22-23

Memoria Ensamblador

COMPRESOR DE VECTORES

GUILLERMO FRANCO GIMENO

Autores:

- Autoría única.
- Guillermo Franco Gimeno:
    - DNI: 51731214M
    - Correo: guillermo.franco@alumnos.upm.es
    - Nº matrícula: 210245
    - Grupo: 3S1M-PAII

Hito 1: LongCad y BuscaCar - 8 horas

Tiempo dedicado

- Lectura y comprensión del enunciado: 30 mins
- Instalar el emulador, aprender a usarlo, aprender a crear casos de prueba…: 3 horas
- LongCad: 2 horas
- BuscaCar: 1.30h
- Aprender cómo hacer la entrega del primer hito: 1 horas

Hito 2: CoincidenCad y BuscaMax – 6 horas

Tiempo dedicado

- Lectura y comprensión del enunciado: 30 mins
- CoincidenCad: 1 hora
- Entender algoritmo BuscaMax: 30 minutos
- BuscaMax: 4 horas

Solucionar errores en BuscaMax - 5.5 horas

Supuse que mis errores venían con el marco de pila ya que fallaban todos los test. Me di cuenta de que no sabía lo que era el marco de pila ni como implementarlo.

- Entender y aprender a implentar el marco de pila: 2 horas
- Implementar marco de pila en BuscaMax: 3 horas
- Solucionar errores menores en BuscaMax: 0.5 horas

Subrutina Comprime – 13.5 horas

Para evitar malgastar correcciones, empecé a hacer Comprime sin saber si BuscaMax pasaba las pruebas del DATSI y sin saber si mi implementación del marco de pila era adecuada.

- Intentar entender el algoritmo “a lo bruto”: 1.5 hora

Tras más de una hora me dí cuenta que no podía intentar convertir directamente el algoritmo a ensambaldor. Decidí escribir la subrutina en pseudocódigo, una decisión muy acertada.

- Elaborar pseucódigo: 2 horas

Una vez hecho el pseudocódigo, me puse a codificarlo en ensamblador. Iba a un buen ritmo hasta que me topé con el Mapa de Bits. Busqué en el manual y en ejemplos la mejor forma para escribir bit a bit. Llegué a la conclusión de que la mejor opción era utilizar las isntrucciones “set” y “clr” (si lo volviera a hacer creo que usaría otro método) así que en un programa aparte me puse a cacharrear con estas instrucciones para aprender a usarlas

- Aprender a usar “set” y “clr”: 1.30 horas

Continué implementando el pseudocódigo sin mucho problema

- Primera implementación del pseudocódigo: 6 horas

Para probar de forma más rápida el programa programé en Python un pequeño script que me ayudaba con la conversión de binario a hexadecimal, de hexadecimal a ASCII… de una forma muy rápida ya que no tenía que ir byte a byte, sino que podía convertir cadenas de bytes. Fue una pequeña inversión de tiempo que me ha ahorrado mucho tiempo.

- Elaboración de Scripts en Python: 0.5 horas

Con la ayuda de esta herramienta y con el pseudocódigo como referencia me puse a depurar la subrutina.

- Depuración de errores: 2 horas

Subrutina Descomprime – 5.5 horas

Con la experencia de Comprime, empecé elaborando el psudocódigo. Esta vez tarde menos que la anterior vez y en el pseudocódigo utilzaba instrucciones más sencillas que el pseudocódigo de Comprime para facilitar su conversión a ensamblador.

- Elaborar pseudocódigo: 1 hora

Para la lectura del mapa de bits utilicé otro método, más sencillo. Trabajaba con potencias y con la operación lógica “and” para leer los bits.

- Implementación de pseudocódigo: 3 horas

Tras la primera implementación, me puse a depurarlo con los casos de prueba.

- Depuración de errores: 1.5 horas

Terminé esta subrutina el jueves 15 de diembre cerca de la hora límite para la corrección. Todavía no había probado las pruebas del DATSI con la nueva versión de BuscaMax, la subrutina Comprime y la recién implementada Descomprime. Cuando intenté subir los archivos, la página del DATSI se cayó así que no pude comprobarlo.

Subrutina Verifica

Empecé la subrutina verifica el viernes 16, sin haber comprobar las subrutinas desde BuscaMax.