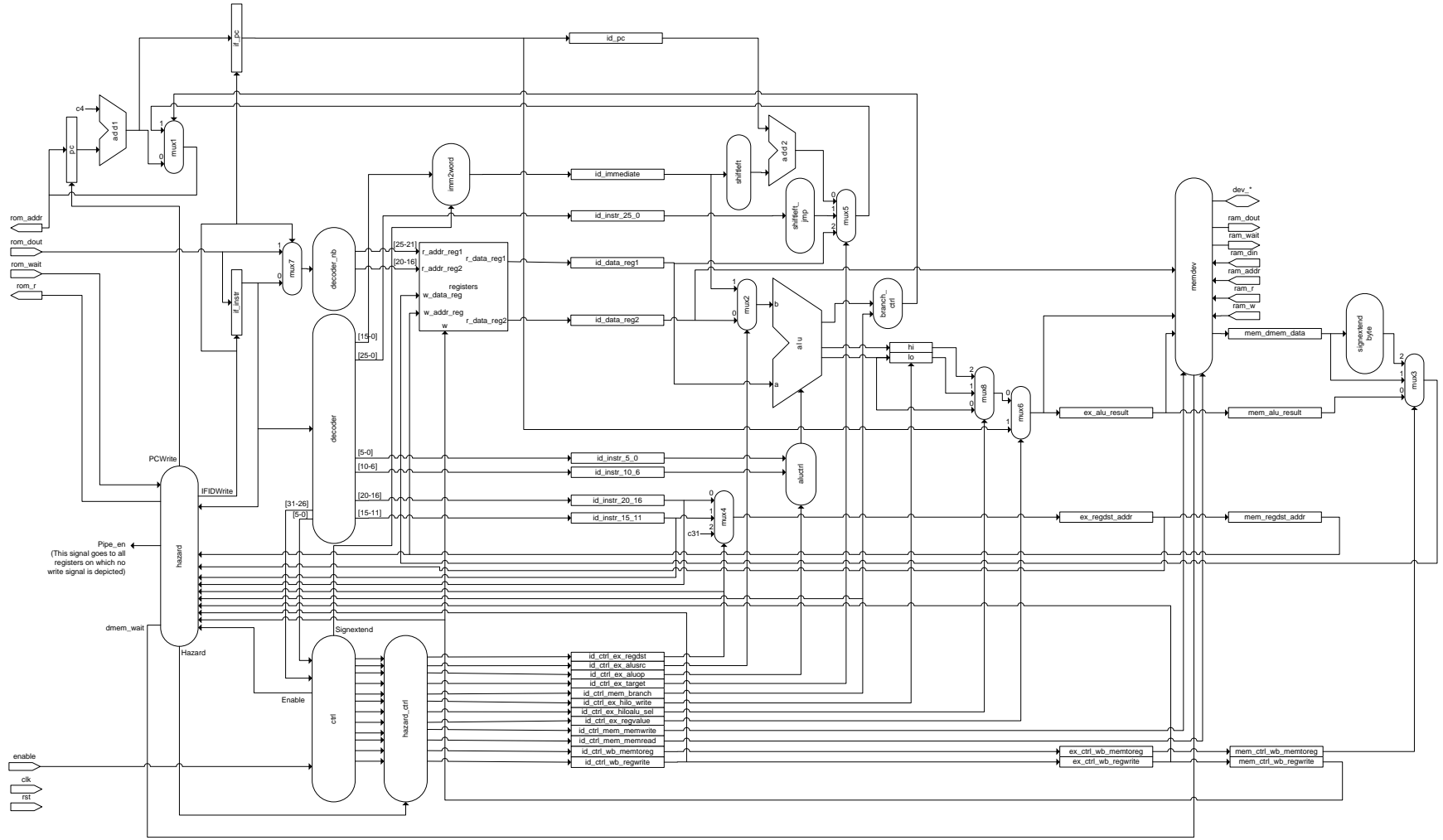


# Synthetiseren van de pipelined mMIPS processor

Sander Stuijk



The image displays three windows illustrating the compilation and memory layout of a program:

- Assembly Output (Top Left):** Shows MIPS assembly code. The instruction `lui t8, 0x40` is highlighted with an orange box. This instruction loads the immediate value `0x40` into register `t8`.
- C Source File (Top Right):** Shows the C code `int *mem = (int*) (0x400080);` highlighted with an orange box. This line declares a pointer to an integer at memory address `0x400080`.
- Memory Dump (Bottom):** Shows the memory dump in Hex Editor for `mips_mem.bin`. The address `00400080` is highlighted with an orange box, and the corresponding memory contents are `00 40 00 80`, which is the little-endian representation of the address `0x400080`.

The image displays three windows illustrating the compilation and execution of a program:

- Assembly Window (Top Left):** Shows MIPS assembly code. The instruction `lw s6, 0(t8)` at address `b8` is highlighted with an orange box. An arrow points from this instruction to the corresponding C code.
- C Source Code Window (Top Right):** Shows the C program `1.c`. The declaration `int *mem = (int*) (0x400080);` is highlighted with an orange box. The assignment `a = mem[0];` is also highlighted with an orange box. An arrow points from this line to the assembly instruction.
- Hex Editor Window (Bottom):** Shows the memory dump for `mips_mem.bin`. The address `00400080` is highlighted with an orange box, and its value is `00 00 00 01`. An arrow points from this memory location to the C code's pointer variable.

```

/cygdrive/d/mmips/mips/mmips
80: 08000010      j      0x40
84: 00000000      nop

a0: 27bdfbf0      addiu   sp,sp,-16
a4: afb60000      sw     s6,0(sp)
a8: afb70004      sw     s7,4(sp)
ac: 3c180040      lui    t8,0x40
b0: 8f180000      lw     t8,0(t8)
b4: 00000000      nop
b8: 8f160000      lw     s6,0(t8)
bc: 8f180004      lw     t7,4(t8)
c0: 00000000      nop
c4: afaf000c      sw     t7,12(sp)
c8: 8f180008      lw     t8,8(t8)
cc: 00000000      nop
d0: afb80008      sw     t8,8(sp)
d4: 32d10001      andi   t8,s6,0x1
d8: 17000008      bnez   t8,0xfc
dc: 00000000      nop
e0: 8fb8000c      lw     t8,12(sp)
e4: 00000000      nop
e8: 02d8b821      addu   s7,s6,t8
ec: 8fb80008
  
```

```

Crimson Editor - [D:\mmips\mips\mmips\1.c]
File Edit Search View Document Project To
1.c
int *mem = (int*) (0x400080);

void main(void)
{
    int a,b,c,d;

    a = mem[0];
    b = mem[1];
    c = mem[2];

    if ((a & 1) == 0)
    {
        d = a + b;
        d = d + c;
    }
    else
    {
        d = a + c;
        d = d - b;
    }

    mem[3] = d;
}
  
```

```

Hex Editor - [mips_mem.bin]
File Edit View Tools Window Help
003ffff0: 00 00 00 00 00 00 00 00 00 00 00 00
00400000: 00 40 00 80 00 00 00 00 00 00 00 00
00400010: 00 40 00 04 00 40 00 04 00 40 00 04
00400020: 00 00 00 00 00 00 00 00 00 00 00 00
00400030: 00 00 00 00 00 00 00 00 00 00 00 00
00400040: 00 00 00 00 00 00 00 00 00 00 00 00
00400050: 00 00 00 00 00 00 00 00 00 00 00 00
00400060: 00 00 00 00 00 00 00 00 00 00 00 00
00400070: 00 00 00 00 00 00 00 00 00 00 00 00
00400080: 00 00 00 01 00 00 00 02 00 00 00 03
  
```

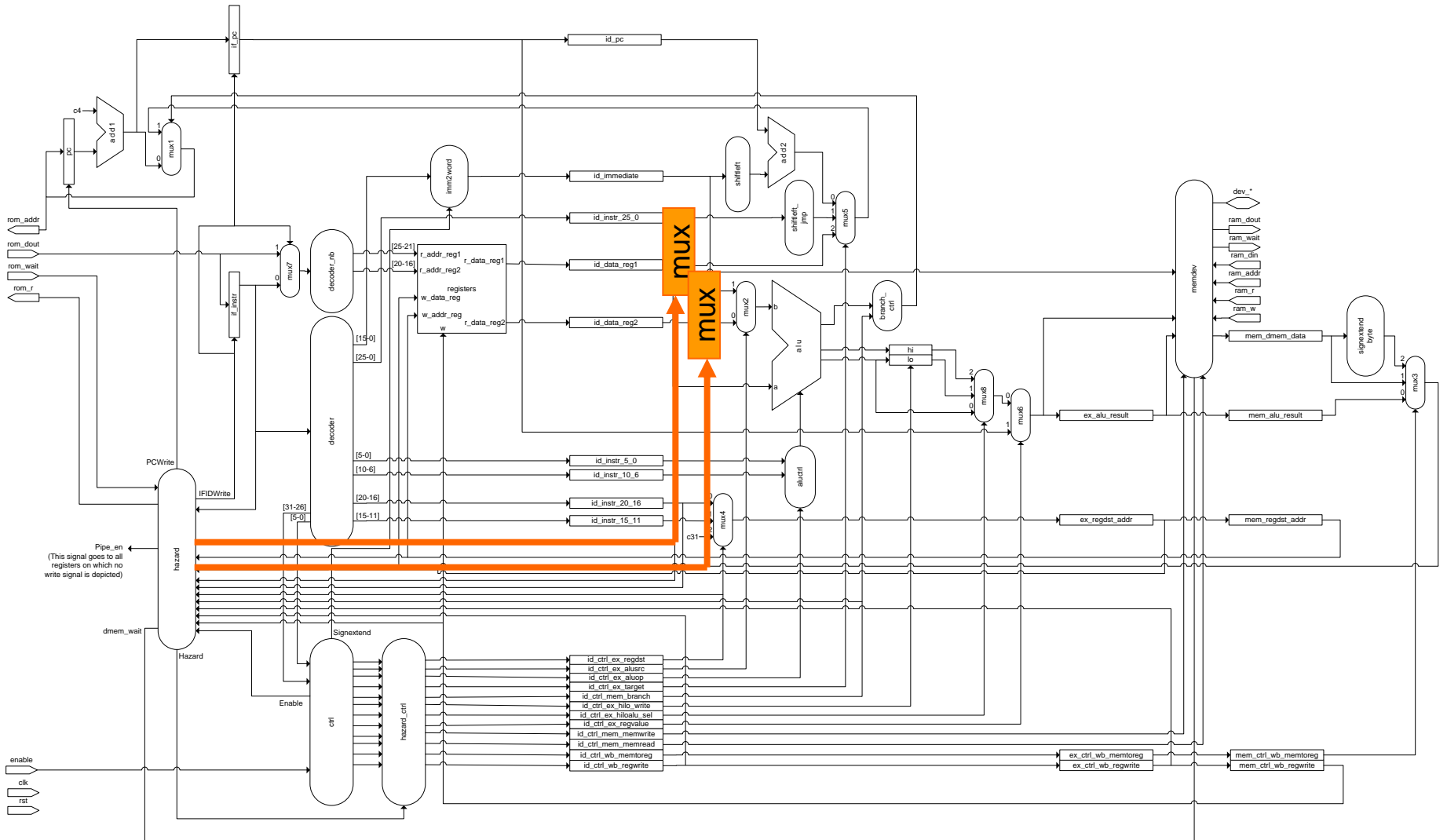
For Help, press F1

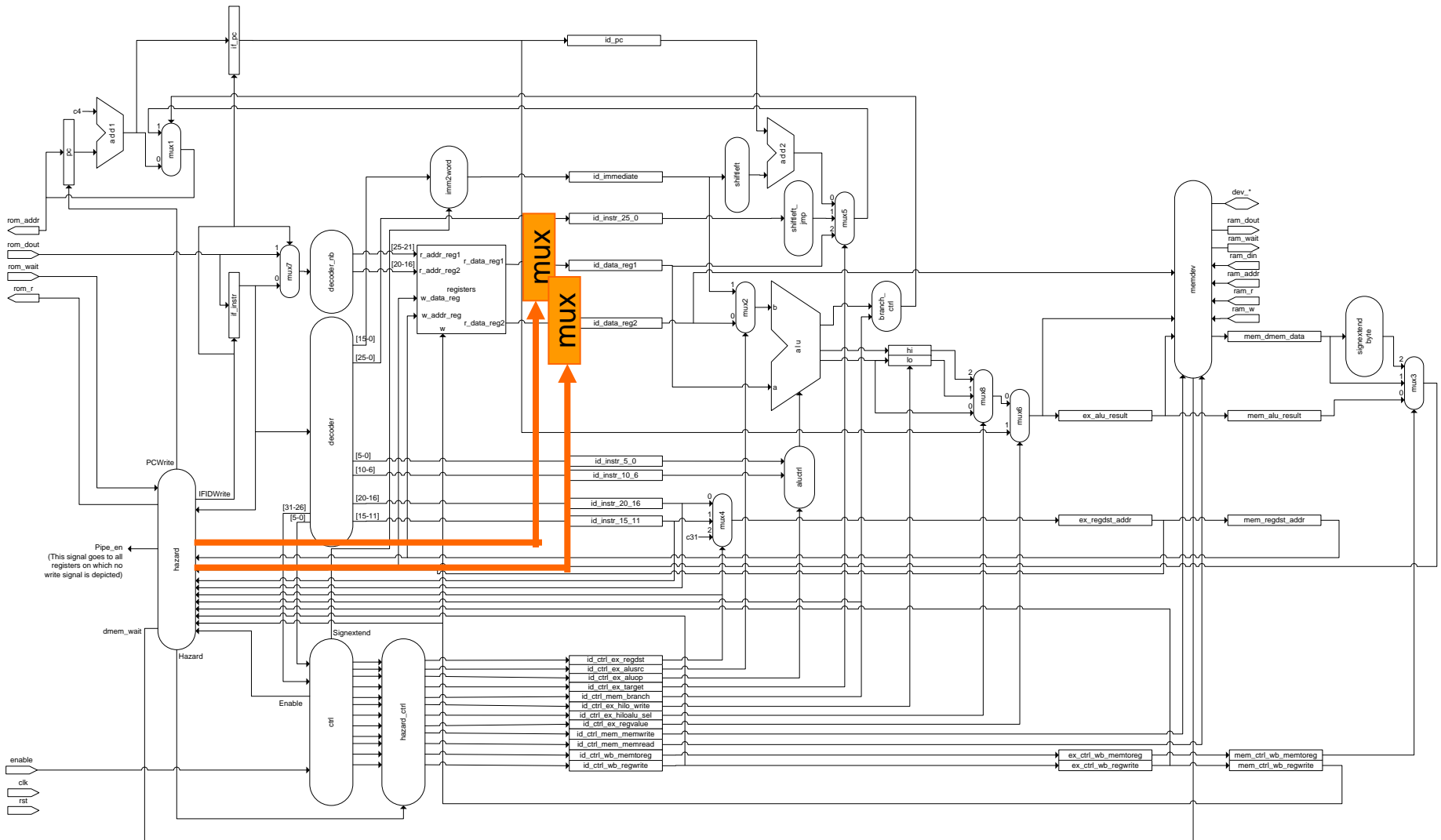
Offset: 003ffff0 of 0040008c, 99%

The image displays three windows illustrating the execution of assembly code generated by the LCC compiler:

- Assembly Window (Top Left):** Shows assembly instructions for MIPS architecture. An orange box highlights the instruction `b8: 8f160000 lw s6,0(t8)`, which is linked by an arrow to the C code line `a = mem[0];`.
- C Code Window (Top Right):** Shows the source code in `1.c`. The line `int *mem = (int*) (0x400080);` is highlighted in yellow. The assignment `a = mem[0];` is also highlighted with an orange box.
- Logic Analyzer Window (Bottom):** Shows a timing diagram for signals `SystemC.mips.bus_pc`, `SystemC.mips.ram_addr`, and `SystemC.mips.ram_din`. The waveform shows the address `0x400080` being accessed, with an orange box highlighting the data value `0x00000001` returned from memory, which is linked by an arrow to the assembly instruction `b8: 8f160000 lw s6,0(t8)`.

# Forwarding optie 1





- 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



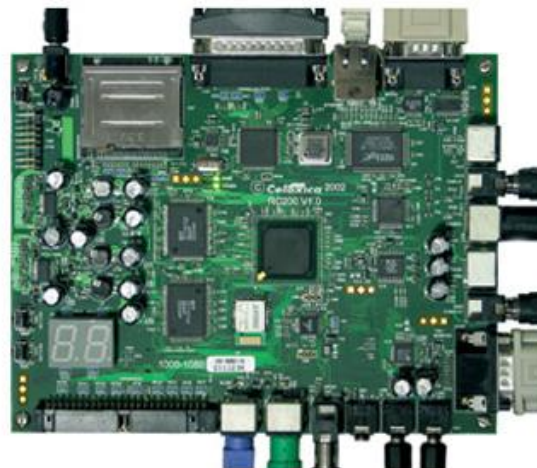
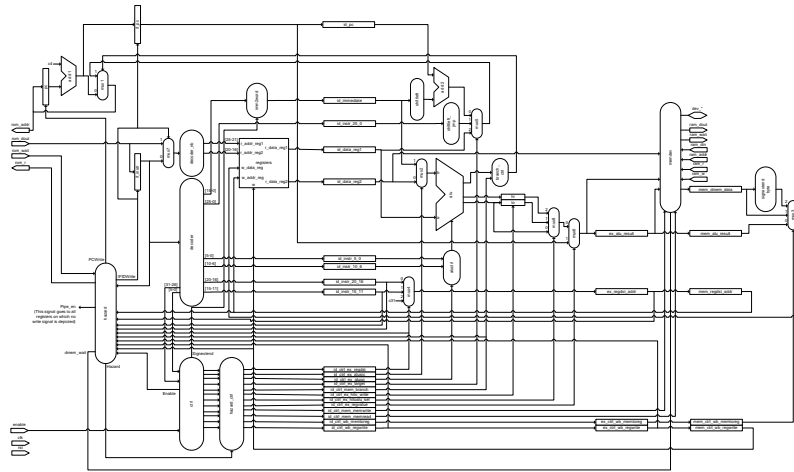
- Hazard detectie in WB stage

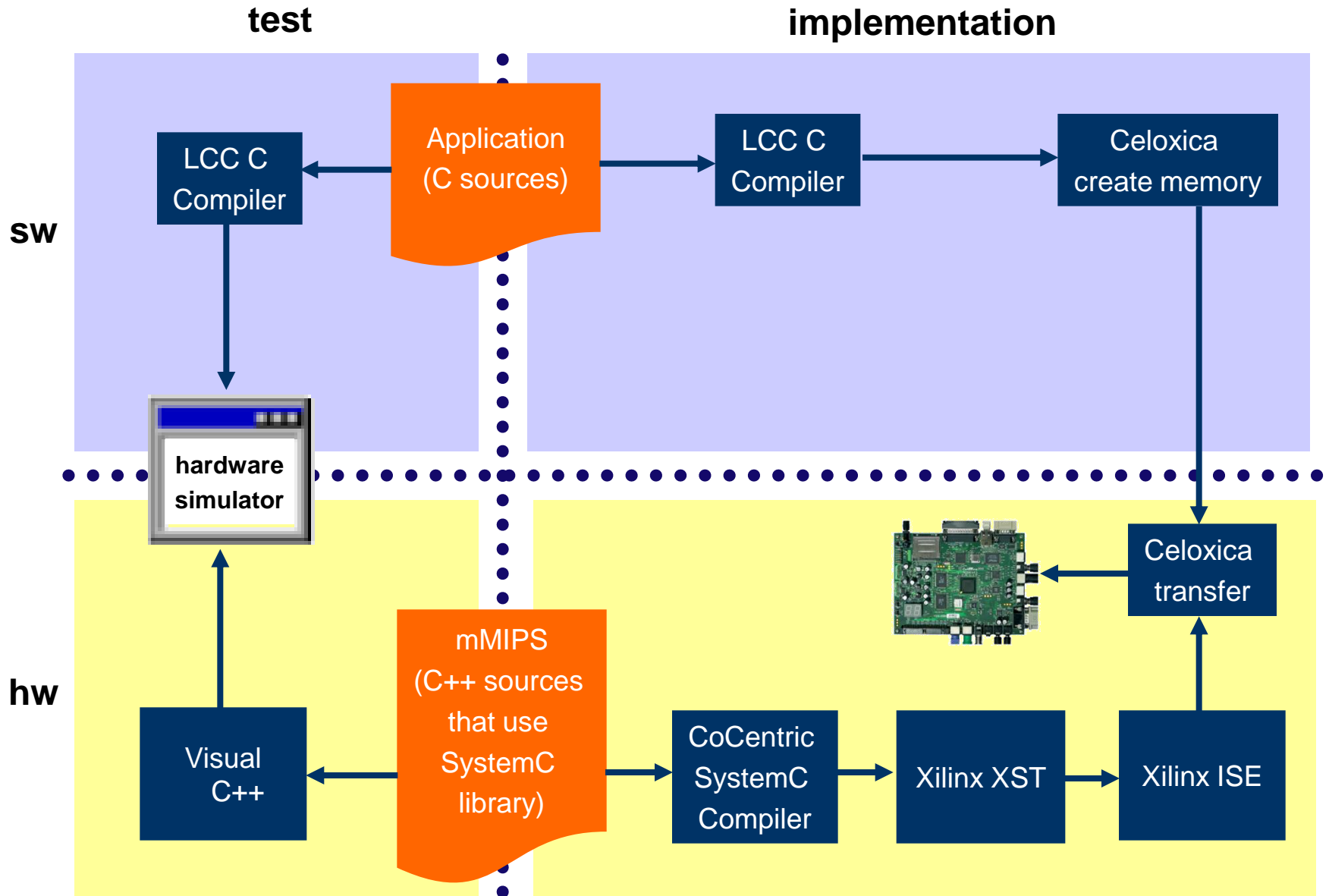
```
else if (memwbregwrite_t == 1 &&
        ((memwbwriteregister_t == ifidreadregister1_t) ||
         (memwbwriteregister_t == ifidreadregister2_t)))
{
    hazard = 1;
}
```

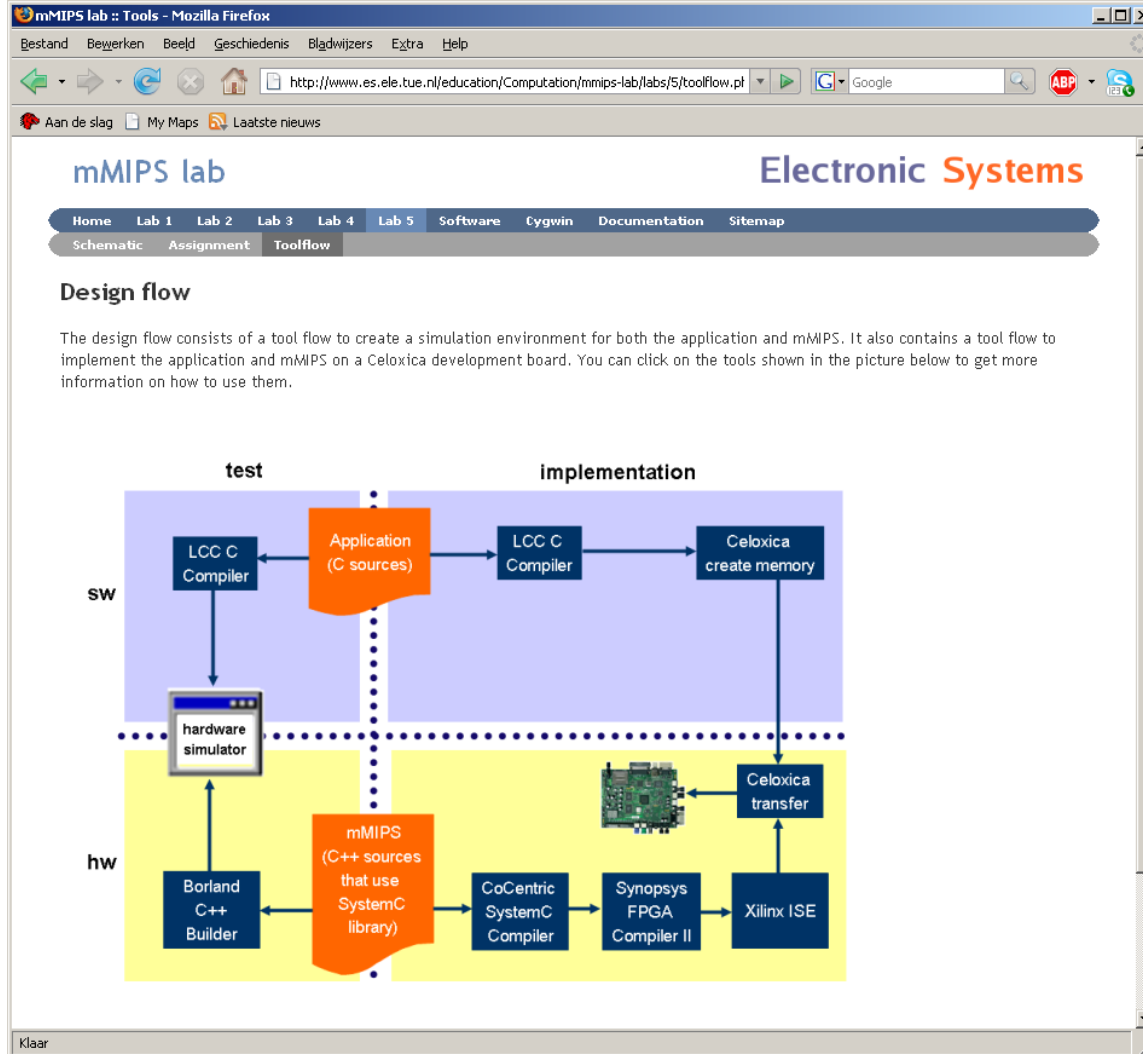
- Forwarding vanuit WB stage

```
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);
}
```







[www.es.ele.tue.nl/education/5JJ55-65/mmips-lab/labs/5/toolflow.php](http://www.es.ele.tue.nl/education/5JJ55-65/mmips-lab/labs/5/toolflow.php)