CS516: Parallelization of Programs

Instruction Execution

Vishwesh Jatala

Assistant Professor

Department of CSE

Indian Institute of Technology Bhilai

vishwesh@iitbhilai.ac.in

2023-24-W

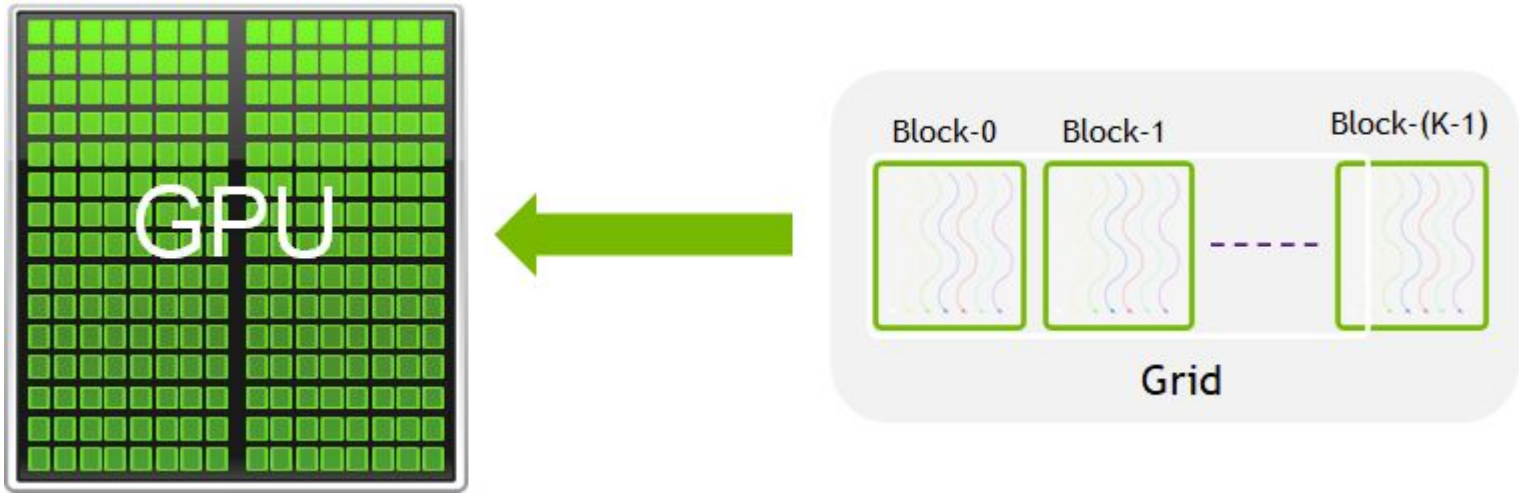
Recap

- CUDA Programming
  - Thread organizations 2D & 3D

Today's outline

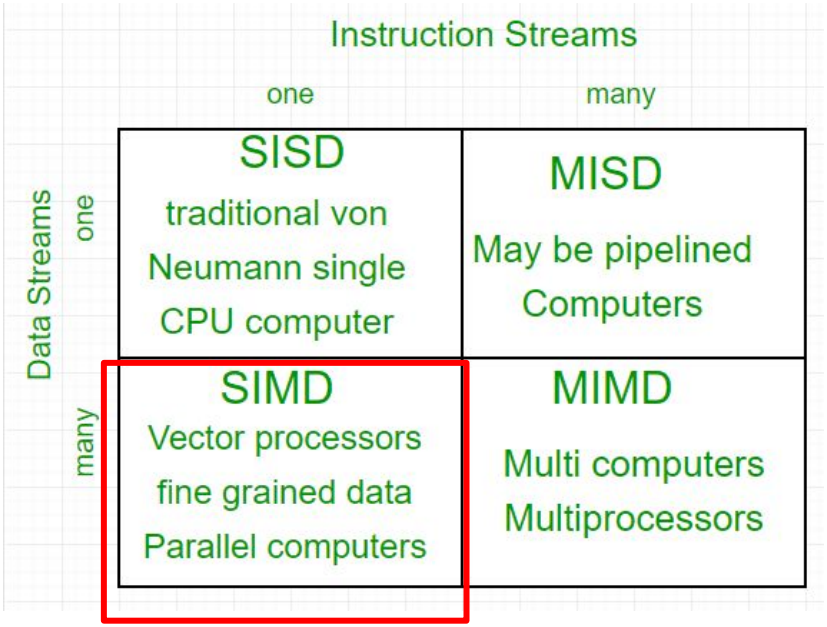
- This Lecture
  - GPU Instruction Execution

GPU Thread Blocks

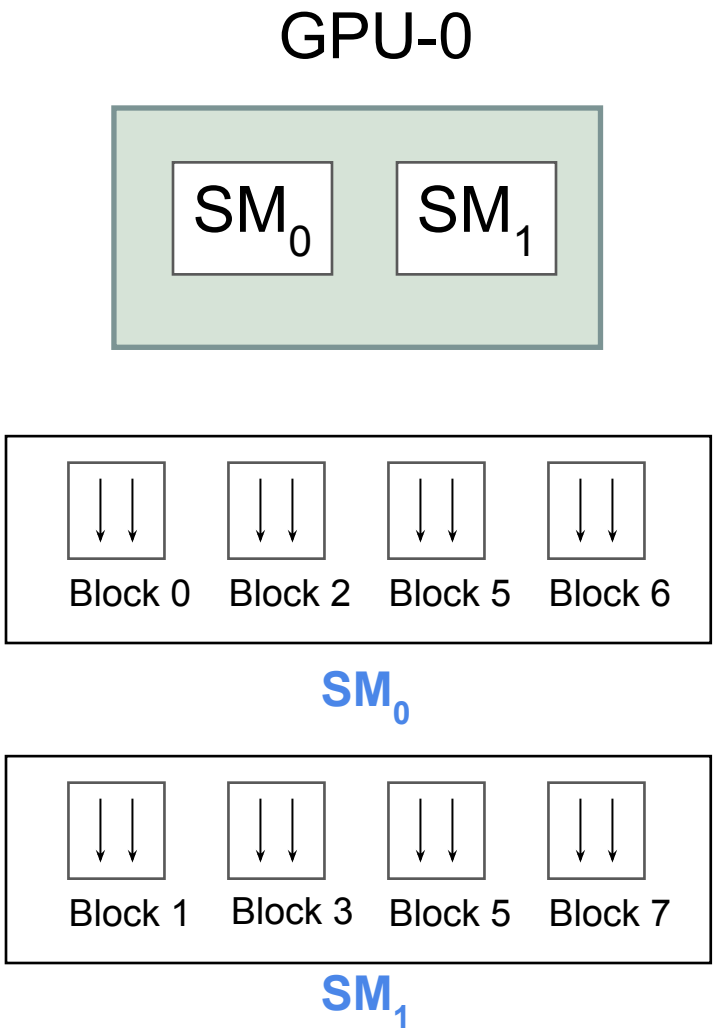


Flynn's Taxonomy

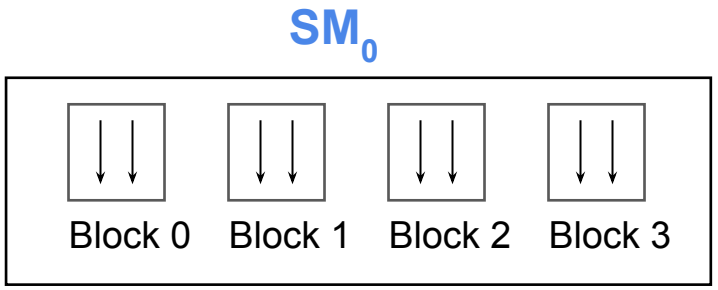
- Flynn's classification of computer architecture



Scalability

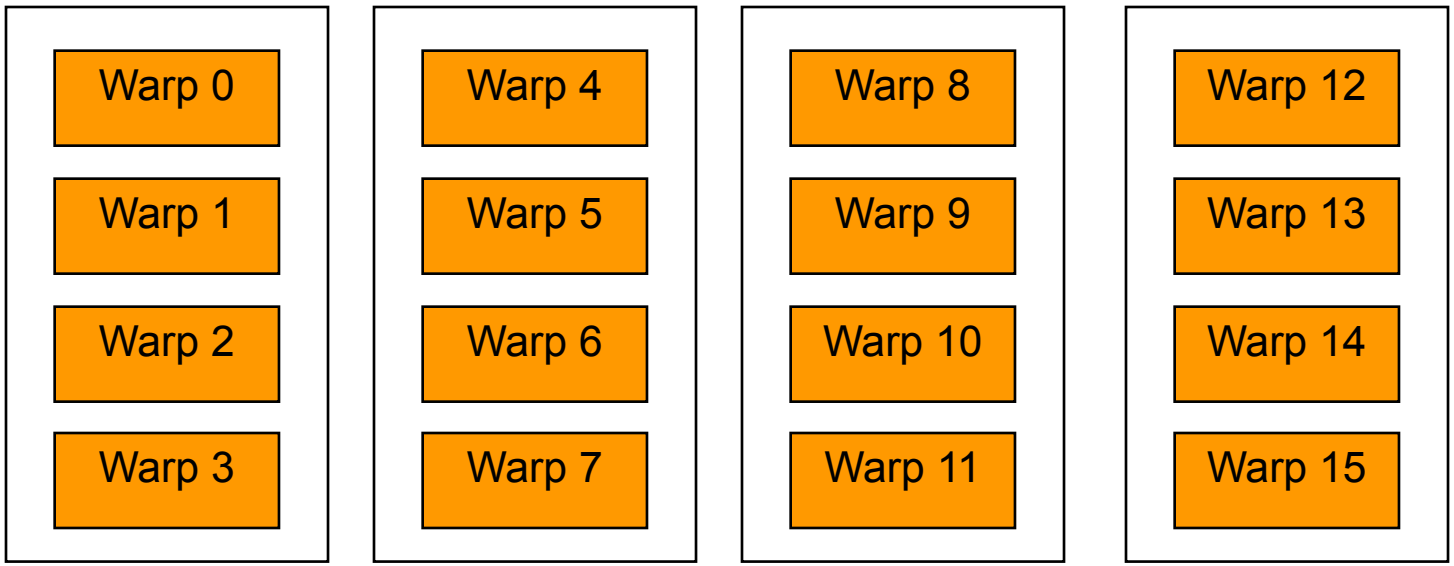


Execution in GPU

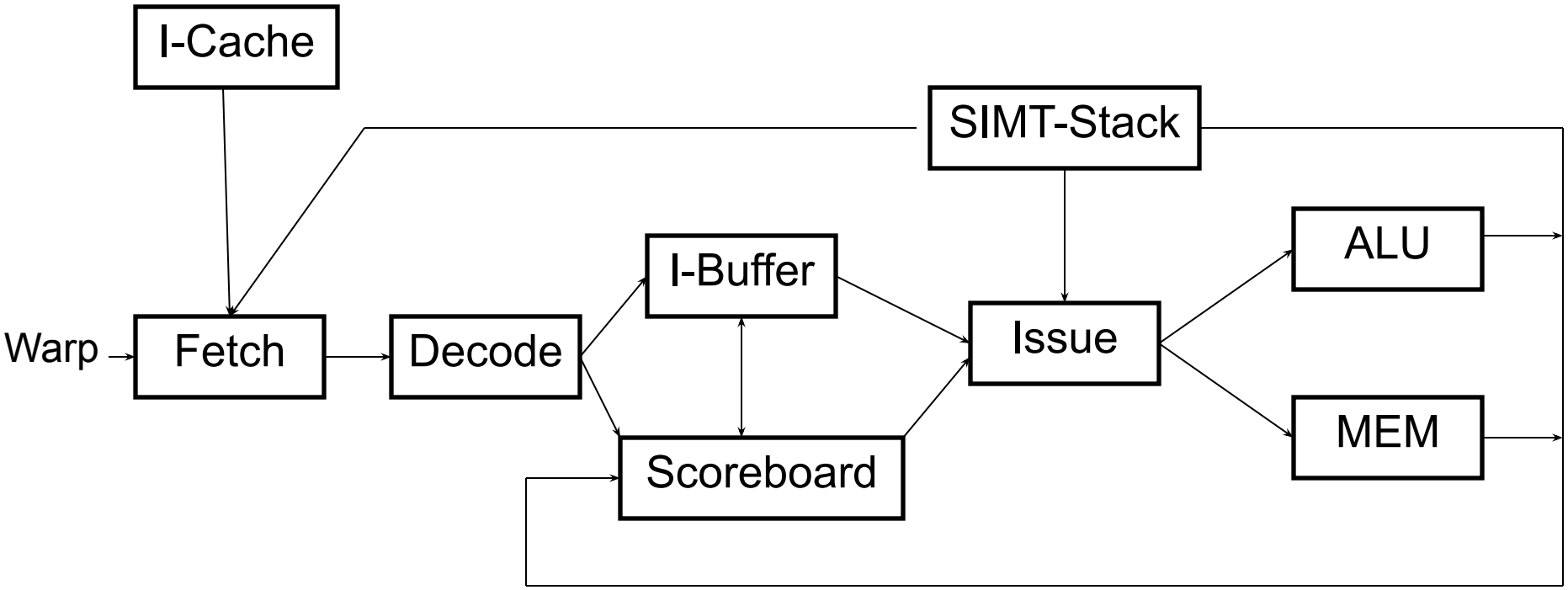


Block Size: 128 threads
Number of Blocks: 4
Total threads: 512

Threads grouped into warps
Size of each warp = 32 threads

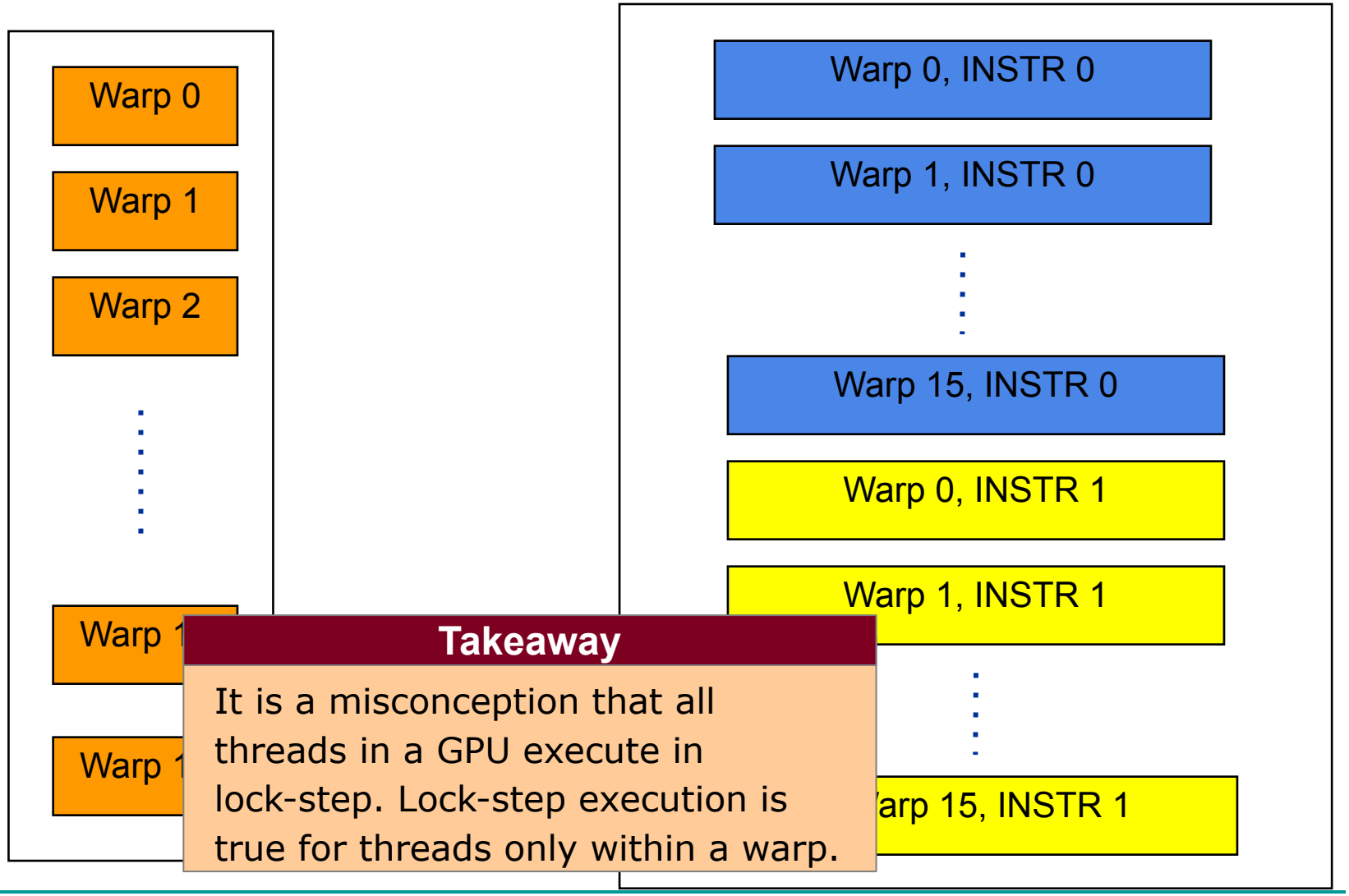


GPU Pipeline*



*As per the GPGPU-Sim Simulator

Instruction Execution



Simple Example (Revisit)

```
__global__ void Square_Kernel(float a){
      /* Compute index i based on thread id */
      a[i] = a[i] * a[i];
}
int main() {
        h_a = malloc(..)//host array
        cudaMalloc(d_a,….) //device

         /* Initialize h_a */

        cudaMemcpy(d_a, h_a, cudaMemcpyHostToDevice)

        Square_Kernel<<<ThreadConfig>>> (d_a);

        cudaMemcpy(h_a, d_a, cudaMemcpyDeviceToHost)
        process(h_a);
        cudaFree(d_a);
        free(h_a);
}
```

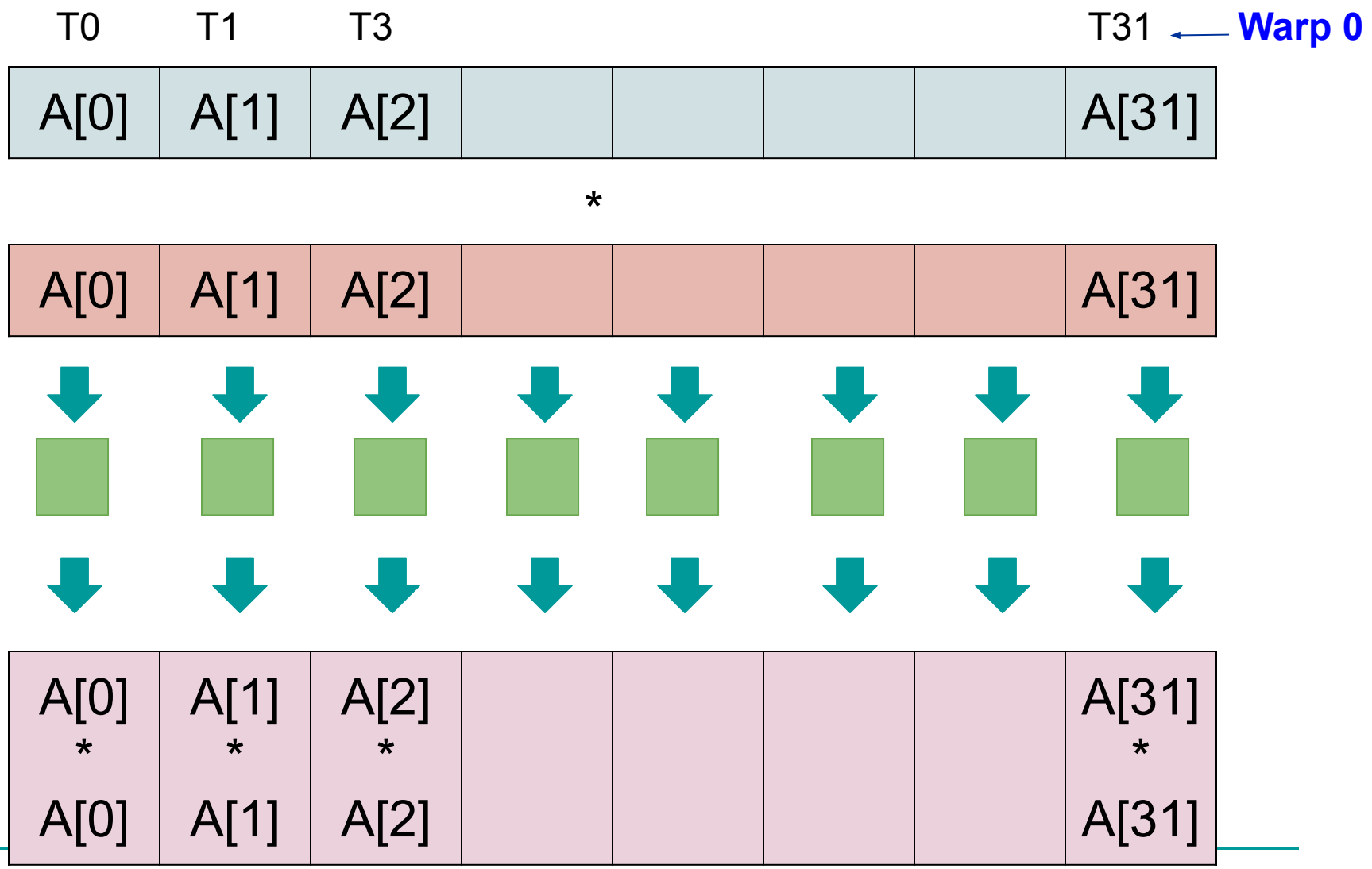
Compute Kernel

CPU to GPU Data transfer

Invoke Kernel

GPU to CPU Data transfer

Warp Execution



Warp Execution with Conditions

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
{
    unsigned id = blockIdx.x * blockDim.x + threadIdx.x;
    if (id % 2) vector[id] = id;
    else vector[id] = vectorsize * vectorsize;
    vector[id]++;
}
```

S0

S1

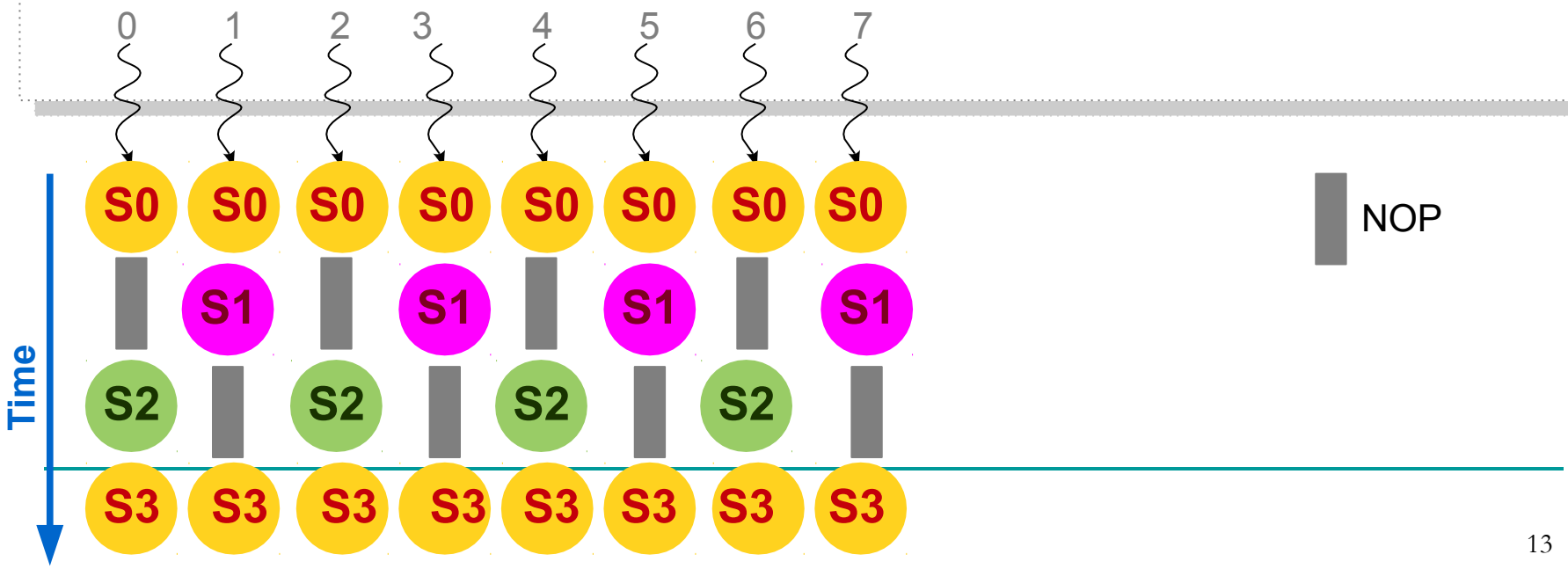
S2

S3

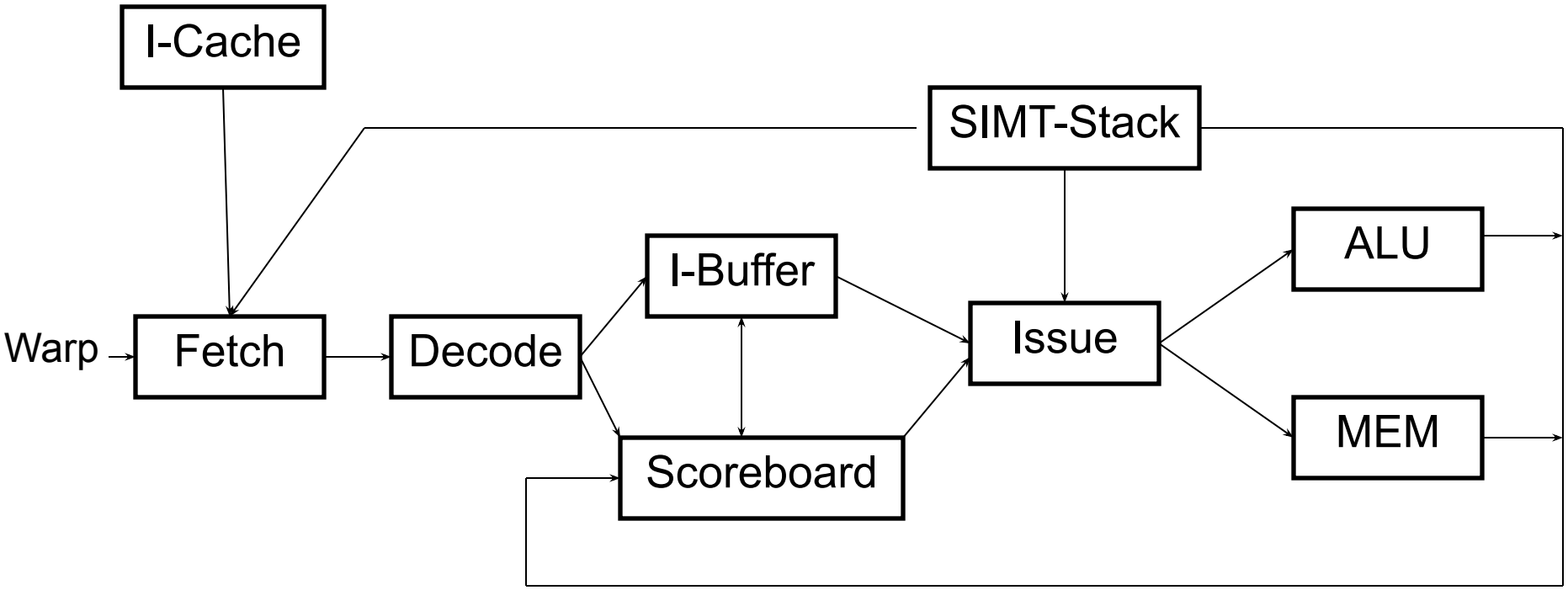
Warp Execution with Conditions

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
{
    unsigned id = blockIdx.x * blockDim.x + threadIdx.x;
    if (id % 2) vector[id] = id;
    else vector[id] = vectorsize * vectorsize;
    vector[id]++;
}
```

S0 S1 S2 S1 S3



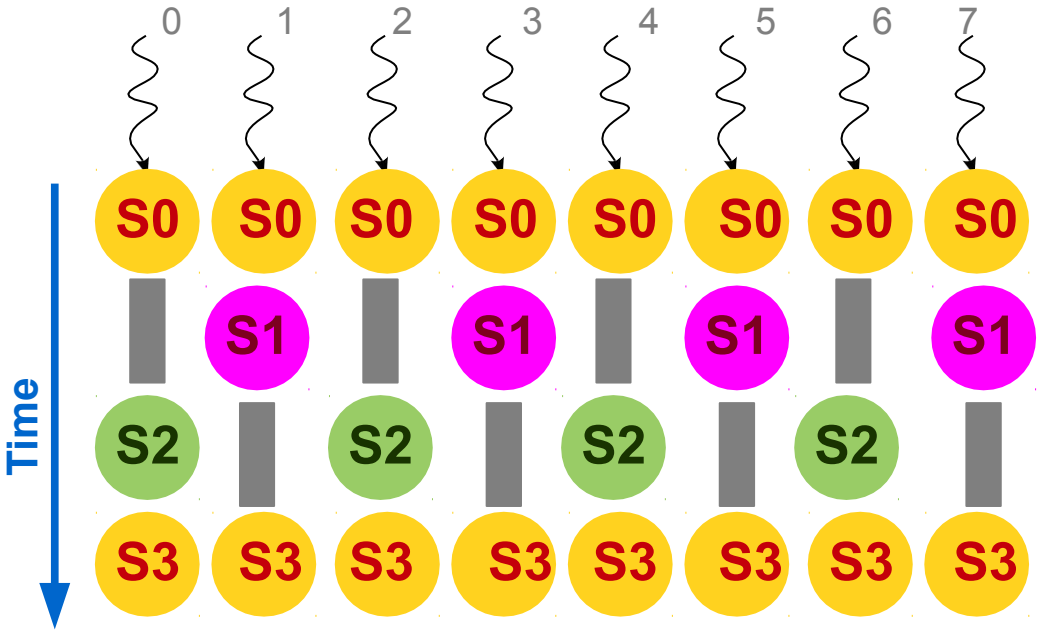
GPU Pipeline*



*As per the GPGPU-Sim Simulator

Warp Execution with Conditions

- When different warp-threads execute different instructions, threads are said to diverge.
- Hardware executes threads satisfying same condition together
- This adds sequentiality to the execution.
- This problem is termed as **thread-divergence**.



Thread Divergence

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
    {
     unsigned id = //compute id.
  if (id % 3 == 0) vector[id] = id;
  else if (id % 3 == 1) vector[id] = id+1;
  else vector[id]=id+2;
```

Degree of Divergence

- DoD for a warp is the number of steps required to complete one instruction for each thread in the warp.
- Without any thread-divergence, DoD = 1.
- For fully divergent code, DoD = 32.

Thread-Divergence

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
        {
         unsigned id = //compute id.
      if (id % 2 == 0) vector[id] = id;

      else vector[id] = vectorsize * vectorsize;
}
```

What is the degree of divergence?

Exercise: Thread-Divergence

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
    {
     unsigned id = //compute id.
  if (id % 3 == 0) vector[id] = id;
  else if (id % 3 == 1) vector[id] = id+1;
  else vector[id]=id+2;
```

What is the degree of divergence?

Exercise: Thread-Divergence

```
__global__ void dkernel(unsigned *vector, unsigned vectorsize)
    {
     unsigned id = //compute id.
    switch (id) {
       case 0: vector[id] = 0;                          break;
       case 1: vector[id] = vector[id];                 break;
       case 2: vector[id] = vector[id - 2];             break;
       case 3: vector[id] = vector[id + 3];             break;
       case 4: vector[id] = 4 + 4 + vector[id];         break;
       case 5: vector[id] = 5 - vector[id];             break;
       case 6: vector[id] = vector[6];                  break;
       case 7: vector[id] = 7 + 7;                      break;
       case 8: vector[id] = vector[id] + 8;             break;
       case 9: vector[id] = vector[id] * 9;             break;
       default: vector[id] = vector[id] + vector[id];   break;
                What is the degree of divergence?
    }
}
```

References

- CS6023 GPU Programming
  - https://www.cse.iitm.ac.in/~rupesh/teaching/gpu/jan20/
- Miscellaneous resources from internet