# Assignment

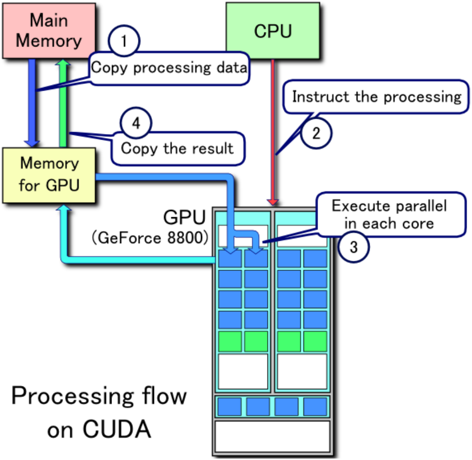
Implement a Parallel ODD-Even Sort algorithm using GPU or ARM equivalent.

Who? Nitin Sankpal/Vrushabh Bhavsar

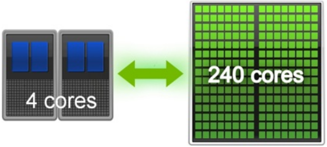
When? March 25, 2015

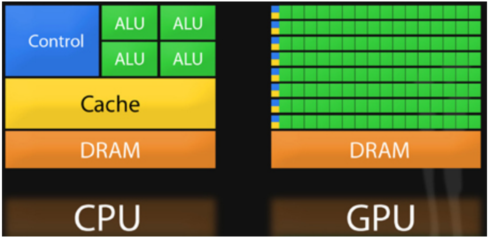
# CUDA?

- Compute Unified Device Architecture, is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce.



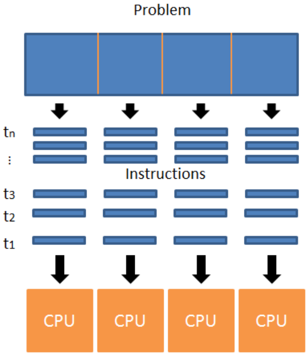
# GPU Vs CPU


4 cores
240 cores


Control
ALU
ALU
ALU
ALU
Cache
DRAM
CPU
DRAM
GPU

# Parallel computing

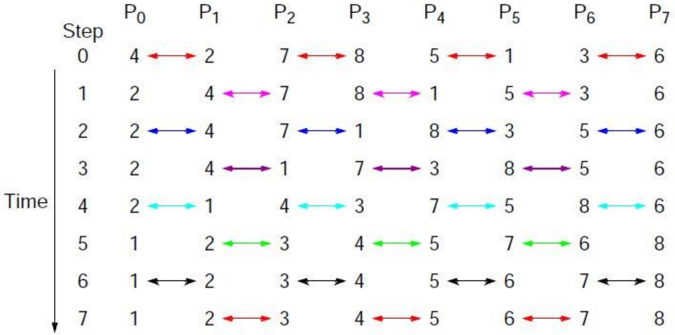
- parallel computing is the simultaneous use of multiple compute resources to solve a computational problem.
- A problem is broken into discrete parts that can be solved concurrently.



# Odd-Even Sort

- Based on bubble-sort technique
- Adjacent pairs of items in an array are exchanged if they are found to be out of order
- operates in two alternate phases:
  - Phase-even: even processes exchange values with right neighbors.
  - Phase-odd: odd processes exchange values with right neighbors.

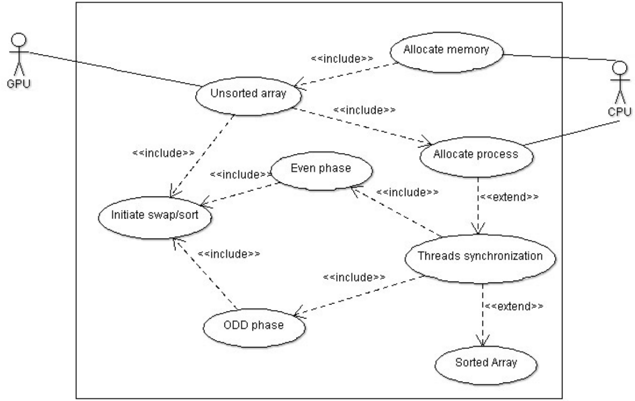
# Odd-Even Sort



| Step | $P_0$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 2 | 7 | 8 | 5 | 1 | 3 | 6 |
| 1 | 2 | 4 | 7 | 8 | 1 | 5 | 3 | 6 |
| 2 | 2 | 4 | 7 | 1 | 8 | 3 | 5 | 6 |
| 3 | 2 | 4 | 1 | 7 | 3 | 8 | 5 | 6 |
| 4 | 2 | 1 | 4 | 3 | 7 | 5 | 8 | 6 |
| 5 | 1 | 2 | 3 | 4 | 5 | 7 | 6 | 8 |
| 6 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

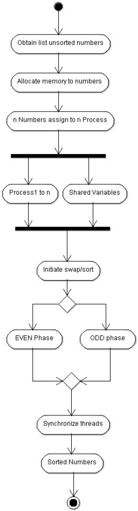
# UML diagrams

Use-case diagram



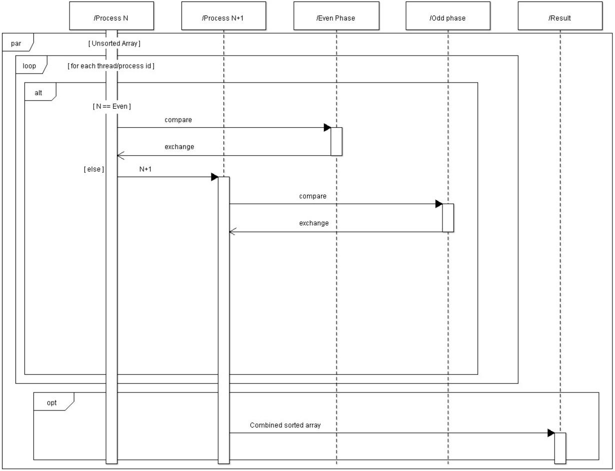
# UML diagrams

## Activity diagram



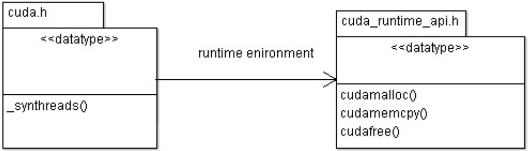
# UML diagrams

## Sequence Diagram

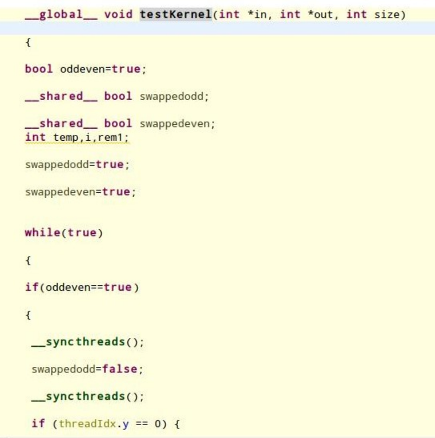


# UML diagrams

Class Diagram

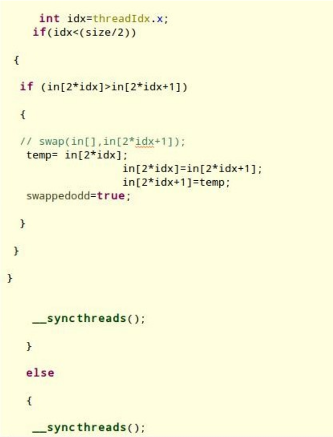


# Code:



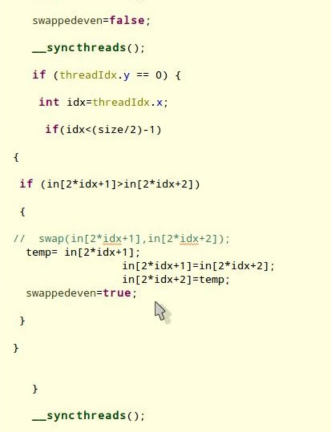
```
__global__ void testKernel(int *in, int *out, int size)

{

bool oddeven=true;

__shared__ bool swappedodd;

__shared__ bool swappedeven;
int temp,i,rem1;

swappedodd=true;

swappedeven=true;


while(true)

{

if(oddeven==true)

{

 __syncthreads();

 swappedodd=false;

 __syncthreads();

 if (threadIdx.y == 0) {
```

# Code:



```
     int idx=threadIdx.x;
    if(idx<(size/2))

 {

  if (in[2*idx]>in[2*idx+1])

  {

  // swap(in[],in[2*idx+1]);
   temp= in[2*idx];
                  in[2*idx]=in[2*idx+1];
                  in[2*idx+1]=temp;
   swappedodd=true;

  }

 }

}


    __syncthreads();

   }

   else

   {

    __syncthreads();
```

# Code:



```
   swappedeven=false;

   __syncthreads();

   if (threadIdx.y == 0) {

    int idx=threadIdx.x;

     if(idx<(size/2)-1)

{

 if (in[2*idx+1]>in[2*idx+2])

 {

//  swap(in[2*idx+1],in[2*idx+2]);
  temp= in[2*idx+1];
                 in[2*idx+1]=in[2*idx+2];
                 in[2*idx+2]=temp;
  swappedeven=true;

 }

}


   }

   __syncthreads();
```