Embedded SoC

Term Project

Designing General Purpose Parallel Compute Unit

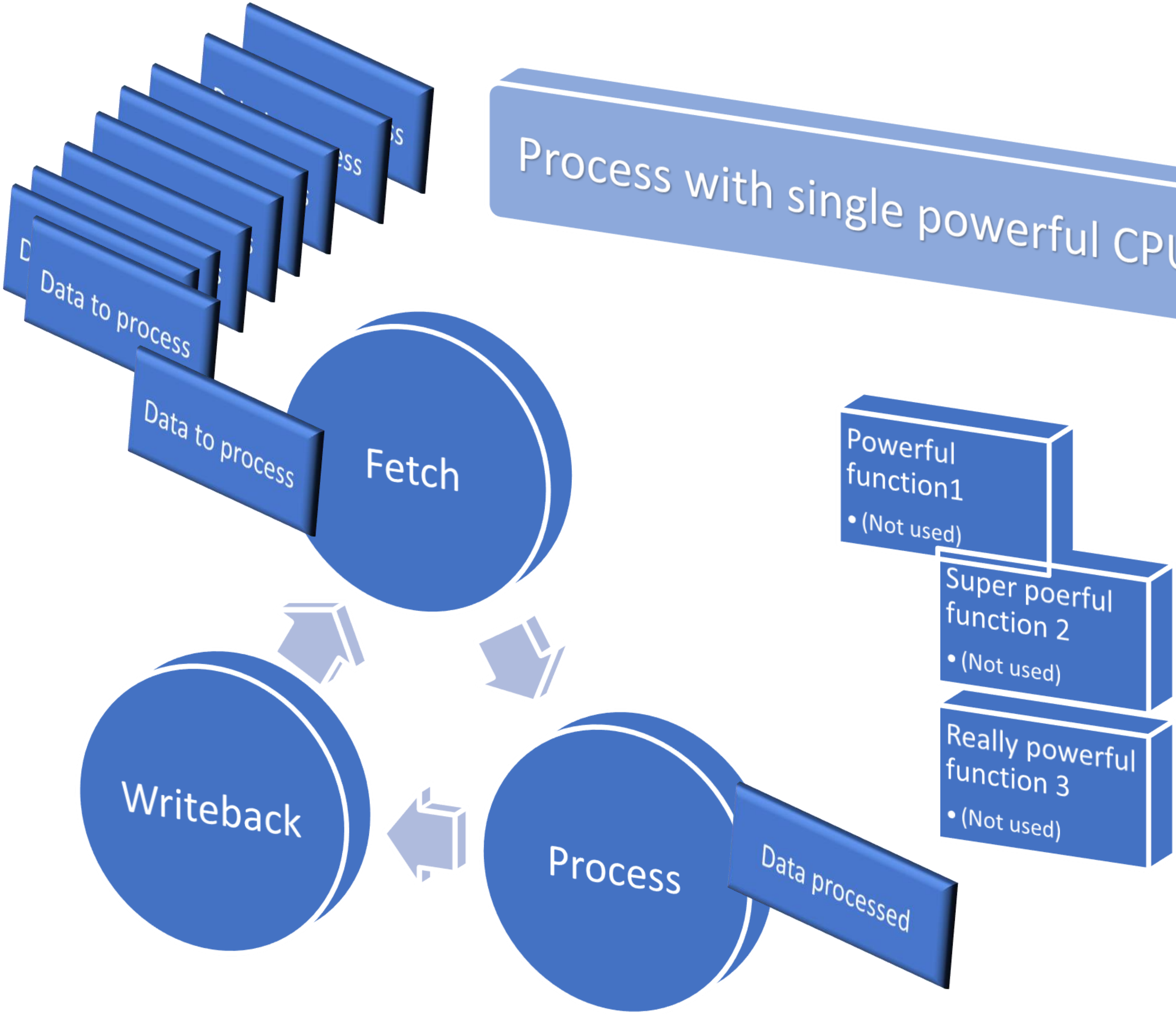
범용 병렬 계산기 설계

우주선 조

2014161001 강승우

KOREATECH Univ, ki6080@gmail.com

Process with single powerful CPU



Data to process

Data to process

Fetch

Process

Data processed

Writeback

Powerful function1

- (Not used)

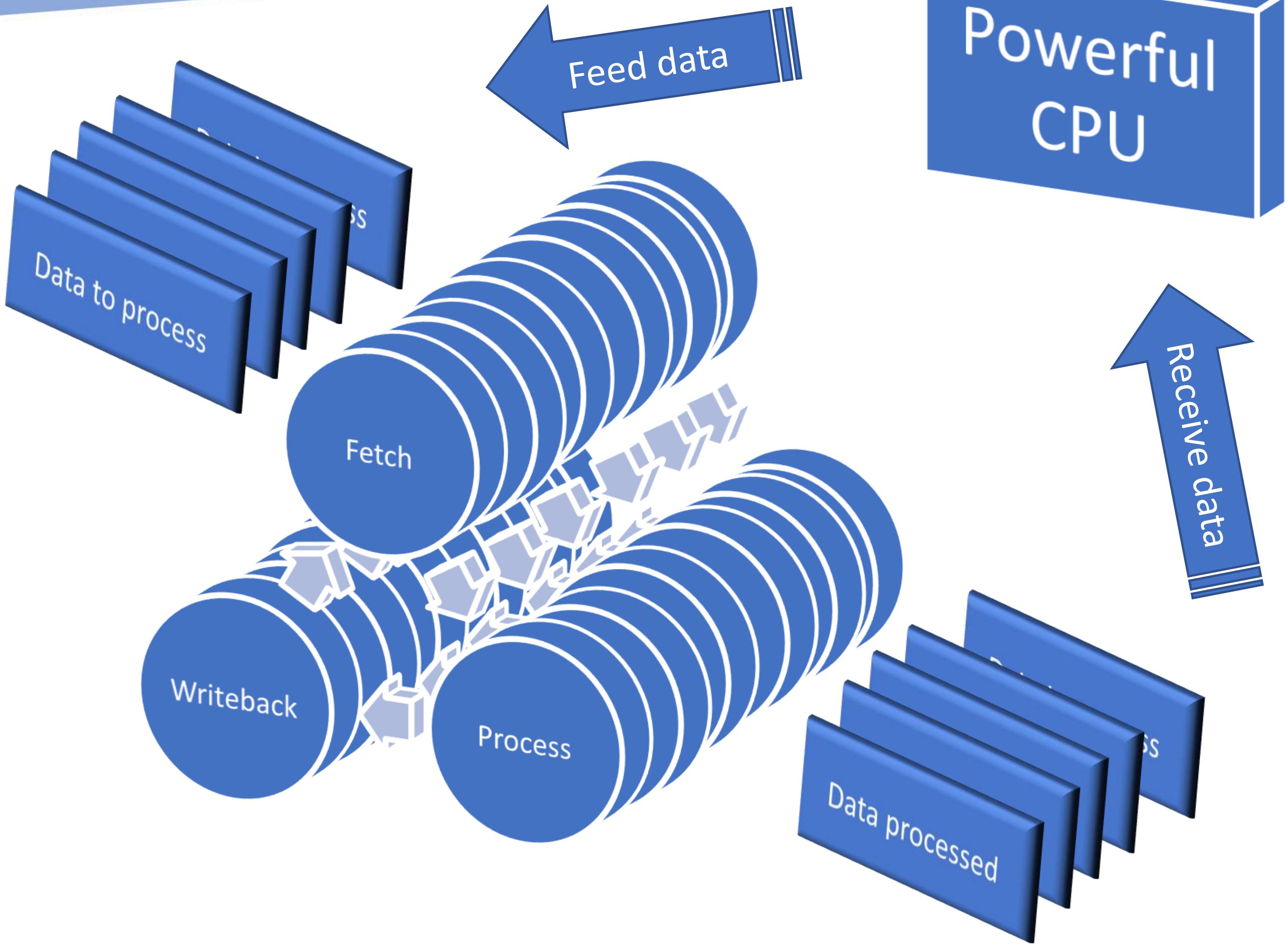
Super poerful function 2

- (Not used)

Really powerful function 3

- (Not used)

Process with numerous simple processors



Feed data

Powerful CPU

Data to process

Fetch

Writeback

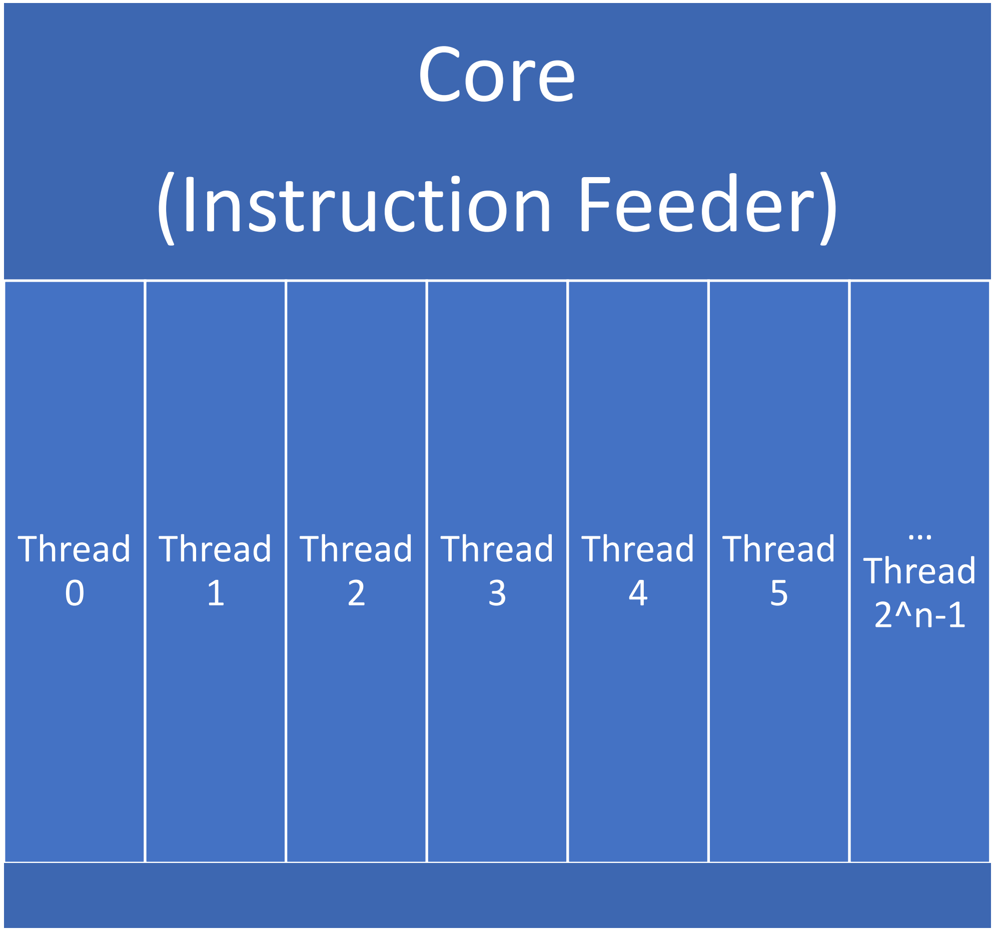
Process

Data processed

Receive data

Concept

- Single Instruction performs 2^n operations
- Each thread has 16 32 bit general purpose registers.
- No programmable instruction memory, no flow control on its own.
- only logic & arithmetic operation supported.
  Instructions should be fed from outside.



Core

(Instruction Feeder)

Thread 0

Thread 1

Thread 2

Thread 3

Thread 4

Thread 5

…
Thread 2^n-1

Goal

Advanced SIMD operation

- Numerous independent processors executes same command at the same time
- Fast local memory access

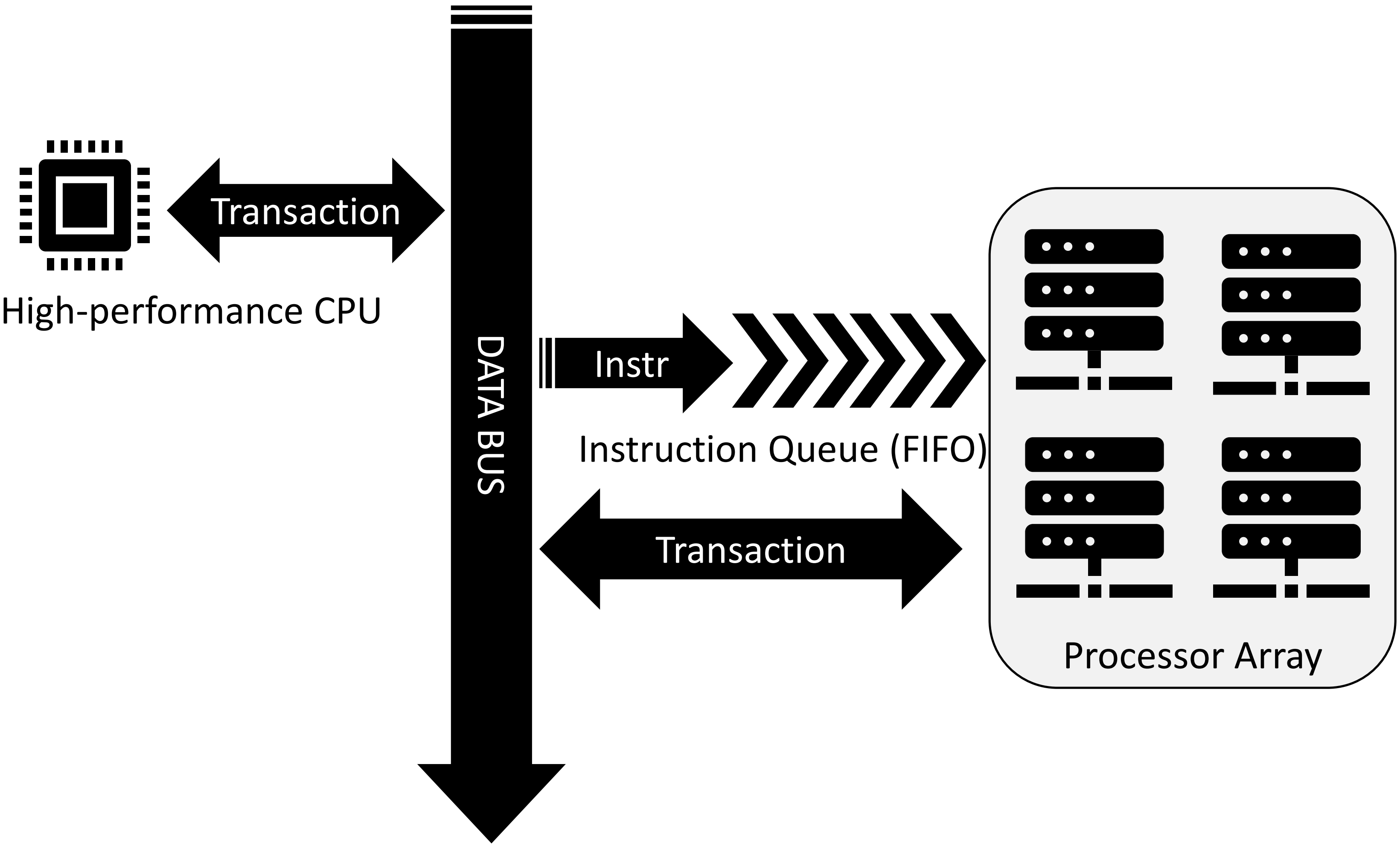
Platform Independence

- Synthesis level independence
  (IP Interfacing)
- Application level independence

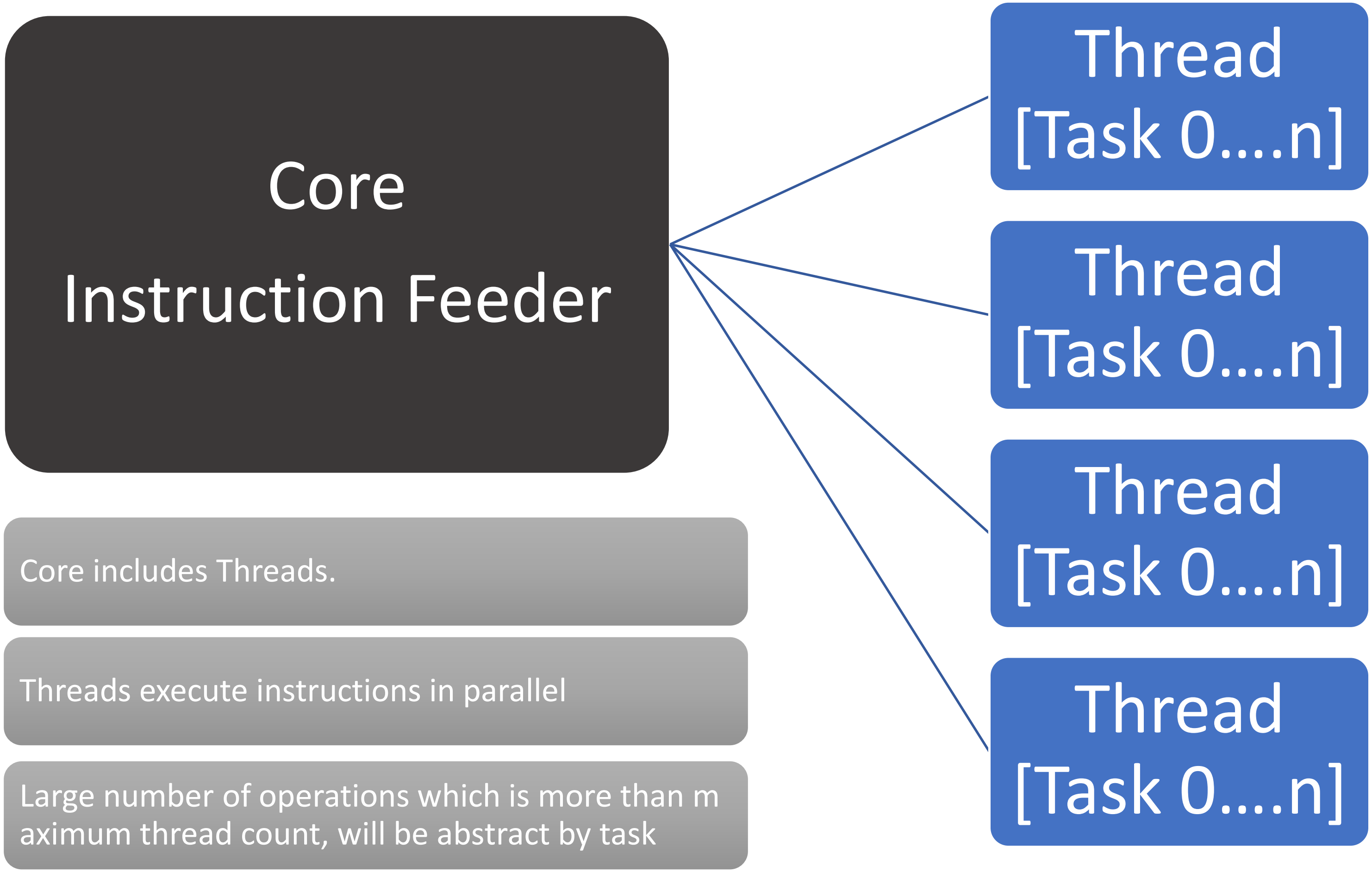
Ultra Fast Massive Data Processing

- 5 Stage pipelined architecture (Nested 2-stage pipelined FPU)
- Intensive RISC Instructions
- Can accept 200MHz input clock
  → 64 * 50MHz = 3.2GFlops per Core

Blueprint


Transaction
High-performance CPU
DATA BUS
Instr
Instruction Queue (FIFO)
Transaction
Processor Array

Core, Thread, Task

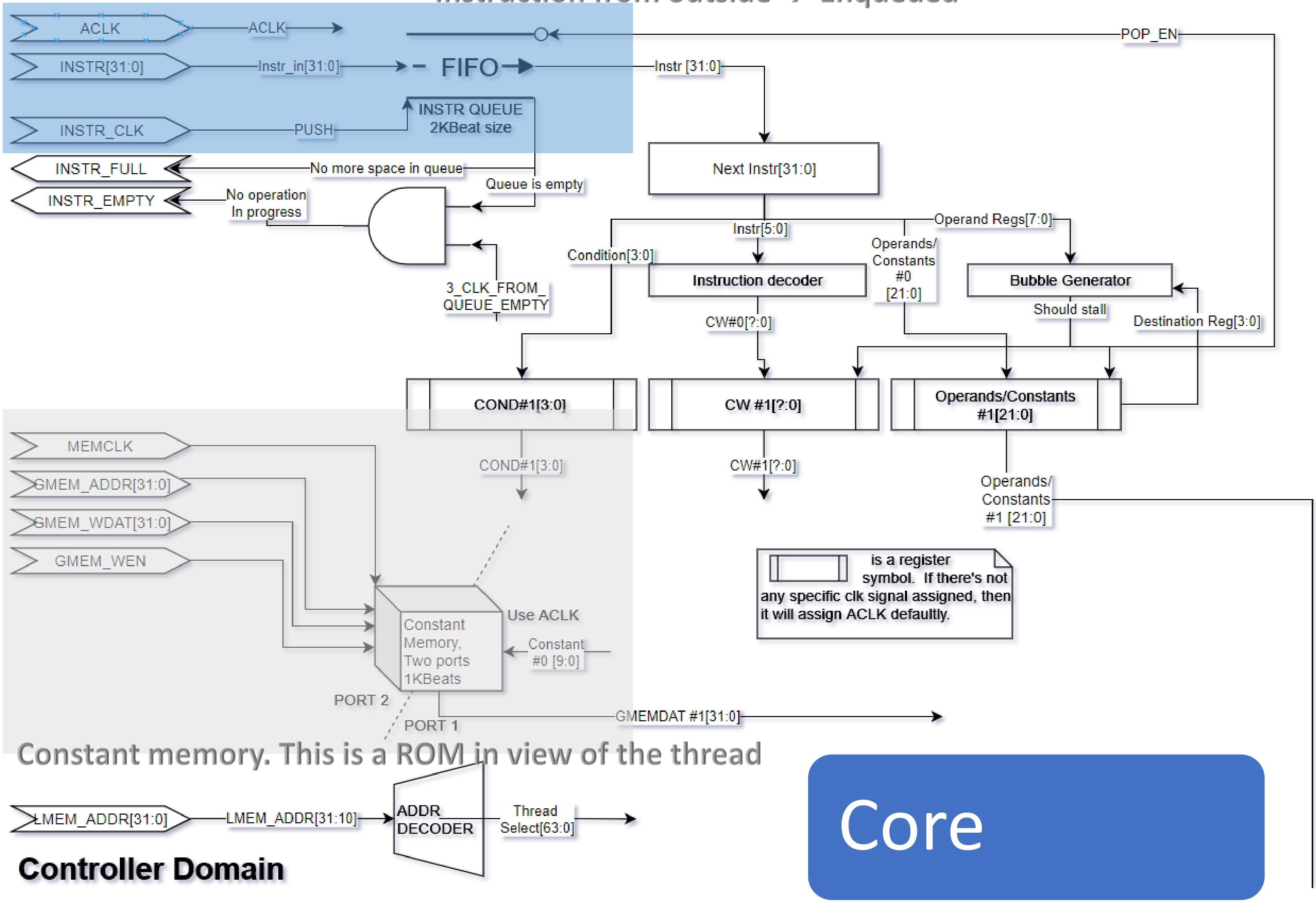


Core includes Threads.

Threads execute instructions in parallel

Large number of operations which is more than maximum thread count, will be abstract by task

Instruction from outside → Enqueued

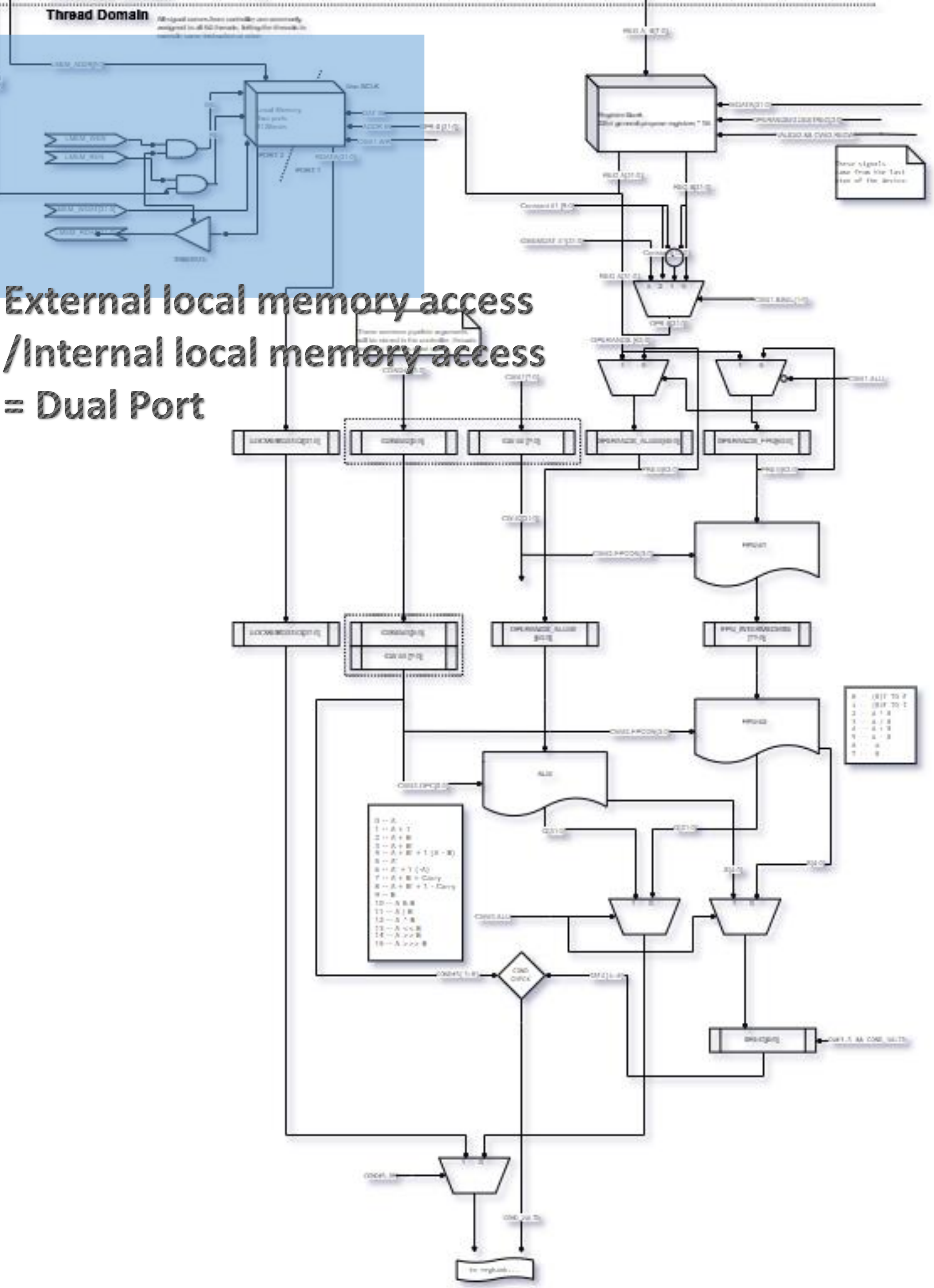


is a register symbol. If there's not any specific clk signal assigned, then it will assign ACLK defaultly.

Constant memory. This is a ROM in view of the thread

Controller Domain

Core



External local memory access
/Internal local memory access
= Dual Port

Thread

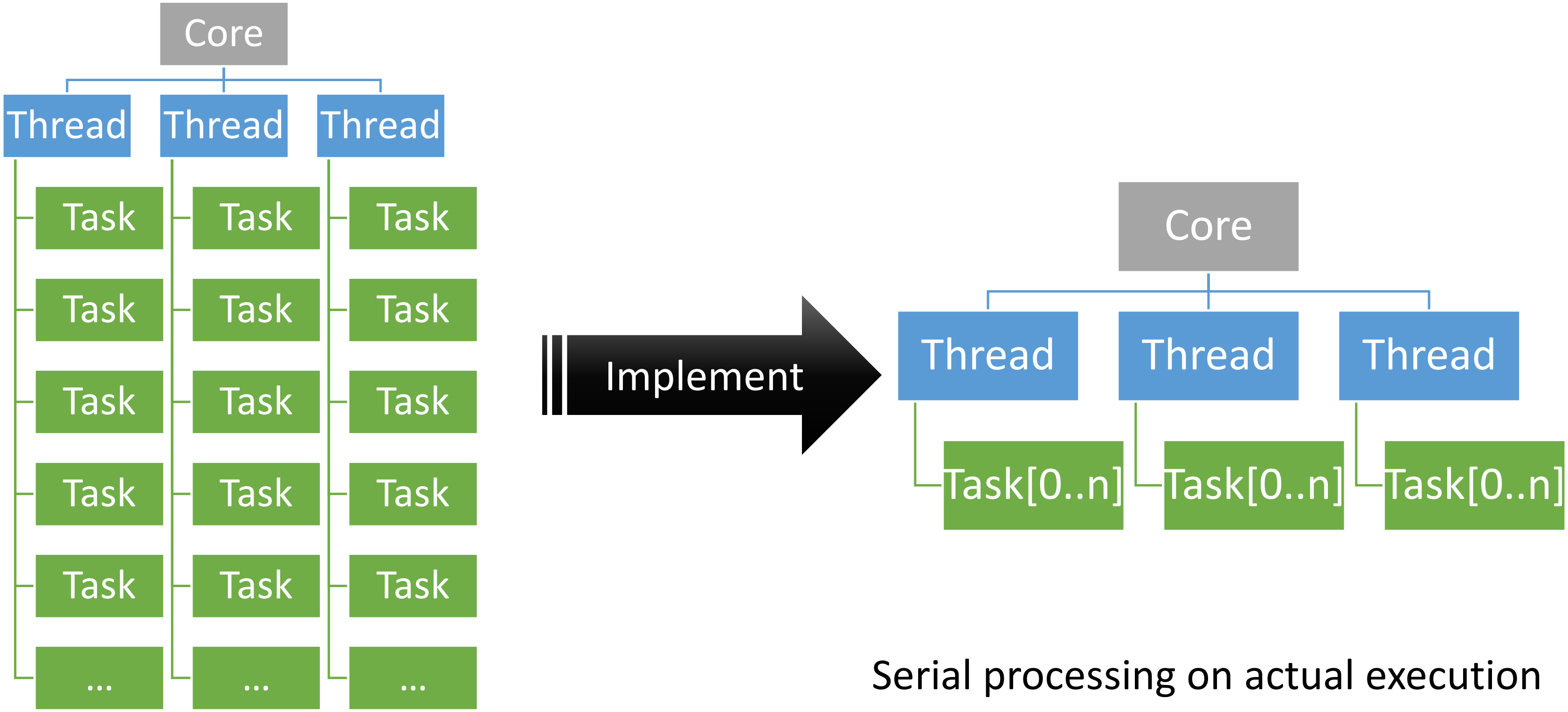
Two-stage Pipelined FPU

Nested 1KBeat local memory (DP SRAM)

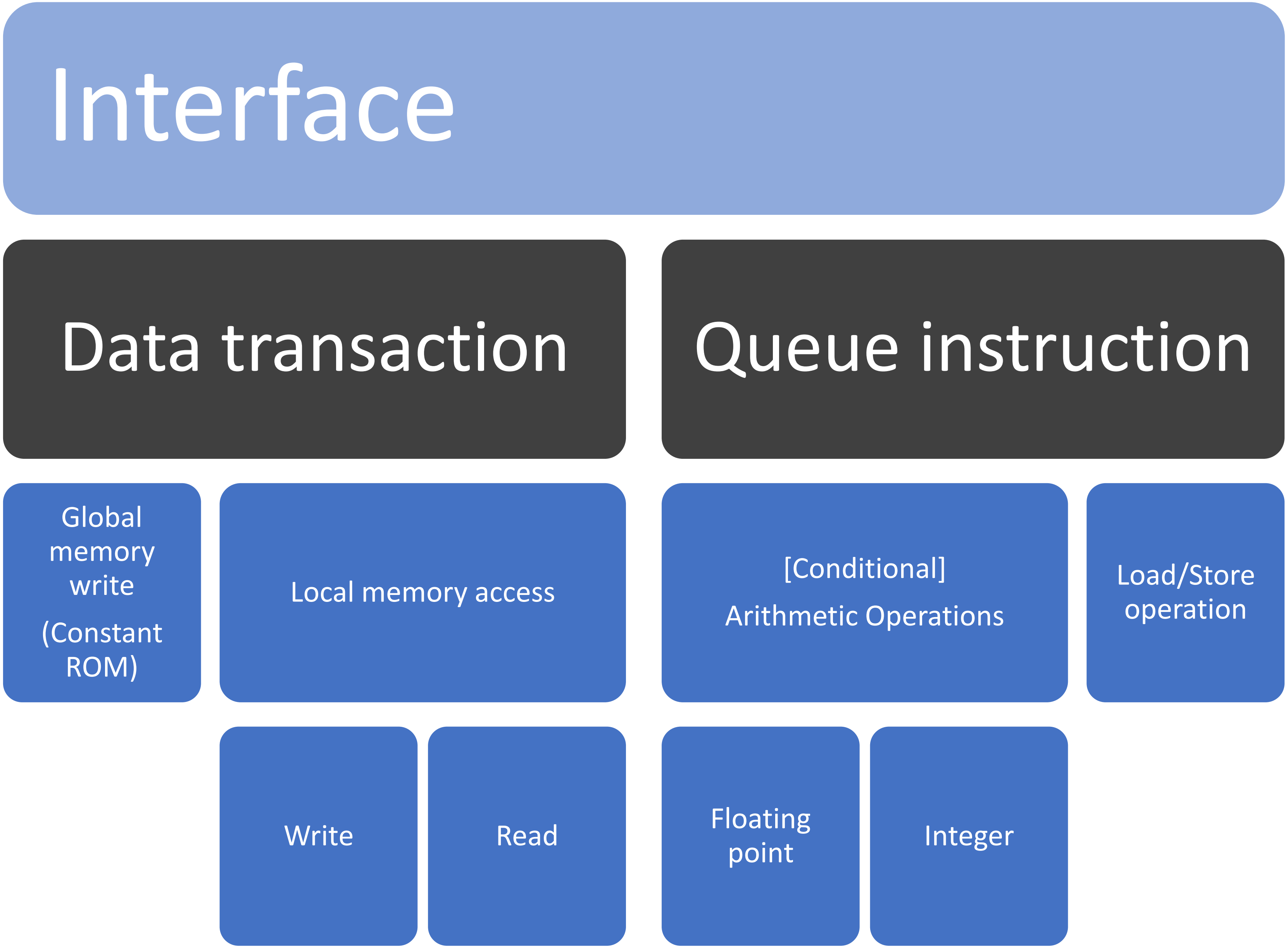
Conditional Instruction Execution

Task

Abstract parallel process



Serial processing on actual execution



Interface

Data transaction

Queue instruction

Global memory write

(Constant ROM)

Local memory access

[Conditional]

Arithmetic Operations

Load/Store operation

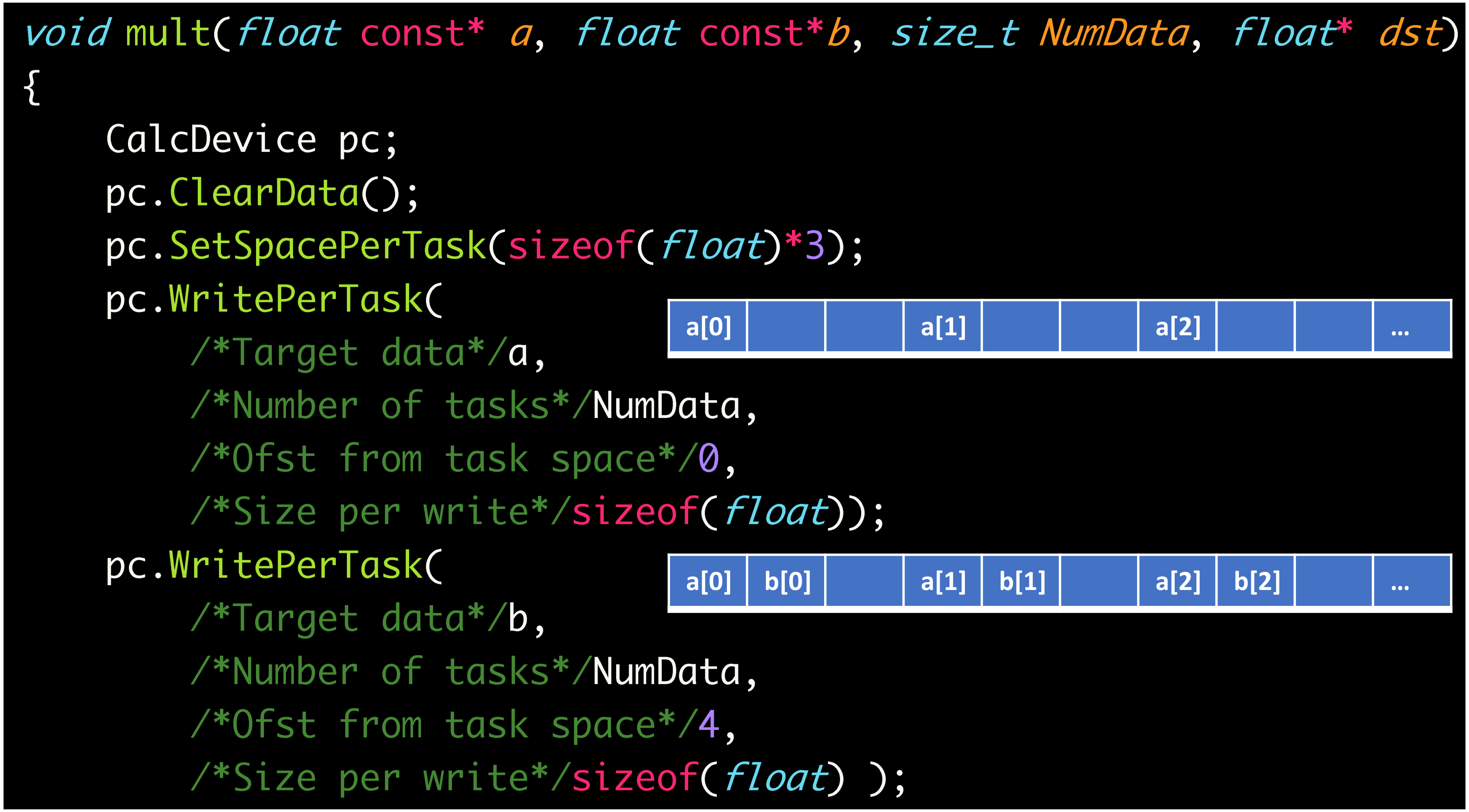
Write

Read

Floating point

Integer

Code Example



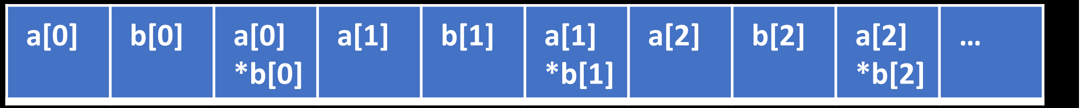
```
void mult(float const* a, float const*b, size_t NumData, float* dst)
{
    CalcDevice pc;
    pc.ClearData();
    pc.SetSpacePerTask(sizeof(float)*3);
    pc.WritePerTask(
        /*Target data*/a,
        /*Number of tasks*/NumData,
        /*Ofst from task space*/0,
        /*Size per write*/sizeof(float));
    pc.WritePerTask(
        /*Target data*/b,
        /*Number of tasks*/NumData,
        /*Ofst from task space*/4,
        /*Size per write*/sizeof(float) );
```

a[0]			a[1]			a[2]			…

a[0]	b[0]		a[1]	b[1]		a[2]	b[2]		…

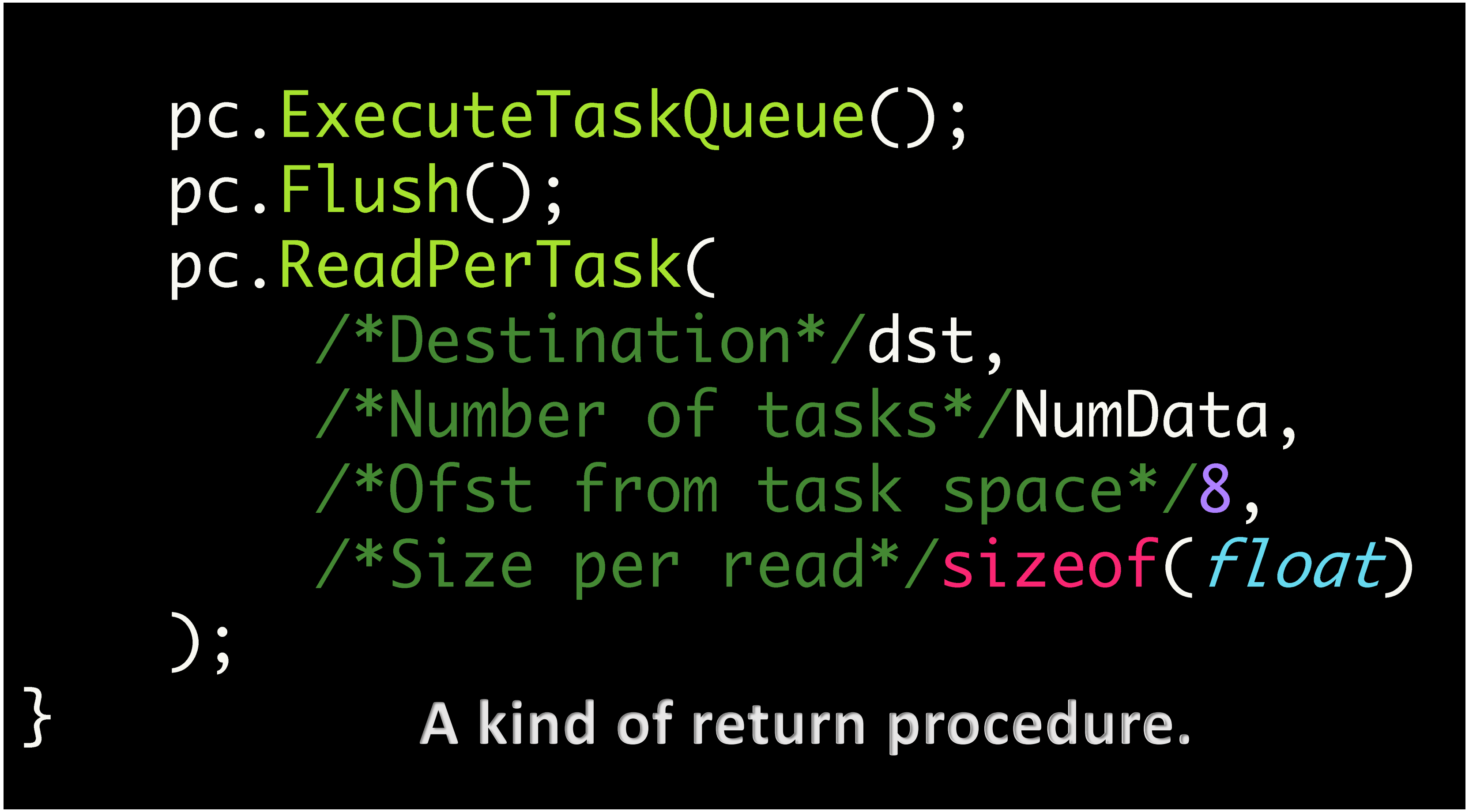
Code Example

```
pc.ClearTaskQueue();
pc.QueueTaskLoadWord(
    // This operation will execute
    //  LDR DST, [r15 + OFST]
    // per task
    /*Target Register*/ r1,
    /*Data Ofst*/0);
pc.QueueTaskLoadWord(
    /*Target Register*/ r2,
    /*Data Ofst*/4);
pc.QueueTaskFMul(
    /*Target Register*/ r0,
    /*OPR A, B*/r1,
     r2);
pc.QueueTaskStoreWord(
    /*Target Register*/ r0,
    /*Data Ofst*/8);
```



a[0]	b[0]	a[0] *b[0]	a[1]	b[1]	a[1] *b[1]	a[2]	b[2]	a[2] *b[2]	…

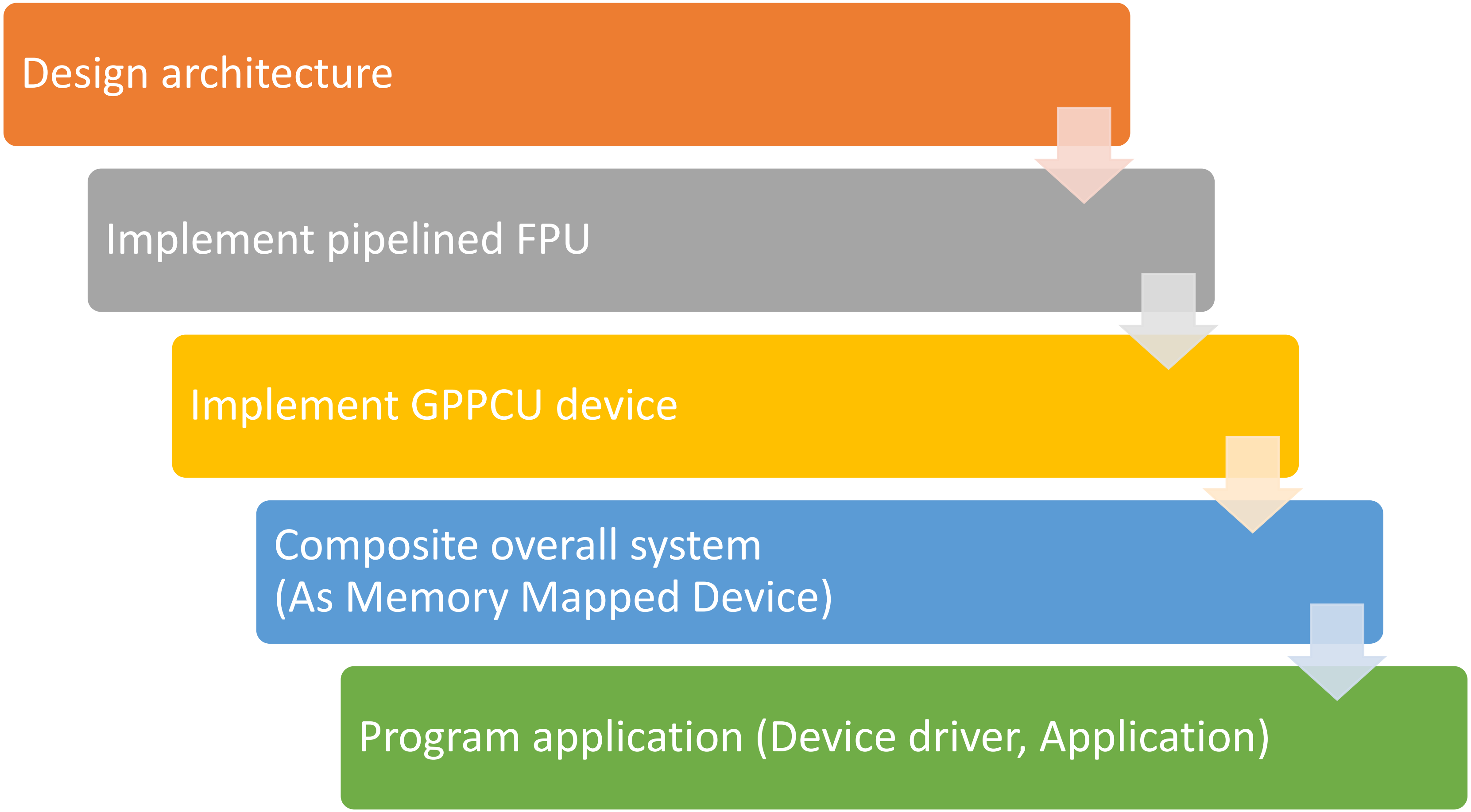
Code Example



```
    pc.ExecuteTaskQueue();
    pc.Flush();
    pc.ReadPerTask(
        /*Destination*/dst,
        /*Number of tasks*/NumData,
        /*Ofst from task space*/8,
        /*Size per read*/sizeof(float)
    );
}
```

A kind of return procedure.

Milestone



Design architecture

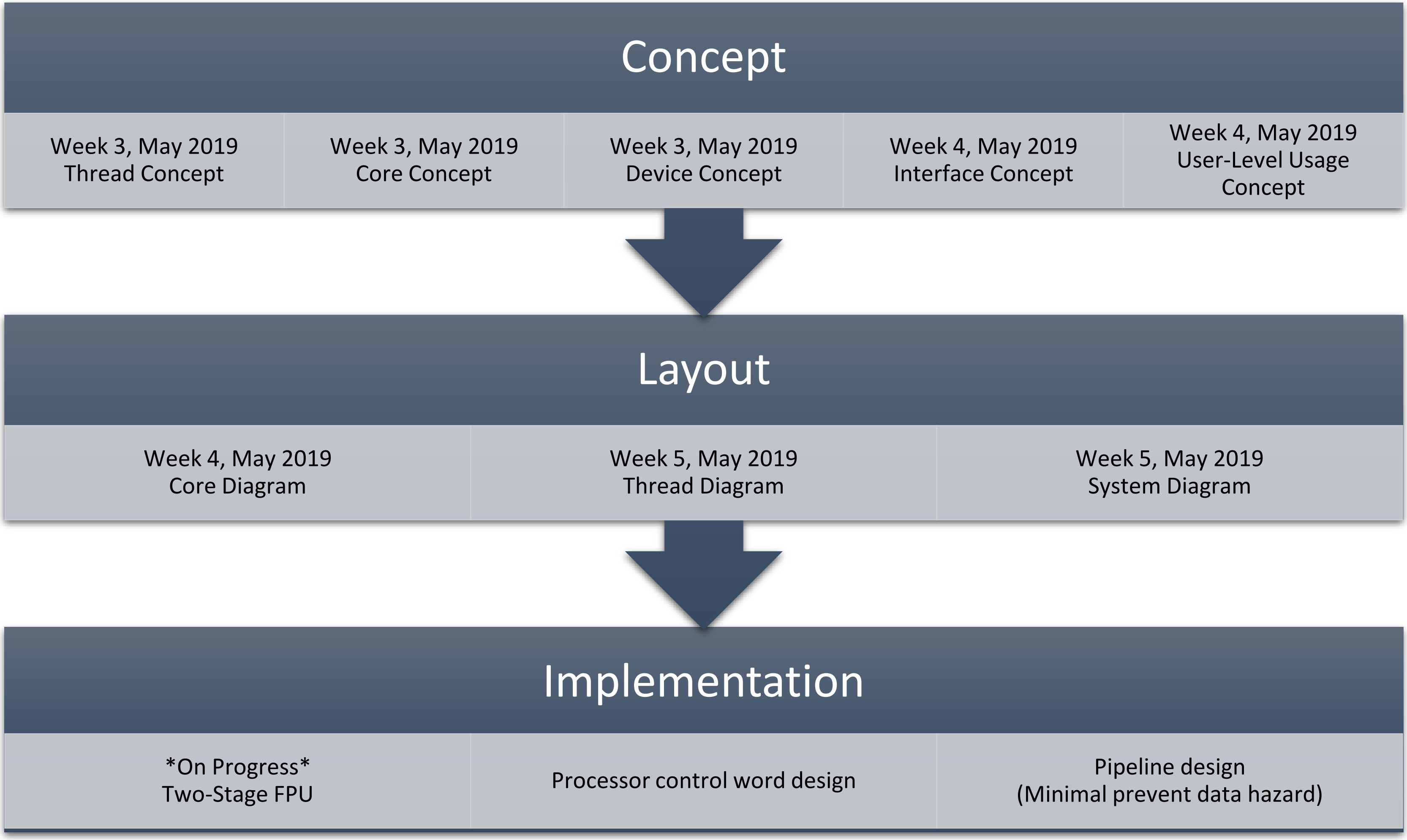
Implement pipelined FPU

Implement GPPCU device

Composite overall system
(As Memory Mapped Device)

Program application (Device driver, Application)

Progress
On Week 5, May 2019



Concept

Week 3, May 2019
Thread Concept

Week 3, May 2019
Core Concept

Week 3, May 2019
Device Concept

Week 4, May 2019
Interface Concept

Week 4, May 2019
User-Level Usage Concept

Layout

Week 4, May 2019
Core Diagram

Week 5, May 2019
Thread Diagram

Week 5, May 2019
System Diagram

Implementation

On Progress
Two-Stage FPU

Processor control word design

Pipeline design
(Minimal prevent data hazard)

Thank you

Q & A