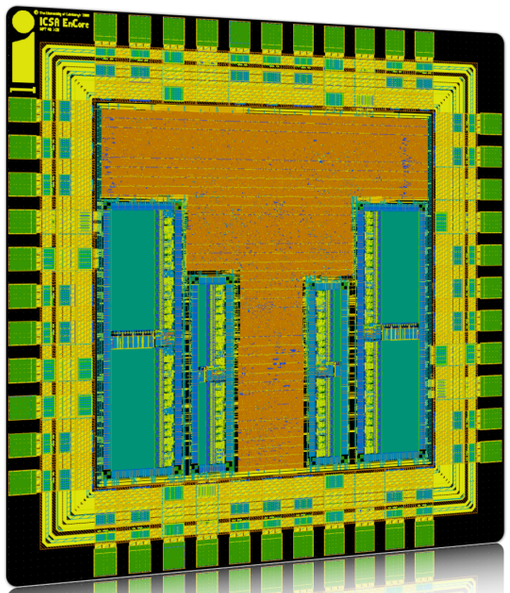
THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

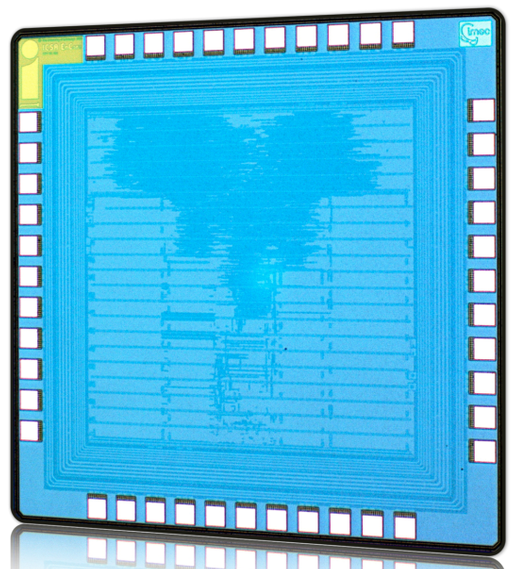
Compact Code Generation

Tobias Edler von Koch
Igor Böhm and Björn Franke



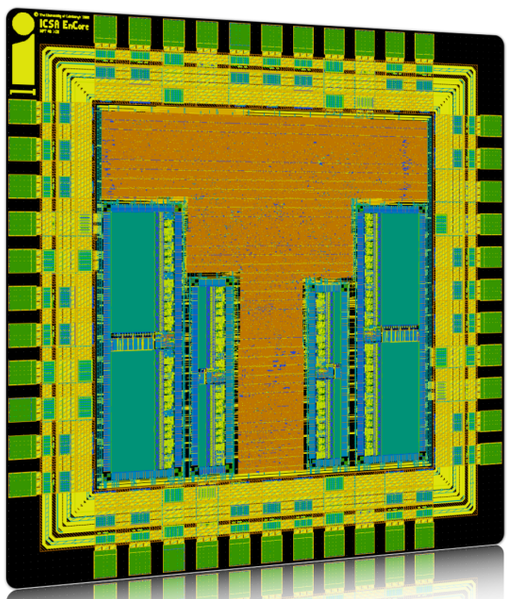
PASTA

Processor **A**utomated **S**ynthesis
by i**T**erative **A**nalysis



Integrated Instruction Selection and Register Allocation for Compact Code Generation Exploiting Freeform Mixing of 16- and 32-bit Instructions

Tobias Edler von Koch
Igor Böhm and Björn Franke



PASTA

Processor **A**utomated **S**ynthesis by i**T**erative **A**nalysis

Does Code Size Matter?

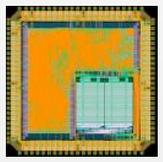
128-Mbit Flash
27.3mm² at 0.13µm

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

Does Code Size Matter?

128-Mbit Flash
27.3mm² at 0.13µm



Thumb-2 ISA

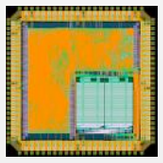
ARM Cortex M3
0.43mm² at 0.13µm

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

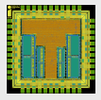
Does Code Size Matter?

128-Mbit Flash
27.3mm² at 0.13µm



Thumb-2 ISA

ARM Cortex M3
0.43mm² at 0.13µm



ARCompact ISA

ENCORE Calton
0.15mm² at 0.13µm

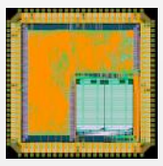
PASTA
Processor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

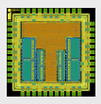
Does Code Size Matter?

128-Mbit Flash
27.3mm² at 0.13µm



Thumb-2 ISA

ARM Cortex M3
0.43mm² at 0.13µm



ARCompact ISA

ENCORE Calton
0.15mm² at 0.13µm

RISC Architectures

sacrifice code density
in order to simplify
implementation circuitry
and decrease die area

PASTA
Processor **A**utomated **S**ynthesis
by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

Solution to Code Size Problem

- Dual instruction sets providing 32-bit and 16-bit instruction encodings:

Solution to Code Size Problem

- Dual instruction sets providing 32-bit and 16-bit instruction encodings:

microMIPS

ARM Thumb-2

ARCompact

ARM Thumb

Solution to Code Size Problem

- Dual instruction sets providing 32-bit and 16-bit instruction encodings:

microMIPS

ARM Thumb-2

ARCompact

ARM Thumb

- There's no such thing as a free lunch!

Solution to Code Size Problem

- Dual instruction sets providing 32-bit and 16-bit instruction encodings:

microMIPS

ARM Thumb-2

ARCompact

ARM Thumb

- There's no such thing as a free lunch!
- 16-bit instructions come with constraints!

Common Compact Instruction Format Constraints

- Only subset of registers accessible

Common Compact Instruction Format Constraints

- Only subset of registers accessible
- Explicit instructions necessary to switch between 16-bit and 32-bit modes

Common Compact Instruction Format Constraints

- Only subset of registers accessible
- Explicit instructions necessary to switch between 16-bit and 32-bit modes
- Reduced size of immediate operands

Common Compact Instruction Format Constraints

- Only subset of registers accessible
- Explicit instructions necessary to switch between 16-bit and 32-bit modes
- Reduced size of immediate operands
- Not every 32-bit instruction has a 16-bit counterpart

Common Compact Instruction Format Constraints

- Only subset of registers accessible
- Explicit instructions necessary to switch between 16-bit and 32-bit modes
- Reduced size of immediate operands
- Not every 32-bit instruction has a 16-bit counterpart
- Free mixing of 16- and 32-bit encodings not always possible

ARCompact 16-bit Instruction Format Constraints

- Only subset of registers accessible
- ~~Explicit instructions necessary to switch between 16-bit and 32-bit modes~~
- Reduced size of immediate operands
- Not every 32-bit instruction has a 16-bit counterpart
- **Free** mixing of 16- and 32-bit encodings ~~not always possible~~

Motivating Example

32-Bit Only

Basic Block

```
ld  r2,[sp,0]
ld  r3,[sp,4]

ld  r4,[sp,8]
add r2,r2,r3
asl r2,r2,2
sub r2,r2,r4
```

24 bytes

Motivating Example

Basic Block

32-Bit Only	Mixed Mode Aggressive

`ld  r2,[sp,0]`
`ld  r3,[sp,4]`
 	
`ld  r4,[sp,8]`
`add r2,r2,r3`
`asl r2,r2,2`
`sub r2,r2,r4`
	`ld_s r2,[sp,0]`
	`ld_s r3,[sp,4]`
	
	`ld_s rX,[sp,8]`
	`add_s r2,r2,r3`
	`asl_s r2,r2,2`
	`sub_s r2,r2,rX`

24 bytes	12 bytes

Instruction Selection

Motivating Example

Basic Block

32-Bit Only	Mixed Mode Aggressive

`ld  r2,[sp,0]`
`ld  r3,[sp,4]`
 	
`ld  r4,[sp,8]`
`add r2,r2,r3`
`asl r2,r2,2`
`sub r2,r2,r4`
	`ld_s r2,[sp,0]`
	`ld_s r3,[sp,4]`
	`mov  r4,r1`
	`ld_s r1,[sp,8]`
	`add_s r2,r2,r3`
	`asl_s r2,r2,2`
	`sub_s r2,r2,r1`
	`mov  r4,r1`

24 bytes	20 bytes

Instruction Selection

Register Allocation

Motivating Example

Basic Block

32-Bit Only	Mixed Mode Aggressive	Mixed Mode Integrated
`ld  r2,[sp,0]`	`ld_s r2,[sp,0]`	`ld_s r2,[sp,0]`
`ld  r3,[sp,4]`	`ld_s r3,[sp,4]`	`ld_s r3,[sp,4]`
	`mov  r4,r1`	
`ld  r4,[sp,8]`	`ld_s r1,[sp,8]`	`ld  r4,[sp,8]`
`add r2,r2,r3`	`add_s r2,r2,r3`	`add_s r2,r2,r3`
`asl r2,r2,2`	`asl_s r2,r2,2`	`asl_s r2,r2,2`
`sub r2,r2,r4`	`sub_s r2,r2,r1`	`sub  r2,r2,r4`
	`mov  r4,r1`	
24 bytes	20 bytes	16 bytes

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

Efficient Compact Code Generation is an **Integrated** Instruction Selection and Register Allocation Problem!

PASTA
Processor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH
informatics

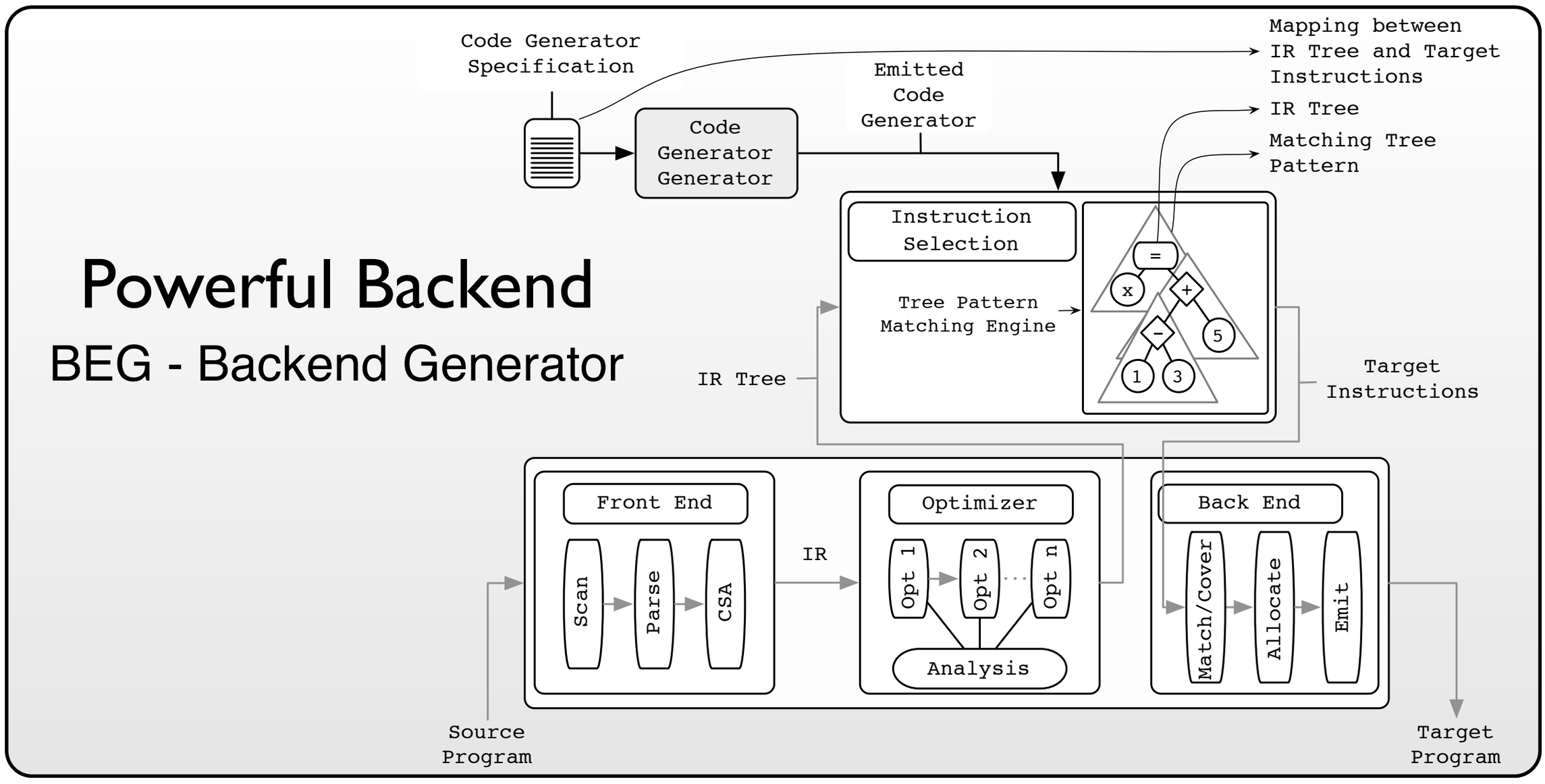
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

ECC - EnCore C Compiler based on commercial CoSy compiler by ACE©.

Powerful Backend

BEG - Backend Generator



PASTA
Processor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic

Feedback-Guided

THE UNIVERSITY of EDINBURGH
informatics

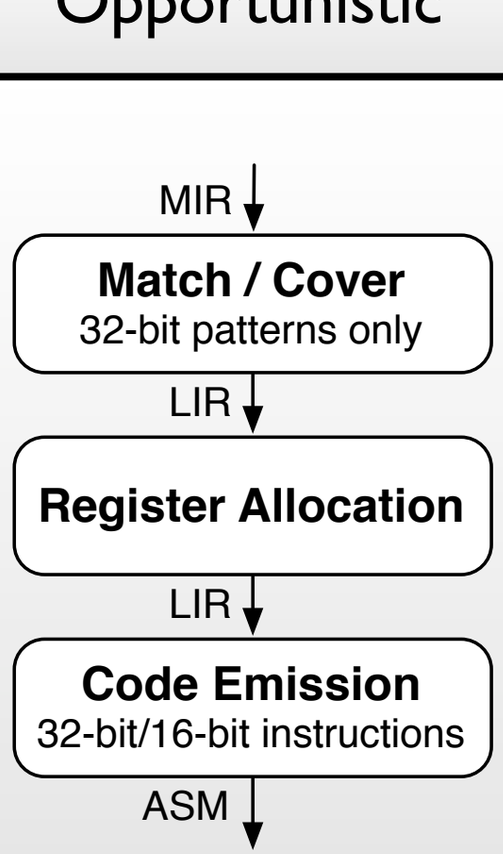
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided

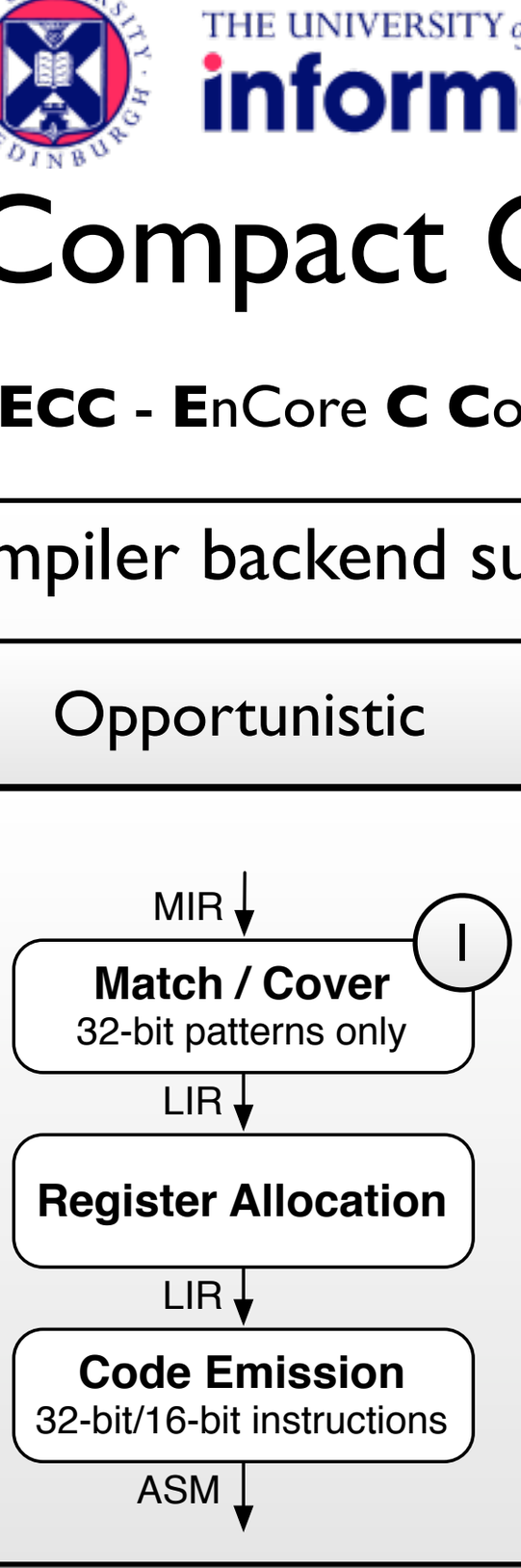
PASTA **P**rocessor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

Compact Code Generation Approach

ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



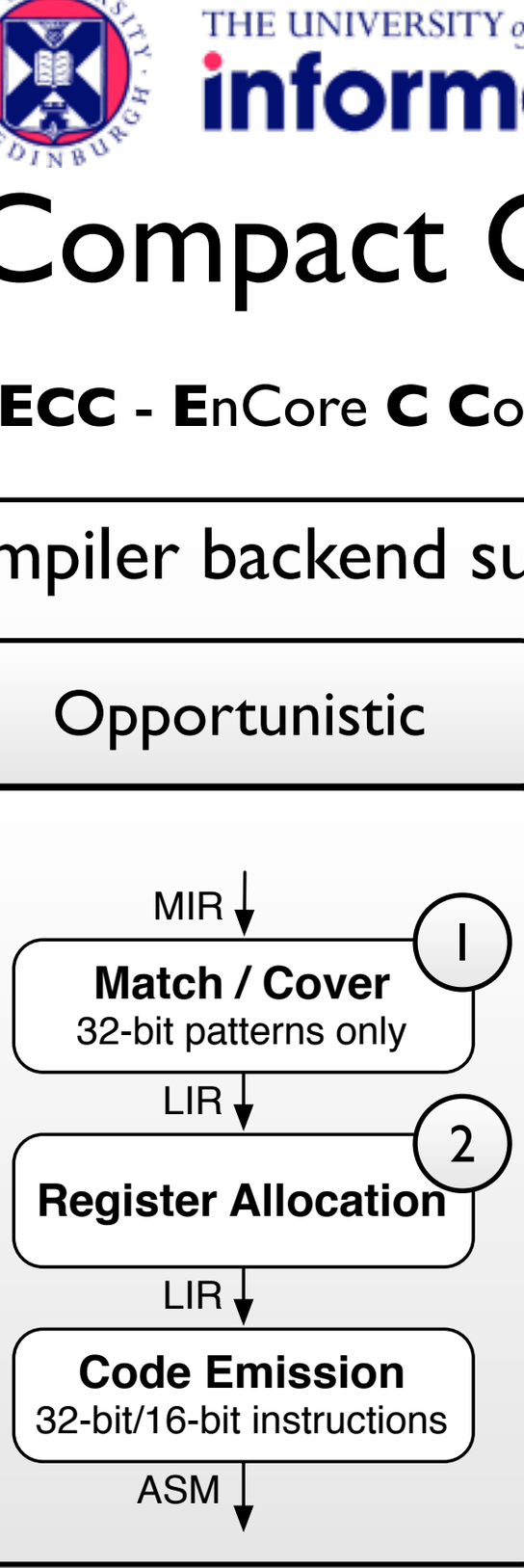
Feedback-Guided

Compact Code Generation Approach

ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



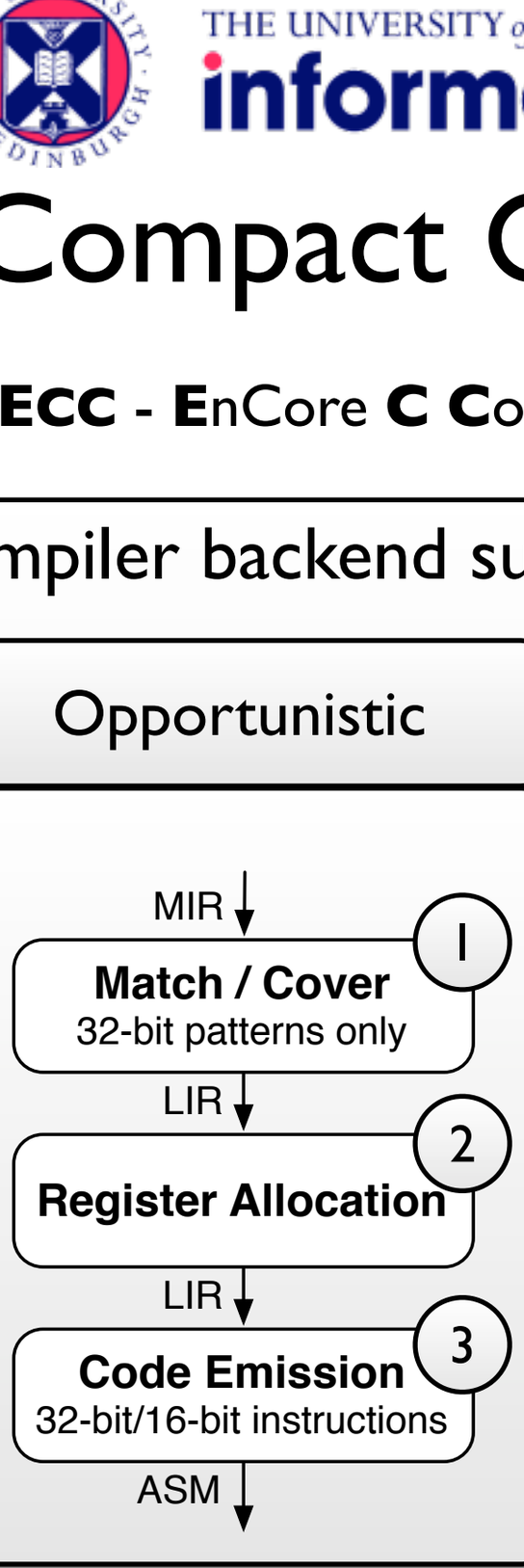
Feedback-Guided

Compact Code Generation Approach

ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided

THE UNIVERSITY of EDINBURGH informatics

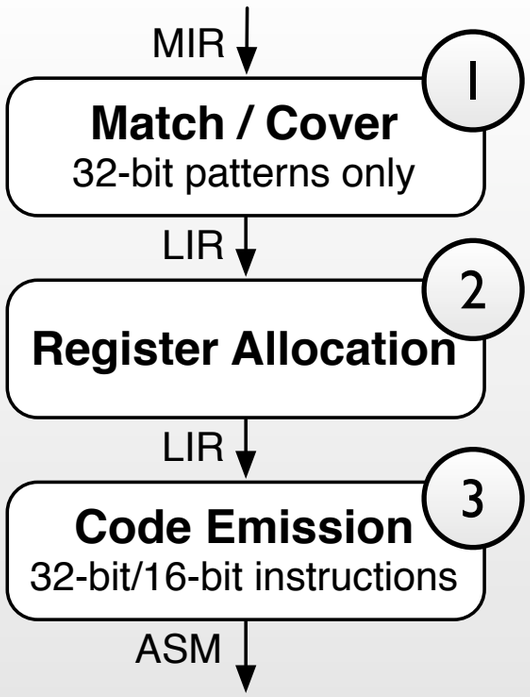
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

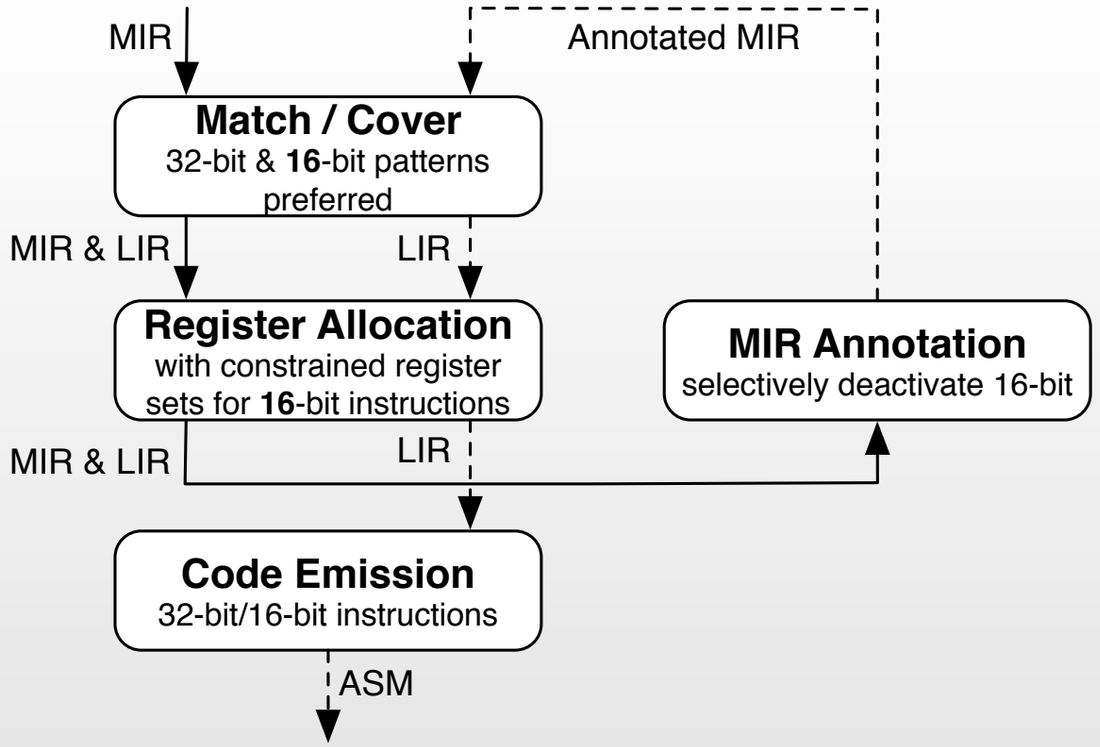
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

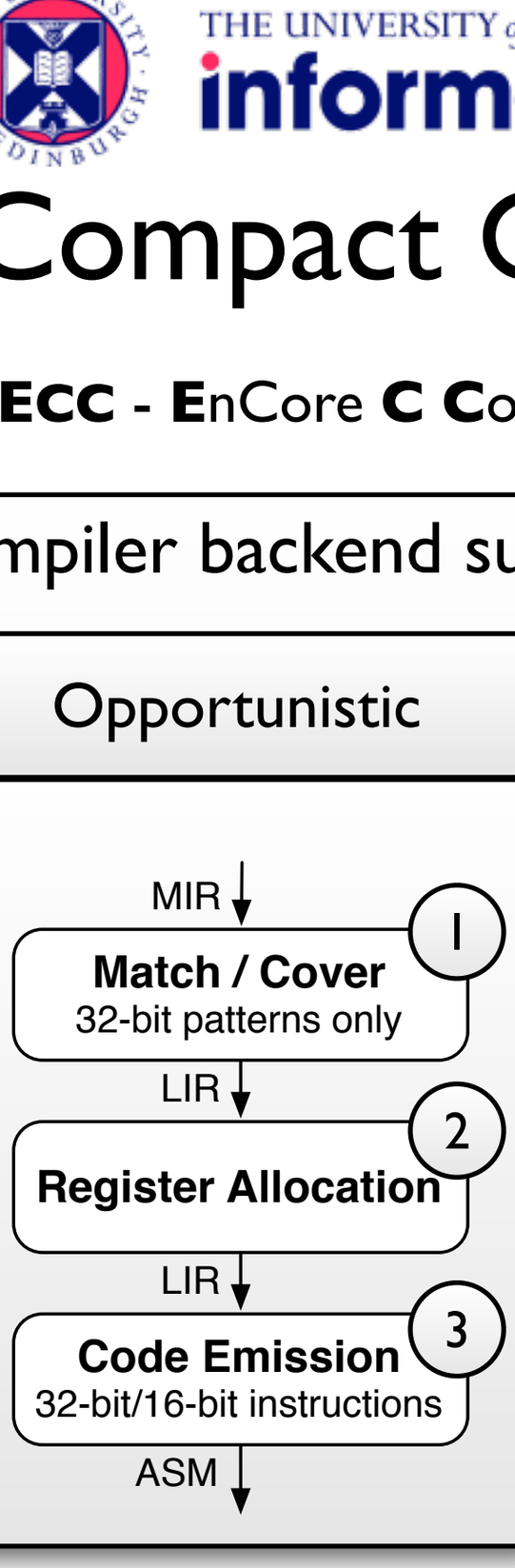
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

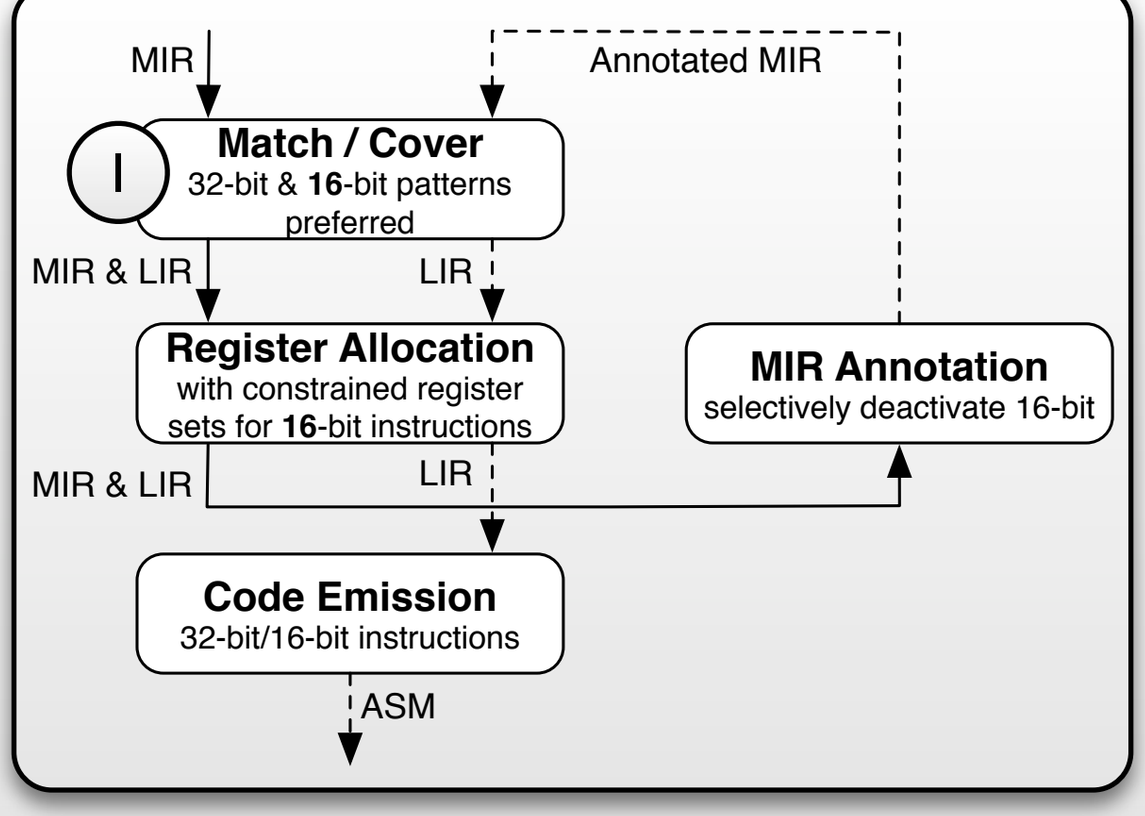
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



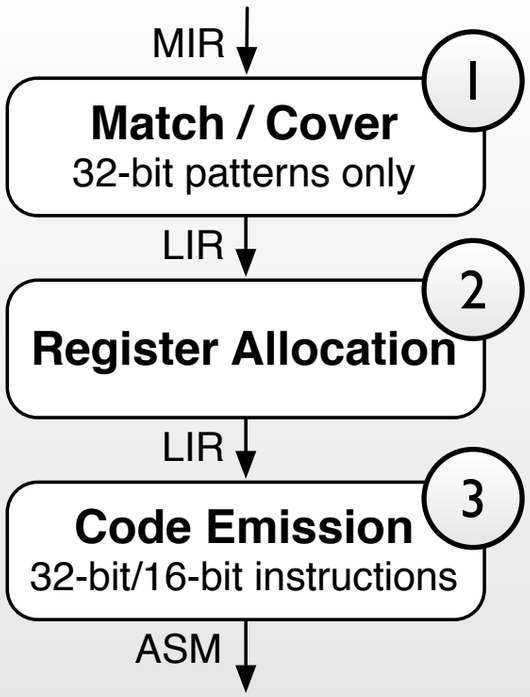
PASTA **P**rocessor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

Compact Code Generation Approach

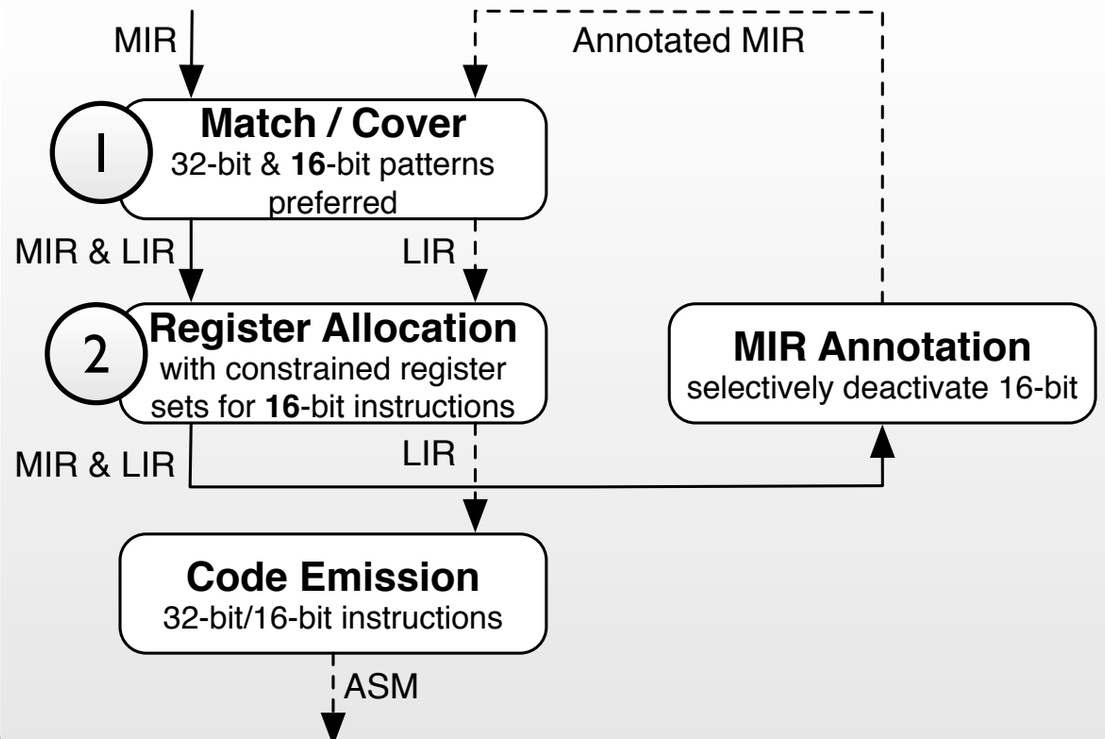
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



PASTA **P**rocessor **A**utomated **S**ynthesis by i**T**erative **A**nalysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

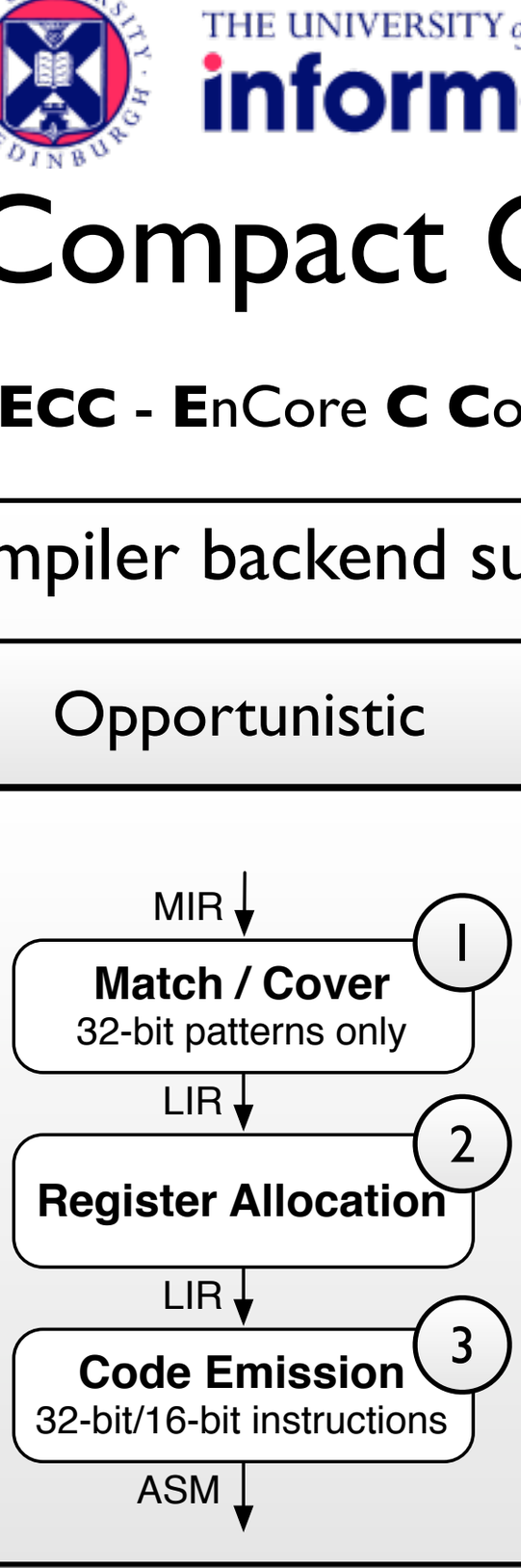
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

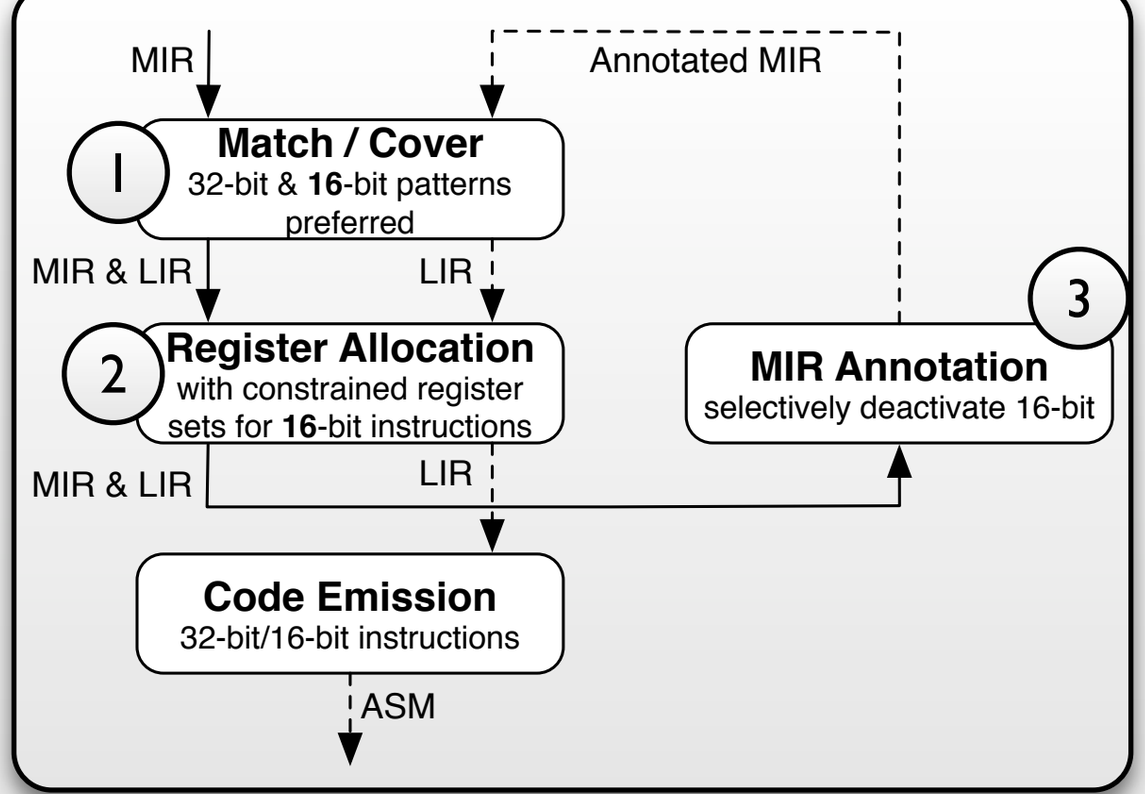
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



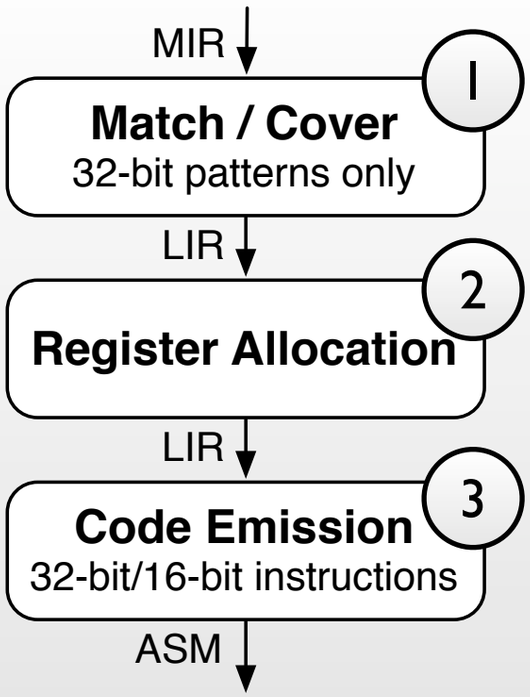
PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

Compact Code Generation Approach

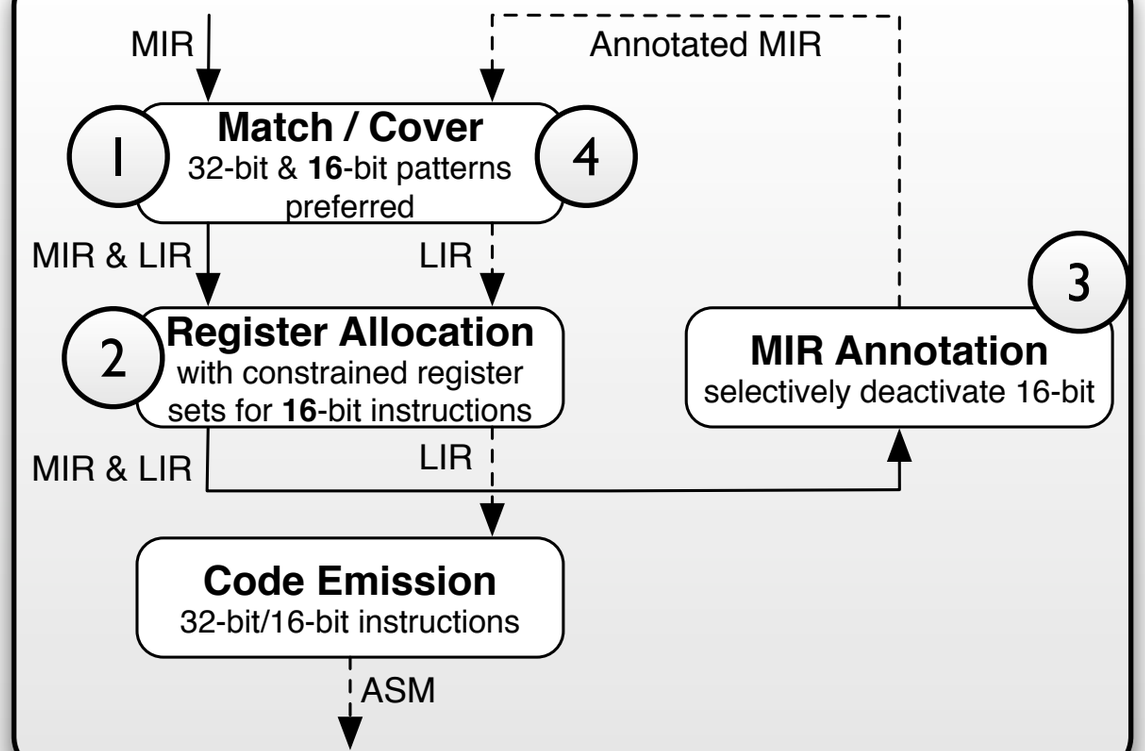
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

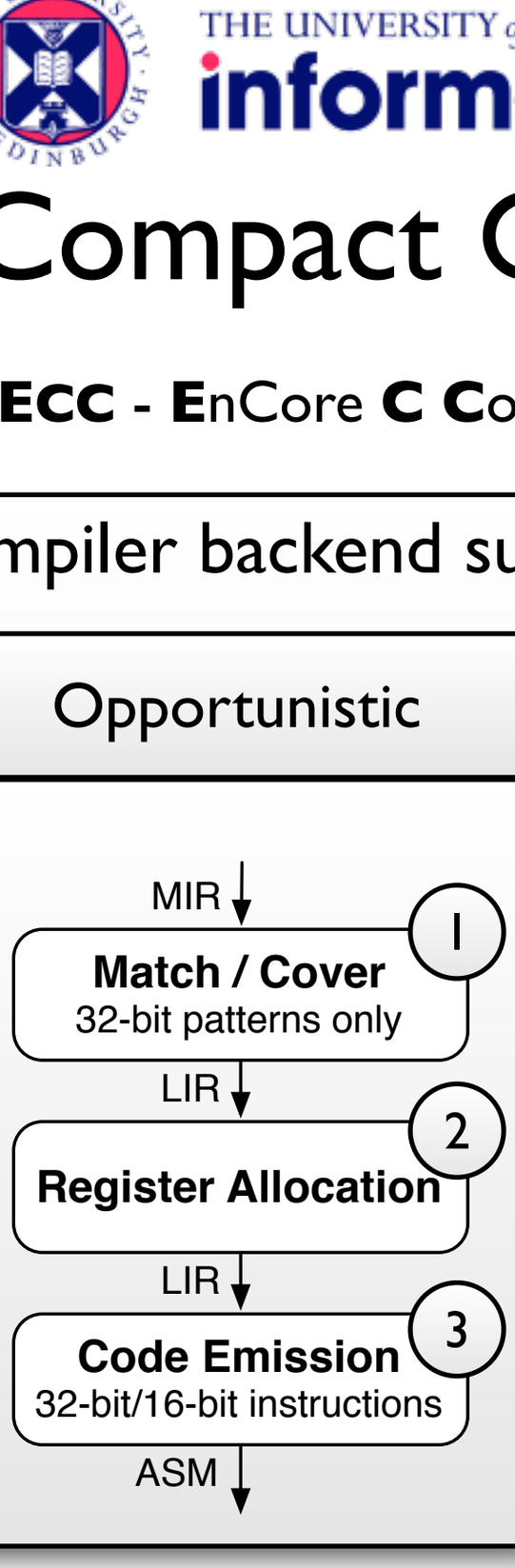
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

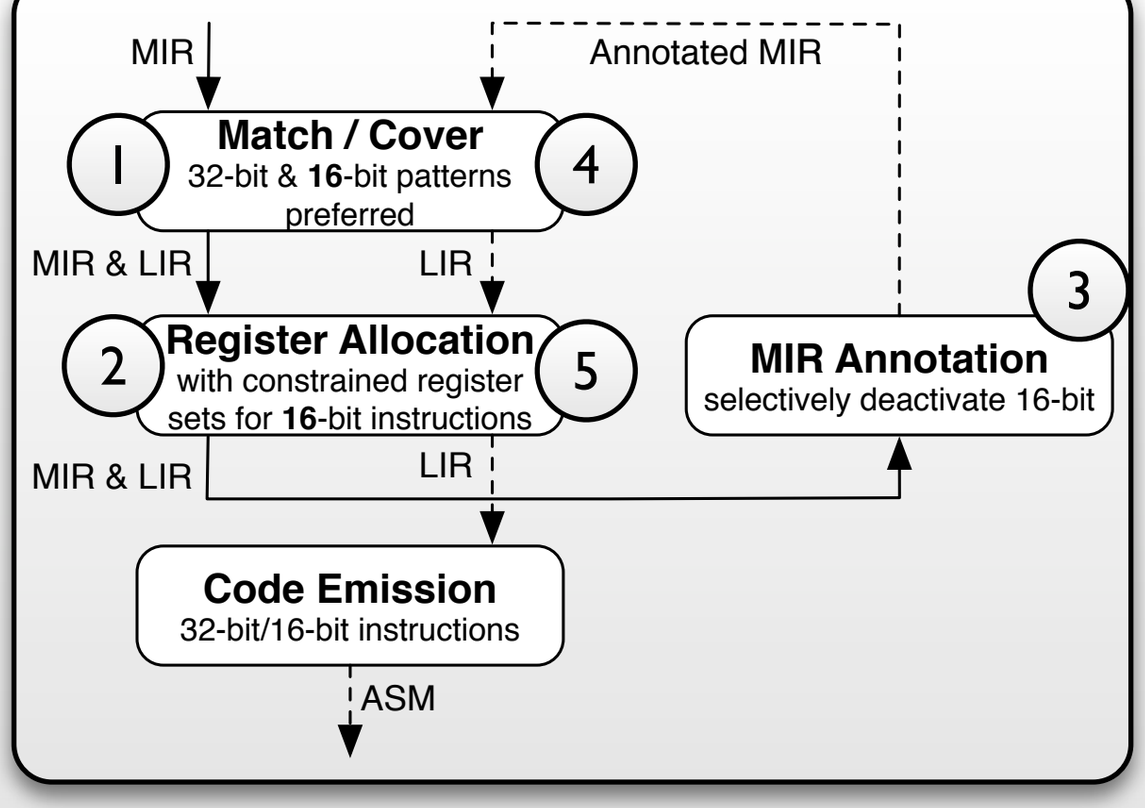
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

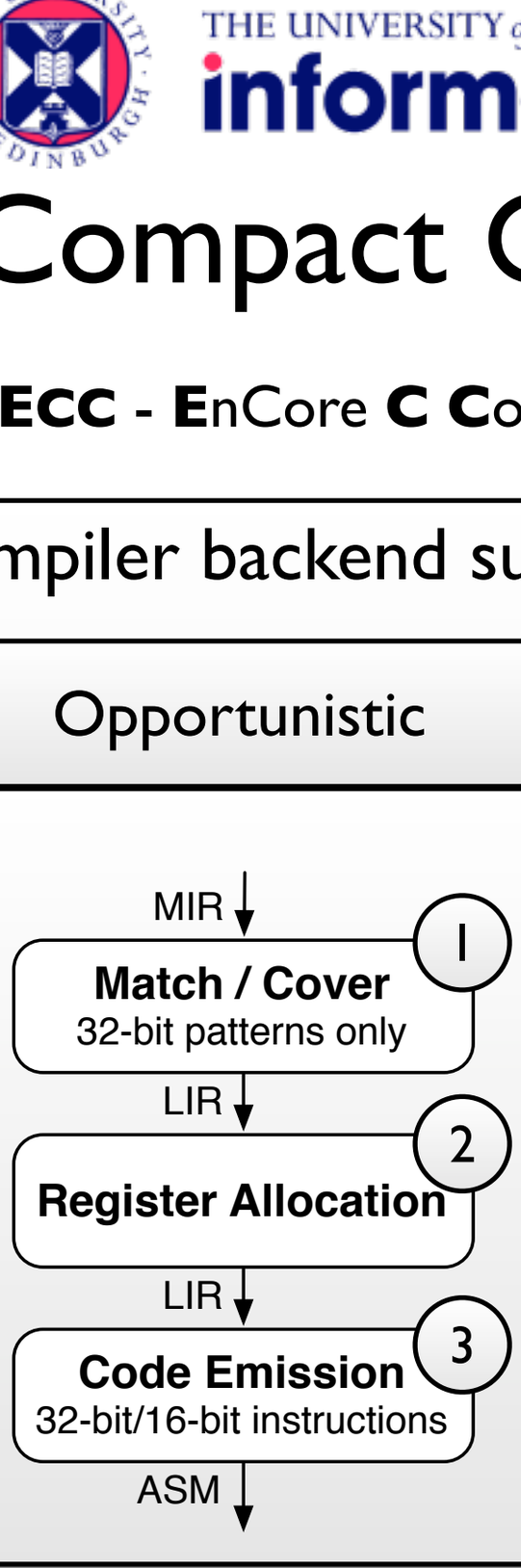
icsa | Institute for Computing Systems Architecture

Compact Code Generation Approach

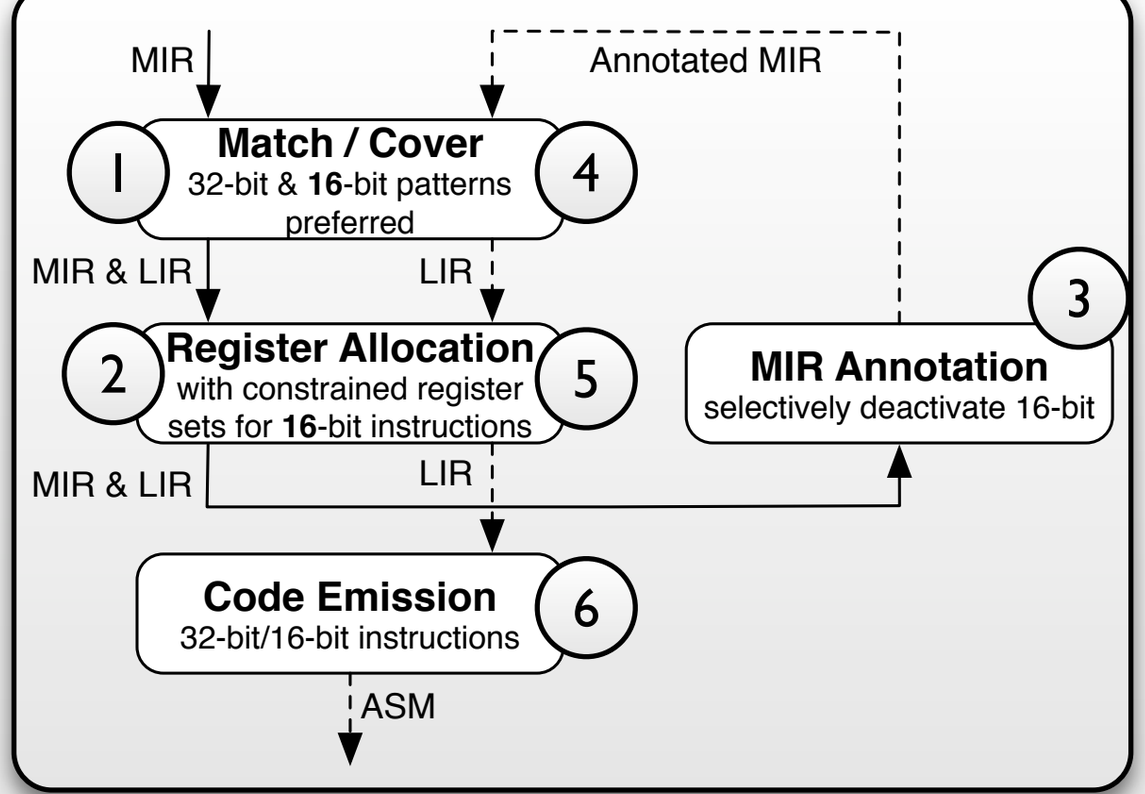
ECC - EnCore **C C**ompiler based on commercial **CoSy** compiler by **ACE©.**

Compiler backend supports **two** compact code generation strategies

Opportunistic



Feedback-Guided



PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

Feedback-Guided Instruction Selection

```
vX ... Virtual Register
rX ... Physical Register
```

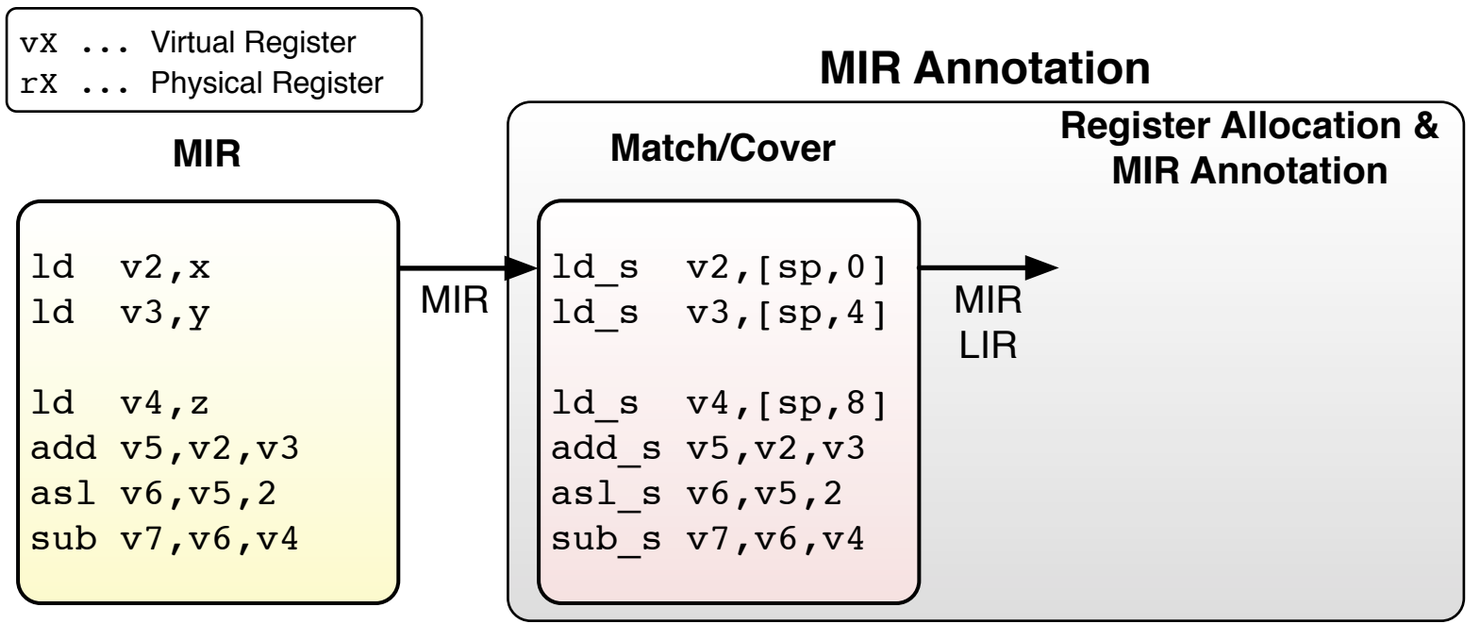
MIR

```
ld  v2,x
ld  v3,y

ld  v4,z
add v5,v2,v3
asl v6,v5,2
sub v7,v6,v4
```

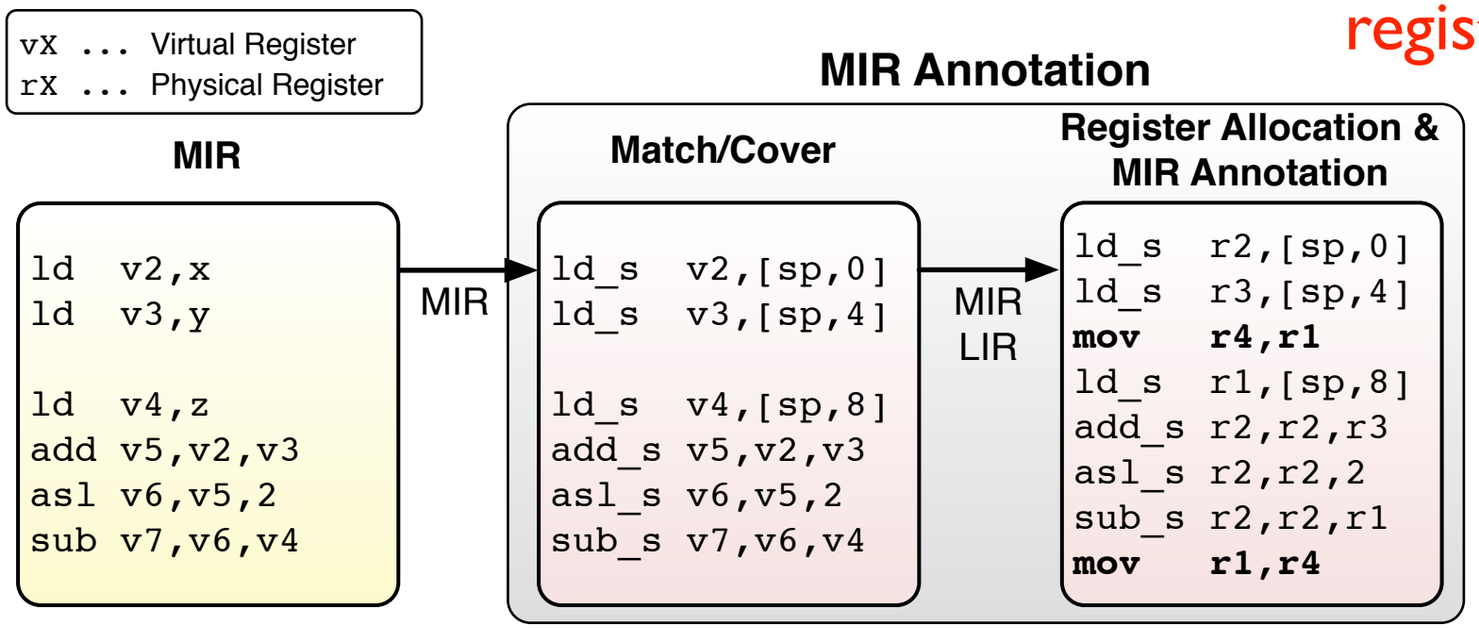
MIR →

Feedback-Guided Instruction Selection



aggressively select 16-bit instructions

Feedback-Guided Instruction Selection

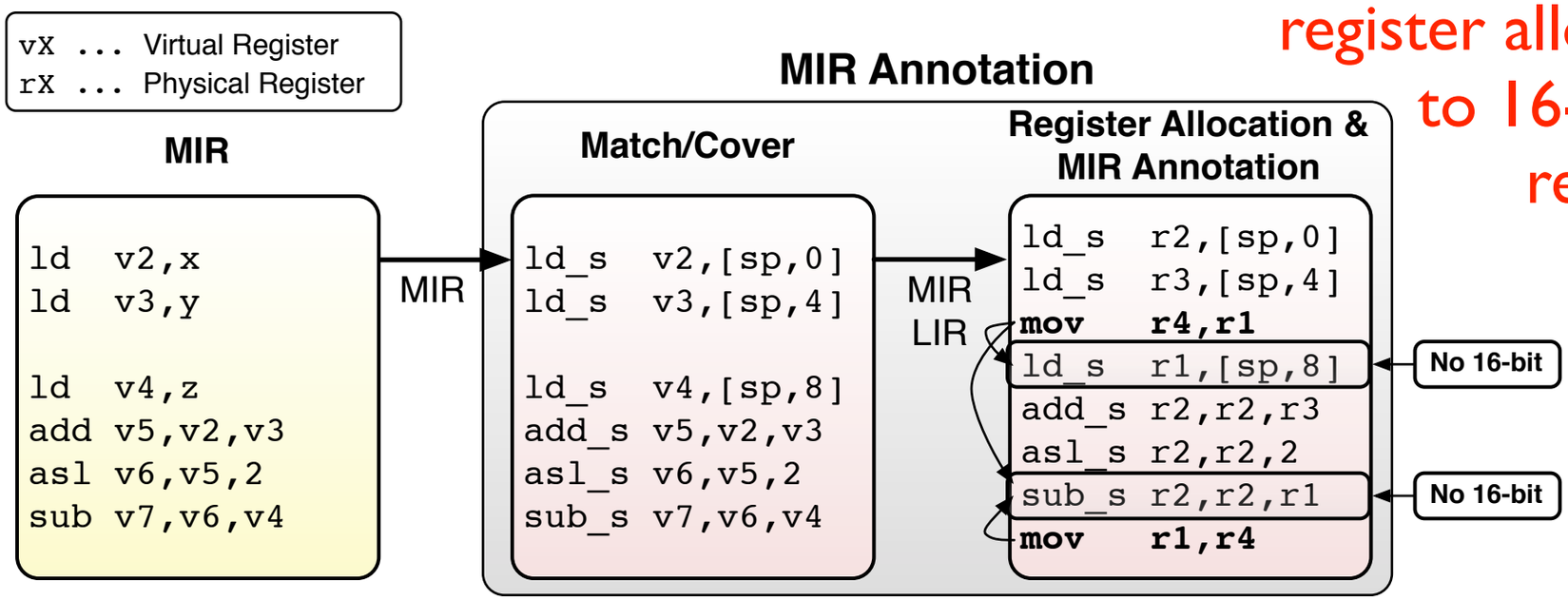


register allocator constrained to 16-bit accessible register set

aggressively select 16-bit instructions

Feedback-Guided Instruction Selection

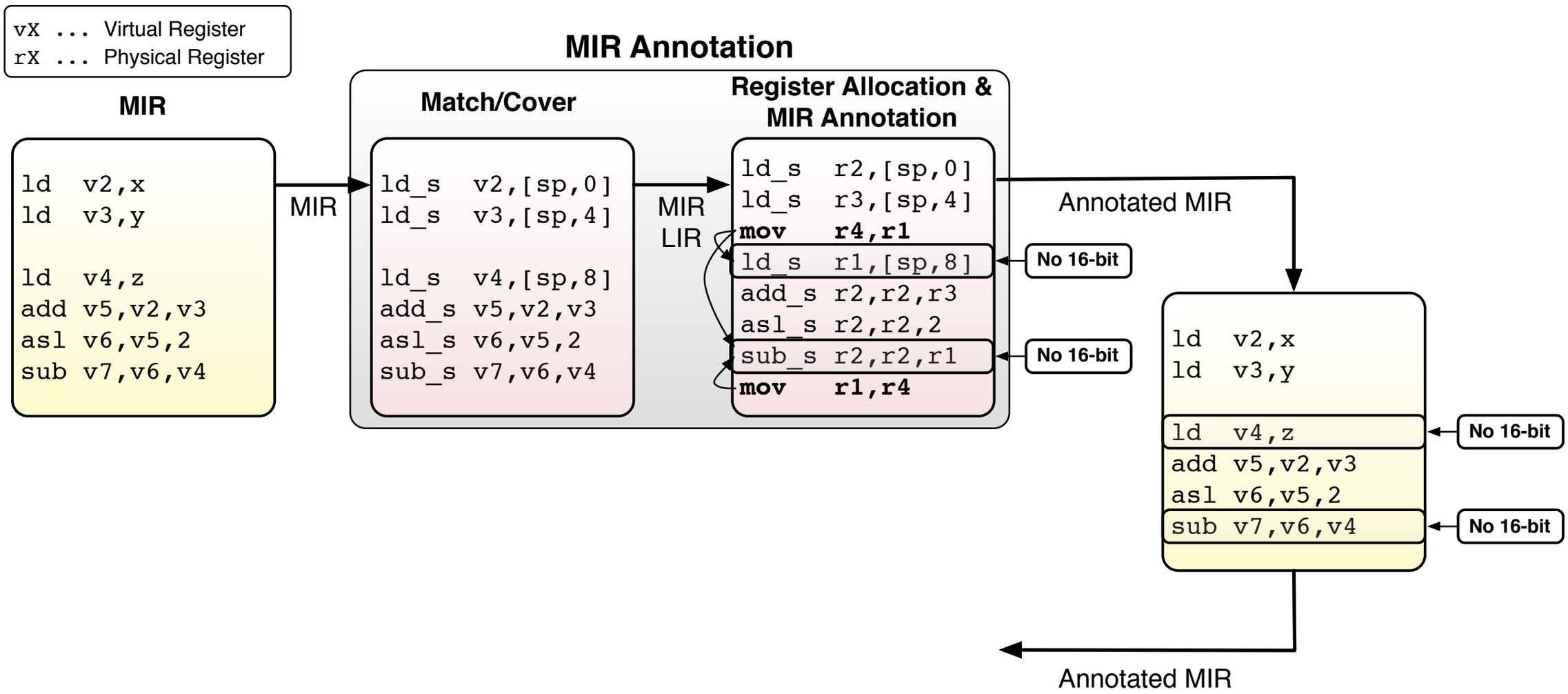
register allocator constrained to 16-bit accessible register set



THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

Feedback-Guided Instruction Selection

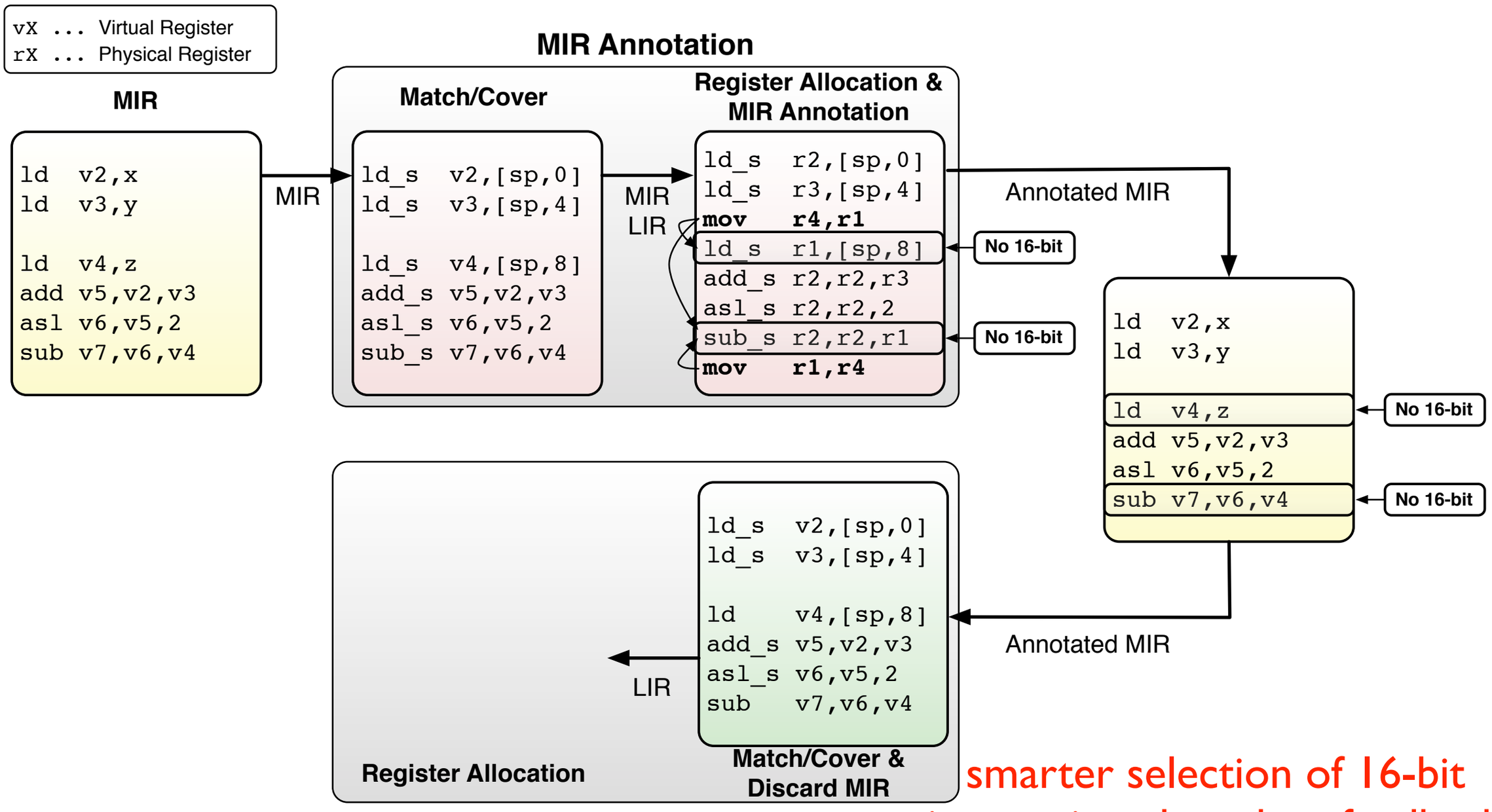


PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

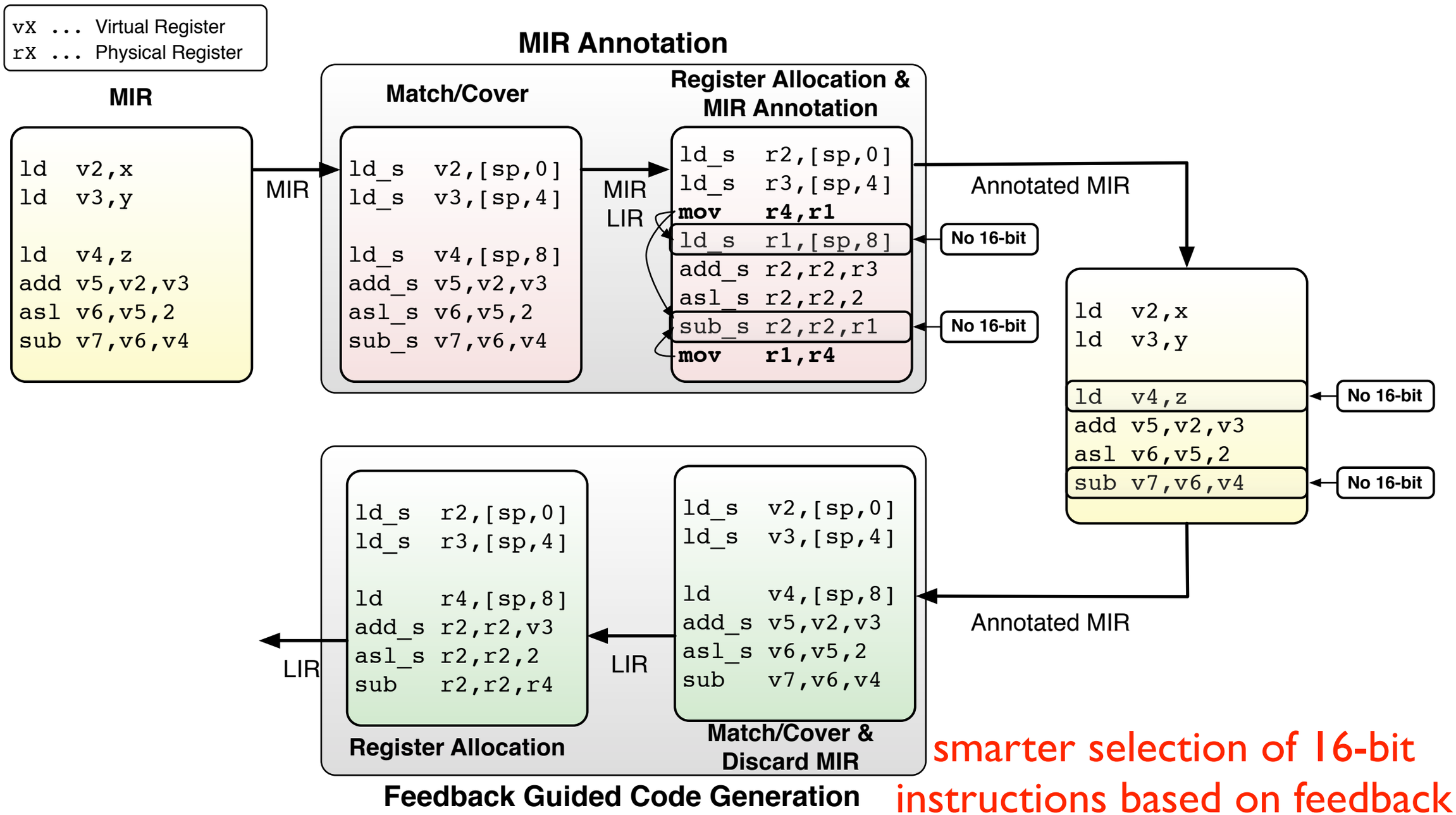
Feedback-Guided Instruction Selection



smarter selection of 16-bit instructions based on feedback

PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

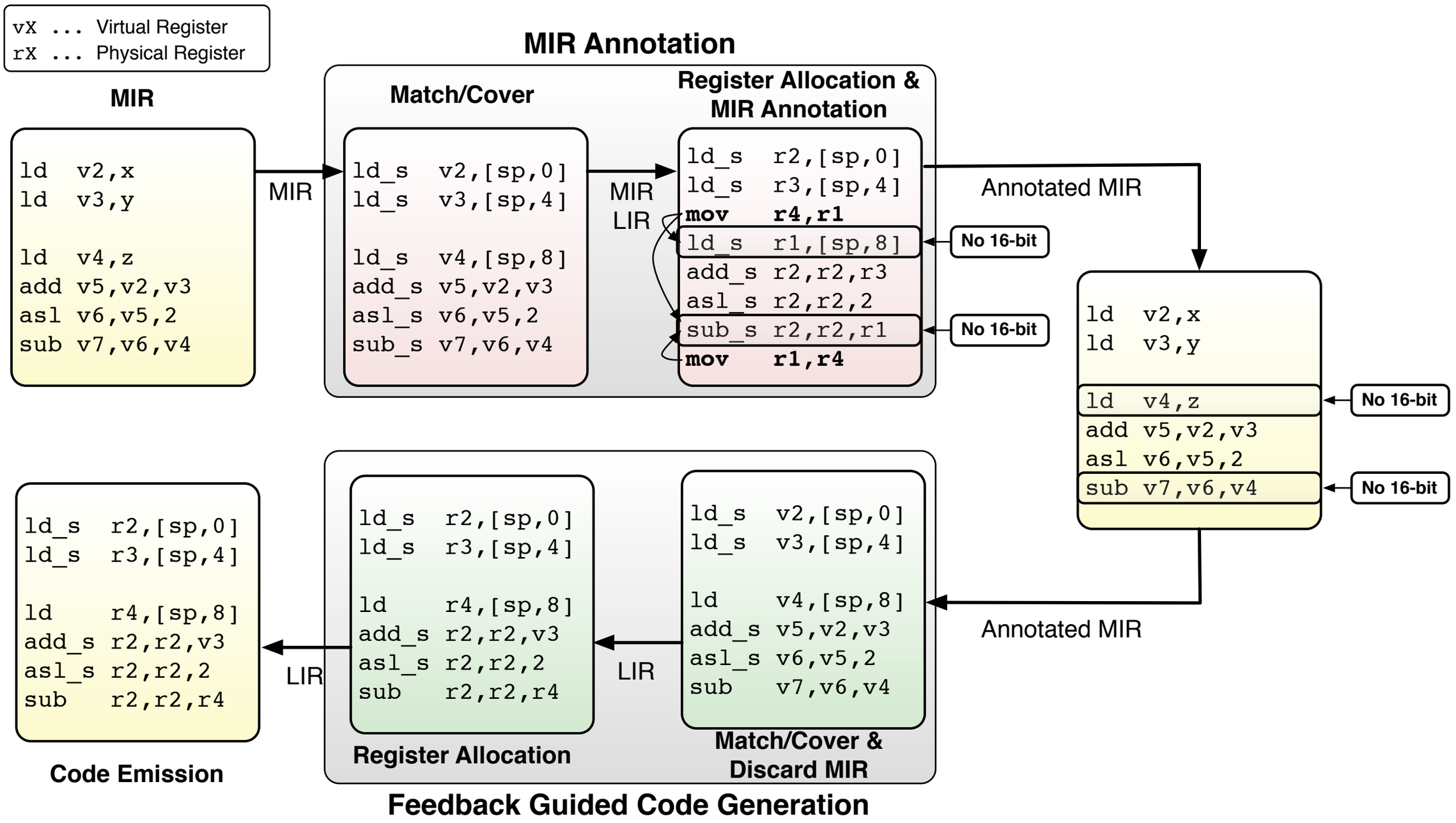
Feedback-Guided Instruction Selection



fewer constraints for register allocator

smarter selection of 16-bit instructions based on feedback

Feedback-Guided Instruction Selection



fewer constraints for register allocator

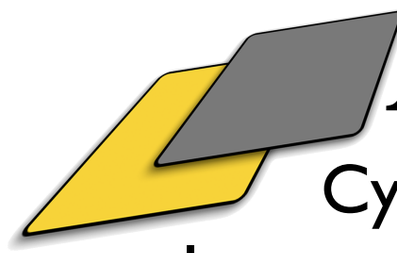
THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

Evaluation - Experimental Setup

BenchMarks	EEMBC 1.1

Core	ARC750D
Pipeline	7-stage interlocked
Execution Order	In-Order
Branch Prediction	Yes
ISA	ARCompact
Floating Point	Hardware
Memory Subsystem	
L1 Cache	Yes
Instruction	8k/2-way associative
Data	8k/2-way associative
L2 Cache	No



ArCSiM
Cycle Accurate
Instruction Set Simulator

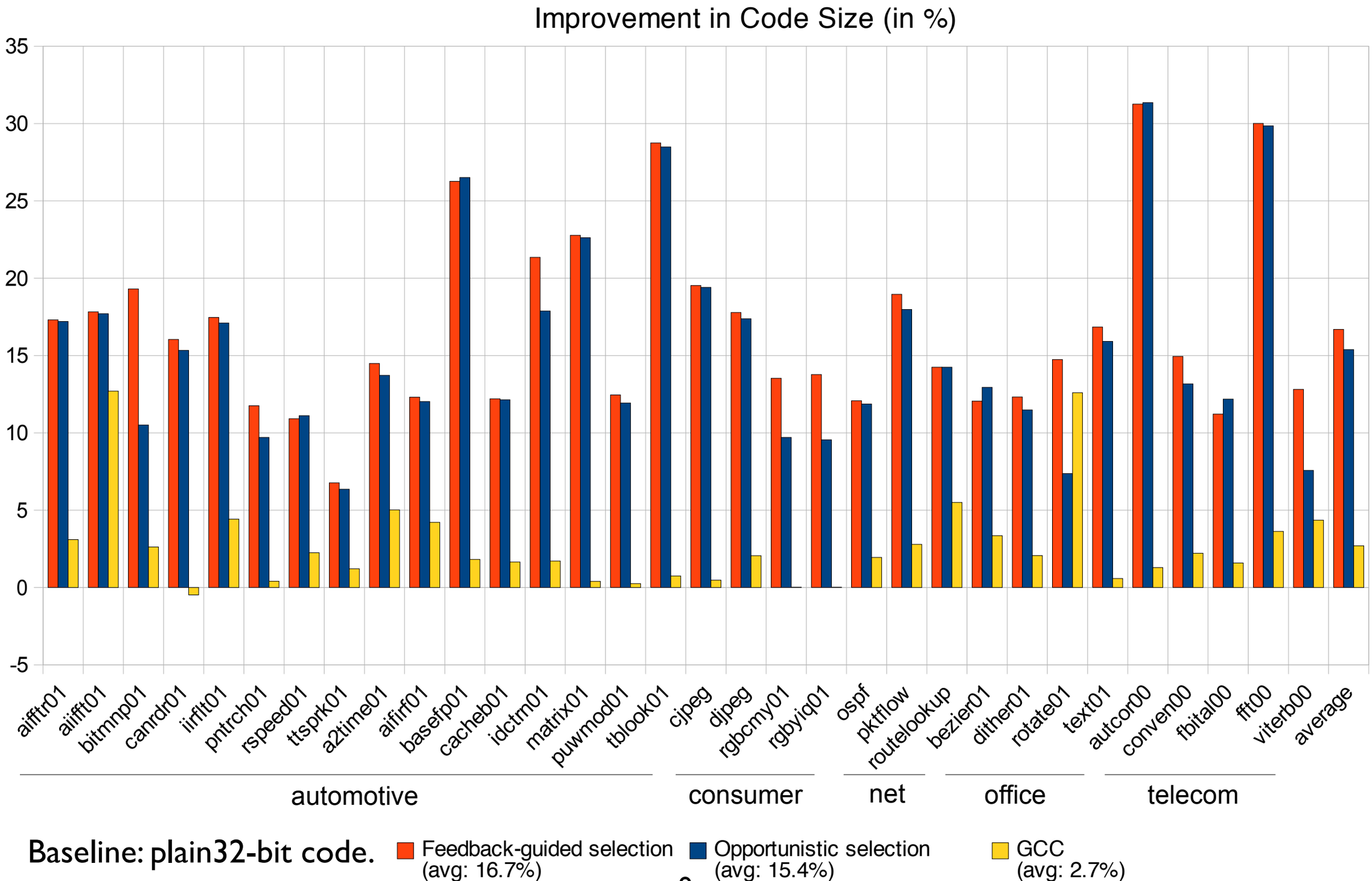
Simulator	ArcSim
Simulation Mode	Full System, Cycle Accurate
Accuracy	Cycle accurate mode validated against real HW
Options	Default
I/O & System Calls	Emulated

PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

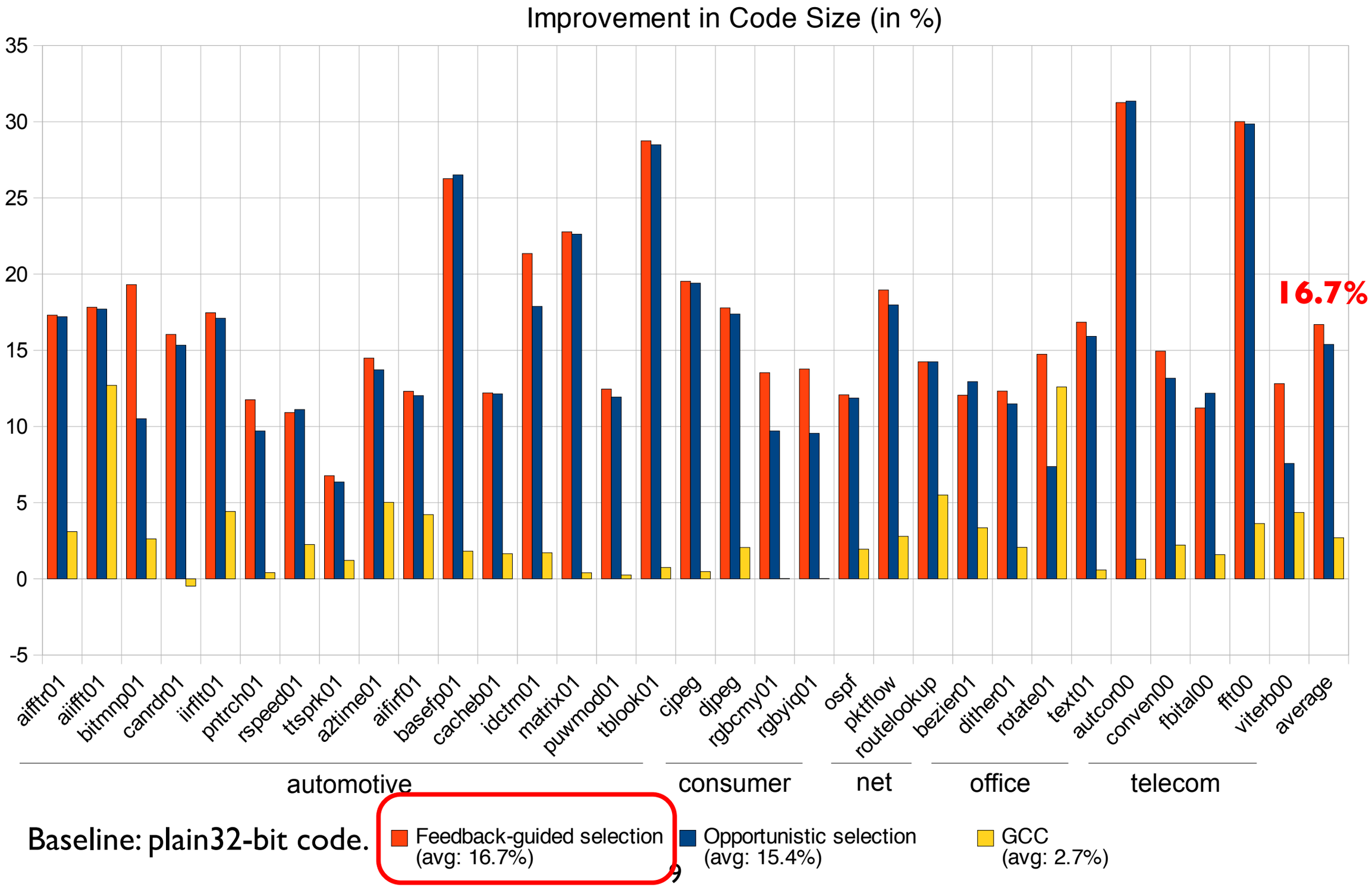
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

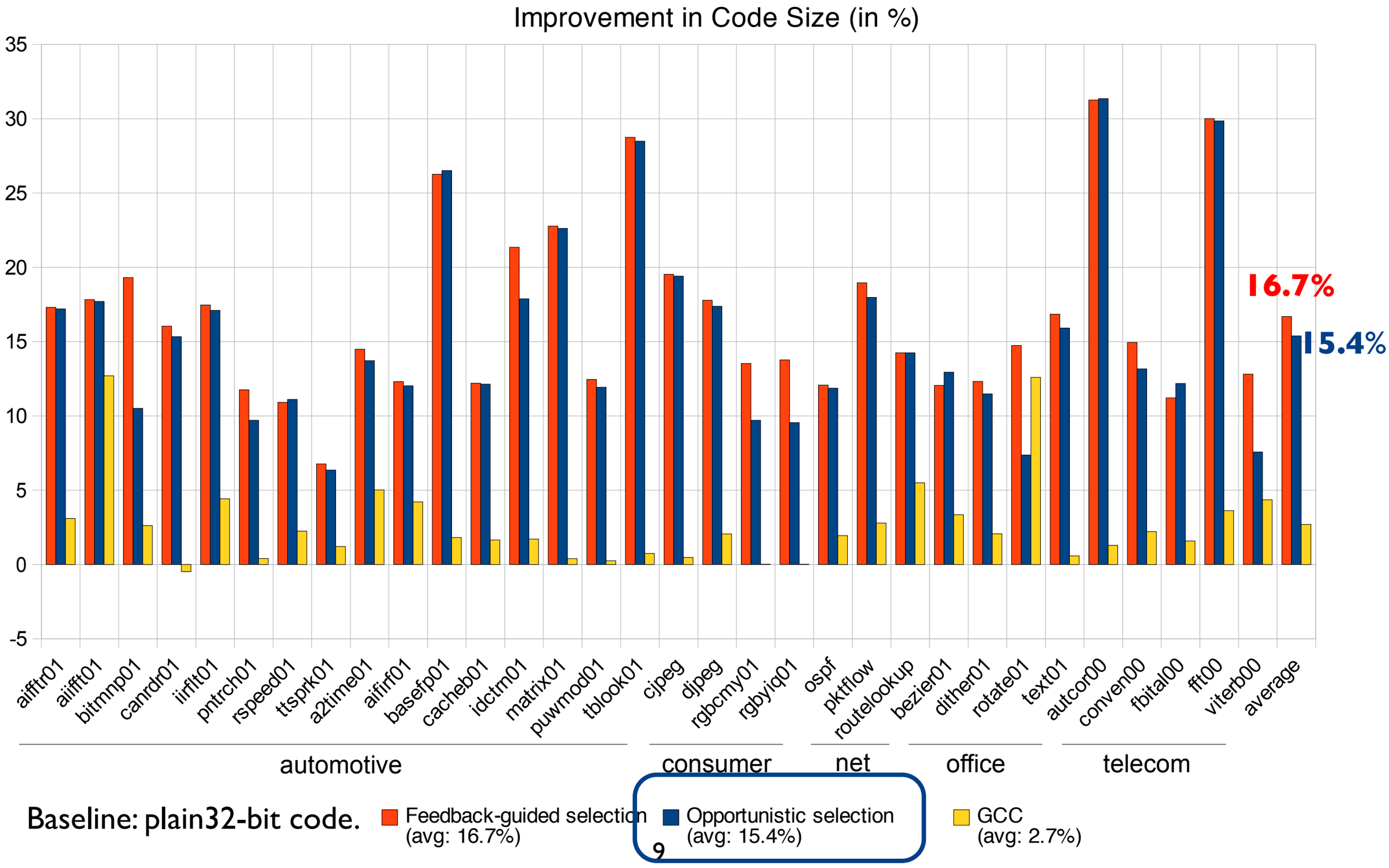
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

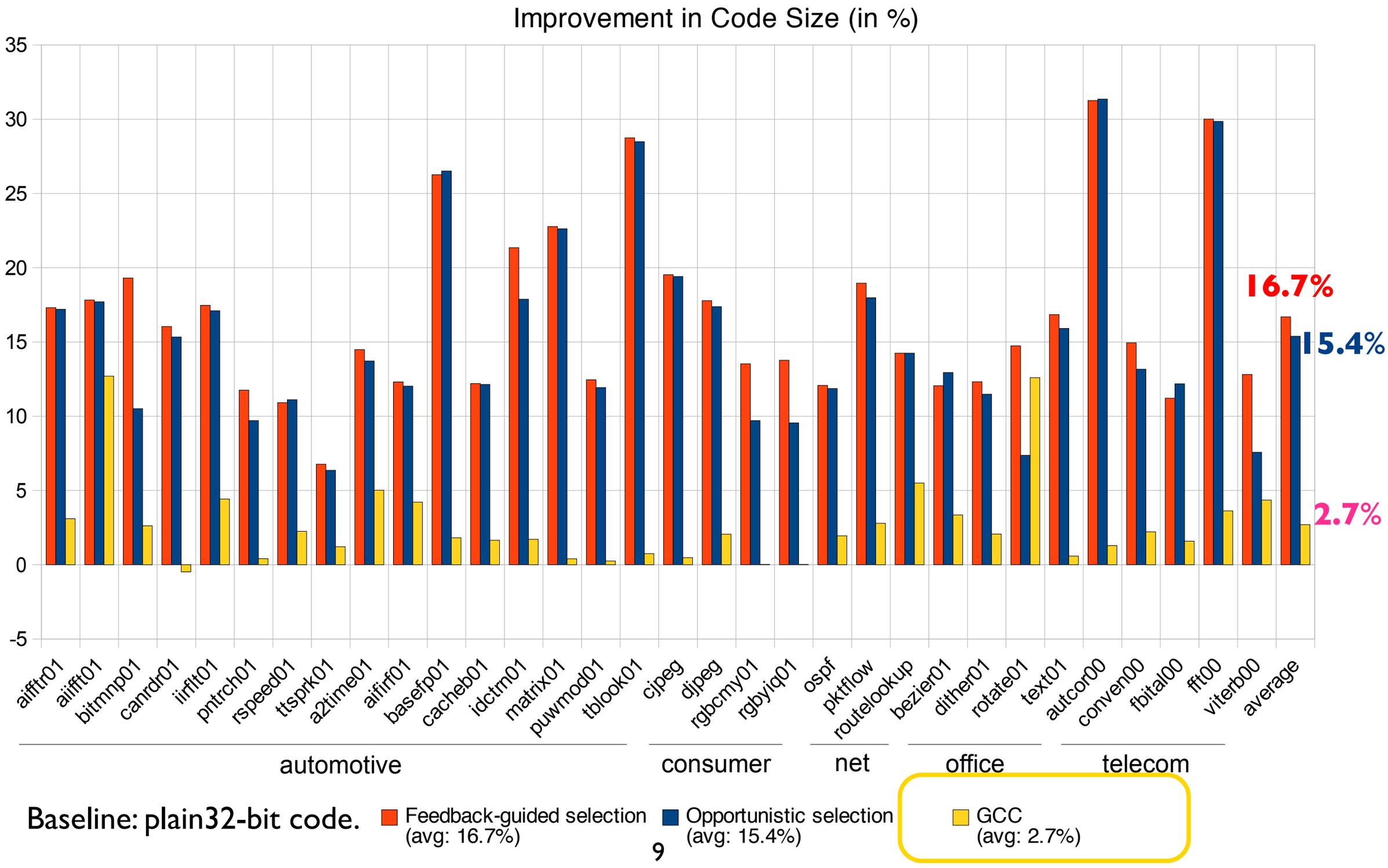
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

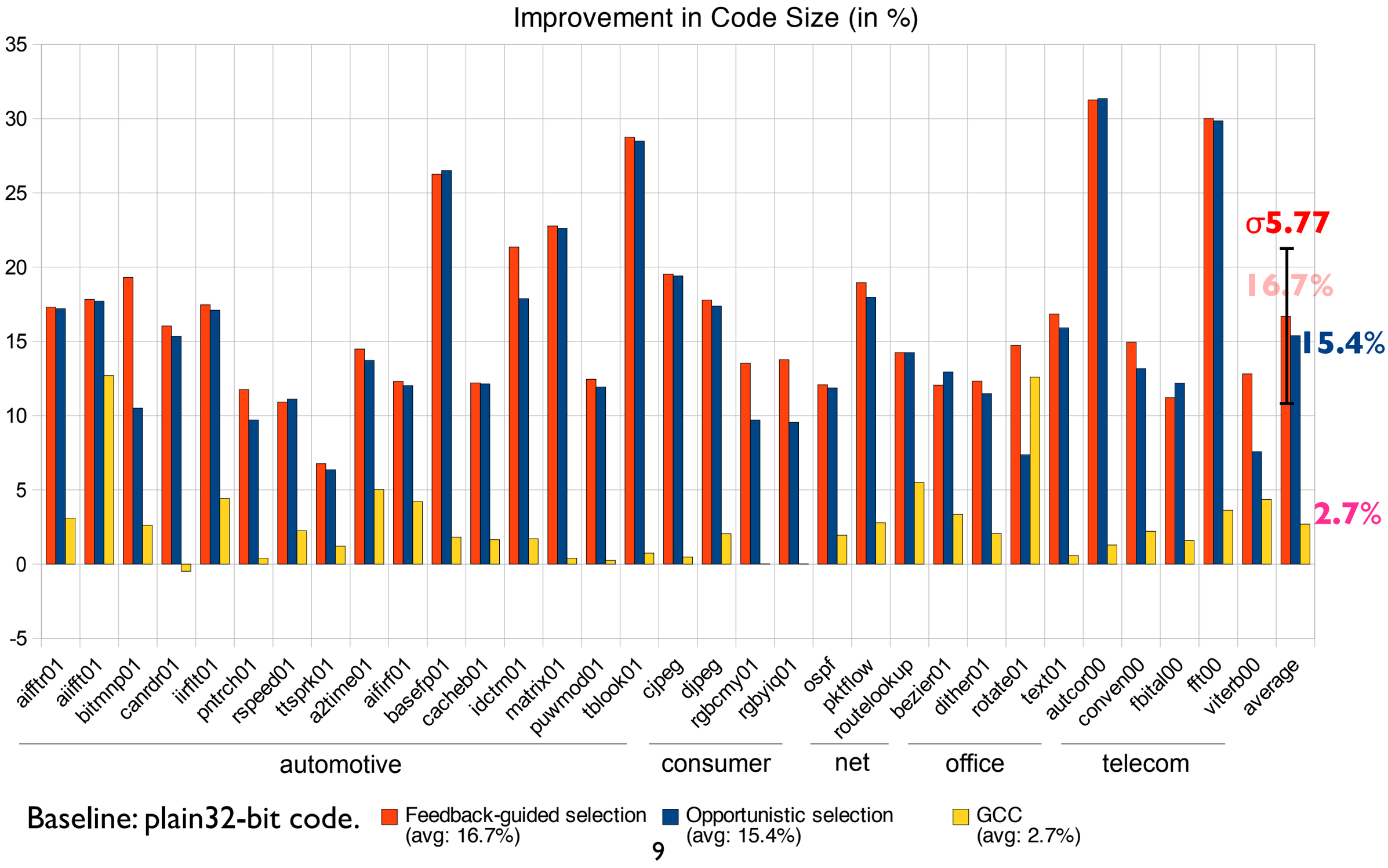
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

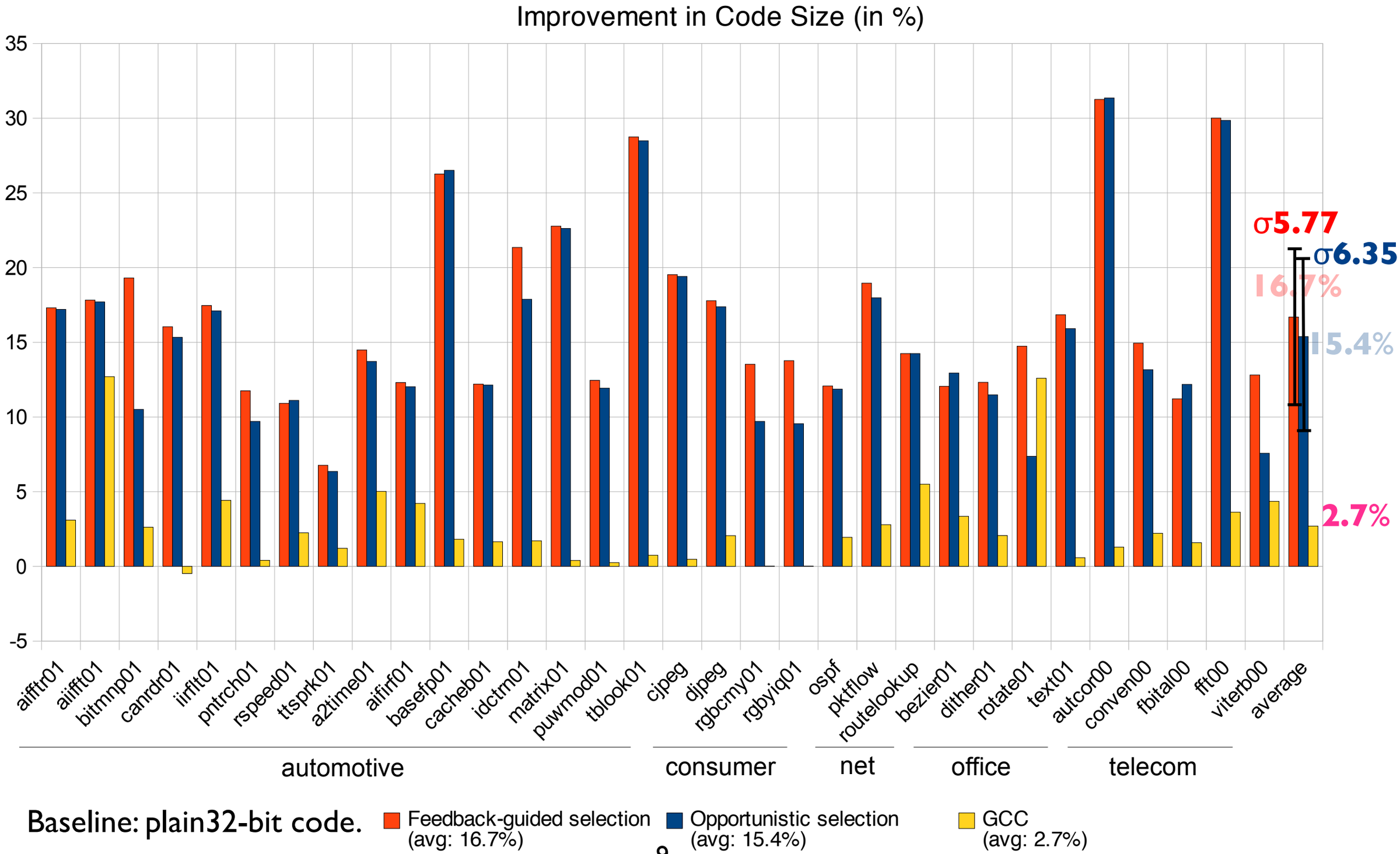
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

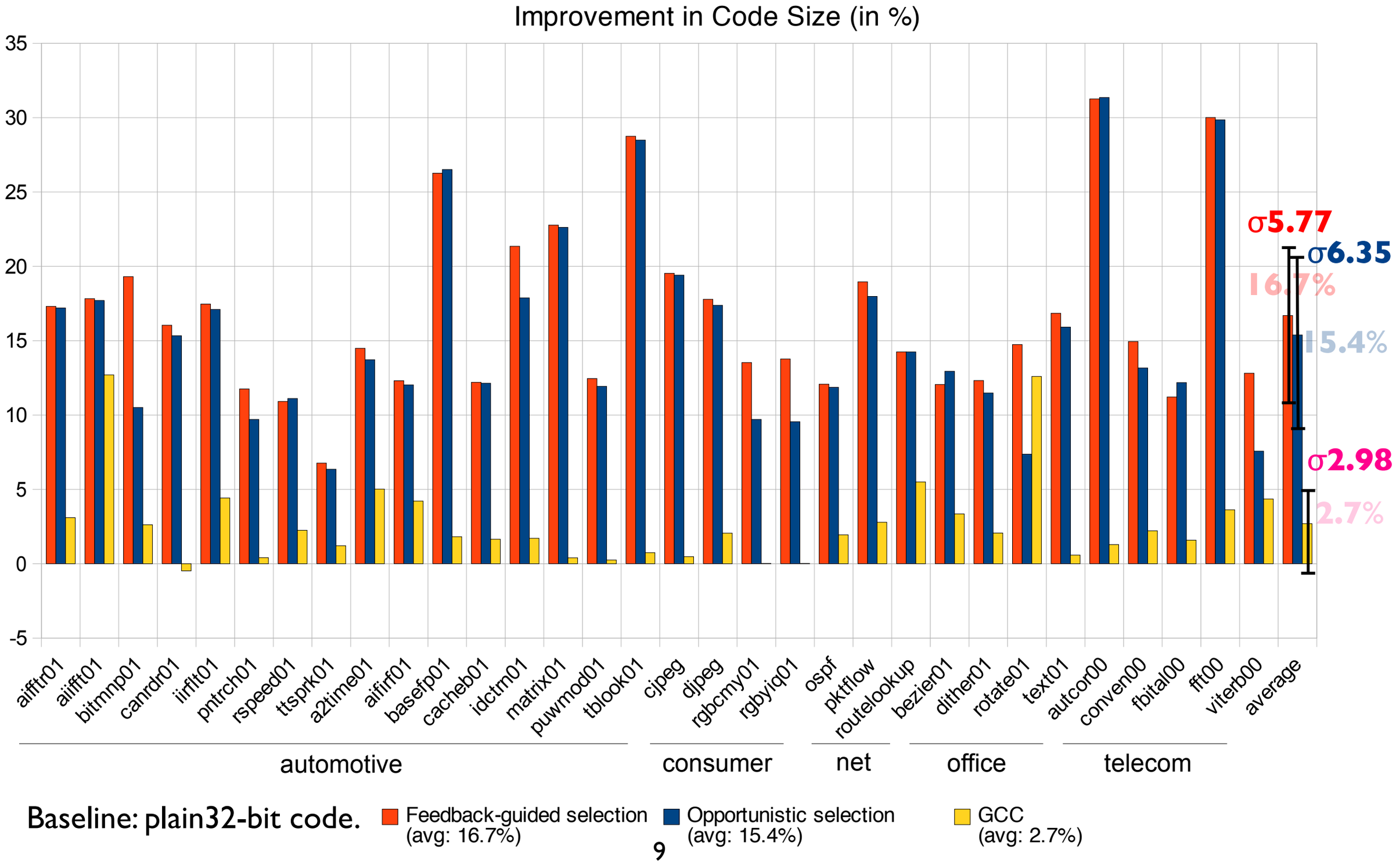
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

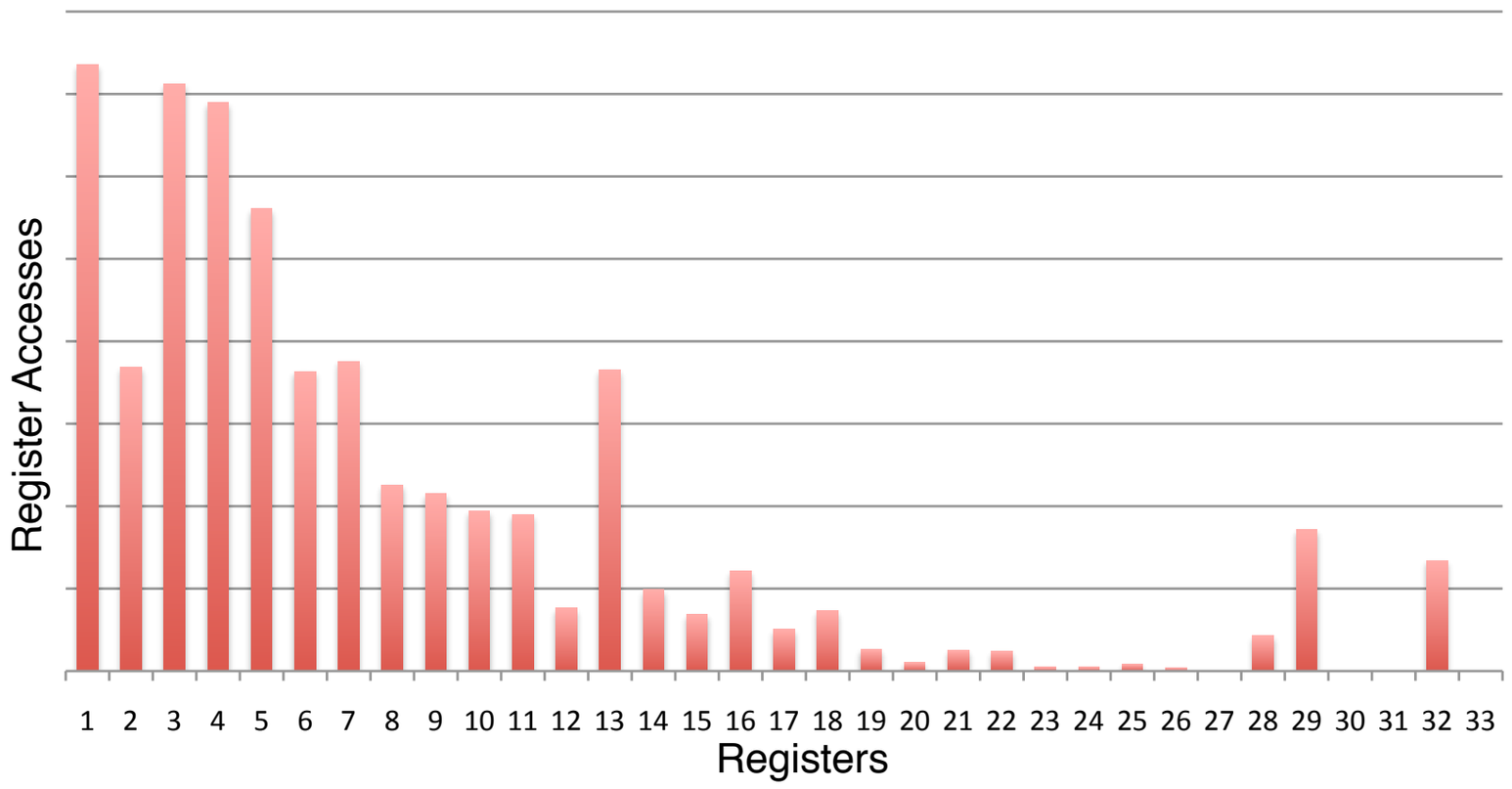
Evaluation - **Code Size** Reduction



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

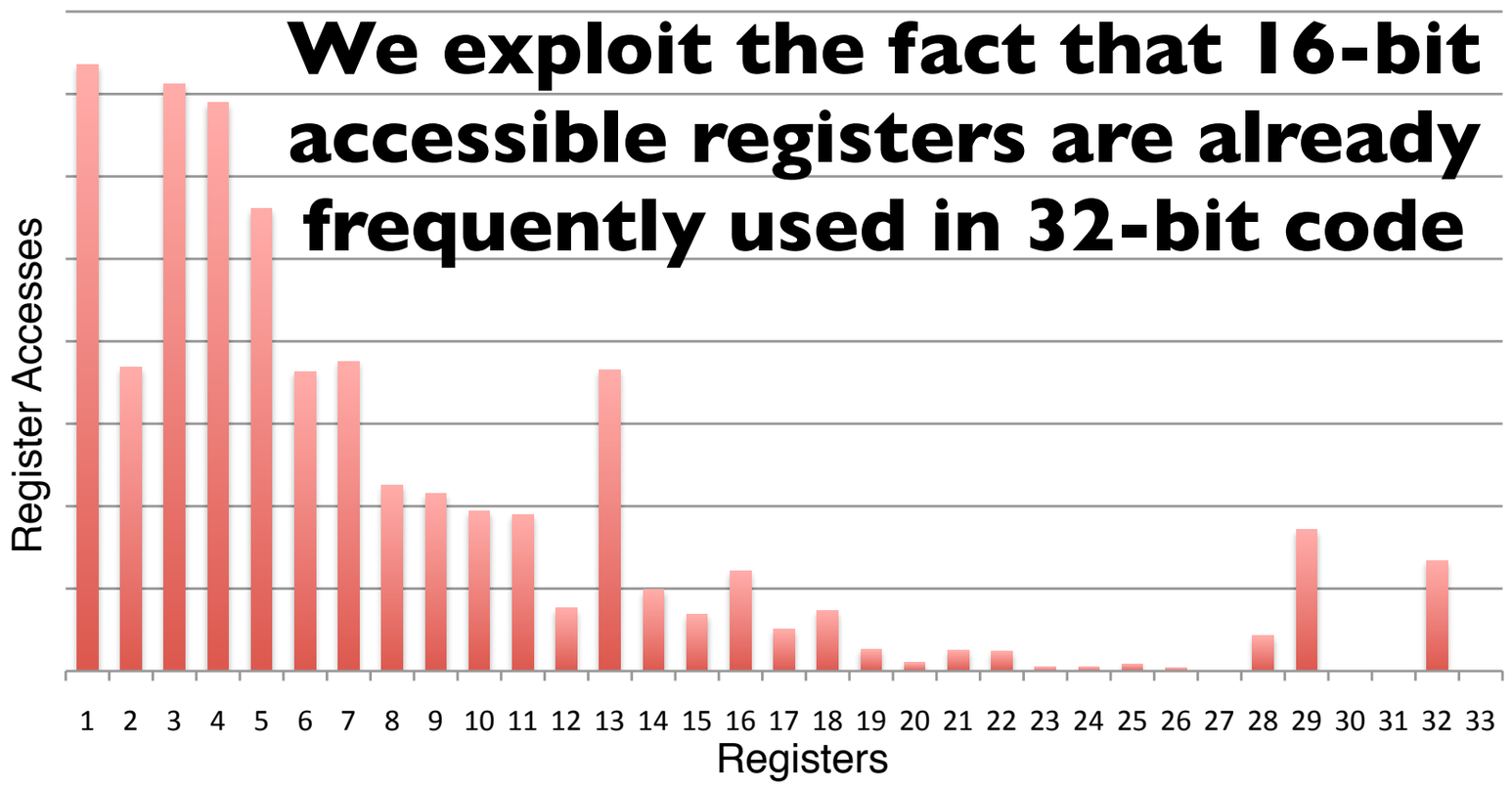
Why does the simple Opportunistic Mode perform so well?



THE UNIVERSITY of EDINBURGH
informatics

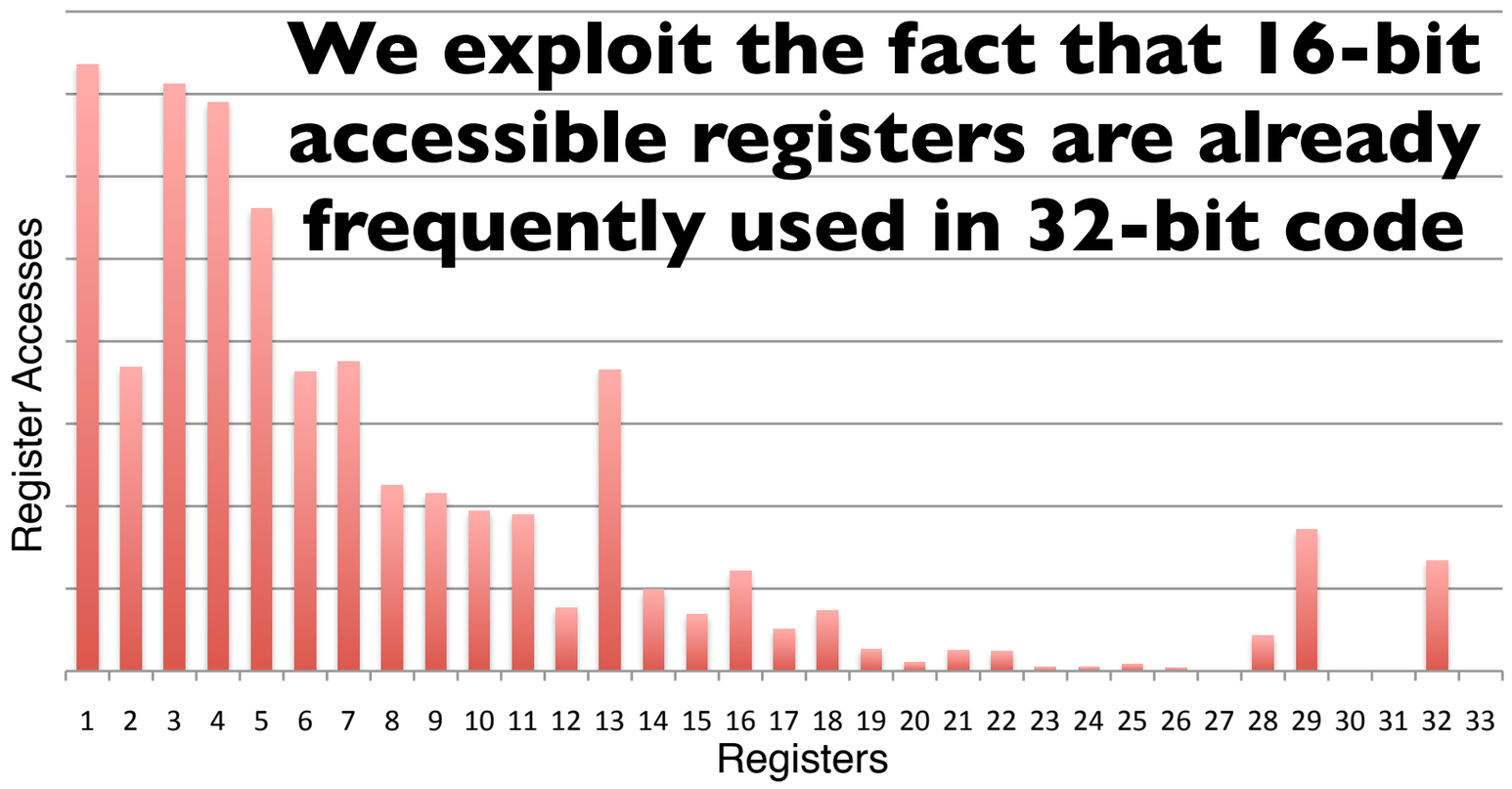
icsa | Institute for Computing Systems Architecture

Why does the simple Opportunistic Mode perform so well?



Why does the simple Opportunistic Mode perform so well?

- register allocator selects registers with lower ID from set of possible registers

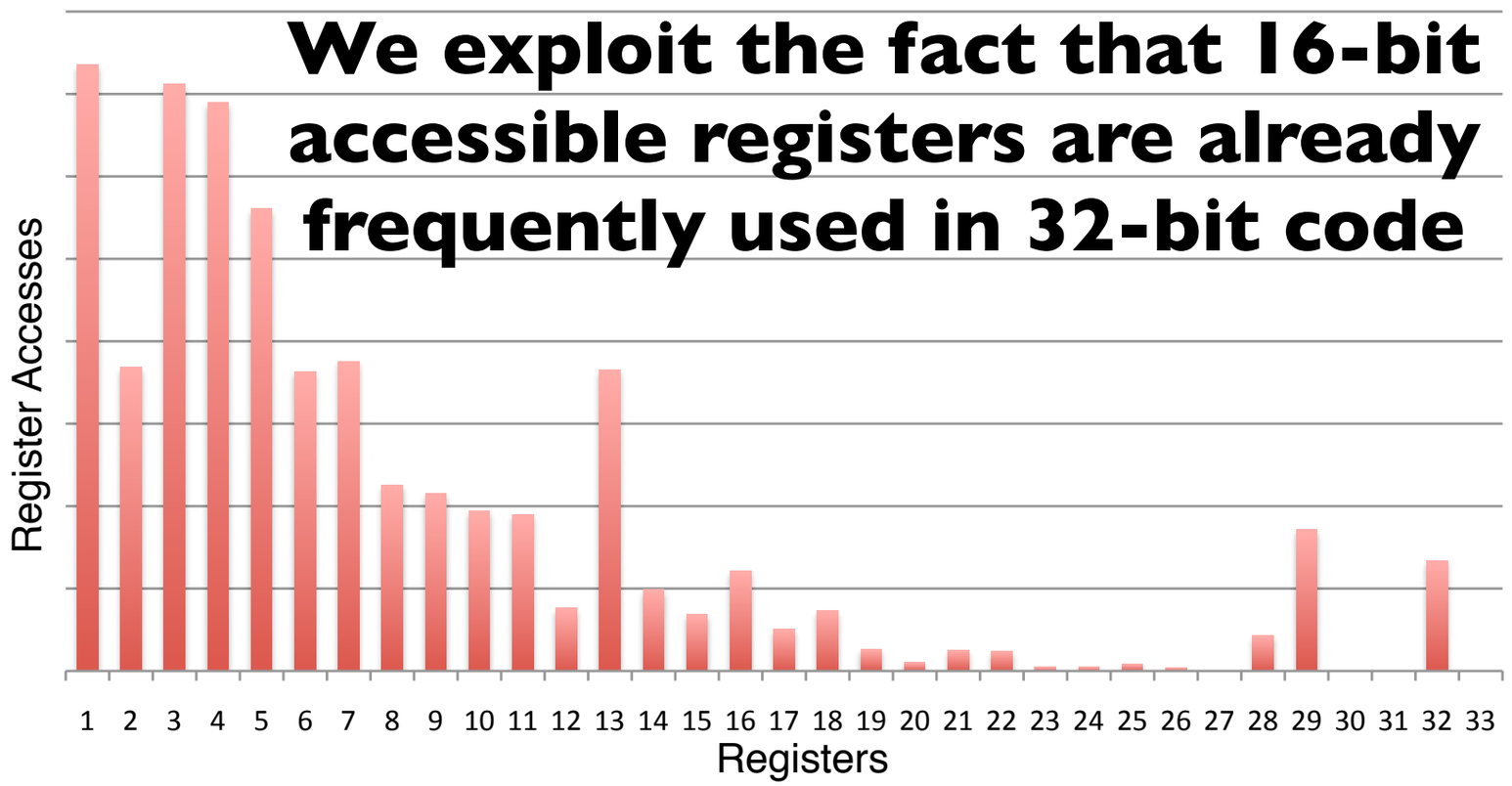


THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

Why does the simple Opportunistic Mode perform so well?

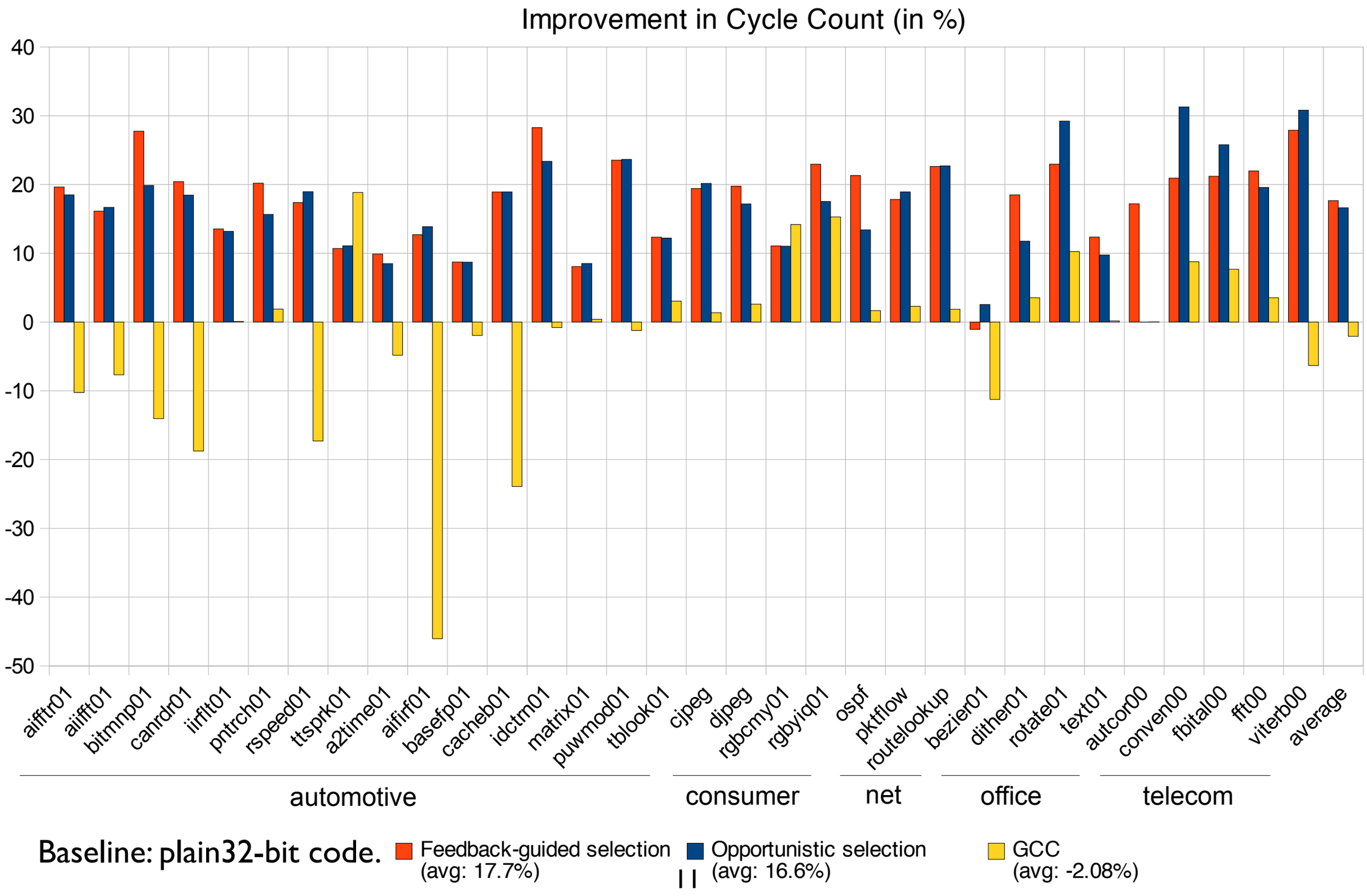
- register allocator selects registers with lower ID from set of possible registers
- calling conventions constrain register allocator



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

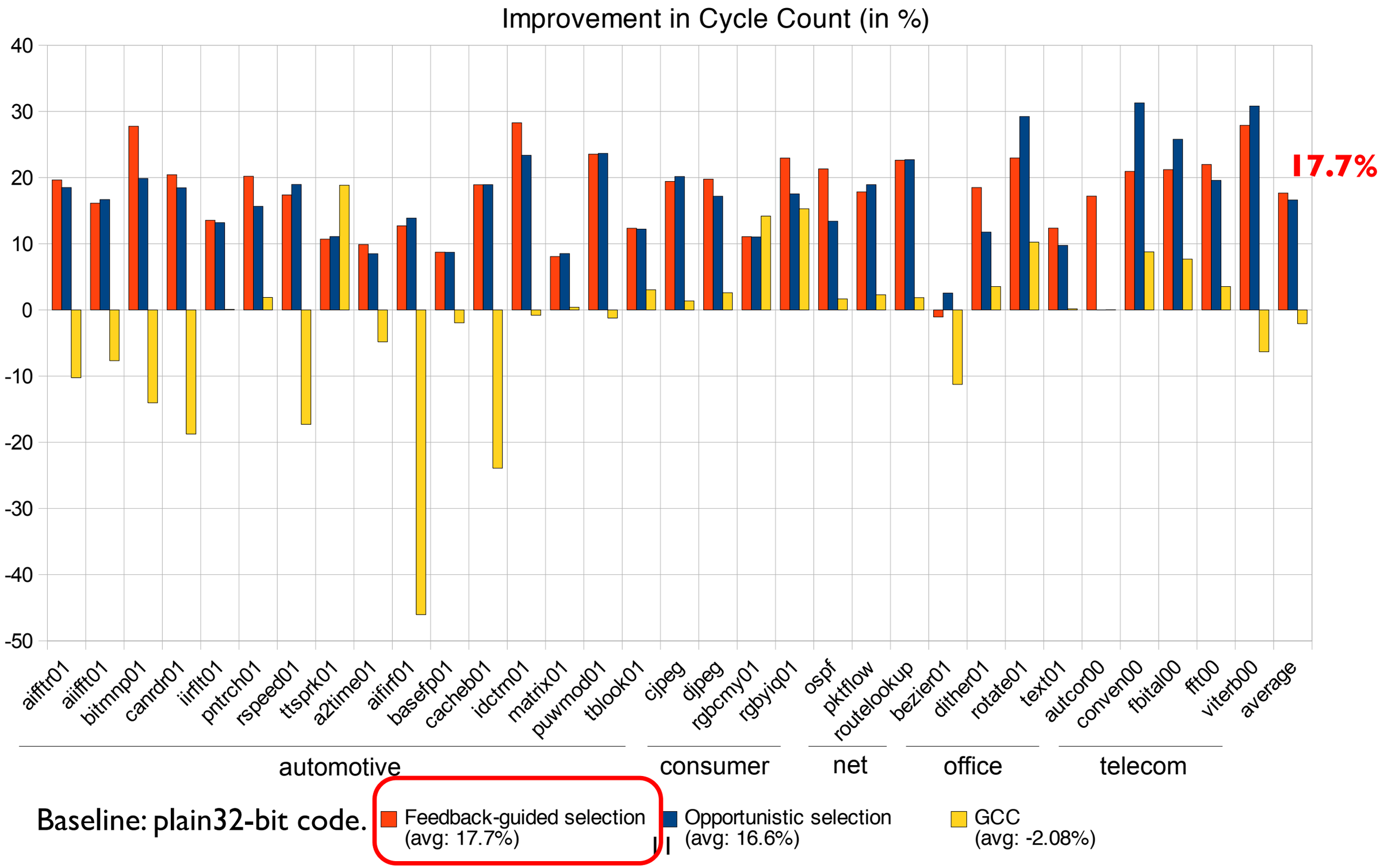
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

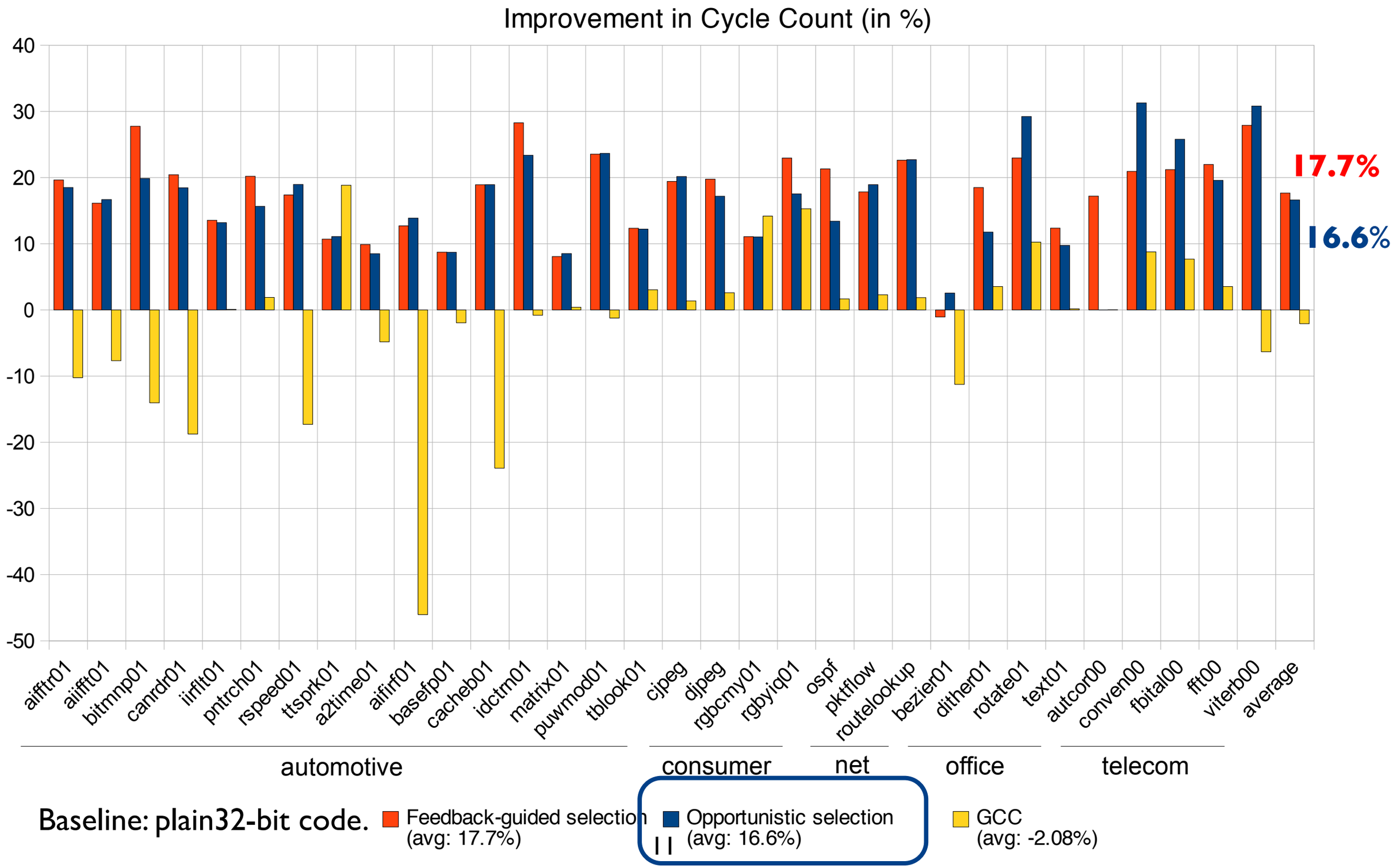
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

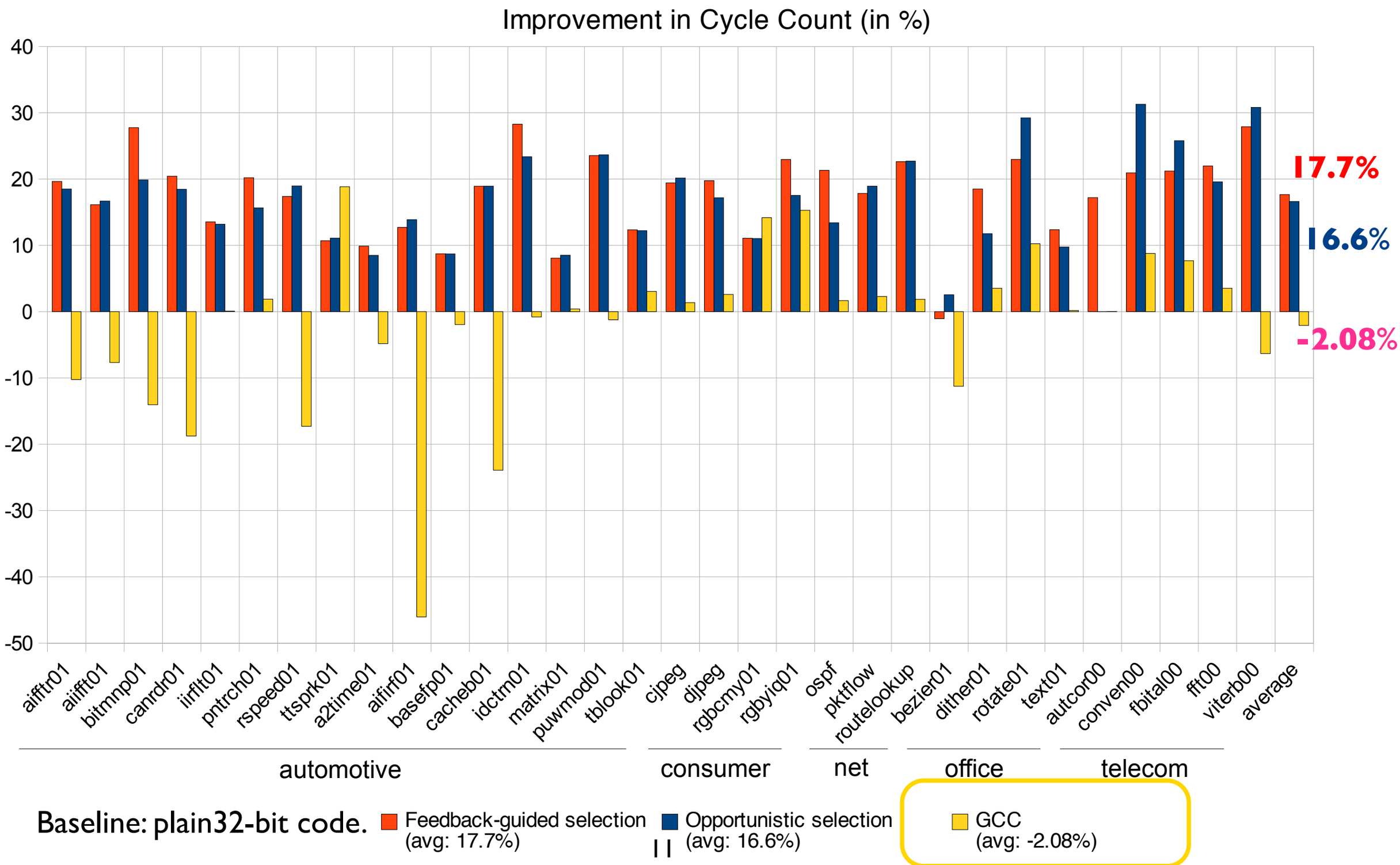
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

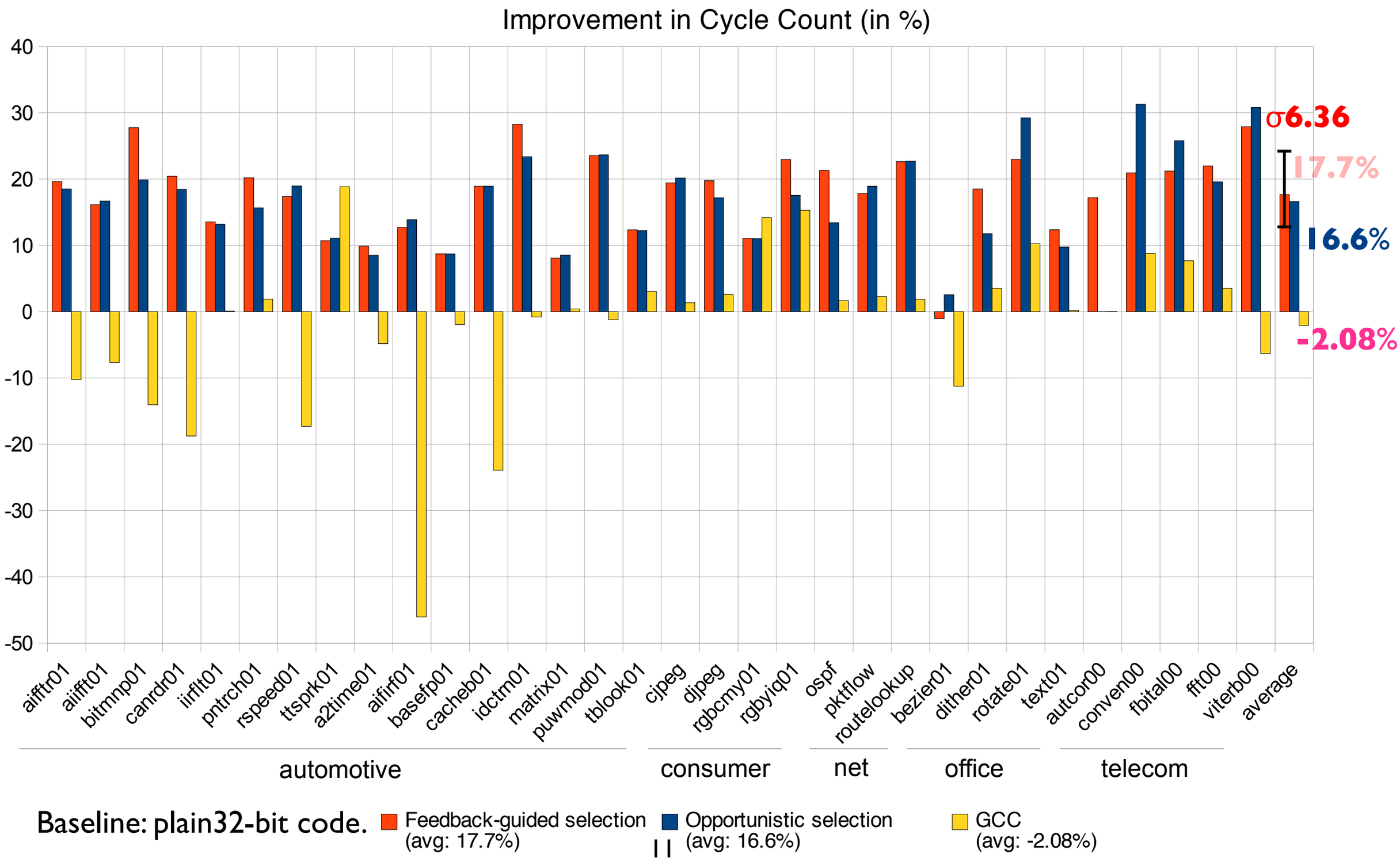
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

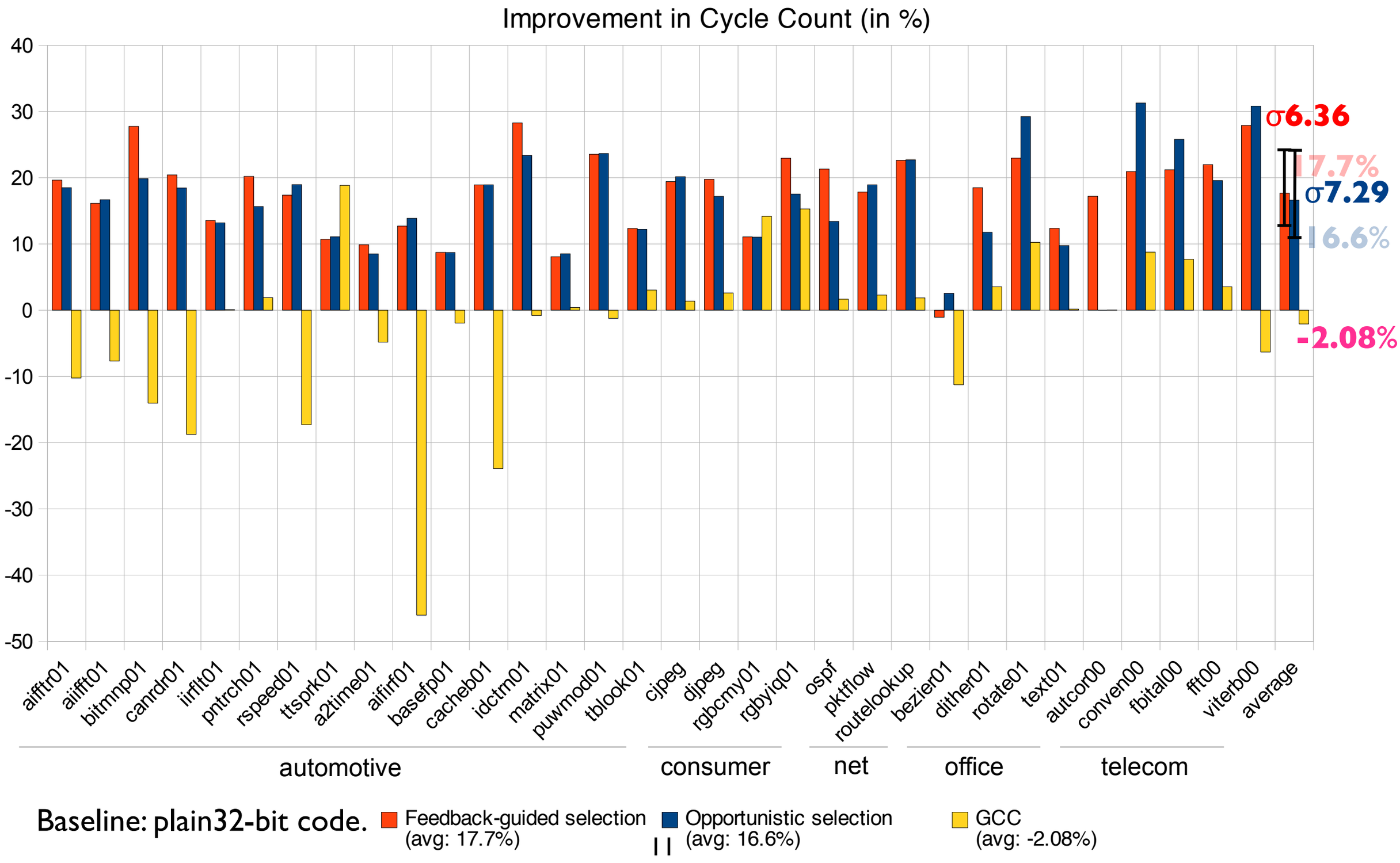
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

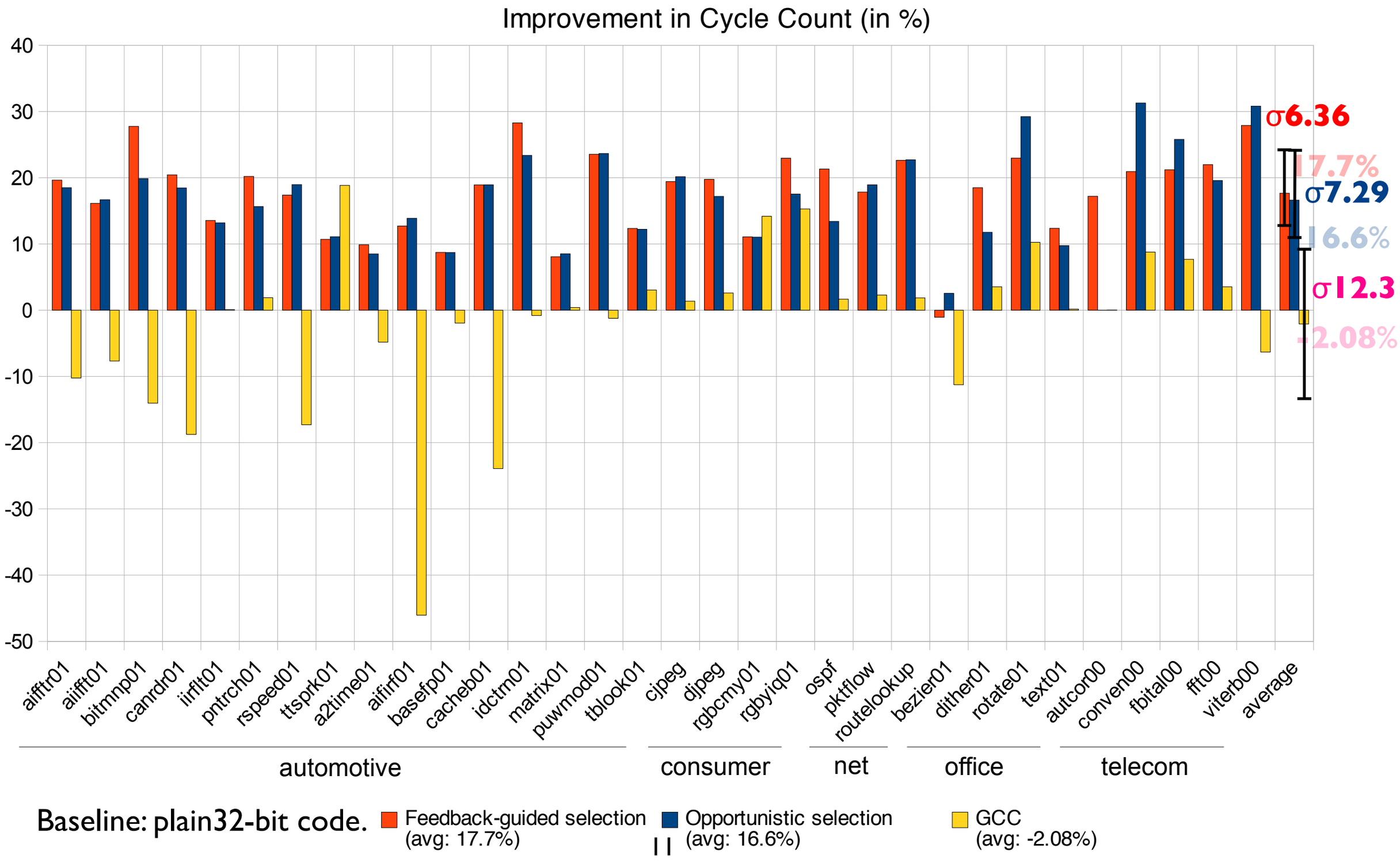
Evaluation - **Performance** Improvements



THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

Evaluation - **Performance** Improvements



Conclusions

- Compact Code Generation is an *integrated* Instruction Selection and Register Allocation problem.

Conclusions

- Compact Code Generation is an *integrated* Instruction Selection and Register Allocation problem.
- While our simple *opportunistic* mode works well, our *feedback-directed* mode produces more consistent results and does not rely on calling conventions or register-allocation implementations.

Conclusions

- Compact Code Generation is an *integrated* Instruction Selection and Register Allocation problem.
- While our simple *opportunistic* mode works well, our *feedback-directed* mode produces more consistent results and does not rely on calling conventions or register-allocation implementations.
- Our scheme is the first one demonstrating that small code size can be achieved whilst improving performance.

THE UNIVERSITY of EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture

Thanks!

For more information visit our website or *search for* the term '*PASTA project*'.

http://groups.inf.ed.ac.uk/pasta/

EPSRC Engineering and Physical Sciences Research Council

PASTA Processor Automated Synthesis by iTerative Analysis
http://groups.inf.ed.ac.uk/pasta/