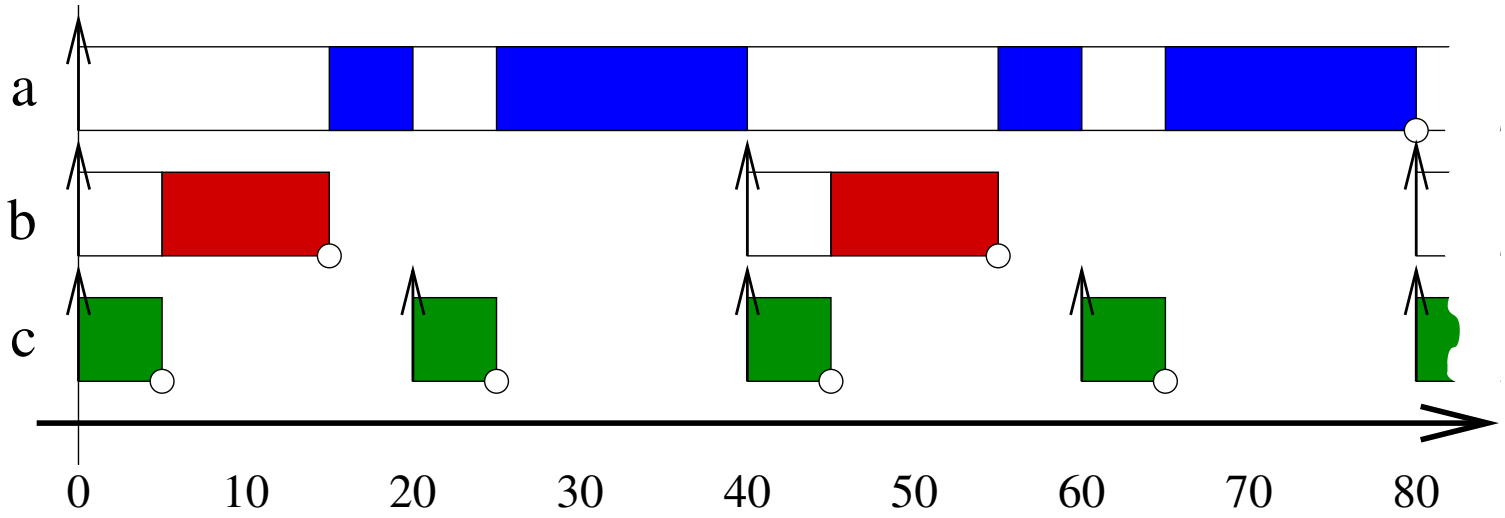
CPUs For WCET Reduction

Jack Whitham

18th January 2008

What is WCET?

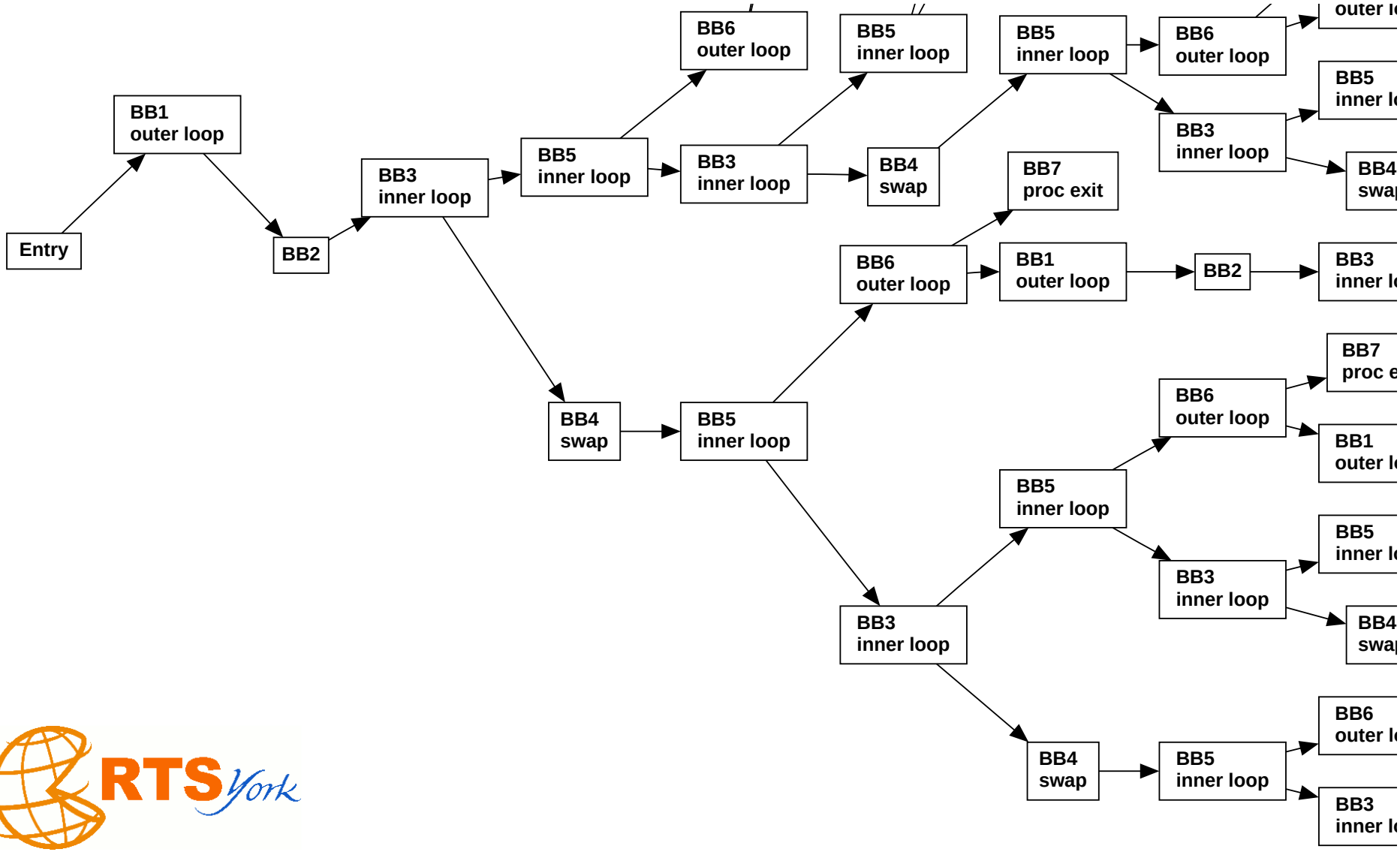


Task	a	b	c
Period, T	80	40	20
Deadline, D	80	40	20
Priority, P	1	2	3
Computation time, C	40	10	5

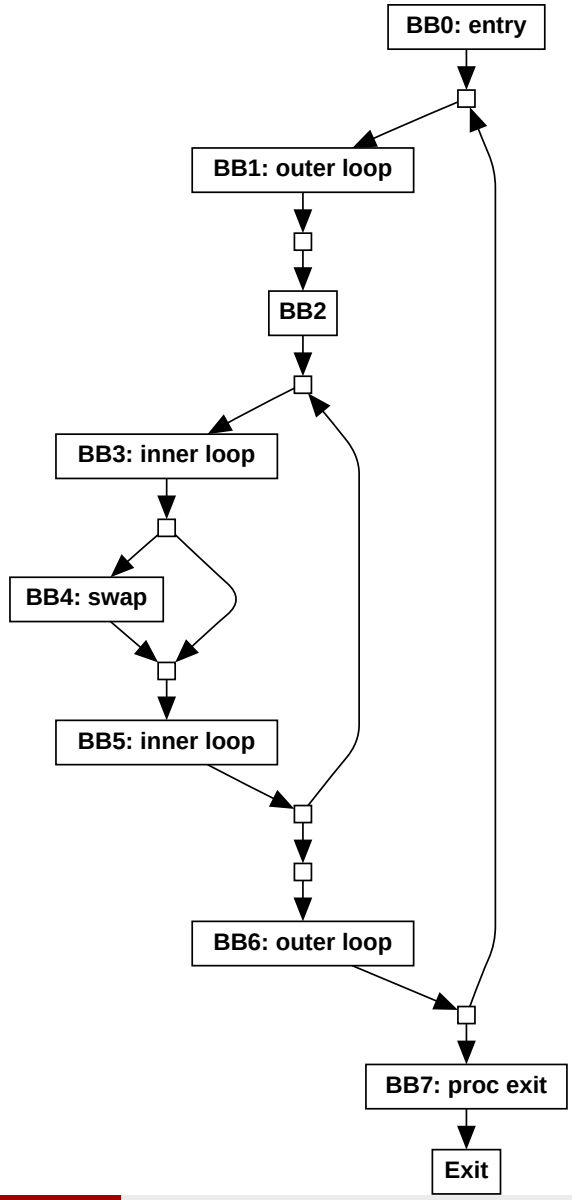
How do we obtain C?

```
void Task ( void ) {
    int     i , swapped ;
    do {
        swapped = 0 ;
        for ( i = 0 ; i < ( SORT_SIZE - 1 ) ; i ++ ) {
            int     si0 = to_sort [ i ] ;
            int     si1 = to_sort [ i + 1 ] ;
            if ( si0 > si1 ) { /* swap */
                to_sort [ i ] = si1 ;
                to_sort [ i + 1 ] = si0 ;
                swapped = 1 ;
            }
        }
    } while ( swapped ) ;
}
```

Data dependence



Control flow graph



What do we want to know?

- C - maximum computation time

What do we want to know?

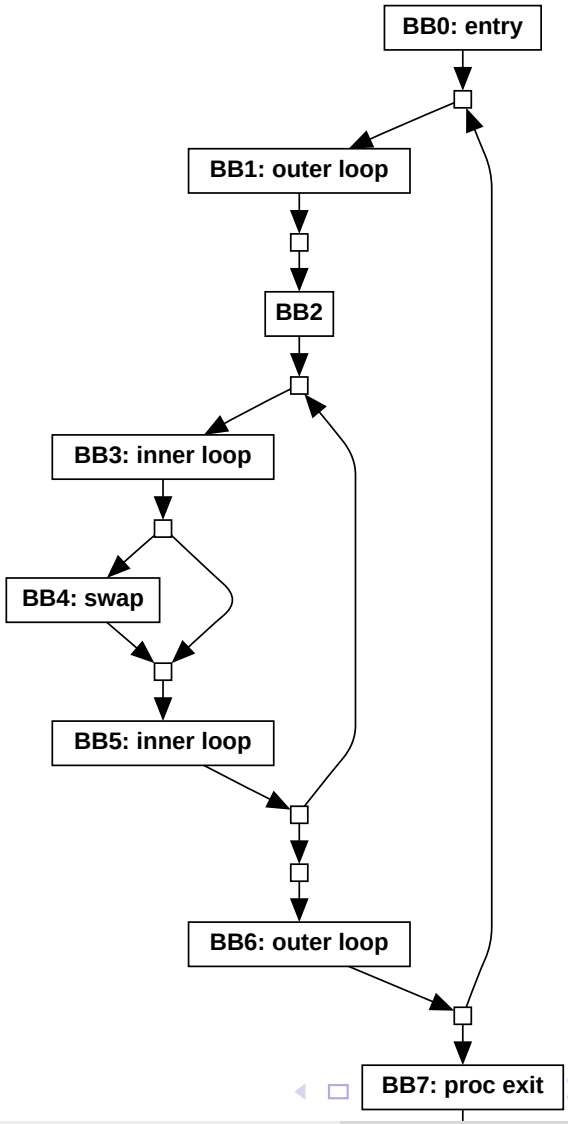
- C - maximum computation time
- C can be expressed as:

$$C = \sum_{x} \gamma(x) f(x)$$

- $\gamma(x) =$ execution cost of bb x
- $f(x) =$ number of times bb x runs during worst case execution path

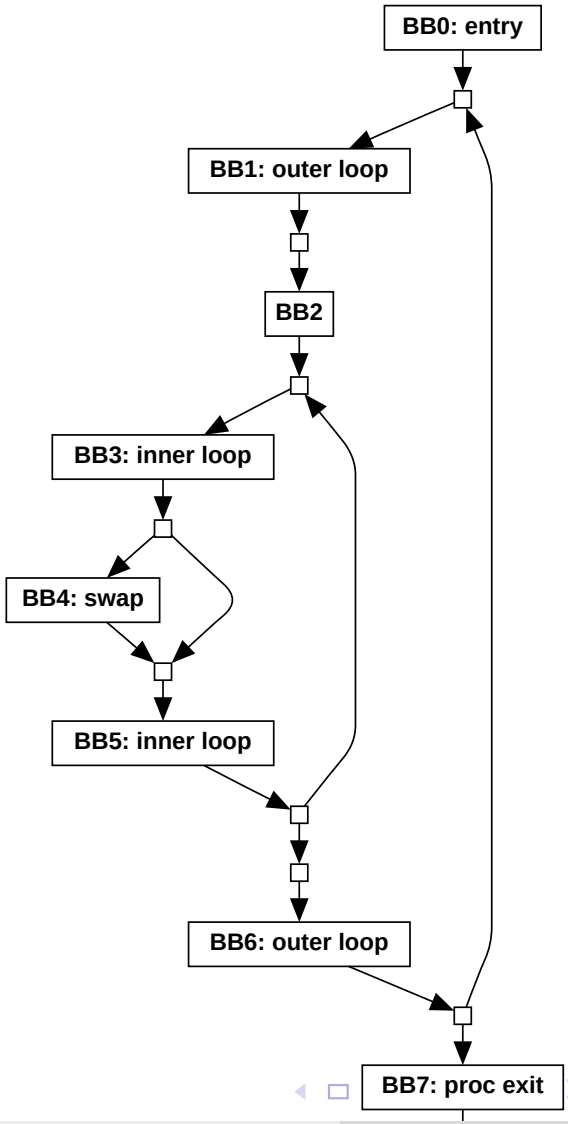
Execution costs

BB x	$\gamma(x)$	$f(x)$
0	14	1
1	6	?
2	6	?
3	18	?
4	10	?
5	6	?
6	6	?
7	12	1



Equations

$$
\begin{aligned}
f(x_0) + f(x_A) &= f(x_1) \\
f(x_1) &= f(x_2) \\
f(x_2) + f(x_B) &= f(x_3) \\
f(x_3) &= f(x_4) + f(x_C) \\
f(x_C) + f(x_4) &= f(x_5) \\
f(x_5) &= f(x_B) + f(x_6) \\
f(x_6) &= f(x_A) + f(x_7) \\
f(x_1) &= f(x_7) = 1 \\
C &= \sum_x \gamma(x) f(x)
\end{aligned}
$$



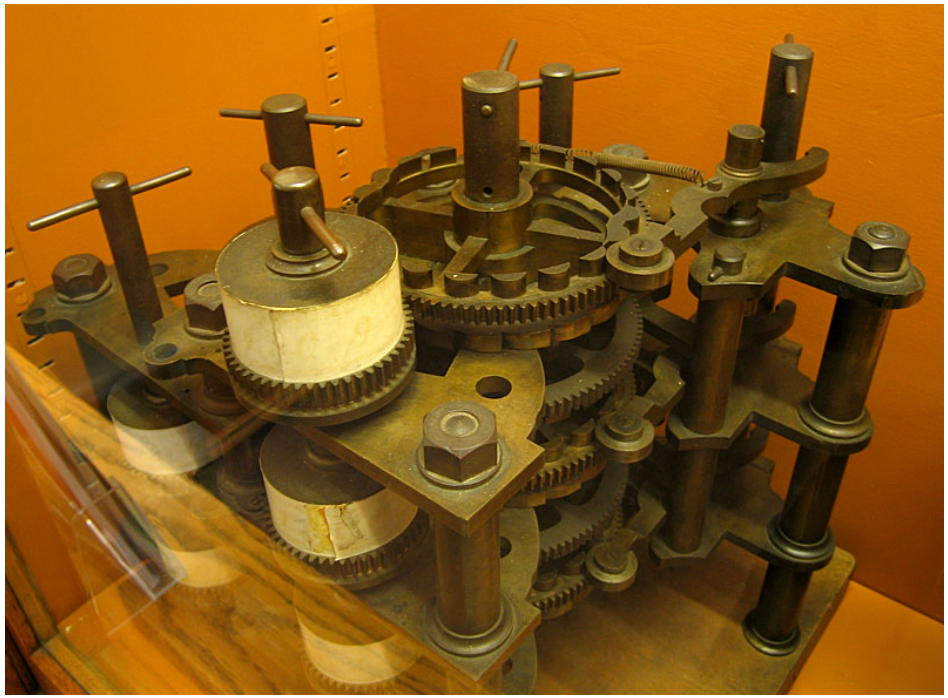
Behavioural constraints

$$
\begin{aligned}
f(x_1) &\leq 100f(x_0) \\
f(x_3) &\leq 99f(x_2) \\
f(x_4) &\leq 50f(x_2) \\
C &= 14f(x_0) + 6f(x_1) + 6f(x_2) + 18f(x_3) + \\
&\quad 10f(x_4) + 6f(x_5) + 6f(x_6) + 10f(x_7)
\end{aligned}
$$

$$
\begin{aligned}
f(x_0) + f(x_A) &= f(x_1) \\
f(x_1) &= f(x_2) \\
f(x_2) + f(x_B) &= f(x_3) \\
f(x_3) &= f(x_4) + f(x_C) \\
f(x_C) + f(x_4) &= f(x_5) \\
f(x_5) &= f(x_B) + f(x_6) \\
f(x_6) &= f(x_A) + f(x_7) \\
f(x_1) &= f(x_7) = 1
\end{aligned}
$$

Solving

So what is C?
We use a linear program solver...
e.g. **glpk**, cplex, lp_solve...



Result

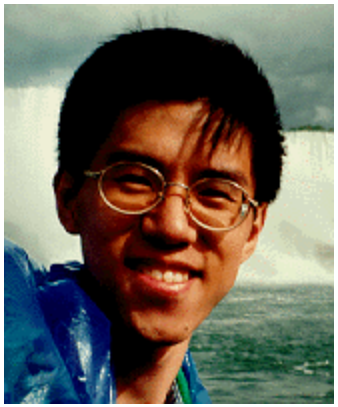
```
Problem:    bubble_eqns
Rows:       15
Columns:    13 (13 integer, 0 binary)
Non-zeros:  40
Status:     INTEGER OPTIMAL
Objective:  obj = 289424 (MAXimum) 289424 (LP)
```

No.	Column name		Activity	Lower bound	Upper bound
1	fx0	*	1	0	
2	fx1	*	100	0	
3	fx2	*	100	0	
4	fx3	*	9900	0	
5	fx4	*	5000	0	
6	fx5	*	9900	0	
7	fx6	*	100	0	
8	fx7	*	1	0	
9	fxA	*	99	0	
10	fxB	*	9800	0	
11	fxC	*	4900	0	
12	fxD	*	100	0	
13	C	*	289424	0	

$$C = 289424$$

Implicit Path Enumeration Technique

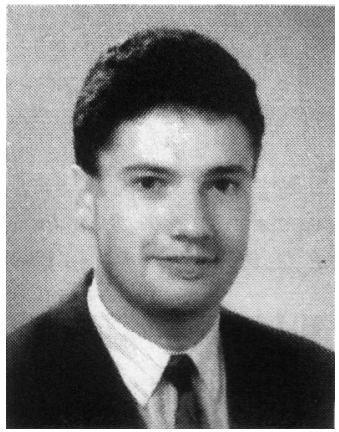
- **IPET** term coined by Li and Malik, 1995.
- My work is based on a paper by Puschner and Schedl, 1997.



Li

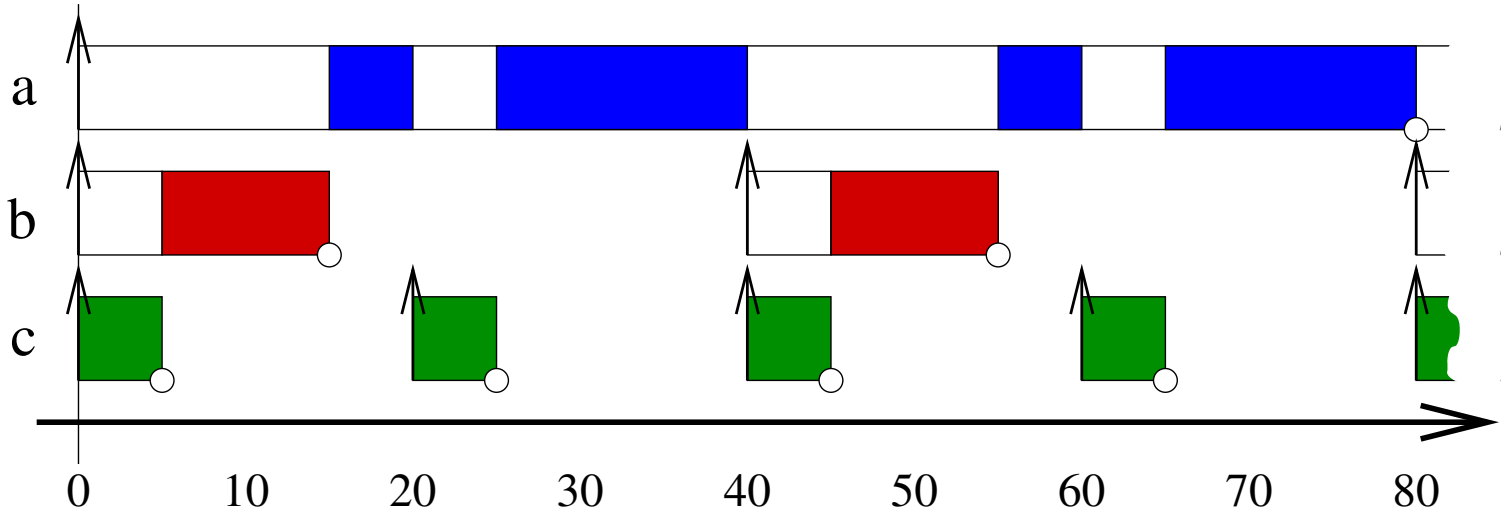
Malik

Puschner



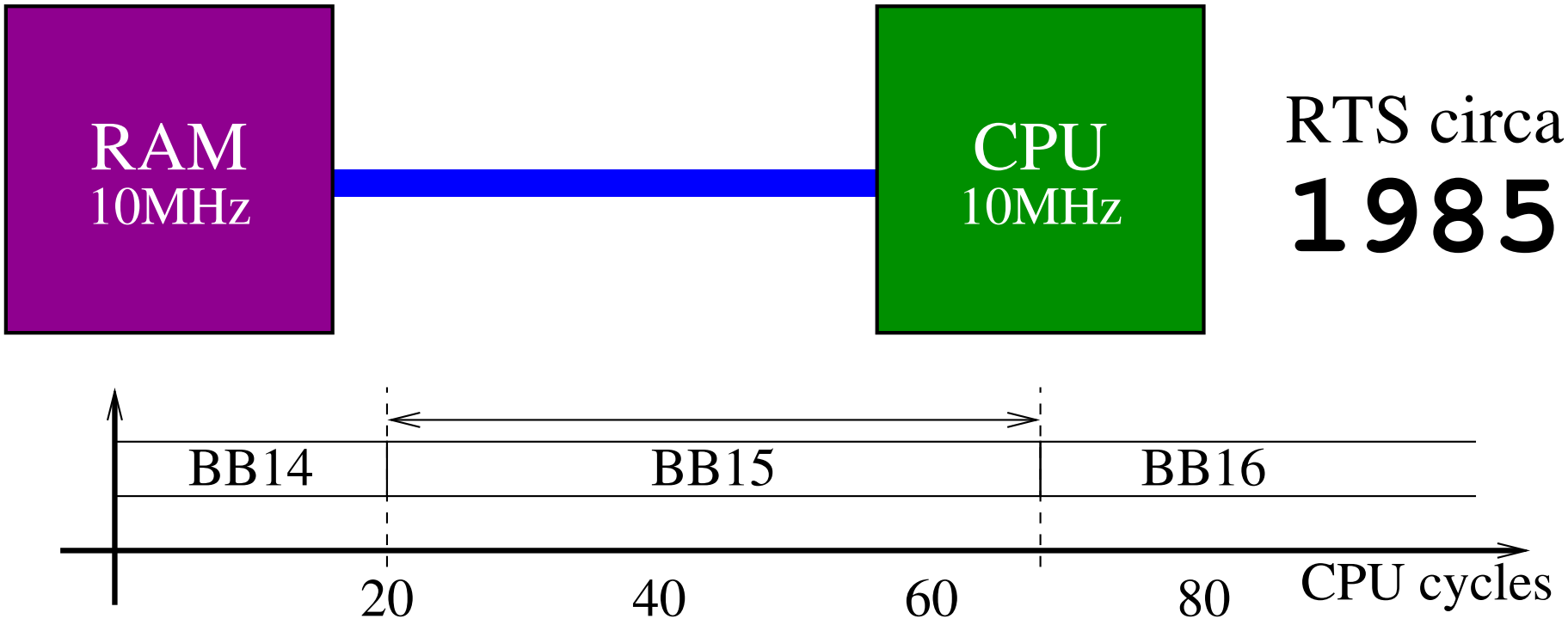
Schedl

Problem Solved?

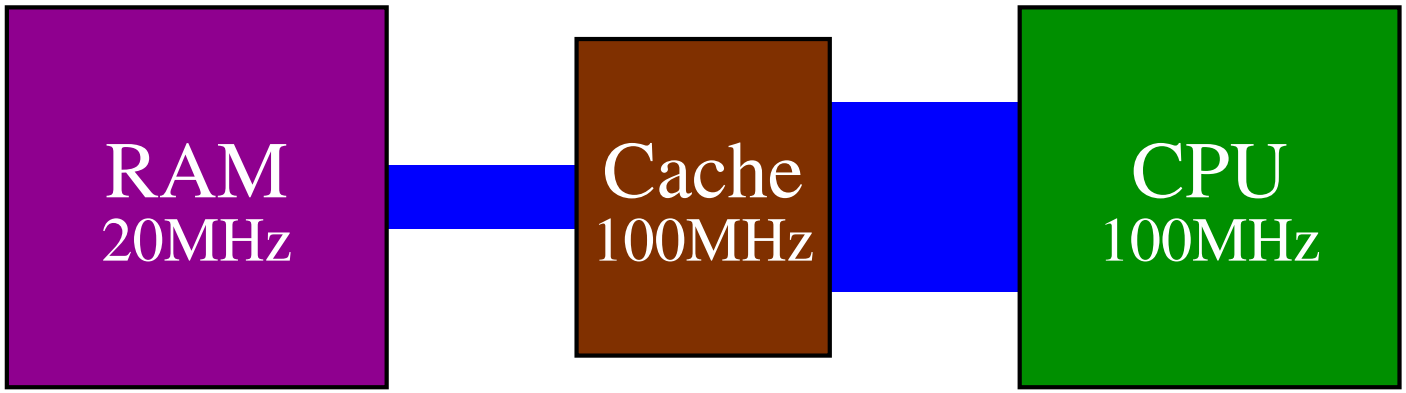


Depends on the system...

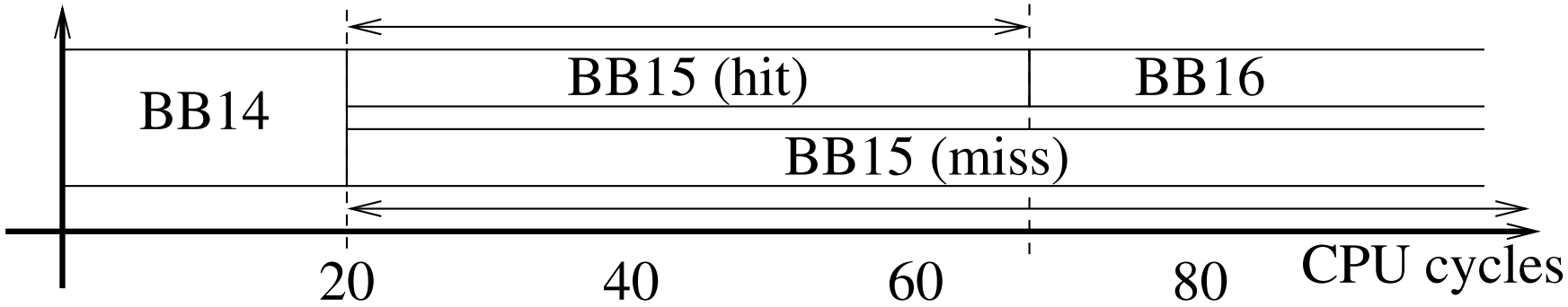
Basic Block Timing Invariance - 1



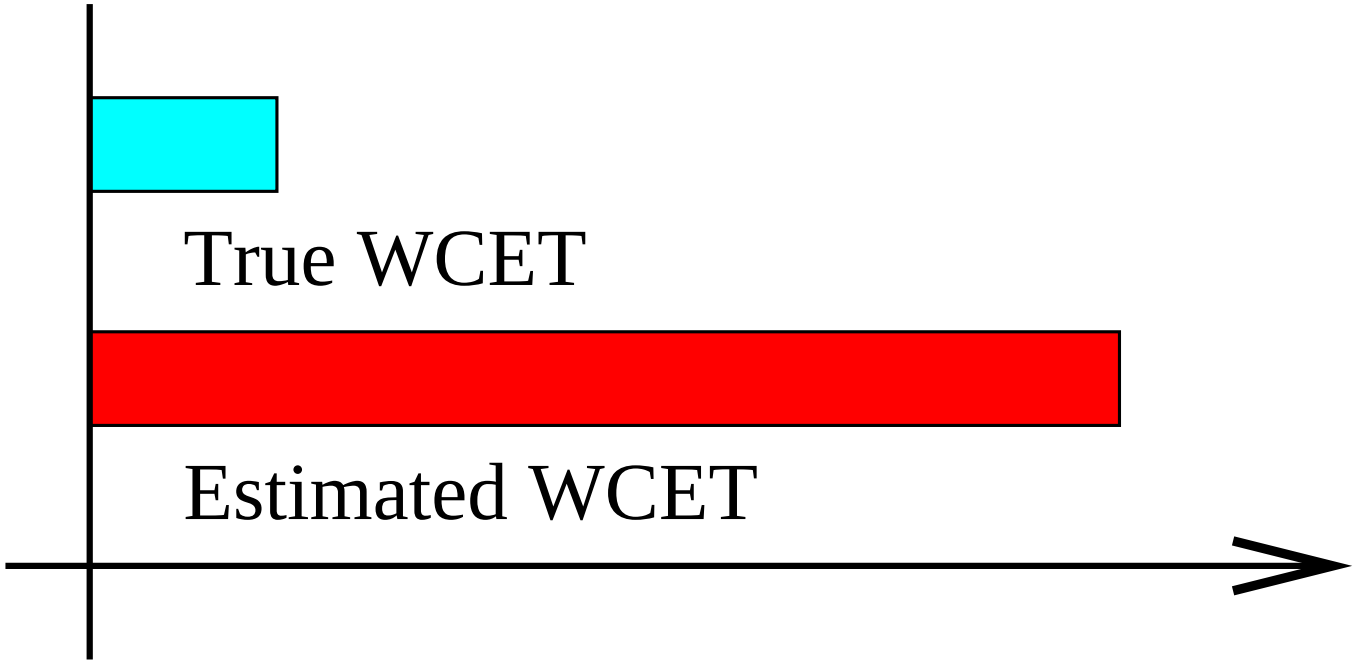
Basic Block Timing Invariance - 2



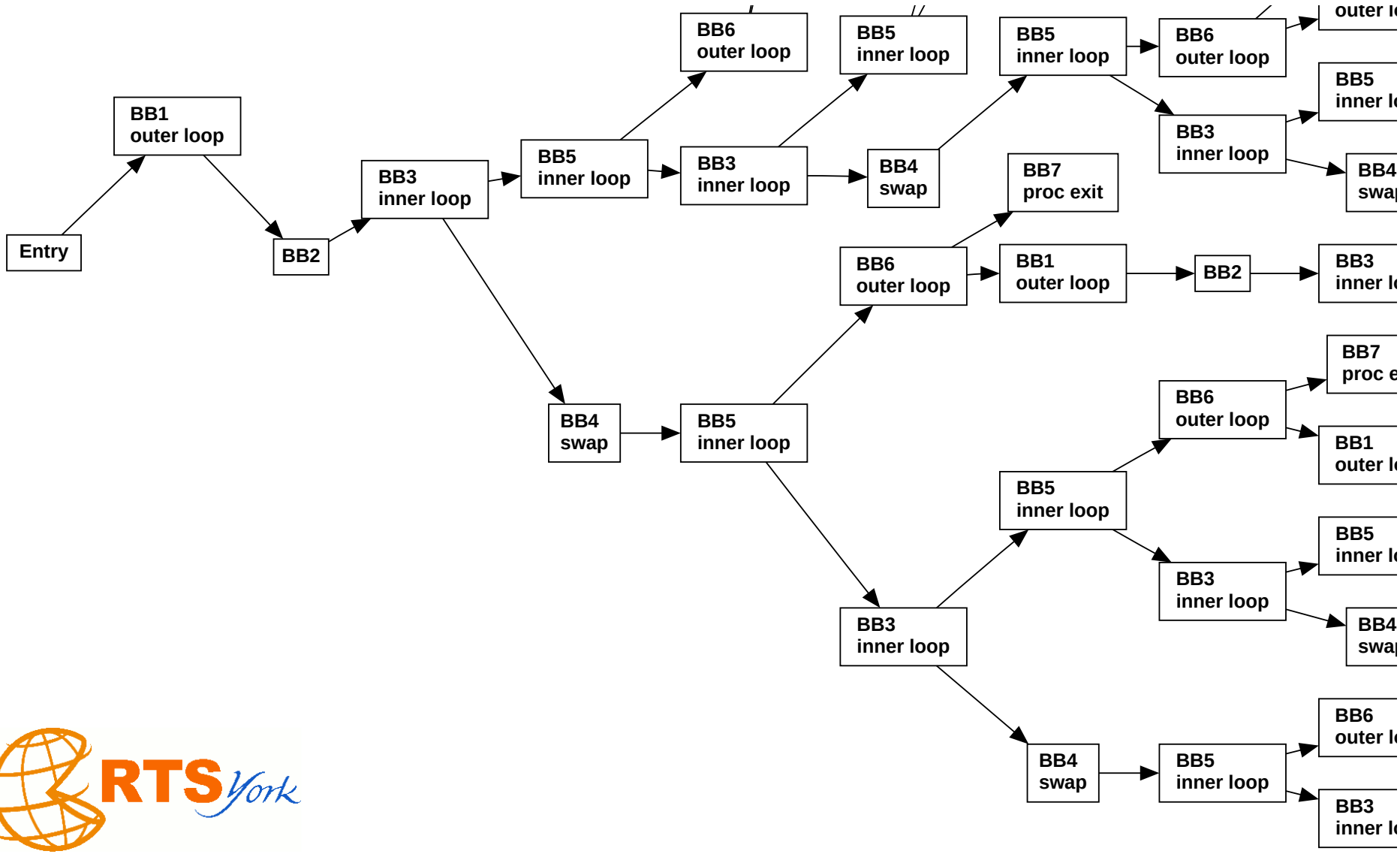
RTS circa
1995



Just use worst case?



Data dependence - again



IPET + Cache Model

Why AI + ILP is good for WCET, but MC is not, nor ILP alone

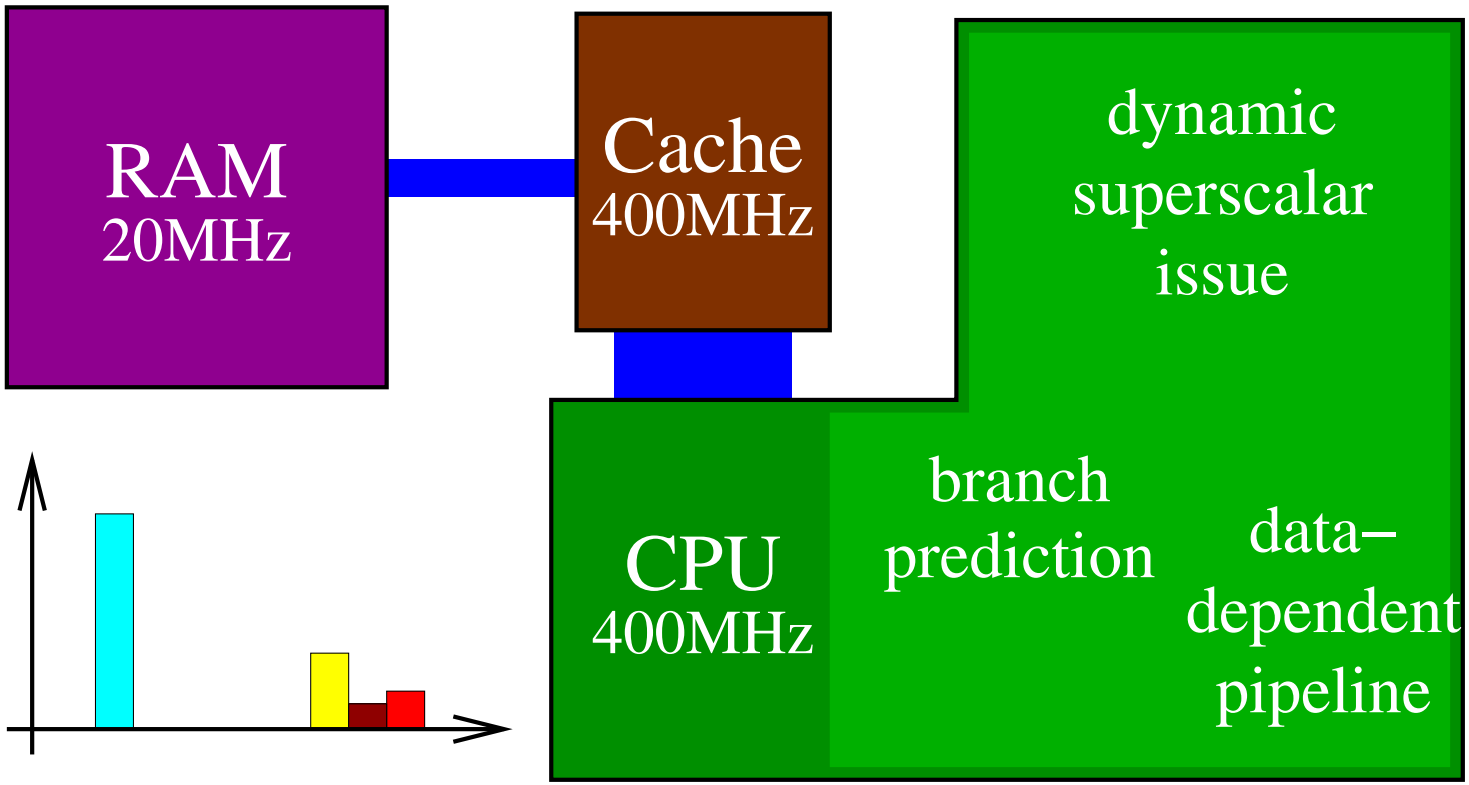
Wilhelm

5 WCET Determination by ILP

Li, Malik, and Wolfe were very successful with an ILP-only approach to WCET determination [9–12] . . . at least as far as getting papers about their approach published. Cache and pipeline behavior prediction are formulated as a single linear program. Using this approach for super-scalar pipelines does not seem very promising, considering the analysis times reported in the article. Thus, the size of the ILP is exponential in the size of the program. Even though the problem is claimed to be a network-flow problem the size of the ILP is killing the approach. Growing associativity of the cache increases the number of competing l-blocks. Thus, also increasing cache-architecture complexity plays against this approach.

Nonetheless, their method of modeling the control flow as an ILP, the so-called *Implicit Path Enumeration*, is elegant and can be efficient if the size of the ILP is kept small.

Basic Block Timing Invariance - 3



RTS circa
Today

Abstract Interpretation

- Cache replacement strategies: these should be immune against "chaos". This means that when cache contents are not known at one point, subsequent accesses can recover knowledge about the new cache contents.

Lundqvist

Stenstrom

1. Cache chaos
2. High modelling cost
3. Interaction effects
4. Timing anomalies (domino effect)

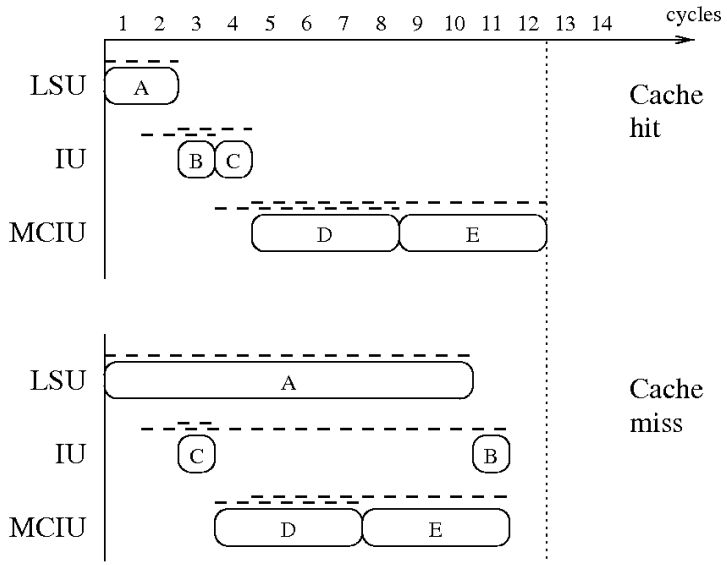
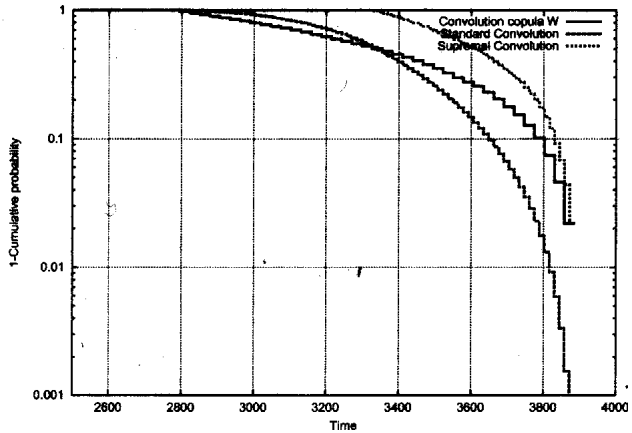


Figure An example when a cache hit causes a longer execution time than a cache miss.

Other solutions

1. Don't model the system explicitly! Use measurements to make a statistical model.
   G. Bernat and A. Burns and M. Newby. Probabilistic timing analysis: An approach using copulas, 2005



2. Simplify the system so that IPET modelling is possible.

What is the point?

1. Compute C
2. **Reduce** C

Question

Which technology...

1. Has predictable timing?
2. Can reduce memory access times?

Answer...

Question

Which technology...

1. Has predictable timing?
2. Can reduce memory access times?

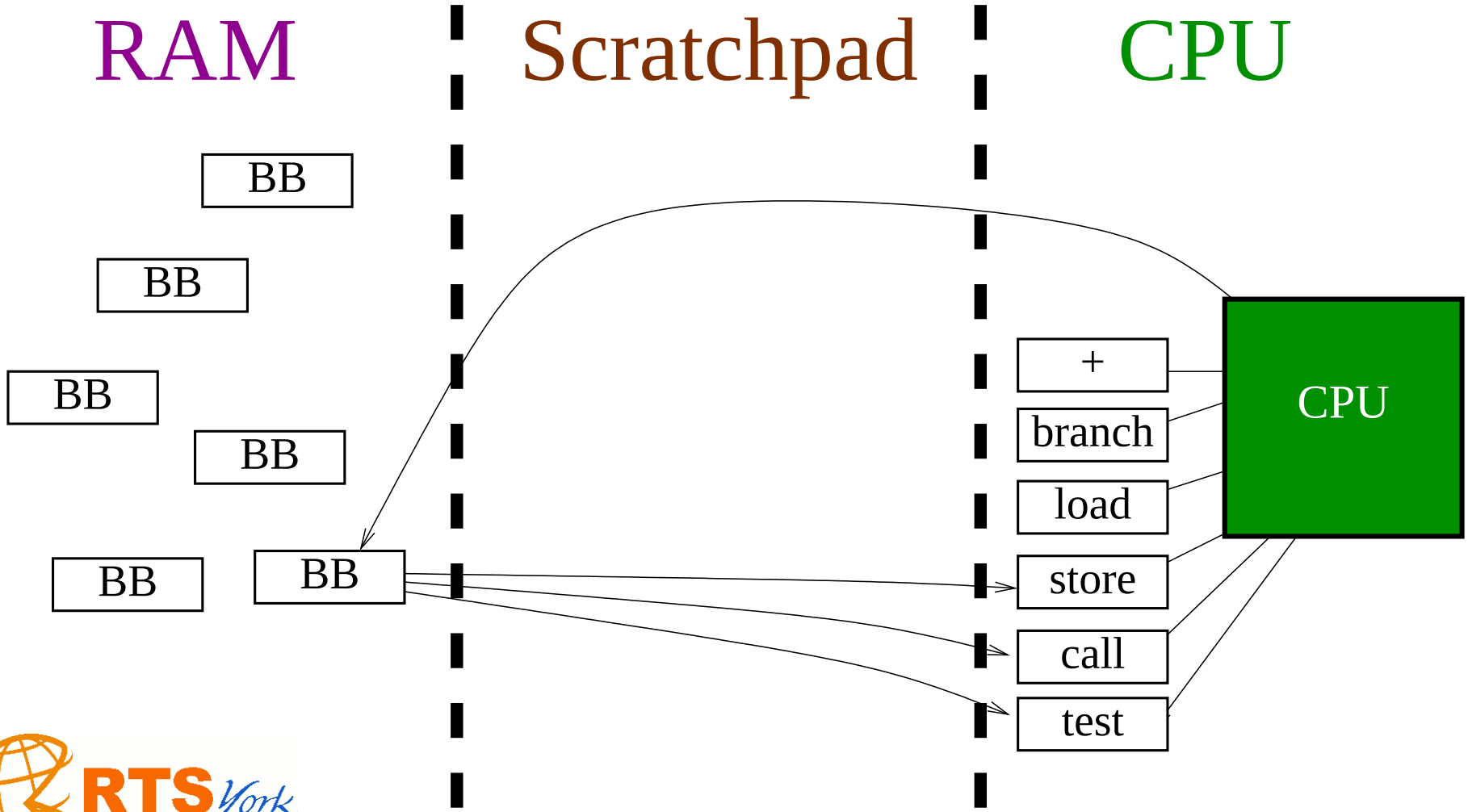
Answer...

- Locked Cache
  Puaut 2002, Falk 2006
- Scratchpad RAM
  Wehmeyer and Marwedel 2005, Suhendra et al. 2005, Puaut and Pais 2007

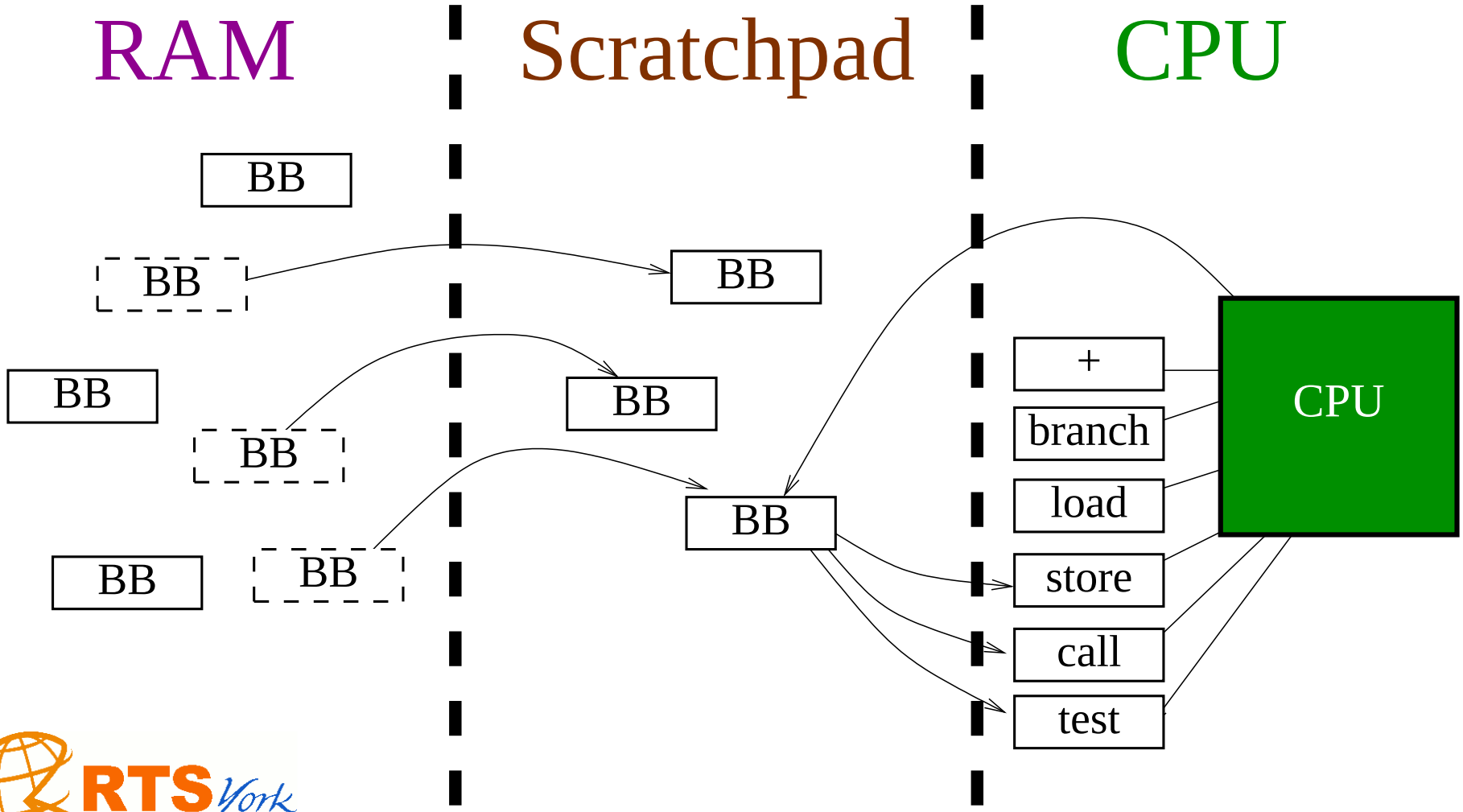
Scratchpads and Locked Caches

Name	Instruction Cache	Data Cache	Scratchpad Memory
Motorola Coldfire 5206e	512B	None	512B
Motorola Coldfire 5272	1KB	None	4KB
Motorola Coldfire 5249	8KB	None	32 and 64KB
ARM 940T	4KB	4KB	None
ARM 1020	32KB	32KB	None
Infineon Tricore TC1912	8KB	8KB	24KB (code) + 24KB (data)
MPC5567	8KB (unified)		64KB
MPC5554	32KB (unified)		64KB
MPC7410	32KB	32KB	None
Raza MIPS64 XL7100	32KB	32KB	None

Allocating scratchpad space - 1



Allocating scratchpad space - 2



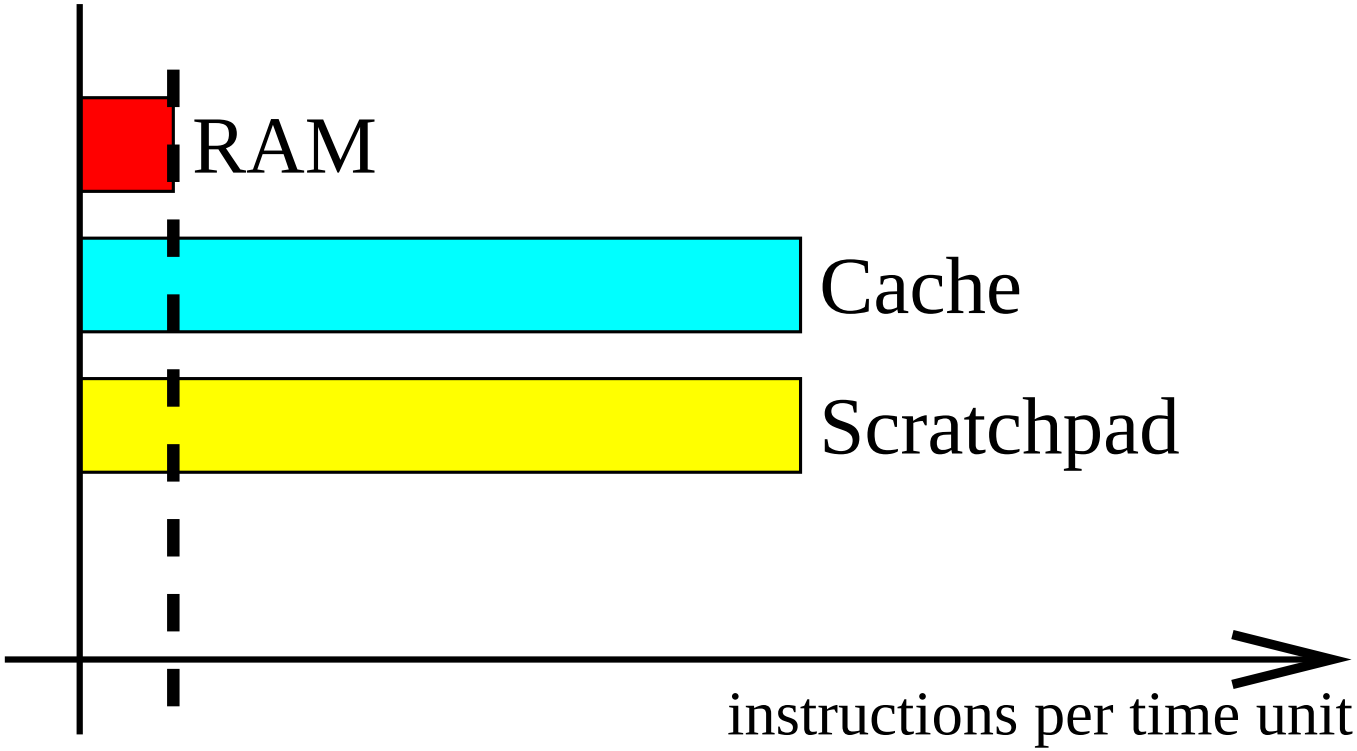
Suhendra's scratchpad allocation algorithm

While the scratchpad is not full, do the following:

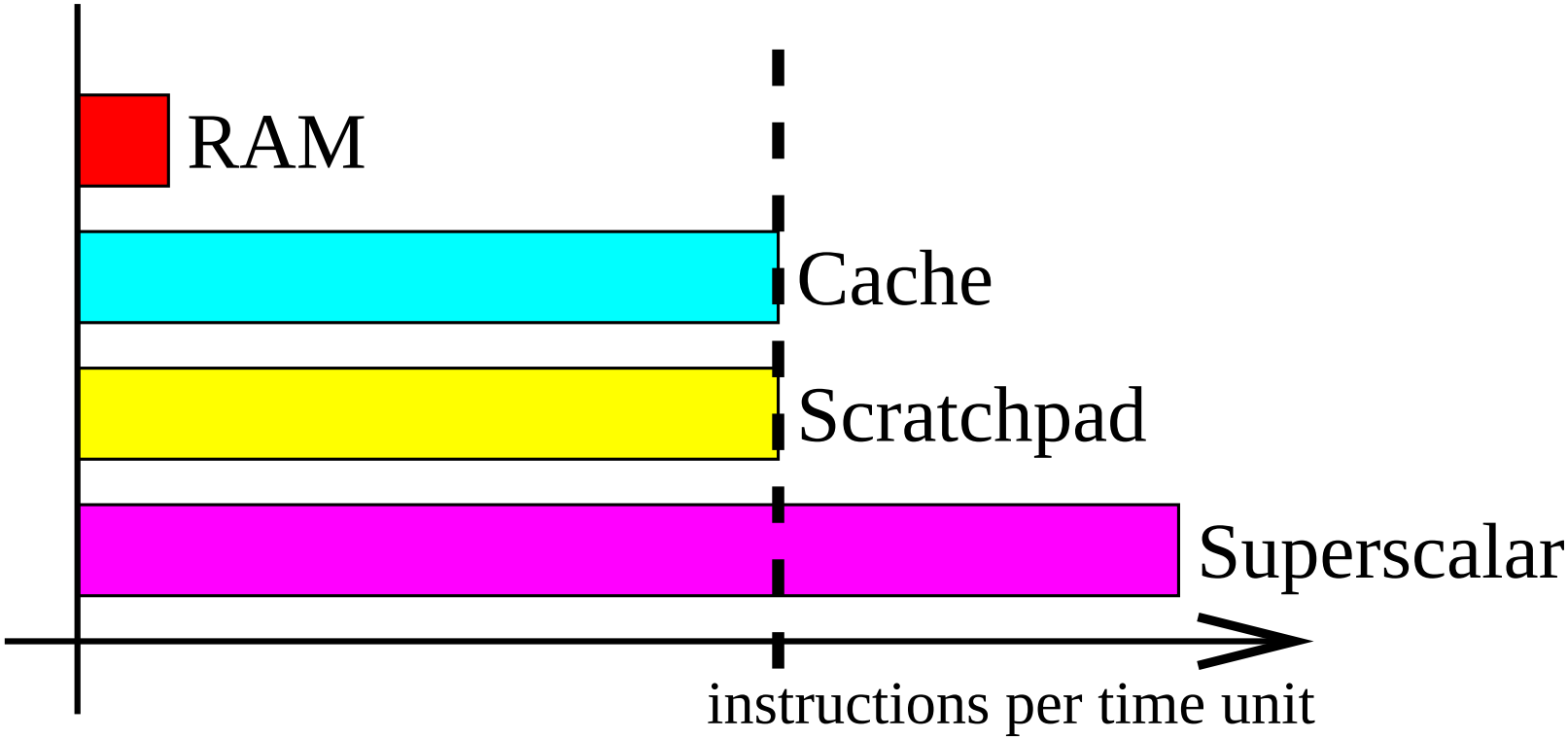
- Compute WCET: finds $f(x)$ and C.
- $L =$ all basic blocks not in scratchpad.
- Find $x \in L$ to maximise $\gamma(x)f(x)$.
- Decision: Migrate x in scratchpad.

Suhendra

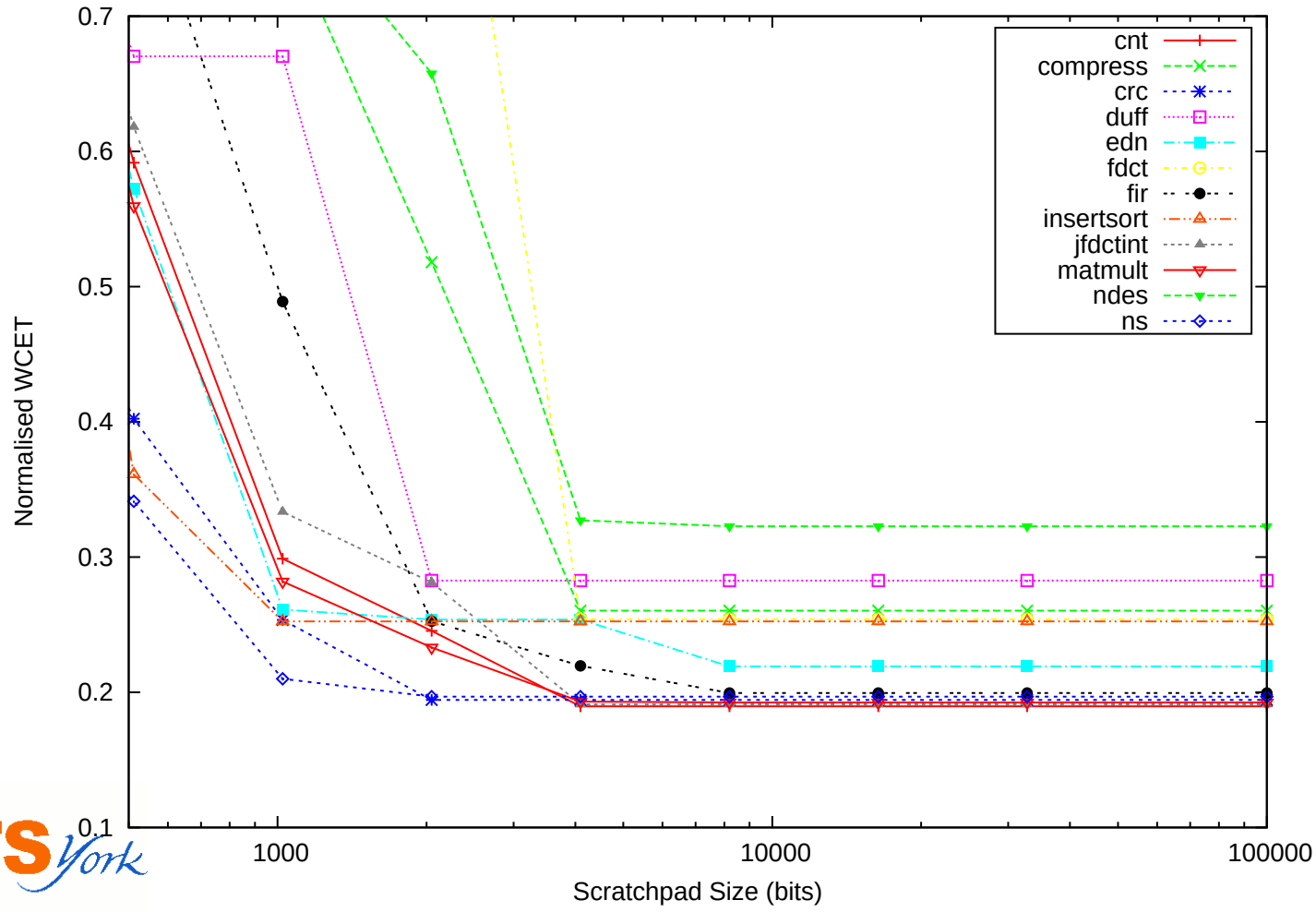
Scratchpad effectiveness



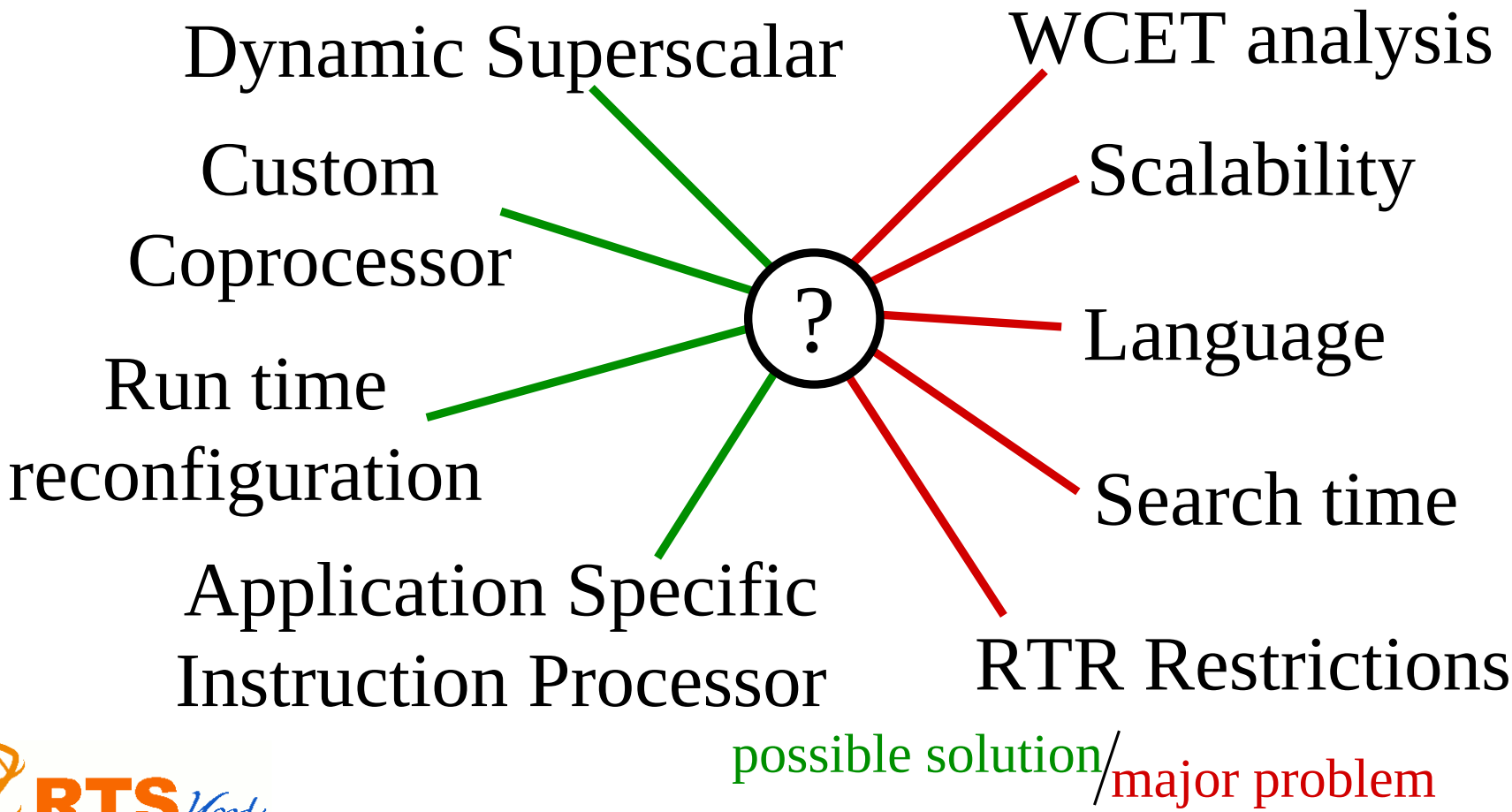
Bottlenecks



Maximum effective size

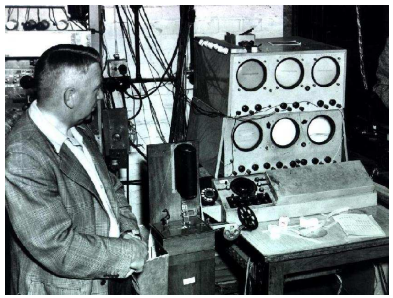


Where next?


Dynamic Superscalar
Custom
Coprocessor
Run time
reconfiguration
Application Specific
Instruction Processor
?
WCET analysis
Scalability
Language
Search time
RTR Restrictions
possible solution/major problem

Interesting bits of history - 1

1953

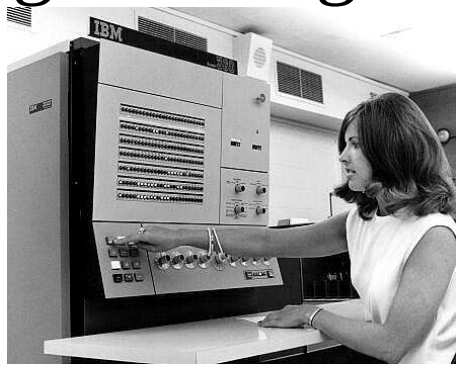


Microprogramming introduced
Used by EDSAC computer

Wilkes

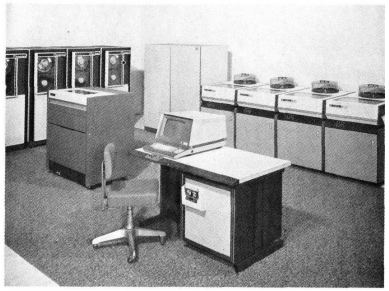
1964

Large scale use of microprogramming
IBM System 360

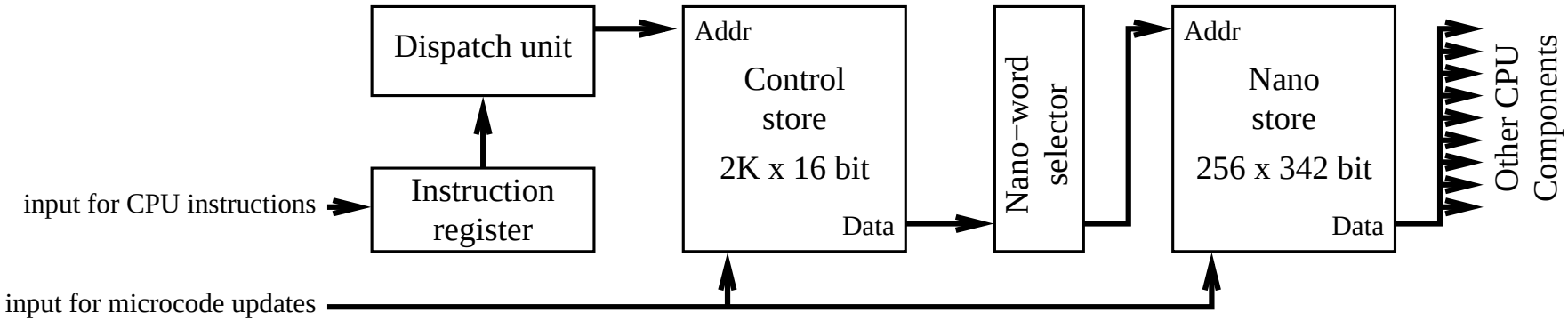


Interesting bits of history - 2

1970s

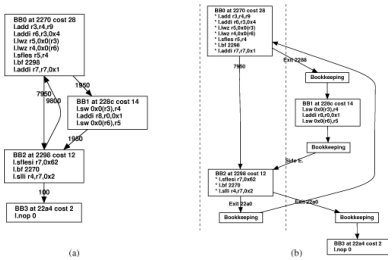


"User microprogramming" proposed and tested



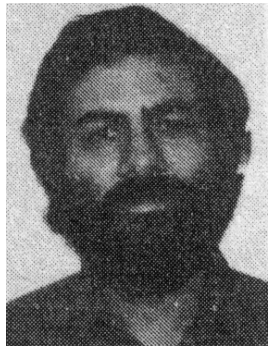
Interesting bits of history - 3

1980



Trace scheduling introduced for microprogramming

...code gen problem solved



Fisher

1985–

RISC CPUs are major success

...thanks to cache

Hennessy

Patterson

Interesting bits of history - 4

1990s

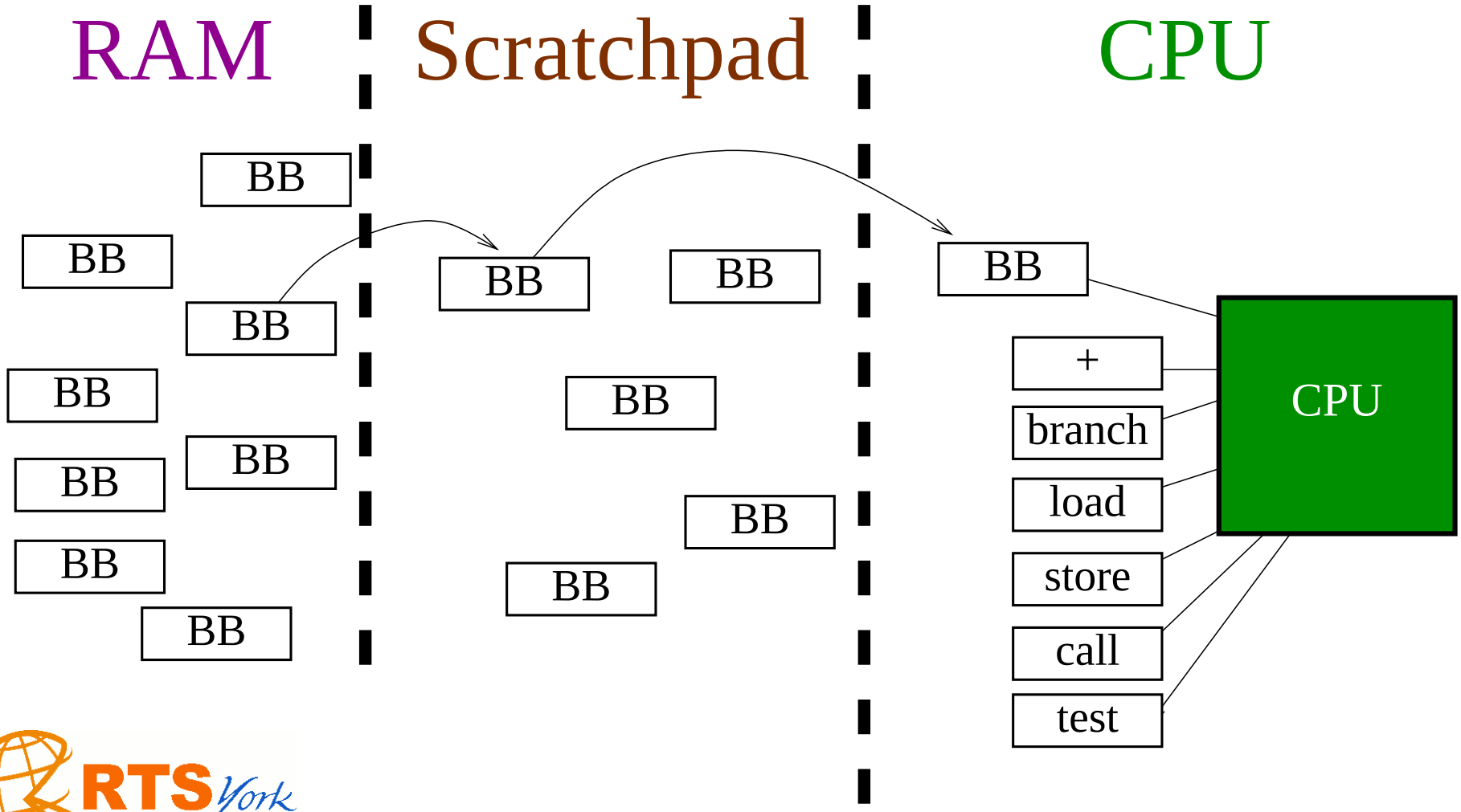
Performance of low−cost PCs exceeds that of specialist machines (RISC, VLIW)

RTS community responds with better models for modern systems

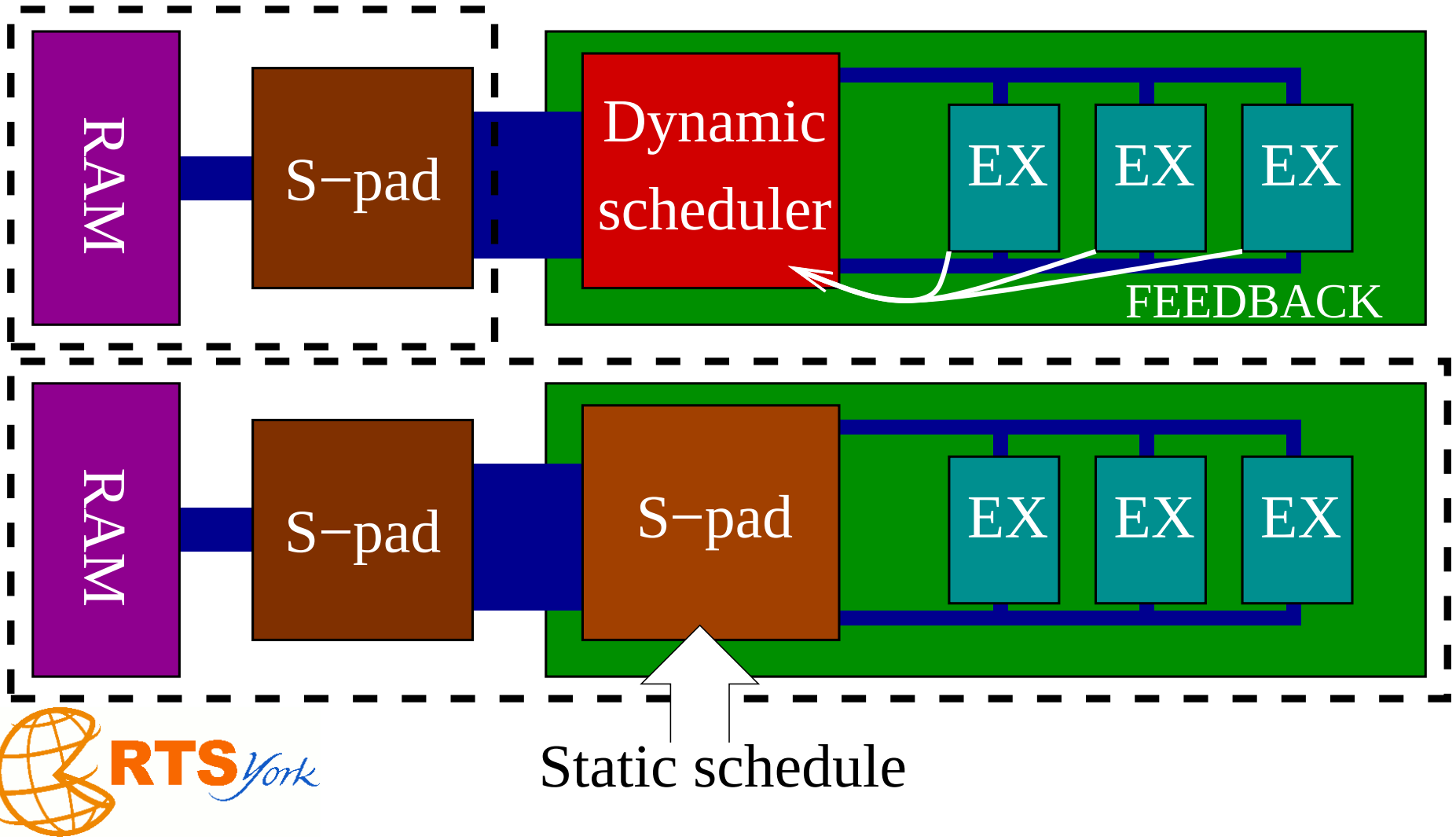
And then...

- CPU development was faster than analysis development.
- Barriers reached:
  - High cost of modelling.
  - Cache modelling problems.
  - Timing anomalies.

User microprogramming



From another viewpoint

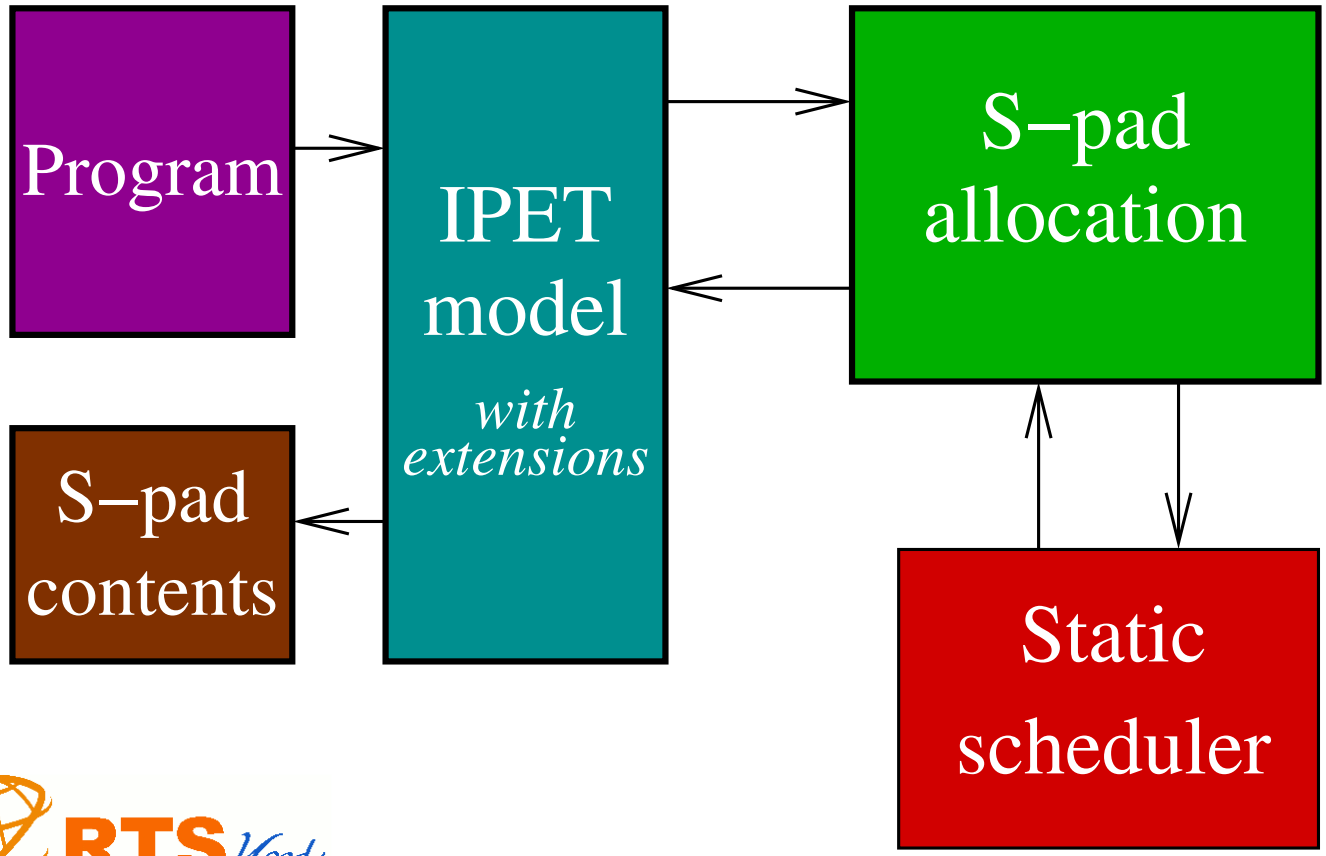


Caveats

- Can't migrate one basic block at a time: *allocation is more difficult.* (Traces)
- CPU must support user microprogramming: *very rare!*

So work is required...

Implementation

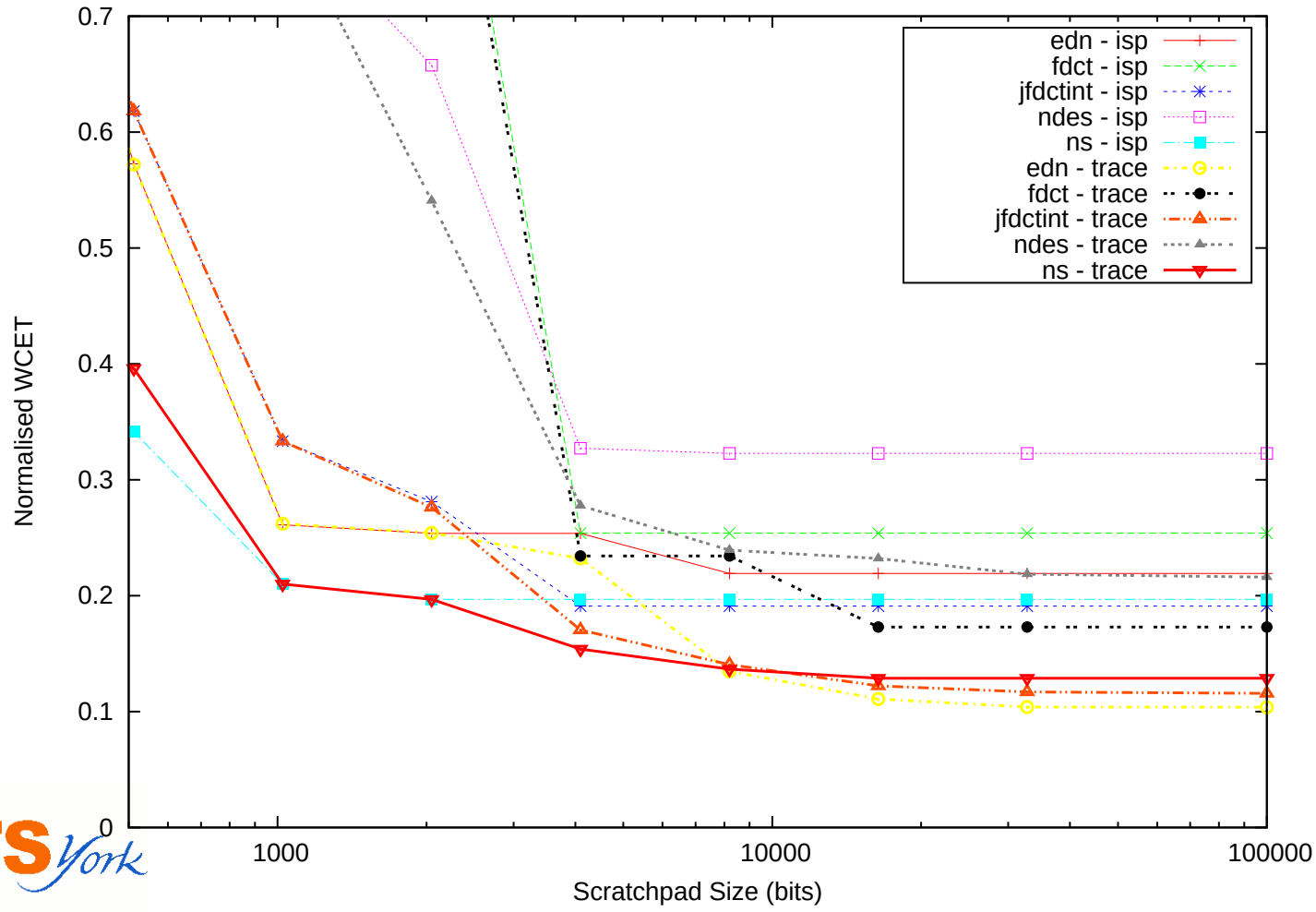


Sohi

One thesis later...

- CPU implemented: MCGREP-2.
- Static trace-style scheduler implemented.
- Allocation algorithm implemented.
- Results obtained.

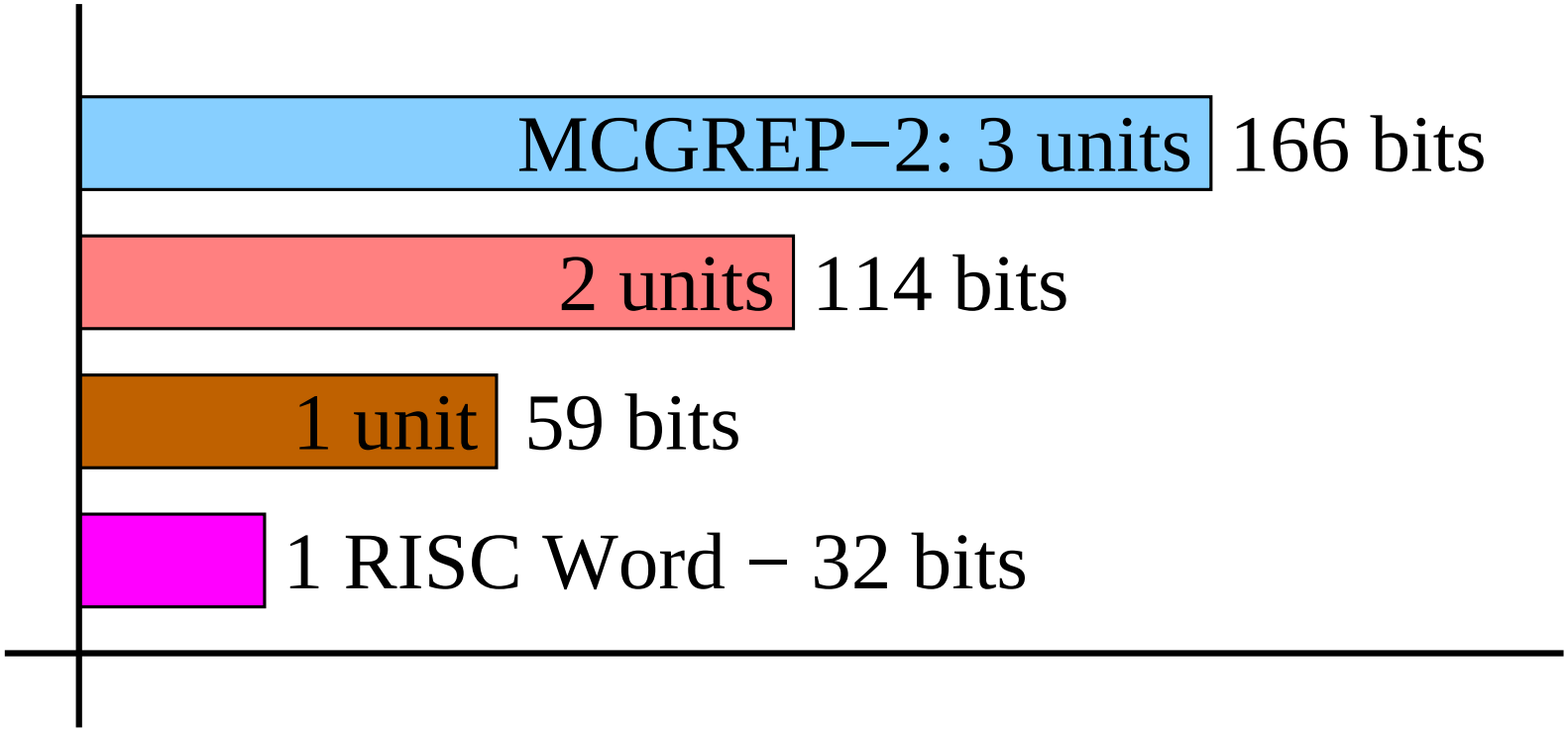
Results


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Normalised WCET
1000
10000
100000
Scratchpad Size (bits)
edn - isp
fdct - isp
jfdctint - isp
ndes - isp
ns - isp
edn - trace
fdct - trace
jfdctint - trace
ndes - trace
ns - trace

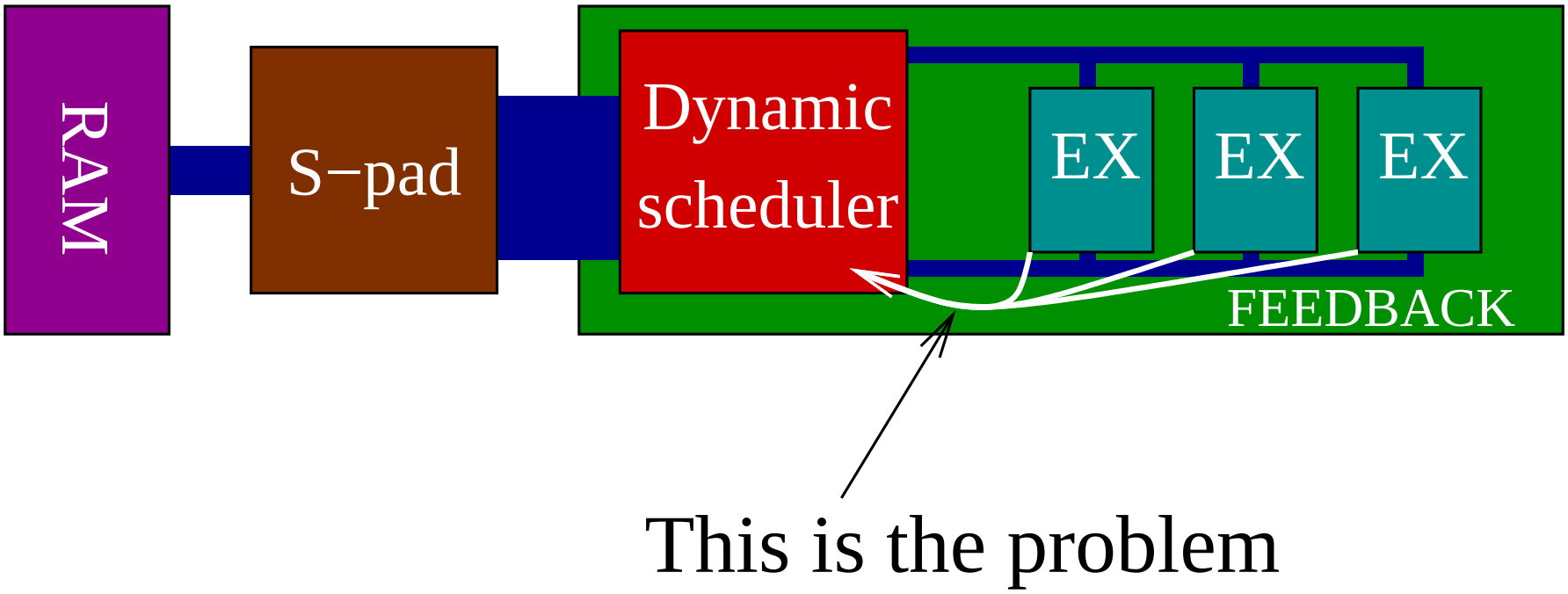
Success?

- Successful experiments.
- Up to 149% improvement observed over instruction scratchpad.
- Can be dynamically reloaded.
- Thesis finished.
- All software available for download (GPL).

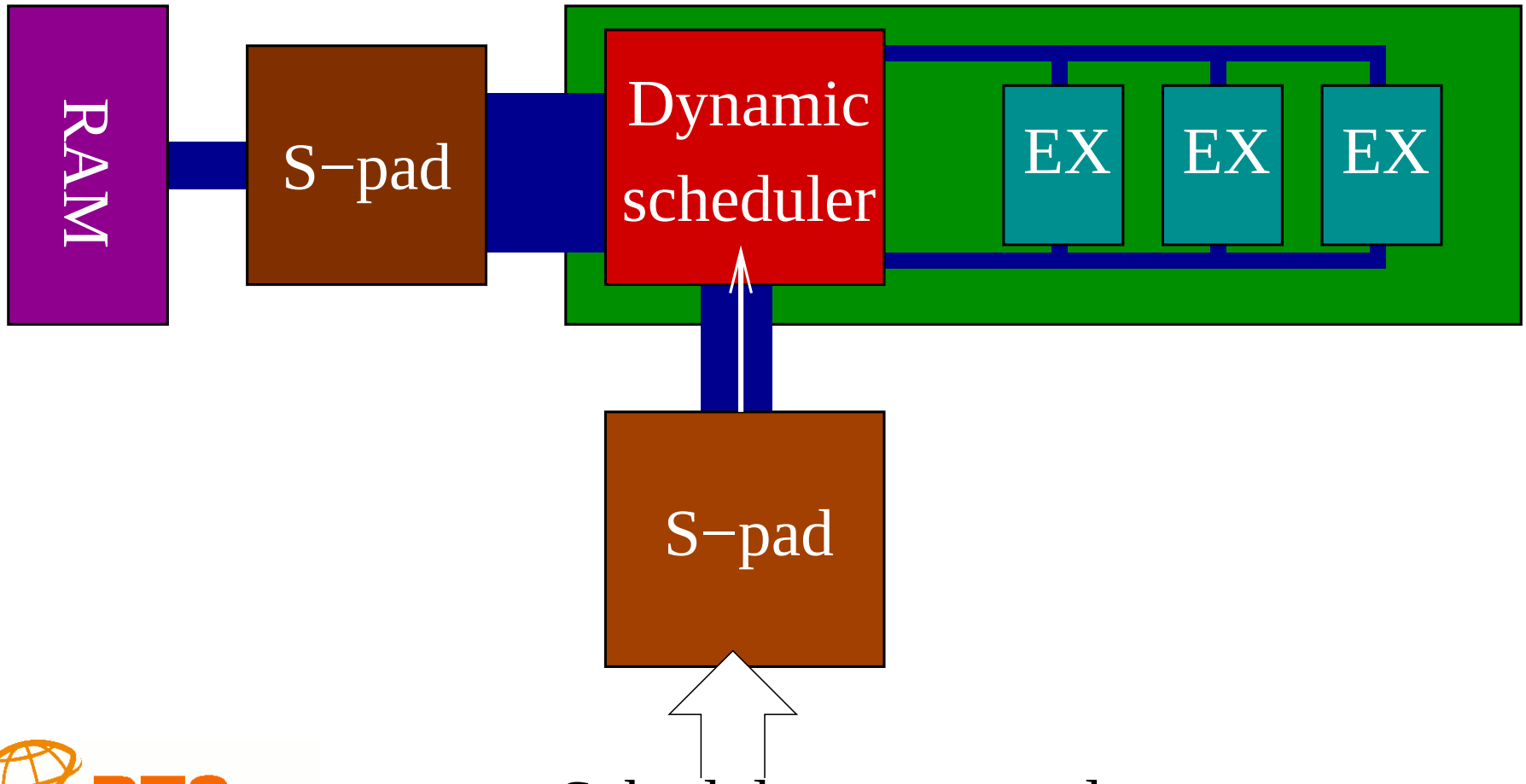
How to make it better



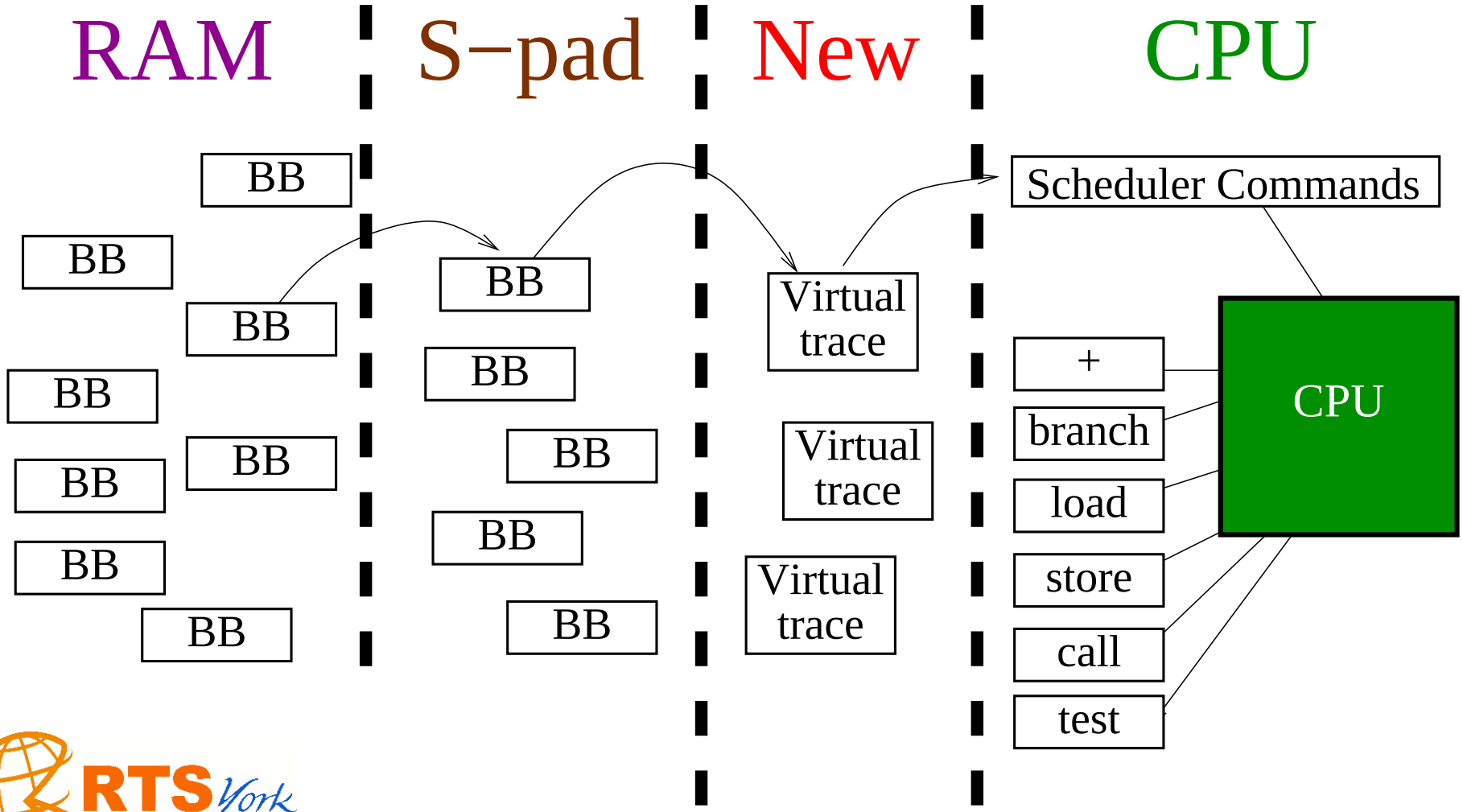
What's the problem?



Another solution



Virtual Trace



Conclusion

A solution for WCET reduction has been implemented.

- No need for a complex WCET model: some things with complex behaviour have predictable analogues.

Conclusion

A solution for WCET reduction has been implemented.

- No need for a complex WCET model: some things with complex behaviour have predictable analogues.
- Scratchpads are good, but WCET reductions are limited by the *instruction rate bottleneck*.

Conclusion

A solution for WCET reduction has been implemented.

- No need for a complex WCET model: some things with complex behaviour have predictable analogues.
- Scratchpads are good, but WCET reductions are limited by the *instruction rate bottleneck*.
- Microprogramming is a bottleneck solution that works, but there are *storage space issues*.

Conclusion

A solution for WCET reduction has been implemented.

- No need for a complex WCET model: some things with complex behaviour have predictable analogues.
- Scratchpads are good, but WCET reductions are limited by the *instruction rate bottleneck*.
- Microprogramming is a bottleneck solution that works, but there are *storage space issues*.
- Virtual traces are a solution for those issues, and a topic for subsequent research.

Conclusion

A solution for WCET reduction has been implemented.

- No need for a complex WCET model: some things with complex behaviour have predictable analogues.
- Scratchpads are good, but WCET reductions are limited by the *instruction rate bottleneck*.
- Microprogramming is a bottleneck solution that works, but there are *storage space issues*.
- Virtual traces are a solution for those issues, and a topic for subsequent research.
- THE END. Thankyou.