Exercises on Instruction Level Parallelism

1. Recommended exercises

The following exercises are recommended. All of them appear in the book:
Source: Computer Architecture: A Quantitative Approach. 5 Ed
Hennessy and Patterson. Morgan Kaufmann. 2012.

- Appendix C: Exercises C.1, C.2, C.3, C.4 y C.5.
- Chaprter 3: Exercises 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.10, 3.12.

2. Exam exercises

Exercise 1 *June 2015 exam.*

Given a MIPS processor with a pipeline that has two separate register files (one for integers and the other one for floating point numbers). The integer register bank has 32 registers. The floating-point register bank has 16 double-precision registers (**$f0**, **$f2**, ..., **$f30**).

We assume that we have enough fetch and decode bandwidth to execute one instruction per cycle without stalls (with the exception of stalls associated to data dependencies).

Table 1 shows the extra latencies related to some types of instructions. These latencies have to be considered when there are data dependencies. When there are no dependencies, these extra latencies are not applicable.

Cuadro 1: Additional latencies per instruction

Instruction	Additional latency	Operation
ldc1	+2	Loads a 64 bit value into a floating point register.
sdc1	+2	Stores a 64 bit value into main memory.
add.d	+4	Adds double precision floating point registers
mul.d	+6	Multiplies double precision floating point registers.
addi	+0	Adds a value and an integer register.
subi	+0	Subtracts a value from an integer register.
bnez	+1	Branches if a register value is zero.

Instruction **bnez** uses delayed branching with one *delay slot*.

We intend to execute the following code in the previous architecture:

```
loop:   ldc1 $f0, ($t0)
        ldc1 $f2, ($t1)
        add.d $f4, $f0, $f2
        mul.d $f4, $f4, $f6
        sdc1 $f4, ($t2)
        addi $t0, $t0, 8
        addi $t1, $t1, 8
        subi $t3, $t3, 1
        bnez $t3, loop
        addi $t2, $t2, 8
```

The initial register values are:

- **$t0**: **0x00100000**.
- **$t1**: **0x00140000**.
- **$t2**: **0x00180000**.
- **$t3**: **0x00000100**.

Complete the following tasks:

1. Enumerate the RAW dependencies related to the previous code.
2. Show all the stalls that are produced when one single code iteration is being executed. Show the overall number of cycles per iteration.
3. Schedule the loop instructions in order to reduce the number of stalls.
4. Unroll the loop in the following way: each unrolled iteration processes four array positions. Obtain the resulting speedup. Note: use real register names (**$f0**, **$f2**, ..., **$f30**).

IMPORTANT: The solutions that do not use real existing registers (e.g.: **$f2'** o **$f2"**) will not be considered valid.

Exercise 2 *January 2015 exam.*

The following code fragment is stored starting from memory address **0x1000100C** in a machine where all instructions occupy 4 bytes:

```
loop:   lw $r2, 0($r0)
        addi $r3, $r2, 20
        sw $r3, 0($r1)
        addi $r0, $r0, 4
        addi $r1, $r1, 4
        bnez $r2, loop
```

This code runs in a machine with a L1 data cache which is 2 ways set-associative and with a size of 32 KB and a L1 instruction cache with the same characteristics. It also has a L2 unified cache which is 8 ways set-associative with a size of 1MB. In both cases the line size is 32 bytes. It is assumed that a cache hit in L1 requires 4 cycles, a cache hit in L2 requires 14 additional cycles, and penalty for bringing a memory block from main memory to L2 is 80 cycles. All caches have a write-back policy.

Initially, value of registers are:

- **$r0**: **0x00010000**.
- **$r1**: **0x00080000**.

Starting from location **0x00010000** all the values in memory are different from zero until location **0x000100FC**. In memory location **0x000100FC** there is a zero value.

1. Determine which should be the average access time assuming that a program (different from the one above) performs on average 2 data accesses per instruction and has the following miss rate:
   - L1 instructions: 10 %
   - L1 data: 5 %
   - L2: 2 %
2. Determine the number of misses produced during the execution of the provided code fragment for data L1 cache, instruction L1 cache, and L2 cache.
3. Prepare a time diagram for a MIPS architecture with a 5-stage pipeline, for the first loop iteration assuming that initially there are no data and no instructions in caches and with the following considerations:
   - There is no forwarding hardware.
   - Architecture allows that an instruction writes a register and another instruction reads that same register without problems.
   - Branches are handled flushing the pipeline.
   - Effective branch addresses are computed in the execution stage.

   NOTE:
4. Keep in mind when preparing the diagram the stalls due to misses in cache hierarchy for instructions (stage IF) as well as for data reads and writes (stage M).
5. Repeat the time diagram for the second iteration.

Exercise 3 *October 2014.*

Let's consider the following code fragment:

```
bucle:  lw $f0, 0($r1)
        lw $f2, 0($r2)
        mul.f $f4, $f0, $f2
        add.d $f6, $f6, $f4
        addi $r1, $r1, 4
        addi $r2, $r2, 4
        sub $r3, $r3, 1
        bnez $r3, bucle
```

1. Make a list with all possible data dependences , without considering a specific structure of the segmented architecture. For each dependency you must indicate, register, origin instruction, instruction of destination and type of dependency.
2. Create a time diagram for a MIPS architecture with a pipeline of 5 stages, with the following considerations:
   - There is no forwarding hardware
   - The architecture allows an instruction to write in a register and other instruction to read that same register without any problems.

- The branches are processed by means of flushing the pipeline.
- Memory references require a clock cycle.
- The effective branch address is calculated at the execution stage.

3. Determine how many cycles are needed to execute N loop iterations.
4. Create a time diagram for a MIPS architecture with a pipeline of 5 stages with the following considerations:
   - There is fordwarding hardware
   - Assume that bifurcations are treated by predicting all branches as taken.
5. Determine how many cycles are needed to run N iterations of the The conditions of paragraph 4.

Exercise 4 *October 2014.*

Be the following code fragment:

```
bucle:  lw $f0, 0($r1)
        lw $f2, 0($r2)
        sub.f $f4, $f0, $f2
        mul.d $f4, $f4, $f4
        add.d $f6, $f6, $f4
        addi $r1, $r1, 4
        addi $r2, $r2, 4
        sub $r3, $r3, 1
        bnez $r3, bucle
```

1. Make a list with all possible dependence on data, without considering any specific structure of the segmented architecture. For each dependency you must indicate, register, source instruction, target instruction and dependency type.
2. Create a time diagram for a MIPS architecture with a pipeline in 5 stages, with the following considerations:
   - There is no forwarding hardware.
   - The architecture allows instructions to write to a register and another instruction to read that same register in the same clock cycle without any problems
   - Bifurcations are treated by emptying the pipeline.
   - Memory references require one clock cycle.
   - The effective branch direction is calculated at the execution stage.
3. Determine how many cycles are needed to execute N loop iterations.
4. Create a time diagram for a MIPS architecture with a pipeline In 5 stages with the following considerations:
   - There is a complete forwarding hardware.
   - Assume that bifurcations are treated by predicting all branches as taken.
5. Determine how many cycles are needed to run N iterations of the The conditions of paragraph 4.

Cuadro 2: Latencies between instructions

Instruction producing the result	Instruction using the result	Latency
FP ALU operation	Other FP ALU operation	6
FP ALU operation	Store double	3
Load double	FP ALU operation	2
Load double	Store double	0

Exercise 5 *June 2014 exam.*

A given processor has the latencies between instructions given by Table 2.
In this machine we want to run the following piece of code:

```
LOOP:   L.D  F0, 0(R1)
        L.D  F2, 0(R2)
        ADD.D F4, F0, F2
        S.D  F4, 0(R3)
        DADDUI R1, R1, #-8
        BNE  R1, R4, LOOP
```

Initially registers have the following values:

- **R1**: Address of last element in first source array.
- **R2**: Address of last element in second source array.
- **R3**: Address of last element in target array.
- **R4**: Precomputed with **8(R4)** being first element in first source array.

All arrays have a size of 4,000 elements.
Complete the following tasks:

1. Determine how many cycles are required to execute all loop iterations without modifications.
2. Determine how many cycles are required to execute all loop iterations if loop scheduling is performed.
3. Determine how many cycles are required to execute all loop iterations if loop unrolling is performed for every two iterations.
4. Determine how many cycles are required to execute all loop iterations if loop unrolling is performed for every four iterations.

Exercise 6 *January 2014 exam.*

A given processor is intended to run the following code fragment:

```
i0: lw $r4, 0($r1)
i1: lw $r5, 0($r2)
i2: add $r4, $r4, $r5
i3: sw $r4, 0($r3)
i4: addi $r1, $r1, 4
i5: addi $r2, $r2, 4
i6: addi $r3, $r3, 4
i7: bne $r3, $r0, i0
```

Assume that the processor has a segmented architecture of 5 steps (fetch, decode, execute, memory and writeback) without forwarding. All operations are executed in one cycle per stage, except:

- Load and store instructions, that require two cycles for the memory stage (an additional cycle).
- Branch instructions require an additional cycle in the execution stage. Assume that these instructions do not have any branch prediction.

Answer the following questions:

1. Determine the RAW data hazards in the code that have impact in code execution.
2. Show a timing diagram with the stages for each instruction in one iteration.
3. Determine how many cycles are required to execute one loop iteration if there is no branch prediction.
4. Propose a loop unrolling assuming that the loop runs for 1000 iterations. Unroll with a factor of four iterations.
5. Determine the obtained speedup obtained through unrolling performed in the previous section.

Exercise 7 *October 2013 exam.*

A certain processor runs the following code segment:

```
i0:    lw $r3, 0($r0)
i1:    lw $r1, 0($r3)
i2:    addi $r1, $r1, 1
i3:    sub $r4, $r3, $r2
i4:    sw $r1, 0($r3)
i5:    bnz $r4, i0
```

Assume that the processor has 5 stages pipelined architecture (fetch, decode, execute, memory and write-back) without forwarding. All stages run in one cycle, except load and store operations which require two additional cycles for memory access latency, and branching instructions which require one additional execution cycle.

1. Identify RAW data hazards in the code.
2. Show a timing diagram with execution stages for each instruction in one iteration.
3. Determine how many cycles are required for executing one loop iteration when there is no branch prediction.
4. Determine how many cycles are required for executing one loop iteration when a branch predictor (always predicting to taken) is used.

Exercise 8 *October 2013 exam.*

The following code is written in MIPS assembler. Assume that, before starting instructions execution, registers **R3** and **R5** contain, respectively, the memory addresses of the first and last element in an array with 9 entries (initial value for **R1=0x010** and **R5=0x018**).

```
Loop:  LD    R4 0(R1)
       DIV   R2 R2 R4
       ADD   R1 R1 #1
       SUB   R5 R5 #1
       SD    R4 0(R5)
       SUB   R6 R1 R5
       BNEZ  R6 Loop
```

1. Express all RAW and WAW data hazards in the code.
2. Provide a timing diagram assuming that the processor is pipelined with 5 stages (*fetch*, *decode*, *execution*, *memory* y *write back*). An instruction per cycle is issued and the processor **does not use forwarding**. Assume that there is pipeline freezing for branches and that **there is one additional cycle per memory access in reads (LD)**, which does not happen in writes.
3. Determine the number of cycles needed by the loop (all iterations) to run.

Exercise 9 *October 2013 exam.*

Consider the following code fragment:

```
Loop:   LD R4, 0(R2)
        LD R5, 0(R3)
        ADD R6, R4, R5
        SD R6, 0(R3)
        BNZ R6, Loop
```

1. Number of needed cycles to run one loop iteration in a non-pipelined processor. Memory access instructions have a 3 cycles latency. Branch instruction has a 1 cycle latency.
2. Identify RAW data dependencies in the code.
3. Compute the number of needed cycles to run one iteration of the loop in a 5 stages pipelined processor. The processor uses the forwarding technique and the branch prediction strategy is pipeline freezing. Complete a timing diagram.

Exercise 10 *June 2013 exam.*

In the following code, each instruction has as associated cost one cycle, besides those included in table 3. Besides, assume that the machine is able to issue one instruction per cycle, except waits due to stalls and that the processor is pipelined with a single data path.

Due to structural hazards, stalls happen always independently that there are (or not) data dependencies with the following instructions. Initially the following values are in registers **R1=0**, **R2=24**, **R3=16**.

```
Loop1:  LD F2, 0(R1)
        ADDD F2,F0,F2
Loop2:  LD F4,0(R3)
        MULTD F4,F0,F4
        DIVD F10,F4,F0
        ADDD F12,F10,F4
        ADDI R1,R1,#8
        SUB R18,R2,R1
        BNZ R18,Loop2
        SD F2,0(R3)
        ADDI R3, R3, #8
        SUB R20,R2,R3
        BNZ R20,Loop1
```

1. Compute the number of needed cycles to run one iteration of the external loop and thirty of the internal loop.
2. Unroll three iterations from the internal loop, do not unroll the external loop and perform again the computation from the previous section.

Cuadro 3: Additional costs per instruction

Instruction	Additional cost
LD	3
SD	1
ADD	2
MULTD	4
DIVD	10
ADDI	0
SUB	0
BNZ	1

3. Compare the timing results (number of cycles) from the first and second sections. Justify your answers.

Exercise 11 *January 2013 exam.*

Given the following code written in MIPS assembler, where the same memory address is read and written several times and with values for **R1** and **R2** are **R1=1** and **R2=1000**.

```
        ADDI R3, R3,1
Loop:   LD R4, 0(16)
        MULT R5, R4,R4
        ADD R5, R3,R5
        SD R5, 0(16)
        DADDI R3, R4,1
        DADDI R1, R1,1
        BNE R1, R2,Loop
```

The code runs in a MIPS-like pipelined processor with the following execution stages: fetch, decode, execute, memory and write-back. The processor issues one instruction per cycle and has forwarding capability. All the stages in the data path run in one cycle except the following cases: **SD** instruction uses and additional cycle to read **register R5 from the register file**, instruction **LD** uses an additional cycle to write **value from memory into register R4 from the register file** and instructions **ADD** and **MULT** use an additional cycle to complete its execution in the ALU. Assume that branch prediction strategy in *not taken*.

1. Express only WAW data hazards in the code showing the instructions causing the hazard and the associated register. In which situation could a WAW hazard originate and incorrect result for the program?

2. Draw a time diagram assuming forwarding. Show the number of cycles that the program would take to execute.

Exercise 12 *June 2012 exam.*

Given the following code fragment:

```
Loop:   LD R4,0(R2)
        LD R5, 0(R3)
        ADD R6, R4, R5
        SD R6, 0(R3)
        ADD R6, R6, R4
        ADDI R3, R3, #8
        SUB R20, R4, R3
        BNZ R20, Loop
```

1. Enumerate the existing data dependencies in the code. For each one determine the datum causing the dependency.

2. Provide a timing for this sequence for the 5 stages RISC pipeline without forwarding or bypassing hardware, but assuming that a data read and a data write to the register file can be performed in the same cycle (assuming forwarding through the register file). Assume that branches are handled by flushing the pipeline and that all memory accesses (including instruction fetch) take two cycles. Justify your answer.

Exercise 13 *May 2012 exam.*

Given the following code section:

```
        DADDUI R3, R1, #40      ; I1
LOOP:   L.D F0, 0(R1)           ; I2
        L.D F2, 0(R2)           ; I3
        ADD.D F4, F0, F2        ; I4
        S.D F4, 0(R1)           ; I5
        DADDUI R1, R1, #8       ; I6
        DADDUI R2, R2, #8       ; I7
        BLE R1, R3, LOOP        ; I8
```

And considering that it runs in a machine with additional latencies between instructions expressed in Table 4.

Cuadro 4: Additional latencies

Instruction producing the result (previous)	Instruction using the result (subsequent)	Latency
FP ALU operation	FP ALU operation	5
FP ALU operation	Load/store double	4
FP ALU operation	Branch instruction	4
Load double	FP ALU operation	2
Load double	Load double	1
Store double	FP ALU operation	2

The *branch* instruction has a latency of one cycle and no *delay slot*.

Besides, assume that the machine is able to issue one instruction per cycle, except waiting due to stalls and that the processor has a pipeline with a single data path.

1. Identify all the data dependencies.

2. Determine the total number of cycles needed to run the complete section of code.

3. Modify the code to reduce stalls through the loop scheduling technique. Determine the obtained speedup versus to the non-scheduled version.

4. Modify the code performing a loop unrolling with two loop iterations. Determine the obtained speedup versus the non-scheduled version.