UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Proyecto - Laboratorio
Pregrado
2023-II

Indicaciones generales:

- El proyecto, requiere la implementación del paradigma de **Programación Orientada a Objetos**, es decir, debe diseñar el algoritmo con el **diseño adecuado de clases** que solucionen el problema.
- El proyecto se desarrollará en grupos de 3 estudiantes. Los **nombres** de los integrantes deberán aparecer al inicio del **informe de proyecto**
- Los archivos de proyecto se subirán directamente a www.gradescope.com (**Proyecto Laboratorio**)
- Solo un integrante del grupo entregará el proyecto. No olvide **registrar a los 3 integrantes de proyecto** en Gradescope. Los nombres aparecerán también en el informe.
- La entrega de proyecto será hasta el **miercoles 29 de noviembre**, a medianoche
- La presentación oral será el **jueves 30 de noviembre** y **viernes 01 de diciembre**
- La sección 2 describe **Indicaciones importantes para el proyecto**, y la sección 3, la **Rúbrica**

---

1 Modelo de Compilador

Desarrolle un modelo simple de compilador que traduzca un lenguaje de alto nivel como C++ a un **código ensamblador** utilizando Programación Orientada a Objetos
Por ejemplo, un código en **C++** como el siguiente:

```
while(true) {
cin>>a;
cin>>b;
c=a+b;
cout<<c;
}
```

se traduce a un **ensamblador** con las siguientes instrucciones:

```
in 100
in 101
load a, 100
load b, 101
add c,a,b
store c, 102
out 102
jmp 0
```

El procesador tiene 8 registros, con nombres a,b,c,d,e,f,g,h. Para lograr la traducción, el procesador conoce las siguientes instrucciones:

Instrucción	Ejemplo	Descripción
in<dirección de memoria>	in 100	lee un valor de entrada y lo almacena en la dirección de memoria 100
out<dirección de memoria>	out 100	muestra (en consola) el valor almacenado en la dirección de memoria 100
load<reg>,<dirección de memoria>	load a,100	lee el valor de la dirección de memoria 100 y lo almacena en el registro a
store<reg>,<dirección de memoria>	store a,100	almacena el valor del registro a en la direccion de memoria 100
mov<reg>,<reg>	mov a,c	copia el contenido del registro a al registro c
mvi<reg>,constante	mvi a,4	inicializa el registro a con una constante
add<reg>,<reg> ,<reg>	add c,a,b	suma el valor de los registros a,b y almacena el resultado en c
mul<reg>,<reg> ,<reg>	mul c,a,b	multiplica el valor del registro a, por el valor de b y almacena el resultado en c
inc<reg>	inc a	incrementa el contenido del registro a en 1
dec<reg>	inc a	decrementa el contenido del registro a en 1
jump<numero de instrucción>	jump 20	salta a la instrucción de la linea 20

Se solicita lo siguiente:

- Escriba un código simple en C++, distinto al del ejemplo, que pueda ser compilado (traducido) a ensamblador utilizando por lo menos 6 de las instrucciones dadas en la tabla, y **almacénelo en un archivo**.

- Desarrolle un código en C++, utilizando **clases y objetos**, que **lea el archivo** y lo compile (traduzca) a lenguaje ensamblador, utilizando las instrucciones de la tabla.
  Por ejemplo,
  - la línea de código **c=a;** se compila a
  mov a,c
  - la línea de código **cin>>a;** se compila a
  in 100
  load a, 100
  Aquí, 100 es el número que representa la dirección de memoria (un entero sin signo).
  Note que algunas instrucciones se compilan a más de una línea, por lo que el código ensamblador tiene generalmente más lineas que el código original

- No tiene que usar todas las instrucciones de la tabla, solo 6 de ellas. Es suficiente que el programa compile el código que contenga el archivo de entrada

- Escriba el código ensamblador obtenido **en otro archivo** (no en consola)

Opcional: (1 pt):
Agregue por lo menos 3 instrucciones de ensamblador a las usadas en el caso anterior y traduzca otro código de C++ a ensamblador.

2 Indicaciones importantes para la entrega del proyecto

2.1 Algoritmo y código

- **Diseñe adecuadamente el algoritmo**: defina las funciones necesarias para resolver el problema, que corresponda al diagrama UML presentado
- **Programe en C++** el código que solucione el problema planteado e imprima los resultados en forma ordenada y clara para el usuario
- **Optimice el uso de memoria** manejando la data en forma eficiente (utilice memoria dinámica donde sea necesario). El *main()* debe consistir solo en la declaración/inicialización de variables y las llamadas a funciones.

2.2 Entrega de proyecto

El proyecto se realizará en forma **grupal** (3 integrantes). En caso el grupo no tenga 3 integrantes, se deberá justificar debidamente
Se pide realizar un **Informe de Proyecto** con los siguientes puntos:

- **Nombre del proyecto**, de los **integrantes** y **porcentaje de participación** de cada integrante en el proyecto (de 0 a 100 %)
- **Introducción** (descripción del problema)
- **Método** (diseño del algoritmo, donde debe mostrar y describir el **diagrama UML**)
- **Resultados** (descripción de partes relevantes del código, y sus salidas)
- **Conclusiones** (resumen del trabajo y posibles mejoras)

El proyecto se subirá a Gradescope (Proyecto de Laboratorio)
Presente su proyecto en formato **.pdf**. Puede usar una plantilla de Plantillas Latex o convertir un documento a pdf. No olvide adjuntar el código de programación (en formato **.cpp**) y el archivo de entrada (en formato **.txt**) que contenga el código en C++ que será traducido (compilado). Puede comprimir todos los documentos en un **zip** y subirlos a Gradescope

3 Rúbrica

Se utilizará la siguiente rúbrica para la calificación del proyecto

Criterio	Logrado	Parcialmente Logrado	No Logrado
Diseño del algoritmo	El diseño del algoritmo es correcto y corresponde a la implementación en C++ (`5 pts`)	Existen algunos errores menores en el diseño del algoritmo (`3 pts`)	Existen muchos errores en el diseño del algoritmo(`0pts`).
Código (1)	El código no muestra errores, es ordenado y corresponde al algoritmo planteado (`7 pts`)	El código contiene errores menores de programación. (`3 pts`).	El código contiene múltiples errores de programación (`0pts`).
Presentación escrita (informe) y trabajo en grupo (1)	Describe los puntos del informe en forma ordenada y satisfactoria en un grupo de 3 integrantes (`5 pts`)	No describe en forma adecuada todos los puntos del informe(`3 pts`).	No describe en forma adecuada ningún punto del informe (`0pts`).
Presentación oral	Se presentó el proyecto en forma ordenada y clara (`3 pts`)	No se demostró un dominio total del contenido del proyecto durante la presentación (`1 pts`).	No se presentó conocimiento del proyecto durante la presentación (`0pts`).

(1) en este punto de la rúbrica se aplica una penalidad de (-0.5 pts) si se presenta el proyecto de a dos sin justificación , o (-1 pt) si se hace en forma individual