Compiler Design BCSE307L

Digital Assignment:3

Implementation of ISA code generation using LLVM

VIT®
CHENNAI
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Under the guidance of DR. MANJU G [Associate Professor]

Team members

22BRS1195 Niranjan P
22BRS1100 Adithya PR
22BRS1120 Kamaleswar S

IMPLEMENTATION OF ISA CODE GENERATION USING LLVM

Step 1: Update Your System

sudo apt update && sudo apt upgrade -y

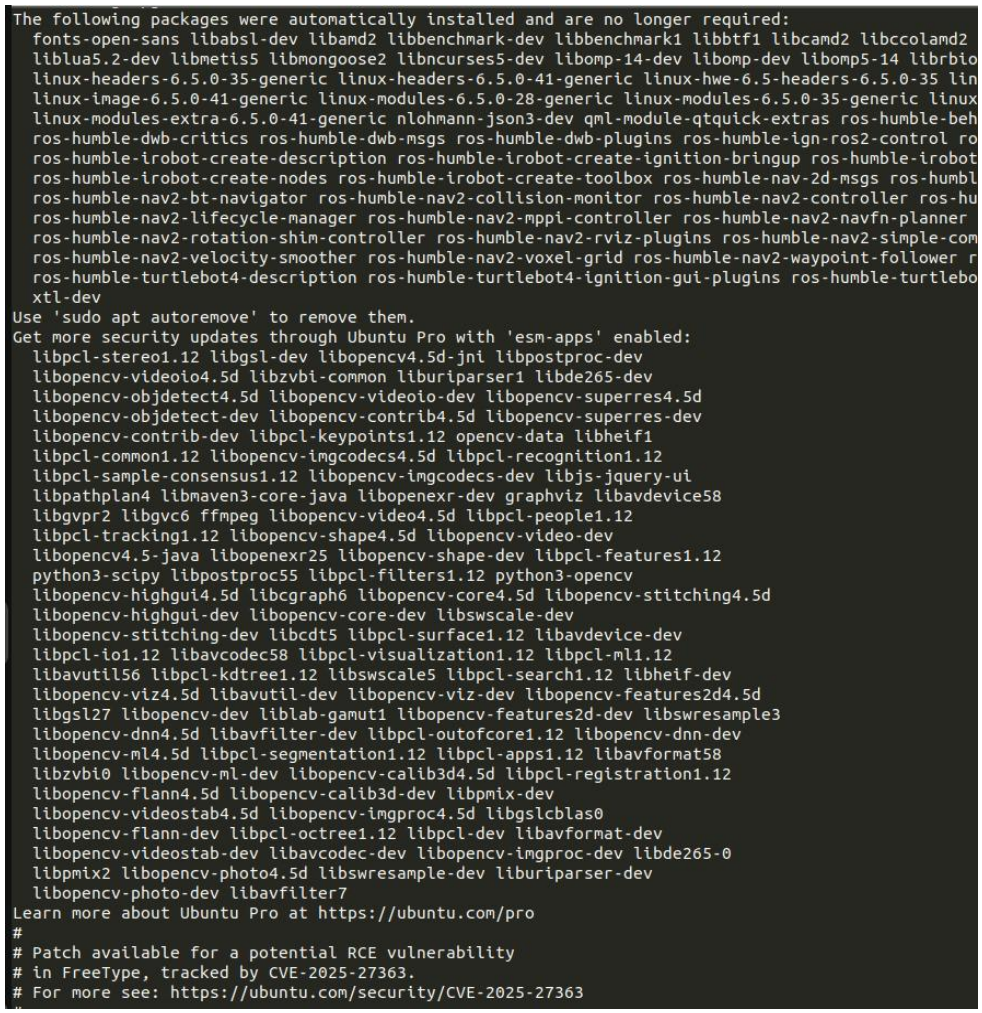
This updates all installed packages and dependencies.

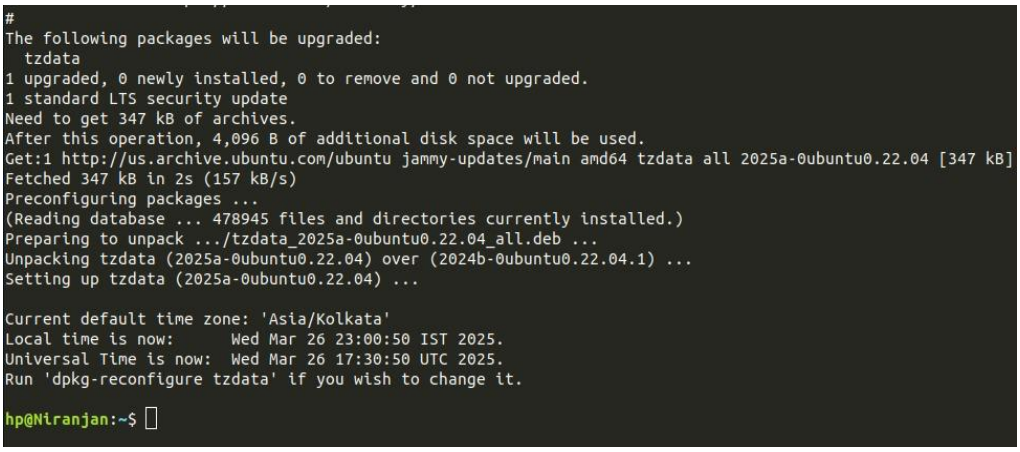
sudo apt update: Refreshes the package lists.

sudo apt upgrade -y: Installs available updates for all packages.

This step is crucial to ensure compatibility when installing LLVM and other required tools.

```
hp@Niranjan:~$ sudo apt update && sudo apt upgrade -y
Hit:1 https://packages.microsoft.com/repos/code stable InRelease
Hit:2 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://packages.ros.org/ros2/ubuntu jammy InRelease
Hit:4 https://librealsense.intel.com/Debian/apt-repo jammy InRelease
Hit:5 http://packages.osrfoundation.org/gazebo/ubuntu-stable jammy InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:8 https://hub-dist.unity3d.com/artifactory/hub-debian-prod-local stable InRelease
Hit:10 https://linux.teamviewer.com/deb stable InRelease
Hit:11 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:12 http://us.archive.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,422 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [777 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [400 kB]
Get:16 http://us.archive.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,178 kB]
Get:17 http://us.archive.ubuntu.com/ubuntu jammy-security/main i386 Packages [608 kB]
Get:18 http://us.archive.ubuntu.com/ubuntu jammy-security/main Translation-en [337 kB]
Get:19 http://us.archive.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [3,053 kB]
Get:20 http://us.archive.ubuntu.com/ubuntu jammy-security/restricted Translation-en [541 kB]
Get:21 http://us.archive.ubuntu.com/ubuntu jammy-security/universe i386 Packages [654 kB]
Get:22 http://us.archive.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [969 kB]
Get:23 http://us.archive.ubuntu.com/ubuntu jammy-security/universe Translation-en [208 kB]
Fetched 12.4 MB in 8s (1,651 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
```


The following packages were automatically installed and are no longer required:
fonts-open-sans libabsl-dev libamd2 libbenchmark-dev libbenchmark1 libbtf1 libcamd2 libccolamd2
liblua5.2-dev libmetis5 libmongoose2 libncurses5-dev libomp-14-dev libomp-dev libomp5-14 librbio
linux-headers-6.5.0-35-generic linux-headers-6.5.0-41-generic linux-hwe-6.5-headers-6.5.0-35 lin
linux-image-6.5.0-41-generic linux-modules-6.5.0-28-generic linux-modules-6.5.0-35-generic linux
linux-modules-extra-6.5.0-41-generic nlohmann-json3-dev qml-module-qtquick-extras ros-humble-beh
ros-humble-dwb-critics ros-humble-dwb-msgs ros-humble-dwb-plugins ros-humble-ign-ros2-control ro
ros-humble-irobot-create-description ros-humble-irobot-create-ignition-bringup ros-humble-irobot
ros-humble-irobot-create-nodes ros-humble-irobot-create-toolbox ros-humble-nav-2d-msgs ros-humbl
ros-humble-nav2-bt-navigator ros-humble-nav2-collision-monitor ros-humble-nav2-controller ros-hu
ros-humble-nav2-lifecycle-manager ros-humble-nav2-mppi-controller ros-humble-nav2-navfn-planner
ros-humble-nav2-rotation-shim-controller ros-humble-nav2-rviz-plugins ros-humble-nav2-simple-com
ros-humble-nav2-velocity-smoother ros-humble-nav2-voxel-grid ros-humble-nav2-waypoint-follower r
ros-humble-turtlebot4-description ros-humble-turtlebot4-ignition-gui-plugins ros-humble-turtlebo
xtl-dev
Use 'sudo apt autoremove' to remove them.
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
libpcl-stereo1.12 libgsl-dev libopencv4.5d-jni libpostproc-dev
libopencv-videoio4.5d libzvbi-common liburiparser1 libde265-dev
libopencv-objdetect4.5d libopencv-videoio-dev libopencv-superres4.5d
libopencv-objdetect-dev libopencv-contrib4.5d libopencv-superres-dev
libopencv-contrib-dev libpcl-keypoints1.12 opencv-data libheif1
libpcl-common1.12 libopencv-imgcodecs4.5d libpcl-recognition1.12
libpcl-sample-consensus1.12 libopencv-imgcodecs-dev libjs-jquery-ui
libpathplan4 libmaven3-core-java libopenexr-dev graphviz libavdevice58
libgvpr2 libgvc6 ffmpeg libopencv-video4.5d libpcl-people1.12
libpcl-tracking1.12 libopencv-shape4.5d libopencv-video-dev
libopencv4.5-java libopenexr25 libopencv-shape-dev libpcl-features1.12
python3-scipy libpostproc55 libpcl-filters1.12 python3-opencv
libopencv-highgui4.5d libcgraph6 libopencv-core4.5d libopencv-stitching4.5d
libopencv-highgui-dev libopencv-core-dev libswscale-dev
libopencv-stitching-dev libcdt5 libpcl-surface1.12 libavdevice-dev
libpcl-io1.12 libavcodec58 libpcl-visualization1.12 libpcl-ml1.12
libavutil56 libpcl-kdtree1.12 libswscale5 libpcl-search1.12 libheif-dev
libopencv-viz4.5d libavutil-dev libopencv-viz-dev libopencv-features2d4.5d
libgsl27 libopencv-dev liblab-gamut1 libopencv-features2d-dev libswresample3
libopencv-dnn4.5d libavfilter-dev libpcl-outofcore1.12 libopencv-dnn-dev
libopencv-ml4.5d libpcl-segmentation1.12 libpcl-apps1.12 libavformat58
libzvbi0 libopencv-ml-dev libopencv-calib3d4.5d libpcl-registration1.12
libopencv-flann4.5d libopencv-calib3d-dev libpmix-dev
libopencv-videostab4.5d libopencv-imgproc4.5d libgslcblas0
libopencv-flann-dev libpcl-octree1.12 libpcl-dev libavformat-dev
libopencv-videostab-dev libavcodec-dev libopencv-imgproc-dev libde265-0
libpmix2 libopencv-photo4.5d libswresample-dev liburiparser-dev
libopencv-photo-dev libavfilter7
Learn more about Ubuntu Pro at https://ubuntu.com/pro
#
Patch available for a potential RCE vulnerability
in FreeType, tracked by CVE-2025-27363.
For more see: https://ubuntu.com/security/CVE-2025-27363


#
The following packages will be upgraded:
tzdata
1 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
1 standard LTS security update
Need to get 347 kB of archives.
After this operation, 4,096 B of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 tzdata all 2025a-0ubuntu0.22.04 [347 kB]
Fetched 347 kB in 2s (157 kB/s)
Preconfiguring packages ...
(Reading database ... 478945 files and directories currently installed.)
Preparing to unpack .../tzdata_2025a-0ubuntu0.22.04_all.deb ...
Unpacking tzdata (2025a-0ubuntu0.22.04) over (2024b-0ubuntu0.22.04.1) ...
Setting up tzdata (2025a-0ubuntu0.22.04) ...
Current default time zone: 'Asia/Kolkata'
Local time is now: Wed Mar 26 23:00:50 IST 2025.
Universal Time is now: Wed Mar 26 17:30:50 UTC 2025.
Run 'dpkg-reconfigure tzdata' if you wish to change it.
hp@Niranjan:~$

Step 2: Install LLVM and Clang

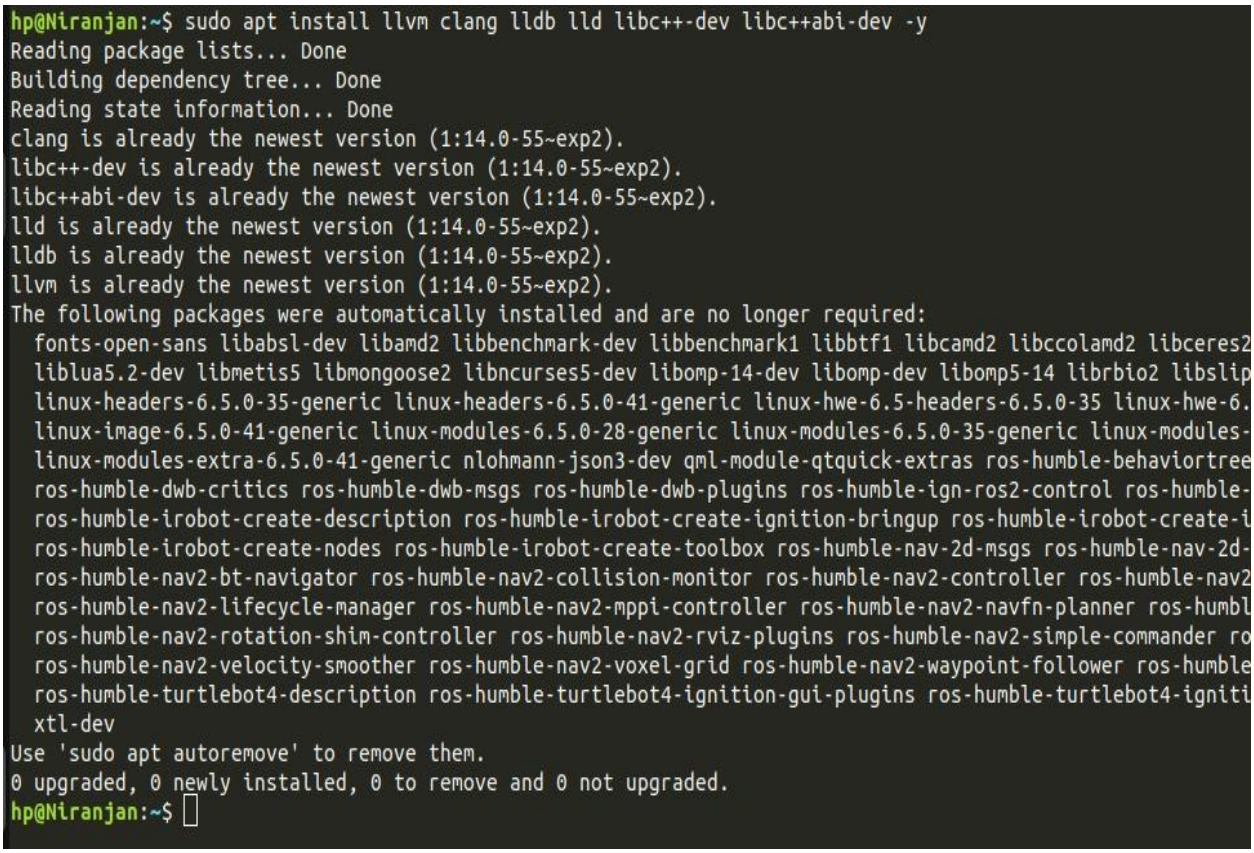
To begin, create a new C++ file that will be compiled into LLVM IR

Code:

nano program.cpp

nano: Opens a simple text editor.

•program.cpp: The filename for our C++ source code.



```
hp@Niranjan:~$ sudo apt install llvm clang lldb lld libc++-dev libc++abi-dev -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
clang is already the newest version (1:14.0-55~exp2).
libc++-dev is already the newest version (1:14.0-55~exp2).
libc++abi-dev is already the newest version (1:14.0-55~exp2).
lld is already the newest version (1:14.0-55~exp2).
lldb is already the newest version (1:14.0-55~exp2).
llvm is already the newest version (1:14.0-55~exp2).
The following packages were automatically installed and are no longer required:
  fonts-open-sans libabsl-dev libamd2 libbenchmark-dev libbenchmark1 libbtf1 libcamd2 libccolamd2 libceres2
  liblua5.2-dev libmetis5 libmongoose2 libncurses5-dev libomp-14-dev libomp-dev libomp5-14 librbio2 libslip
  linux-headers-6.5.0-35-generic linux-headers-6.5.0-41-generic linux-hwe-6.5-headers-6.5.0-35 linux-hwe-6.
  linux-image-6.5.0-41-generic linux-modules-6.5.0-28-generic linux-modules-6.5.0-35-generic linux-modules-
  linux-modules-extra-6.5.0-41-generic nlohmann-json3-dev qml-module-qtquick-extras ros-humble-behaviortree
  ros-humble-dwb-critics ros-humble-dwb-msgs ros-humble-dwb-plugins ros-humble-ign-ros2-control ros-humble-
  ros-humble-irobot-create-description ros-humble-irobot-create-ignition-bringup ros-humble-irobot-create-i
  ros-humble-irobot-create-nodes ros-humble-irobot-create-toolbox ros-humble-nav-2d-msgs ros-humble-nav-2d-
  ros-humble-nav2-bt-navigator ros-humble-nav2-collision-monitor ros-humble-nav2-controller ros-humble-nav2
  ros-humble-nav2-lifecycle-manager ros-humble-nav2-mppi-controller ros-humble-nav2-navfn-planner ros-humbl
  ros-humble-nav2-rotation-shim-controller ros-humble-nav2-rviz-plugins ros-humble-nav2-simple-commander ro
  ros-humble-nav2-velocity-smoother ros-humble-nav2-voxel-grid ros-humble-nav2-waypoint-follower ros-humble
  ros-humble-turtlebot4-description ros-humble-turtlebot4-ignition-gui-plugins ros-humble-turtlebot4-igniti
  xtl-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
hp@Niranjan:~$
```

LLVM Configuration version

Clang Version

```
hp@Niranjan:~$ llvm-config --version
14.0.0
hp@Niranjan:~$ clang --version
Ubuntu clang version 14.0.0-1ubuntu1.1
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
hp@Niranjan:~$
```

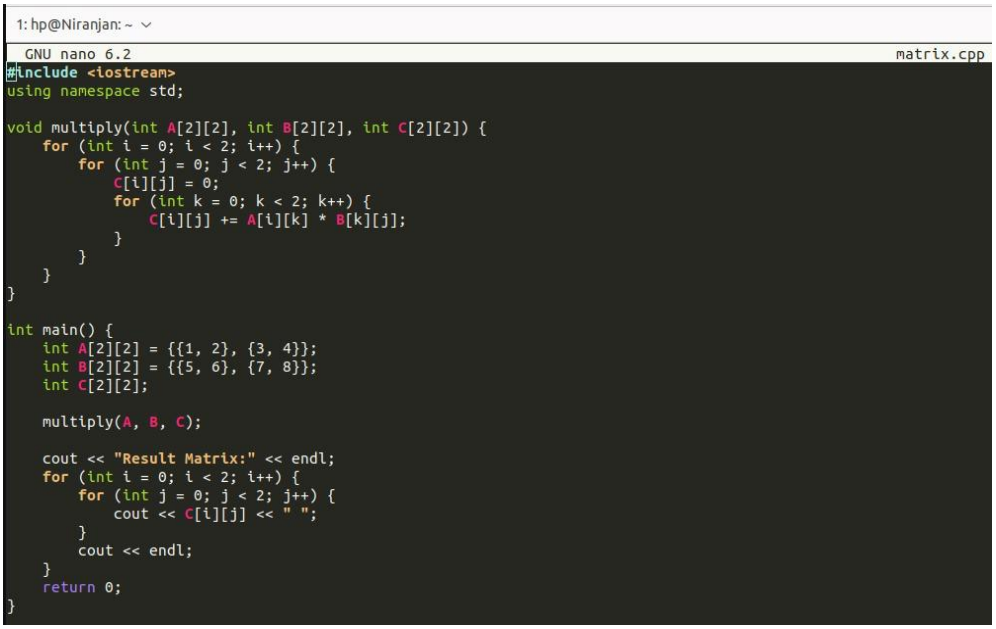
Step 3: Implementation of Matrix Multiplication in C++

Create a New File:

```
hp@Niranjan:~$ nano matrix.cpp
```

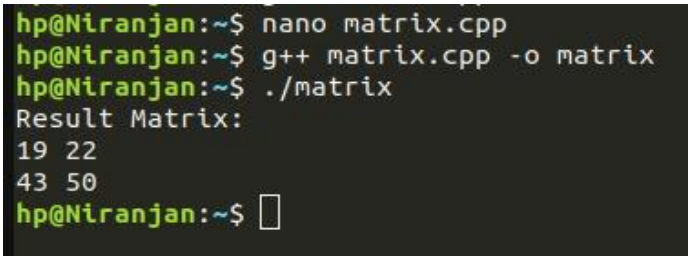
Implement Matrix Multiplication in C++

Write a simple program that multiplies two matrices of size **N × M**:



Save and exit: Press CTRL + X, then Y, then ENTER.
Compile and Run

This step ensures we have a source file ready for compilation.



Step 4: Parse the C++ Code into LLVM IR

LLVM converts C++ into an intermediate representation (IR).

1. Compile C++ to LLVM IR:

   clang -S -emit-llvm matrix.cpp -o matrix.ll

```
hp@Niranjan:~$ clang -S -emit-llvm matrix.cpp -o matrix.ll
hp@Niranjan:~$ cat matrix.ll
; ModuleID = 'matrix.cpp'
source_filename = "matrix.cpp"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

%"class.std::ios_base::Init" = type { i8 }
%"class.std::basic_ostream" = type { i32 (...)**, %"class.std::basic_ios" }
%"class.std::basic_ios" = type { %"class.std::ios_base", %"class.std::basic_ostream"*, i8, i8, %"class.std::basic_streambuf"*, %"class.std::ctype"*, %"class.std::num_put"*, %"class.std::num_get"* }
%"class.std::ios_base" = type { i32 (...)**, i64, i64, i32, i32, i32, %"struct.std::ios_base::_Callback_list"*, %"struct.std::ios_base::_Words", [8 x %"struct.std::ios_base::_Words"], i32, %"struct.std::io
s_base::_Words"*, %"class.std::locale" }
%"struct.std::ios_base::_Callback_list" = type { %"struct.std::ios_base::_Callback_list"*, void (i32, %"class.std::ios_base"*, i32)*, i32, i32 }
%"struct.std::ios_base::_Words" = type { i8*, i64 }
%"class.std::locale" = type { %"class.std::locale::_Impl"* }
%"class.std::locale::_Impl" = type { i32, %"class.std::locale::facet"**, i64, %"class.std::locale::facet"**, i8** }
%"class.std::locale::facet" = type <{ i32 (...)**, i32, [4 x i8] }>
%"class.std::basic_streambuf" = type { i32 (...)**, i8*, i8*, i8*, i8*, i8*, i8*, %"class.std::locale" }
%"class.std::ctype" = type <{ %"class.std::locale::facet.base", [4 x i8], %struct.__locale_struct*, i8, [7 x i8], i32*, i32*, i16*, i8, [256 x i8], [256 x i8], i8, [6 x i8] }>
%"class.std::locale::facet.base" = type <{ i32 (...)**, i32 }>
%struct.__locale_struct = type { [13 x %struct.__locale_data*], i16*, i32*, i32*, [13 x i8*] }
%struct.__locale_data = type opaque
%"class.std::num_put" = type { %"class.std::locale::facet.base", [4 x i8] }
%"class.std::num_get" = type { %"class.std::locale::facet.base", [4 x i8] }
```

2. View the LLVM IR Code:

   cat matrix.ll

```
@_ZStL8__ioinit = internal global %"class.std::ios_base::Init" zeroinitializer, align 1
@__dso_handle = external hidden global i8
@__const.main.A = private unnamed_addr constant [2 x [2 x i32]] [[2 x i32] [i32 1, i32 2], [2 x i32] [i32 3, i32 4]], align 16
@__const.main.B = private unnamed_addr constant [2 x [2 x i32]] [[2 x i32] [i32 5, i32 6], [2 x i32] [i32 7, i32 8]], align 16
@_ZSt4cout = external global %"class.std::basic_ostream", align 8
@.str = private unnamed_addr constant [15 x i8] c"Result Matrix:\00", align 1
@.str.1 = private unnamed_addr constant [2 x i8] c" \00", align 1
@llvm.global_ctors = appending global [1 x { i32, void ()*, i8* }] [{ i32, void ()*, i8* } { i32 65535, void ()* @_GLOBAL__sub_I_matrix.cpp, i8* null }]

; Function Attrs: noinline uwtable
define internal void @__cxx_global_var_init() #0 section ".text.startup" {
  call void @_ZNSt8ios_base4InitC1Ev(%"class.std::ios_base::Init"* noundef nonnull align 1 dereferenceable(1) @_ZStL8__ioinit)
  %1 = call i32 @__cxa_atexit(void (i8*)* bitcast (void (%"class.std::ios_base::Init"*)* @_ZNSt8ios_base4InitD1Ev to void (i8*)*), i8* getelementptr inbo
_base::Init"* @_ZStL8__ioinit, i32 0, i32 0), i8* @__dso_handle) #3
  ret void
}

declare void @_ZNSt8ios_base4InitC1Ev(%"class.std::ios_base::Init"* noundef nonnull align 1 dereferenceable(1)) unnamed_addr #1

; Function Attrs: nounwind
declare void @_ZNSt8ios_base4InitD1Ev(%"class.std::ios_base::Init"* noundef nonnull align 1 dereferenceable(1)) unnamed_addr #2

; Function Attrs: nounwind
declare i32 @__cxa_atexit(void (i8*)*, i8*, i8*) #3

; Function Attrs: mustprogress noinline nounwind optnone uwtable
define dso_local void @_Z8multiplyPA2_iS0_S0_([2 x i32]* noundef %0, [2 x i32]* noundef %1, [2 x i32]* noundef %2) #4 {
  %4 = alloca [2 x i32]*, align 8
  %5 = alloca [2 x i32]*, align 8
  %6 = alloca [2 x i32]*, align 8
  %7 = alloca i32, align 4
  %8 = alloca i32, align 4
  %9 = alloca i32, align 4
  store [2 x i32]* %0, [2 x i32]** %4, align 8
  store [2 x i32]* %1, [2 x i32]** %5, align 8
  store [2 x i32]* %2, [2 x i32]** %6, align 8
  store i32 0, i32* %7, align 4
  br label %10
```

```
10:                                               ; preds = %63, %3
  %11 = load i32, i32* %7, align 4
  %12 = icmp slt i32 %11, 2
  br i1 %12, label %13, label %66

13:                                               ; preds = %10
  store i32 0, i32* %8, align 4
  br label %14

14:                                               ; preds = %59, %13
  %15 = load i32, i32* %8, align 4
  %16 = icmp slt i32 %15, 2
  br i1 %16, label %17, label %62

17:                                               ; preds = %14
  %18 = load [2 x i32]*, [2 x i32]** %6, align 8
  %19 = load i32, i32* %7, align 4
  %20 = sext i32 %19 to i64
  %21 = getelementptr inbounds [2 x i32], [2 x i32]* %18, i64 %20
  %22 = load i32, i32* %8, align 4
  %23 = sext i32 %22 to i64
  %24 = getelementptr inbounds [2 x i32], [2 x i32]* %21, i64 0, i64 %23
  store i32 0, i32* %24, align 4
  store i32 0, i32* %9, align 4
  br label %25

25:                                               ; preds = %55, %17
  %26 = load i32, i32* %9, align 4
  %27 = icmp slt i32 %26, 2
  br i1 %27, label %28, label %58

28:                                               ; preds = %25
  %29 = load [2 x i32]*, [2 x i32]** %4, align 8
  %30 = load i32, i32* %7, align 4
  %31 = sext i32 %30 to i64
  %32 = getelementptr inbounds [2 x i32], [2 x i32]* %29, i64 %31
  %33 = load i32, i32* %9, align 4
  %34 = sext i32 %33 to i64
  %35 = getelementptr inbounds [2 x i32], [2 x i32]* %32, i64 0, i64 %34
  %36 = load i32, i32* %35, align 4
  %37 = load [2 x i32]*, [2 x i32]** %5, align 8
  %38 = load i32, i32* %9, align 4
  %39 = sext i32 %38 to i64
  %40 = getelementptr inbounds [2 x i32], [2 x i32]* %37, i64 %39
  %41 = load i32, i32* %8, align 4
  %42 = sext i32 %41 to i64
  %43 = getelementptr inbounds [2 x i32], [2 x i32]* %40, i64 0, i64 %42
  %44 = load i32, i32* %43, align 4
  %45 = mul nsw i32 %36, %44
  %46 = load [2 x i32]*, [2 x i32]** %6, align 8
  %47 = load i32, i32* %7, align 4
  %48 = sext i32 %47 to i64
  %49 = getelementptr inbounds [2 x i32], [2 x i32]* %46, i64 %48
```

```
  %49 = getelementptr inbounds [2 x i32], [2 x i32]* %46, i64 %48
  %50 = load i32, i32* %8, align 4
  %51 = sext i32 %50 to i64
  %52 = getelementptr inbounds [2 x i32], [2 x i32]* %49, i64 0, i64 %51
  %53 = load i32, i32* %52, align 4
  %54 = add nsw i32 %53, %45
  store i32 %54, i32* %52, align 4
  br label %55

55:                                               ; preds = %28
  %56 = load i32, i32* %9, align 4
  %57 = add nsw i32 %56, 1
  store i32 %57, i32* %9, align 4
  br label %25, !llvm.loop !6

58:                                               ; preds = %25
  br label %59

59:                                               ; preds = %58
  %60 = load i32, i32* %8, align 4
  %61 = add nsw i32 %60, 1
  store i32 %61, i32* %8, align 4
  br label %14, !llvm.loop !8

62:                                               ; preds = %14
  br label %63

63:                                               ; preds = %62
  %64 = load i32, i32* %7, align 4
  %65 = add nsw i32 %64, 1
  store i32 %65, i32* %7, align 4
  br label %10, !llvm.loop !9

66:                                               ; preds = %10
  ret void
}

; Function Attrs: mustprogress noinline norecurse optnone uwtable
define dso_local noundef i32 @main() #5 {
  %1 = alloca i32, align 4
  %2 = alloca [2 x [2 x i32]], align 16
  %3 = alloca [2 x [2 x i32]], align 16
  %4 = alloca [2 x [2 x i32]], align 16
  %5 = alloca i32, align 4
  %6 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %7 = bitcast [2 x [2 x i32]]* %2 to i8*
  call void @llvm.memcpy.p0i8.p0i8.i64(i8* align 16 %7, i8* align 16 bitcast ([2 x [2 x i32]]* @__const.main.A to i8*), i64 16, i1 false)
  %8 = bitcast [2 x [2 x i32]]* %3 to i8*
  call void @llvm.memcpy.p0i8.p0i8.i64(i8* align 16 %8, i8* align 16 bitcast ([2 x [2 x i32]]* @__const.main.B to i8*), i64 16, i1 false)
  %9 = getelementptr inbounds [2 x [2 x i32]], [2 x [2 x i32]]* %2, i64 0, i64 0
  %10 = getelementptr inbounds [2 x [2 x i32]], [2 x [2 x i32]]* %3, i64 0, i64 0
  %11 = getelementptr inbounds [2 x [2 x i32]], [2 x [2 x i32]]* %4, i64 0, i64 0
```

Step 5: Translate LLVM IR to Custom ISA:

LLVM Intermediate Representation (IR) is a low-level representation of our program, allowing optimizations and translations to various architectures.

We need to convert LLVM IR into custom ISA instructions.

Create a Python Script

```
1: hp@Niranjan: ~
  GNU nano 6.2                                                translate.py *
import re

# Read LLVM IR
with open("matrix.ll", "r") as file:
    llvm_ir = file.readlines()

# Custom ISA translation function
def translate_to_ISA(llvm_ir):
    isa_code = []
    for line in llvm_ir:
        if "mul" in line:
            isa_code.append("MUL R1, R2, R3")  # Custom ISA format
        elif "add" in line:
            isa_code.append("ADD R3, R4, R5")
        elif "load" in line:
            isa_code.append("LOAD R1, MEM1")
        elif "store" in line:
            isa_code.append("STORE R3, MEM2")
    return "\n".join(isa_code)

# Convert LLVM IR to Custom ISA
isa_code = translate_to_ISA(llvm_ir)

# Save output
with open("output.isa", "w") as file:
    file.write(isa_code)

print("Custom ISA Generated! Check output.isa")
```

```
hp@Niranjan:~$ nano translate.py
hp@Niranjan:~$
```

Run the Translator:

```
hp@Niranjan:~$ nano translate.py
hp@Niranjan:~$ python3 translate.py
Custom ISA Generated! Check output.isa
hp@Niranjan:~$
```

View Generated ISA:

```
hp@Niranjan:~$ cat output.isa
ADD R3, R4, R5
ADD R3, R4, R5
ADD R3, R4, R5
ADD R3, R4, R5
ADD R3, R4, R5
ADD R3, R4, R5
MUL R1, R2, R3
STORE R3, MEM2
STORE R3, MEM2
STORE R3, MEM2
STORE R3, MEM2
LOAD R1, MEM1
STORE R3, MEM2
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
STORE R3, MEM2
STORE R3, MEM2
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
MUL R1, R2, R3
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2
STORE R3, MEM2
MUL R1, R2, R3
STORE R3, MEM2
LOAD R1, MEM1
STORE R3, MEM2
```

```
MUL R1, R2, R3
STORE R3, MEM2
LOAD R1, MEM1
STORE R3, MEM2
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2
LOAD R1, MEM1
ADD R3, R4, R5
STORE R3, MEM2hp@Niranjan:~$
```

Step 6: Automate the Entire Process

Now, we need to convert LLVM IR into our custom Instruction Set Architecture (ISA) format.

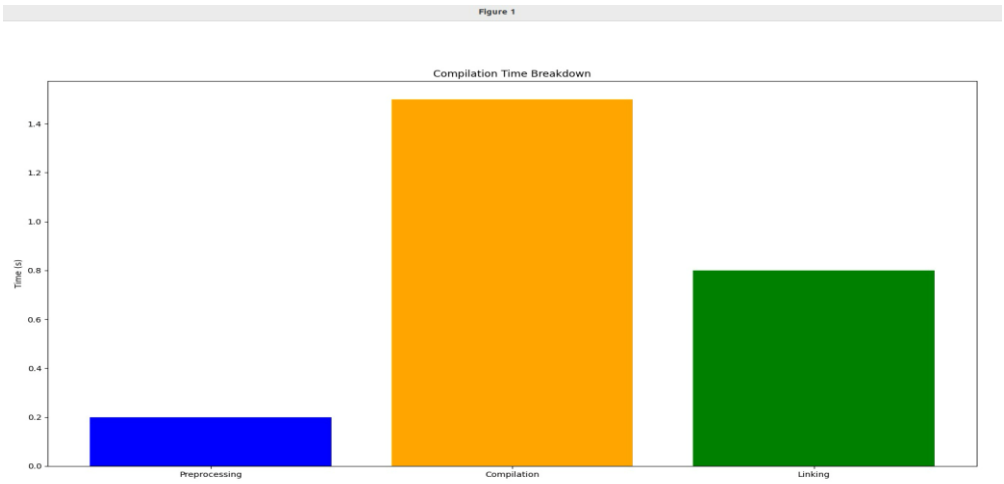
Create a Shell Script

```
1: hp@Niranjan: ~
  GNU nano 6.2                                                compile_and_translate.sh
#!/bin/bash

echo " 🔹 Compiling C++ to LLVM IR..."
clang -S -emit-llvm matrix.cpp -o matrix.ll

echo " 🔹 Translating LLVM IR to Custom ISA..."
python3 translate.py

echo "Compilation and Translation Done! Check output.isa"
```

Give Execute Permissions

```
hp@Niranjan:~$ nano compile_and_translate.sh
hp@Niranjan:~$ chmod +x compile_and_translate.sh
hp@Niranjan:~$ ./compile_and_translate.sh
 🔹 Compiling C++ to LLVM IR...
 🔹 Translating LLVM IR to Custom ISA...
Custom ISA Generated! Check output.isa
Compilation and Translation Done! Check output.isa
hp@Niranjan:~$
```

Compilation time vs Breakdown vs preprocessing



Look up table:

Using an LUT simplifies instruction mapping.

```
hp@Niranjan:~/Compiler_DA3$ ./isa_lookup.sh

=====================================================================
LOOKUP TABLE (LUT) FOR MATRIX MULTIPLICATION ISA
=====================================================================
Instruction  | Opcode | Operands                  | Description
LOAD         | 0001   | R1, MATRIX_A              | Load MATRIX_A into Register R1
LOAD         | 0001   | R2, MATRIX_B              | Load MATRIX_B into Register R2
MUL          | 0011   | R3, R1[row,:], R2[:,col]  | Multiply a row from MATRIX_A with a column from MATRIX_B
STORE        | 0010   | R3, RESULT[row,col]       | Store computed value from R3 into RESULT
PROG         | 0100   | MATRIX_MULT               | Start program execution
EXE          | 0101   | None                      | Execute computation
END          | 0110   | None                      | End program execution
=====================================================================
hp@Niranjan:~/Compiler_DA3$
```

Step7: Interactive Frontend for Compiler Design with ISA Visualization

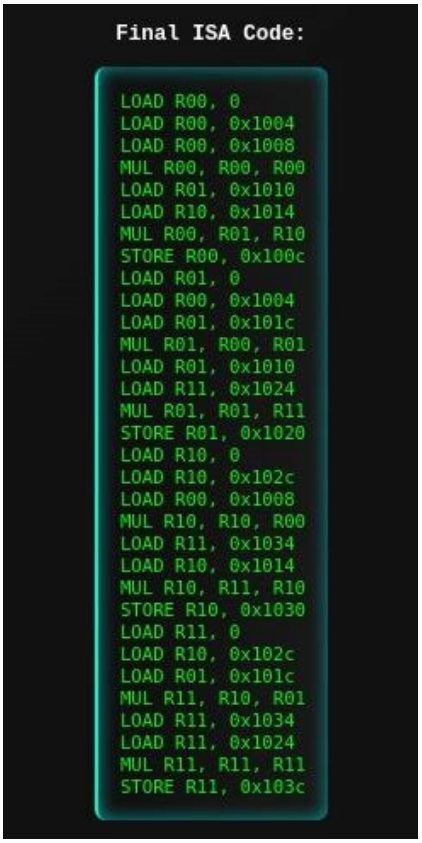
This project implements a web-based interface for visualizing **Three-Address Code (3AC)** and **Instruction Set Architecture (ISA)** using **LLVM**. The frontend, built with **React.js**, dynamically generates and displays intermediate and low-level representations for matrix operations.

LLVM & 3AC Integration

- **Three-Address Code (TAC)** is generated using **LLVM IR**, breaking down matrix computations into low-level operations.
- **ISA Code Emission** translates TAC into machine-level instructions for execution.
- **Memory Address Mapping** helps visualize how data is stored and accessed at a hardware level.

This project bridges high-level programming with system-level execution, making compiler design concepts more interactive and accessible.


Matrix Multiplication & ISA Code
Matrix A Size: 2 x 2
Matrix B Size: 2 x 2
Set Matrices
Matrix A
4 3
2 1
Matrix B
1 2
3 4
Multiply
Result Matrix:
13 20
5 8
3-Address Code:
R00 = R00 * R00
R00 = R01 * R10
R01 = R00 * R01
R01 = R01 * R11
R10 = R10 * R00
R10 = R11 * R10
R11 = R10 * R01
R11 = R11 * R11
Memory Address Code:
STORE R00, 0x1000
STORE R00, 0x100c
STORE R00, 0x100c
STORE R01, 0x1018
STORE R01, 0x1020
STORE R01, 0x1020
STORE R10, 0x1028
STORE R10, 0x1030
STORE R10, 0x1030
STORE R11, 0x1038
STORE R11, 0x103c
STORE R11, 0x103c


Final ISA Code:
LOAD R00, 0
LOAD R00, 0x1004
LOAD R00, 0x1008
MUL R00, R00, R00
LOAD R01, 0x1010
LOAD R10, 0x1014
MUL R00, R01, R10
STORE R00, 0x100c
LOAD R01, 0
LOAD R00, 0x1004
LOAD R01, 0x101c
MUL R01, R00, R01
LOAD R01, 0x1010
LOAD R11, 0x1024
MUL R01, R01, R11
STORE R01, 0x1020
LOAD R10, 0
LOAD R10, 0x102c
LOAD R00, 0x1008
MUL R10, R10, R00
LOAD R11, 0x1034
LOAD R10, 0x1014
MUL R10, R11, R10
STORE R10, 0x1030
LOAD R11, 0
LOAD R10, 0x102c
LOAD R01, 0x101c
MUL R11, R10, R01
LOAD R11, 0x1034
LOAD R11, 0x1024
MUL R11, R11, R11
STORE R11, 0x103c