**Universidad del valle de Guatemala**
**Departamento de ciencias de la computación**
**Construcción de Compiladores**
**Msc. Ing. Bidkar Pojoy**

**ROBERTO ALEJANDRO CASTILLO DE LEON 18546**
**SEBASTIAN MALDONADO ARNAU 18003**


# Generación de código intermedio

**9 de Octubre de 2023**

# Retrospectiva de la Fase de Código Intermedio del Compilador YAPL

## 1. Introducción:

**Objetivo:** Reflexionar sobre el desarrollo y desafíos enfrentados en la fase de generación de código intermedio de nuestro compilador, permitiendo un ajuste de enfoques y técnicas conforme avancemos en el proceso.

## 2. Revisión de Objetivos:

**Cumplimientos:**
Se diseñó y construyó una representación intermedia apta para optimizaciones posteriores y la generación de código objetivo.

Logramos la integración exitosa con la fase anterior, tomando en cuenta el análisis semántico y la tabla de símbolos.

## 3. Inicio:

**Exploración de Representaciones:** Antes de comprometernos a una representación específica, debemos explorar múltiples opciones para garantizar la flexibilidad y eficiencia del código intermedio.

## 4. Detener:

**Rigidez en la Estructura:** Debemos ser cautelosos al establecer una estructura demasiado rígida para el código intermedio, ya que puede limitar oportunidades futuras de optimización.

## 5. Continuar:

**Comunicación Fluida:** Dada la complejidad creciente de las fases, es crucial mantener una comunicación transparente y constante entre los miembros del equipo.

**Uso de Herramientas:** La experiencia positiva con Git y JIRA debe mantenerse y adaptarse conforme las necesidades cambian en esta fase.

**Validación Continua:** A medida que generamos código intermedio, es esencial realizar pruebas y validaciones para garantizar su correcta traducción en fases posteriores.

## 6. Logros Principales:

**Representación Intermedia:** Logramos una representación intermedia eficiente, sentando las bases para optimizaciones y generación de código final.

**Integración:** La transición fluida desde el análisis semántico al código intermedio refleja la cohesión de nuestro enfoque y técnica.

**Automatización:** Establecimos procedimientos automatizados para la transformación de estructuras de alto nivel a representaciones intermedias.

## 7. Feedback sobre la Retrospectiva:

**¿Se sienten reflejados en esta retrospectiva?**

**Roberto Castillo:** La fase de código intermedio fue desafiante, pero siento que supimos adaptarnos y mantener la coherencia del proyecto. Tal vez podamos explorar más herramientas que faciliten esta fase en el futuro.

**Sebastián Maldonado:** Coincido con Roberto, y añadiría que debemos estar preparados para las optimizaciones. El código intermedio es fundamental para ello, y creo que hemos hecho un buen trabajo hasta ahora.

## 8. Acciones a seguir:

**Optimización:** Con una base sólida en el código intermedio, el siguiente paso es considerar las estrategias de optimización antes de la generación de código final.

**Integración:** Asegurar una transición sin problemas a la fase de generación de código, teniendo en cuenta todas las representaciones y estructuras previas.

**Documentación:** Dada la complejidad de esta fase, es vital mantener una documentación clara y actualizada para referencia futura y para el resto del equipo

## 9. Cierre:

La dedicación y esfuerzo conjunto en la fase de código intermedio han sido ejemplares. A medida que avanzamos, debemos continuar con esta sinergia y compromiso, abordando los desafíos futuros con la misma pasión y enfoque que hemos demostrado hasta ahora.

**Repositorio de Github:** https://github.com/sebas411/Compiler-Construction