МЦСТ ЭЛЬБРУС

Hardware-Software solutions of the Elbrus platform for Supercomputers

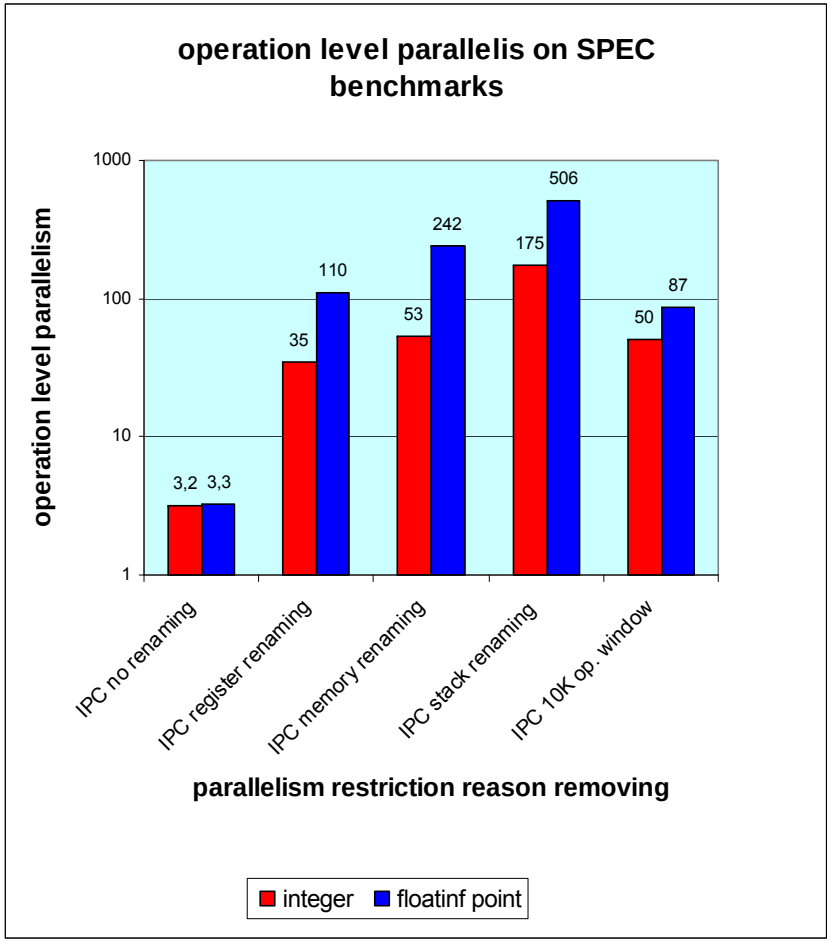
Alexander Kim, Ignat Bychkov, Vladimir Volkonskiy,Feodor Gruzdov, Murad Neiman-zade, Sergey Semenikhin, Vladimir Tikhorsky, Vladimir Feldman

JSC “MCST”

Russian Supercomputing Days, September 2016

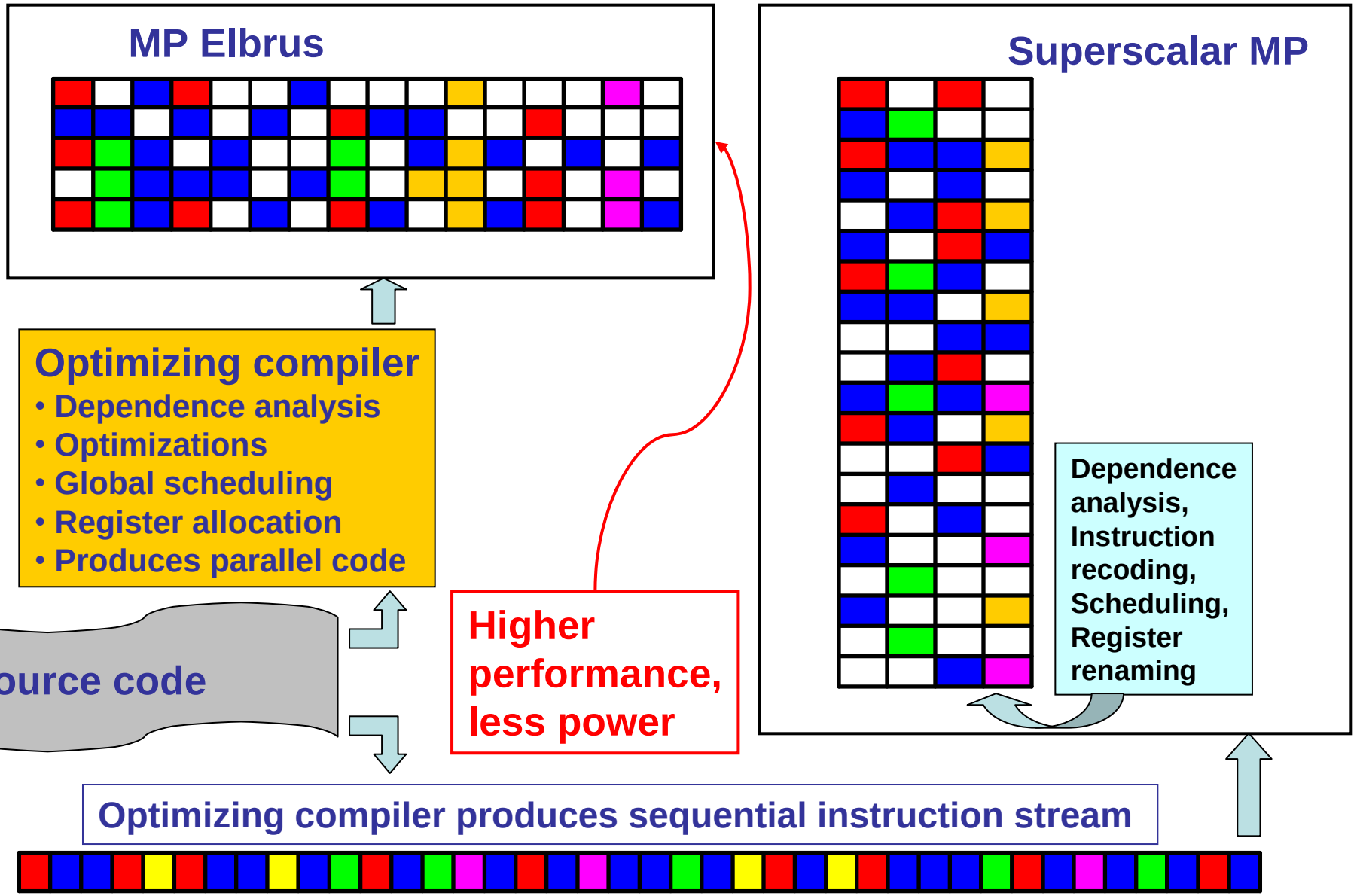
Instruction (operation) level parallelism

- Benchmark trace analysis demonstrates considerable operation level parallelism (IPC – op/cycle)
  - Integer: 81 - 240 IPC
  - Floating point: 36 - 4003 IPC
- Memory and register dependencies are main reason of parallelism restriction
  - Register renaming increases it from 3,2 / 3,3 to 35 / 110 IPC
  - Memory renaming increases it 5 times to 175 / 506 IPC
- Floating point benchmarks have more parallelism potential
  - Loops dominate
  - Vector parallelism available
- Parallelism is not local if memory renaming done
  - This demonstrates thread level parallelism



Possible operation level parallelism is huge, it can and should be utilized

Elbrus parallelism vs. superscalar



МЦСТ ЭЛЬБРУС

Deep Hardware & Software Integration in the Elbrus Architecture

- *HW architecture provides*
  - Parallel resources by wide instruction (VLIW-like)
    - Up to 25 scalar operations per cycle per core
    - Up to 12 Flops DP (24 packed SP) per cycle per core
      - Doubling in the Elbrus-8CV
    - Multicore
    - Multiprocessor support (ccNUMA)
  - Large-scale register file
  - Optimization supporting features
  - Binary compatibility supporting features
  - Secure program execution supporting features
- *Compilers and OS provides*
  - Program parallelization by optimizing compiler
    - Instruction level parallelism (many operations per cycle)
    - Packed (vector) operation parallelism
    - multicore, multithreading parallelism
  - Viable binary compatibility with Intel x86, x86-64 on the basis of transparent dynamic binary compilation technology
  - Programming languages implementation for secure program execution

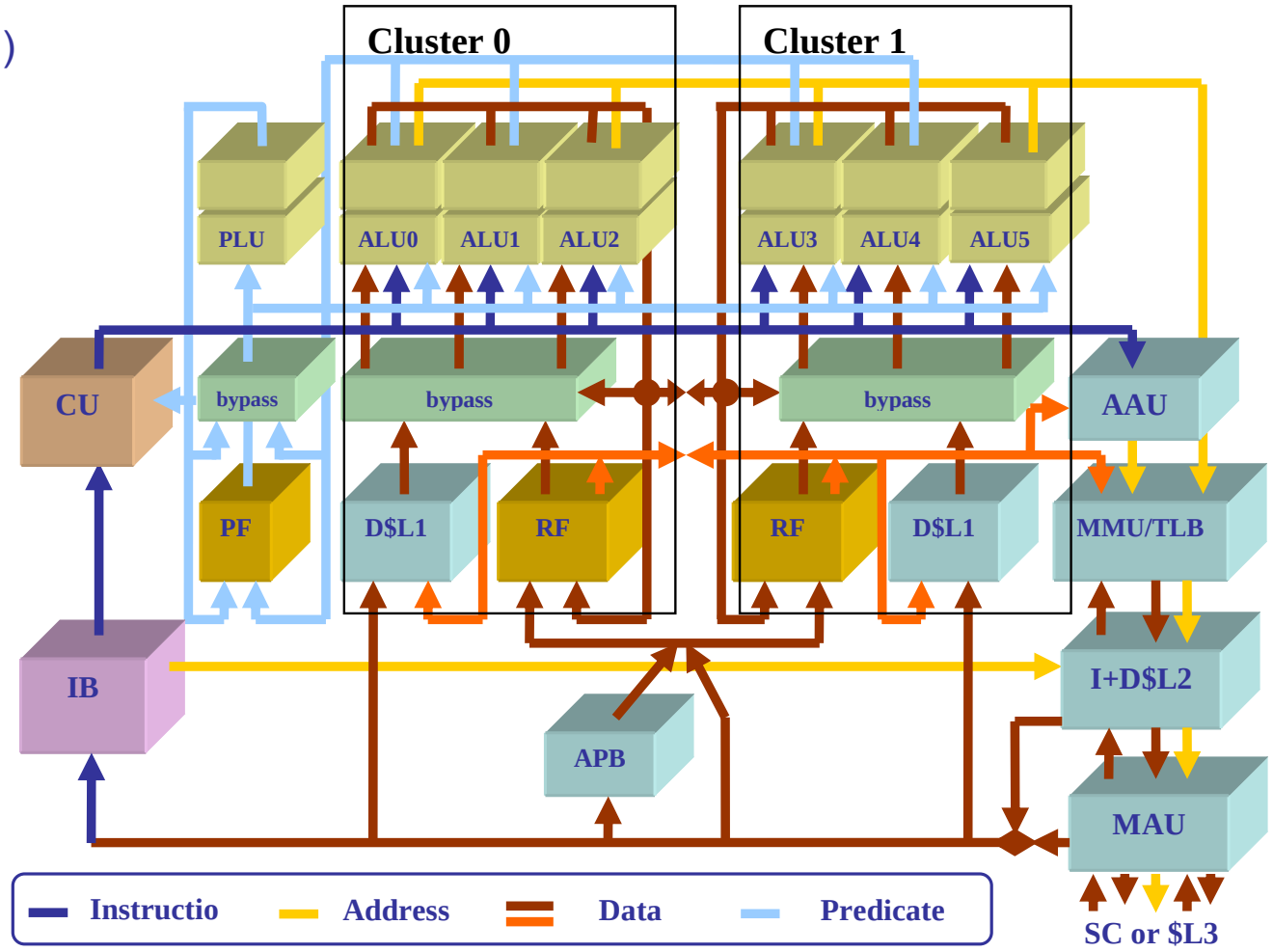
Parallelism provided by integration of HW & SW
- Instruction (RISC operation) level, packed (vector) level, multicore, multithreading (TLP)

МЦСТ ЭЛЬБРУС

Elbrus core structure

2 Clusters

IB –instruction buffer (I$L1)
CU – control unit
PF – predicate file
PLU – predicate logic unit
D$L1 – L1 data cache
ALU1-6 – arithmetic and logic units
RF – register file
APB – array prefetch buffer
AAU – asynchronous array access unit
MMU – memory management unit
TLB – translation look aside buffer
I+D$L2 – L2 cache
MAU – memory access unit
SC – system controller
$L3 – L3 cache



МЦСТ ЭЛЬБРУС

Optimization and parallelization compiler methods

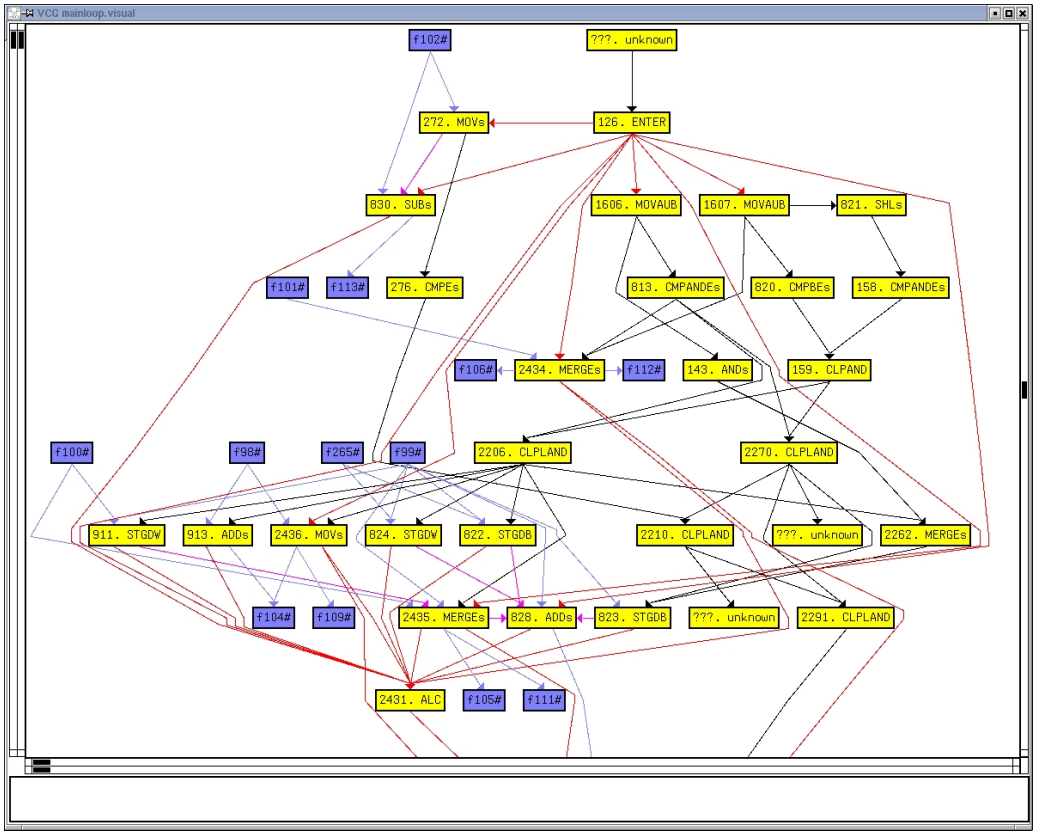
Implemented in Elbrus optimizing compiler

- Profiling
  - static (two phase compilation)
  - dynamic (binary compiler implemented, native compiler in process)
- Special optimization heuristics for classified program regions
- Program feedback analysis in region optimization heuristics
- Dynamic and static dependence analysis
- Global inter-procedural analysis and parallelization
  - Instruction (operation) level parallelism
  - Vector parallelism (packed instruction utilization)
  - Thread level parallelism including OpenMP and MPI
  - Distributed memory parallelism including MPI
- Efficient automatic ILP and Vector parallelization, TLP with OpenMP/MPI, DMP with MPI

МЦСТ ЭЛЬБРУС

Case study: Integer program parallelization

```
LOCAL void sweep(void)
{
    struct segment *seg;
    NODE *p;
    int n;
    /* empty the free list */
    fnodes = NIL;
    nfree = 0;
    /* add all unmarked nodes */
    for (seg = segs; seg != NULL; seg = seg->sg_next) {
        p = &seg->sg_nodes[0];
        for (n = seg->sg_size; n--; p++)
            if (!(p->n_flags & MARK)) {
                switch (ntype(p)) {
                case STR:
                        if (p->n_strtype == DYNAMIC && p->n_str != NULL) {
                            total -= (long) (strlen(p->n_str)+1);
                            free(p->n_str);
                        }
                        break;
                case FPTR:
                        if (p->n_fp)
                            fclose(p->n_fp);
                        break;
                case VECT:
                        if (p->n_vsize) {
                            total -= (long) (p->n_vsize * sizeof(NODE **));
                            free(p->n_vdata);
                        }
                        break;
                }
                p->n_type = FREE;
                p->n_flags = 0;
                rplaca(p,NIL);
                rplacd(p,fnodes);
                fnodes = p;
                nfree++;
            }
            else
                p->n_flags &= ~(MARK | LEFT);
    }
```



Source code

Compiler IR for loop

```
fc050446 98c11d24 a01dd460 90c00826 9208c106 a229c141 802bd028 001ffffc 61630263 44410000
fe471467 80003000 a026c046 900bcc09 260bc822 8e220b20 8e28c022 240bc080 19231001 10020000 61634068 08400000 cc644440 50641064
fe4f1477 806f05e7 9017c115 90c01715 240bc122 a224d042 8e162114 260bc480 00000000 00000540 00004ae3 61636326 68666064 c8624840 0d640c40 12695064
```

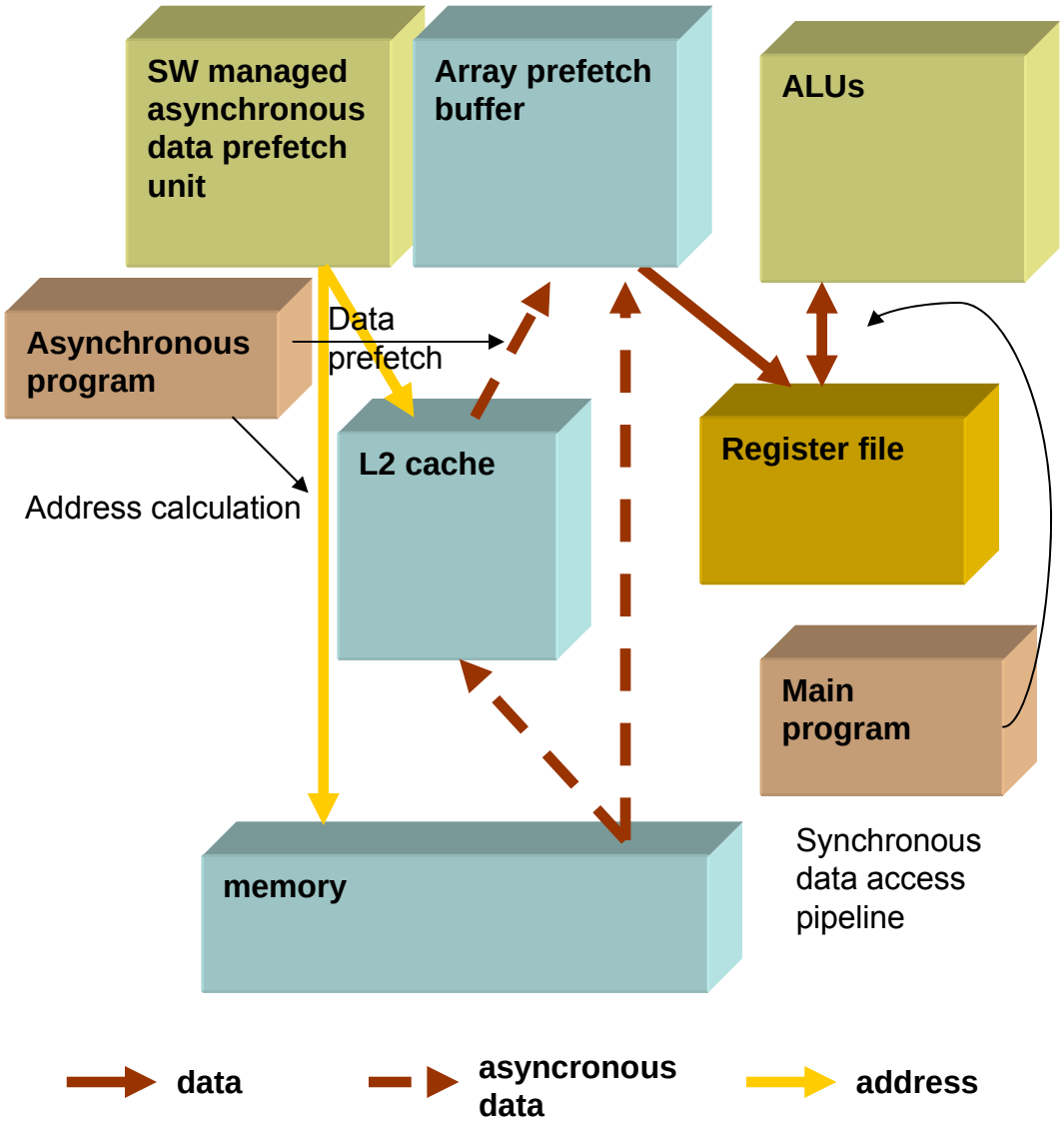
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Resulting parallel code for the loop – up to 16 op/cycle

Compiler provides good ILP utilization for Elbrus Platform

SW managed asynchronous data prefetch

- Main features
  - Removes from main pipeline address and memory access calculation
  - Minimizes stalls by memory access pipeline
  - No cache pollution
  - Increase memory access parallelism
  - Minimize live time of registers
- Applicable to both high performance and big data tasks



Considerably reduces memory access stalls

МЦСТ ЭЛЬБРУС

Performance Library

1000+ strongly optimized functions

- Vector
  - Arithm. functions, Math functions, Statistics, etc.
- Algebra
  - BLAS1/2/3, LAPACK (subset), SpBLAS (in process), etc.
    - DGEMM – 97% of peak performance
    - HPL – 82% of peak performance (84 GFLOPS) on 4 CPU (16 cores) server
    - SpMV- 61% of peak performance (memory throughput) on 4 CPU server
- Signal
  - FFT,FHT, Auto/CrossCorr, Conv, IIR, FIR, LMS, etc.
- Image
  - DFT, Auto/CrossCorr, Statistics, Filtering, Color, Geometry, etc.
- Video
  - I/DCT, De/Quantize8x8, SumAbsDiff, Interp/InterpAve/X/Y/XY, etc.
- Graphics (2D)
  - Draw/Fill (Point, Line,Triangle, Rectangle, Polygon, Circle, Arc, Ellipse), Anitaliasing, Alpha Blending, Gourad Shading, Z Buffering, Xor, etc.
- Volume (3D data)
  - RayCast_General/Blocked_Parallel/ Divergent_Nearest/Trilinear, etc.

МЦСТ ЭЛЬБРУС

Operating System Elbrus

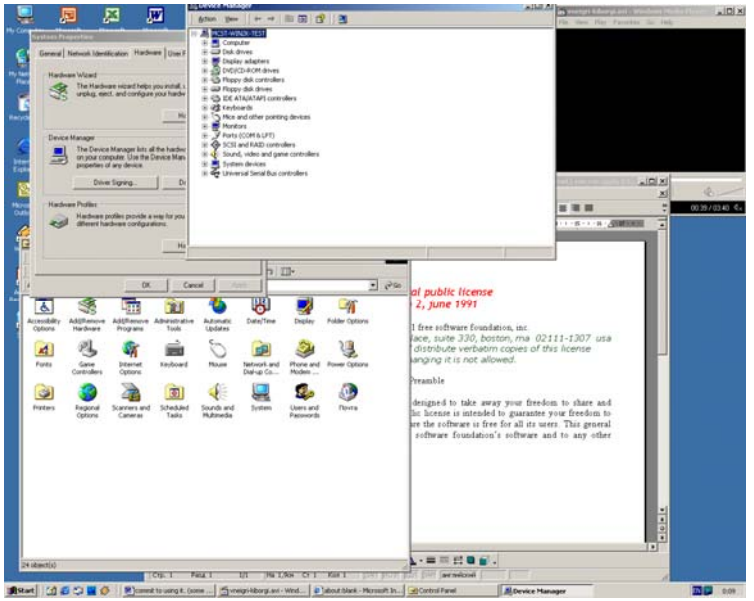
- Elbrus OS kernel based on OS Linux kernel
  - Real time mode support
  - Elbrus technologies support
    - Binary compatibility for Linux applications in Intel x86 codes
    - Efficient secure execution of programs
- Software development kit
  - Optimizing compilers (C, C++, Fortran, Java), linker, debugger, profiler, math libraries
    - Program parallelization
      - MPI, OpenMP, automatic parallelization for ILP, vectorization, multithreading
    - Performance libraries
  - Open source software stack
    - Compatibility with GCC features

- Operating system user package
  - Utilities, services, general purpose libraries
  - Graphics subsystem, network, databases, office package
  - Cluster resource management
    - slurm, irqbalance, torque, ganglia, nfs-server, iscsi-target
  - Drivers from open-source Linux world

Other important Elbrus CPU Technologies

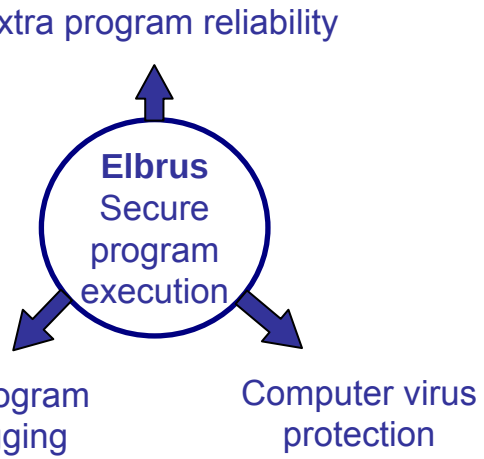
Binary compatibility with Intel x-86, x86-64 via BT

- Functionality
  - Direct execution of 20+ operating systems, including: MSDOS, Windows XP, 7, Linux, QNX, PS/2
  - Direct execution of 1000+ popular applications
  - Execution of applications under operating system Elbrus (Linux Distributive)
- Performance – up to 80% from native
  - By transparent optimizing binary translation system
  - Based on strong and powerful hardware support
- Independent from Intel license



Secure program execution technology

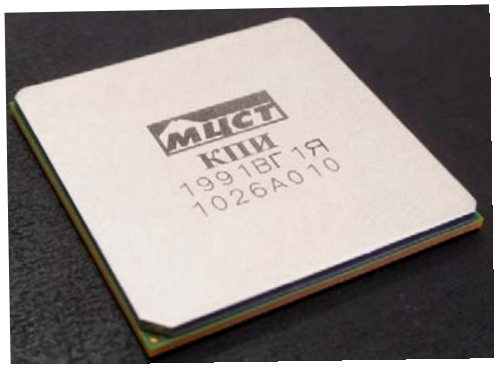
- Memory and data protection
  - Structured memory
  - Object access by descriptors
  - Scopes access supported
- Critical vulnerabilities detection
  - Buffer overflow
  - Uninitialized data access
  - Dangling pointer access



MP Elbrus-2C+, Elbrus-4C, south bridge KPI-1

Elbrus-2C+:

- TSMC 90 nm process, 10 metal layers
- 0.5 GHz clock frequency
- Power - 25 W
- Chip structure
  2 Elbrus architecture cores,
  4 DSP Multicore architecture cores
- Total performance -
  28/8 Gflops sp/dp:
  2 Elbrus cores – 16/8 Gflops sp/dp,
  4 DSP cores – 12 Gflops sp
- Die size - 17,2x16,8 mm
- Sampling 2011
- Production 2012H1



South bridge KPI-1:

- TSMC130 nm process, 9 metal layers
- 250 MHz clock frequency
- Power – 5 W
- 14 interfaces provides:
  system, PCI Express, PCI, Ethernet (10/100/1000), SATA 2.0, USB 2.0, RS 232/485, etc.
- Die size – 10,6x10,6 mm
- Sampling - 2010
- Production – 2011H1

Elbrus-4C:

- 4 Elbrus architecture cores
- 8 MB L2 cache (2 MB per core)
- TSMC 65 nm process
- Die size 380 mm^2
- Тактовая частота 0.8 GHz
- Power – 45 W
- Performance 50/25 Gflops sp/dp
- Memory throughput 38,4 GB/sec (3 DDR3 channels)
- 3x16 GB/sec inter CPU channels for 4 CPU ccNUMA 16 GB/sec
- 2 IO links
- Sampling 2013
- Production 2014H1

Servers and Clusters with Elbrus-4C CPUs

Server Elbrus-4.4 (based on Elbrus-4C CPU)

- 4 CPUs Elbrus-4C (4 cores, 800 МГц), total of 200 GFLOPs , 2 southbridge controllers
- RAM: 96 GB, 12x DIMM DDR3-1600
- Interfaces: SATA 2.0 – 8 channels, Gigabit Ethernet – 2 channels, PCI Express 1.0 x8 – 2 slots, PCI – 2 slots, USB – 6 slots
- Case height: 2U,1U

Cluster based on Elbrus-4C CPUs

- Cabinet 47U – 1;
- 4-processor servers – up to 64
- CPUs – up to 256 (1024 cores)
- RAM – 6-12 TB
- HD – 32-64 TB
- FPGA-based interconnect (design by MCST)
- Air Cooling system
- Power – up to 20 KW
- Peak performance – up to 13,8 TFLOPs

CephFS

Database server

Next generation CPUs and controllers

Elbrus-8C

- 8 Elbrus cores
  - 30+ ops per cycle
- 1,3 GHz clock frequency
- Peak performance 125/250 Gflops dp/sp
- TSMC 28 nm process
- Die area 321,4 mm^2
- L2 Cache – 512 KB per core
- L3 Cache – 16 MB, shared
- sampling – 2015Q4
- production – 2016H1

South bridge KPI-2

- TSMC 65 nm process
- CPU channel - 16 GB/sec
- interfaces
  PCI Express 8+8+4 lines,
  Gigabit Ethernet – 3 ports,
  SATA 3.0 – 8 ports,
  USB 2.0 – 8 ports
- SPMC controller
- interrupt controller
- sampling – 2015Q4
- production – 2016H1

Elbrus-8CB

- 8 Elbrus cores
  - 50+ ops per cycle
- 1,5 GHz clock frequency
- Peak performance 580/290 Gflops sp/dp
- Die area 335 mm^2
- L2 Cache – 512 KB per core
- L3 Cache – 16 MB, shared
- TSMC 28 nm process

Sampling – 2018Q2
Production – 2018Q4

1 CPU (8 core) desktop and 4 CPU server with KPI-2 fabricated

Performance increase of the Elbrus MP series

Elbrus-2C+		Elbrus-4C		Elbrus-8C		Elbrus-8CV
0.5 GHz, 2+4 C	2 years	0.8 GHz, 4 C	2 years	1.3 GHz, 8 C	2 years	1.5 GHz, 8 C
2*DDR2-800		3*DDR3-1600		4*DDR3-1600		4*DDR4-2400
16+12 Gflops sp	**3x**	50-60 Gflops sp	**4-5x**	250 Gflops sp	**2x+**	512+ Gflops sp
25 W		45…60 W		~60…90 W		~60…90 W
90 nm		65 nm		28 nm		28 nm
2011		2013		2015		2018

We are developing next generation MP, computers, and system software

МЦСТ ЭЛЬБРУС

Elbrus-16C microprocessor

- Current stage – conceptual design
- Features
  - ✓ Peak performance - up to 1500/750 Gflops sp/dp
  - ✓ 16 cores;
  - ✓ 2 GHz clock frequency
  - ✓ Memory ddr4-3200, 102 GB/s throughput
  - ✓ SoC includes PCIe 3.0, 10 Gb Ethernet, SATA 3.0, USB 3.0
  - ✓ Power ~90 W
  - ✓ TSMC16 nm process
  - ✓ New core and system technologies
- Sampling - 2020H2
- Production - 2021H1

Important step toward exaflops performance supercomputer

МЦСТ
ЭЛЬБРУС

Thank you