Solutions to exercises on Instruction Level Parallelism

J. Daniel García Sánchez (coordinator)
David Expósito Singh
Javier García Blas

Computer Architecture
ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid

1. 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.

Solution 1

Dependencies If the instructions are numbered sequentially starting at **I1** (Up to **I10**), the following RAW dependencies can be identified:

1. **$f0**: **I1** $\rightarrow$ **I3**.
2. **$f2**: **I2** $\rightarrow$ **I3**.
3. **$f4**: **I3** $\rightarrow$ **I4**.
4. **$f4**: **I4** $\rightarrow$ **I5**.
5. **$t3**: **I8** $\rightarrow$ **I9**.

Stalls The following are the stops that occur when executing the code:

```
ldc1 $f0, ($t0)          #I1
ldc1 $f2, ($t1)          #I2
<stall> x 2
add.d $f4, $f0, $f2      #I3
<stall> x 4
mul.d $f4, $f4, $f6      #I4
<stall> x 6
sdc1 $f4, ($t2)          #I5
addi $t0, $t0, 8         #I6
addi $t1, $t1, 8         #I7
subi $t3, $t3, 1         #I8
bnez $t3, bucle #I9
addi $t2, $t2, 8         #I10
```

In total, 22 cycles are required.

Loop scheduling Reordering instructions can reduce the number of stalls:

```
ldc1 $f0, ($t0)          #I1
ldc1 $f2, ($t1)          #I2
addi $t0, $t0, 8         #I6
addi $t1, $t1, 8         #I7
add.d $f4, $f0, $f2      #I3
subi $t3, $t3, 1         #I8
<stall> x 3
mul.d $f4, $f4, $f6      #I4
<stall> x 6
sdc1 $f4, ($t2)          #I5
bnez $t3, bucle #I9
addi $t2, $t2, 8         #I10
```

A total of 19 cycles are required.

Unrolled loop Unrolling the loop with a factor of four obtains:

```
ldc1 $f0, ($t0)
ldc1 $f2, ($t1)
ldc1 $f8, 8($t0)
ldc1 $f10, 8($t1)
ldc1 $f14, 16($t0)
ldc1 $f16, 16($t1)
ldc1 $f20, 24($t0)
ldc1 $f22, 24($t1)
add.d $f4, $f0, $f2
add.d $f12, $f8, $f10
add.d $f18, $f14, $f16
add.d $f24, $f20,$f22
<stall>
mul.d $f4, $f4, $f6
mul.d $f12, $f12, $f6
mul.d $f18, $f18, $f6
mul.d $f24, $f24, $f6
addi $t0, 32
addi $t1, 32
<stall>
sdc1 $f4, ($t2)
sdc1 $f12, 8($t2)
sdc1 $f18, 16($t2)
sdc1 $f24, 24($t2)
subi $t3, 4
bnez $t3, bucle
addi $t2, 32
```

In total, 27 cycles are required every 4 iterations. That is 6,75 cycles per iteration

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.

Solution 2

Average access time Regarding the accesses to the level 1 cache, there are 2 data access for each access to instructions. Therefore, the failure rate is obtained through a weighted average:

$$m_{L1} = \frac{m_{L1I} + 2 \cdot m_{L1D}}{3}$$

$$m_{L1} = \frac{0{,}1 + 2 \cdot 0{,}05}{3} = \frac{0{,}2}{3} = 0{,}0667$$

Therefore the average access time would be:

$$T = 4 + m_{L1} \cdot (14 + m_{L2} \cdot 80) = 4 + 0{,}0667 \cdot (14 + 0{,}02 \cdot 80) = 5{,}04 \text{ ciclos}$$

Number of misses The loop is executed $\frac{2^8}{4} = 2^6 = 64$ iterations.

We will analyze the instructions and data access separately.

The first statement is stored in the **0x1000100C** address. The last Instruction is stored at address **0x1000100C** $+(6-1)*4 =$ **0x10001020**.

In the first iteration, the first instruction generates a cache miss and brings the address block **0x10001000** - **0x1000101F**, which contains the instruction. The following instructions (I2, I3, I4, And I5) generate cache hits. Lastly, instruction I6 re-generates a miss. Therefore, the first iteration generates 2 misses and 4 hits. The rest of iterations generate hits in all cases.

Since the loop is executed 64 times, access to the instructions generates following accesses:

- L1I misses: 2
- L1I hits: $4 + 63 \cdot 6$
- L2 misses: 1
- L2 hits: 0

At each iteration of the loop, a memory address is read in the range **0x00010000** - **0x000100FC**. This corresponds to $\frac{2^8}{2^5} = 8$ cache lines.

In the same way, values are written in the **0x00080000 0x000800FC**, which are written in 8 lines of cache. Given that caches are set-associative, there is no conflict between the data read and written. As no L1 cache line is replaced there are no writes in the L2 cache.

1. L1D misses: 8 reads + 8 writes
2. L1D hits: 56 reads + 56 writes
3. L2 misses: 8 reads
4. L2 hits: 0

Time diagram for first iteration If the instructions are numbered as follows:

```
bucle:  lw   $r2, 0($r0)         #1
        addi $r3, $r2, 20        #2
        sw   $r3, 0($r1)         #3
        addi $r0, $r0, 4         #4
        addi $r1, $r1, 4         #5
        bnez $r2, bucle #6
```

We have the following RAD dependencies

1. **$r2**: **I2** → **I1**
2. **$r3**: **I3** → **I2**
3. **$r0**: **I4** → **I1**
4. **$r1**: **I3** → **I5**

In the absence of *forwarding*, when there is RAW dependency, the destination instruction stalls until completing the WB cycle of the source instruction. Table 2 shows the corresponding timing diagram.

- The first statement stops 98 cycles (80 + 14 + 4) in the fetch because it is a miss in the entire memory hierarchy.
- The reading of data from the first instruction is a read miss and requires 98 cycles as well.
- The second instruction is a hit and requires four cycles to perform the L1I cache fetch.
- The instruction I3 is a hit and requires four cycles to perform the capture of the L1I cache.
- Writing data from instruction I3 is a write miss and requires 98 cycles.
- The instruction I4 is a hit and requires four cycles to perform the capture of the L1I cache.
- The I4 instruction can not start its memory cycle until I3 memory access is completed.
- The instruction I5 is a hit and requires four cycles to perform the L1I instruction fetch.
- The I5 instruction can not start its execution cycle until the execution of I4 is finished.
- Instruction I6 is a fault and requires 98 cycles to perform L1I instruction fetch.
- Instruction I7 (the next to bnez) can not start fetching until the fetch unit is released.
- Although the branch address at the end of the decoding step of I6 is known, the branch direction (take or non-taken) is not known until the end of the execution step.

In total, 310 cycles are required.

Timing diagram for the second iteration Table 3 shows the corresponding timing diagram. In total, 28 clock cycles are required.

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.

Instruction	1–98	99	100	101	102	103–198	199	200	201	202	203	204	205	206	207	208	209	210
I1: **lw $r2**, **0**(**$r0**)	IF	ID	EX	M	M	M	WB											
I2: **addi $r3**, **$r2**, **20**		IF1	IF2	IF3	IF4	–	ID	EX	M	WB								
I3: **sw $r3**, **0**(**$r1**)							IF1	IF2	IF3	IF4	ID	EX	M	M	M	M	M	M
I4: **addi $r0**, **$r0**, **4**											IF1	IF2	IF3	IF4	ID	EX	–	–
I5: **addi $r1**, **$r1**, **4**															IF1	IF2	IF3	IF4

Instruction	211	212	213	214	215–302	303	304	305	306	307	308	309	310	311	312
I1: **lw $r2**, **0**(**$r0**)															
I2: **addi $r3**, **$r2**, **20**															
I3: **sw $r3**, **0**(**$r1**)	M	M	M	M	M	WB									
I4: **addi $r0**, **$r0**, **4**	–	–	–	–	–	M	WB								
I5: **addi $r1**, **$r1**, **4**	ID	–	–	–	–	EX	M	WB							
I6: **bnez $r2**, **bucle**	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	ID	EX	M	
I7: **?**												IF	flush		
I1: **lw $r2**, **0**(**$r0**)														IF	

Cuadro 2: Timing diagram of the first iteration of the exercise 2.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw $r2, 0($r0)	IF	IF	IF	IF	ID	EX	M	M	M	M	WB										
I2: addi $r3, $r2, 20					IF	IF	IF	IF	–	–	ID	EX	M	WB							
I3: sw $r3, 0($r1)											IF	IF	IF	IF	ID	EX	M	M	M	M	WB
I4: addi $r0, $r0, 4															IF	IF	IF	IF	ID	EX	M
I5: addi $r1, $r1, 4																			IF	IF	IF

Instruction	22	23	24	25	26	27	28	29	30
I1: lw $r2, 0($r0)									
I2: addi $r3, $r2, 20									
I3: sw $r3, 0($r1)									
I4: addi $r0, $r0, 4	WB								
I5: addi $r1, $r1, 4	IF	ID	EX	M	WB				
I6: bnez $r2, bucle		IF	IF	IF	IF	ID	EX	M	WB
I7: ?						IF	flush		
I1: lw $r2, 0($r0)								IF	

Cuadro 3: Timing diagram of the second iteration of the exercise 2.

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.

Solution 3

Data dependencies If the instructions are numbered from **I1** (first instruction) to **I8** (last instruction) we have the following dependencies:

- **$f0**: **I1** → **I3** (RAW)
- **$f2**: **I2** → **I3** (RAW)
- **$f4**: **I3** → **I4** (RAW)
- **$r3**: **I7** → **I8** (RAW)

First timming diagram Given that there is no forwarding, when there is a RAW dependency we have to wait for the WB of the source instruction before starting the execution of the destination instruction. Table 4 shows the corresponding timing diagram.

- Instruction I3 has a stall until I2 has written the value read in**$f2**. We can perform the reading of register file in the same cycle than the one in which I2 writes in the register file (cycle 6).
- Instruction I4 can not start until the fetch unit is released (cycle 6).
- Instruction I4 has a stall until I3 has written the value calculated in **$f4**. We can perform the reading of register file in the same cycle than the one in which I3 writes in the register file (cycle 9).
- Instruction I5 can not start until the Fetch unit is available (cycle 9).
- Instruction I8 has a stall until I7 has written the value **$r3**. We can perform the reading of register file in the same cycle in which I7 writes in the register file (cycle 15).

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: **lw $f0**, 0(**$r1**)	ID	ID	EX	M	WB																
I2: **lw $f2**, 0(**$r2**)		IF	ID	EX	M	WB															
I3: **mul.f $f4**, **$f0**, **$f2**			IF	–	–	ID	EX	M	WB												
I4: **add.d $f6**, **$f6**, **$f4**						IF	–	–	ID	EX	M	WB									
I5: **addi $r1**, **$r1**, **4**									IF	ID	EX	M	WB								
I6: **addi $r2**, **$r2**, **4**										IF	ID	EX	M	WB							
I7: **sub $r3**, **$r3**, **1**											IF	ID	EX	M	WB						
I8: **bnez $r3**, **bucle**												IF	–	–	ID	EX	M	WB			
I9: (sig a **bnez**)															IF						
I1: **lw $f0**, **0**(**$r1**)																	IF	ID	EX	M	WB

Cuadro 4: First timing diagram for exercise 3.

- Instruction I9 (the next to bnez) can not start fetching until the fetch unit is available (cycle 15).
- Although the branch address is known at the end of the decoding of I8, the branch result (taking or not taking) is not known until the end of the execution. stage Therefore the instruction fetch is repeated in the cycle 17.

First cycle estimation To determine the number of cycles, we need to determine how many cycles are required for any iteration and how many for the last iteration.

The cost of a different iteration of the last is obtained by determining the number of cycles from the start of execution of I1 until it is executed again. This is 16 cycles.

The cost of the last iteration is obtained by determining the number of cycles until the execution of Instruction I8 is completed. These are 18 cycles.

$$Coste = 16 \cdot n + 2$$

.

Second cycle estimation Forwarding is now allowed whenever possible and there is no need to wait for the WB stage.

Table 5 shows the timing diagram:

- Instruction I3 can start execution after the memory stage of I2 (cycle 6) because of the forwarding.
- Instruction I4 can not start decoding until the decoding unit has been released (cycle 6).
- Instruction I4 can start execution after the execution stage of I3 (cycle 7) because of the forwarding.
- Instruction I5 can not start the fetch until the fetch unit is released (cycle 6).
- Instruction I8 can not start decoding until it has been calculated the value of $\mathring{3}$ (cycle 10) and passed through forwarding (cycle 11).

Second cycle estimation The cost of an iteration different from the last iteration is 10 cycles. The last iteration requires 14 cycles.

$$Coste = 10 \cdot n + 4$$

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.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: **lw $f0**, 0(**$r1**)	ID	ID	EX	M	WB																
I2: **lw $f2**, 0(**$r2**)		IF	ID	EX	M	WB															
I3: **mul.f $f4**, **$f0**, **$f2**			IF	ID	–	EX	M	WB													
I4: **add.d $f6**, **$f6**, **$f4**				IF	–	ID	EX	M	WB												
I5: **addi $r1**, **$r1**, **4**						IF	ID	EX	M	WB											
I6: **addi $r2**, **$r2**, **4**							IF	ID	EX	M	WB										
I7: **sub $r3**, **$r3**, **1**								IF	ID	EX	M	WB									
I8: **bnez $r3**, **bucle**									IF	ID	EX	M	WB								
I1: **lw $f0**, **0**(**$r1**)											IF	ID	EX	M	WB						

Cuadro 5: Second time diagram of exercise 3.

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.

Solution 4

Data dependencies If the instructions are numbered from **I1** (first statement) to **I9** (last instruction) have the following dependencies:

- **$f0**: **I1** → **I3** (RAW)
- **$f2**: **I2** → **I3** (RAW)
- **$f4**: **I3** → **I4** (RAW, WAW)
- **$f4**: **I4** → **I5** (RAW)
- **$r3**: **I8** → **I9** (RAW)

First timing diagram When there is no forwarding, when there is a RAW dependency, it is necessary to wait for the cycle WB from the source instruction before starting the target instruction cycle. Table 6 shows the corresponding timing diagram.

- Instruction I3 has a stall until I2 has written the value read from **$f2**. We can perform the register file reading in the same cycle than the one in which I2 writes in the register bank (cycle 6).
- Instruction I4 can not start the fetch until the fetch unit is released (cycle 6).
- Instruction I4 has a stall until I3 has written the value calculated in **$ f4**. We can perform the reading of the register file in the same cycle as I3 writes in the register file (cycle 9).
- Instruction I5 can not start the fetch until the fetch unit is released (cycle 9).
- Instruction I5 has a stall until I4 has written the value calculated from **$f4**. We can perform the reading of the register file in the same cycle as I4 writes in the register file (cycle 12).

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: **lw $f0**, 0(**$r1**)	IF	ID	EX	M	WB																
I2: **lw $f2**, **0**(**$r2**)		IF	ID	EX	M	WB															
I3: **sub.f $f4**, **$f0**, **$f2**			IF	–	–	ID	EX	M	WB												
I4: **mul.d $f4**, **$f4**, **$f4**						IF	–	–	ID	EX	M	WB									
I5: **add.d $f6**, **$f6**, **$f4**									IF	–	–	ID	EX	M	WB						
I6: **addi $r1**, **$r1**, **4**												IF	ID	EX	M	WB					
I7: **addi $r2**, **$r2**, **4**													IF	ID	EX	M	WB				
I8: **sub $r3**, **$r3**, **4**														IF	ID	EX	M	WB			
I9: **bnez $r3**, **bucle**															IF	–	–	ID	EX	M	WB
I10: (next to **bnez**)																		IF	–		
I1: **lw $f0**, 0(**$r1**)																				IF	ID

Cuadro 6: First timing diagram for exercise 4.

- Instruction I6 can not start the fetch until the fetch unit is released (cycle 12).
- Instruction I9 has a stall until I8 has written the value **\$r3**. We can perform the reading of the register file in the same cycle in which I7 writes in the register file (cycle 18).
- Instruction I10 (the next to bnez) can not start the fetch until the fetch unit is released (cycle 18).
- Although the branch address is known at the end of the Decoding of I9, the branch result (taken or not takend) is not known until the end of the execution stage. Therefore the fetch up is repeated in the cycle twenty.

First cycle estimation To determine the number of cycles, we need to determine how many cycles are require for a generic iteration and how many cycles for the last iteration.

The cost of a different iteration of the last is obtained by determining the number of cycles from the start of the I1 execution until I1 is started again. This is 19 cycles.

The cost of the last iteration is obtained by determining the number of cycles until the execution of Instruction I9 is completed. These are 21 cycles.

$$Coste = 19 \cdot n + 2.$$

Second cycle estimation Forwarding is now allowed when is possible to use it and there is no waiting for the WB stage. Table 7 shows the corresponding timing diagram.

- Instruction I3 can start execution after the memory stage of I2 (Cycle 6) because of forwarding.
- Instruction I4 can not start decoding until decoding unit is released (cycle 6).
- Instruction I4 can start execution after the execution stage of I3 (cycle 7) because of forwarding.
- Instruction I5 can not start the fetch until the Fetch unit is available (cycle 6).
- Instruction I5 can start execution after the execution stage of I4 (cycle 8) because of forwarding.
- Instruction I9 can not start decoding until the value of $\mathring{3}$ has been calculated (cycle 11) and passed via forwarding (Cycle 12).

Second cycle estimation The cost of an iteration different from the last iteration is 11 cycles. The last iteration requires 15 cycles.

$$Coste = 11 \cdot n + 4$$

Exercise 5 *June 2014 exam.*

A given processor has the latencies between instructions given by Table 8.

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:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: **lw $f0**, 0(**$r1**)	IF	ID	EX	M	WB																
I2: **lw $f2**, **0**(**$r2**)		IF	ID	EX	M	WB															
I3: **sub.f $f4**, **$f0**, **$f2**			IF	–	ID	EX	M	WB													
I4: **mul.d $f4**, **$f4**, **$f4**				IF	–	ID	EX	M	WB												
I5: **add.d $f6**, **$f6**, **$f4**						IF	ID	EX	M	WB											
I6: **addi $r1**, **$r1**, **4**							IF	ID	EX	M	WB										
I7: **addi $r2**, **$r2**, **4**								IF	ID	EX	M	WB									
I8: **sub $r3**, **$r3**, **4**									IF	ID	EX	M	WB								
I9: **bnez $r3**, **bucle**										IF	–	ID	EX	M	WB						
I1: **lw $f0**, 0(**$r1**)												IF	ID	EX	M	WB					

Cuadro 7: Second timing diagram of exercise 4.

Cuadro 8: 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

- **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.

Solution 5

Section 1 The execution of an iteration of the loop would be:

```
L.D F0, 0(R1)
L.D F2, 0(R2)
<stall> x 2
ADD.D F4, F0, F2
<stall> x 3
S.D F4, 0(R3)
DADDUI R1, R1, #-8
BNE R1, R4, BUCLE
```

In total, each iteration requires 11 cycles to have started all the instructions. This gives a total of 44,000 cycles.

Section 2 DADDUI instruction can be executed before

```
L.D F0, 0(R1)
L.D F2, 0(R2)
DADDUI R1, R1, #-8
<stall> x 1
ADD.D F4, F0, F2
<stall> x 3
S.D F4, 0(R3)
BNE R1, R4, BUCLE
```

In total, each iteration now requires 10 cycles. This results in a total of 40,000 cycles

```
L.D F0, 0(R1)
L.D F2, 0(R2)
L.D F6, −8(R1)
L.D F8, −8(R2)
DADDUI R1, R1, #−16
ADD.D F4, F0, F2
ADD.D F10, F6, F8
<stall> x 2
S.D F4, 0(R3)
S.D F10, −8(R3)
BNE R1, R4, BUCLE
```

A total of 12 cycles per iteration are required. This results in a total of 24,000 cycles

```
L.D F0, 0(R1)
L.D F2, 0(R2)
L.D F6, −8(R1)
L.D F8, −8(R2)
L.D F12, −16(R1)
L.D F14, −16(R2)
L.D F18, −24(R1)
L.D F20, −24(R2)
DADDUI R1, R1, #−32
ADD.D F4, F0, F2
ADD.D F10, F6, F8
ADD.D F16, F10, F12
ADD.D F22, F18, F20
S.D F4, 0(R3)
S.D F10, −8(R3)
S.D F16, −16(R3)
S.D F22, −24(R3)
BNE R1, R4, BUCLE
```

A total of 18 cycles per iteration are required. This results in a total of 18,000 cycles

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.

Solution 6

Section 1

- $r4: i0 → i2
- $r5: i1 → i2
- $r4: i2 → i3
- $r3: i6 → i7

Section 2 Table 9 shows the corresponding timing diagram.

- I0: Requires two memory cycles.
- I1: It can not start memory stage until I0 does not end memory. Requires two memory cycles.
- I2: It can not start decoding until I1 does not do WB..
- I3: No puede empezar captación hasta que se libera unidad por I2.
- I4: It can not start pickup until unit is released by I2.
- I5: It can not start execution stage until I4 releases drive execution.
- I6: It can not start decode stage until I5 releases unit Of decoding.
- I7: It can not start decoding until I6 does WB.

Section 3 If it is considered that the decoding step includes a comparator over the register bank the following instruction to I7 can begin after the decoding stage of I7 (that is, in cycle 19) is completed. Each iteration requires 18 clock cycles.

If the decoding step is not considered to include a comparator, the Comparison of two records should be made with the general ALU and therefore it may not take the decision until the completion stage of I7 (In cycle 21). In this case each iteration requires 20 clock cycles.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I0	IF	ID	EX	M1	M2	WB																
I1		IF	ID	EX	–	M1	M2	WB														
I2			IF	–	–	–	–	ID	EX	M	WB											
I3								IF	–	–	ID	EX	M1	M2	WB							
I4											IF	ID	EX	–	M	WB						
I5												IF	ID	–	EX	M	WB					
I6													IF	–	ID	EX	M	WB				
I7															IF	–	–	ID	X1	X2	M	WB

Cuadro 9: First timing diagram of the exercise 6.

Section 4 A possible solution is presented below. Note, however, that possible more aggressive solutions that could lead to better res8ults.

speedups.

```
I0: lw $r4, 0($r1)
i1: lw $r5, 0($r2)
i2: lw $r6, 4($r1)
i3: lw $r7, 4($r2)
i4: lw $r8, 8($r1)
i5: lw $r9, 8($r2)
i6: lw $r10, 12($r1)
i7: lw $r11, 12($r2)
i8: add $r4, $r4, $r5
i9: add $r6, $r6, $r7
i10: add $r8, $r8, $r9
i11: add $r10, $r10, $r11
i12: sw $r4, 0($r3)
i13: sw $r6, 4($r3)
i14: sw $r8, 8($r3)
i15: sw $r10, 12($r3)
i16: addi $r3, $r3, 16
i17: addi $r2, $r2, 16
i18: addi $r1, $r1, 16
i19: bne $r3, $r0, i0
```

Section 5 Table 10 shows the corresponding timing diagram.

Depending on the criterion chosen in Section 2, the number of cycles for Iteration will be 33 or 35 and the number of cycles per iteration will be

$\frac{33}{4} = 8{,}25$ o $\frac{35}{4} = 8{,}75$

Therefore the speedup will be:

$$S = \frac{18}{8{,}25} = 2{,}18$$

So...

$$S = \frac{20}{8{,}75} = 2{,}28$$

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.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I0	IF	ID	EX	M1	M2	WB																
I1		IF	ID	EX	–	M1	M2	WB														
I2			IF	ID	–	EX	–	M1	M2	WB												
I3				IF	–	ID	–	EX	–	M1	M2	WB										
I4						IF	–	ID	–	EX	–	M1	M2	WB								
I5								IF	–	ID	–	EX	–	M1	M2	WB						
I6										IF	–	ID	–	EX	–	M1	M2	WB				
I7												IF	–	ID	–	EX	–	M1	M2	WB		
I8														IF	–	ID	–	EX	–	M	WB	
I9																IF	–	ID	–	EX	M	WB
I10																		IF	–	ID	M	WB
I11																				IF	ID	EX
I12																					F	D
I13																						F

Instruction	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
I10	WB														
I11	M	WB													
I12	EX	M1	M2	WB											
I13	ID	EX	–	M1	M2	WB									
I14	IF	ID	–	EX	–	M1	M2	WB							
I15		IF	–	ID	–	EX	–	M1	M2	WB					
I16				IF	–	ID	–	EX	–	M	WB				
I17						IF	–	ID	–	EX	M	WB			
I18								IF	–	ID	EX	M	WB		
I19										IF	ID	EX	EX	M	WB

Cuadro 10: Second timing diagram of the exercise 6.

4. Determine how many cycles are required for executing one loop iteration when a branch predictor (always predicting to taken) is used.

Solution 7

Section 1

- **$r3**: **i0** → **i1**
- **$r1**: **i1** → **i2**
- **$r1**: **i2** → **i4**
- **$r4**: **I3** → **i5**

Section 2 Table 11 shows the corresponding Timing diagram.

- **I0**: Requires three memory cycles
- **I1**: You can not start decoding until WB of **$r3** is done. Requires 3 cycles of memory.
- **I2**: It can not be fetched until the stage is released by **I1**. You can not start to decode until the WB of **$r1**
- **I4**: The decoding can not be started until **I2** does WB of **$r1**
- **I5**: Fetch can not start until **I4** decoding starts. The memory cycle cannot be started until the memory cycle of **I4** does not end.

Section 3 In this case the **I0** statement of the next iteration can not fetched until cycle 18, so an iteration requires 18 cycles.

Section 4 In this case, the prediction is performed in the decoding step, which is when it is known that it is a branch instruction, so **I0** starts in cycle 16, and one iteration requires 16 cycles.

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.

Solution 8

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I0	IF	ID	EX	M1	M2	M3	WB														
I1		IF	–	–	–	–	ID	EX	M1	M2	M3	WB									
I2							IF	–	–	–	–	ID	EX	M	WB						
I3												IF	ID	EX	M	WB					
I4													IF	–	ID	EX	M1	M2	M3	WB	
I5															IF	ID	E1	E2	–	M	WB

Cuadro 11: Timing diagram for exercise 7.

Section 1 When instructions **I1** and **I7** are executed, (being **I1** the first one), the following hazards are obtained:

- **R4**: **I1** → **I2**
- **R4**: **I1** → **I5**
- **R5**: **I4** → **I5**
- **R5**: **I4** → **I6**
- **R1**: **I3** → **I6**
- **R6**: **I6** → **I7**

Section 2 Table 12 shows the corresponding time diagram

Section 3 Total execution cycles of a loop iteration: 15

Given there are 5 iterations (values of **r1** and **r5** respectively: (0x010, 0x018), (0x011, 0x017),(0x012, 0x016), (0x013, 0x015) y (0x014, 0x014)) and it is necessary to wait for the **BNZ** instruction termination in the last iteration (3 extra cycles). We have:

$$15 \text{ cycles} \cdot 5 \text{ iterations} + 4 \text{ extra cycles} = 79 \text{ cycles}$$

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.

Solution 9

Section 1 Analysing separately the different types of instructions:

3 memory instruction · (1 issue + 3 latency) = 12 cycles
1 ALU instruction · 1 issue = 1 cycle
1 branch instruction · (1 issue + 1 latency) = 2 cycles
Total = 15 cycles

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LD R4 0(R1)	IF	ID	EX	M	–	WB														
DIV R2 R2 R4		IF	–	–	–	ID	EX	M	WB											
ADD R1 R1 #1						IF	ID	EX	M	WB										
SUB R5 R5 #1							IF	ID	EX	M	WB									
SD R4 0(R5)								IF	–	–	ID	EX	M	WB						
SUB R6 R1 R5											IF	ID	EX	M	WB					
BNEZ R6 Loop												IF	–	–	ID	EX	M	WB		
LD R4 0(**R1**)															–	IF	ID	EX	M	WB

Cuadro 12: Time diagram for exercise 8.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12
LD R4,0(R2)	IF	ID	EX	M	WB							
LD R5,0(R3)		IF	ID	EX	M	WB						
ADD R6,R4,R5			IF	ID	–	EX	WB					
SD R6,0(R3)				IF	–	ID	EX	M	WB			
BNZ R6, Bucle						IF	ID	EX	M	WB		
(**Next**)								IF	ID	EX	M	WB

Cuadro 13: Timing diagram for exercise 9.

Section 2 The following dependencies are identified:

- **R4**: **LD** → **ADD**.
- **R5**: **LD** → **ADD**.
- **R6**: **ADD** → **SD**.

Section 3 Table 13 shows the timing diagram:

Section 4

$$S = \frac{15 \cdot 2}{(7 \cdot 2) + 3} = \frac{30}{17} = 1{,}76$$

Exercise 10 *June 2013 exam.*

In the following code, each instruction has as associated cost one cycle, besides those included in table 14. 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.
3. Compare the timing results (number of cycles) from the first and second sections. Justify your answers.

Solution 10

Cuadro 14: Additional costs per instruction

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

Section 1 1 Analysis:

```
Loop1: LD F2, 0(R1)        1+3
ADDD F2,F0,F2              1+2
Loop2: LD F4,0(R3)         1+3
MULTD F4,F0,F4             1+4
DIVD F10,F4,F0             1+10
ADDD F12,F10,F4            1+2
ADDI R1,R1,#8              1
SUB R18,R2,R1              1
BNZ R18,Loop2              1+1

Inner-loop iteration       27

SD F2,0(R3)                1+1
ADDI R3,R3,#8              1
SUB R20,R2,R3              1
BNZ R20,Loop1              1+1

Outer-loop iteration:      13
```

Thee inner-loop iterations and one outer-loop iteration: $= 13 + (27 \cdot 3) = 94$ cycles

Section 2 Analysis:

```
Loop1: LD F2, 0(R1)        1+3
ADDD F2,F0,F2              1+2
Loop2: LD F4,0(R3)         1+3
LD F6,0(R3)                1+3
LD F8,0(R3)                1+3
MULTD F4,F0,F4             1+4
MULTD F6,F0,F6             1+4
MULTD F8,F0,F8             1+4
DIVD F10,F4,F0             1+10
DIVD F12,F6,F0             1+10
DIVD F14,F8,F0             1+10
ADDD F16,F10,F4            1+2
ADDD F18,F12,F6            1+2
ADDD F20,F14,F8            1+2
ADDI R1,R1,#24             1
SUB R18,R2,R1              1
BNZ R18,Loop2              1+1

Three inner-loop iterations 73

SD F2,0(R3)                1+1
ADDI R3,R3,#8              1
SUB R20,R2,R3              1
BNZ R20,Loop1              1+1
```

Thee inner-loop iterations and one outer-loop iteration $= 13 + (73 \cdot 1) = 86$ cycles

Section 3 It can be seen that there is no significant reduction of cycles. That's because the inner loop has very few iterations and that the weight of the control of the loop (increase indices and jump) is relatively small compared to the rest of instructions.

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.

Solution 11

Section 1

- **WAW**: **I4** con **I3** en **R5**.
- **WAW**: **I1** con **I6** en **R3**.

You would get an incorrect result if **I4** is executed before **I3** on segmented processors where instructions can be reordered.

Section 2 Table 15 shows the corresponding timing diagram.
The cycles required to execute the loop can be divided:

- Number of cycles before the loop: 1.
- Number of cycles per iteration of the loop: 12.
- Number of extra cycles of the last loop instruction: 3

The loop executes a total of 999 times. Therefore the number of cycles will be:

$$\text{cycles} = 1 + (12 \cdot 999) + 3 = 11902$$

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ADDI R3 ,R3, 1	IF	ID	EX	M	WB											
LD R4 , 0(16)		IF	ID	EX	M	WB	WB									
MULT R5 ,R4 ,R4			IF	ID	–	EX	EX	M	WB							
ADD R5 ,R3 ,R5				IF	–	ID	–	EX	EX	M	WB					
SD R5 , 0(16)						IF	–	ID	ID	EX	M	WB				
DADDI R3 ,R4, 1								IF	–	ID		EX	M	WB		
DADDI R1 ,R1, 1										IF	ID	EX	M	WB		
BNE R1 ,R2 , Loop											IF	–	ID	EX	M	WB
(sig)													IF	IF	ID	EX

Cuadro 15: Timing Diagram of Exercise 11.

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.

Solution 12

Section 1 The following data dependencies can be identified:

- **R4**: **I1** → **I3** (RAW).
- **R5**: **I2** → **I3** (RAW).
- **R6**: **I4** → **I3** (RAW).
- **R6**: **I5** → **I4** (WAR).
- **R6**: **I5** → **I3** (WAW).
- **R4**: **I5** → **I1** (RAW).
- **R3**: **I6** → **I4** (WAR).
- **R3**: **I7** → **I6** (RAW).
- **R20**: **I8** → **I7** (RAW).

If a read and a write of a data in the register file can be done in the same clock cycle, then the Decoding (ID) and write-back (WB) can be done in the same cycle.

Section 2 Table 16 shows the corresponding timing diagram.

Exercise 13 *May 2012 exam.*

Given the following code section:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
LD R4, 0(R2)	IF	–	ID	EX	M	–	WB																	
LD R5, 0(R3)			IF	–	ID	EX	M	–	WB															
ADD R6,R4,R5					IF	–	–	–	ID	EX	M	B												
SD R6, 0(R3)									IF	–	–	ID	EX	M	–	WB								
ADD R6,R6,R4											IF	–	ID	EX	M	WB								
ADDI R3,R3, #8													IF	–	ID	EX	M	WB						
SUB R20,R4,R3															IF	–	–	ID	EX	M	WB			
BNZ R20, Bucle																		IF	–	–	ID	EX	M	WB
(sig)																			–	–	–	–	–	

Cuadro 16: Exercise timing diagram 12.

```
         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 17.

Cuadro 17: 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.

Solution 13

Section 1 The following dependencies are produced:

- **I4** → **I2** (**F0**: RAW), **I4** → **I3** (**F2**: RAW)
- **I5** → **I4** (**F4**: RAW)
- **I6** → **I1** (**R1**: WAR), **I6** → **I2** (**R1**: WAR), **I6** → **I5** (**R1**: WAR)
- **I7** → **I3** (**R2**: WAR)
- **I8** → **I7** (**R2**: RAW), **I8** → **I1** (**R3**: RAW)

Section 2 The following are the stalls in the execution:

```
        DADDUI R3, R1, #40  ;Solamente la primera vez
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        Stall
        Stall
        ADD.D F4, F0, F2
        Stall
        Stall
        Stall
        Stall
        S.D F4, 0(R1)
        DADDUI R1, R1, #8
        DADDUI R2, R2, #8
        BLE R2, R3, BUCLE
```

A cycle is required for the initiation code. Each iteration needs 13 cycles. As the loop is executed 5 times, we have a total of:

$$1 + 5 \cdot 13 = 66$$

Because there is no delay slot it is not necessary to add a stall cycle after the branch instruction (**BLE**).

Section 3 Next, the modified code:

```
        DADDUI R3, R1, #40 ;Only the first time
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        DADDUI R1, R1, #8
        DADDUI R2, R2, #8
        ADD.D F4, F0, F2
        Stall
        Stall
        Stall
        Stall
        S.D F4, -8(R1)
        BLE R1, R3, BUCLE
```

A total time of $1 + 5 \cdot 11 = 56$. Again, we need a cycle for the initiation code. Each iteration needs 13 cycles. Because there is no delay slot, it is not necessary to add a stall after the jump instruction (**BLE**).

$$\text{Speedup} = 66/56 = 1{,}17$$

Section 4 Next, the modified code:

```
        DADDUI R3, R1, #32  ;Only the first time
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        Stall
        Stall
        ADD.D F4, F0, F2
        Stall
        Stall
        Stall
        Stall
        S.D F4, 0(R1)
        L.D F6, 8(R1)
        L.D F8, 8(R2)
        Stall
        Stall
        ADD.D F10, F6, F8
```

```
            Stall
            Stall
            Stall
            Stall
            S.D F10, 8(R1)
            DADDUI R1, R1, #16
            DADDUI R2, R2, #16
            BLE R2, R3, BUCLE
            L.D F0, 0(R1)
            L.D F2, 0(R2)
            Stall
            Stall
            ADD.D F4, F0, F2
            Stall
            Stall
            Stall
            Stall
            S.D F4, 0(R1)
```

In total, we now have 2 iterations inside the loop. In addition, the fifth iteration of the original loop is now done at the end. The time required is:

$$1 + 2 \cdot 23 + 10 = 57$$

If you also re-schedule the instruction, you can have:

```
            DADDUI R3, R1, #32   ; Only the first time
BUCLE:      L.D F0, 0(R1)
            L.D F2, 0(R2)
            L.D F6, 8(R1)
            L.D F8, 8(R2)
            ADD.D F4, F0, F2
            Stall
            ADD.D F10, F6, F8
            DADDUI R1, R1, #16
            DADDUI R2, R2, #16
            S.D F4, -16(R1)
            Stall
            S.D F10, -8(R1)
            BLE R2, R3, BUCLE
            L.D F0, 0(R1)
            L.D F2, 0(R2)
            Stall
            Stall
            ADD.D F4, F0, F2
            Stall
            Stall
            Stall
            Stall
            S.D F4, 0(R1)
```

The new time is:

$$1 + 2 \cdot 13 + 10 = 37 \text{ciclos}$$