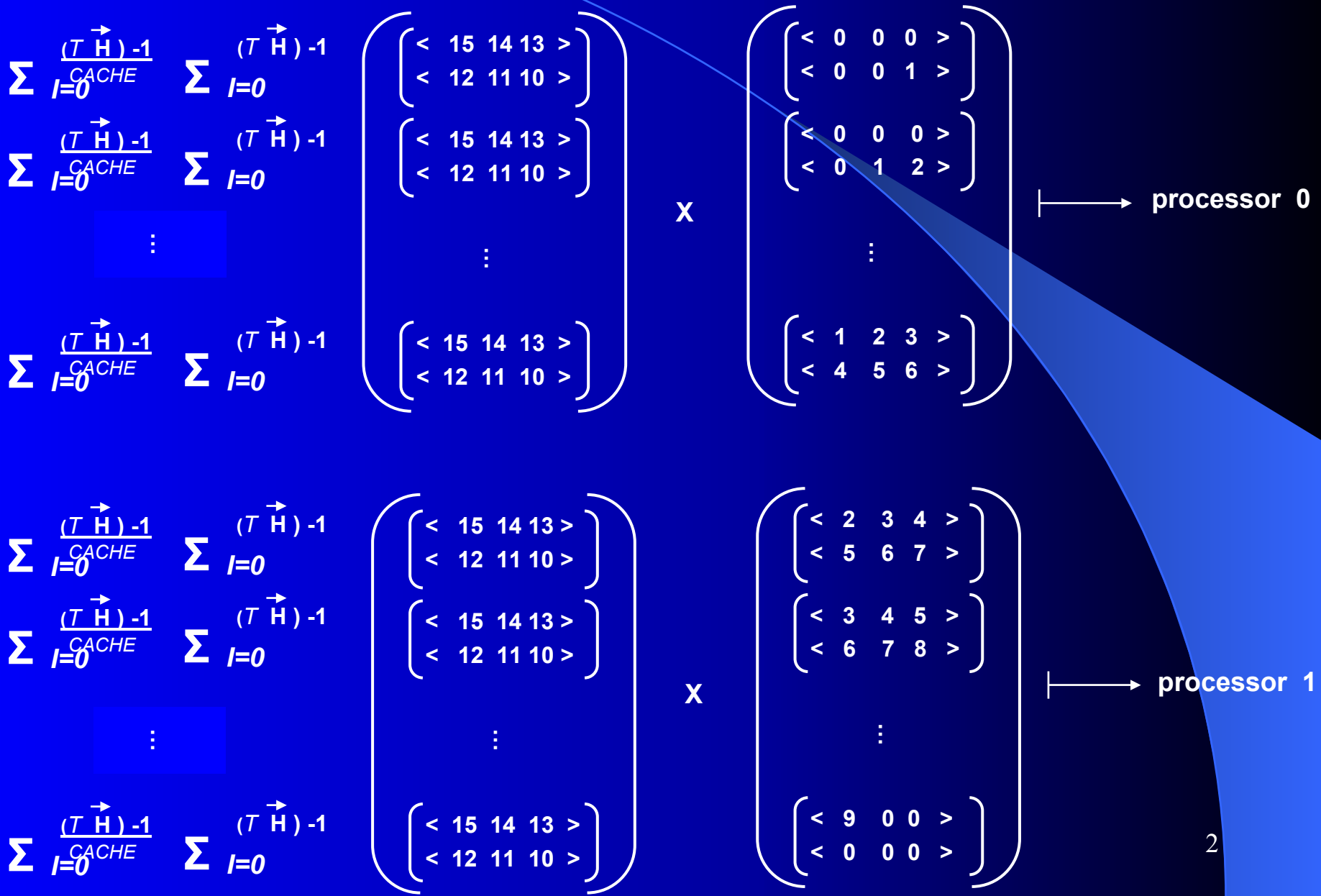
Paper: **Building the Support for Radar Processing Across Memory Hierarchies:**

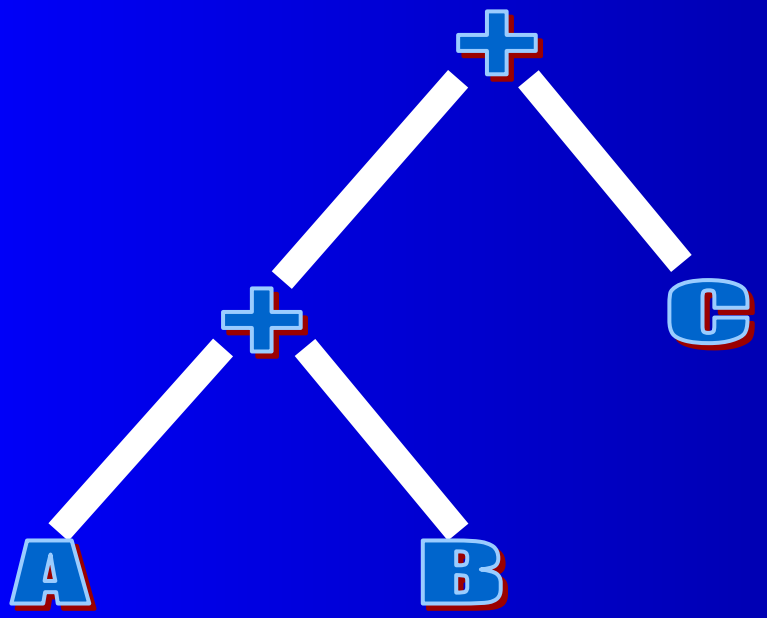
On the Development of an Array Class with Shape using C++ Expression Templates

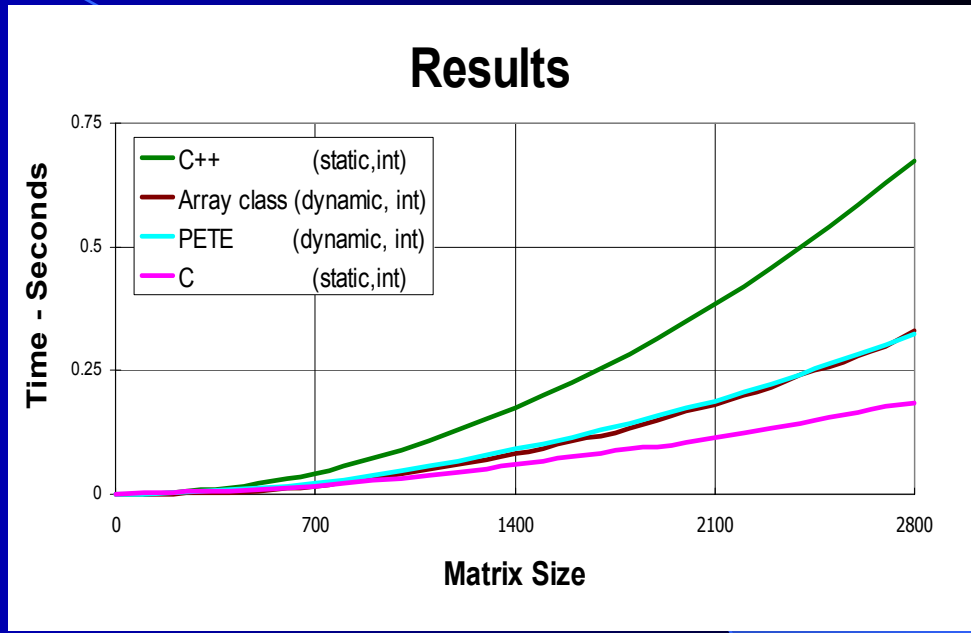
by Lenore R. Mullin, Xingmin Luo and Lawrence A. Bush

Processor / Memory Mapping



<PETE> Shape





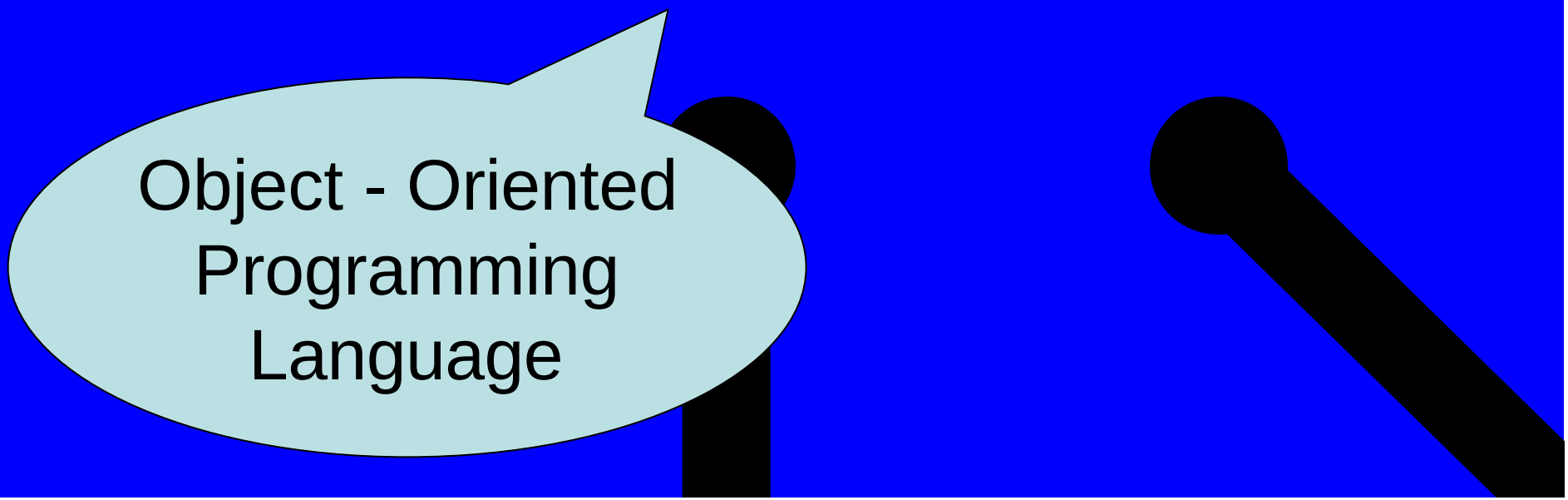
const Expression <BinaryNode<OpAdd, Reference<Array> ,

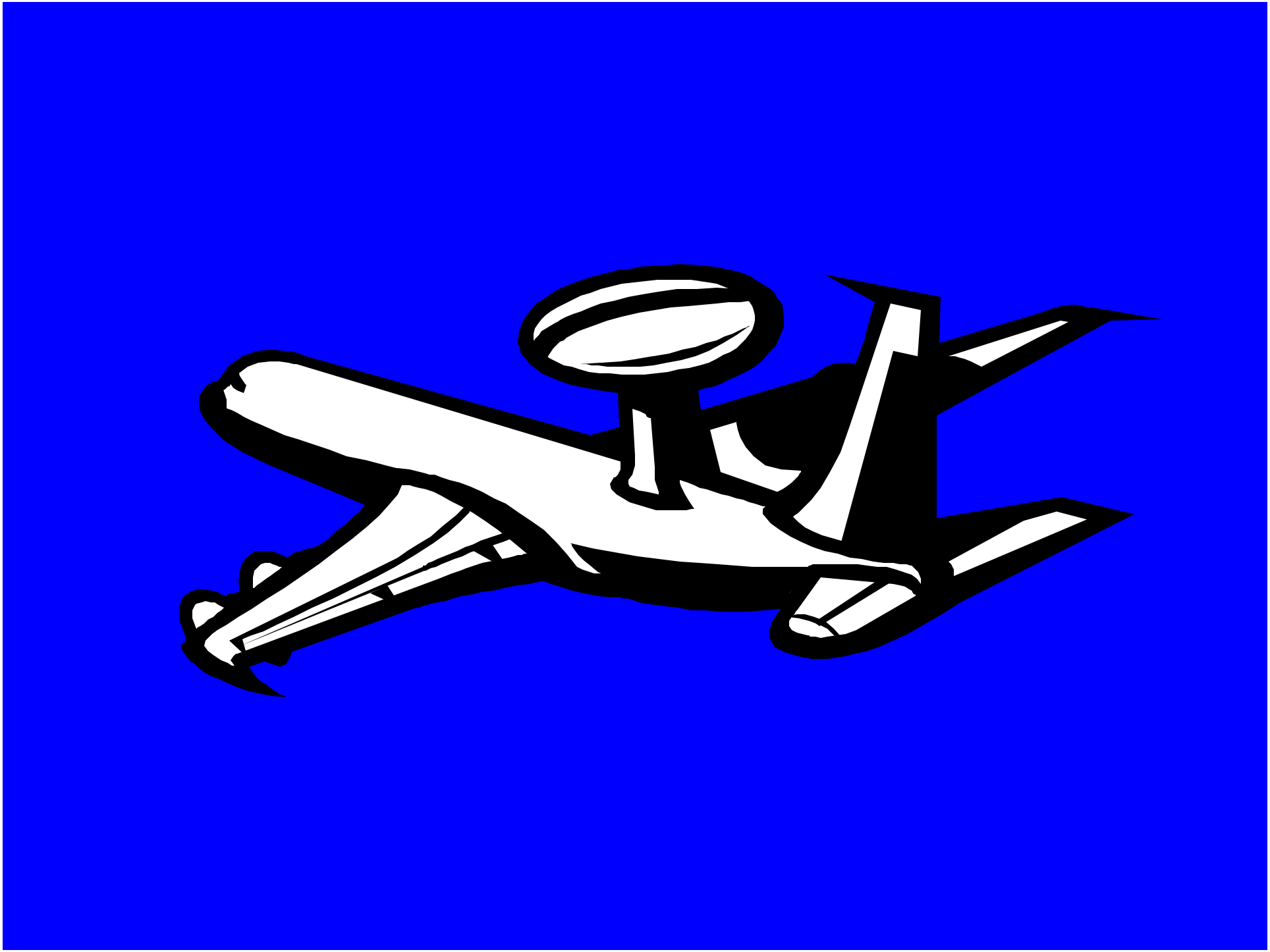
BinaryNode<OpAdd, Reference<Array> ,

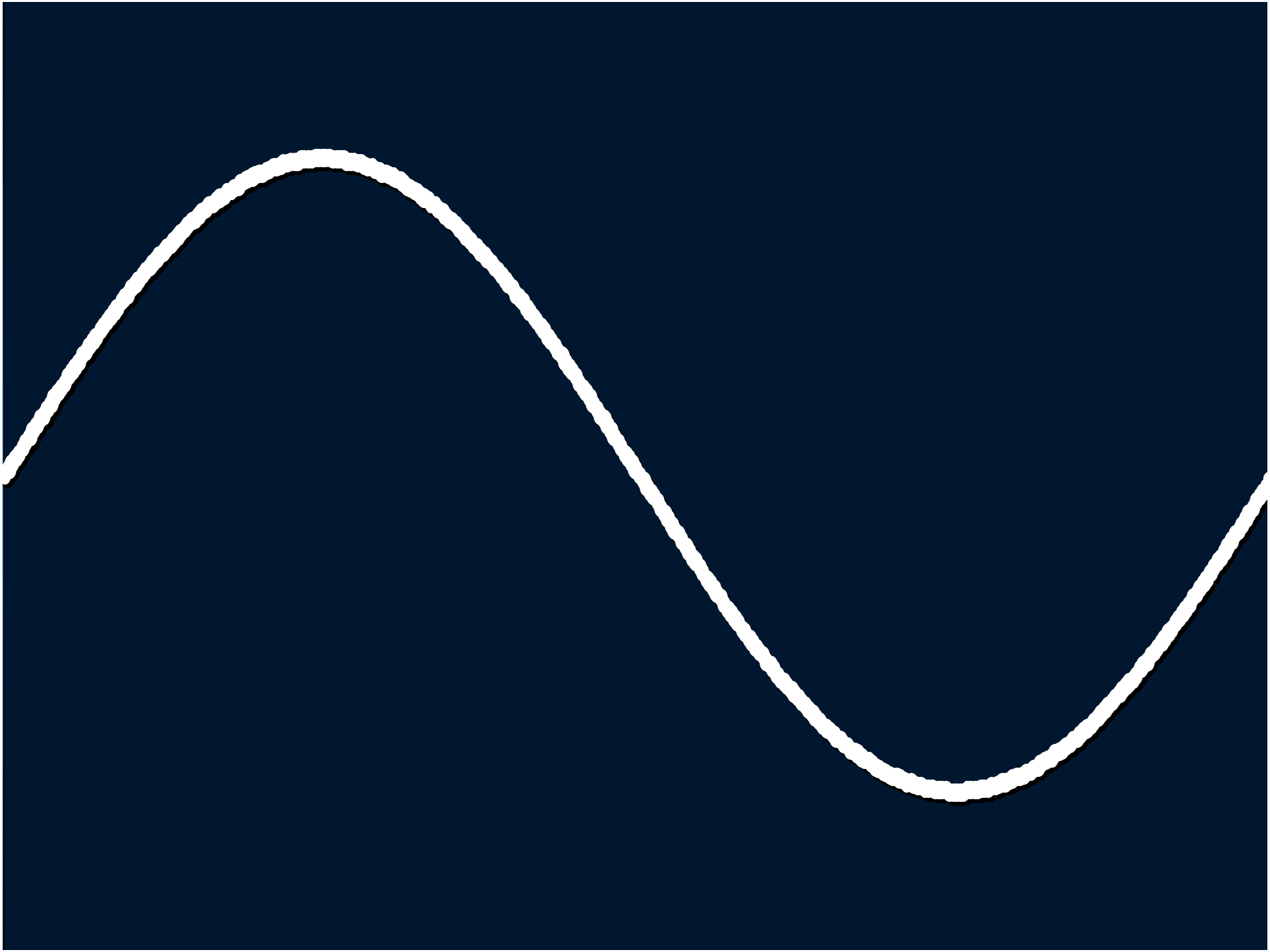
Reference<Array> > > > &expr = $A + B_3 + C$

Programmability

C++



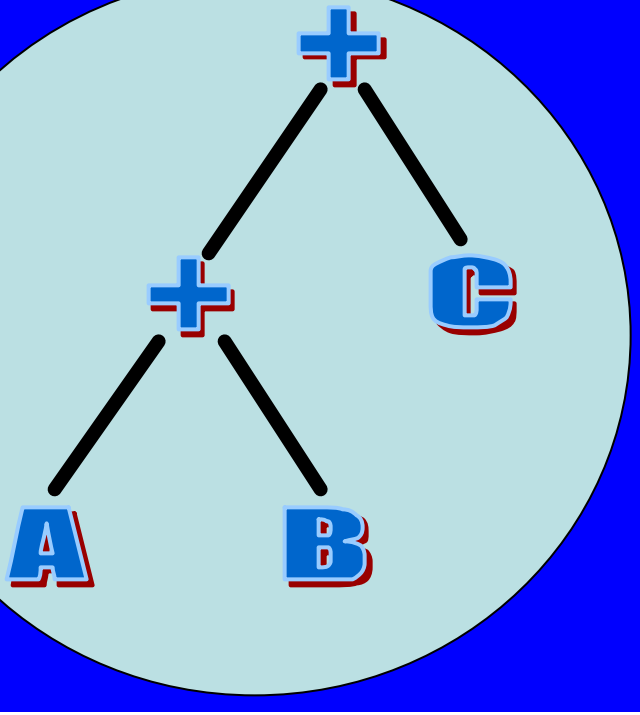


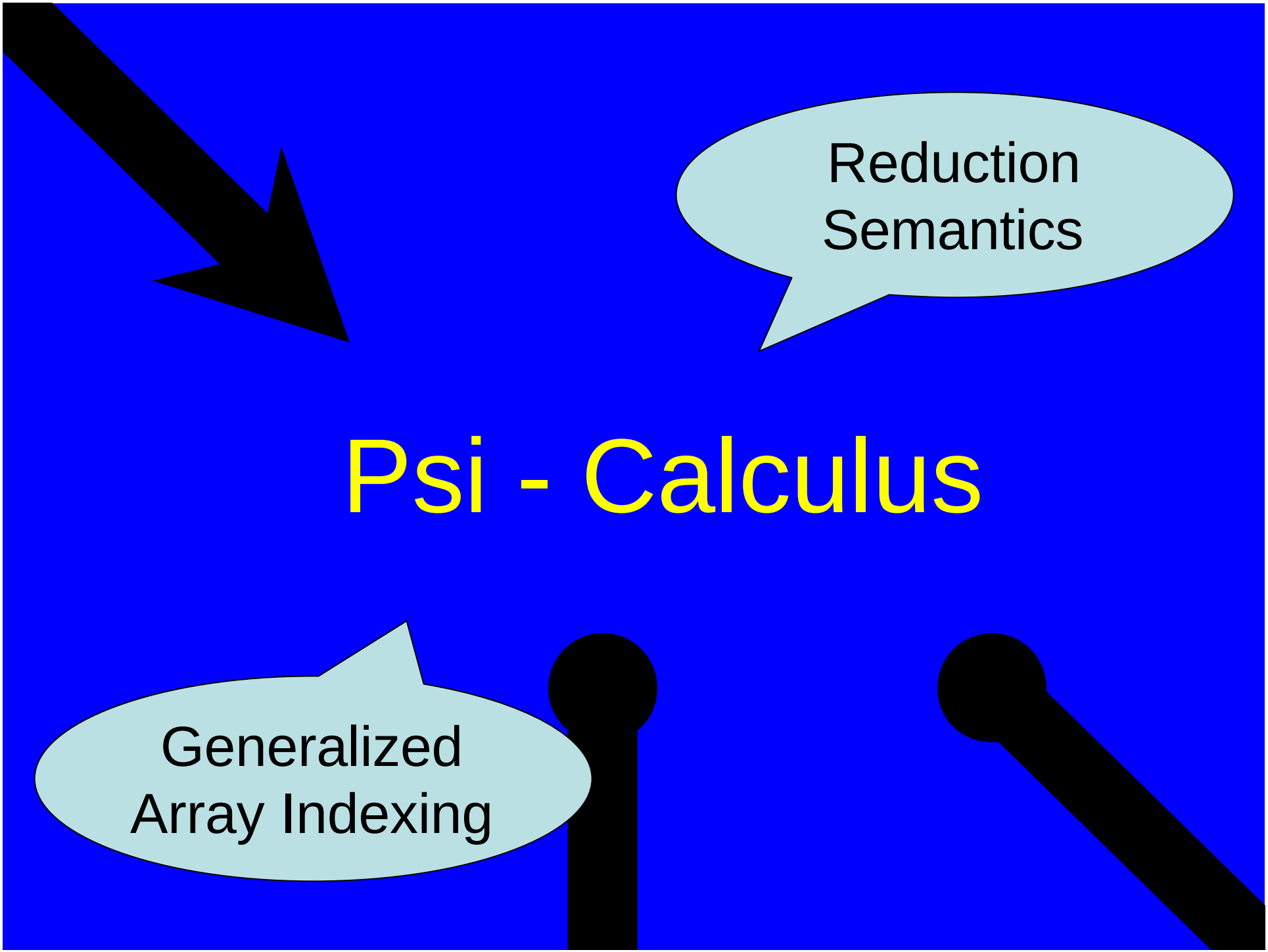


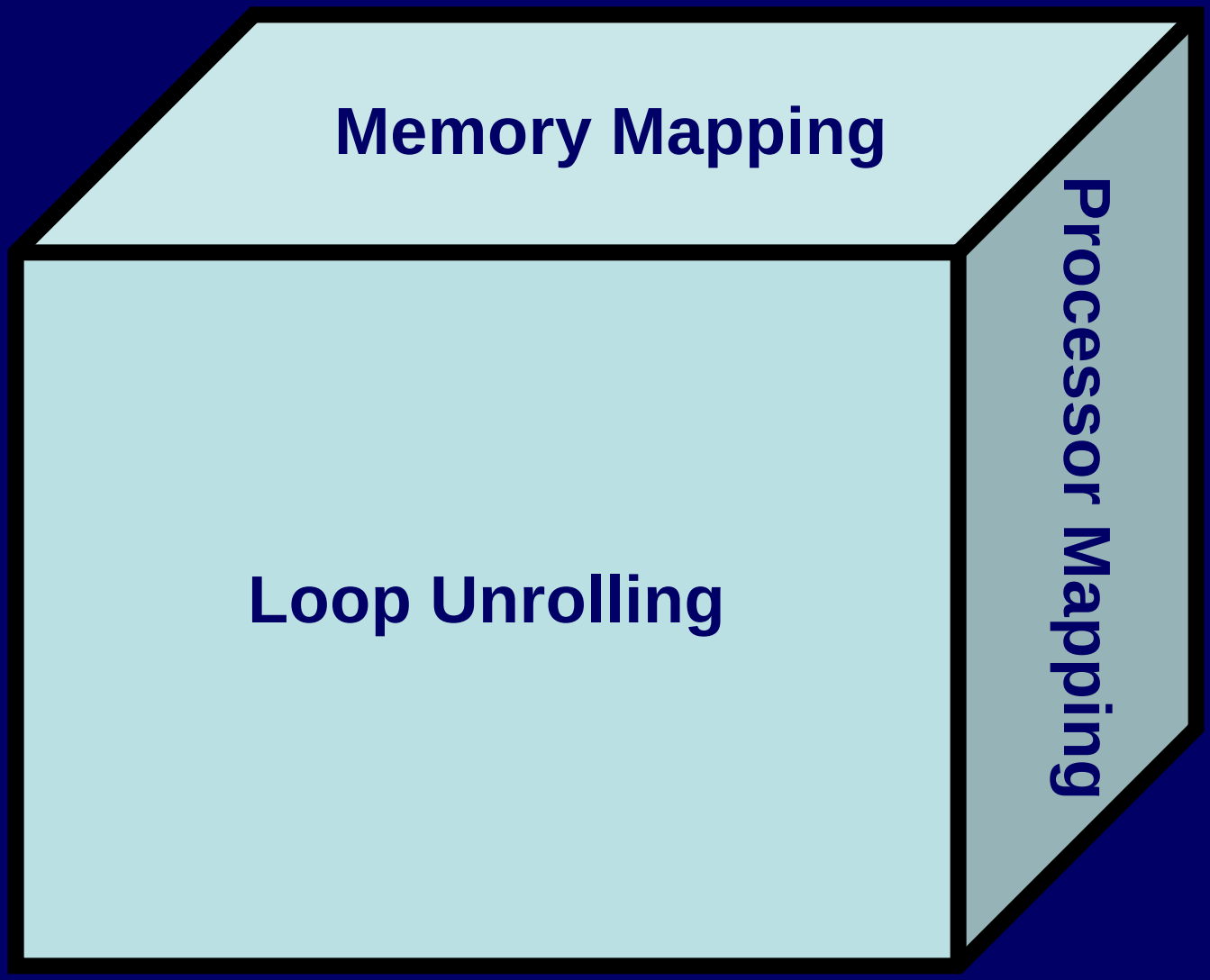
Mechanization

<PETE>

Portable
Expression
Template
Engine



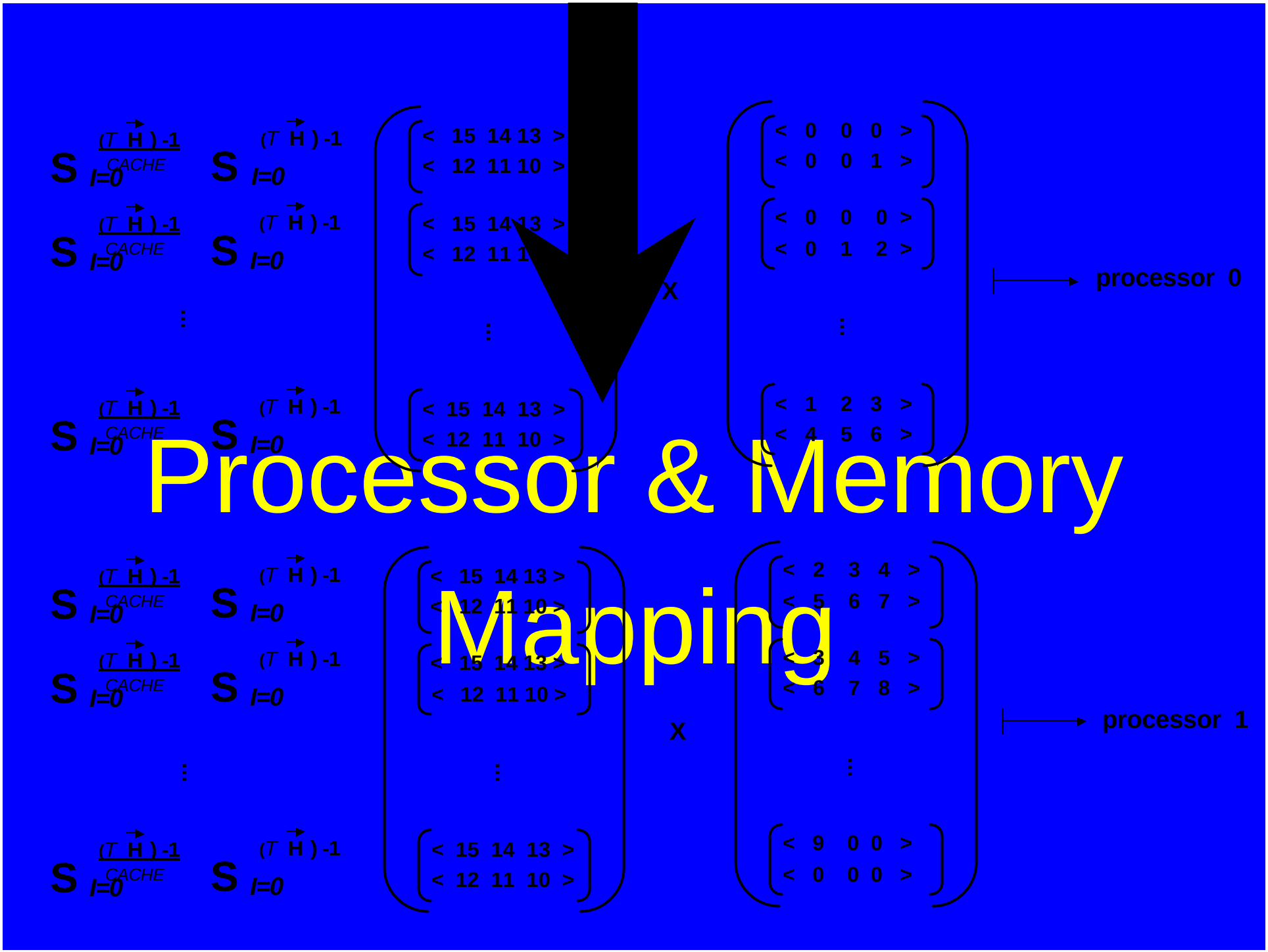

Reduction
Semantics
Psi - Calculus
Generalized
Array Indexing

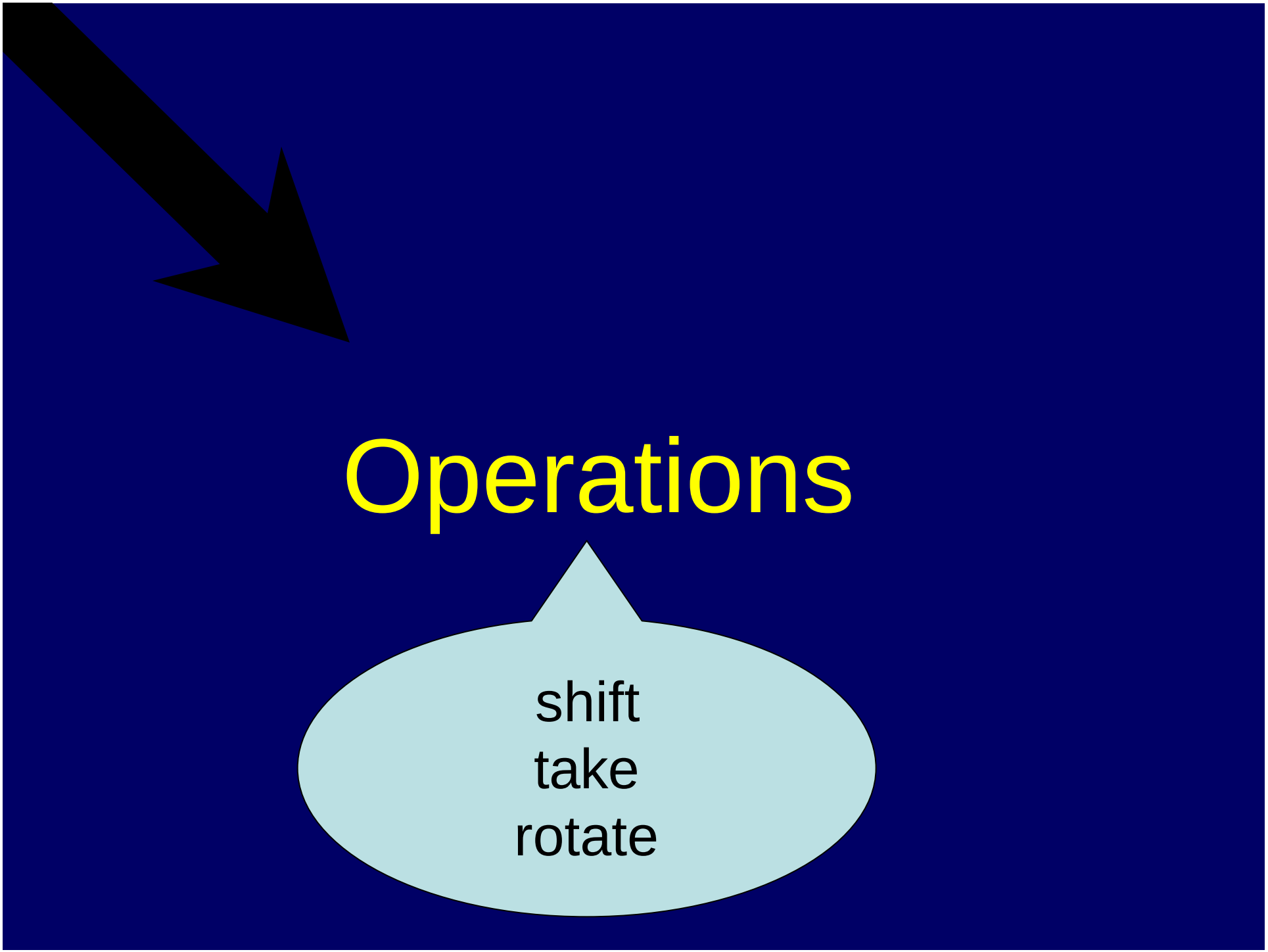

Memory Mapping
Processor Mapping
Loop Unrolling

Performance

Efficient Loops

Compile Time
Loop
Translation


Processor & Memory Mapping
processor 0
processor 1


Operations
shift
take
rotate

Array.h

```
template <class T = int>
class Array
{
    . . .
    template<class RHS>
    Array &operator=(const Expression<RHS> &rhs)
    {
        for(long i=0; i<this->size; i++)
            d[i] = forEach(rhs, EvalLeaf1(i), OpCombine());
        return *this;     //equivalent to: a.d[i] = b.d[i]+c.d[i]+d.d[i]
    }
    . . .
    private:
        T * d;
        vector <int> shape;
        long size;
}
```

Array.h

integrates with <PETE>

Psi-Calculus platform

N – dimensional capability

(required for processor / memory mapping)

Array.h

Results : Comparable to Hand Coded C