



Operating System Practice

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University

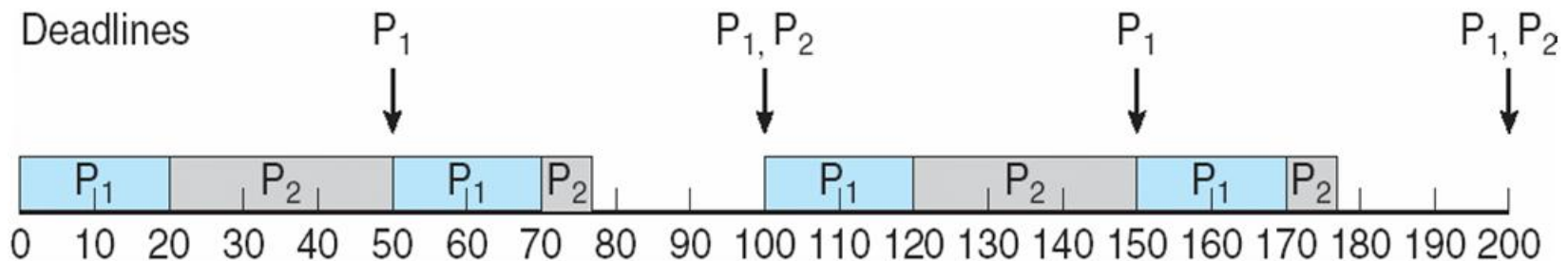


Worst-Case Execution Time (WCET) Analysis

Recall the Rate Monotonic Real-Time Scheduling

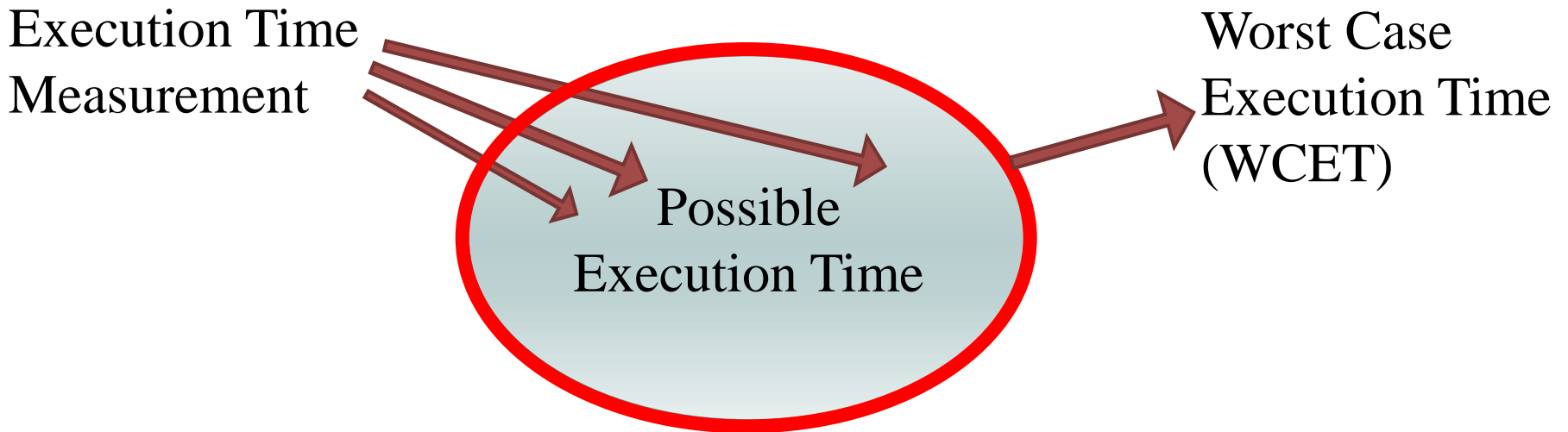
- ▶ A static priority is assigned to each task based on the inverse of its period
 - A task with shorter period → higher priority
 - A task with longer period → lower priority
 - For example:
 - P_1 has its period 50 and execution time 20
 - P_2 has its period 100 and execution time 37
 - P_1 is assigned a higher priority than P_2

How can we get the **EXECUTION TIME**



Execution Time of a Program

- ▶ The execution time of a program might not be a constant



WCETs are most essential assumptions for schedulability analysis

How to get the WCET of a program!?

Factors for WCET Analysis

- ▶ Input parameters
 - Algorithm parameters
 - Problem size
- ▶ States of the system
 - Cache configuration, cache replacement policies
 - Pipeline configuration
 - Speculations
- ▶ Interferences from the environment
 - Scheduling policies
 - Interrupts

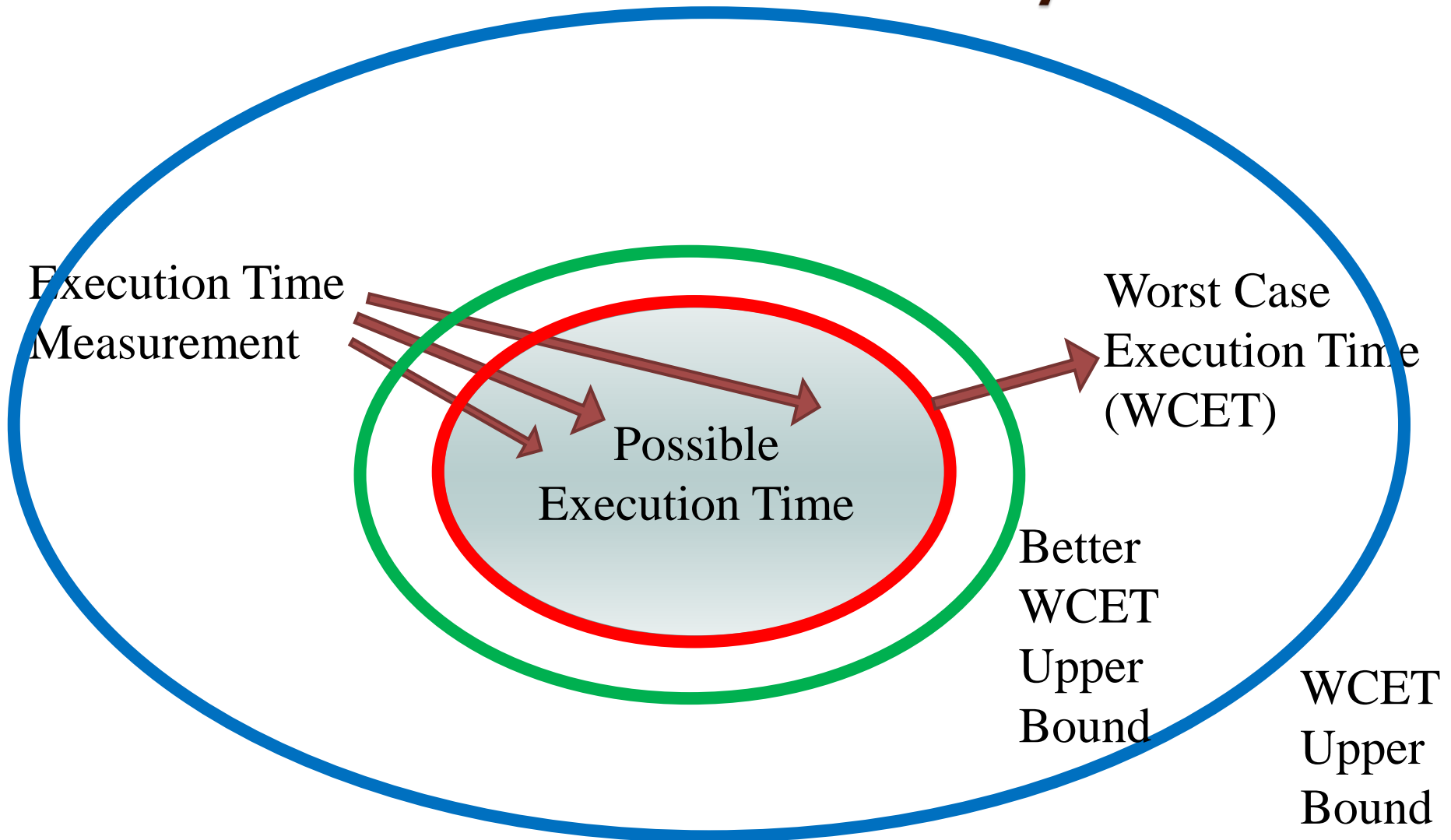


WCET Analysis

- ▶ Can we always get the WCET of a program?
 - **Halting Problem** tells us that we can not use an algorithm to decide whether another algorithm m halts on a specific input x .
 - Thus, **WCET is also undecidable**
- ▶ Most of industry's best practice
 - Measure it: determine WCET directly by running or simulating a set of inputs.
 - Exhaustive execution: by considering the set of all the possible inputs
- ▶ Another approach: compute an upper bound of the WCET
 - It should be no less than the WCET
 - It should be close to the WCET
 - It can not always be tight



Research of WCET Analysis



Challenges of Analyzing WCET

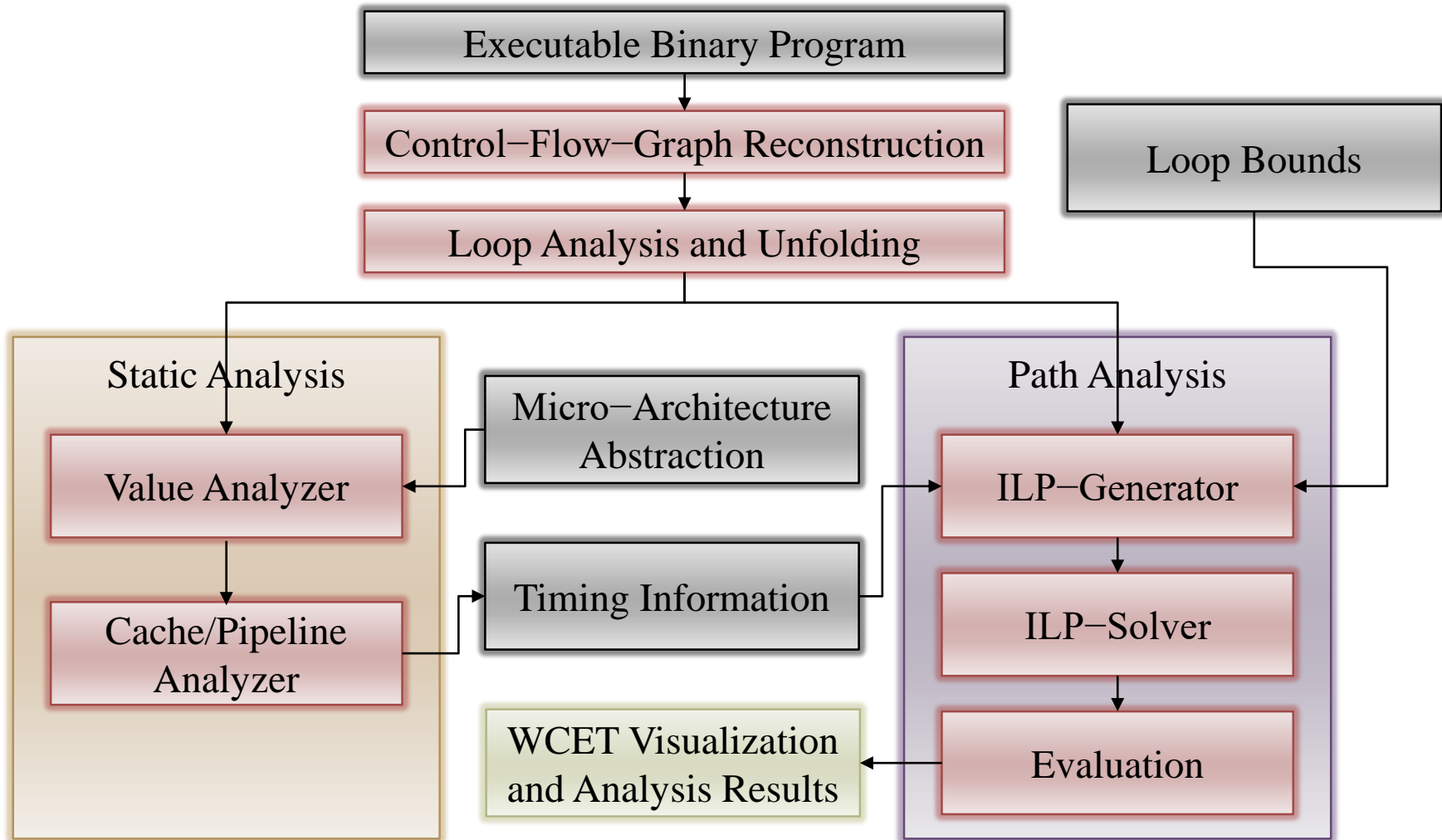
- ▶ Execution time $e(i)$ of machine instruction i
 - $e(i)$ is not a constant
 - The (architectural) execution state s should be considered
 - Thus, $e(i)$ is within the following range
$$\min \{e(i, s) | s \in S\} \leq e(i) \leq \max \{e(i, s) | s \in S\},$$
where S is the set of all states
- ▶ Using $\max \{e(i, s) | s \in S\}$ as the upper bound of WCET
 - It is safe
 - But it might be not tight

Timing Accidents and Penalties

- ▶ Timing Accident: cause for an increase of the execution time of an instruction
- ▶ Timing Penalty: the associated increase
- ▶ Types of timing accidents
 - Cache misses
 - TLB misses
 - Page faults
 - Pipeline stalls
 - Branch prediction errors
 - Bus collisions

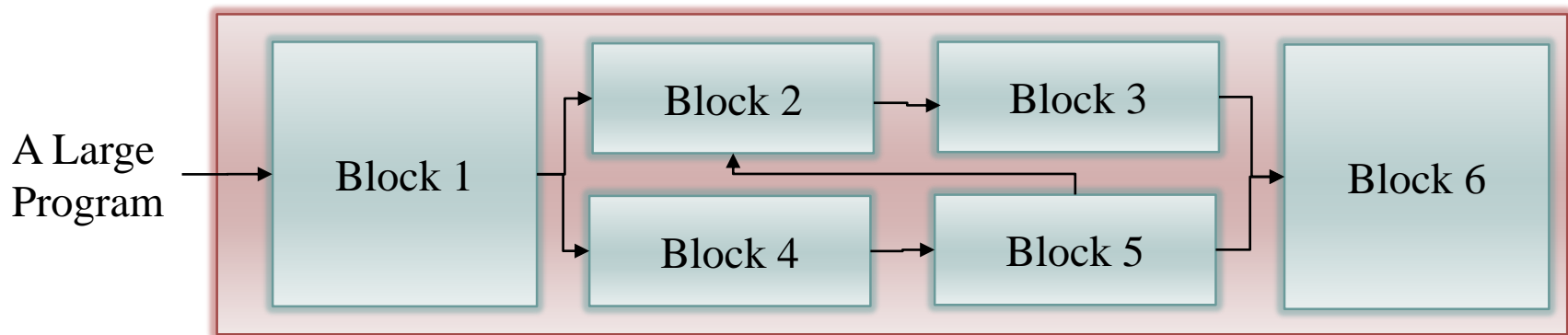


Overall Structure of WCET Analysis



Basic Blocks

- ▶ Beginning of Basic Blocks
 - The first instruction
 - The targets of un/conditional jumps
- ▶ Ending of Basic Block
 - The basic block consists of the block beginning and runs until the next block beginning (exclusive) or until the program ends



Value Analysis

▶ Motivation

- Provide access information to data-cache/pipeline analysis
- Detect infeasible paths
- Derive loop bounds

▶ Method

- Calculate intervals at all program points
- Consider addresses, register contents, local/global variables

▶ Abstract Interpretation

- Perform the program's computation using value descriptions or abstract values in place of the concrete values



Abstract Interpretation

▶ Abstract Domain

- Replace an integer/double operator by using intervals
- For example, $L = [3,5]$ stands for L is a value between 3 and 5

▶ Abstract Transfer

- For example, operator $+$: $[3, 5] + [2, 6] = [5, 11]$
- For example, operator $-$: $[3, 5] - [2, 6] = [-3, 3]$

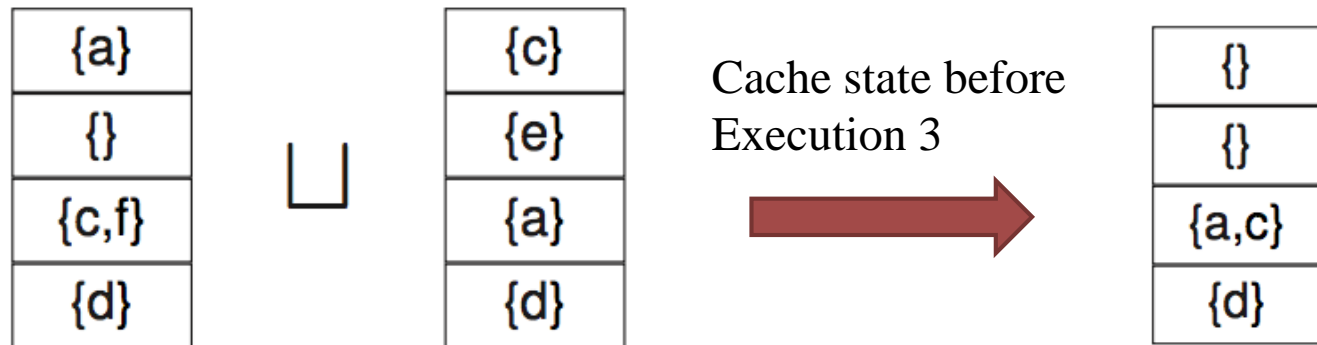
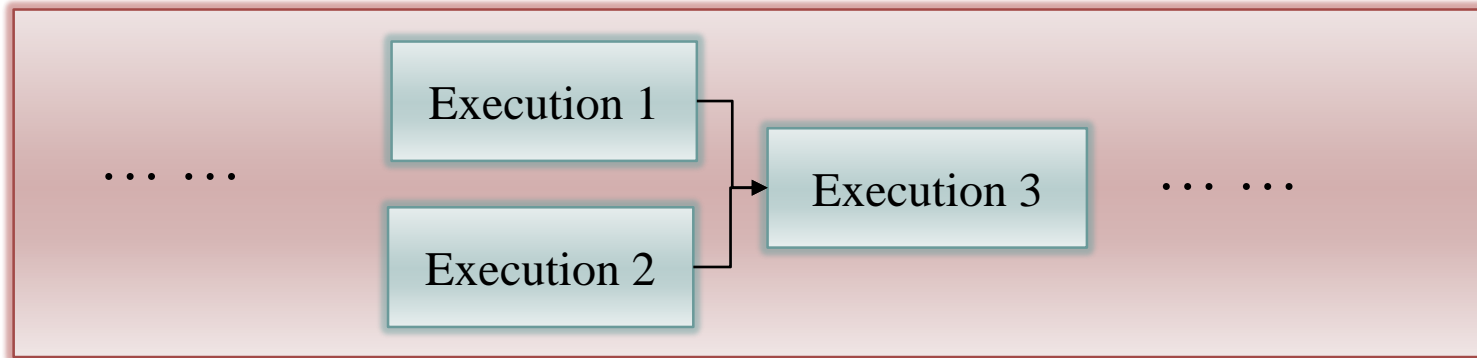
▶ Join Combining

- For example, $[a, b]$ join $[c, d]$ becomes $[\min\{a, c\}, \max\{b, d\}]$
- That is, $[3, 5]$ join $[2, 4]$ becomes $[2, 5]$



A Case Study with LRU: Join Management

Program
Execution



Pipelines

- ▶ An instruction execution consists of several sequential phases, e.g.,
 - Fetch
 - Decode
 - Execute
 - Write Back

Inst 1	Inst 2	Inst 3	Inst 4
Fetch			
Decode	Fetch		
Execute	Decode	Fetch	
Write Back	Execute	Decode	Fetch
	Write Back	Execute	Decode
		Write Back	Execute
			Write Back



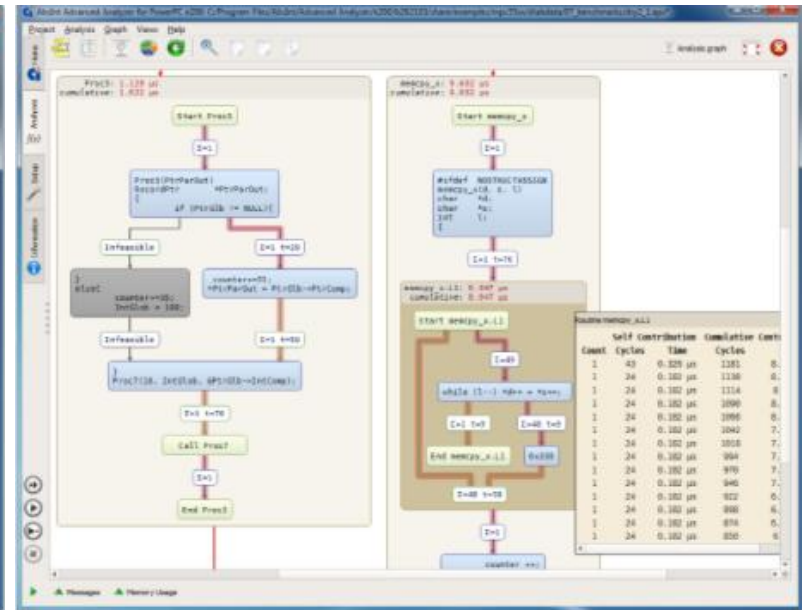
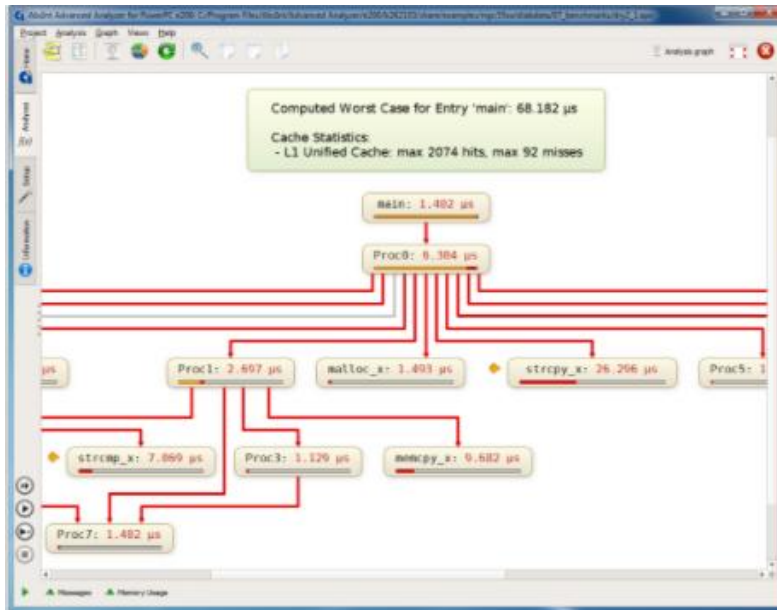
Hardware Features of Pipelines

- ▶ Instruction execution is split into several stages
- ▶ Several instructions can be executed in parallel
- ▶ Some pipelines can start more than one instruction per cycle: VLIW, Superscalar
- ▶ Some CPUs can execute instructions out-of-order
- ▶ Practical Problems: Hazards and cache misses
 - **Data Hazards**: Operands not yet available (Data Dependences)
 - **Control Hazards**: Conditional branch
 - **Resource Hazards**: Consecutive instructions use same resource
 - **Instruction-Cache Hazards**: Instruction fetch causes cache miss



WCET Analysis Tools (1 / 2)

▶ aiT WCET Analyzers



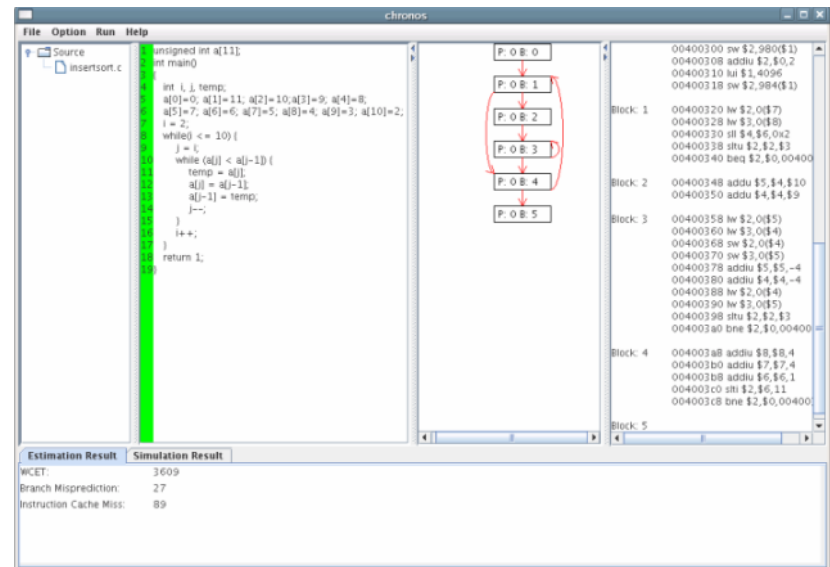
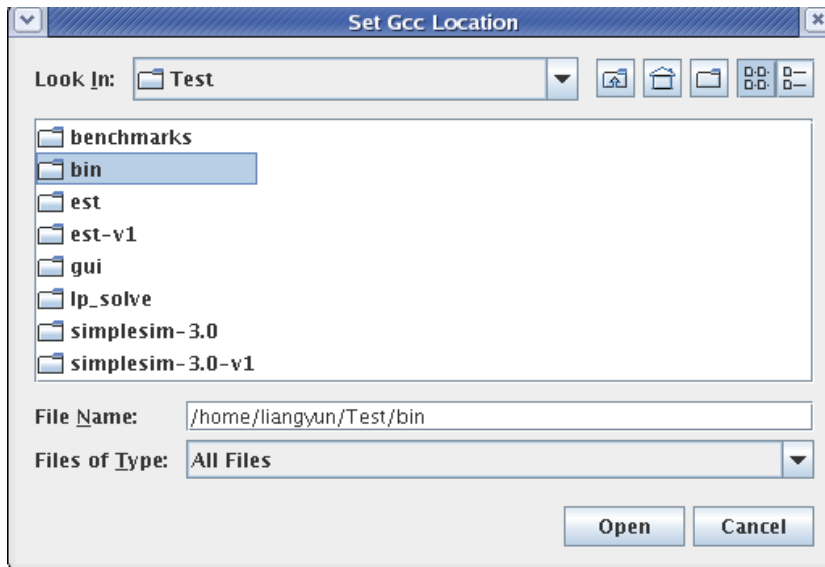
▶ It is not free

▶ <https://www.absint.com/ait/>



WCET Analysis Tools (2/2)

▶ Chronos



- ▶ It is free and open-source for academic
- ▶ But it is not stable
- ▶ <http://www.comp.nus.edu.sg/~rpembed/chronos/>

