PARALLEL ARCHITECTURE & DISTRIBUTED PROGRAMMING(21CS71)

DR. MINAL MOHARIR

Introduction to Parallel Computing

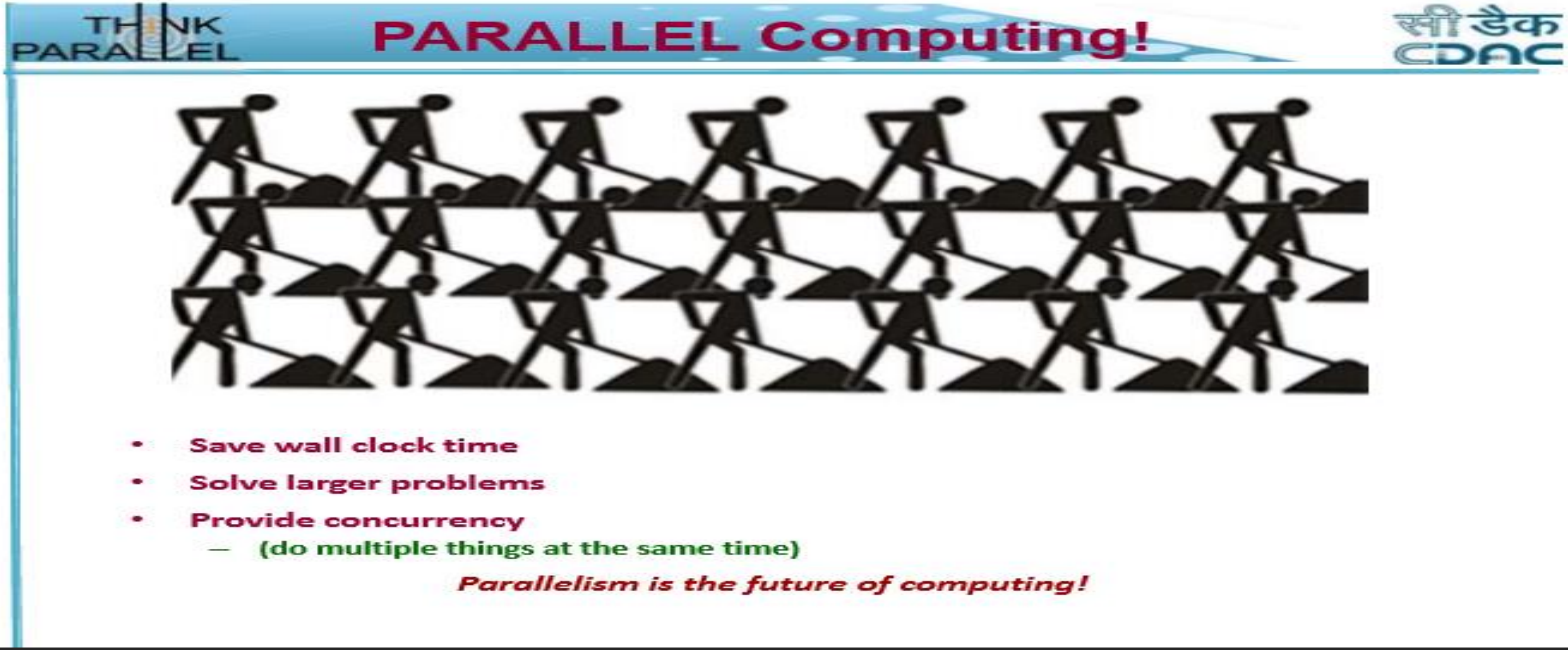
Objective : Motivation to learn Parallel Computing

Serial Computing

If 1 Man takes 10 hours to complete the job; then

How long will it take if 10 Men work together

Parallel Computing

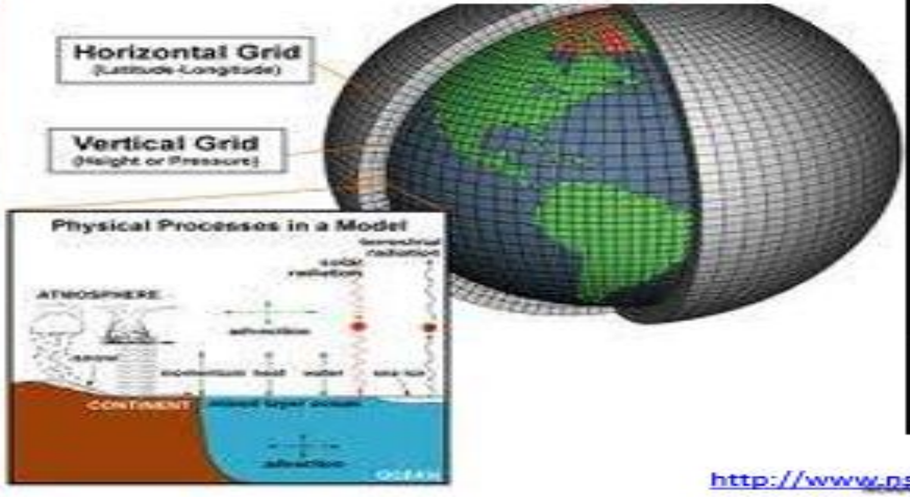


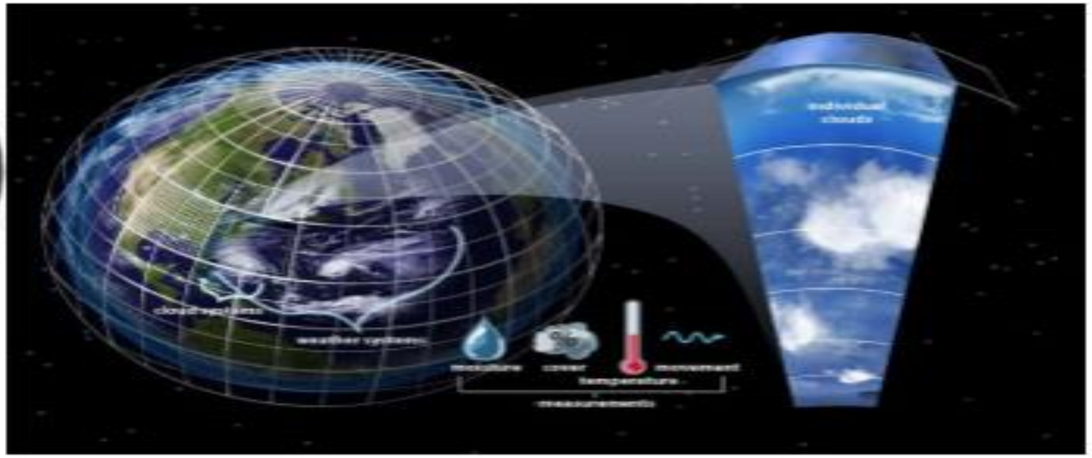
Applications

THINK PARALLEL — Application Drivers for HPC — सी-डैक CDAC

Weather Forecasting

- Atmosphere modeled by dividing it into 3-dimensional cells.
- Calculations of each cell repeated many times to model passage of time.





http://www.nsf.gov/news/special_reports/clouds/images/nsf_clouds_computing.pdf

Applications

Weather Forecasting Example

THINK PARALLEL

सीडैक CDAC

- Consider atmosphere to be divided into cells of size 1 mile $\times$ 1 mile $\times$ 1 mile to a height of 10 miles
- Global region can be schemed as 2×10^9 cells.

- Each cell has many physical variables (temperature, pressure, humidity, wind speed and direction, etc) to be computed
- Suppose each calculation requires 200 floating point operations. In *one time step*, 10^{11} floating point operations necessary.
- To forecast the weather over 7 days using 1-minute intervals, takes $(7 * 24 * 60 * 10^{11}) = 10080 * 10^{11} = 10^{15}$ FP operations
- a computer operating at 1Gflops (10^9 floating point operations/s) takes 10^6 seconds or over 10 days.
- To perform calculation in 5 minutes requires computer operating at 3.4 Tflops (3.4×10^{12} floating point operations/sec).

UNITS

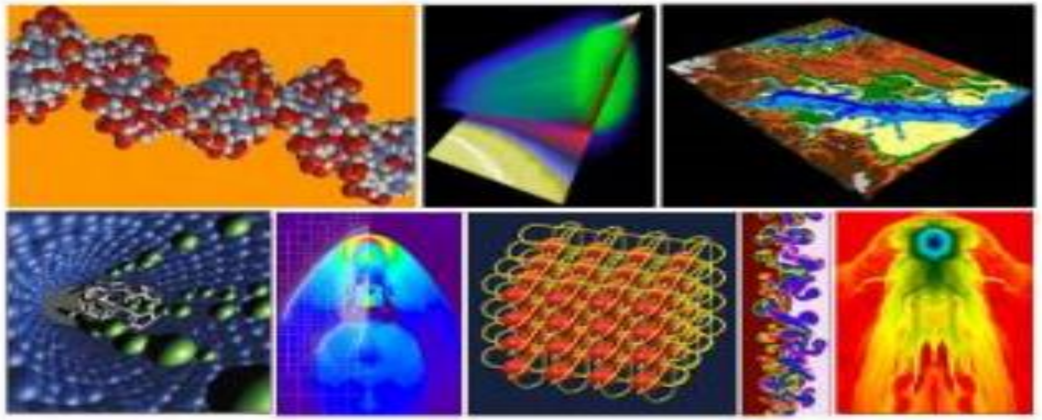
Units of Measurement in HPC

- Mflop/s — 10^6 flop/sec
- Gflop/s — 10^9 flop/sec
- Tflop/s — 10^{12} flop/sec
- Pflop/s — 10^{15} flop/sec
- Mbyte — 10^6 byte (also $2^{20} = 1048576$)
- Gbyte — 10^9 byte (also $2^{30} = 1073741824$)
- Tbyte — 10^{12} byte (also $2^{40} = 10995211627776$)
- Pbyte — 10^{15} byte (also $2^{50} = 1125899906842624$)

Applications

Applications of Parallel Computing

- Prediction of weather, climate, global changes
- Modeling motion of astronomical bodies
- Human genome
- Challenges in materials sciences
- Semiconductor design
- Superconductivity
- Structural biology
- Design of drugs
- Challenges in transportation
- Vehicle dynamics
- Nuclear fusion
- Enhanced oil and gas recovery
- Computational ocean sciences
- Speech
- Vision
- Visualization and graphics



Where are we Today?

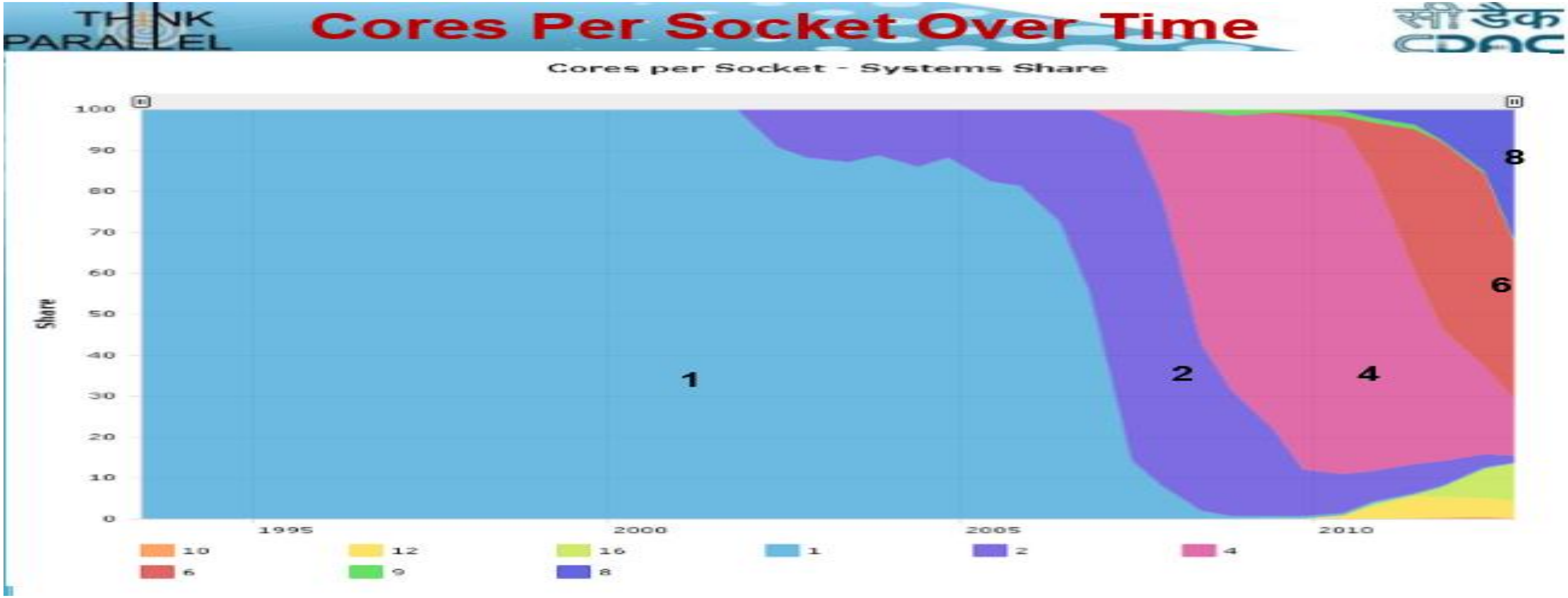
Cost, Size, Performance

Computer Technology Today have made incredible progress in roughly 65-years after first general purpose electronic computer was created.

Where are we Today?

Where are we Today?

Issue: increase in clock speed, increase Power, Heat



Definition

High Performance Computing

- **Definition 1 (Wikipedia)**
  - High Performance Computing (HPC) uses Supercomputers and Computer Clusters to solve advanced computational problems. Today, Computer Systems approaching the Teraflops- region are counted as HPC-Computers

- **Definition 2**
  - High Performance Computing (HPC) is the use of Parallel Processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to systems that function above a TeraFlop or 10^{12} Floating-Point Operations Per Second.

Benchmarking

THINK PARALLEL **TOP500** सी डैक CDAC

Jack Dongarra, H. Simon, E. Strohmaier and H. Meuer

- Listing of the 500 most powerful Supercomputers in the World
- Based on LINPACK Benchmark :Measure of System's Floating Point Computing Power
- How fast a Computer solves a Dense N by N System of Linear Equations

$Ax = b$, *dense problem*

- Result is reported as Rate of Execution in MFlops /Sec
- Updated twice a year

Supercomputing Conference (SC) in United States in November

International Supercomputing Conference (ISC) in Europe in June

www.top500.org June 2024

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
6	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194
7	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA	1,824,768	241.20	306.31	7,494

Principles of CA Design

Take Advantage of Parallelism

- e.g. multiple processors, disks, memory banks, pipelining, multiple functional units

Principle of Locality

- Reuse of data and instructions
- Temporal locality states that recently accessed items are likely to be accessed in the near future.
- Spatial locality says that items whose addresses are near one another tend to be referenced close together in time.

Focus on the Common Case

- The instruction fetch and decode unit of a processor may be used much more frequently than a multiplier, so optimize it first
- Addition of 2-nos., Overflow

Dependency Analysis

- **Their order of execution must not matter!**

- **Example 1**

```
a=1;
b=2;
```

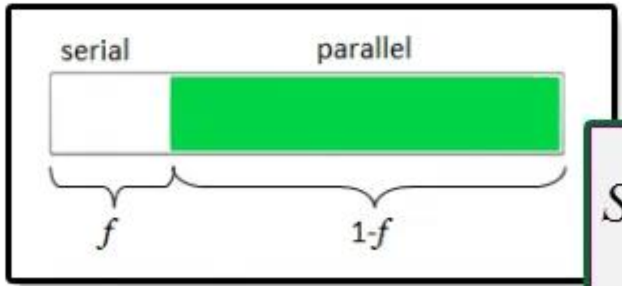
Example 2

```
a=2;
b=a;
```

Amdahl's Law

Inherently serial: I/O

Speedup



$$S_{\max} = \frac{1}{f + \frac{(1-f)}{P}}$$

Program Programming Paradigm

OpenMp

MPI

CUDA

OpenACC

OpenCL

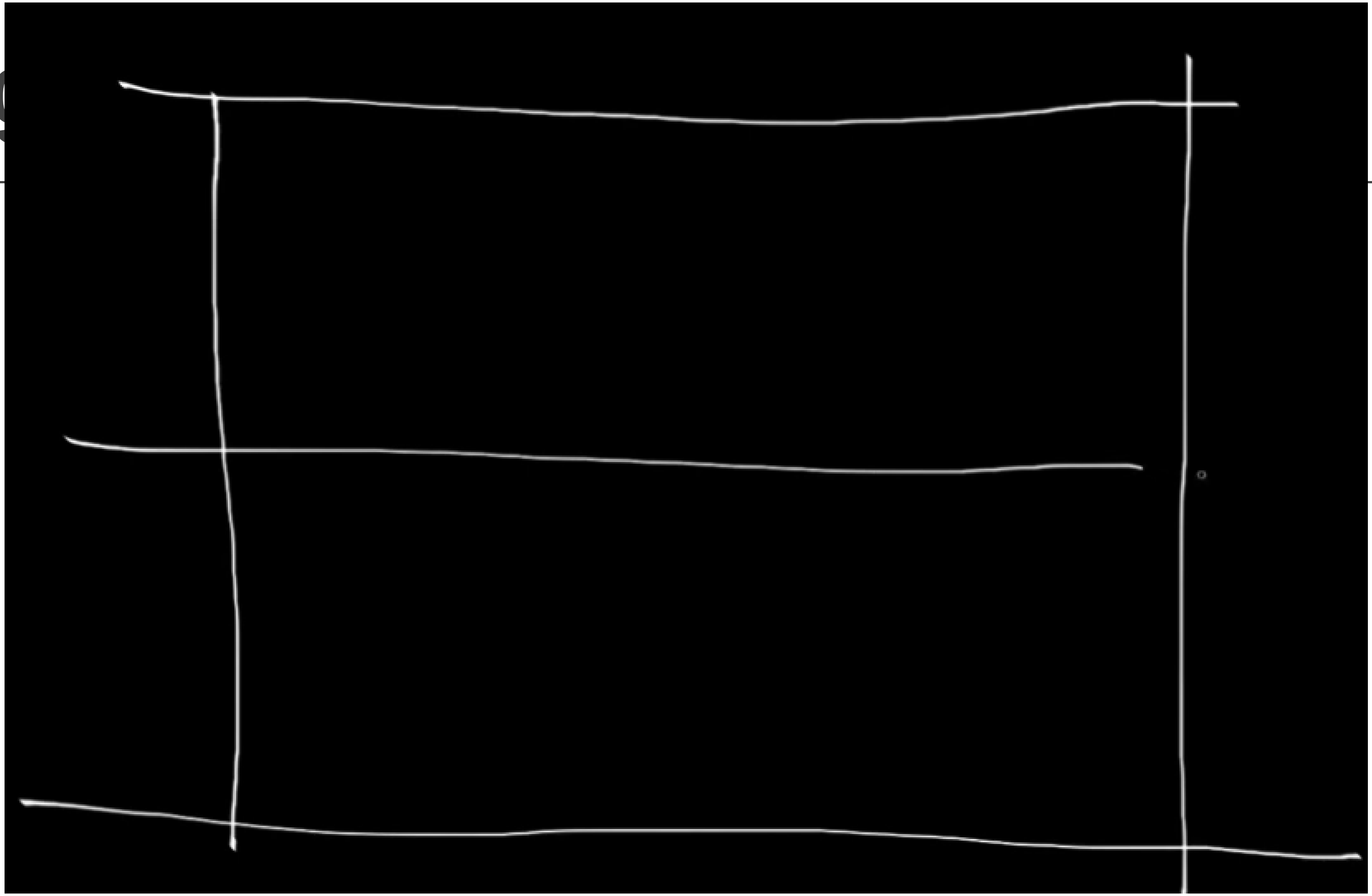
Program Programming Paradigm

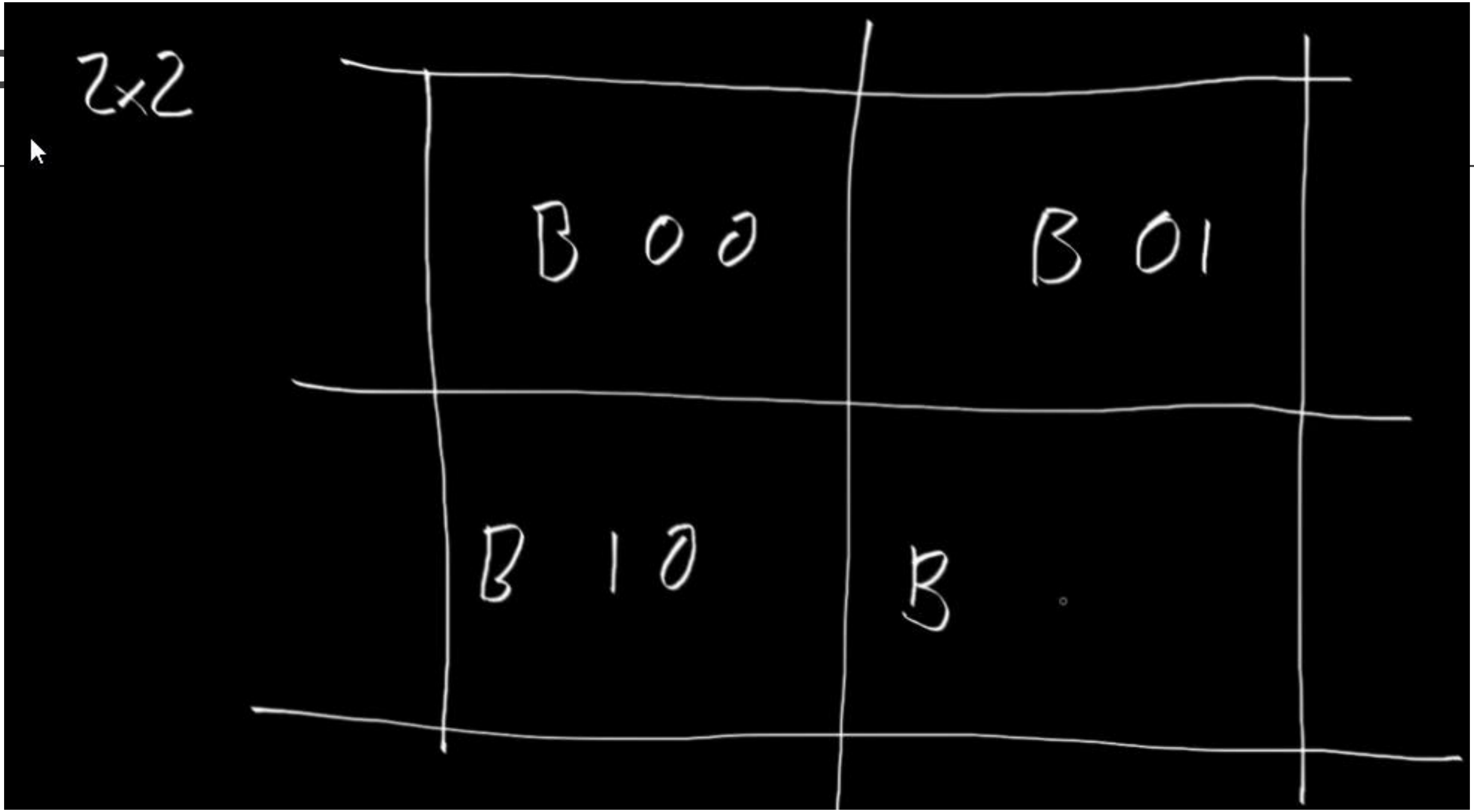
```
#include <stdio.h>

void hello_cuda() {
    printf("Hello Cuda\n");
}

int main(int argc, char **argv) {
    hello_cuda();
    return 0;
}
```

Program Programming Paradigm

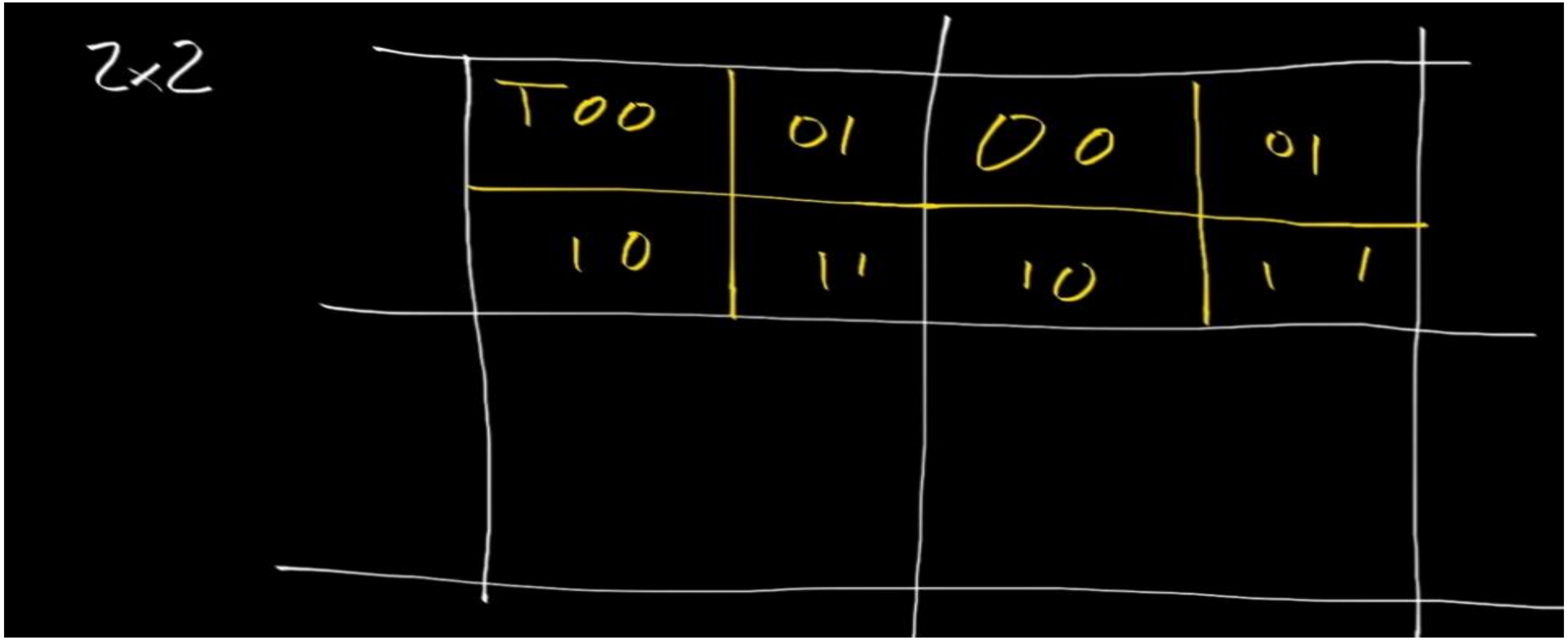
```
#include <stdio.h>

__global__ void hello_cuda() {
    printf("Hello Cuda\n");
}

int main(int argc, char **argv) {
    hello_cuda<<<1,1>>>();
    cudaDeviceSynchronize();

    return 0;
}
```

Prog




2x2
B 00
B 01
B 10
B

Program Programming Paradigm



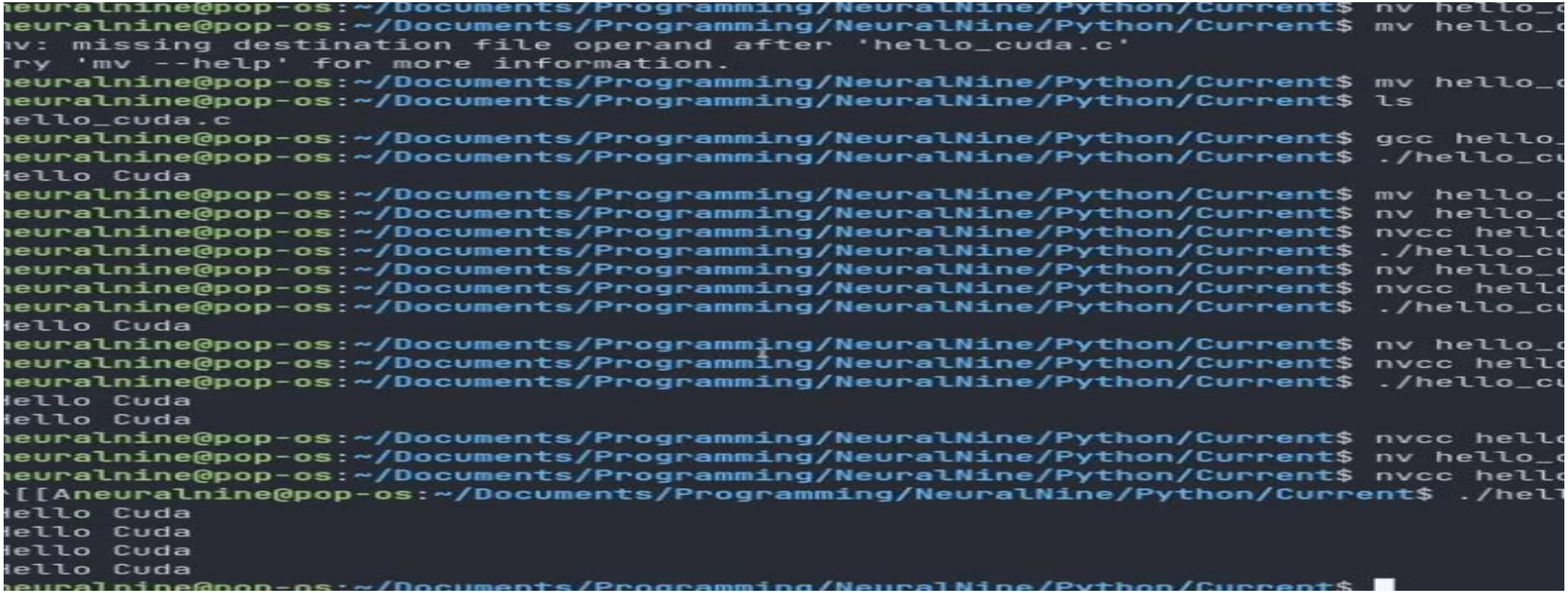
Program Programming Paradigm

```
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ mv hello_cuda.c
v: missing destination file operand after 'hello_cuda.c'
ry 'mv --help' for more information.
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ mv hello_cuda.cu hello_cuda.c
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ls
ello_cuda.c
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ gcc hello_cuda.c -o hello_cuda
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
ello Cuda
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ mv hello_cuda.c hello_cuda.cu
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.cu
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cuda.cu -o hello_cuda
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.cu
euralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cuda.cu -o hello_cuda
```

Program Programming Paradigm

```
#include <stdio.h>

__global__ void hello_cuda() {
    printf("Hello Cuda\n");
}

int main(int argc, char **argv) {
    hello_cuda<<<2,2>>>();
    cudaDeviceSynchronize();

    return 0;
}
```

Program Programming Paradigm



Program Programming Paradigm

```
#include <stdio.h>

__global__ void hello_cuda() {
    printf("Hello Cuda\n");
    printf("Block Index X: %d, Block Index Y: %d, Thread Index X: %d, Thread Index Y: %d\n",
            blockIdx.x, blockIdx.y, threadIdx.x, threadIdx.y);
}

int main(int argc, char **argv) {
    hello_cuda<<<2,2>>>();
    cudaDeviceSynchronize();

    return 0;
}
```

Program Programming Paradigm

```
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
Hello Cuda
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
Hello Cuda
Hello Cuda
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
^[[Aneuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cu
Hello Cuda
Hello Cuda
Hello Cuda
Hello Cuda
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nv hello_cuda.
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ nvcc hello_cud
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$ ./hello_cuda
Hello Cuda
Hello Cuda
Hello Cuda
Hello Cuda
Block Index X: 1, Block Index Y: 0, Thread Index X: 0, Thread Index Y: 0
Block Index X: 1, Block Index Y: 0, Thread Index X: 1, Thread Index Y: 0
Block Index X: 0, Block Index Y: 0, Thread Index X: 0, Thread Index Y: 0
Block Index X: 0, Block Index Y: 0, Thread Index X: 1, Thread Index Y: 0
neuralnine@pop-os:~/Documents/Programming/NeuralNine/Python/Current$
```

THANK YOU