A major breakthrough in Computer Science

FIONa

Chariot Lab

Chariot Technologies Lab., Inc.
919, North Market Street, Suite 950,
Wilmington, County of New Castle,
DE, Zip 19801
www.chariotlab.com

Conditional statements - bottleneck for computing

In most cases processors are unable to execute conditional statements as fast as arithmetic operations

	PROCESSING	PARALLEL EXECUTION	POWER CONSUMPTION	MODULE IN ALU
«IF-DO/ELSE» STATEMENTS	SLOW 2 CLOCK CYCLES	NOT POSSIBLE FOR MULTILEVEL TASKS	HIGH POWER CONSUMPTION	EXECUTED BY DIGITAL COMPARATOR
ARITHMETIC OPERATIONS	QUICK 1 CLOCK CYCLE	EASILY PARALLELIZED	LOW POWER CONSUMPTION	EXECUTED BY ARITHMETIC UNIT

AS A RESULT SEARCHING, SORTING, COMPARISON ALGORITHMS, MACHINE-LEARNING TASKS AND MANY OTHER PROGRAMS COMPRISING (MULTI-LEVEL) CONDITIONAL STATEMENTS CONSUME SIGNIFICANT COMPUTING AND TIMING RESOURCES TO EXECUTE

FIONa - “NO IF” technology

FIONa transforms conditional statements into full arithmetic equivalent

Completely new way of conditional statements processing since first integrated circuit was invented!

RESULTING IN FUNDAMENTAL CHANGE OF EXECUTING LOGICAL CONDITIONS ON BOTH SOFTWARE AND HARDWARE LEVELS

LEADING TO DRAMATIC IMPROVEMENT OF EXECUTION SPEED OF LOGICAL TASKS

The magic of conversion

Conditional statements involve comparisons of values (the "IF part" of the "IF A=><B_ DO y, ELSE DO z") represented by three main CMP operations that can be resolved with specific arithmetic formulas

CMP OPERATION TYPE	A = B	A < B	A > B
ARITHMETIC EQUIVALENT	ANSWER = FORMULA 1	ANSWER = FORMULA 2	ANSWER = FORMULA 3

ANSWER IS ALWAYS TRUE (1) OR FALSE (0) – TO BE APPLIED TO "DO Y" (IF TRUE) OR "ELSE DO Z" (IF FALSE)

Universal character with vast perspectives

Being a purely mathematical solution, FIONa:

IS UNIVERSAL AND CAN BE IMPLEMENTED AT ANY LEVEL – FROM HIGH-LEVEL PROGRAMMING LANGUAGE SOURCE CODE TO INSTRUCTION SET ARCHITECTURE

ALLOWS CONVERSION OF ANY PROGRAM CODE INTO A SINGLE LINEAR ARITHMETIC FORMULA WHICH OPENS UP VAST OPPORTUNITIES FOR OPTIMIZED PROCESSING

Limitations and effects

FIONa techniques are aimed at accelerated execution of conditional statements only. Arithmetic operations are not the subject of this innovation

The level of acceleration depends on the share of conditional statements – the more conditional statements to resolve, the greater the FIONa speed advantage

Example of conversion

```
// Python standard code (with IFs)

answer = 0

if num > 1:
    answer = 2
elif  num < 1:
    answer = 1
else:
    answer = 0
```

```
// FIONa-modified equivalent Python code (no IFs)

answer = 0

dif = num - 1
alt_dif  = 1 - dif
res = ((1<<abs(dif)-dif) % 2) * 2 + (1<<abs(alt_dif)-alt_dif) % 2
```

«CMP free» assembly after conversion

FIONa-converted source code produces less and more compact assembly instructions

C CODE STANDARD (STD)

```c
int std_compare(int a, int b){
    return a == b;
}
```

C CODE FIONA (OUR)

```c
int our_simple_compare(int a, int b){
    return (1 << (a-b)) % 2;
}
```

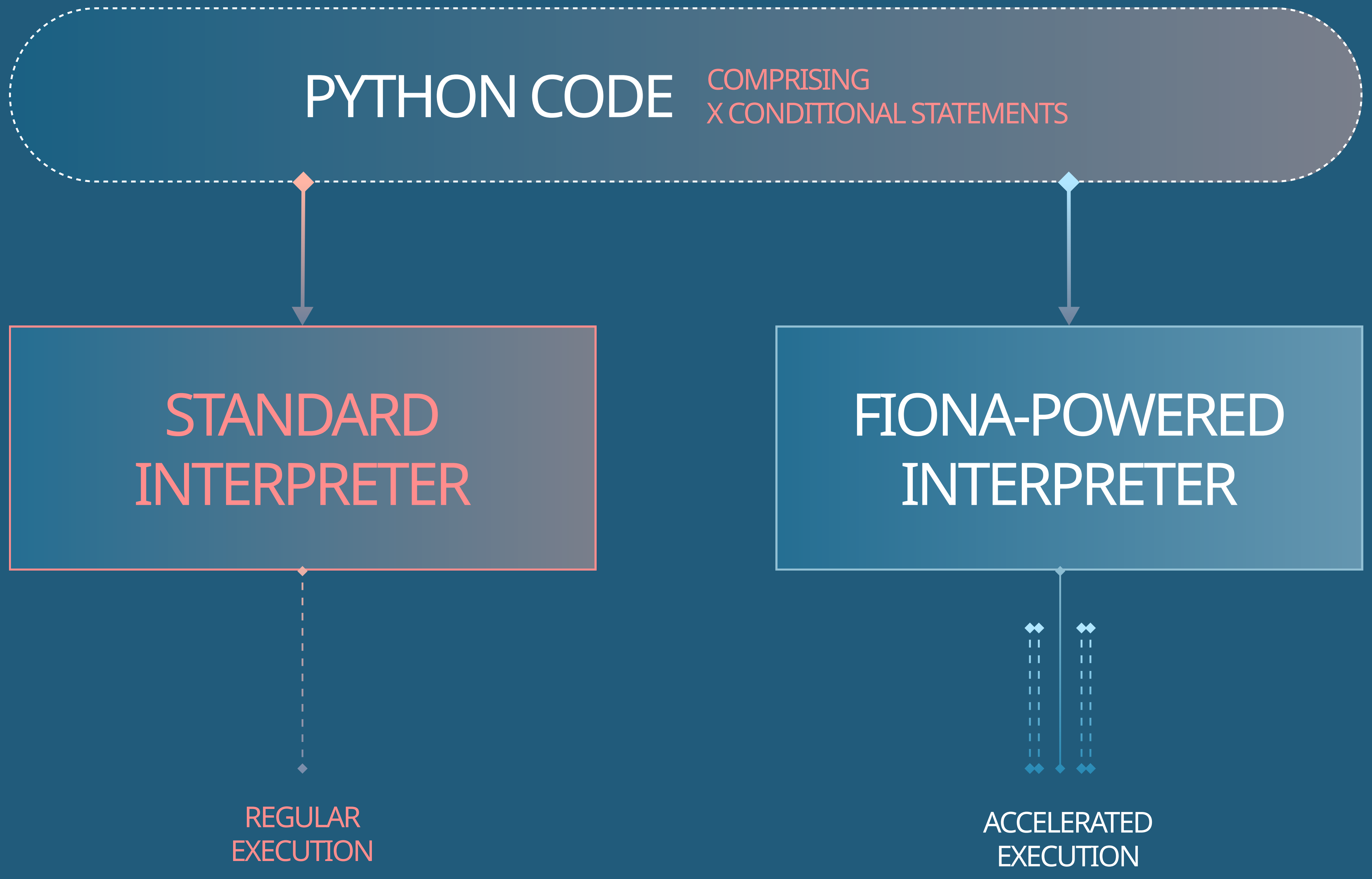
ASSEMBLER EQUIVALENTS
(FOR X86_X64 ARCHITECTURE)

```
std_compare(int, int):
                        push    rbp
                        mov     rbp, rsp
                        mov     DWORD PTR [rbp-4], edi
                        mov     DWORD PTR [rbp-8], esi
                        mov     eax, DWORD PTR [rbp-4]
                        cmp     eax, DWORD PTR [rbp-8]
                        sete    al
                        movzx   eax, al
                        pop     rbp
                        ret
```

```
our_simple_compare(int, int):
                        mov     ecx, edi
                        sub     ecx, esi
                        mov     eax, 1
                        sal     eax, cl
                        mov     edx, eax
                        shr     edx, 31
                        add     eax, edx
                        and     eax, 1
                        sub     eax, edx
                        ret
```

FIONa-powered Python interpreter

Instead of modifying every source code FIONa conversion can be embedded straight into a Python interpreter

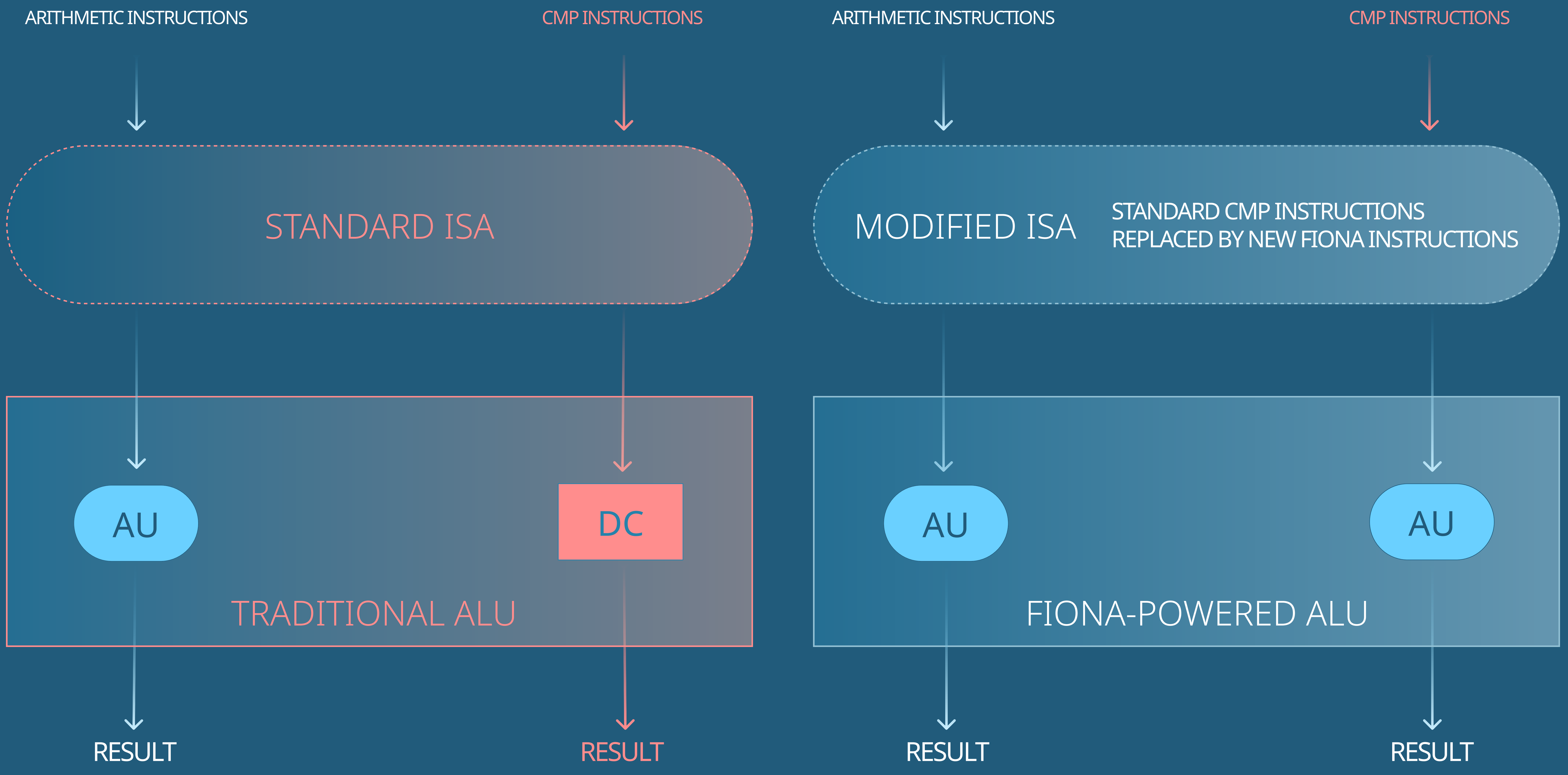


FIONa-powered vs standard algorithms

	Task	Task Details	STANDARD algorithm	FIONa-powered algorithm	Hardware
UNIVERSAL TEST	Matrix comparison	1,000,000,000 elements each	353,7 sec. (Python) 239,6 sec. (Python/Numba)	9,3 sec. (Python) 4,6 sec. (Python/Numba)	Intel(R) Core(TM) i7-8565U CPU
MACHINE LEARNING	DLRM (Pytorch) recom. model Polyfit (TensorFlow/Pytorch) K-mean clustering (TensorFlow/Pytorch)	CTR prediction dataset Spring Data dataset Classification & Clusterisation dataset	38 epoch - 56 min. (loss - 0,52) 3 sec. (accuracy - 0,75) 15 sec.	49 epoch - 48 min. (loss - 0,35) 3 sec. (accuracy - 0,93) 10 sec.	McBook Pro 13' 2020 M1
DATA PROCESSING	Prime number factorization	Factorization of 80-bit prime number	24 hours	5 min.	MacBook Pro 15, 2017 (core i7)
SEARCHING	Array search	Dataset (10,000 words in 200 pages)	57 sec.	2 sec.	MacBook Pro 15, 2017 (core i7)
OPTIMIZATION	Shortest way search task Clustering + shortest way task	80 destin. points for 1 rider 100 destin. points for 2 riders	3,9 min. (Deep First Search) 8,3 min. (Google TensorFlow)	0,6 sec. 2 sec.	MacBook Pro 15, 2017 (core i7)

How it works in hardware

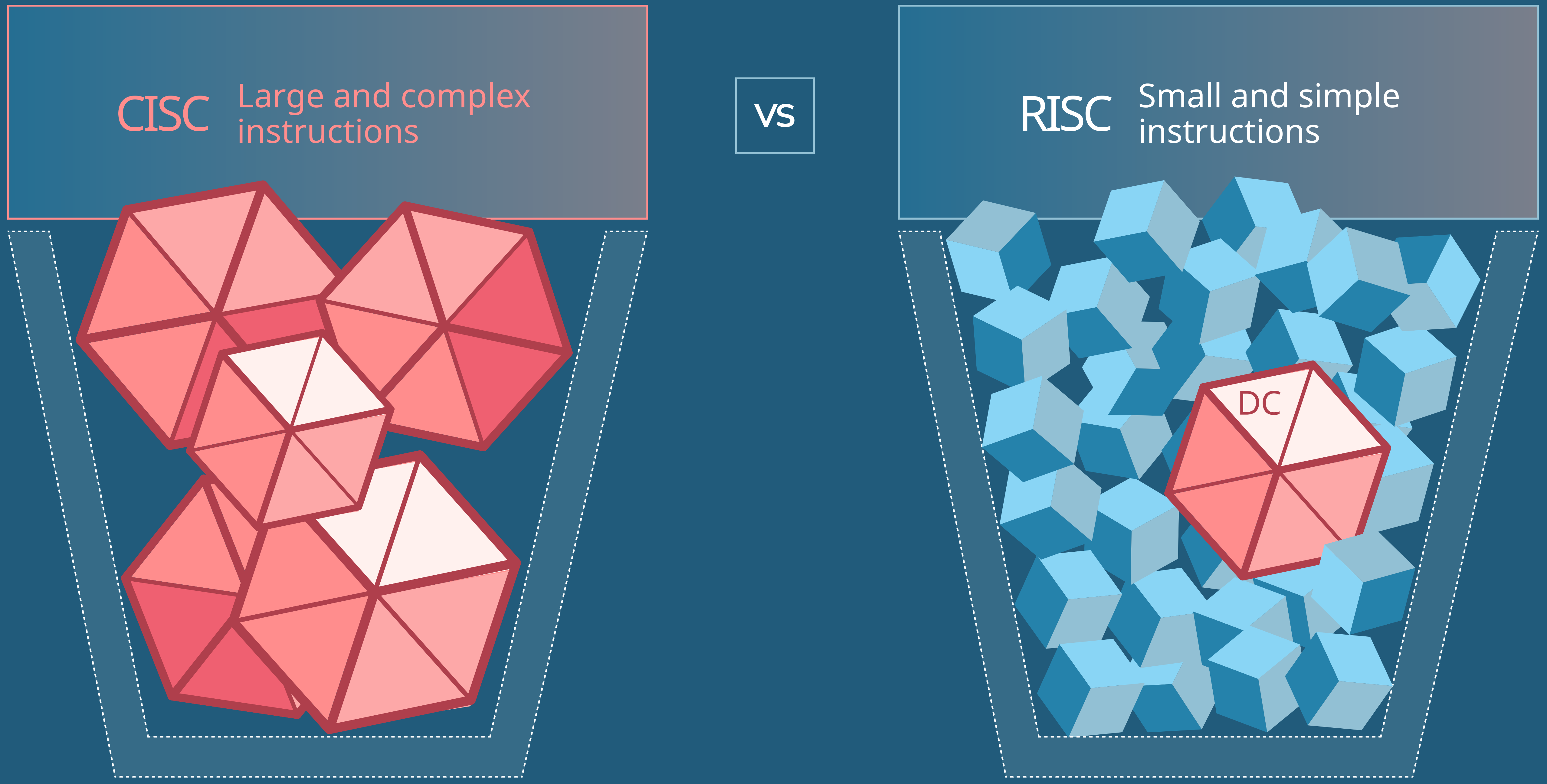
CMP operations can now be processed by AU. Redundant DC can be removed or replaced



Abbreviations:

ALU – Arithmetic Logic Unit
ISA – Instruction Set Architecture
CMP – ”compare” instructions
AU – Arithmetic Unit
DC – Digital Comparator (and/or AU zero state flags)

Architecture evolution – CISC vs RISC



RISC advantages:

EASIER TO INCREASE PROCCESSOR CLOCK SPEED

PARALLELISATION OF (ARITHMETIC) OPERATIONS POSSIBLE

SIMPLER AND FASTER PROCESSOR STRUCTURE

LOWER POWER CONSUMPTION

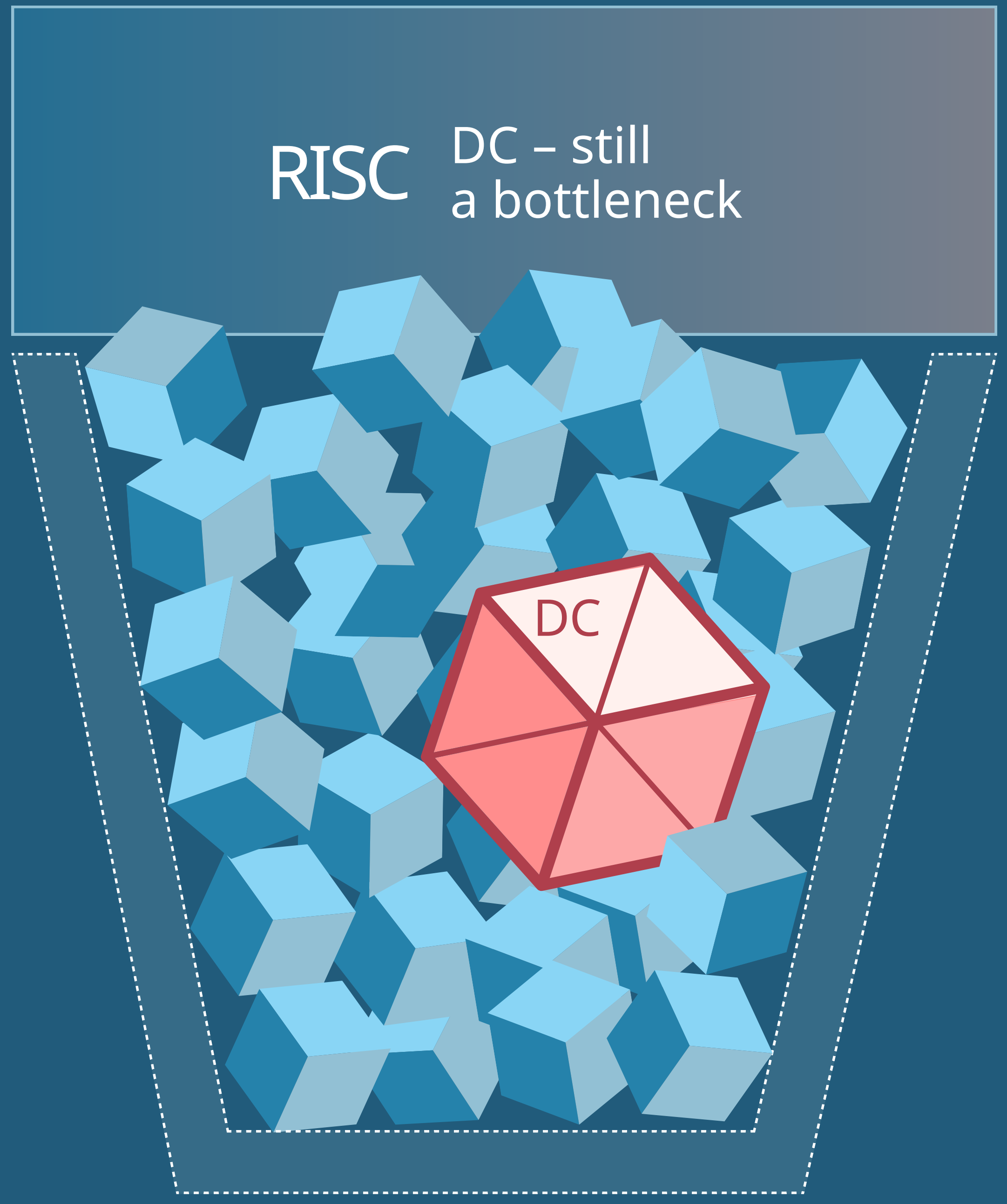
Abbreviations:

CISC – complex instruction set computer

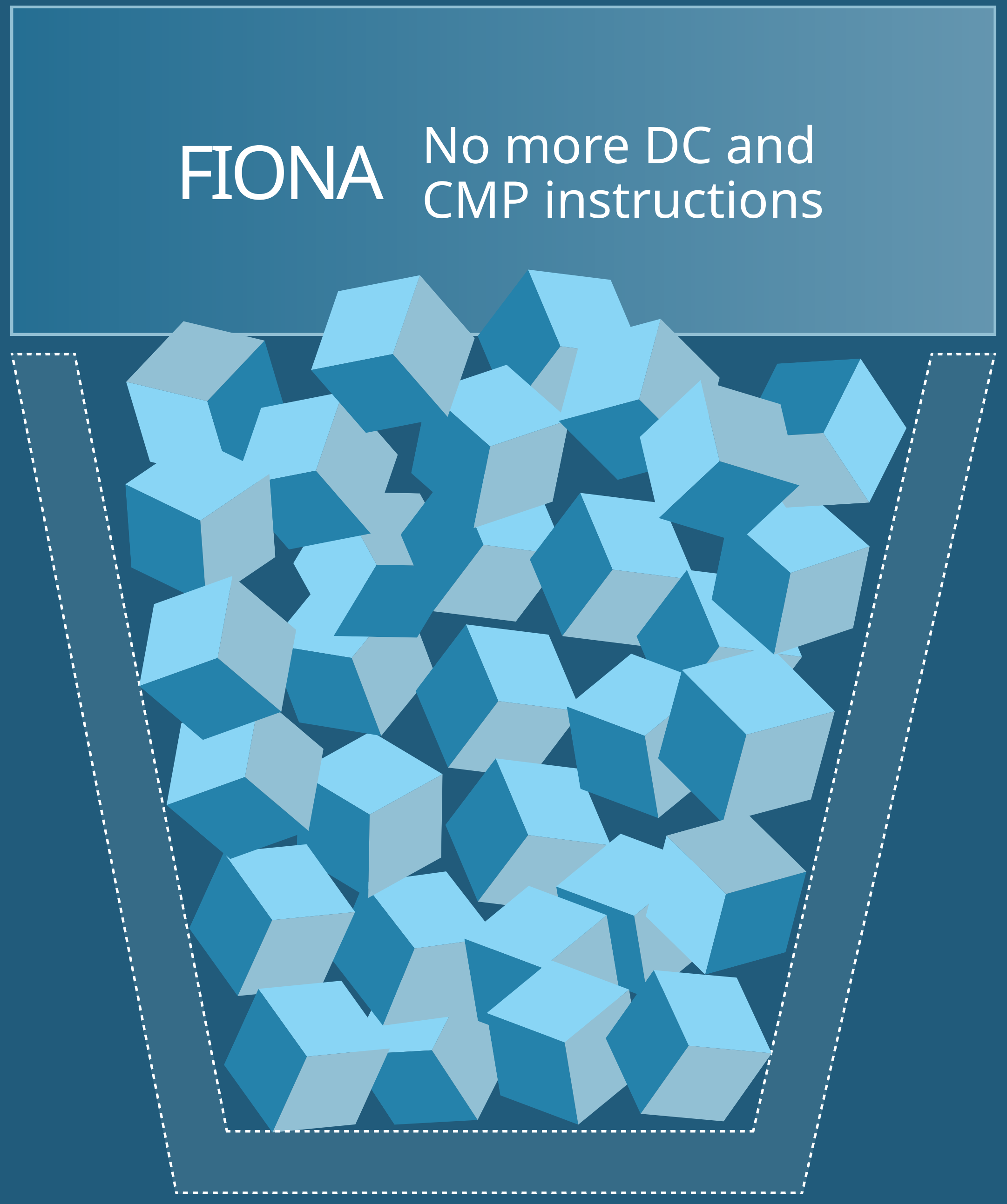
RISC – reduced instruction set computer

DC – digital comparator and/or AU zero state flags

Architecture evolution – RISC vs FIONa



VS



FIONa advantages:

ACCELERATED EXECUTION OF CONDITIONAL STATEMENTS
1 CLOCK VS 2 BEFORE

PARALLEL EXECUTION OF MULTILEVEL CONDITIONAL STATEMENTS
NOT POSSIBLE BEFORE

LOWER POWER CONSUMPTION

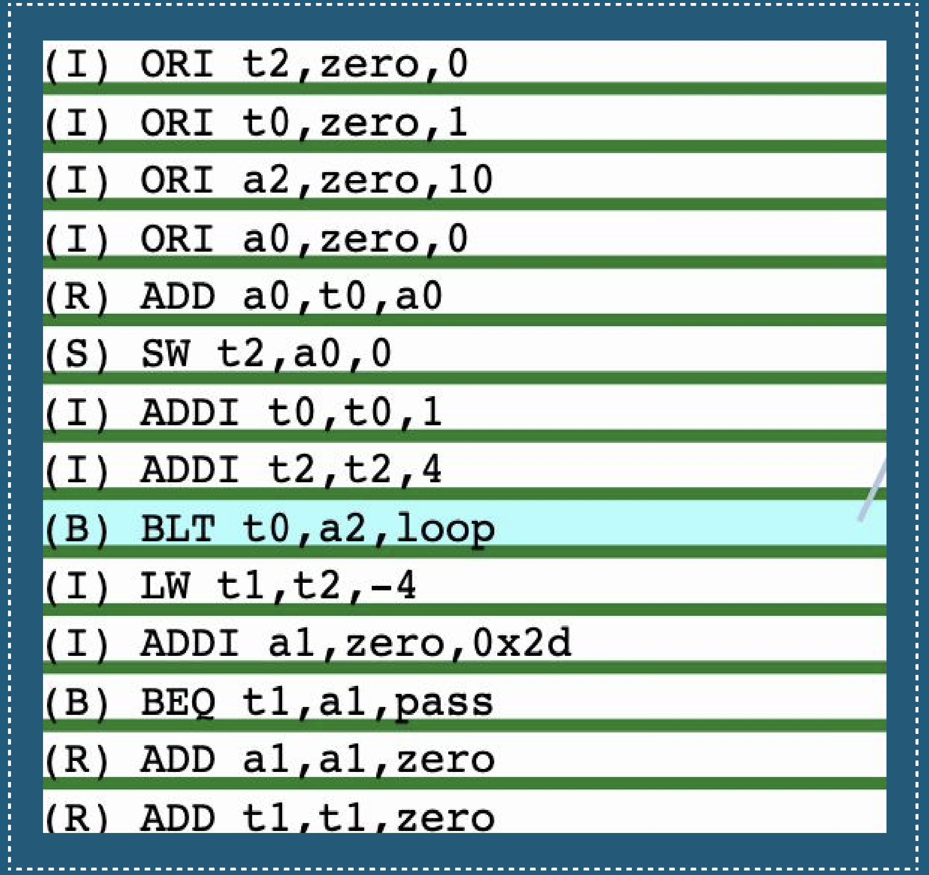
MORE POWERFUL AND COMPACT PROCESSOR

Abbreviations:
RISC – reduced instruction set computer CMP – compare instructions
DC – digital comparator and/or AU zero state flags

FIONa-powered RISC-V architecture

The first FIONa hardware prototype was simulated on MakerChip design platform

SAME ASSEMBLER ALGORITHM WAS EXECUTED FROM 1 TO 3 MLN. CYCLES ON TWO ARCHITECTURES (CMP OPERATIONS SET AT 25% OF TOTAL):


```
(I) ORI t2,zero,0
(I) ORI t0,zero,1
(I) ORI a2,zero,10
(I) ORI a0,zero,0
(R) ADD a0,t0,a0
(S) SW t2,a0,0
(I) ADDI t0,t0,1
(I) ADDI t2,t2,4
(B) BLT t0,a2,loop
(I) LW t1,t2,-4
(I) ADDI a1,zero,0x2d
(B) BEQ t1,a1,pass
(R) ADD a1,a1,zero
(R) ADD t1,t1,zero
```

STANDARD
1 CORE RISC-V

FIONA-POWERED
1 CORE RISK-V

makerchip

2-3x

better performance FIONa over standard

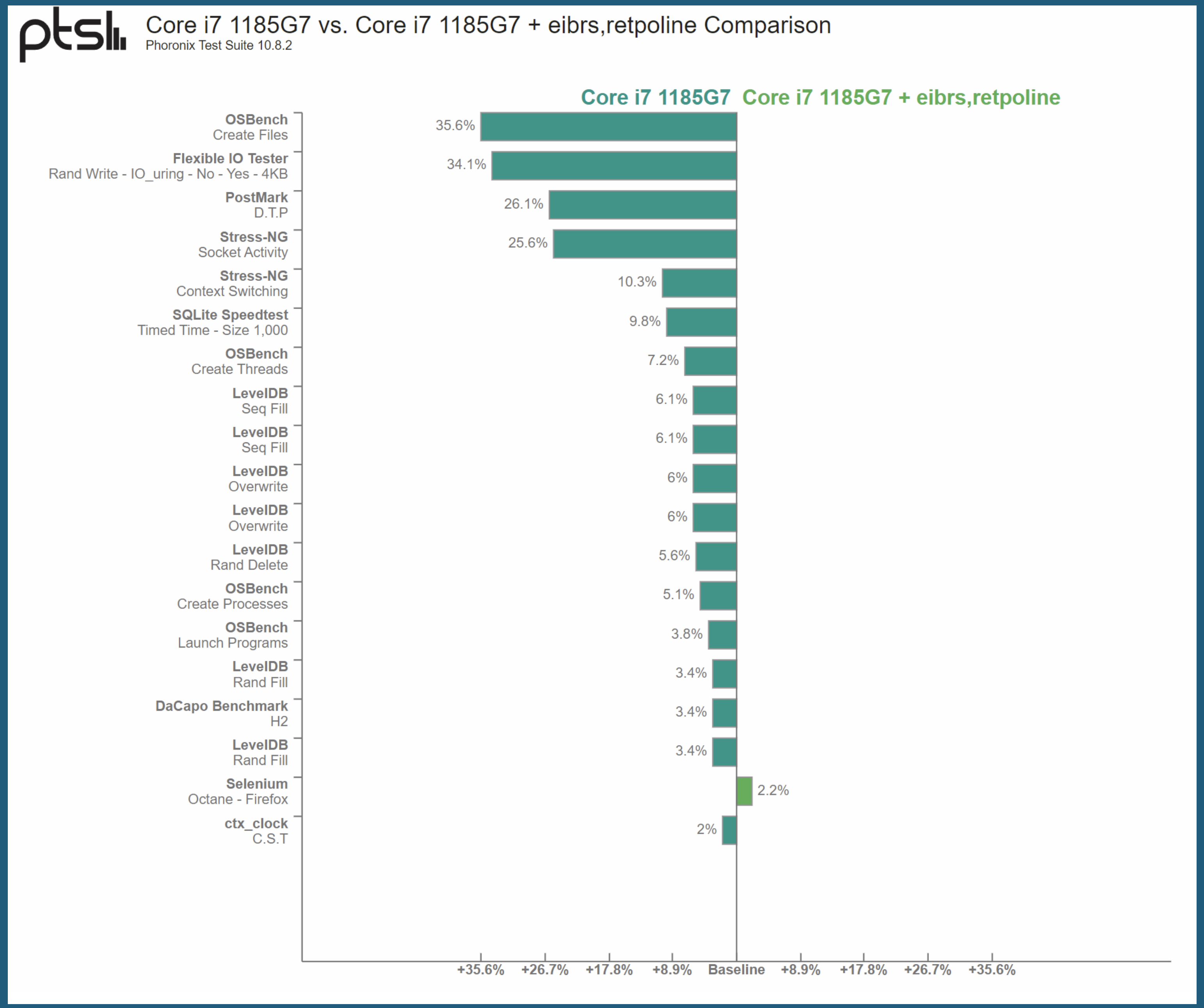
DEPENDING ON THE NUMBER OF CYCLES

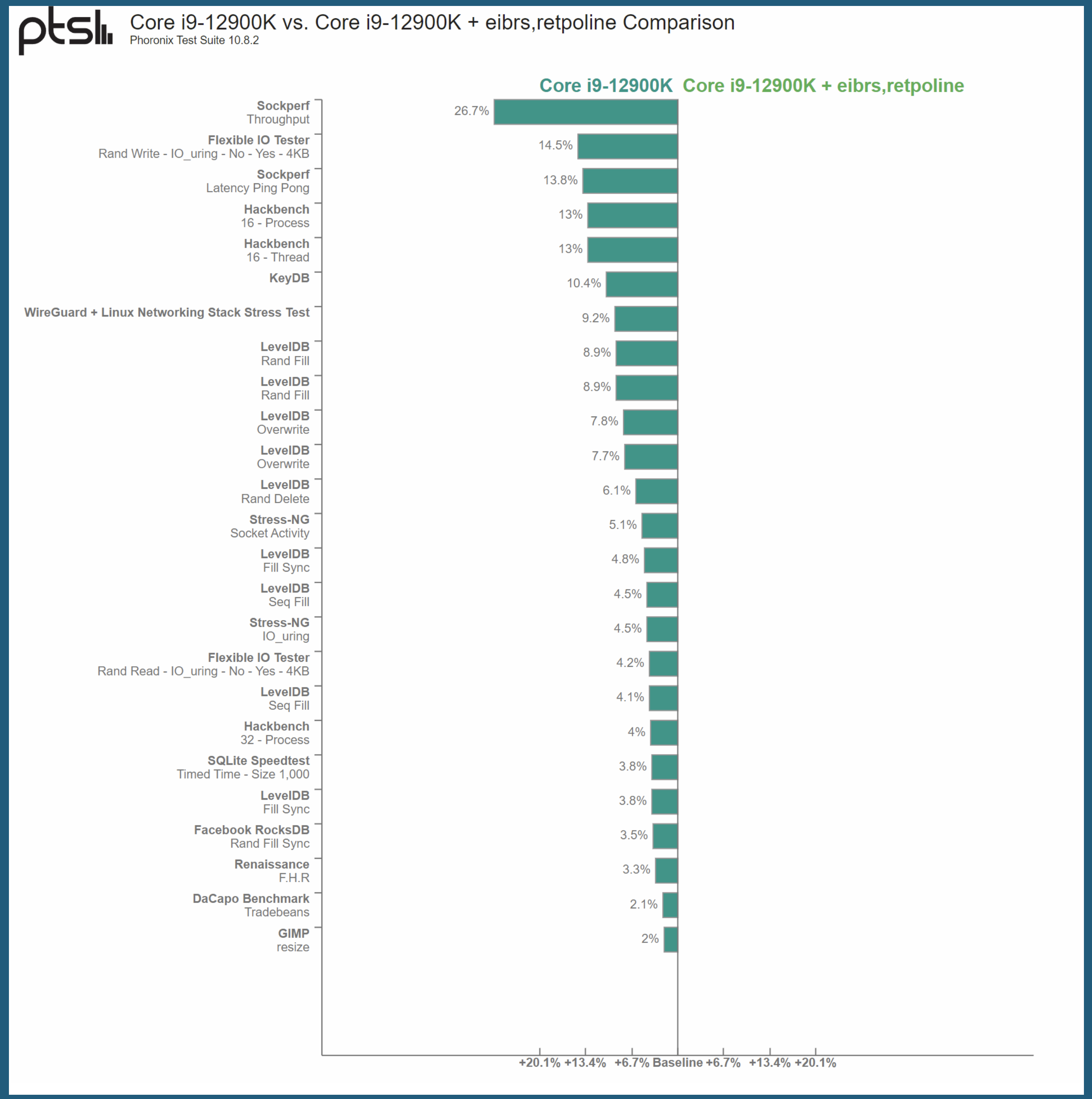
FIONa remedies serious Spectre V2 vulnerability

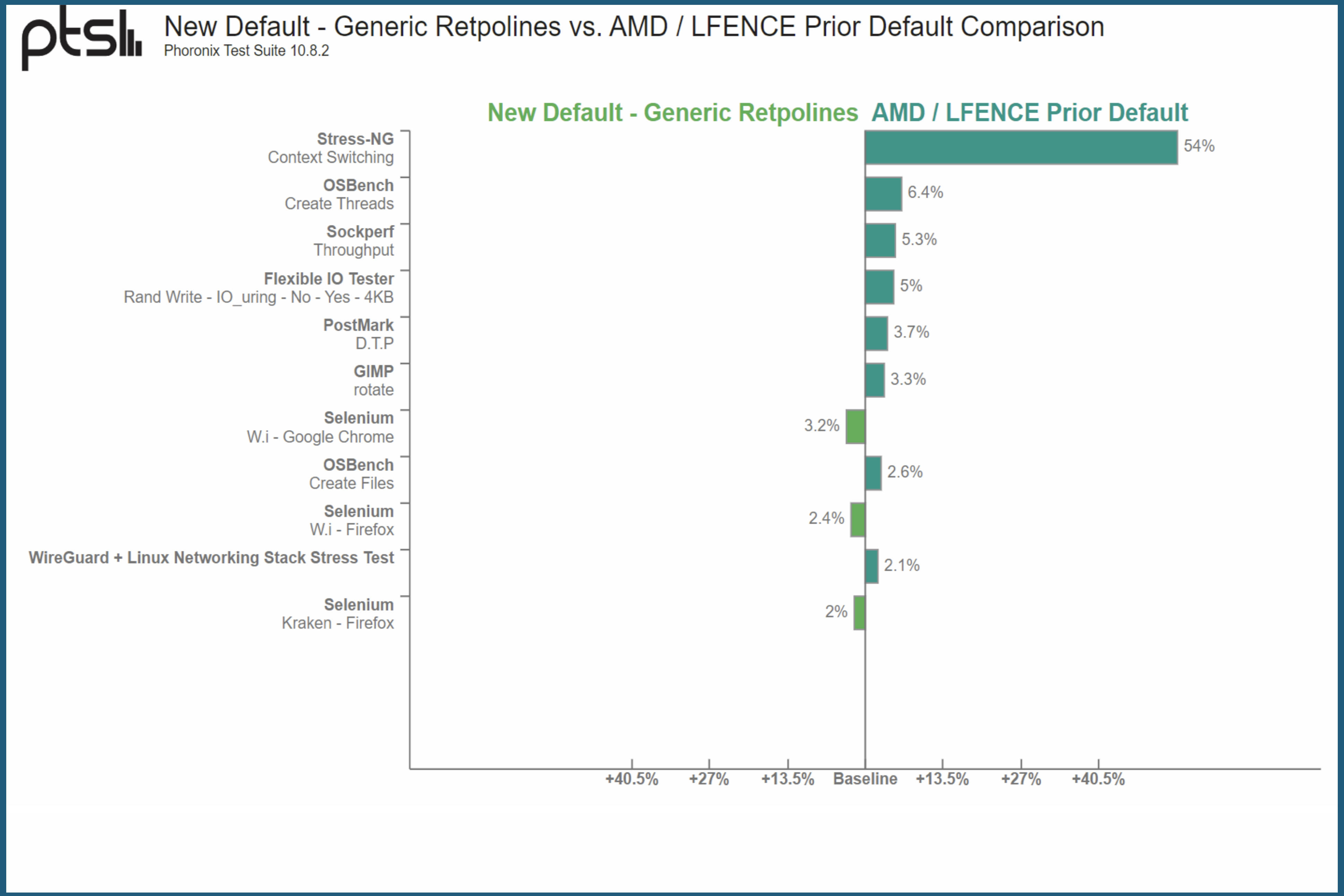
FIONa allows to avoid branch predictor speculations thus giving no foundation for a Spectre Attack

26-35% AFFECTING intel CORES

50+% AFFECTING AMD CORES







A PASS TO HIGH PERFORMANCE COMPUTING

FIONA TECHNIQUES REPRESENT A DISRUPTIVE APPROACH IN PROCESSING OF CONDITIONAL STATEMENTS

SOFTWARE IMPLEMENTATION ALLOWS EASY AND INSTANT ACCELERATION OF ANY PROGRAM

HARDWARE EMBODIMENT IS MOST EFFICIENT AND JUST REQUIRES CONVERSION EMBEDDED AT ISA LEVEL

OPTIONAL ADJUSTMENTS TO ARCHITECTURE DESIGN (REMOVAL AND REPLACEMENT OF REDUNDANT BLOCKS) CAN ENHANCE PRODUCTIVITY AND REDUCE PHYSICAL SIZE OF SILICON

BUT WHAT IS MORE IMPORTANT

FIONa paves the way for many more innovations

AS IT ALLOWS EASY REPRESENTATION OF ANY PROGRAM AS A SET OF ARITHMETIC OPERATIONS