ISA-Aging

Envelhecimento de Conjuntos de Instruções

Rodolfo Azevedo

rodolfo@ic.unicamp.br

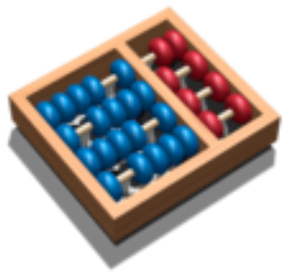
Slides baseados na apresentação do artigo

SHRINK: Reducing the ISA Complexity Via Instruction Recycling

ISCA 2015

UNICAMP

LSC COMPUTER SYSTEMS LABORATORY



ERAD-SP 2017

UNICAMP

NO FOOD OR DRINK IN THIS ROOM

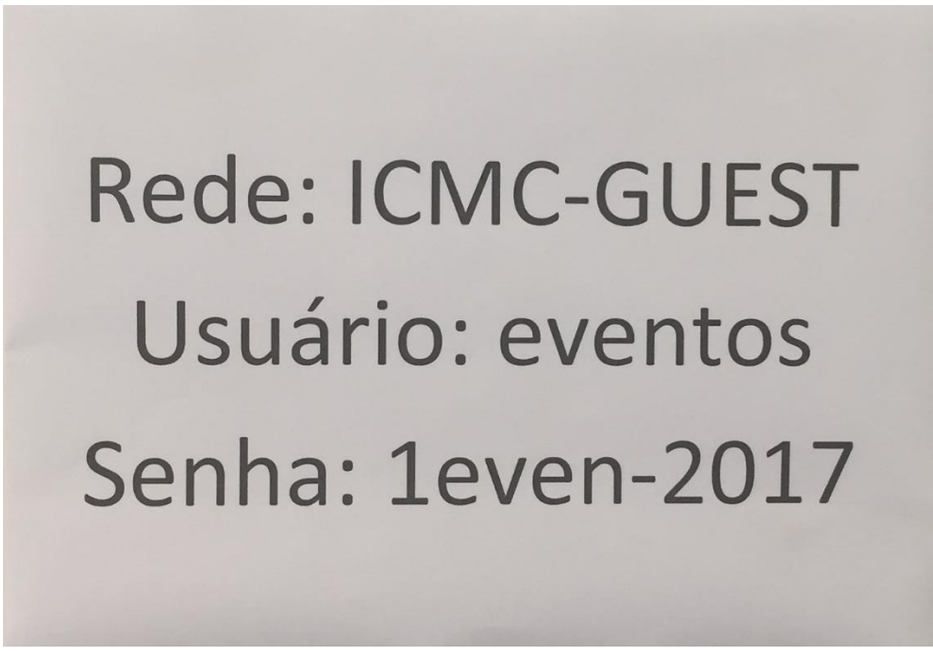
ICMC USP SÃO CARLOS

Thank you

Quiz

Abra a página:

kahoot.it



Aguarde o PIN number

ISA-Aging

Envelhecimento de Conjuntos de Instruções

Rodolfo Azevedo

rodolfo@ic.unicamp.br

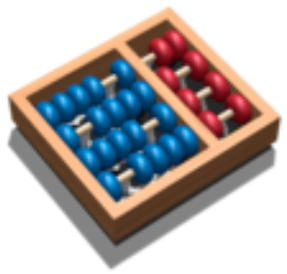
Slides baseados na apresentação do artigo

SHRINK: Reducing the ISA Complexity Via Instruction Recycling

ISCA 2015

UNICAMP

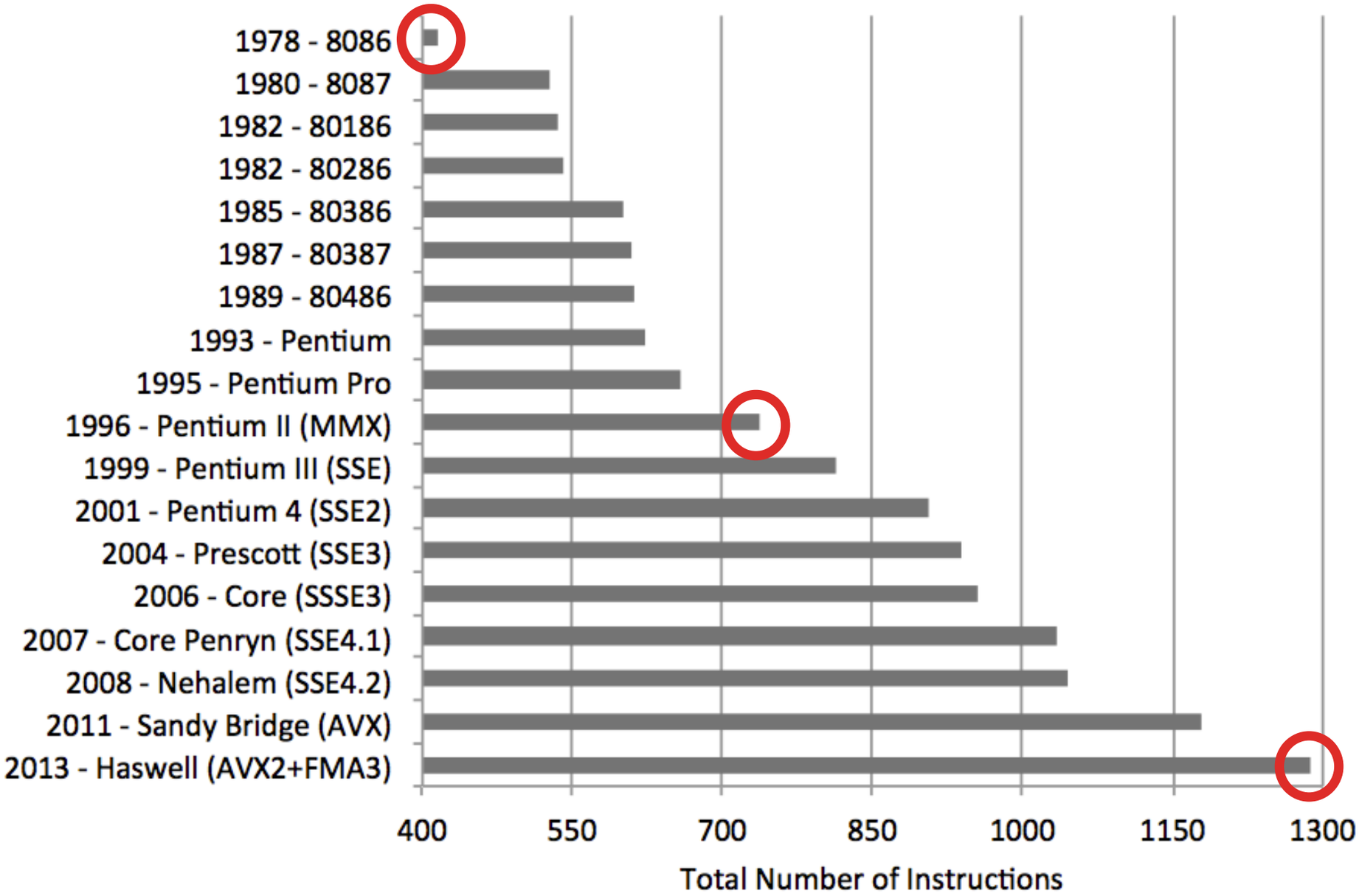
LSC COMPUTER SYSTEMS LABORATORY



ERAD-SP 2017

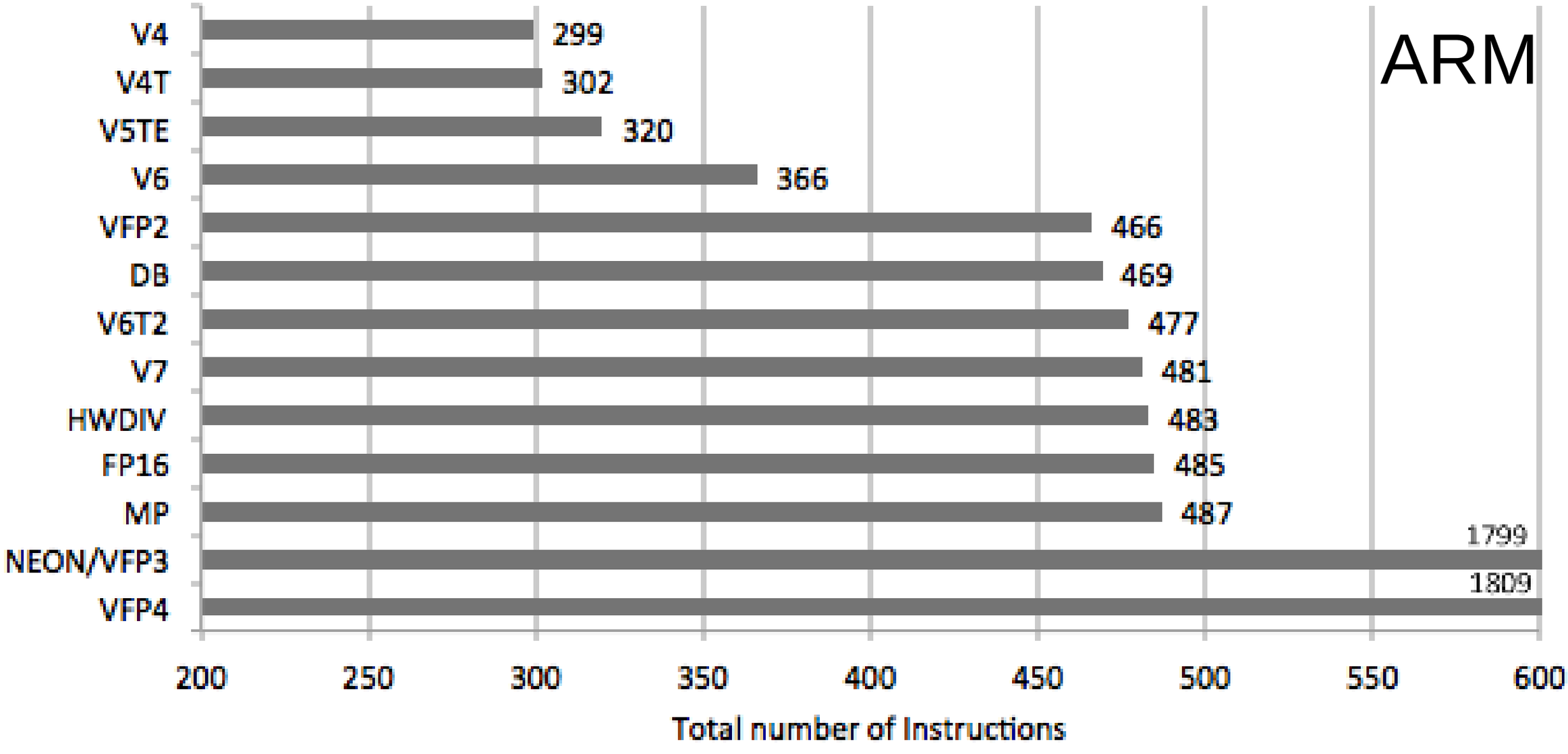
EU

Introduction

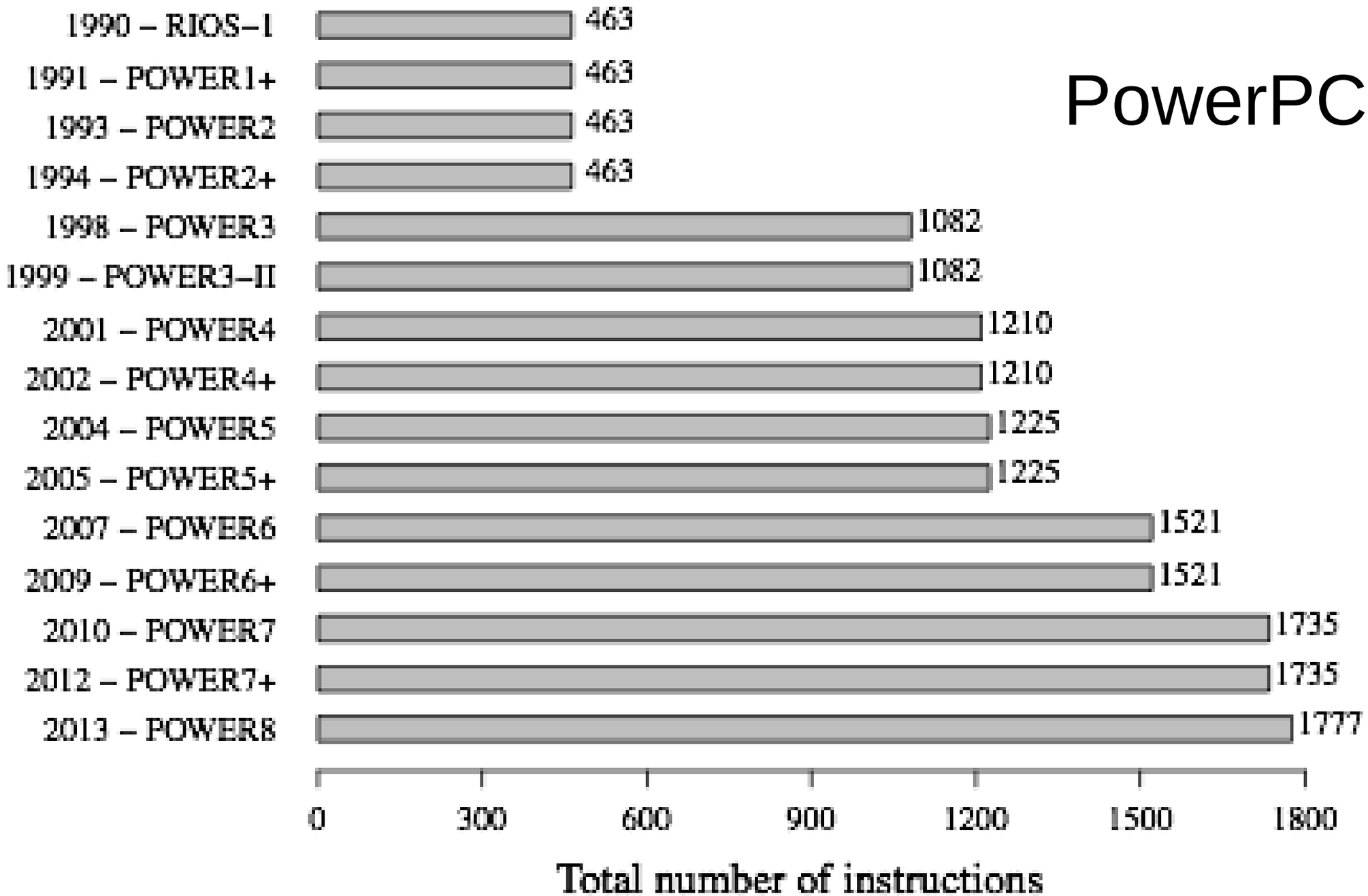


x86 code is bigger than RISC (ARM)

What about other architectures?



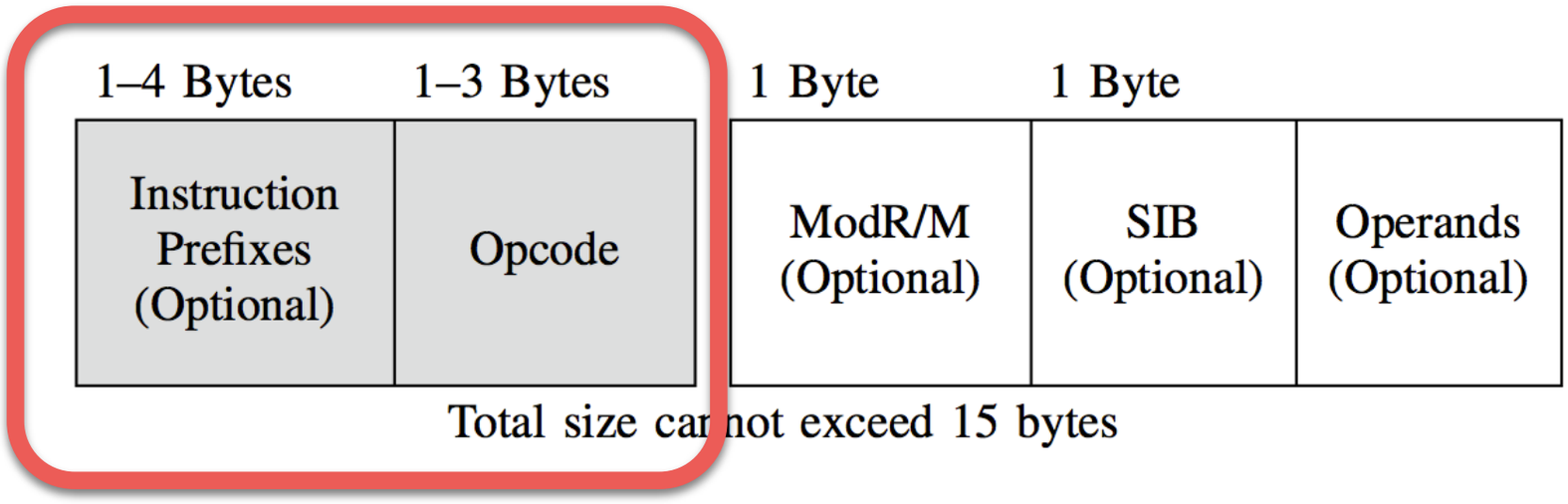
What about other architectures?



PowerPC

The x86 instruction set

- Intel 8086 family, **variable-length** format
- **Operation code**: opcode + other bits to uniquely identify an instruction



Example

Instruction to disassemble: **FF 15 14 12 40 00**

CALL (absolute address)

1	1	1	1	1	1	1	1

FF **OPCODE BYTE**

INSTRUCTION BITS

0	0	0	1	0	1	0	1

15 **MOD-REG-R/M BYTE**

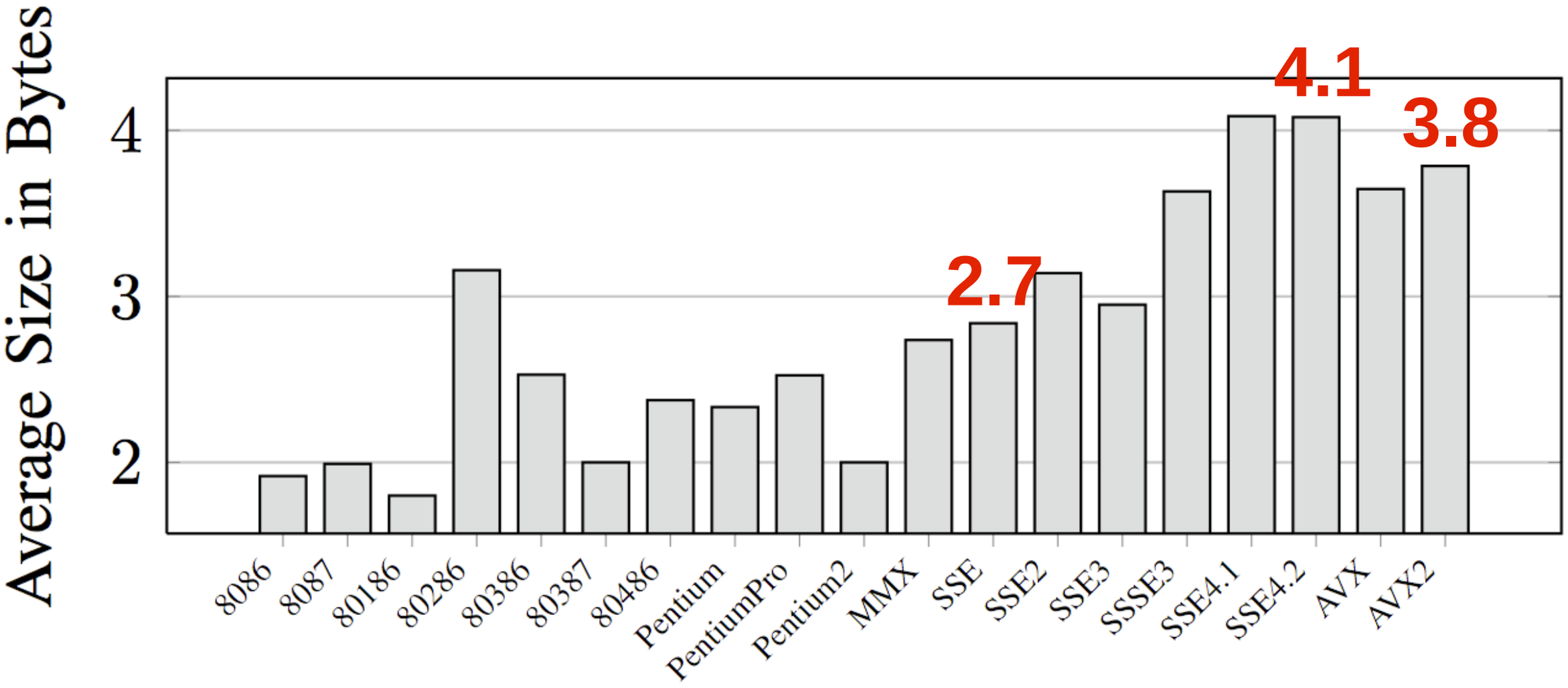
MOD
00 – Indirect addressing mode

REG
010 = Decimal 2. the opcode extension value

R/M
101 = Together with Mod (01), this tells us to read a 4 byte displacement value

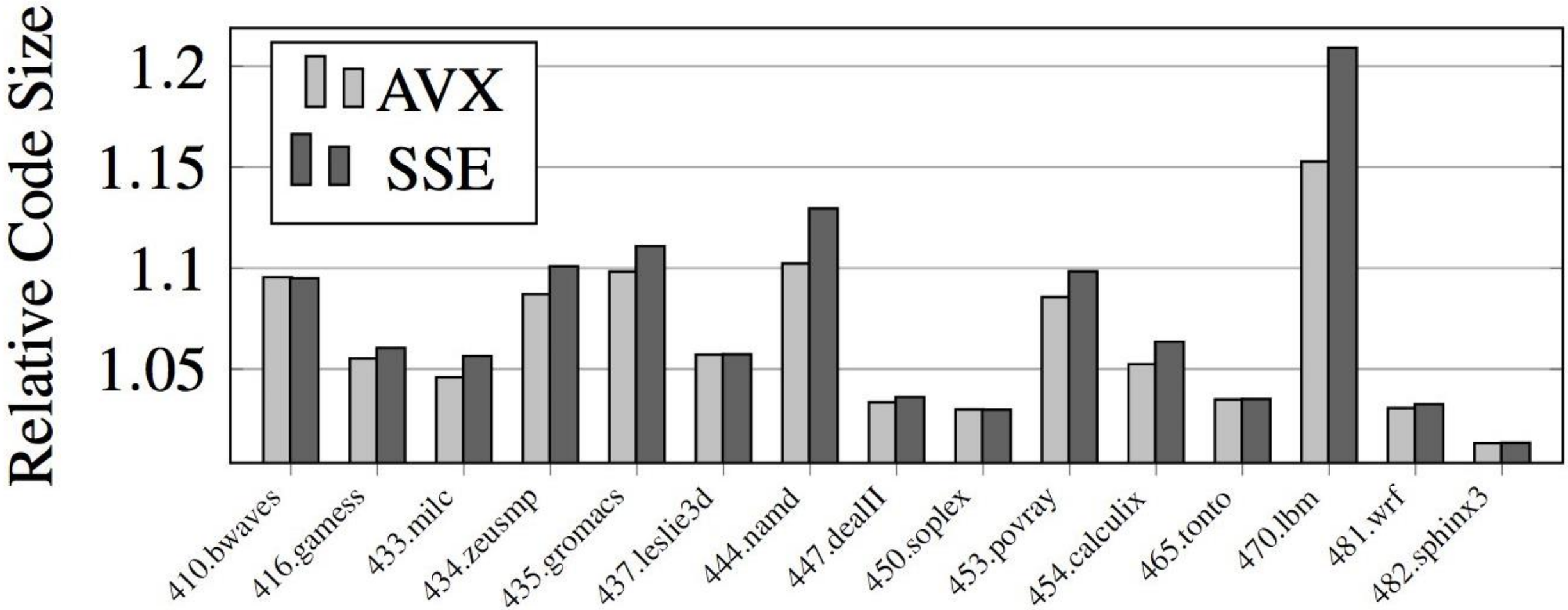
Disassembly: **CALL (absolute) 0x401214**

Average instruction *opcode* size by x86 features



- Variable-length format no longer benefits most used instruction

AVX & SSE (vs x87)
SPEC2006FP



- Modern compilers use AVX or SSE as default ISA for floating point calculations

UNICAMP

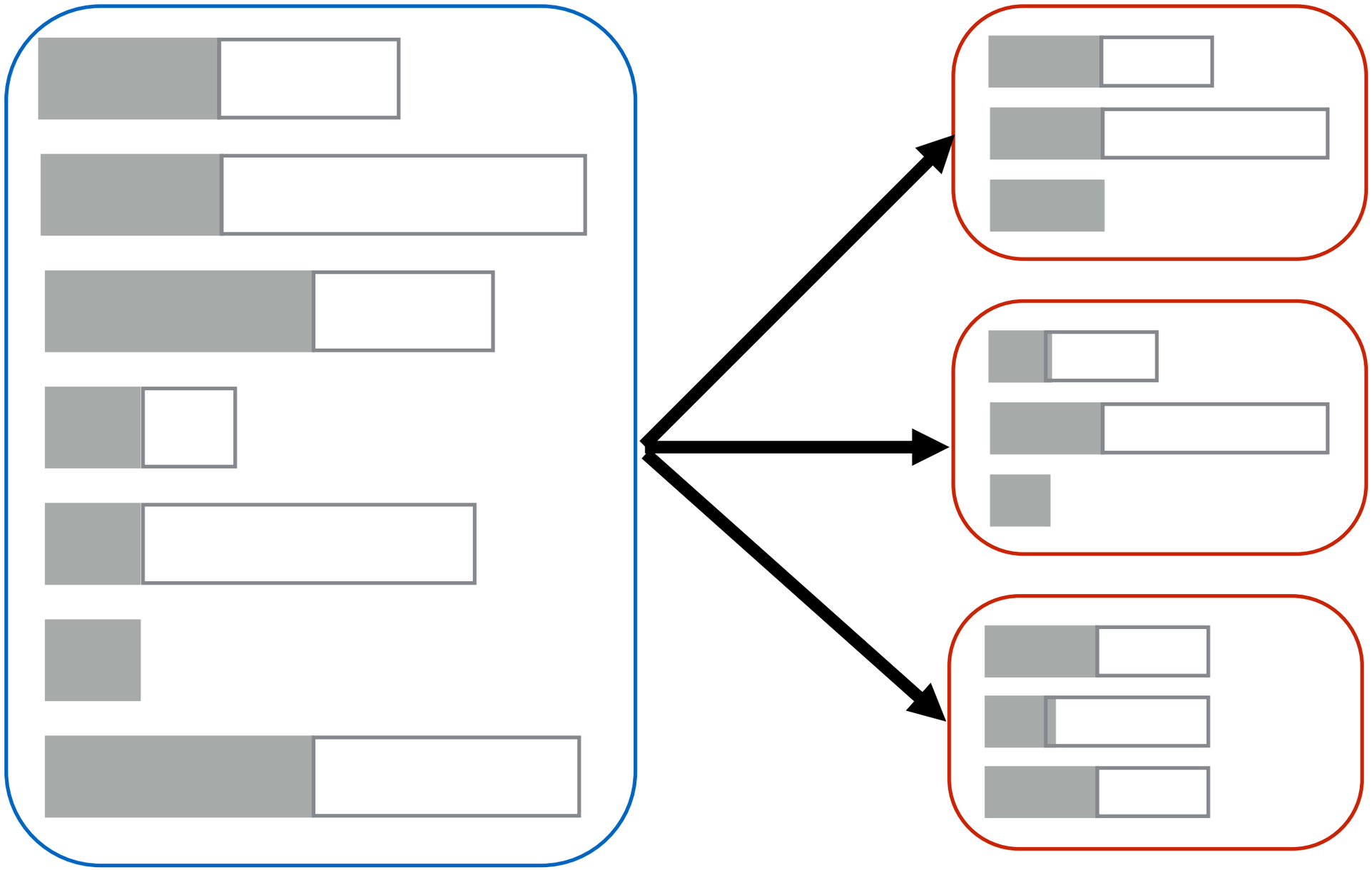
Solutions?

Breaking Backward Compatibility

UNICAMP

- 3 Radical approaches:
  - **(A)** Reduce all opcodes to 2 bytes
  - **(B)** Reduce all opcodes to 1 or 2 bytes
    - 240 instruction encoded using 1-byte opcodes
  - **(C)** Convert to a RISC-like ISA encoding
    - Use ARM ISA for evaluation

Breaking Backward Compatibility

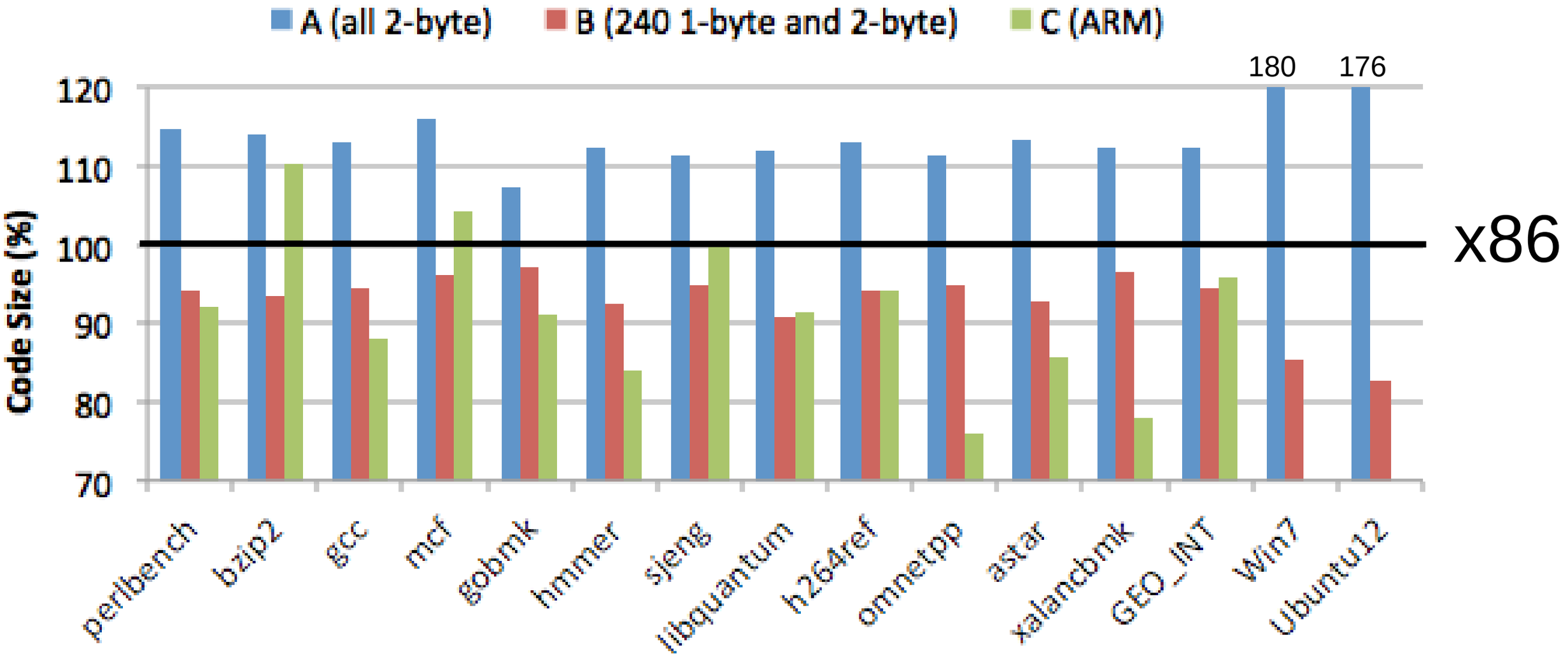


A
2-byte opcodes

B
1 or 2-byte opcodes

C
RISC-like ISA

Evaluation



x86 code is bigger than RISC (ARM) for most programs

- Solution **(B)** encoding shows that variable-length is better than RISC and x86.

However...

- Breaking x86 backward compatibility is **not an option**.
  - Software base
  - Market
- What now?

Recycling Mechanism

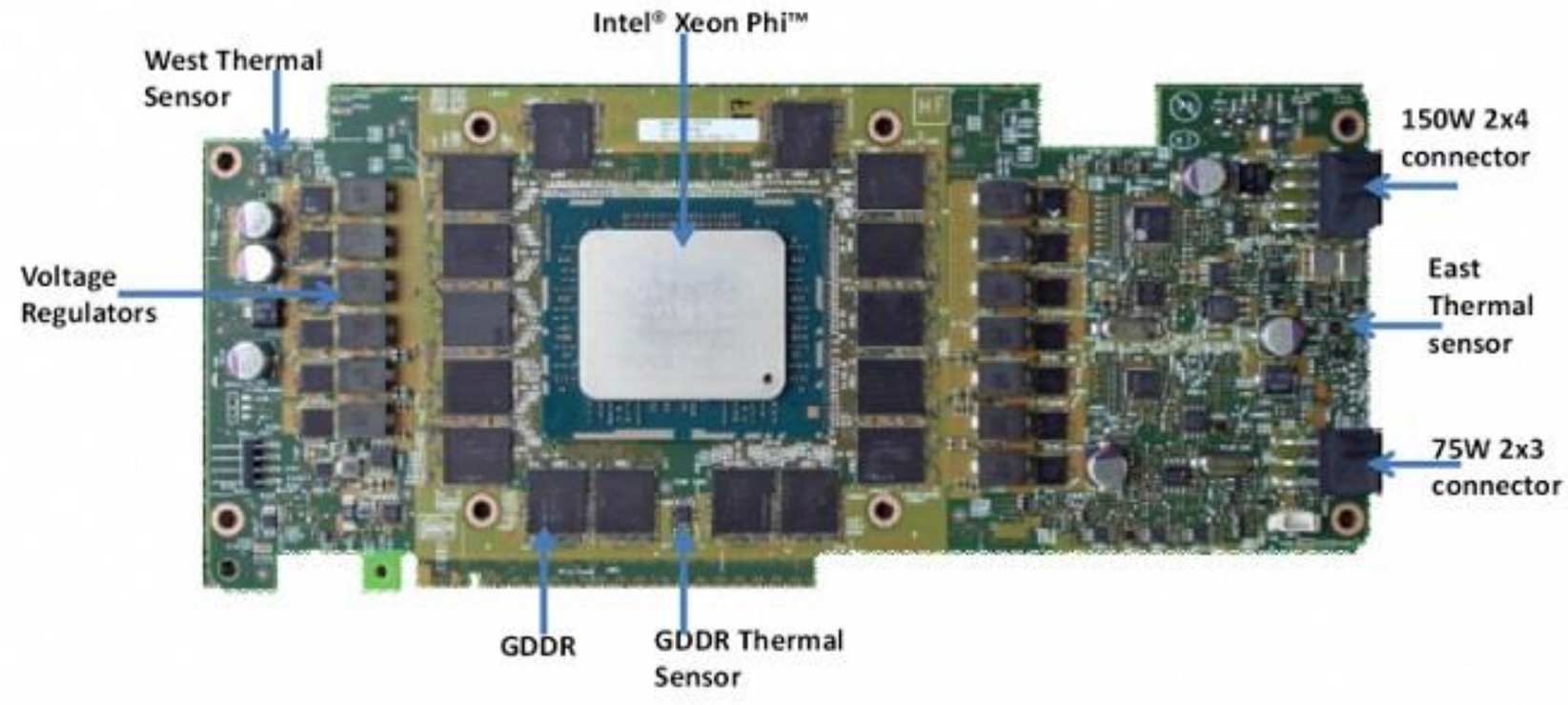
Recycling Mechanism

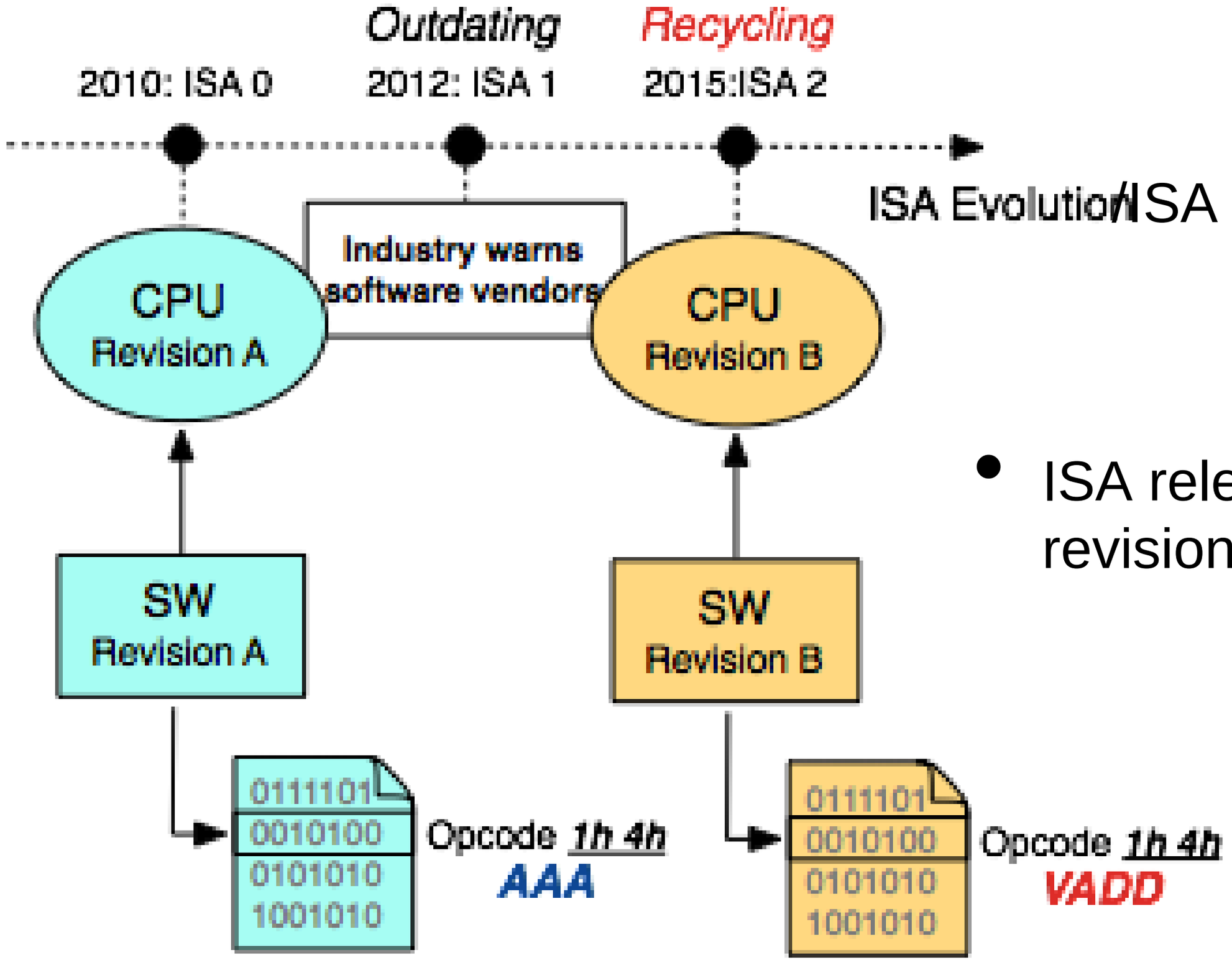
- Remove outdated and unused instructions
- Re-use opcode space to encode new instructions while maintaining backward compatibility

Benefits

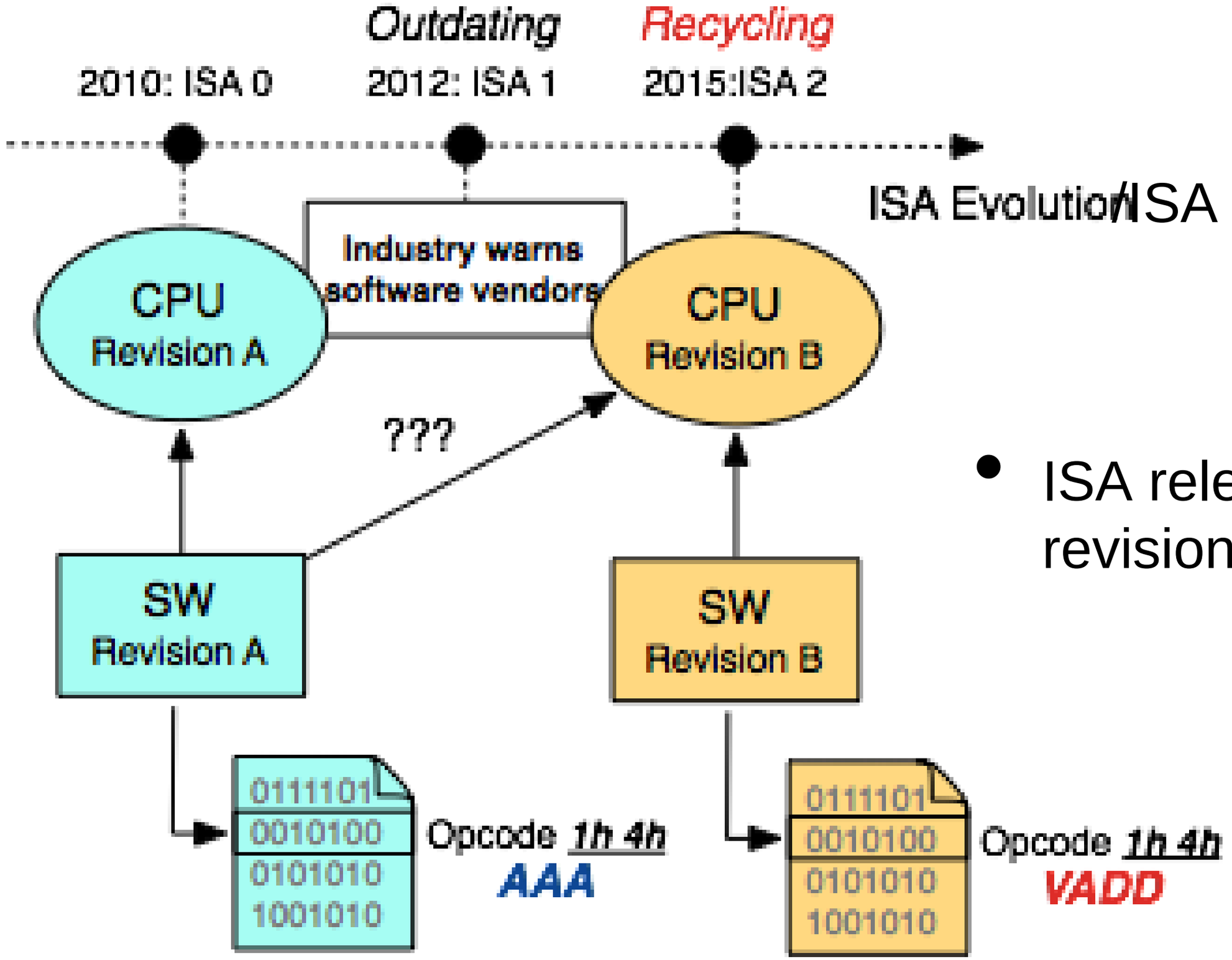
- Open room for encoding new instructions with less bits - improving program size and cache.
- x86 complexity can be reduced, opening market for specific domains; e.g. low-end embedded devices (Quark?).

Two examples





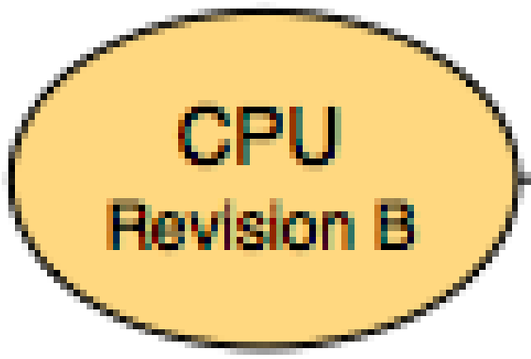
- ISA release vs revisions



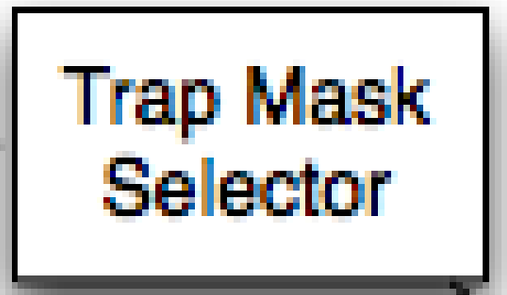
- ISA release vs revisions

Emulation

- Old software revision executing on new processor revision leads to **backward compatibility** issues
- Solution: software emulation mechanism via **CPU generated traps**.
- Allows non-sequential ISA evolution disputes over new extensions (XOP, FMA4, ...): vendors could emulate each other instructions using the trap mechanism.



Revision Mismatch



Execution hits opcode ***1h 4h***

Revision A	
Opcode	Trap?
0h	**N**
...	...
1h 4h	**Y**
...	...

...

...

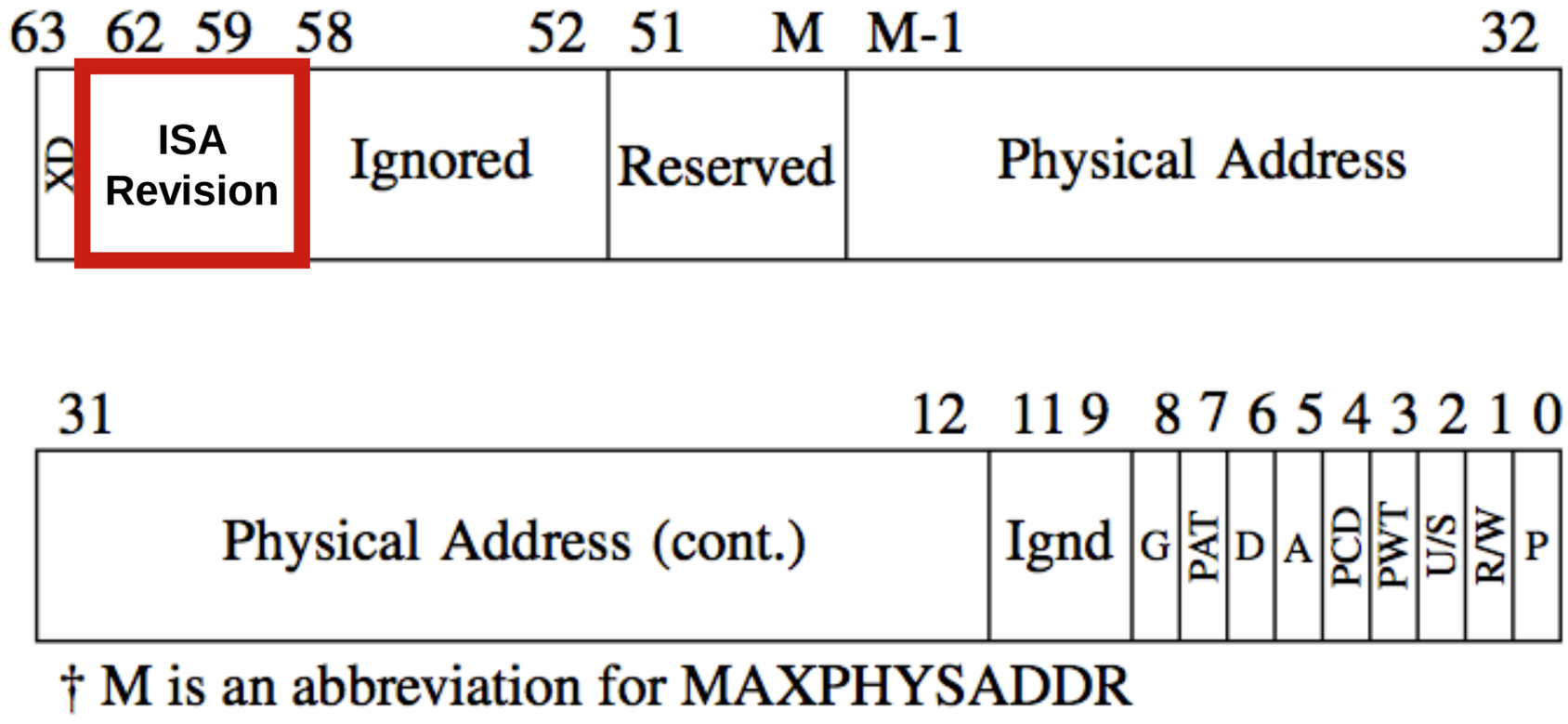
Trap Mask Vectors for revisions A against Z

Emulation

- Emulation must avoid using outdated instructions
- Emulation Routines:
  - **Operating System**
  - Firmware

Hardware

- 4-bits for ISA revision: extend PTE & TLB (6% increase in Core i7 920)
- Processor Front-end



† M is an abbreviation for MAXPHYSADDR

Software Support

- Linker
- Operating System Loader
- Executable header annotated with software revision

Evaluation

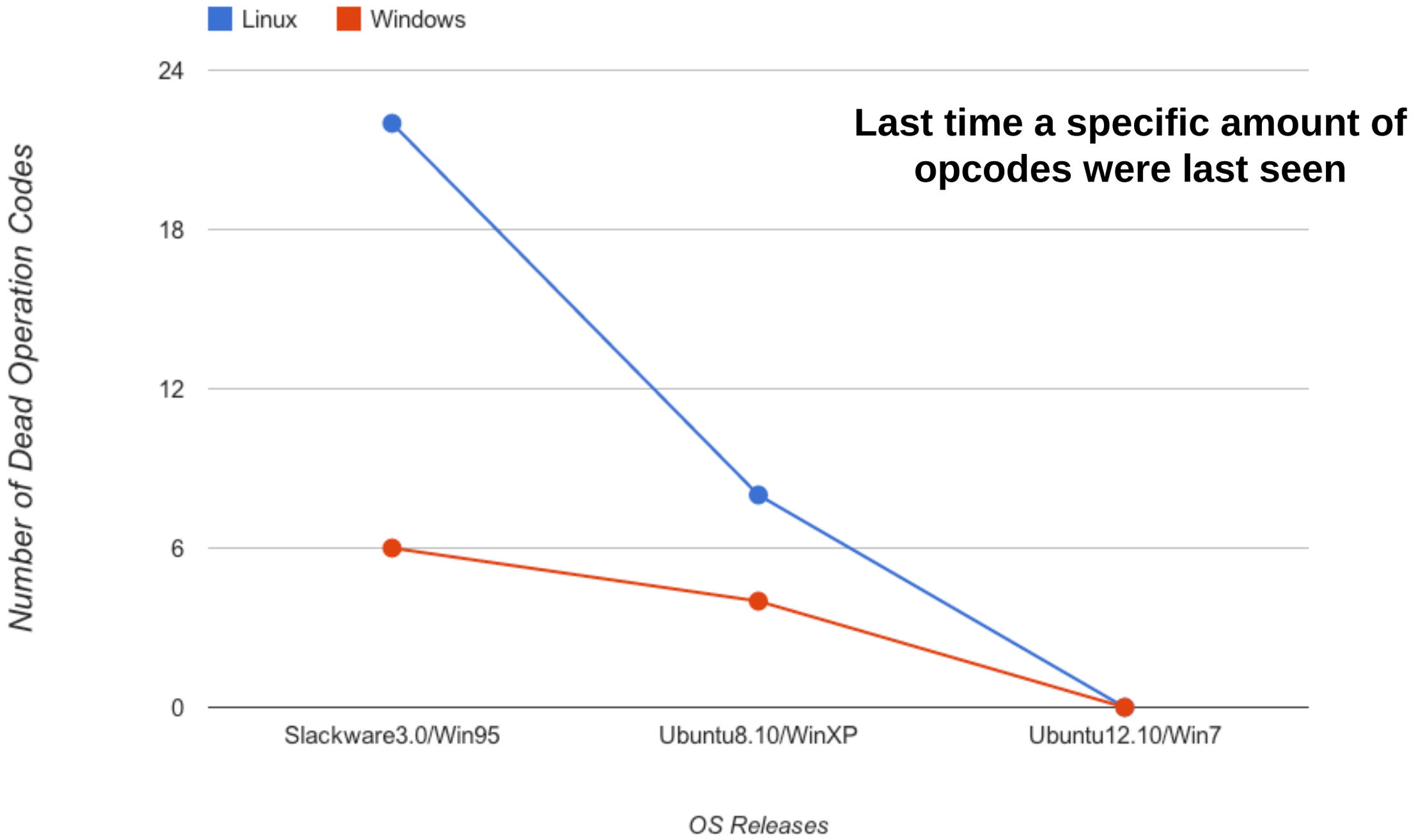
- Static and Dynamic instruction analysis of Linux and Windows from 1995-2012

Static	Dynamic	Release	Operating System	Additional Software
Yes	Yes	1996-1997	Slackware Linux 3	Netscape 4.0.1, StarOffice 3.1
Yes	Yes	2003-2004	Ubuntu 4.10	Firefox 0.9.2, OpenOffice 1.1.2
Yes	No	2005-2006	Ubuntu 6.10	
Yes	No	2006-2007	Ubuntu 7.10	
Yes	Yes	2007-2008	Ubuntu 8.10	Firefox 3.0.3, OpenOffice 2.4
Yes	No	2009-2010	Ubuntu 10.10	
Yes	Yes	2011-2012	Ubuuntu 12.04	Firefox 11, LibreOffice 3.5
Yes	Yes	1995-1996	Windows 95	I.E. 3, Office 95
Yes	Yes	1998-2000	Windows 98 SE	I.E. 5, Office 2000
Yes	Yes	2001-2004	Windows XP SP2	I.E. 6, Office 2003
Yes	Yes	2007-2009	Windows Vista	I.E. 7, Office 2007
Yes	Yes	2010-2012	Windows 7 SP1	I.E. 8, Office 2010

Static Analysis

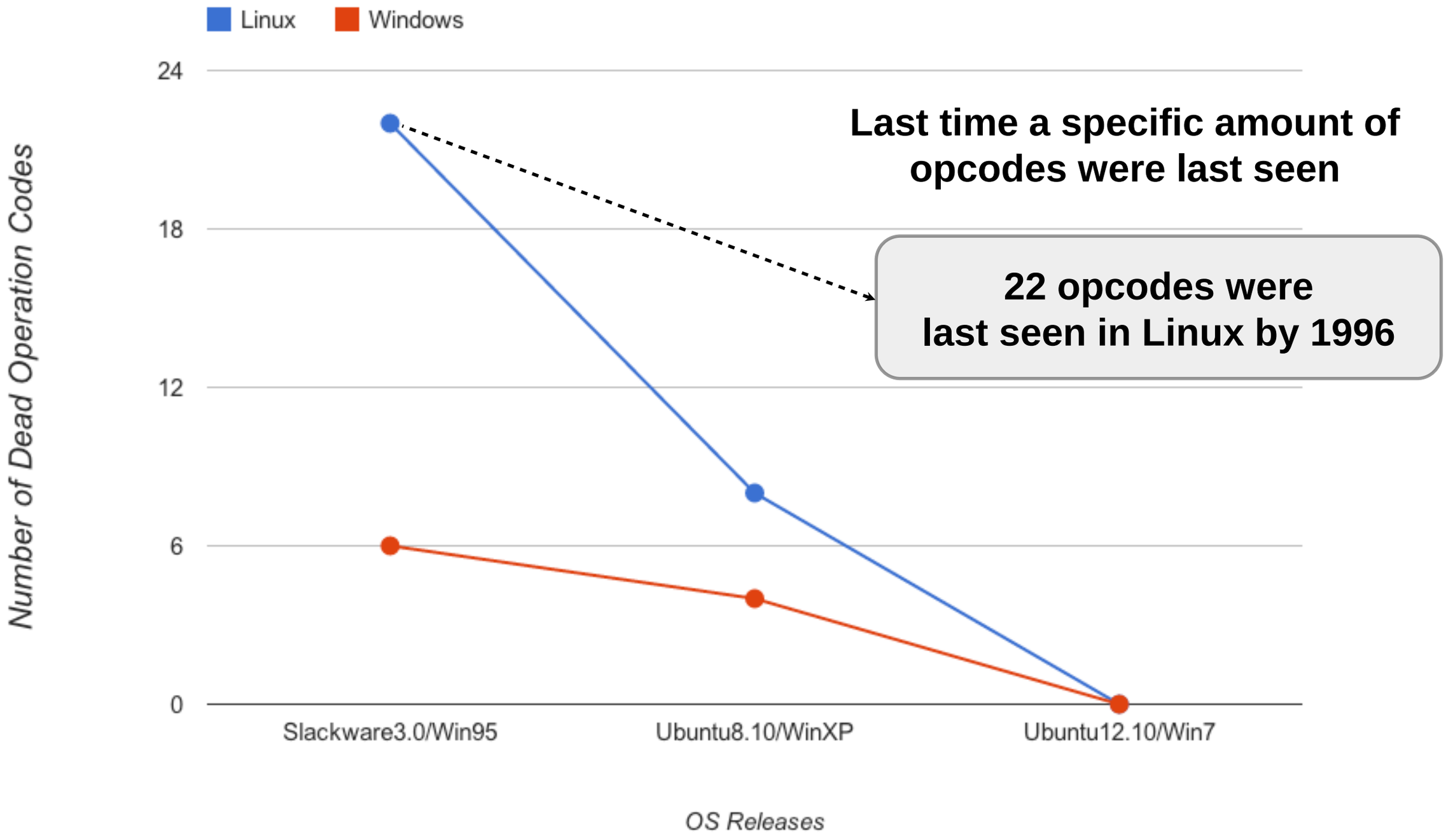
- 505 unused instruction opcodes in all disks (**30%** of all 32-bit opcodes)
  - 80% multimedia instructions - still on adoption
    - *There were no unused 1 and 2 bytes opcodes*
- From 1995 to 2012:
  - **30** instructions disappeared in Linux and **10** on windows.

Dead Instructions

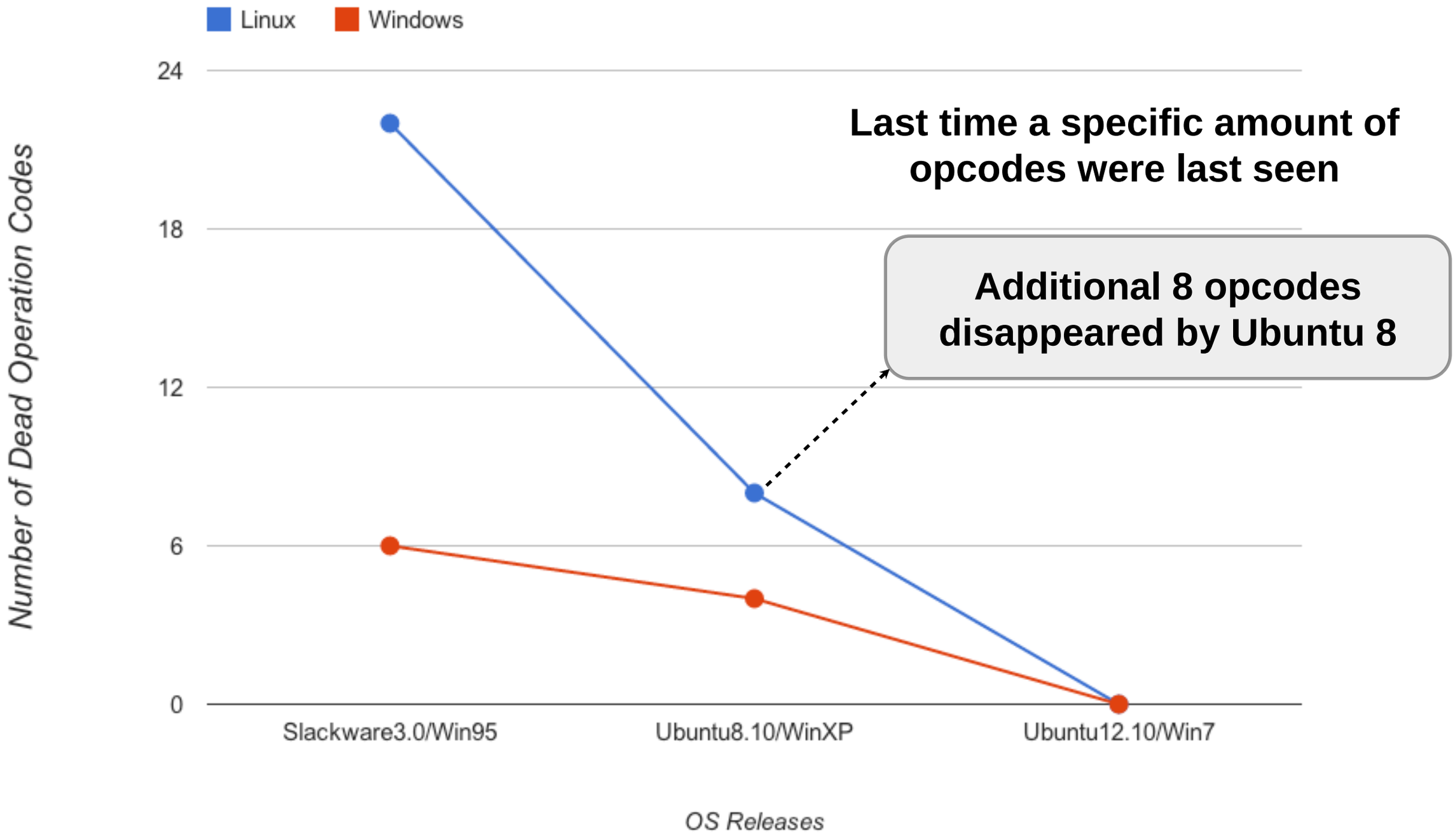


Last time a specific amount of opcodes were last seen

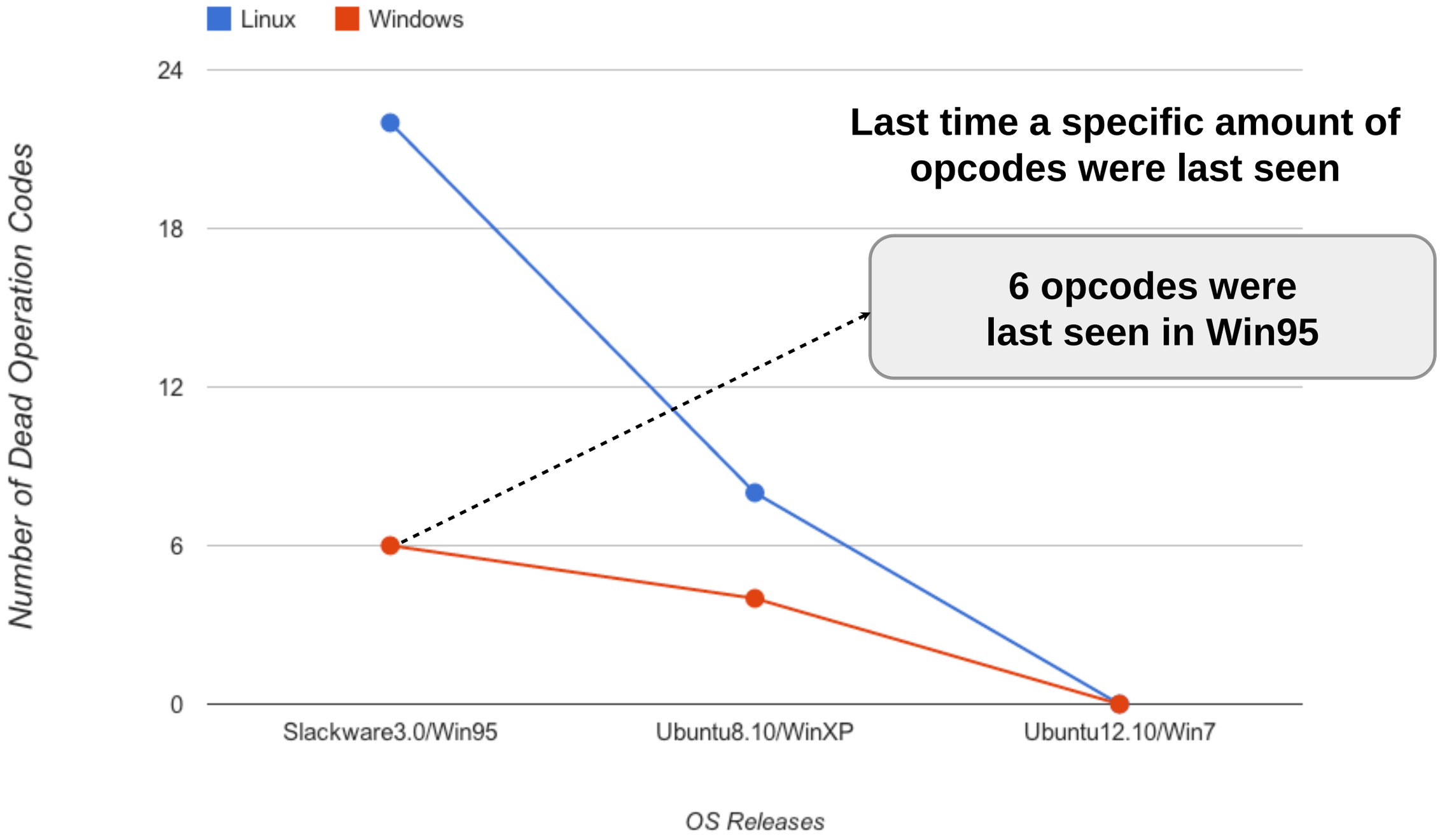
Dead Instructions



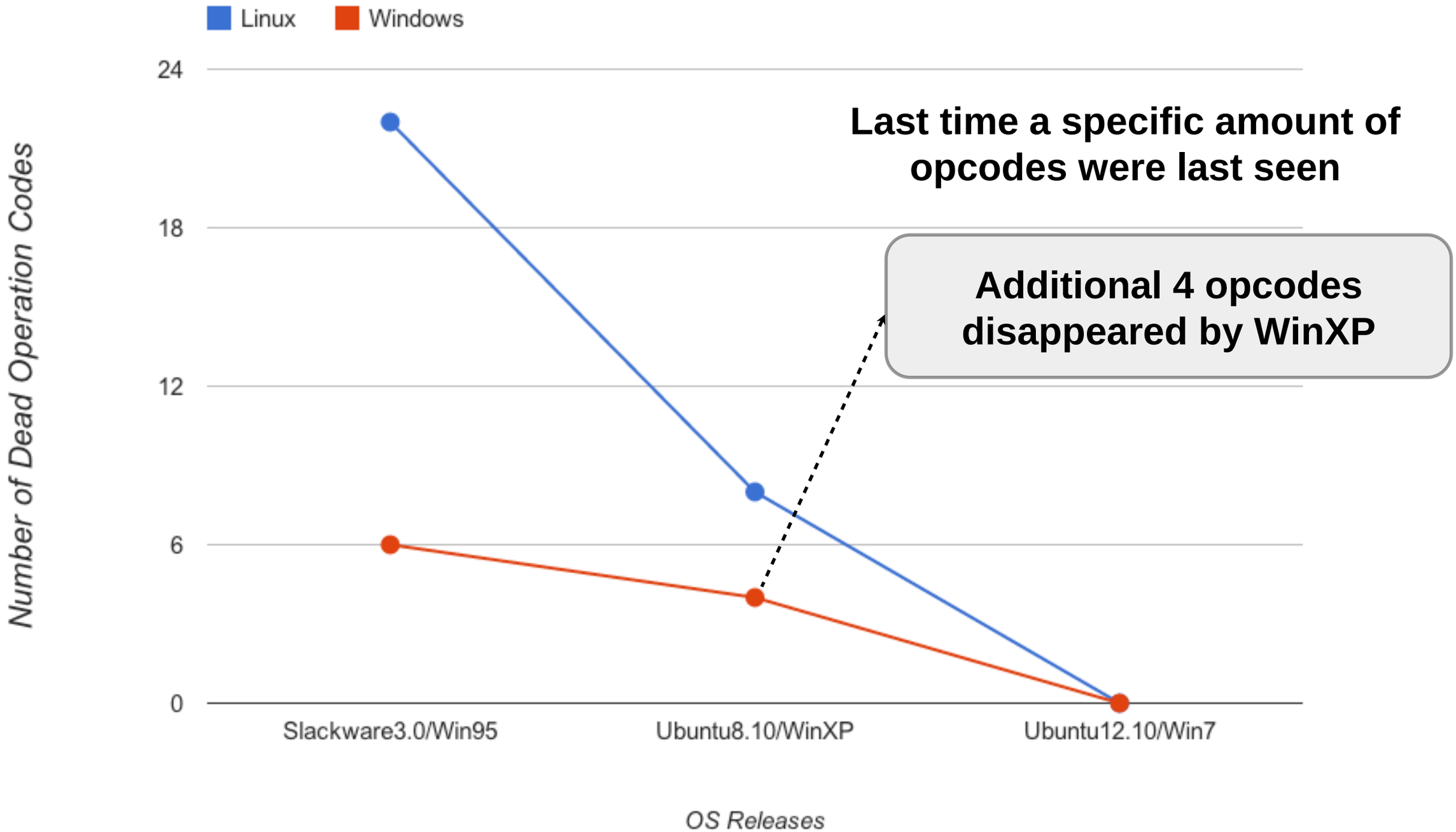
Dead Instructions



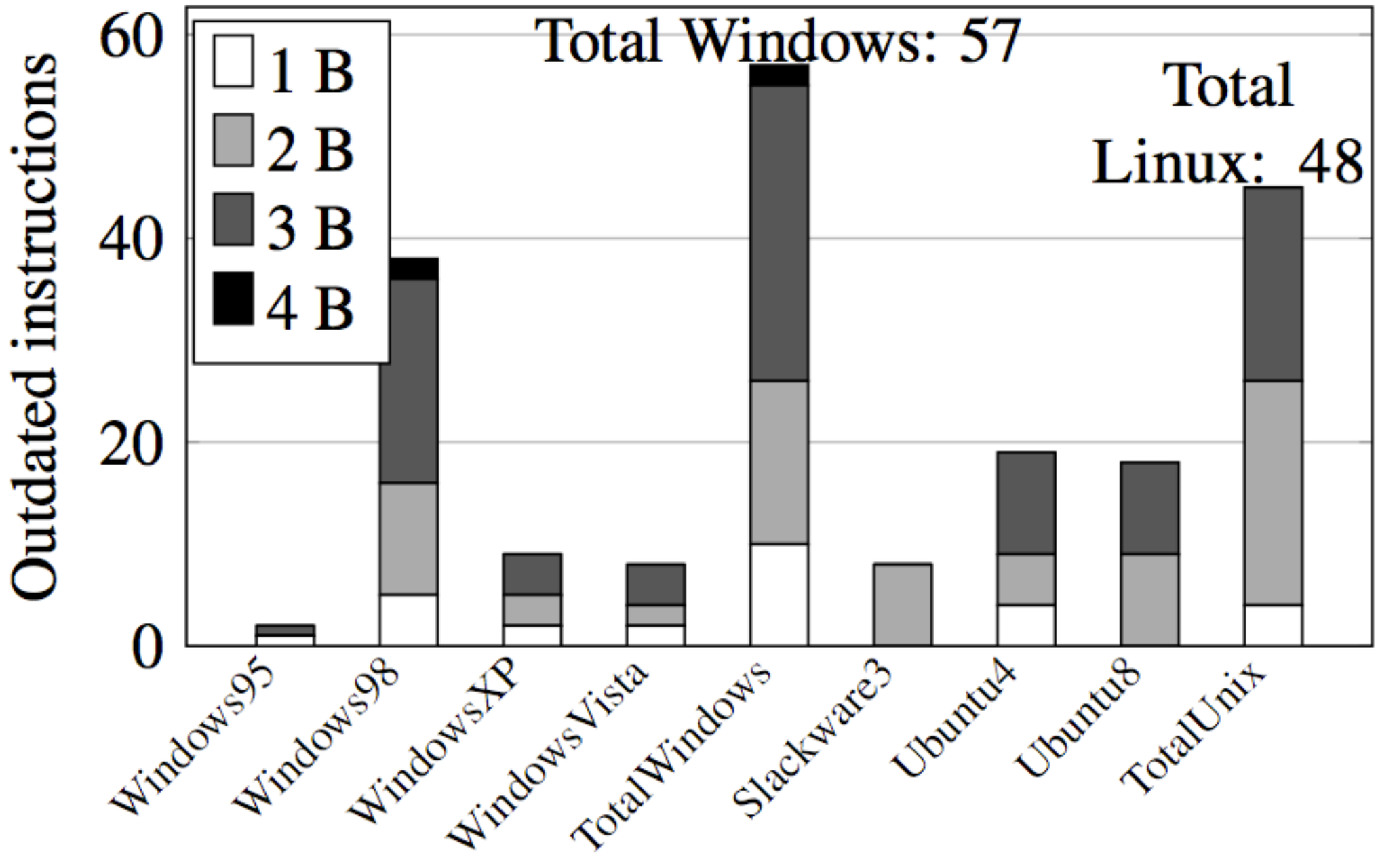
Dead Instructions



Dead Instructions

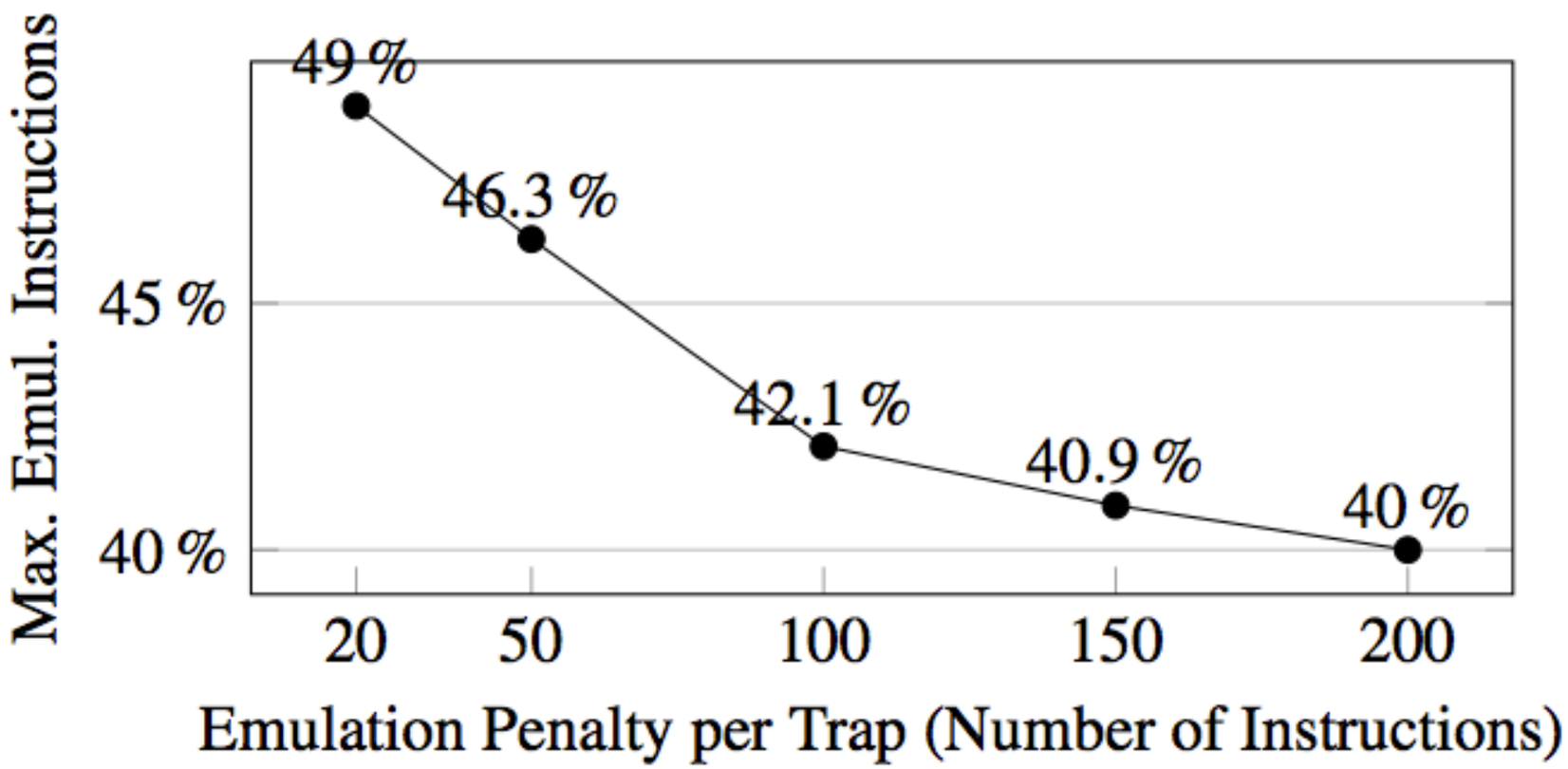


Dynamic Analysis



Emulation Overhead

- Experiment - Linux kernel trap implementation
- Tolerating a 5% overhead: we can re-encode 40% of the x86 ISA

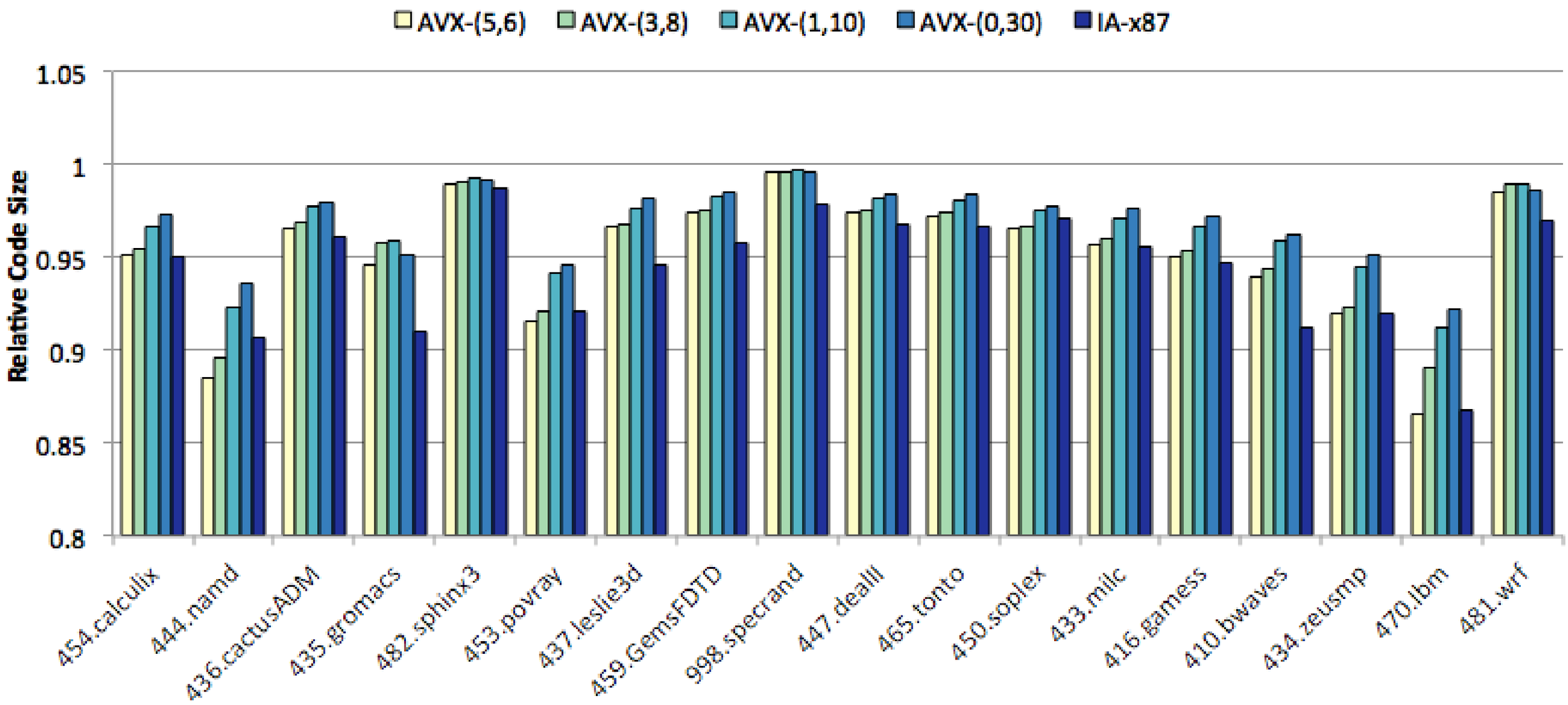


Case Study
AVX Re-encoding

- Re-encode most used AVX instructions using 1-byte and 2-byte opcodes
- Several scenarios = AVX(n, m):
  - n - number of reused **1-byte** opcodes
  - m - number of reused **2-byte** opcodes

SPEC2006FP - Code Size

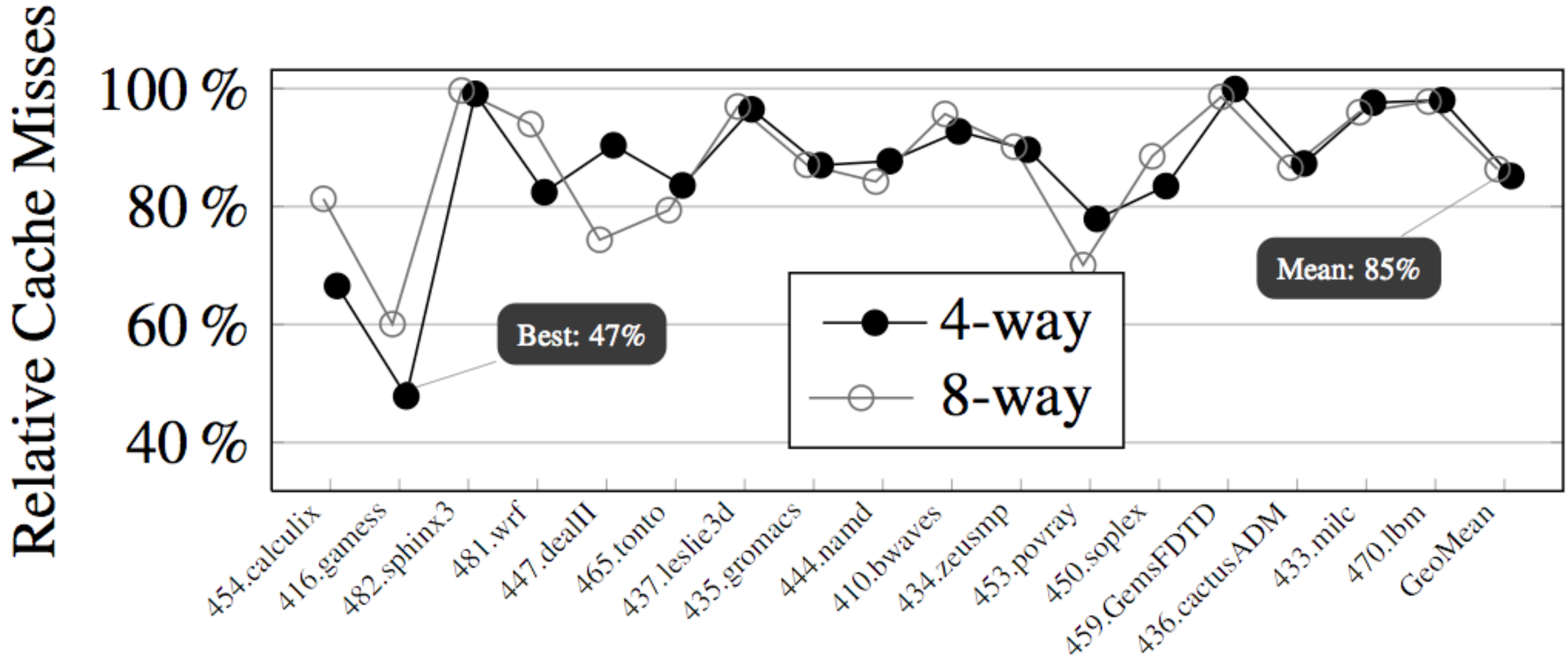
- AVX-(5,6) is 5.3% smaller on average



Relative to the original compiled AVX version

Cache Effects

AVX(5,6)



32K L1 I-Cache

Conclusion

- Static and Dynamic analysis shows that a great number of x86 instructions are obsolete.
- Recycling mechanism: re-encoding instructions without breaking backward compatibility
- Tolerating a 5% overhead: we can re-encode 40% of the x86 ISA
- Case study: AVX re-encoding yields 5.3% smaller binaries and reduction up to 53% in cache misses.

UNICAMP

Questions?