EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE IZTAPALAPA
TECNM TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Proyecto:

Multi-Level Intermediate Representation Overview

Presenta:

KIMBBERLY NICOLE CLAVEL VERGARA
DILAN EDUARDO DIEZ BONILLA GUERRERO
EMMANUEL ALEJANDRO GONZÁLEZ RIVERA
MARIO IVÁN LÓPEZ TELLO RODRÍGUEZ

No. De control:

171080181
171080167
171080089
171080102

Profesor:

Parra Hernández Abiel Tomas

CIUDAD DE MÉXICO **ENERO / 2021.**

RESUMEN GENERAL

El proyecto MLIRO se basa en un enfoque novedoso para construir una infraestructura de compilación extensible y reutilizable. Así mismo, MLIRO cumple con muchos objetivos entre los cuales podemos encontrar los siguientes como los más importantes: abordar la fragmentación del software, esto permitirá que el sistema operativo que utilicemos, de alguna manera, pueda almacenar los datos de una forma mucho más rápida, también logra mejorar la compilación para hardware heterogéneo, esto, como ya sabemos ayuda a mejorar en gran medida el rendimiento de nuestro dispositivo por el hecho de añadir procesadores de diversos tipos. También ayuda a reducir significativamente los costos para crear compiladores específicos de dominio y de esta manera ayudar a conectar compiladores existentes.

Es importante hacer mención que el proyecto tiene además como objetivo extra el poder definir de manera clara y concisa una representación intermedia común que unirá la infraestructura necesaria para poder así ejecutar los modelos de aprendizaje automático de alto rendimiento en tensorflow, el cual consiste en una biblioteca de software de código abierto de computación numérica misma que utiliza gráficos de flujo de datos para analizar el movimiento de los datos dentro de un sistema determinado.

Por otra parte, dentro del proyecto, se incluye la capacidad de representar todos los gráficos de tensorflow, incluidas las formas dinámicas, el sistema operativo extensible por el usuario, variables, etc.

MLIR también admite operaciones específicas de hardware siendo estas las lógicas y las aritméticas, pero sin admitir algoritmos de generación de código de máquina de bajo nivel (como la asignación de registros y la programación de instrucciones).

CRONOGRAMA

ACTIVIDAD		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RECOPILAR INFORMACIÓN DEL TEMA	P																
	R																
DEPURAR INFORMACIÓN	P																
	R																
INSTALAR COMPLEMENTOS	P																
	R																
RIESGOS Y SOLUCIONES PROPUESTAS	P																
	R																
COMPROBACIÓN DEL PROYECTO	P																
	R																
DOCUMENTACIÓN DE RESULTADOS	P																
	R																
CONCLUSIONES	P																
	R																
OBSERVACIONES:																	

DESCRIPCIÓN GENERAL DE LA REPRESENTACIÓN INTERMEDIA DE VARIOS NIVELES

MLIR está diseñado para ser un IR híbrido que puede soportar múltiples requisitos diferentes en una infraestructura unificada. Por ejemplo, esto incluye:

La capacidad de representar gráficos de flujo de datos (como en TensorFlow), incluidas las formas dinámicas, el ecosistema operativo extensible por el usuario, las variables de TensorFlow, etc.

Las optimizaciones y transformaciones se realizan normalmente en tales gráficos (por ejemplo, en Grappler).

Representación de kernels para operaciones ML en una forma adecuada para la optimización.

Capacidad para albergar optimizaciones de bucle de estilo informático de alto rendimiento en los núcleos (fusión, intercambio de bucle, ordenamiento en teselas, etc.) y para transformar los diseños de memoria de los datos.

Transformaciones de "reducción" de generación de código, como inserción DMA, gestión explícita de caché, ordenamiento en teselas de memoria y vectorización para arquitecturas de registro 1D y 2D.

Capacidad para representar operaciones específicas de un objetivo, por ejemplo, operaciones de alto nivel específicas de un acelerador.

Cuantización y otras transformaciones de gráficos realizadas en un gráfico de aprendizaje profundo.

MLIR es un IR común que también admite operaciones específicas de hardware. Por lo tanto, cualquier inversión en la infraestructura que rodea a MLIR (por ejemplo, el compilador transmite ese trabajo) debería generar buenos rendimientos; muchos objetivos pueden utilizar esa infraestructura y se beneficiarán de ella.

MLIR es una representación poderosa, pero tampoco tiene objetivos. No intentamos admitir algoritmos de generación de código de máquina de bajo nivel (como la asignación de registros y la programación de instrucciones). Se adaptan mejor a optimizadores de nivel inferior (como LLVM). Además, no pretendemos que MLIR sea un lenguaje fuente en el que los propios usuarios finales escribirían núcleos (análogo a CUDA C ++). Por otro lado, MLIR proporciona la columna vertebral para representar cualquier DSL e integrarlo en el ecosistema.

INFRAESTRUCTURA DEL COMPILADOR

MLIR, o Representación intermedia multinivel, es un formato de representación y una biblioteca de utilidades del compilador que se encuentra entre la representación del modelo y los compiladores/ejecutores de bajo nivel que generan código específico del hardware.

MLIR es, en esencia, una infraestructura flexible para los compiladores optimizadores modernos. Esto significa que consta de una especificación para representaciones intermedias (IR) y un juego de herramientas de código para realizar transformaciones en esa representación. (En el lenguaje del compilador, a medida que pasa de representaciones de nivel superior a representaciones de nivel inferior, estas transformaciones se pueden llamar "reducciones")

Nos beneficiamos de la experiencia adquirida en la construcción de otros IR (LLVM IR, XLA HLO y Swift SIL) al construir MLIR. El marco MLIR fomenta las mejores prácticas existentes, por ejemplo, escribir y mantener una especificación de IR, construir un verificador de IR, brindar la capacidad de volcar y analizar archivos MLIR en texto, escribir pruebas unitarias extensas con la herramienta FileCheck y construir la infraestructura como un conjunto de bibliotecas modulares que se pueden combinar de nuevas formas.

Otras lecciones se han incorporado e integrado en el diseño de manera sutil. Por ejemplo, LLVM tiene errores de diseño no obvios que impiden que un compilador multiproceso funcione en múltiples funciones en un módulo LLVM al mismo tiempo. MLIR resuelve estos problemas al tener un alcance SSA limitado para reducir las cadenas use-def y al reemplazar referencias de función cruzada con explícitas symbol reference.

Cabe mencionar que MLIR tiene un sistema de tipos flexible, y permite representar, analizar y transformar grafos combinando múltiples niveles de abstracción en una misma unidad de compilación. Estas abstracciones incluyen operaciones de TensorFlow, así como regiones de bucle poliédrico anidadas e incluso instrucciones LLVM y operaciones y tipos de hardware fijo.

ANÁLISIS DE RIESGOS

RIESGOS	SOLUCIONES
No contar con el material o requisitos necesario para la realización del proyecto	Rentar el material que se necesita
No encontrar demasiada información sobre el tema en internet	Buscar información en libros y revistas sobre el tema
No comprender alguna parte del tema	Buscar información en diferentes fuentes para poder comprender el tema
Falta de conocimiento en cuestiones de programación	Conseguir apoyo de personas con los conocimientos adecuados para la realización del proyecto
Tiempos ajustados y aglomeración de actividades a realizar	Crear un plan de trabajo para organizar las actividades del proyecto

JUSTIFICACIÓN

A lo largo de la elaboración de este proyecto el equipo se planteó el propósito de integrar a este documento la mayor cantidad de información posible y disponible, asegurándonos de que esta perteneciera a una fuente de consulta verídica y que no presentará algún tipo de vaguedad que pudiese entorpecer nuestros resultados obtenidos.

Así mismo, nos centraremos en abordar a detalle todos y cada uno de los factores que intervienen o conforman la estructura del Multi-Level Intermediate Representation Overview.

La realización de este proyecto tiene como fin principal el poder descubrir, conocer y sobre todo interpretar a detalle el correcto funcionamiento del Multi-Level Intermediate Representation Overview, dado que una de sus principales funciones es brindar un tipo de ayuda para los compiladores, así como para los usuarios o personas que están dedicadas a la elaboración de dichos compiladores.

Así mismo, resulta fundamental hacer mención que el Multi-Level Intermediate Representation Overview es el encargado de realizar la traducción de un lenguaje de programación a un lenguaje máquina, siendo este uno de los principales factores del porque es de vital estudiar esta parte de los compiladores. En términos generales podríamos decir que esta es sin duda la parte con mayor importancia del compilador.

Además, buscaremos abordar las múltiples implementaciones que tiene tanto de compilación como de representación, mismas que ayudarán a mejorar de manera significativa el rendimiento, pero que a su vez causará algunas complicaciones como pueden ser la producción de mensajes de error confusos entre los sistemas.

A grandes rasgos, lo que queremos lograr con todo lo anteriormente mencionado, es permitir exploraciones novedosas para así poder optimizar el diseño y la implementación del compilador, respaldados por componentes de calidad de producción, así como crear un sentimiento de interés dentro de todas aquellas personas dedicadas o próximas a dedicarse a la elaboración de compiladores que buscan optimizar el rendimiento y sobre todo el consumo de memoria de los modelos de aprendizaje automático, al igual que aquellos que desean centrar su atención en aprovechar la optimización de los compiladores y la aceleración del hardware.

REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

FUNCIONALES

- El programa explica paso a paso cada una de las operaciones que se están realizando, de tal forma que el usuario comprenda cada una de ellas durante la ejecución del programa.
- Que el código del programa se encuentre libre de errores.
- El programa cumple la función de una calculadora para resolver ecuaciones cuadráticas.

NO FUNCIONALES

- Que el programa sea de fácil manejo para el usuario.
- Que el programa utilice una mínima cantidad de recursos.
- Que la velocidad de ejecución del programa sea rápida.
- La interfaz del programa se muestre en orden y sea agradable a la vista del usuario final.

PLANTEAMIENTO

Conocer el funcionamiento del Multi-Level Intermediate Representation Overview es una de las cosas más interesantes que debemos aprender con respecto al funcionamiento y las partes que conforman un compilador, ya que es aquí en donde se realiza la traducción de un lenguaje de programación de alto nivel a un lenguaje de bajo nivel o lenguaje máquina para poder interactuar con la misma directamente.

Para poder comprender a fondo el proceso de lo anteriormente mencionado, se realizará un proyecto referido al Multi-Level Intermediate Representation Overview, de esta manera podremos conocer cuál es el funcionamiento correcto que tendrá el compilador, también podemos decir que la realización de este trabajo nos permitirá reconocer su estructura.

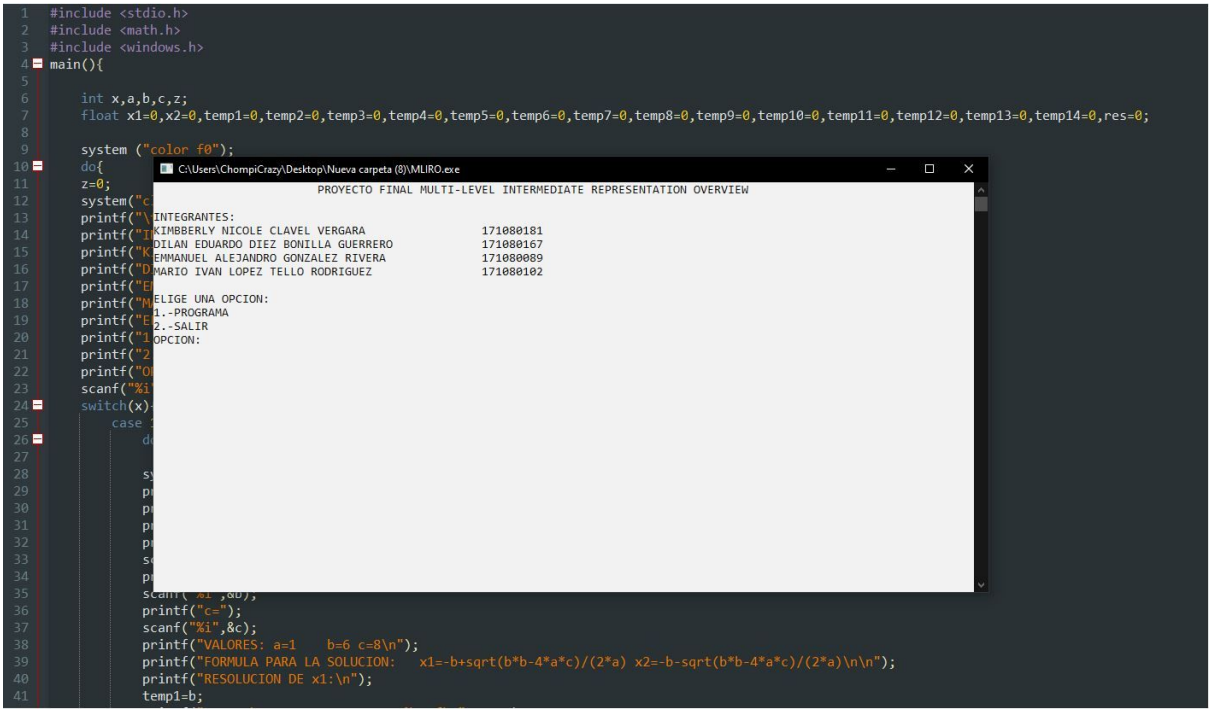
Además, explicaremos paso a paso la estructura interna que lo compone, para también conocer de manera detallada su funcionamiento dentro del compilador, mostrando la capacidad de representar gráficos de flujo de datos (hablamos del tensor Flow, mismo que fue explicado un poco más a detalle en la justificación de este documento), incluyendo también las formas dinámicas, el ecosistema operativo, la capacidad que tendrá de albergar optimizaciones de bucle que sean

de un estilo igual al informático, el cual, como ya sabemos, proporciona un alto rendimiento en los núcleos.

Es necesario realizar este proyecto para darnos cuenta de la necesidad de diseñar generadores de código, optimizadores y traductores ya que los marcos de aprendizaje automático de hoy en día tienen diferentes tiempos de ejecución, compiladores y tecnologías de gráficos.

En términos generales, son gran cantidad de puntos los que se tienen y se pretender abarcar, puesto que ninguno es menos importante que otro y al estudiarlos y comprenderlos podremos sacar nuestras propias conclusiones de manera clara y concisa de tal forma que se determine la ayuda o el apoyo que brindan principalmente a las tareas para las cuales fue desarrollada esta parte dentro de los compiladores.

DISEÑO Y DESARROLLO



```c
#include <stdio.h>
#include <math.h>
#include <windows.h>
main(){

    int x,a,b,c,z;
    float x1=0,x2=0,temp1=0,temp2=0,temp3=0,temp4=0,temp5=0,temp6=0,temp7=0,temp8=0,temp9=0,temp10=0,temp11=0,temp12=0,temp13=0,temp14=0,res=0;

    system ("color f0");
    do{
    z=0;
    system("c
    printf("\
    printf("I
    printf("K
    printf("D
    printf("E
    printf("M
    printf("E
    printf("1
    printf("2
    printf("O
    scanf("%i
    switch(x)
        case
            d
            S
            p
            p
            p
            p
            s
            p
            scanf("%i",&b);
            printf("c=");
            scanf("%i",&c);
            printf("VALORES: a=1    b=6 c=8\n");
            printf("FORMULA PARA LA SOLUCION:   x1=-b+sqrt(b*b-4*a*c)/(2*a) x2=-b-sqrt(b*b-4*a*c)/(2*a)\n\n");
            printf("RESOLUCION DE x1:\n");
            temp1=b;
```

C:\Users\ChompiCrazy\Desktop\Nueva carpeta (8)\MLIRO.exe

```
temp11=temp10+temp9             temp11=-4.00
temp12=2                        temp12=2.00
temp13=temp12*temp3             temp13=2.00
temp14=temp11/temp13            temp14=-2.00
res=temp14                      res=-2.00
res=-2.00

RESOLUCION DE x2:
temp1=b                         temp1=6.00
temp2=temp1*temp1               temp2=36.00
temp3=a                         temp3=1.00
temp4=c                         temp4=8.00
temp5=temp3*temp4               temp5=8.00
temp6=4                         temp6=4.00
temp7=temp6*temp5               temp7=32.00
temp8=temp2*temp7               temp8=4.00
temp9=sqrt(temp8)               temp9=2.00
temp10=-1*temp1                 temp10=-6.00
temp11=temp10+temp9             temp11=-8.00
temp12=2                        temp12=2.00
temp13=temp12*temp3             temp13=2.00
temp14=temp11/temp13            temp14=-4.00
res=temp14                      res=-4.00
res=-4.00

ELIGE UNA OPCION:
1.-UTILIZAR OTROS VALORES
2.-REGRESAR AL MENU
3.-SALIR
OPCION:
```

FUENTES DE INFORMACIÓN

Hugo. (2012). Marco del compilador de infrarrojos multinivel. nov 25, 2020, de MLIR Sitio web: https://mlir.llvm.org/

@pypyproject. (2019). Compilador, Generación de código, Lenguaje de programación, C, Java, Programa de computadora. nov 27 2020, de Essentials Sitio web: https://essentials.news/ai/research/article/multi-level-intermediate-representation-overview-42b8bb3a5e

joker-eph. (2019). MLIR is now part of LLVM, avoid any mistake by removing the code and a README pointing to the new location. dic 1 2020, de GitHub Sitio web: https://github.com/tensorflow/mlir

stellaraccident. (2012). [mlir][CAPI] Add result type inference to the CAPI.. dic 4 2020, de GitHub Sitio web: https://github.com/llvm/llvm-project/tree/master/mlir/

Ramón Chávez González. (2013). Teoría de Lenguajes y Compiladores. dic 8 2020, de sites.google.com Sitio web: https://sites.google.com/site/teoriadelenguajesycompiladores/procesadores-de-lenguaje/generacion-de-codigo

www.frro.utn.edu.ar/. (2015). SINTAXIS. dic 11 2020, de SSyL Sitio web: https://www.frro.utn.edu.ar/repositorio/catedras/sistemas/2_anio/sintaxis/SSyL-cap1_2015_Introduccion.pdf

Aho, A.V., Sethi, R., Ullman, J.D. (1990), Compiladores: principios, técnicas y herramientas, Tema 8, 9, 10 (pag. 478- 666).

Louden, K.C. (1997), Compiler Construction: Principles and Practice, Tema 8, páginas: 398-481.

Ing. M.A. Sheyla Esquivel. (2020). Compiladores - Código Intermedio. dic 15 2020, de Youtube Sitio web: https://www.youtube.com/watch?v=Yj0cOCcXR6I