# RA S4



1.

Sign extention is necessary to preserve the number's sign.

→ Adress-offsets can be positive or negative

and immediates are only 16 bit long

=> sign extend immediate to 32 bits

2.

| | $0_x$ | | |
|---|---|---|---|
| $0x0 \&\& 0x\bar{F}\bar{F}$ | $(0x0)_{16} = (0)_{10} = (0000)_2$ | $(0x\bar{F}\bar{F})_{16} = (1111\,1111)_2$ | $0 \&\& 239 = 0$ false |
| $0xD3 \& 0x5B$ | $(0xD3)_{16} = (1101\,0011)$ | $(0x5B)_{16} = (0101\,1011)_2$ | $= (0101\,0011)_2$ |
| $0x0 \mid\mid 0x\bar{F}\bar{F}$ | $(0x0)_{16} = (0)_{10}$ | $(0x\bar{F}\bar{F})_{16} = (1111\,1111)$ | $= 1$ true |
| $0xA3 \mid 0x3A$ | $(0xA3)_{16} = (1010\,0011)_2$ | $(0x3A)_{16} = (0011\,1010)_2$ | $= (1011\,1011)_2 = (0xBB)_{16}$ |
| $!0x\,\bar{F}\bar{E}$ | $(0x\bar{F}\bar{E}) = (1111\,1110)_2$ | – | 0 (false) |
| $\sim 0x\bar{F}\bar{E}$ | $(0x\bar{F}\bar{E})_{16} = (1111\,1110)_2$ | – | 1 (true) |

3.

```
1.  1:  $t0, 1
2.
3.  loop:
4.  jal  foo                 # (call procedure) jump and link foo
5.  addi $t0, $t0, -1        # $t0 = 9 + (-1) = 8
6.  bne  $t0, 0, loop        # ($t0 != 0) -> loop
7   j end
8.  foo
9.  li  $t1, 0xAFFFFF04
10. sw $t0$, 0($t1)          # array[0] = $t0  (save $t0 on stack)
11  jr $ra                   # return to caller
12  end
```

Das Problem ist, auch wenn der loop zu ende ist springen wir immer wieder zur Zeile 11 -> so terminiert das Programm nie. => Einfügen von j end.

4.

beq : $s1, $2, Label | if($s1 == $s2) go to Label

bne : $s1, $s2, Label | if($s1 != $s2) go to Label

a) single cycle

invert the flag (if $s1 == $s2) -> (if $s1 != $s2)

b) invert the two-Output (as above)

5.

Registers are used to safe the result of an Operation.

These temporary results are used in the next cycle step as inputs for further calculations.

The registers are used to buffer those temporary results.

6.

Control hazard:

The pipeline processor does not know what instruction to fetch next because

e.g. the branch decision has not been made by the time the next instruction is fetched.

Data hazard:

Occur if an instruction reads a register that a previous instruction overwrites in a further cycle

Structural hazards:

Occurs if two (or more) instructions that are already in pipeline need the same resource.

7.

Registers are written to in the first half and read from in the second half of 1 cycle

Slide 15:

The register of sub is readable in the second half of the cycle

Slide 15:

lw needs to wait until beq is finished to check if it should be executed.

8.

1. add $t0 , $t5 , $t4
2. lw $s2 , 0($t0) → load falsche Daten (forwarding)
3. sub $t5 , $t0 , $s2 → kann nicht load bevor sub $s2 braucht (stall)
4. sw $t4 , 4($s3) → (forwarding) → resultat nicht ready von sub für sw ($s3)