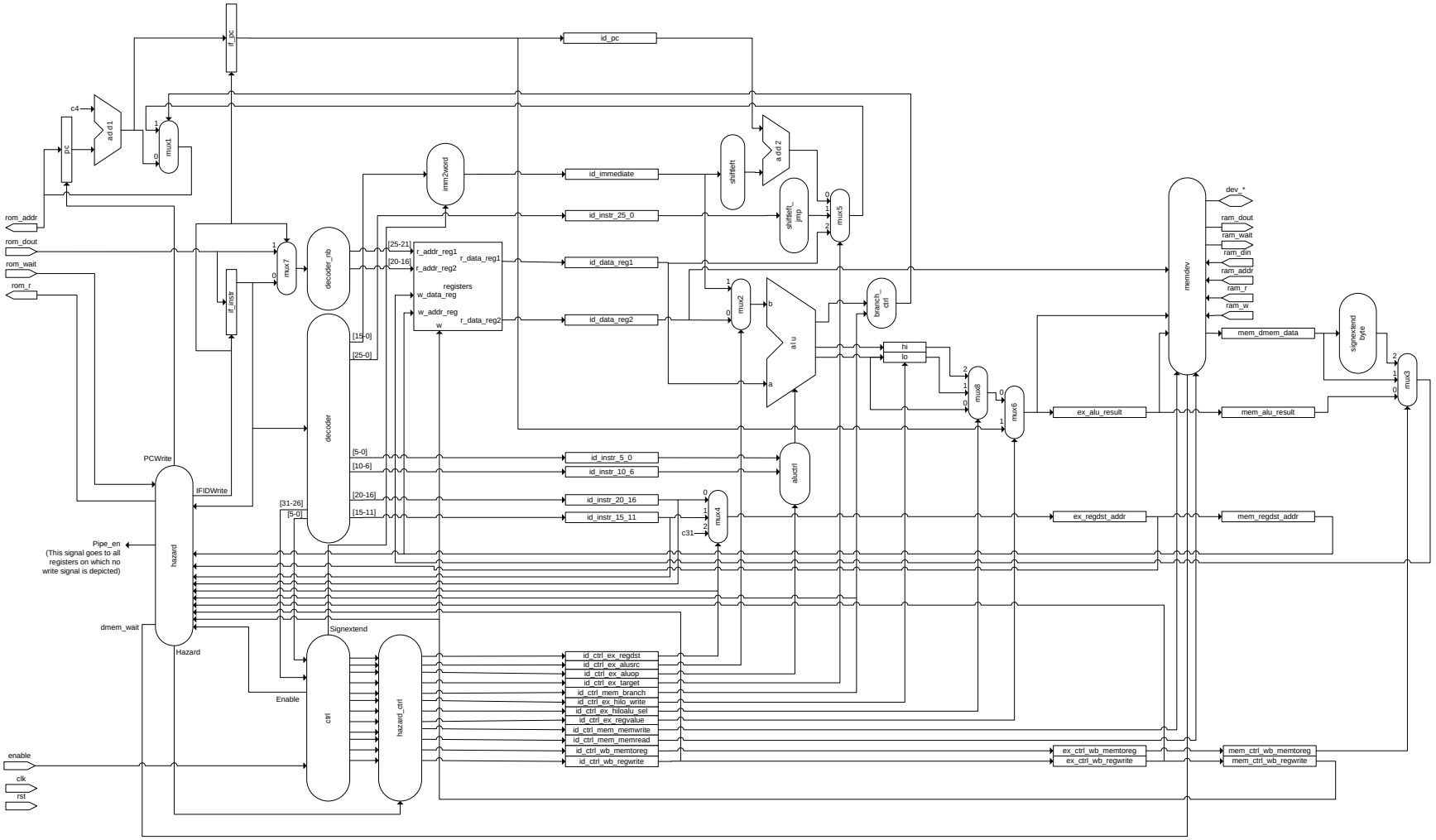
Synthetiseren van de pipelined mMIPS processor

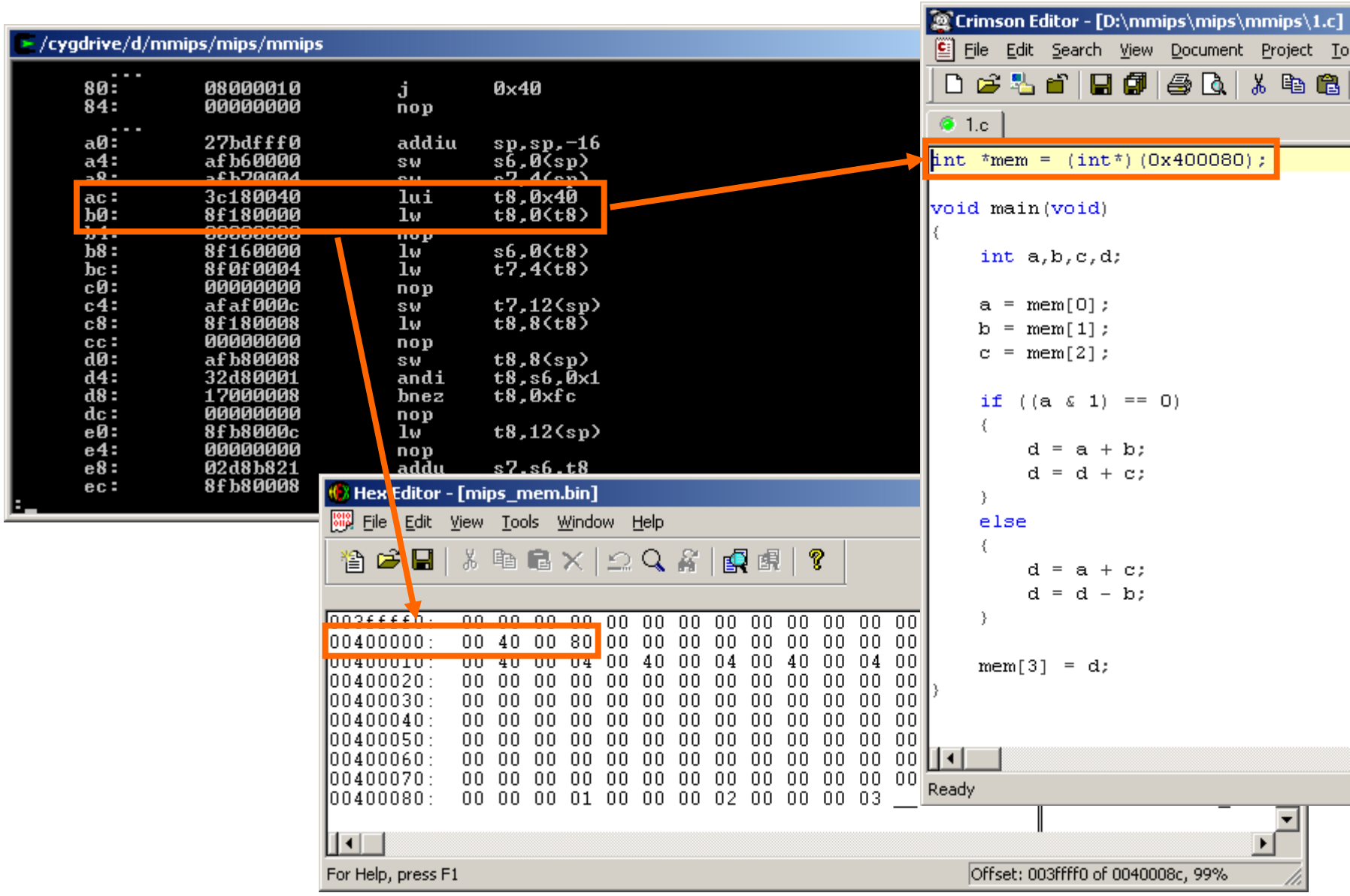
Sander Stuijk

Department of Electrical Engineering
Electronic Systems

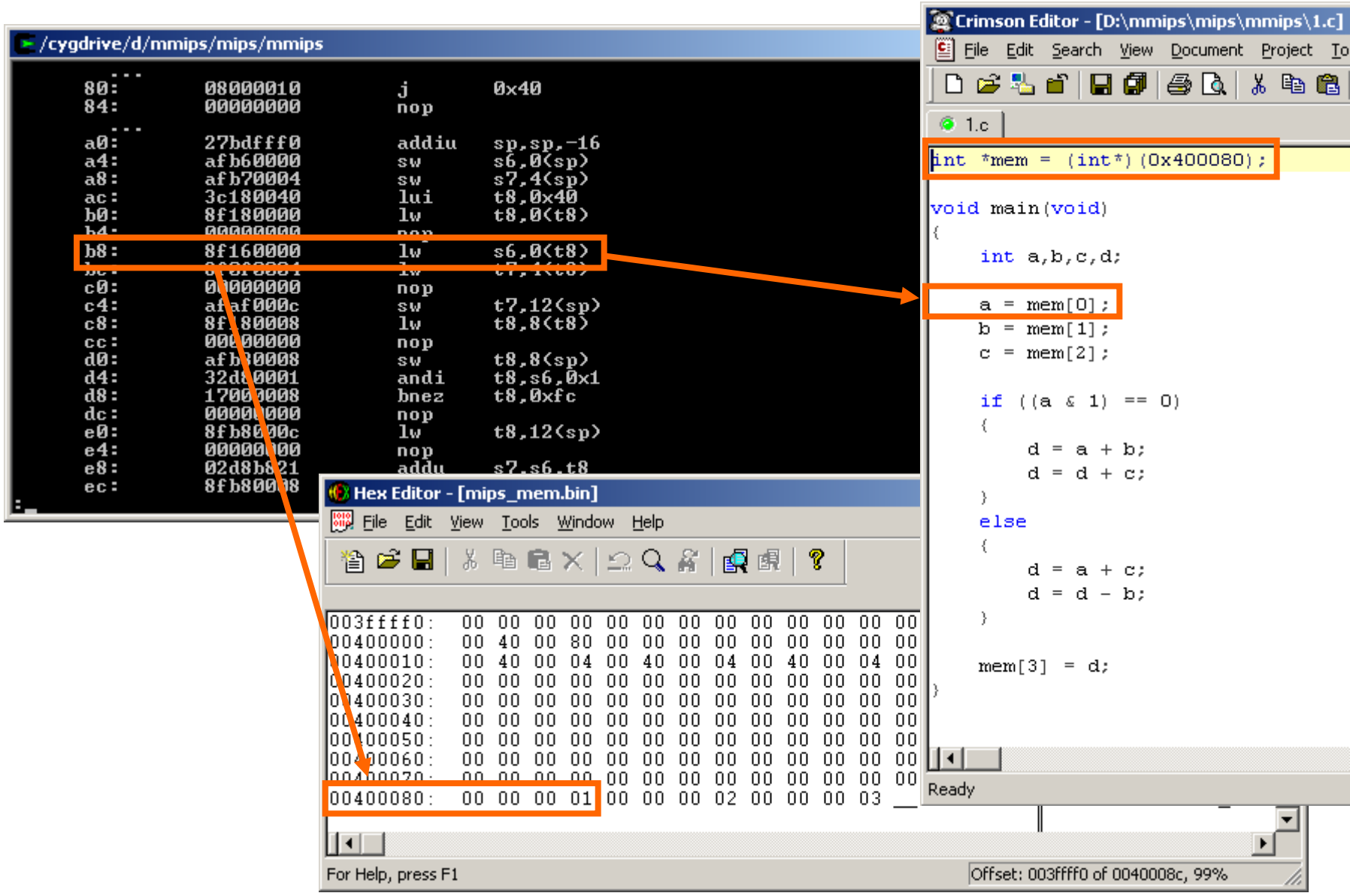
mMIPS


if_pc
id_pc
c4
pc
add
mux1
imm2word
id_immediate
shiftleft
add 2
shiftleft_jmp
mux5
rom_addr
rom_dout
rom_wait
rom_r
if_instr
mux7
decoder_rtb
[25-21]
[20-16]
r_addr_reg1
r_addr_reg2
registers
w_data_reg
w_addr_reg
w
r_data_reg1
r_data_reg2
id_instr_25_0
id_data_reg1
id_data_reg2
mux2
alu
branch_ctrl
hi
lo
mux8
mux6
memdev
dev_*
ram_dout
ram_wait
ram_din
ram_addr
ram_r
ram_w
mem_dmem_data
signextend byte
mux3
ex_alu_result
mem_alu_result
[15-0]
[25-0]
decoder
[5-0]
[10-6]
[20-16]
[15-11]
id_instr_5_0
id_instr_10_6
id_instr_20_16
id_instr_15_11
aluctrl
PCWrite
IFIDWrite
[31-26]
[5-0]
mux4
c31
ex_regdst_addr
mem_regdst_addr
Pipe_en
(This signal goes to all registers on which no write signal is depicted)
hazard
dmem_wait
Hazard
Signextend
Enable
ctrl
hazard_ctrl
id_ctrl_ex_regdst
id_ctrl_ex_alusrc
id_ctrl_ex_aluop
id_ctrl_ex_target
id_ctrl_mem_branch
id_ctrl_ex_hilo_write
id_ctrl_ex_hiloalu_sel
id_ctrl_ex_regvalue
id_ctrl_mem_memwrite
id_ctrl_mem_memread
id_ctrl_wb_memtoreg
id_ctrl_wb_regwrite
ex_ctrl_wb_memtoreg
ex_ctrl_wb_regwrite
mem_ctrl_wb_memtoreg
mem_ctrl_wb_regwrite
enable
clk
rst

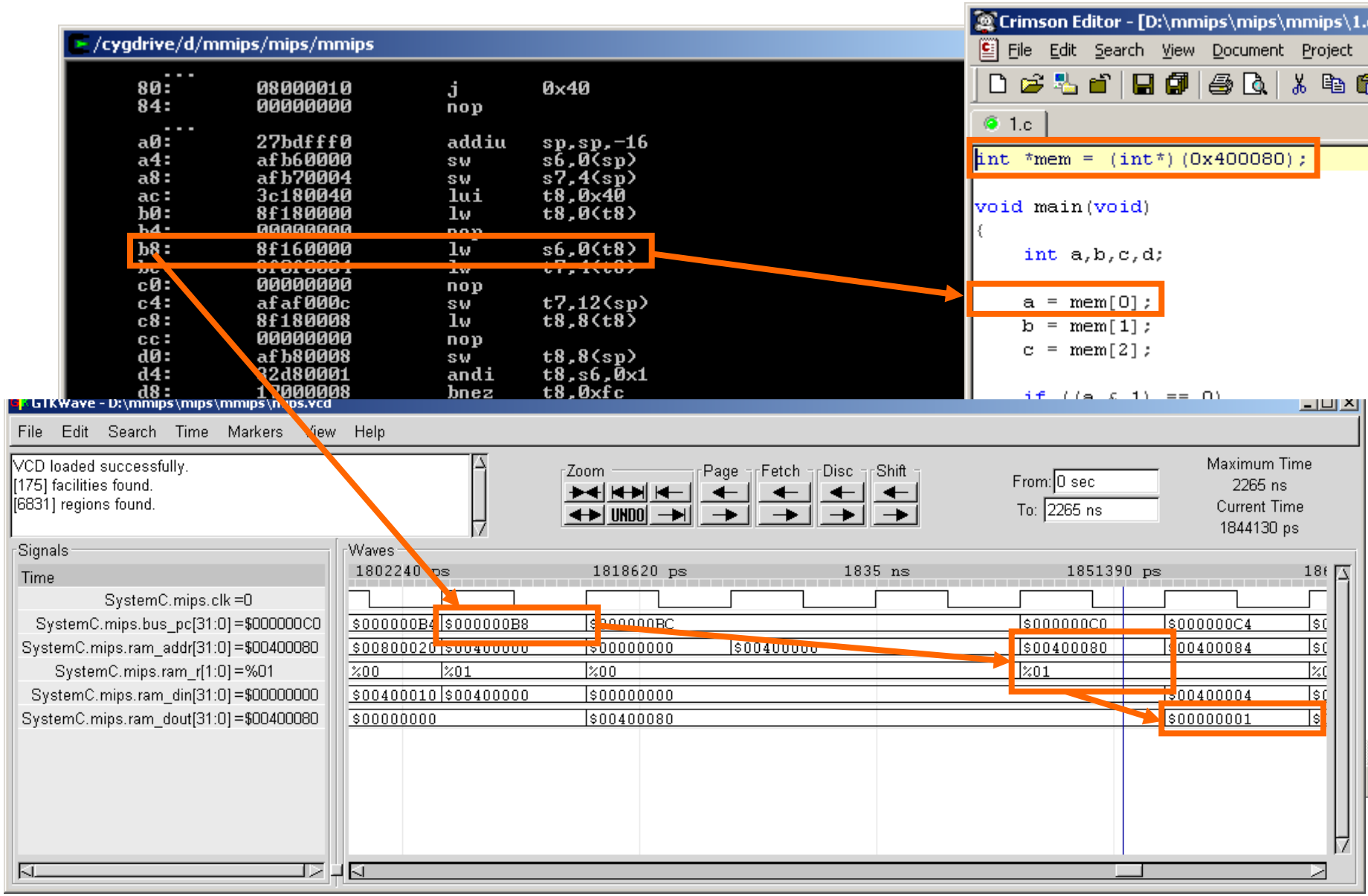
LCC compiler



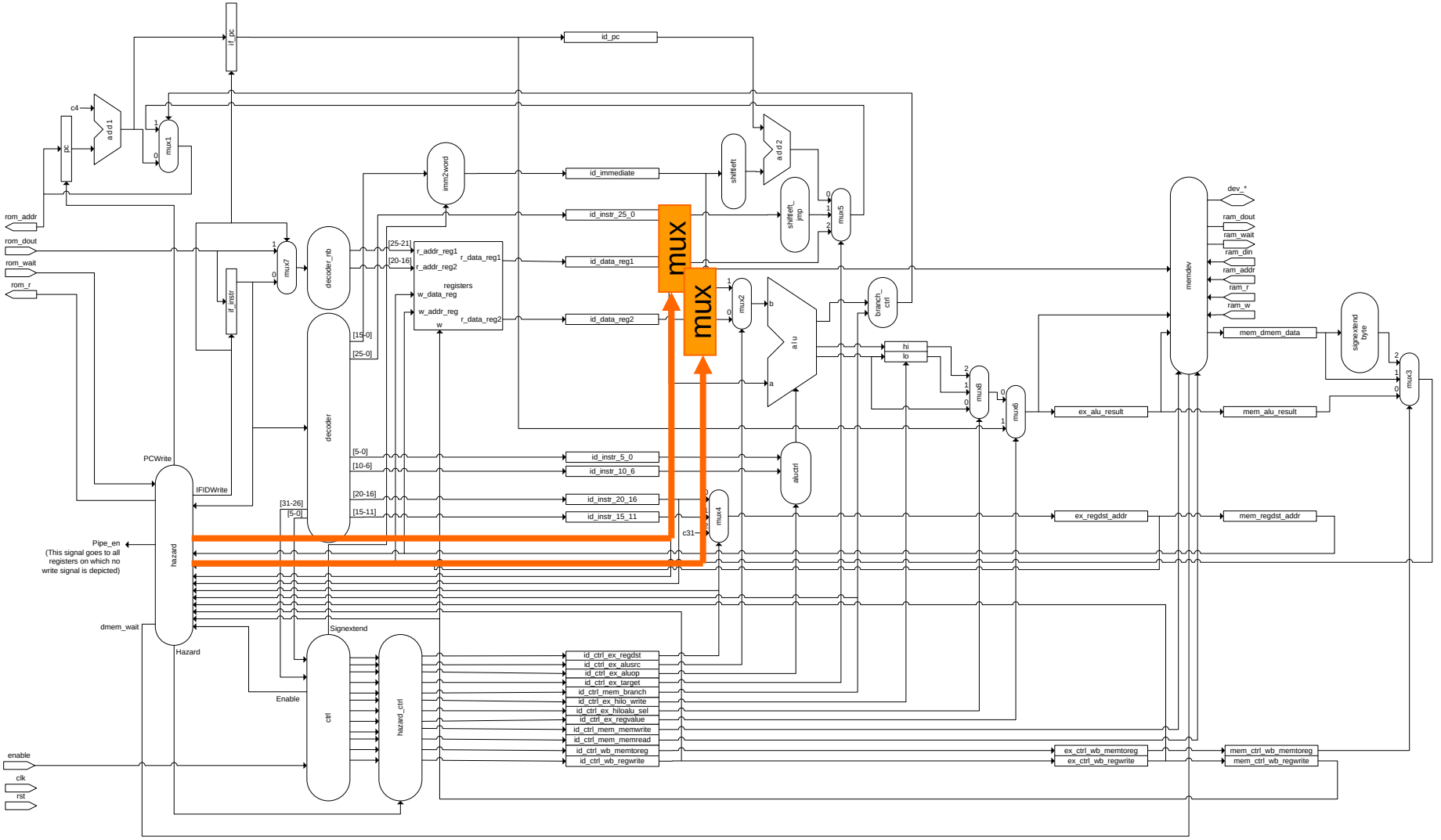
LCC compiler



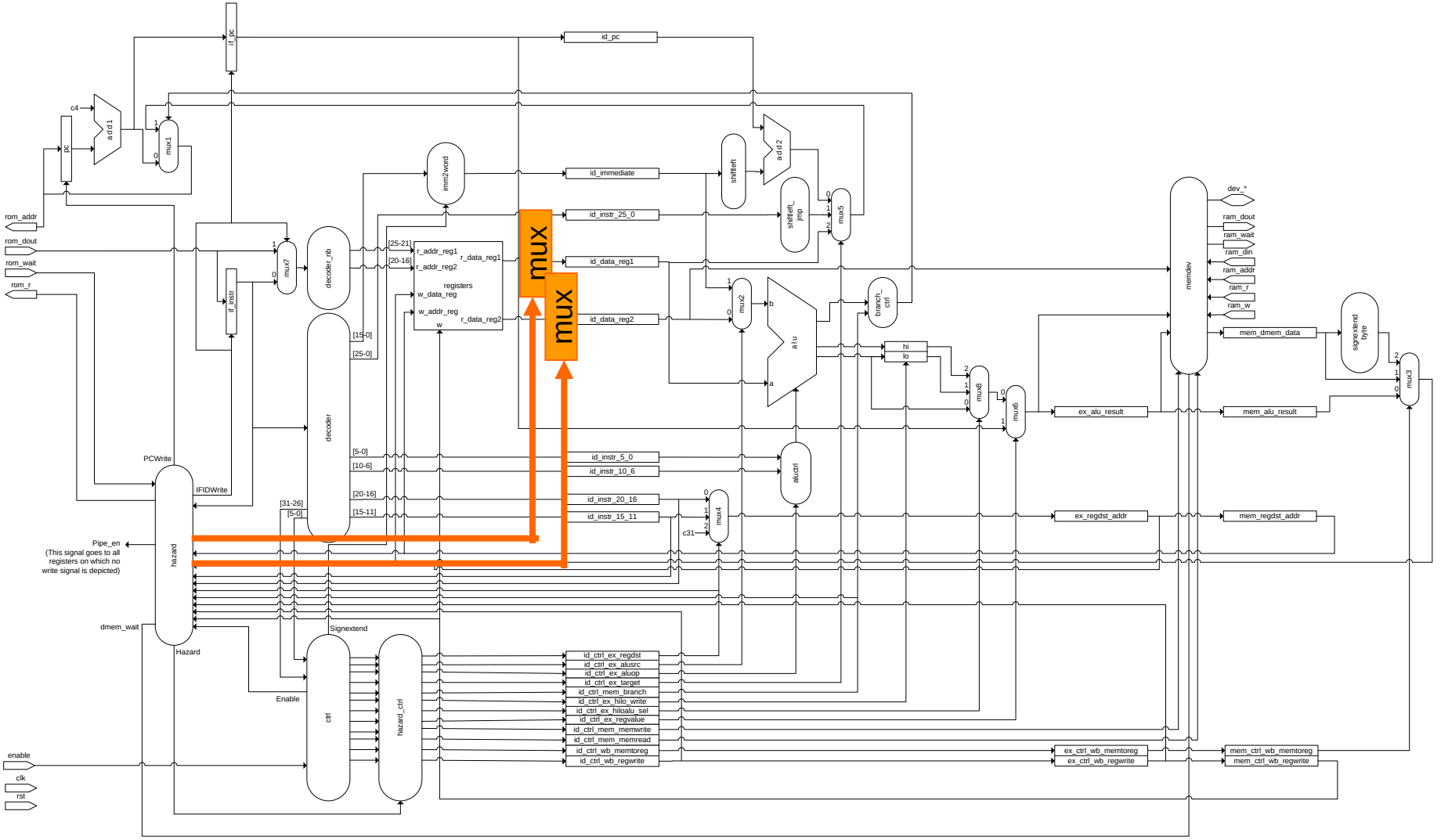
LCC compiler



Forwarding optie 1



Forwarding optie 2



Forwarding - aandachtspunten

- Register 0 heeft altijd de waarde 0, ongeacht wat er in dit register wordt geschreven
  - Forward dus nooit register 0
- Forward altijd de nieuwste data
  - EX gaat voor MEM, MEM gaat voor WB, WB gaat voor ID
- De twee register inputs kunnen uit dezelfde of verschillende pipeline stages komen
  - De twee forwarding multiplexers moeten los worden aangestuurd
- De lees operaties (lw, lb) hebben 1 delay slot, de data uit het geheugen is pas beschikbaar is in de WB stage
  - Er kan een data hazard zijn tussen een load instructie in the MEM stage en een instructie in de ID stage, deze kan niet worden opgelost met forwarding

Forwarding – WB stage

- Hazard detectie in WB stage

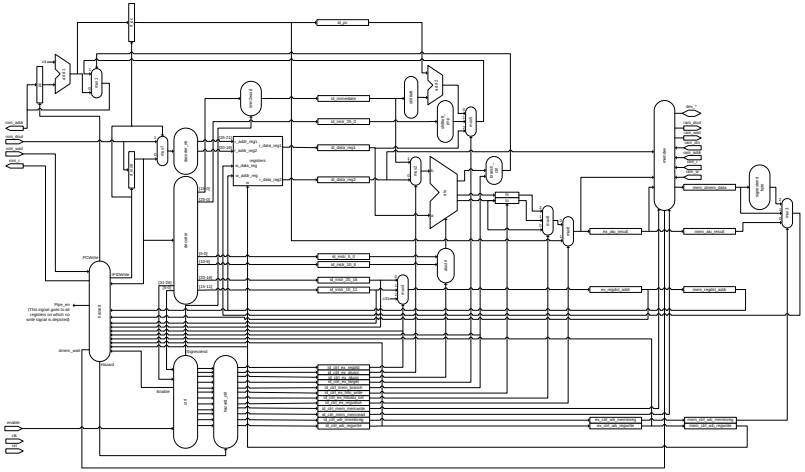
```
else if (memwbregwrite_t == 1 &&
                ((memwbwriteregister_t == ifidreadregister1_t) ||
                (memwbwriteregister_t == ifidreadregister2_t)))
{
   hazard = 1;
}
```

- Forwading vanuit WB stage

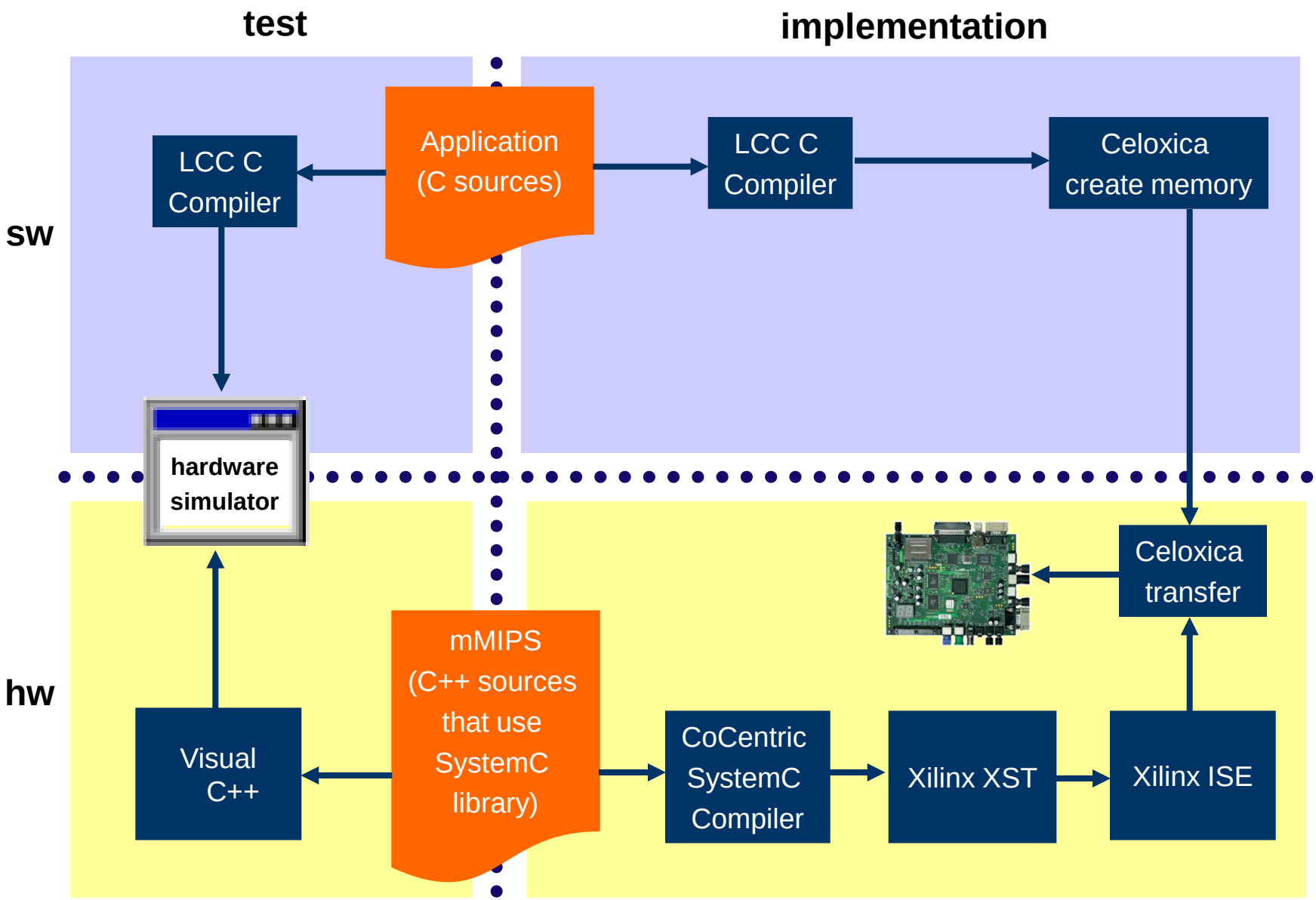
```
if (memwbregwrite_t == 1 && memwbwriteregister_t == ifidreadregister1_t
                         && memwbwriteregister_t != 0)
{
   forwardA.write(3);
}

if (memwbregwrite_t == 1 && memwbwriteregister_t == ifidreadregister2_t
                         && memwbwriteregister_t != 0)
{
   forwardB.write(3);
}
```

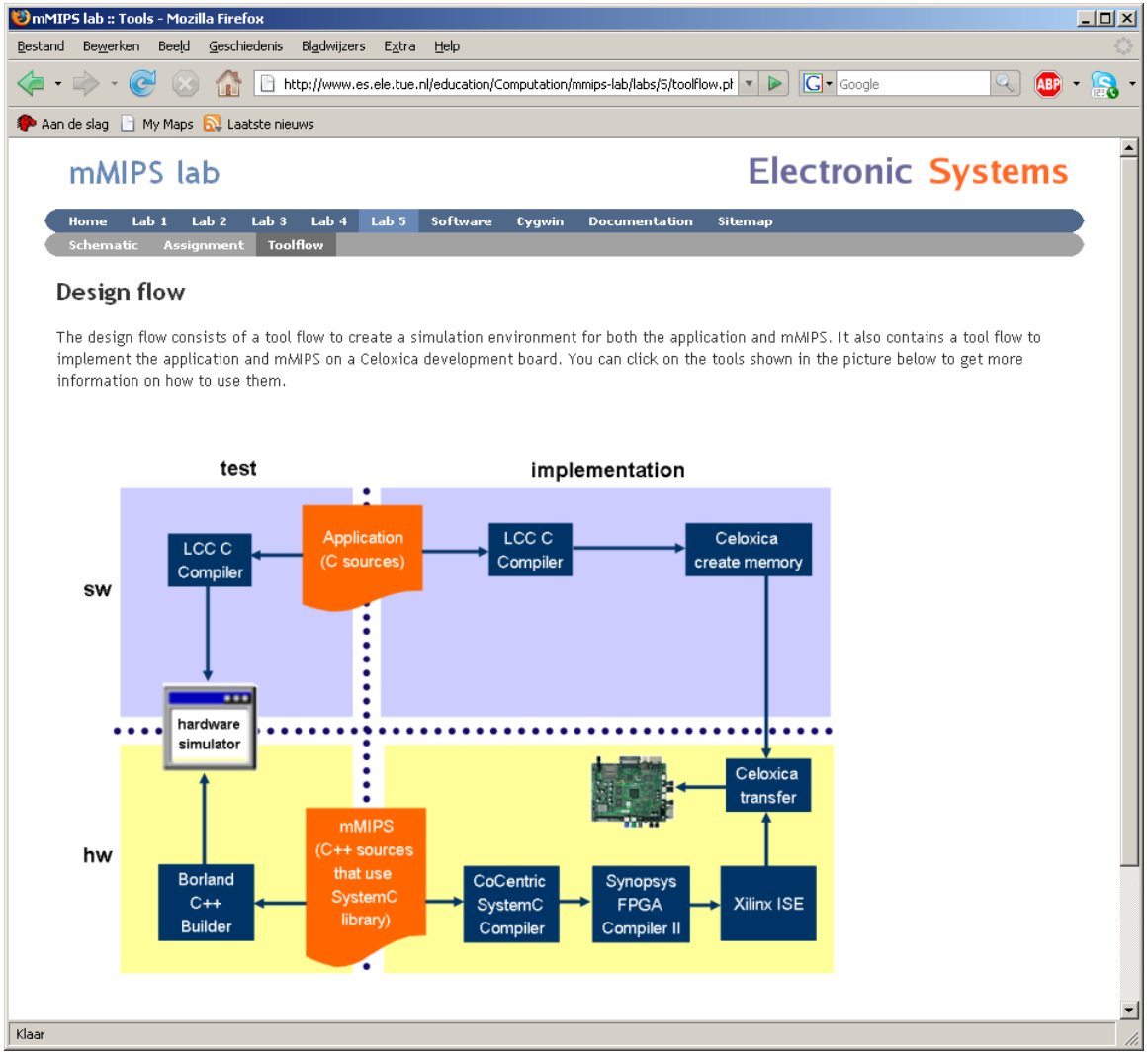
Lab 5 - Synthesis



Design flow


test
implementation
sw
hw
LCC C Compiler
Application (C sources)
LCC C Compiler
Celoxica create memory
hardware simulator
Celoxica transfer
mMIPS (C++ sources that use SystemC library)
Visual C++
CoCentric SystemC Compiler
Xilinx XST
Xilinx ISE

Design flow



www.es.ele.tue.nl/education/5JJ55-65/mmips-lab/labs/5/toolflow.php