir de desc.

8. Verifica(texto, CheckSum1, CheckSum2)

La función se encarga de verificar que el funcionamiento de los procedimientos de compresión y descompresión es coherente

HISTORIAL DE DESARROLLO

Tras dedicar un tiempo al planteamiento del proyecto y la división del trabajo, el histórico del desarrollo de las rutinas se especifica a continuación:

- **08/12/2021**

Se ha comenzado y finalizado las rutinas del primer hito: LongCad, BuscaCar y CoincidenCad. No se han encontrado dificultades excesivamente preocupantes. Trabajo realizado por

Robert. Horas invertidas: 8.

- **09/12/2021**

Se han añadido pruebas adicionales para comprobar el correcto funcionamiento de las rutinas del primer hito.

Robert. Horas: 1.5.

- **10/12/2021-21/12/2021**

Nuevas pruebas del primer hito. Comprensión del funcionamiento de compresión de texto. Primeras líneas de código en la subrutina comprime.

Se ha realizado el esquema general de las subrutinas BuscaMax y CheckSum correspondientes al Hito 2, aunque todavía no se ha comprobado su funcionamiento.

Robert. Horas invertidas: 10.

Julia. Horas invertidas: 12

- **13/06/2022-15/06/2022**

Correcciones importantes en las subrutinas BuscaCar y BuscaMax (trabajo en la rutina BuscaMax en colaboración con mi compañera). BuscaCar se había complicado mucho, con pruebas innecesarias no indicadas por el enunciado, como comprobar si from es mayor que LongCad. Corrección de errores como en el uso del marco de pila, control del caso P+L > max y otros bugs. Nuevas pruebas más exhaustivas en BuscaCar.

Robert. Horas invertidas: 15. Comunicación con los profesores vía email: 4.

Julia. Horas invertidas: 12.

- **16/06/2022-20/06/2022**

Comprensión de los errores de alineamiento a palabra y media palabra, corrección con la instrucción extu. Desarrollo de los bucles Comprime y Descomprime. Dificultades encontradas en el funcionamiento del mapa de bits (tanto escribirlo como leerlo, utilizando instrucciones set y extu) y de guardar referencias a subcadenas previas (P y L). Dificultades por alineamiento de palabra. Nuevas pruebas en BuscaMax, Comprime y Descomprime. Realización de pruebas en la subrutina CheckSum, realizando los cambios adecuados para su correcto funcionamiento. Desarrollo de la subrutina Verifica y de sus pruebas. Una vez finalizadas las subrutinas pasamos de nuevo las pruebas codificadas y comprobamos que su comportamiento fuera el adecuado.

Robert. Horas invertidas: 40. Comunicación con los profesores vía email: 5.

Julia. Horas invertidas: 46. Comunicación con los profesores vía email: 4

- **Aproximación de horas totales invertidas:**

Robert: 75 horas.

Julia: 70 horas

JUEGO DE ENSAYO

A continuación se detallarán de forma resumida el conjunto de casos de prueba para cada rutina:

- **LongCad**

Primera prueba: longitud de cadena "caracteres arbitrarios\0" para comprobar el funcionamiento esperado en un caso modelo.

Segunda prueba: cadena vacía “\0”, comportamiento en casos especiales.

Tercera prueba: entero, comportamiento en casos especiales.

- **BuscaCar**

Se han realizado pruebas para comprobar el funcionamiento esperado en el caso ideal, cuando no se encuentra el carácter, cuando el carácter tiene un valor numérico mayor a 150 y cuando la cadena es larga

- **CoincidenCad**

Se han realizado pruebas para comprobar el funcionamiento esperado en el caso ideal, cuando las cadenas son idénticas y coinciden hasta el final, cuando una cadena es vacía y cuando las cadenas no coinciden.

- **BuscaMax**

Pruebas de funcionamiento en casos ideales, cuando no se encuentra subcadena coincidente, cuando coincide en solo un carácter.

- **Checksum**

Pruebas de funcionamiento con diferentes tipos de cadenas. Con cadenas múltiplo de 4, cadenas más largas, cadenas múltiplo de 4 +2 y con terminadores seguidos de más caracteres arbitrarios.

- **Comprime**

Se han realizado menos pruebas de las deseadas por falta de tiempo. Aún así, las pruebas realizadas funcionan de la manera esperada

- **Descomprime**

Se han realizado menos pruebas de las deseadas por falta de tiempo. Aún así, las pruebas realizadas funcionan de la manera esperada

- **Verifica.**

Se han realizado menos pruebas de las deseadas por falta de tiempo. Aún así, las pruebas realizadas funcionan de la manera esperada

OBSERVACIONES FINALES Y COMENTARIOS

Este proyecto ayuda mucho a la comprensión del lenguaje ensamblador y del funcionamiento de un procesador convencional de propósito general.

Lo que más hemos aprendido ha sido a la depuración de código y al desarrollo de pruebas para las subrutinas realizadas. Estos dos aspectos han sido muy diferenciales respecto a otras asignaturas ya que era algo que no habíamos trabajado tanto hasta el momento.

Ha sido un proyecto que ha necesitado muchas horas y esfuerzo. Han surgido muchos problemas a lo largo del desarrollo del proyecto pero poco a poco hemos logrado solventarlos.

Estamos muy agradecidos a los profesores encargados del proyecto ya que han logrado resolver todas nuestras dudas con gran rapidez.