Computer Architecture and Engineering - 3rd Course. **1st partial exam** **11/6/2017**

Answer each question in a different answer sheet. Available time: 3h

1. **(2.5 points)** A design project aims at improving the performance of a mobile device that has a Qualcomm processor at 1.4GHz and a Mali-G71 MP8 GPU, with a production cost of 300 €. The designers want to evaluate different alternatives from the cost/performance point of view.

   In order to evaluate the alternatives, a test software will be used that takes 10.5 seconds to execute on the original device. A profiler indicates that such a software uses 55% of the time executing CPU operations. The rest of the time is consumed by operations on the GPU and other devices.

   After replacing the GPU by a Mali-G71 MP20, which offers an improvement of 150% over the original GPU, the test software takes 8.9 seconds to execute. The cost increase of this new GPU is 50€.

   Answer the following questions:

   (a) What is the overall speedup obtained when replacing the GPU?

   (b) In the original device, what fraction of the execution time did the test software devote to operations with the GPU? And to operations with other devices?

   (c) If the processor of the original device is replaced by one with the same architecture, but running at 2.0 GHz and with a cost increase of 70€. What global speedup will be obtained?

   (d) What global speedup will be obtained if we simultaneously apply the changes proposed in the two previous sections to the original device? Consider that this change increases the cost by 100€.

   (e) Finally, from the cost/performance point of view, what alternatives are interesting: replace the GPU, the processor, or both of them?

2. **(2.5 points)** On an Infineon ARM Cortex M4 microcontroller with a clock frequency of 200 MHz and with a load/store instruction set, several concurrent tasks are executed. These tasks share resources and, in order to synchronize access to them, *test-and-set* operations are employed, in which atomicity is achieved by disabling interrupts.

```
                    # test-and-set on a
                    # memory location: lock
                    #
di                  # disable interrupts
addi r2,r0,#1       # r2 <- 1
lb r1,lock(r0)      # r1 <- value read from lock
sb r2,lock(r0)      # lock <- 1
ei                  # enable interrupts
```

   After studying the load generated by the tasks, the following distribution of operations is obtained:

Operation	%	CPI
ALU	36	1
Load	24	2
Store	12	2
Branches	18	1.5
di	4	1
ei	6	1

   The engineers are considering introducing a modification in the architecture so that the *test-and-set* operation could be implemented in a single instruction. This way, all the previous code would be replaced with:

```
TaS r1,lock(r0) # load the value in lock location into r1
                # and then set that location to 1
```

   Such instruction would have a CPI of 3. The clock frequency does not change.

(a) Compute the CPI of the original processor, as well as the execution time in seconds of a task consisting of n instructions.

(b) Knowing that in the original processor 50% of the *di* instructions are used in *test-and-set* operations, compute the new distribution of instructions in the modified processor.

(c) On the modified processor, compute the CPI as well as the number of instructions for a task that had n instructions in the original processor.

(d) Compute the speedup achieved by the new architecture.

3. **(2.5 points)** There is a MIPS processor in which the following loop is executed:

```
loop:   l.d f1, 0(r1)
        l.d f2, 0(r2)
        add.d f4, f1, f7
        add.d f5, f2, f8
        div.d f7, f6, f1
        div.d f8, f6, f2
        mul.d f5, f4, f5
        sub.d f5, f5, f10
        s.d f5, 0(r3)
        daddi r1, r1, 8
        daddi r2, r2, 8
        daddi r3, r3, 8
        bne r1, r5, loop
        trap 0            # when executed, it ends the program
        <next1>
        <next2>
        <next3>
```

The processor has the following floating-point multi-cycle operators:

- Add/Subtract. Lat= 2, IR= 1, stages A1, A2.
- Multiplier. Lat= 3, IR= 1, stages M1, M2, M3.
- Divider. Lat= 5, IR=$\frac{1}{5}$, stages D1, D2, D3, D4, D5.

The structural and data hazards are detected at the ID stage, inserting as many stall cycles as necessary and, in the case of data hazards, using shortcircuits whenever possible.

The control hazards are solved by means of the predict-not-taken technique. The computation of the branch condition, the branch target address, and the PC update are made at the stage `ID`.

Taking into account that the integer and memory instructions use the classic 5-stage pipeline: `IF, ID, EX, ME`, and `WB`, while multi-cycle instructions use `IF, ID,` $<$ execution in the corresponding multi-cycle operator $>$, and `WB`, and that there are two register files (1 for floating point and 1 for integers) with 2 read ports and 1 write port per file:

(a) The instruction-time diagram for the first iteration, including the instruction that is executed after `bne`.

(b) If it has been necessary to introduce stall cycles due to hazards in the question **??**, identify for each case the type of hazard and the instructions involved, explaining the reason for stall cycles.

(c) From the previous diagram, indicate the execution time of an iteration (in cycles) when the predictor hits and when the predictor misses.

(d) What would happen if this same code (**without any modification**) was executed on a processor that would solve the control hazards by means of the delayed branch technique? Reason the answer.

4. **(2.5 points)**

The following program copies the non-zero components of the source vector to the destination vector.

```
i = 0;
j = 0;
n = 10;
do
{
   if (source[i] != 0) /* Branch b1 */
   {
      destination[j] = source[i];
      j++;
   }
   i++;
   n--;
}
while (n != 0) /* Branch b2 */
```

This program is translated by a compiler into the following MIPS64 assembly code.

```
      daddi r1,r0,0   # r1 = Address of the source vector
      daddi r2,r0,80  # r2 = Address of the destination vector
      daddi r3,r0,10  # r3 = Loop control (n)
loop: ld r10,0(r1)
      daddi r1,r1,8
      beqz r10,fi     # Branch b1 (taken if source[i] == 0)
      sd r10,0(r2)
      daddi r2,r2, 8
fi:   daddi r3,r3,-1
      bnez r3,loop    # Branch b2 (taken if n != 0)
      trap 0
      nop
      nop
      nop
```

This code is executed on a pipelined MIPS64 processor with the usual 5 stages, resulting in the following instruction-time diagram for the first iteration of the loop.

```
                          1  2  3  4  5  6  7  8  9 10 11 12 13 14
loop: ld r10,0(r1)       IF ID EX ME WB
      daddi r1,r1,#8        IF ID EX ME WB
      beqz r10,fi              IF ID EX ME WB
      sd r10,0(r2)                IF ID X
      daddi r2,r2,#8                 IF X
fi:   daddi r3,r3,-1                    IF ID EX ME WB
      bnez r3,loop                         IF ID EX ME WB
      trap #0                                 IF ID X
      nop                                        IF X
loop: ld r10,0(r1)                                  IF ID EX ME WB
```

Taking into account that the processor has a dynamic BTB predictor with 1 bit for the branch condition, and assuming that the BTB is empty at the beginning of the execution of the program, it is requested, **reasoning the answers**:

(a) At what stage is the PC updated when the branch is taken? At what stage is the branch condition computed?

(b) In the attached sheet, complete the trace of the execution of branch `b2`. With respect to this branch, how many times will the predictor hit?

(c) Assuming that the source vector contains the components $0, 1, 0, 1, 0, 1, 0, 1, 0, 1$, complete in the attached sheet the execution trace of branch `b1`. With respect to this branch, how many times will the predictor hit?

(d) It has been decided to replace the one-bit predictor with a two-bit predictor with hysteresis and states *Stronly Not Taken* (SNT), *Weakly Not Taken* (WNT), *Weakly Taken* (WT), and *Strongly Taken* (ST). Assuming that the BTB entry for the branch `b1` is in the SNT state before the execution of the program, and that the source vector contains the same components as above, complete in the attached sheet the execution trace of branch `b1`. How many times will the predictor hit with respect to branch `b1`?

Surname and name:	

Exercise 3

a) Instruction-time diagram for the first iteration, including the instruction that is executed after `bne`.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Surname and name:	

Exercise 4

b) Trace of the execution of branch `b2`.

Iteration	01	02	03	04	05	06	07	08	09	10
Taken (Yes/No)										
Prediction (Yes/No)										

c) Trace of the execution of branch `b1`.

Iteration	01	02	03	04	05	06	07	08	09	10
source[i]	0	1	0	1	0	1	0	1	0	1
Taken (Yes/No)										
Prediction (Yes/No)										

d) Trace of the execution of branch `b1` with two-bit predictor with hysteresis.

Iteration	01	02	03	04	05	06	07	08	09	10
source[i]	0	1	0	1	0	1	0	1	0	1
Taken (Yes/No)										
SNT/WNT/WT/ST										
Prediction (Yes/No)										