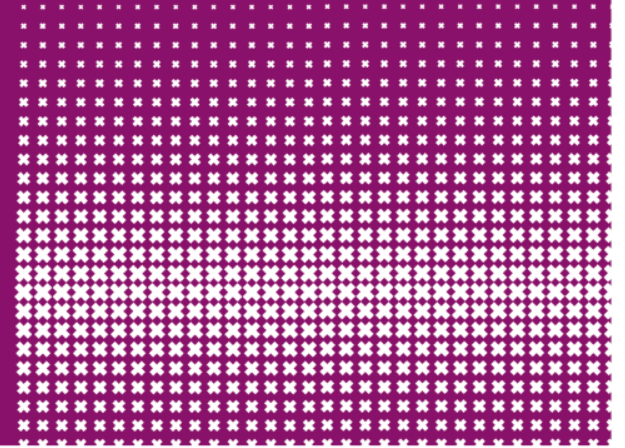




ing. Edwin Steffens

Practical lecturer – Informatics Institute - Faculty of Science



# Computer Architecture

## Lab 1 – Performance measurement



# Computer Architecture course

## Lab Experiments

### Expectations

### Introduction SIM-PL

### Kickstart Lab 1

# Lab Experiments

## Expectations

- Labs are compulsory
- Reports filenames must be conform  
`<Labname>-<YourCompleteName>.pdf`
- Source code filenames must be conform  
`<Labname><exercisenum>-<YourCompleteName>.pdf`
- Handed in via email before deafline: 21-09-2016 12:00 UTC.  
`e.h.steffens@uva.nl` and `cc t.r.walstra@uva.nl`

**No ZIP Files ! Not conforming? It will be rejected !**

More to read at the course page

[staff.fnwi.uva.nl/e.h.steffens/?page\\_id=18](http://staff.fnwi.uva.nl/e.h.steffens/?page_id=18)



# Introduction SIM-PL

## Logic simulator

# SIM-PL – Our tool to build and simulate computer architectures

Executor - SIM-PL V2.3.0 (SingleCycle.sim-pl-ws)

File Simulate Tools Settings Window Help

Mips Processor

File View Settings

Program Editor: (/home/walstra/lexyacc/tests\_singlecycle/test\_...

File Edit Settings

Memory Editor: Registers:Registers

Index	Alias	Value
0	zero	00000000
1	temp	00000000
2		00000005
3		00000000
4		00000000
5		00000000
6		00000000
7		00000000
8		00000000
9		00000000
10		00000000
11		FFFFFFFF
12		FFFFFFFF
13		FFFFFFFF
14		FFFFFFFF
15		FFFFFFFF

Memory Editor: Data Memory:Memory

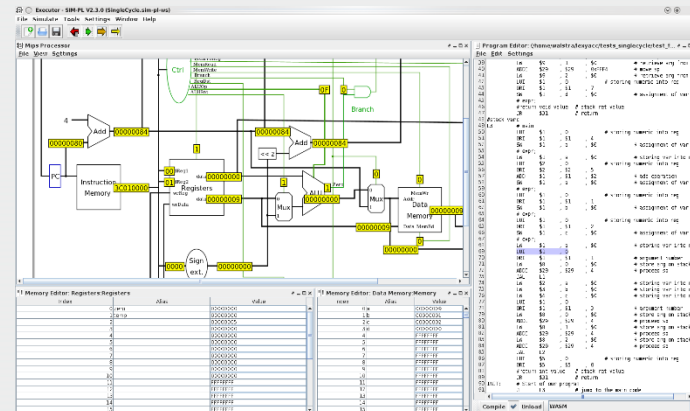
Index	Alias	Value
0	a	00000009
1	b	00000001
2	c	00000002
3	d	00000000
4		FFFFFFFF
5		FFFFFFFF
6		FFFFFFFF
7		FFFFFFFF
8		FFFFFFFF
9		FFFFFFFF
10		FFFFFFFF
11		FFFFFFFF
12		FFFFFFFF
13		FFFFFFFF
14		FFFFFFFF
15		FFFFFFFF

39 LW \$9, 1, \$0 # retrieve arg from  
40 ADDI \$29, \$29, 0xFFFF # move sp  
41 LW \$9, 2, \$0 # retrieve arg from  
42 LUI \$1, 0 # storing numeric into reg  
43 ORI \$1, \$1, 7 # assignment of var  
44 SW \$1, d, \$0 # assignment of var  
45 # expr: void value # stack ret value  
46 #return void value # stack ret value  
47 JR \$31 # return  
48 #stack varc  
49 L3:  
50 LUI \$1, 0 # storing numeric into reg  
51 ORI \$1, \$1, 4 # assignment of var  
52 SW \$1, a, \$0 # assignment of var  
53 # expr:  
54 LW \$1, a, \$0 # storing var into re  
55 LUI \$2, 0 # storing numeric into reg  
56 ORI \$2, \$2, 5 # assignment of var  
57 ADDI \$1, \$1, 2 # add operation  
58 SW \$1, a, \$0 # assignment of var  
59 # expr:  
60 LUI \$1, 0 # storing numeric into reg  
61 ORI \$1, \$1, 1 # assignment of var  
62 SW \$1, b, \$0 # assignment of var  
63 # expr:  
64 LUI \$1, 0 # storing numeric into reg  
65 ORI \$1, \$1, 2 # assignment of var  
66 SW \$1, c, \$0 # assignment of var  
67 # expr:  
68 LW \$1, a, \$0 # storing var into re  
69 LUI \$1, 0 # storing numeric into reg  
70 ORI \$1, \$1, 1 # argument number  
71 LW \$8, 0, \$0 # store arg on stack  
72 ADDI \$29, \$29, 4 # proceed sp  
73 JAL L1  
74 LW \$2, a, \$0 # storing var into re  
75 LW \$3, b, \$0 # storing var into re  
76 LW \$4, c, \$0 # storing var into re  
77 LUI \$1, 0  
78 ORI \$1, \$1, 3 # argument number  
79 LW \$8, 0, \$0 # store arg on stack  
80 ADDI \$29, \$29, 4 # proceed sp  
81 LW \$8, 1, \$0 # store arg on stack  
82 ADDI \$29, \$29, 4 # proceed sp  
83 LW \$8, 2, \$0 # store arg on stack  
84 ADDI \$29, \$29, 4 # proceed sp  
85 JAL L2  
86 LUI \$5, 0 # storing numeric into reg  
87 ORI \$5, \$5, 0 # stack ret value  
88 #return int value # stack ret value  
89 JR \$31 # return  
90 INIT: # Start of our program  
91 J L3 # jump to the main code

Compile Unload WASM

# SIM-PL – Our tool to build and simulate computer architectures

- “Editor” section for building (complex) components.
- “Executer” section for simulating
- Written in Java



# SIM-PL - Executor

The screenshot displays the SIM-PL Executor interface, which is a multi-window application for simulating a MIPS processor. The main window, titled "MultiCycle MIPS Processor", shows a block diagram of the processor. It includes a "Computer" block containing a "CONTROL" unit and a "DATAPATH" unit. The "CONTROL" unit is connected to the "DATAPATH" unit. The "DATAPATH" unit has an "INPUT" and an "OUTPUT". A memory location "0000001C" is highlighted in the diagram.

Below the main window, there are three smaller windows:

- Time Sequence Diagram:** Shows a clock signal and a sequence of events over time. The clock is set to 25.0.
- Memory Editor: Regi...:** A table showing the state of registers. The table has columns for Index, Alias, and Value.
- Me...:** A table showing the state of memory. The table has columns for Index, Alias, and Value.

On the right side of the interface, there is a "Program Editor" window showing the MIPS assembly code for the simulation. The code is as follows:

```

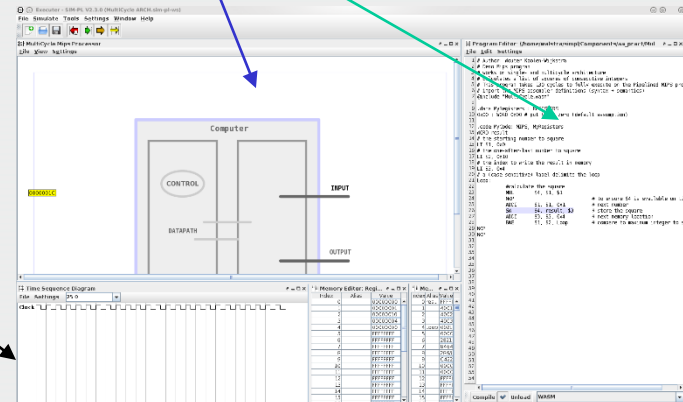
1 # Author: Wouter Koelen-Wijkstra
2 # Demo MIPS program
3 # works on single- and multicyle architecture
4 # calculates a list of squares of consecutive integers
5 # This program takes 130 cycles to fully execute on the Pipelined MIPS pro
6 # import the MIPS assembler definitions (syntax + semantics)
7 @include "MultiCycle.wasm"
8
9 .data MyRegisters : REGISTERS
10 0x00 : WORD 0x00 # put $0 to zero (default assumption)
11
12 .code MyCode: MIPS, MyRegisters
13 WORD result
14 # the starting number to square
15 LI $1, 0x0
16 # the one-after-last number to square
17 LI $2, 0x10
18 # the index to write the result in memory
19 LI $3, 0x4
20 # a (case sensitive) label delimits the loop
21 Loop:
22     #calculate the square
23     MUL $4, $1, $1
24     NOP # to ensure $4 is available on ti
25     ADDI $1, $1, 0x1 # next number
26     SW $4, result, $3 # store the square
27     ADDI $3, $3, 0x4 # next memory location
28     BNE $1, $2, Loop # compare to maximum integer to s
29     NOP
30     NOP
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

At the bottom right, there are buttons for "Compile", "Unload", and "WASM".

# SIM-PL - Executer

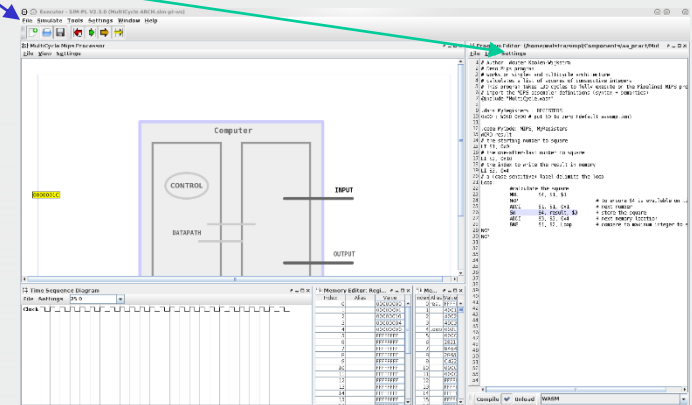
- Executer is started at the command prompt
- Java -jar Executer.jar
- Designs are loaded in the “Component view”.
- Programs are loaded in the “Program editor”.
- Simulation results are shown in
  - Time sequence diagram
  - Memory editor
  - Register editor





# SIM-PL – Executer Running Simulations

- Designs are loaded in the “Component view”.
- File->open or new
- Programs are loaded in the “Program editor”.
- File->open
- Compile
- Compiled code transferred to the design
- Press the Red arrow button
- Simulation starts by pressing the
- Green or Yellow or Orange button.





Downloads available

Manual

SIM-PL 2.3.0

Additional components

Course webpage [e.h.steffens](http://e.h.steffens)



# Lab 1

## Performance of Computer Architectures

# Lab 1 experiment - Performance

## Goal

To understand how performance is measured.

## How

By doing experiments in SIM-PL the logic simulator.

Researching by running three small programs on three architectures

## Tools

SIM-PL (Logic simulator)

## Results

The results of the assignments are handed in for grading.



# Kickstart Lab 1

Description

Course webpage

[staff.fnwi.uva.nl/e.h.steffens](http://staff.fnwi.uva.nl/e.h.steffens)

# Lab 1 – Experiment 1

- Goal is to determine the CPI of a computer architecture
- How ?
- Load an architecture in SIM-PL
- Execute a program
- Count the clock cycles and instructions until the program ends
- Determine the CPI ( $\#clocks / \#instructions$ )

# Lab 1 – Loading an architecture

- Download and install SIM-PL
- Start the executer
- Java -jar executer.jar (Linux case sensitive)
- Load the SingleCycle architecture worksheet
- File->Open
  - SingleCycle-architecture.sim-pl-ws
  - Let's start with an addition of two numbers
- Load the assembly program addition.wasm in the Program Editor
- Press the button “Compile” and wait for the checkmark

# Lab 1 – Running the source code

- Load the compiled source code
  - Press the “Red arrow” button
  - First instruction is highlighted
- Execute the first instruction
  - Press the “Orange arrow” button
  - Clock cycle appears in Timing window
- Continue until all instructions are executed
  - No highlighted instructions is end
  - Don’t forget to count !
- Write down the result in your report.
- Continue with the other programs and architectures





# End of kick start session

Are there any questions ?



# See you next time

# Success !