Exercises

1. Consider the datapath in Figure 1 and the program code in Listning 1 on the last page. Assume that the ALU corresponds to the one presented in Lecture 9, with the extension of a zero signal output. When signal *Inst* corresponds to the machine code for the instructions on code lines 7, 8, and 11, what are then the values of the 8 control signals *Jump*, *RegWrite*, *RegDst*, *ALUSrc*, *ALUControl*, *Branch*, *MemWrite*, and *MemToReg*, for each of these instructions? Answer with either the concrete signal value or with a question mark (?) when it does not matter what the signal value is.

   Your solution:

	Jump	RegWrite	RegDst	ALUSrc	ALUControl	Branch	MemWrite	MemToReg
and $s0,$s0,$s1	0	1	1	0	000	0	0	0
sw $s0,0($a1)	0	0	?	1	010	0	1	?
j loop	1	0	?	?	?	?	0	?

2. Consider again the datapath in Figure 1 and the program code in Listning 1 on the last page. For each of the code lines 3 and 4, when *Inst* has the corresponding machine code value of these code lines, state the values of the eight signals marked as black filled circles. If the signal has a uniquely defined value, (independent of the loop iteration) state that value as a hexadecimal number. If the value is always located in a specific register, (e.g., `$t0`) then state the register name (e.g., `$t0`). Otherwise, state that the signal value is unknown.

   Your solution:

	1	2	3	4	5	6	7	8
slt $t0, $t1, $a0	?	$a0	0x402A	$t1	$t0	$a0	$t0	$0
beq $t0, $t0, done	?	$0	?	$t0	?	$0	?	?

3. Assume that a processor with a 5-stage pipeline executes at 80 MHz and that the processor can fetch at most 1 instruction in each clock cycle.

   (a) How long is then the clock period?

   (b) Which of the following alternatives is correct: i) The CPI is always larger or equal to 1, ii) the CPI is always smaller than 1, or iii) the CPI is always equal to 1.

   Your solution:

A) $t_c = \frac{1}{80\cdot 10^6} = \frac{1}{8}\cdot 10^{-7} = 0{,}125\cdot 10^{7} = 12{,}5\cdot 10^{-9} = 12{,}5 ns$

B) nummer 1.

4. Assume that we have a MIPS processor with a 5-stage pipeline. Consider the MIPS assembly code below. Explain which instructions that cause a hazard and what kind of hazards these are. State how the hazards can be solved for the different cases. You should suggest the best possible solution, that is, that results in that the code takes as few clock cycles as possible to execute.

```
        lui     $s0,0x1001
        addi    $t0,$0,10
        sw      $t0,12($s0)
        lw      $t1,8($s0)
        and     $t0,$t0,$t1
```

   Your solution:

SW : forwarding

and : forwarding + Stall

5. A bne instruction can result in a specific kind of hazard.

   (a) What is this kind of hazard called?

   (b) How can such hazard be solved?

   (c) For deep pipelines, such hazards can have very negative impact on the performance. Explain why.

   (d) Which technique is used in modern processors to remedy this negative impact? Motivate why this is the case.

Your solution:

a) control Hazard B) Stall

c) Eftersom vi stallar efter varje branch får vi flera extra cykel fördröjningar som inte gör någon nytta

d) Branch prediction, funkar genom att försöka förutse hur flödet kommer förgrenas. Detta gör att programet "gissar" vart flödet ska fortsätta.

6. State 6 differences or similarities between the ISAs ARM v7, x86, and MIPS.

Your solution:

1. ARM och Mips anropar return addresser i ett register medan x86 anropar return adresser från stacken iminnet.

2. Mips och ARM är byggd på Risk medan x86 är baserat på cisc.

3. x86 är standarden inom pc och laptops medans Mips och ARM används för embded devices

4. x86 har artimeriska operationer där destination är minnet.

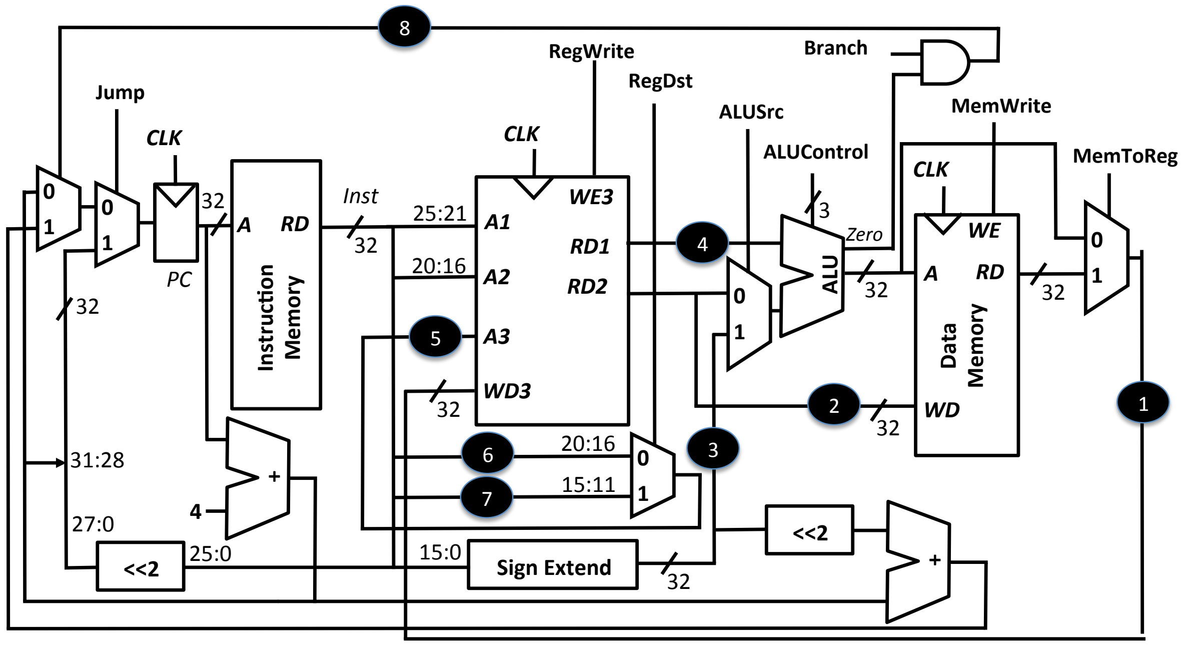
5. x86 har till skillnad från de andra support för 80-bit floating point

6. ARM och mips har en instruktion storlek på 32bits medans x86 varierar mellan 1 till 18 bytes

Corrected by ___________________________. Total number of points: _________

The following figures are used in the seminar exercises above. You need these figures to solve the exercises, but you do not have to print these figures and bring them to the seminar.

Figure 1.



Listning 1.

```
        addi    $t1,$0,0
loop:
        slt     $t0,$t1,$a0
        beq     $t0,$0,done
        lw      $s0,4($a1)
        lw      $s1,8($a1)
        and     $s0,$s0,$s1
        sw      $s0,0($a1)
        addi    $a1,$a1,-12
        addi    $t1,$t1,1
        j       loop
done:
```