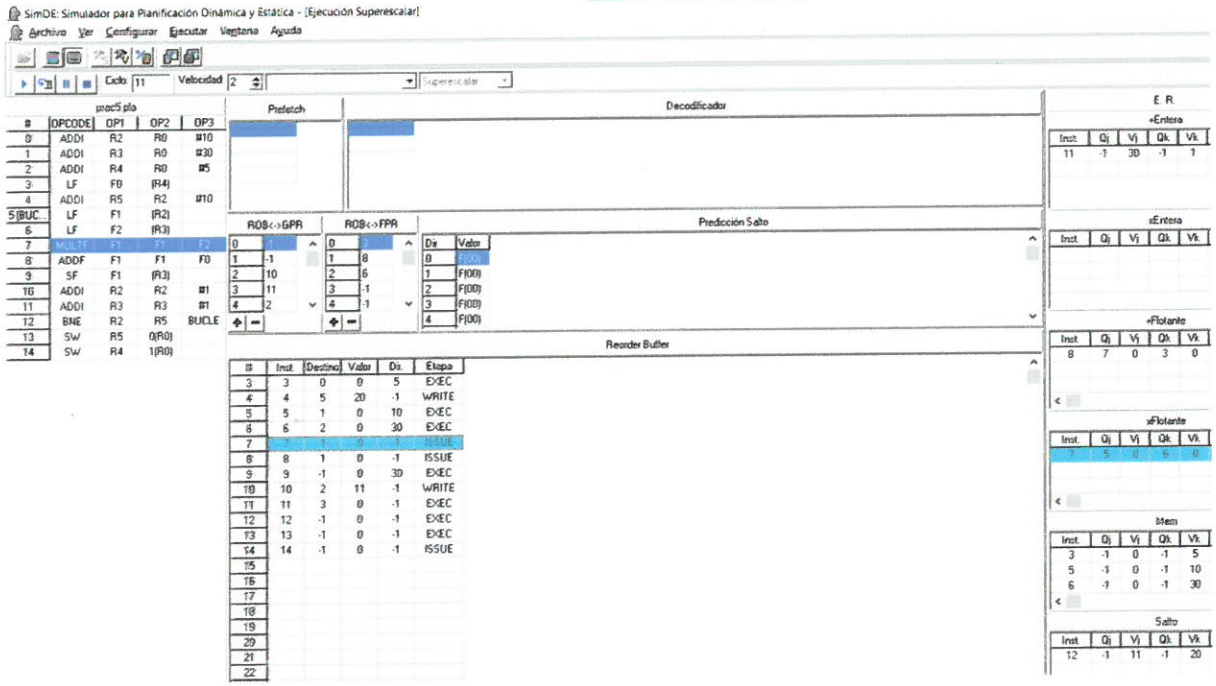
Análisis de las columnas en la ER:



1. **Columna `Qj` (Etiqueta de Dependencia del Operando 1)**:
   - El valor de `Qj = 5` indica que el operando `j` aún está pendiente de ser producido por una instrucción previa (la instrucción 5 en este caso). Esto implica que no está listo para ejecutar porque depende de los datos generados por esa instrucción.
2. **Columna `Qk` (Etiqueta de Dependencia del Operando 2)**:
   - El valor de `Qk = 0` indica que el operando `k` no tiene dependencias y está listo para usarse.
3. **Columna `Vk` (Valor del Operando 2)**:
   - El valor `Vk = 6` confirma que el operando `k` tiene un valor disponible, lo cual corrobora que no hay problemas con este operando.
4. **Columna `Vj` (Valor del Operando 1)**:
   - El valor `Vj = 0` indica que el operando `j` no tiene un valor disponible porque depende de la instrucción identificada en `Qj` (la instrucción 5).
5. **Columna `A`**:
   - Aunque no es directamente relevante en este caso, el valor `A = -1` puede indicar una dirección de memoria o desplazamiento pendiente, lo cual no afecta la ejecución actual de esta instrucción.

LAB ASSIGNMENT 3
STATIC CODE SCHEDULING

Objectives

- Understand the *loop unrolling* technique.
- Understand the *software pipelining* technique.

Procedure

The student must apply different static code scheduling techniques and evaluate by simulation the performance of the resulting code, assuming pipelined multicycle processors.

Consider the SAPY or DAPY operation (Single or Double precision A Plus Y), consisting in computing Z = a + Y, where Y and Z are vectors stored in memory, and *a* is a scalar value.

A possible implementation for the *Simula3MS* simulator is given in Appendix A. Configure the simulator (version 4.12!) with 4 floating-point addition units with a latency of 4 cycles and leave the default configuration for the rest of functional units.

Then, do the following tasks and answer to the corresponding questions:

1. For the SAPY program:
   a. Check its behavior.
   b. Compute the CPI (clock cycles per instruction) and CPR (cycles per result) values.

2. Apply loop unrolling. Unroll the minimum number of times to eliminate any stalls.
   a. Check the behavior of the modified code.
   b. Compute the CPI and CPR values.
   c. Compute the obtained *speedup* (with respect the original code).

3. Apply software pipelining.
   a. Check the behavior of the modified code.
   b. Compute the CPI and CPR values.
   c. Compute the obtained *speedup* (with respect the original code).

1. la pseudo → 2 instruc
No syscall → CPI = Ciclos / Nº instruc = 780 / (8 + (64 − 7)) = 1'7105
781 − 1 syscall
pseudo
vueltas
↳ instruc
CPR = Ciclos / Vueltas vector = 780 / 64 = 12'1875
↳ 243'75 nseg

APPENDIX A

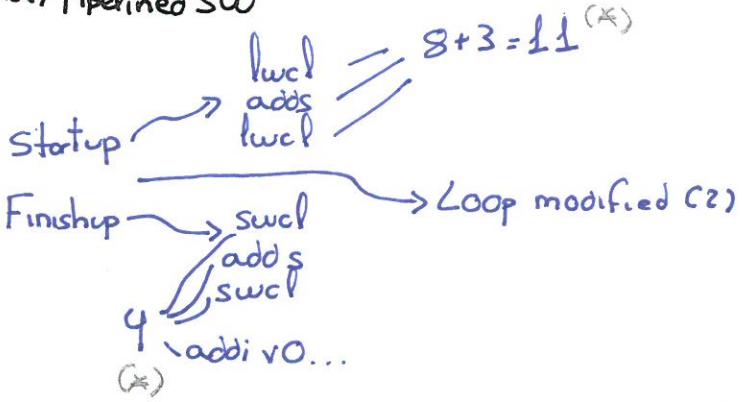
```
# SAPY → z = a + y

# data segment
        .data

        # Vector size = 64 words = 256 bytes

        # Vector y
y:      .float 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
59, 60, 61, 62, 63

        # Vector z
z:      .space 256

        # Scalar a
a:      .float 1

# text segment
        .text
        .globl main

main:
        la $t1, y           # $t1 contains the address of y
        la $t2, z           # $t2 contains the address of z
        la $t3, a           # $t3 contains the address of a

        lwc1 $f0, 0($t3)    # f0 contains a
        addi $t4, $t1, 256  # 64 elements are 256 bytes

loop:   lwc1 $f2, 0($t1)
        add.s $f4, $f0, $f2
        swc1 $f4, 0($t2)
        addi $t1, $t1, 4
        addi $t2, $t2, 4
        bne  $t4, $t1, loop

        addi $v0, $0, 10    # end
        syscall
```

2. 780
lwc1 addi add.s addi swc1 bne → 588 ↳ no depend.
lwc1 addi adds addi bne swc1 → 461 ↳ Haystalls

Loop unrolling:
lwc1 f2, 0(t1)
lwc1 f12, 4(t1)
adds f4, f0, f2
adds f14, f0, f12
addi t1, t1, 8
addi t2, t2, 8
swc1 f4, −8(t2)
bne t4, t1, loop
swc1 f4, −4(t2)

CPI = 301 / (8 + (32 · 9) + 1) = 1'0135
↳ add
64/2
CPR = 780 / 64 = 4'7031
Speedup = Ciclos0 / CiclosNUEVO = 780 / 301 = 2'591

3. Segmentación / Pipelined SW



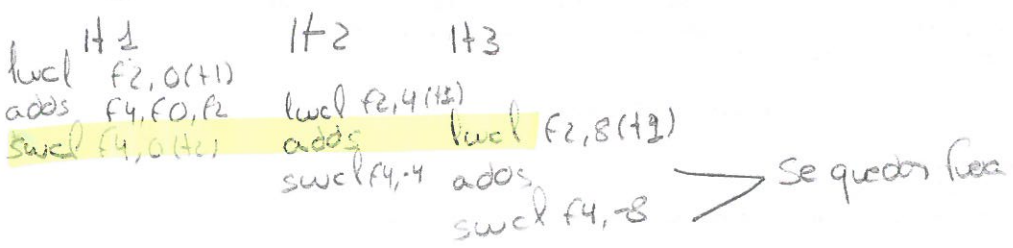
Modified loop → No stalls
↘ salto retardado → problemas??

$$CPI = \frac{397}{11 + (62 \cdot 6) + 4} = 1'05685$$

(A) (A)

$$CPR = \frac{397}{64} = 6'3906$$

$$Speed\text{-}up = \frac{780}{397} = 1'9071$$



QUESTIONS:

---

1. History and Context

Q: What is the historical significance of the Intel 8086 microprocessor?
A: The Intel 8086, released in 1978, is significant for being the foundation of the x86 series of processors, which have become a standard in personal computers. It paved the way for modern microprocessor designs.

Q: How does the Meteor Lake architecture represent a shift for Intel?
A: Meteor Lake introduces a chiplet-based design, moving away from the traditional monolithic design. This allows for better modularity, scalability, and integration of AI capabilities.

---

2. General Characteristics

Q: What is the purpose of Performance (P) and Efficiency (E) cores in the hybrid architecture?
A: P-cores handle demanding, intensive tasks such as 3D rendering and gaming, while E-cores optimize background and low-priority tasks, improving overall efficiency.

Q: How does the Foveros 3D stacking technology contribute to processor performance?
A: Foveros 3D allows for stacking chiplets vertically or horizontally, optimizing each chiplet for its specific function, which enhances performance and energy efficiency.

---

3. AI Capabilities

Q: What role does the Intel Neural Engine play in AI applications?
A: It accelerates AI tasks directly on the processor, improving applications like machine learning, deep neural networks, and natural language processing.

Q: What are the advantages of Neural Compute Engine (NCE) tiles?
A: NCE tiles enable high computational throughput for deep learning applications, supporting both 16-bit floating point and 32-bit brain floating point formats.

---

4. Integrated Graphics and Connectivity

Q: What advancements do Intel Arc Graphics provide?
A: Intel Arc Graphics deliver high performance for gaming and multimedia tasks, offering significant improvements in image and video processing.

Q: Which connectivity technologies are supported by Intel Core Ultra processors?
A: These processors support Wi-Fi 6, Wi-Fi 7, and Thunderbolt 4, providing high-speed connectivity and reduced latency.

---

5. Architecture and Internal Organization

Q: How does the Intel Thread Director optimize performance in hybrid architectures?
A: It dynamically assigns tasks to P-cores and E-cores based on workload requirements, ensuring energy efficiency and better task prioritization.

Q: What is the significance of using Intel 4 manufacturing technology?
A: Intel 4 enables higher transistor density and better performance, facilitating the integration of hybrid architecture and advanced features like AI and integrated graphics.

---

6. Memory Hierarchy

Q: Describe the role of L1, L2, and L3 caches in the memory hierarchy.
A:

- L1 is the smallest and fastest cache for frequently accessed data.
- L2 is larger but slower and supports either single or multiple cores.
- L3 is the largest and slowest, shared across all cores to reduce latency in accessing the main memory.

Q: How does the Intel Smart Cache benefit multi-core processing?
A: It allows shared access among cores, enhancing data exchange and reducing memory access latency, particularly in multi-threaded workloads.

---

7. Performance and Efficiency

Q: What metrics are used to evaluate processor performance?
A: Metrics include execution time, clock cycles per instruction (CPI), and benchmarks like Cinebench and 7-Zip performance.

Q: How does Intel Adaptive Boost Technology improve processor performance?
A: It dynamically increases core frequencies under favorable thermal and power conditions, optimizing performance for demanding tasks like gaming.

---

8. Comparisons and Benchmarks

Diapo 17

Diapo 18

ILP. This technique enables multiple instructions to execute simultaneously within a single CPU cycle.

Speculative Execution, allows the processor to predict and execute instructions before they are confirmed to be necessary.

Key features supporting both:
-Intel Thread Director which dynamically allocates workloads to the most suitable cores, either P or E cores.
And the classics: BTB, GL predictors...
Present in previous intel architectures (no 100% seguro, se supone en base a que estaban antes)

Diapo 19

Cache Strategies, as mentioned before Intel Core Ultra processors use a hierarchical cache system
-L1 dedicated to each core,
-L2 cache shared diff with p-cores and e-cores
-L3 shared across all cores

Dynamic Resource Allocation, The hybrid architecture, supported by the Intel Thread Director, along the integration of Intel Adaptative Boost for adjusting clock speeds.

Diapo 20

Diapo 21

Lower Clock Cycles per Instruction (CPI) are achieved through parallel execution and optimized caching. The graph illustrates how the new architecture design enables higher clock frequencies with lower voltage.

Diapo 22

The Intel Neural Engine is a major innovation in Core Ultra processors compared to previous architectures.Unlike traditional reliance on external GPUs or TPUs, the Neural Engine brings on-chip AI processing. By integrating AI capabilities directly into the CPU, it accelerates workloads, reduces latency, and eliminates the need for separate accelerators in general-purpose applications.

Diapo 23

Intel 4 nodes are built thanks to Intel's advanced manufacturing process. These nodes are foundational to the Core Ultra processors, offering higher transistor density and greater energy efficiency.
And here you can see the blueprint of the previous generation and the next ones that intel has in mind for the chips.

Diapo 24

And finally the Dynamic Workload Distribution as already mentioned before, using the Hybrid Architecture.
P-cores focus on single-threaded tasks, delivering high performance for compute-intensive applications, while E-cores handle multi-threaded and background processes efficiently.

Hybrid arch. again

Multi-Thre
E E
E E
Single-Thre
P

Diapo 25

Last but not least we will take a look at some benchmarks like this one, Cinebench 2024 that uses Redshift to benchmark CPU and GPU performance.
-In multi-core performance, the Intel Core Ultra 9 285K leads with a score of 2523 using DDR5-8200 memory, surpassing competitors like the AMD Ryzen 9 9950X at 2345.
-In single-core performance, the Intel Core Ultra 9 285K again takes the lead with a score of 150, followed closely by the Core Ultra 7 265K at 145

Redshift CPU+GPU perform.

Diapo 26

Cinebench 2024 power efficiency. The Intel Core Ultra 7 265K demonstrates impressive efficiency, delivering performance comparable to the AMD Ryzen 7950X while consuming 29% less power than the Intel 14700K.
However AMD is even more power-efficient, specially the AMD's Ryzen 7950X3D consuming 33% less while offering similar performance to the 265K.

Power consump.

Diapo 27

7-Zip benchmark evaluate CPU performance and memory efficiency using the LZMA (Lempel-Ziv-Markov chain algorithm) algorithm for compression and decompression tasks.

In file compression, the Intel Core Ultra 9 285K performs strongly, achieving 212 MIPS with DDR5-8200 memory, closely trailing AMD's top models like the Ryzen 9 9950X and 7950X. However, the Core Ultra 7 265K lags slightly, performing 6% slower than the Intel 14700K.

In decompression tasks, the Core Ultra 9 285K remains competitive, scoring 203 MIPS, just below AMD's leading Ryzen 9 7950X3D at 205 MIPS. The 7 265K, however, shows its limitations without SMT support, falling 15% behind the 14700K.

LZMA algo en MIPS

- Compatibility with advanced technologies: Allows to integrate the Intel Thread Director

INTEL ADAPTIVE BOOST

It is designed to maximize the performance of gaming and demanding applications while administrating energy limitations.
How:
1. Dynamical increase of frequencies of the cores
2. Advanced thermal management: Administers the digital sensors to avoid overheating

SUPPORT DDR5 AND PCIe 5.0
DDR5: → Better Memory Accesses, Out of order scheduling
It improves speed and bandwidth allowing transfer up to 5600 MT/s
Brings a flexible configuration which gives us compatibility with dual channels that improves the latency and the bandwidth. Including advanced technologies such as Data Scrambling and Out-Ot-Order Scheduling.
PCIe 5.0:
Gives us the ability of scalability and compatibility, this processor admits up to 20 PCie entries. Allowing also the integration and flexibility with advanced configurations such as Lane Bifurcation.

INTEGRATED GRAPHICS PROCESSING UNITS
Based on the Xe architecture and designed to offer a high graphic performance without a specific GPU.
Main characteristics:
- Scalar Xe Architecture
- Advance support for graphic APIs
- Accelerated codification and decodification by hardware
- Improvements in multimedia processing
- Higher video and games performance
- Low energetic cost

Diapo 11

Internal Organization: Cores and Threads

Intel Core Ultra processors feature a hybrid architecture with multiple P-cores (high-performance) and E-cores (energy-efficient).

1. **Caches**: Three levels (L1, L2, L3) balance speed and size, reducing latency and enhancing performance.
2. **Integrated Controllers**: Built-in memory and I/O controllers streamline communication with memory and peripherals.
3. **Security and Encryption**: Technologies like Intel SGX protect data and applications.(Hablar de encriptación en Hardware sin tocar software)
4. **AI Modules**: Integrated AI modules that optimize tasks like natural language processing and image improve..

Diapo 14

Memory Hierarchy and Technology optimizes data access by organizing memory from faster, smaller caches near the CPU to slower, larger ones further away. Intel Core Ultra processors include integrated memory controllers that handle data transfers and DRAM maintenance independently. Each controller manages half the physical memory address space in symmetric configurations, scaling bandwidth with increased channels to enhance efficiency.

Cache Memory Levels:

- **L1 Cache:** Smallest, fastest, dedicated to individual cores for frequently accessed data, reducing latencies.
- **L2 Cache:** Larger, slower, shared or dedicated, storing reused task data efficiently.
- **L3 Cache:** Largest, slowest, shared across cores, bridging lower cache levels and RAM to reduce direct memory access.

Diapo 15

Intel Smart Cache serves as a shared, non-inclusive L3 cache for all AI cores and Processor Graphics. **Adaptive Memory Technology** tailors cache design for task-specific optimization:

Ver presentación

- **P Cores:** Isolated L1/L2 caches per core for high-performance tasks, offering low latency.
- **E Cores:** Shared L2 caches across four cores, balancing energy efficiency and parallelism.

In both designs, L3 cache is shared across all physical cores, ensuring seamless data flow.

Diapo 16

Enseñar un poquito la idea de los Cores + Caches

Diapo 3

1-Historia:

Intel biggest manufacturer of integrated circuits.
Created the x86 architecture, all based in 1978 designs.
Introduce in biggest part la Meteor Lake y el Foveros 3D, ways of connecting and stacking components.
Focus en la IA, CPU functions, more efficiency and performance.
Also in security, with KeyLocker, that is encryption but in hardware, not touching software.

Diapo 4

Diapo 5

2-Características

Core composition: Hybrid core for efficiency and performance. E-Cores for background tasks and P-core for high demand tasks

Architecture and technology: Meteor Lake y el Foveros 3D para modular stacks, aumenta escalabilidad reduce costes

AI Capability: Accelerated with Intel Neural Engine, with subsystems for processing, hosting and computation. All related to power and performance with DNNs and DMA engines

---

Diapo 6

Integrated graphics: Featured used specially for gaming and video processing.

Efficiency and power management: Optimizes CPU, GPU and thermal performance for portable devices with Adaptive Boost. Better hardware life..

Memory and IO/s: DDR5 and LPDDR5 supported, the second one for mobiles and laptops

C
A
AI

Diapo 7

Enhanced connectivity: Supports Wi-Fi 6, 7 and thunderbolt. High speed, reduced latency, more security and efficiency. Stable performance

Target Market: High-end laptops and ultrabooks due to thermal solutions, exceptional performance and energy efficiency. For gaming, content creation and computational tasks

---

(De la 8 a la 10 leer sin más)

Diapo 10

CHARACTERISTICS OF THE ARCHITECTURE AND INTERNAL ORGANIZATION

One of the most important characteristics is the hybrid architecture
If we talk about this hybrid architecture we have to differentiate between:

- **P-cores**: Are designed to carry out tasks that require high performance and they are optimized for intense workloads in only one thread. These cores are based on **Lion Cove** microarchitecture.
- **E-cores**: They produce a low energetic cost. They are designed for scalable multi-threaded performance, offloading background processes and low-priority tasks. They are based on **Skymont** microarchitecture.

P-Cores = Lion Cove Microarch

E-Cores = Skymont microarch

INTEL THREAD DIRECTOR TECHNOLOGY

Improves the assignment of tasks in a system where we have to distribute the workload between the P-cores and E-cores. For this tasks we need to take into account some topics:

- Task prioritization: studies nature of the tasks
- Intelligent distribution: Based on the study
- Dynamical organization: Reorganizes loads depending on changing conditions
- Collaboration with the operating system

E vs P??
Dinamic task dist. + OS

MANUFACTURING PROCESS

These processors of the PS series are manufactured using Intel 4, based in lithography of 10nm or smaller.
This brings different advantages for the process of fabrication:

- High density of transistors: Allowing the optimization of the space and increase performance.
- Energetic efficiency: Smaller transistors reduce the amount of energy required
- Support for new architectures: Support hybrid architecture.

Q: How does Intel Core Ultra compare to previous architectures in benchmarks?
A: Intel Core Ultra processors outperform previous architectures in both single-core and multi-core tasks, with improvements in energy efficiency and AI integration.

Q: What makes Intel Core Ultra processors suitable for gaming and content creation?
A: Their hybrid architecture, advanced AI capabilities, integrated Intel Arc Graphics, and dynamic resource management make them ideal for high-performance applications.

Feel free to use or adapt these questions during an explanation session!

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2022)	Test	Year 2021/22

Name: _____ **Aula:** ___ **Mesa número:** ___

Indicate the correct answer to each test question in the next table:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Important: Test counts **3 points**. Each correct question counts 0.1 points. Wrong answers subtract 0.1/3 points. Not answered questions are not considered for the mark.

1. A current smartphone can exploit:
   a) Thread-level parallelism
   b) Instruction-level parallelism
   c) CPU-level parallelism
   d) All the above

2. According to the Flynn's taxonomy, GPUs are:
   a) SISD computers
   b) MISD computers
   c) SIMD computers
   d) MIMD computers

3. We want to analyze the performance of two computers, M1 and M2, which support the same instruction set. In this instruction set there are three types of instructions, namely, A, B, and C. M1 needs 3, 6, and 4 clock cycles to execute instructions of type A, B, and C, respectively. M2 needs 4 cycles for all types. M1 runs at 400 MHz and M2 runs at 750 MHz. The test program we will use has the following mix of instructions: 40% of type A, 50% of type B, and 10% of type C. Indicate the peak speed in MIPS for M1 and M2.
   a) 133.3 for M1 and 187.5 for M2
   b) 87 for M1 and 187.5 for M2
   c) 125 for M1 and 155 for M2
   d) None of the above

4. If the hard disk of a computer is replaced by another disk six times faster than the current one, calculate the use that should be made of this device if we want to make the computer run three times faster
   a) It is not possible
   b) The disk must be used 20% of the time
   c) The disk must be used 80% of the time
   d) None of the above

5. Consider an iteration of the next loop:

```
LOOP:  LW      R1, 0(R2)      ; I1
       DADD    R1, R1, R3     ; I2
       SW      R1, 0(R2)      ; I3
       DADDUI  R2, R2, #4     ; I4
       DADDUI  R4, R4, #-1    ; I5
       BNEZ    R4, LOOP       ; I6
```

   a) There is a data dependence between I3 and I4
   b) There is an antidependence between I3 and I4
   c) There is an output dependence between I2 and I3
   d) There is an output dependence between I5 and I6

6. Consider two consecutive iterations of the following loop:

```
LOOP:  LW      R1, 0(R2)      ; I1
       DADD    R1, R1, R3     ; I2
       SW      R1, 0(R2)      ; I3
       DADDUI  R2, R2, #4     ; I4
       DADDUI  R4, R4, #-1    ; I5
       BNEZ    R4, LOOP       ; I6
```

   a) There is a data dependence between I5 and I6
   b) There is an antidependence between I1 and I2
   c) There is an antidependence between I1 and I3
   d) There is an output dependence between I1 and I3

7. In a pipelined processor implementing forwarding and using pipelined FP functional units, a *latency* of 3 cycles between a FP addition and an instruction using the result of this operation implies that…
   a) The instruction using the result must be stopped for 2 cycles
   b) The instruction using the result must be stopped for 3 cycles
   c) The execution step of the FP addition takes 2 cycles
   d) The execution step of the FP addition takes 3 cycles

8. An antidependence can lead to a:
   a) WAW hazard
   b) RAW hazard
   c) WAR hazard
   d) None of the above

9. Indicate the correct option:
   a) The Scoreboard technique solves RAW and WAW hazards
   b) The Scoreboard technique solves WAW and WAR hazards
   c) The Tomasulo technique (without speculation) solves RAW and WAW hazards
   d) The Tomasulo technique (without speculation) solves WAW and WAR hazards

10. Software pipelining:
    a) Is implemented by the compiler to reduce stalls due to WAR hazards
    b) Is implemented by the CPU to reduce stalls due to WAR hazards
    c) Is implemented by the compiler and the CPU to reduce stalls due to WAR hazards
    d) None of the above

11. The loop unrolling technique:
    a) Is a dynamic technique to schedule the instructions and increase the ILP
    b) Is a static technique to schedule the instructions and reduce the CPI
    c) Is a static speculative technique to schedule the instructions and increase the ILP
    d) Is a static technique to schedule the instructions and reduce the size of the object code

12. We have a one level 2-bit branch predictor initialized to strongly not taken. Indicate the correct sequence of hits (H) and misses (M) for B1 in the next code:

```
start:  DADDIU R2, R0, #6
        DADDIU R4, R0, #1
loop:   DADDIU R2, R2, #-1
        BNEZ   R2, loop          ; B1
        DADDIU R4, R4, #-1
```

    a) HHMMMM
    b) HHHHHM
    c) MMHHHM
    d) MMHHHH

13. Indicate the width and the number of bits to implement an implicit BTB with 1024 entries, knowing that the address bus uses 24 bits (there are several solutions, but only one of the options is correct).
    a) Width = 24, Number of bits = 1024×24 bits
    b) Width = 1024, Number of bits = 1024×1024 bits
    c) Width = 24×24, Number of bits = $1024 \times 24 \times 24$ bits
    d) Width = 24, Number of bits = $2^{24} \times 24$

14. What is the value of a 4-input BHT knowing that we have just executed a loop 100 times and now continue executing the following instructions sequentially?
    a) 1111
    b) 1110
    c) 0101
    d) 0011

15. Indicate the correct option for the multipath execution technique:
    a) Both paths of a conditional branch are executed, and the instruction is executed depending on the value of a predicate register
    b) One of the paths of a conditional branch is chosen to be executed by using prediction
    c) Both paths of a conditional branch are executed, without using prediction
    d) One of the paths of a conditional branch is randomly chosen to be executed

16. Software speculation…
    a) Is supported by hardware (dynamic) branch prediction
    b) Speculative instructions resulting in terminating exceptions always produce program termination
    c) Is based on the Scoreboard algorithm
    d) None of the above

17. In the poison bits technique, if a normal (non-speculative) instruction results in a terminating exception:
    a) The execution of the program continues, but the poison bit of its destination register is set
    b) The execution of the program continues, but the poison bit of its destination register is reset
    c) The execution of the program is stopped
    d) None of the above

18. In the *commit* step of the speculative version of Tomasulo…
    a) Instruction results are provided to other instructions that need them
    b) The ROB is employed to complete the execution of instructions according to the program order
    c) If the instruction to be completed is a branch, the destination register (or the memory) is updated
    d) None of the above

19. A grade 3-way superscalar has an ideal CPI of
    a) 3
    b) 1.3
    c) 1/3
    d) 1

20. Regarding the branch prediction in a superscalar:
    a) It is carried out in the fetch stage.
    b) It is carried out in the decode stage.
    c) It is carried out in the issue stage.
    d) It is carried out in the commit stage.

21. One of the drawbacks of superscalars is:
    a) They are not compatible at the binary level with a scalar machine.
    b) They introduce many NOP operations in the pipeline.
    c) The hardware is so simple that they have problems due to branches.
    d) There is a higher probability of risks in instructions that are launched at the same time.

22. Indicate which of the following statements about VLIW processors is false:
    a) Since they use static scheduling, they are highly sensitive to cache misses.
    b) They have a register bank that is complex in area and access time.
    c) They duplicate hardware resources to be more efficient.
    d) If the compiler is not efficient the size of the machine code can be exceptionally large.

23. Current multicore architectures are:
    a) Distributed memory computers
    b) Shared memory computers with uniform access time
    c) Shared memory computers with non-uniform memory access
    d) Distributed cache memory computers

24. The P7 microarchitecture (or Netburst) is based on:
    a) An evolution of the P6 microarchitecture aimed at increasing the number of instructions to be processed per cycle
    b) Increasing clock frequency
    c) The P6 microarchitecture, keeping the decoder structure
    d) The Pentium Pro architecture, but with Xeon improvements

25. In the heterogeneous computing paradigm:
    a) The CPU performs general purpose computing
    b) Computing systems become a hybrid structure
    c) Specific applications are run on accelerators
    d) All the above

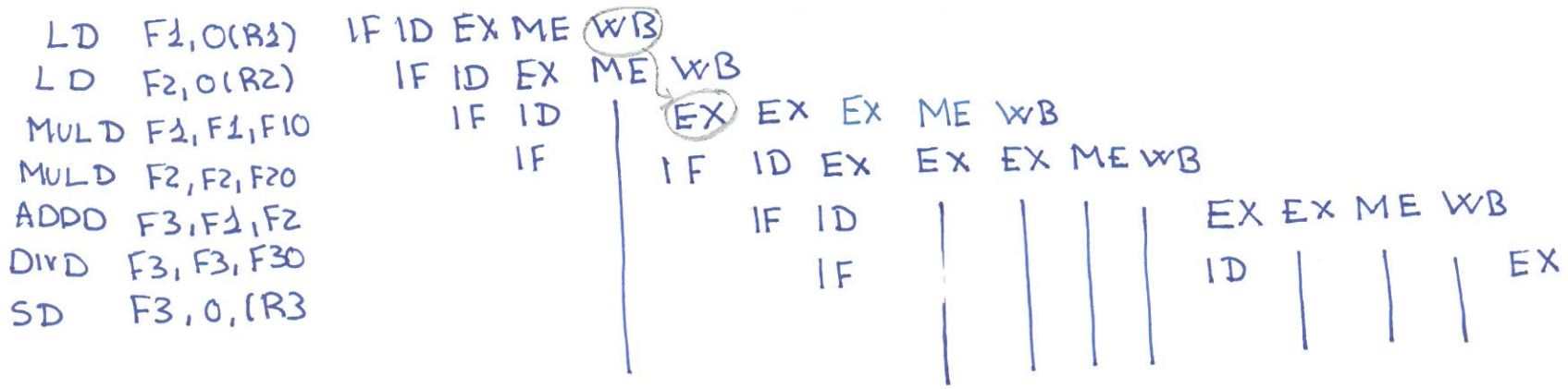
26. On multicore processors with HyperThreading, better execution times for two applications are achieved by using:
    a) The same core for both applications
    b) A different core for each application
    c) Depends on the number of threads per core allowed by the system
    d) None of the above

27. A NUMA system:
    a) Uses a shared memory
    b) Uses the Internet to connect its processors
    c) Has a variable memory access latency
    d) None of the above

28. Indicate the correct statement:
    a) A SIMD architecture is less energy-efficient than a MIMD architecture
    b) The MMX extensions on Intel processors are MISD extensions
    c) There is not penalty due to context switching in GPUs
    d) None of the above

29. Skype or Gmail are examples of:
    a) SaaS Cloud
    b) PaaS Cloud
    c) IaaS Cloud
    d) JaaS Cloud

30. MPI is:
    a) A software product sold by Microsoft
    b) A message-passing library specification
    c) A compiler directive standard
    d) All the above


LD F1,0(R1)
LD F2,0(R2)
MULD F1,F1,F10
MULD F2,F2,F20
ADDD F3,F1,F2
DIVD F3,F3,F30
SD F3,0,(R3
IF ID EX ME WB
IF ID EX ME WB
IF ID EX EX EX ME WB
IF ID EX EX EX ME WB
IF ID EX EX ME WB
IF ID EX
IF

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2022)	Exercises	Year 2021/22

Name: _____

Important: Solve this exercise **on this sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 3 (2.5 points). Consider a 4-way superscalar processor capable of issuing four instructions of any type per cycle, with five functional units (F1, F2, F3, F4, and F5). Assume that instructions require one cycle for the decode stage, one cycle for the execute stage and one cycle for the write stage. The result of an instruction is available to other instructions at the next cycle in which that result is written.

Given the next code:

```
LW   R1, 0(R2)       #I1
DADD R4, R1, R3      #I2
DSUB R6, R7, R8      #I3
LW   R3, 0(R9)       #I4
DSUB R10, R10, R12   #I5
DADD R8, R8, R1      #I6
DSUB R3, R3, R1      #I7
```

Draw a diagram showing, for each clock cycle, which instructions are in the decode step (four at most), in the execute step (five at most), and in the write step (four at most), for each of the following situations:

a) In-order execution and in-order writing.
b) Out-of-order execution, without renaming capacity, and in-order writing.
c) Out-of-order execution with renaming (assuming enough registers for this), and out-of-order writing.

Use the following tables to provide your solutions, detailing the instruction status at each cycle.

a) In-order execution and in-order writing.

Cycle	Decode				Execute					Write			
1	I1												
2					I1								
3	I2	I3								I1			
4					I2	I3							
5	I4	I5								I2	I3		
6					I4	I5							
7										I4	I5		
8													
9													

b) Out-of-order execution, without renaming capacity, and in-order writing.

Cycle	Decode				Execute					Write			
1	I1												
2	I2				I2								
3	I3	I4	I5	I6							I1		
4					I2	I3	I4	I5	I6				
5	I7									I2	I3	I4	I5
6					I7					I6			
7										I7			
8													
9													

c) Out-of-order execution with renaming (assuming enough registers for this), and out-of-order writing.

Cycle	Decode				Execute					Write			
1	I1	I2	I3	I4									
2	I5		I6		I1	I3	I4						
3	I7				I5	I6				I1	I3	I4	
4					I7	I2				I5	I6		
5										I7	I2		
6													
7													
8													
9													

⚠ Dos instrucciones no pueden leer el mismo registro a la vez

⚠ Cuidado con sobreescribir registros antes (o a la vez) de que se usen

No hay que esperar que se libere la FU

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2022)	Exercises	Year 2021/22

Name: _____

Important: Solve this exercise **on this sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 1 (2.5 points). The next loop is executed on a MIPS64 pipelined processor to compute the weighted mean of the marks obtained by a group of Computer Architecture students in two exam exercises. Assume that these marks have been previously stored (as double-precision floating-point values) from the memory addresses contained in R1 (exercise 1 marks) and R2 (exercise 2 marks) and that the resulting mean values must be stored from the memory address contained in R3, R4 contains the number of student marks, F10 and F20 contain the weight assigned to each exercise, and F30 has been initialized to the sum of the weights.

```
Loop: L.D    F1, 0(R1)    // gets the mark from exercise 1
      L.D    F2, 0(R2)    // gets the mark from exercise 2
      MUL.D  F1, F1, F10  // weights the mark from exercise 1
      MUL.D  F2, F2, F20  // weights the mark from exercise 2
      ADD.D  F3, F1, F2
      DIV.D  F3, F3, F30  // obtains the weighted mean value
      S.D    F3, 0(R3)    // stores the result
      DADDUI R1, R1, #8   // points to the next mark from exercise 1
      DADDUI R2, R2, #8   // points to the next mark from exercise 2
      DADDUI R3, R3, #8   // points to the next weighted mean
      DADDUI R4, R4, #-1
      BNEZ   R4, Loop     // more student marks?
```

The processor exhibits the usual characteristics: a value can be written and read from a register in the same clock cycle, dependences are detected and solved in the ID stage, operand forwarding is supported, branches are completely solved in the ID stage, the integer functional unit requires 1 cycle, and the execution step of L.D and S.D requires 1 cycle. Functional units for floating-point addition, multiplication, and division are fully pipelined, and require, respectively, 2, 3, and 5 cycles.

a) Detail the stalls (be careful: NOT DEPENDENCES) required to execute each iteration of the original loop and compute the CPR value.
b) Unroll the loop the minimum number of times to schedule it without stalls. Assume that delayed branch is supported. Keep in mind that the program behavior must be preserved.
c) Compute the CPR value for the resulting loop (after applying loop unrolling and scheduling).

NOTE: The loop computes the weighted mean for each pair of student marks (x_1 and x_2) by using the expression $\bar{x} = \frac{x_1 \times w_1 + x_2 \times w_2}{w_1 + w_2}$, where w_1 and w_2 are the weights considered for exercise 1 and exercise 2.

LD
LD
Stall
MUL
MUL

IF ID EX ME WB
F IDEX ME WB
IF ID EX ME WB
IF ID EX ME WB
IF ID

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2022)	Exercises	Year 2021/22

Name: _____

Important: Solve this exercise **on this sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 2 (2 points). We assume a MIPS64 processor using hardware speculation and dynamic scheduling using the Tomasulo technique with speculation, with the following characteristics:

- The load and store execution stage takes 1 cycle.
- The floating-point multiplication (FP) execution step takes 6 cycles.
- The floating-point addition (FP) execution step takes 4 cycles.
- A 4-input reorder buffer (ROB) is available.
- It has a 2-entry reservation stations for each unit in FP.
- It has a 2-entry load/stage input load buffer.
- The execution stage of an instruction that is waiting for a result cannot start in the same cycle in which the result is written to the CDB (Common Data Bus).
- A resource that is released in one clock cycle cannot be used until the next clock cycle.

a) Detail the execution of the code below when the basic version of Tomasulo's algorithm with speculation is used. To do so, indicate in the table below the start and end cycle number for each of the steps of each instruction (NB: For 1-cycle instruction steps, you must annotate just 1 cycle in the table).

Instruction	Issue	Execution	Write result	Commit
MUL.D F4,F6,F12	1	2-7	8	9
SUB.D F6,F4,F10	2	9-12	13	14
L.D F4,10(R2)	3	4	5	15
DIV.D F0,F14,F16	4	8-13 5-24	14	16
SUB.D F12,F14,F0	10	15-18	19	20
ADD.D F20,F28,F24	15(*)	19-22 16-19	23	24
L.D F0,10(R2)	16(*)	17	18	25
MUL.D F18,F28,F24	17(*)	18-23	24	26

+1 (*)
+2 No ROB space
+1

b) Complete the table below to display the contents of the ROB (which has only 4 entries), just in the same cycle in which the first L.D. instruction ends. (5)

Entry	Busy (yes/no)	Instruction	Status	Destination	Value
0	YES	MULD F4,F6,F12	Exec	F4	F6*F12
1	YES	SUBD F6,F4,F10	Wait Issue	F6	F4-M10
2	YES	LD F4,10(R2)	write	F4	MEM[10+R2]
3	YES	DIVD F0,F14,F16	wait Issue	F0	F14/F0

UCLM	SCHOOL OF COMPUTER SCIENCE AND ENGINEERING COMPUTING SYSTEMS DEPARTMENT COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE	eii escuela superior de ingeniería informática
Final written exam (January 2023)	**Test**	**Year 2022/23**

Name: _____

Indicate the correct answer to each test question in the next table:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	A	C	B	D	D	B	C	B	A	C		A	

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Important: Test counts **3 points**. Each correct question counts 0.1 points. Wrong answers subtract 0.1/3 points. Not answered questions are not considered for the mark.

1. Synthetic benchmarks programs:
   a. They are specially designed to measure the performance of an individual component of a computer.
   b. They are specially designed to measure the performance of an entire computer.
   c. They are specially designed to measure the speed of access to a database system.
   d. None of the above.

2. The organization of a computer should be designed to:
   a. Survive for many years.
   b. Implement a particular architecture specification. → Ex: x86
   c. Implement several architectures.
   d. None of the above.

3. If a machine A executes a program in 30 seconds and a machine B executes the same program in 45 seconds, how much faster is A than B?
   a. A is 50% faster than B
   b. A is 150% faster than B
   c. B is 1.5 times faster than A
   d. B is 50% faster than A

4. Suppose a program takes 200 seconds to run on a machine, 150 seconds of which are for multiplication operations. How much do I have to improve the multiplication speed if I want the program to run twice as fast?
   a. 1.5 times
   b. 2 times
   c. 3 times
   d. None of the above

5. In a pipelined MIPS processor supporting forwarding, when executing two consecutive integer instructions:
   a. It is never necessary to stop the pipeline.
   b. It is necessary to stop if both instructions are arithmetic and present a data dependence.
   c. If the first instruction is a memory access, sometimes it is necessary to stop.
   d. None of the above.

6. In the following loop:

```
Loop: L.D    F8, 0(R1)      ;I1
      ADD.D  F6, F2, F8     ;I2
      S.D    F6, 0(R1)      ;I3
      DADDUI R1, R1, #-8    ;I4
      BNEZ   R1, Loop       ;I5
```

Between instructions I2 and I3…
   a. There is an antidependence between iterations
   b. There is an output dependence in the same iteration
   c. There is a data dependence between iterations
   d. There is an antidependence in the same iteration

7. The loop unrolling technique pursues...
   a. Reduce stalls due to WAR hazards
   b. Reduce stalls due to WAW hazards
   c. Reduce stalls due to structural hazards
   d. Reduce stalls due to control hazards → evito branches

8. In Scoreboard, instructions always get their operands:
   a. Through the common data bus (CDB)
   b. Through the register file (or register bank)
   c. By means of the forwarding technique
   d. From the reorder buffer (ROB)

9. In the Scoreboard technique:
   a. WAW and WAR hazards cannot appear
   b. WAW and WAR hazards may appear, but they are not controlled
   c. WAW and WAR hazards are checked in the first and last step, respectively
   d. None of the above → Issue check, WB check

10. The Qj and Qk fields in a reservation station in Tomasulo store:
    a. The value of the source operands needed by the corresponding instruction.
    b. The reservation stations that will produce the instruction source operands. → De las prácticas
    c. The registers that contain the instruction source operands.
    d. None of the above.

Qj	Qk
7	3

11. What is the latency of a branch instruction?
    a. The number of additional cycles between the IF stage and the cycle in which the branch is resolved.
    b. The number of additional cycles between the ID stage and the cycle in which the branch is resolved.
    c. The number of cycles between the IF and ID stages.
    d. The number of cycles between the IF and EX stages.

12. Assuming a conditional branch, the miss penalty in the prediction depends on
    a. The stage where the branch target address is known.
    b. The stage where the branch direction is known.
    c. a) y b) are true.
    d. All the above are false.

13. For a 5-stage pipeline, we run a code with 30% of instructions being a conditional branch, from which 80% are taken. In this processor, the branch direction is solved in the EX stage and the target address in the ID stage (the CPI is 1 for the rest of the instructions). Indicate the correct answer: → 2 penalty, IF, ID
    a. CPI = 1 + 3 x 0,8 x 0,3
    b. CPI = 1 + 2 x 0,8 x 0,3
    c. CPI = 1 + 3 x 0,2 x 0,7
    d. CPI = 1 + 2 x 0,2 x 0,7

14. Apart from storing prediction bits, what is the purpose of the branch target buffer (BTB) in the computer architecture?
    a. To store the branch target addresses of branch instructions.
    b. To reduce the latency cycles of branch instructions.
    c. Exception management.
    d. All are true.

15. For the software speculation (without hardware support), which of the following sentences is false?
    a. The compiler bets for a specific branch direction, moving instructions of that block to the former block.
    b. It is required to perform an efficient branch (static) prediction during the compiling time.
    c. During the execution of a program, the processor bets for a branch direction and the instructions of this block are executed speculatively. These instructions will be valid when the branch result is finally known.
    d. Renaming is used to allow the speculative execution of some instructions without modifying the data flow.

16. Let's assume we use software speculation when running a code. In the case that a speculative instruction causes a terminating exception, and the software speculation is not correct, we can say that:
    a. If we use the poison bits technique, the exception will always take place and the program will terminate in error.
    b. If we use the poison bits technique, the exception will only take place in some cases in which the program will terminate in error.
    c. Regardless the technique used to deal with exceptions, the exception will always take place and the program will terminate in error.
    d. None of the above.

17. In a superscalar, in the issue stage ....
    a. Name dependencies are removed by renaming.
    b. Instructions are read from the reorder buffer and sent to execution.
    c. An arbiter selects which instructions can be sent to execution.
    d. All are true.

18. A characteristic of dynamic superscalar processors is that they always ...
    a. Use dynamic scheduling.
    b. The number of instructions issued per cycle is fixed.
    c. Instructions are executed in order.
    d. All are true.

19. We have a VLIW processor that can issue four instructions per cycle. A compiler receives as input a program with 36 instructions and provides as output a program for this VLIW processor with an average efficiency of 3 operations per cycle. We can state that...
    a. 3 long instructions at the compiler output contain 4 NOP operations.
    b. 4 long instructions at the compiler output contain 3 NOP operations
    c. The minimum number of long instructions in the compiler output is 9.
    d. The minimum number of long instructions in the compiler output is 12.

20. Through the execution of a test program, we have obtained that the used superscalar has an ideal CPI of 0.25. What can you deduce from this statement?
    a. That the processor uses out-of-order execution but does not use speculation.
    b. That the processor uses in-order execution and speculation.
    c. That the processor has a frequency of ¼ MHz
    d. That the processor is a 4-way superscalar.

21. In a superscalar processor pre-decoding is performed:
    a. At the issue stage
    b. Between L2 and L1
    c. At the write-back stage
    d. None of the above is correct

22. Each iteration of a given loop in a scalar processor requires 10 clock cycles to be issued. We unrolled such a loop 8 times and scheduled it for a 4-way superscalar obtaining that we now perform 8 iterations in 10 cycles. What is the speedup obtained?
    a. 4
    b. 6
    c. 8
    d. 10

23. In the period 2004 to 2020:
    a. The number of transistors on a chip has increased exponentially.
    b. The working frequency of processors has remained almost stable.
    c. It has slightly increased the performance of a single thread.
    d. All are true

24. The architecture that is open source is:
    a. The x86 family
    b. ARM
    c. RISC-V
    d. All are false

25. The market sector where Intel processors are more dominant with respect to AMD is:
    a. Laptops
    b. Desktop
    c. Servers
    d. IoT

26. Cortex-M are ARM processors whose application scope is:
    a. Low power consumption for microcontrollers and IoT.
    b. Real-time for critical applications
    c. HPC
    d. Applications for computationally complex systems

27. A machine with tensor processors is an architecture according to Flynn's classification:
    a. SISD
    b. SIMD
    c. MISD
    d. MIMD

28. CUDA:
    a. It is C language with minimal extensions
    b. Uses a multitude of processing units and a single control unit
    c. Defines a programming model
    d. All are true

29. Which of the following statements is true?
    a. CLOUD is a collection of computers owned by multiple owners in different locations and interconnected so that users can share the combined power of resources.
    b. GRID is a collection of computers usually owned by a single party.
    c. Facebook is an example of CLOUD defined as SaaS (Software as a Service).
    d. None is true

30. A standard that by means of directives tells the compiler the region of the program to parallelize is
    a. OpenMP
    b. PVM
    c. MPI
    d. None is true.

Name: _____

Important: Solve each exercise **in a different sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 1 (2.5 points). Let's consider the following code in MIPS64. You need to assume that registers F2, F4, F6 and F8 are firstly initialized to 0, and that registers RA, RB and RC contain valid memory address.

```
            ADDI R1, R0, #0
            ADDI R2, R0, #3
LOOP1:      SLT R3, R1, R2
            BEQ R3, R0, END_L1        /* B1 */
            L.D F2, 0(RA)
            L.D F4, 0(RB)
            ADDI R4, R0, #0
            ADDI R5, R0, #4
LOOP2:      SLT R6, R4, R5
            BEQ R6, R0, END_L2        /* B2 */
            MUL.D F6, F2, F4
            ADD.D F8, F8, F6
            ADDI R4, R4, #1
            J LOOP2
END_L2:     S.D F8, 0(RC)
            ADDI R1, R1, #1
            ADDI RA, RA, #8
            ADDI RB, RB, #8
            ADDI RC, RC, #8
            J LOOP1
END_L1:                               /* Fin del programa */
```

You need to perform the following activities:

a) Fill the following table, bearing in mind the actual behavior of conditional Branch instructions B1 and B2:

Step	Executed branch instruction (B1 or B2)	R1 value	R4 value	Actual behavior of branch instruction (Taken/Not Taken)
1	B1	0	-	NT
2	B2	0	0	NT

(Use as many files as you need in a separate sheet)

b) Calculate the hit predictions rate for a 2-bit predictor, which is initialized to weakly not taken (WNT = 01). Use the same table of section "a" and add the corresponding columns to show the predictions at every step.

c) Calculate the hit predictions rate for a (2,2)-correlated predictor. All the data structures for this predictor are initialized to 0. Use a table like that of section "a" and show the values for data in these structures for every step in the table.

d) ¿Which predictor achieves the best performance for the assembly code in this exercise? Please, you need to justify your response.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2023)	Exercises	Year 2022/23

Name: _____

Important: Solve each exercise **in a different sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 2 (2 points). We assume a MIPS64 processor using hardware speculation according to the Tomasulo technique, with the following characteristics:

- The execution step of loads and stores takes 2 cycles.
- There are three fully pipelined Floating-Point (FP) functional units:
  - a FP adder (also available for subtracting) that requires 4 cycles,
  - a FP multiplier that requires 6 cycles, and
  - a FP divider that requires 12 cycles.
- A 4-input reorder buffer (ROB) is available.
- There is a 2-entry reservation station per FP unit.
- There is a 1-entry load buffer and a 1-entry store buffer.
- The execution stage of an instruction that is waiting for a result cannot start in the same cycle in which the result is written to the CDB (Common Data Bus).
- A resource (the ROB, a reservation station, etc.) that is released in one clock cycle cannot be used until the next clock cycle.

a) Detail the execution of the code below. To do so, indicate in the table below the start and end cycle number for each of the steps of each instruction **(1 point)**, and, when applicable, the reason to stall the execution of the corresponding step (a RAW hazard, a structural hazard, the program order, etc.) **(0.5 points)**.

	Instruction	Issue		Execution		Write result		Commit	
		cycle(s)	stall cause	cycle(s)	stall cause	cycle(s)	stall cause	cycle(s)	stall cause
1	MUL.D F4,F6,F2	1		2-7		8		9	
2	SUB.D F6,F4,F10	2		9-12	RAW	13		14	
3	L.D F4,0(R1)	3		4-5		6		15	ORDER
4	DIV.D F0,F4,F6	4		14-25	RAW	26		27	
5	L.D F8,0(R2)	16	Struct	17-18		19		28	ORDER (A)
6	SUB.D F2,F8,F0	17		27-30	RAW	31		32	
7	ADD.D F0,F8,F2	18		32-35	RAW	36		37	
8	S.D F0,0(R1)	28	(A)	37-38	RAW	39		40	

Pram: Sin espacio Pram: No reg.

b) Detail the contents of the ROB during the "Issue" cycle of `SUB.D F2,F8,F0` **(0.5 points)**. (6)

Entry number	Busy (Y/N)	Instruction	State	Destination	Value
0	Y	LD F8,0(R2)	Exec	F8	MEM R2->F8
1	Y	SUB.D F2,F8,F0	Issue	F2	F8-F10
2	-				
3	Y	DIV.D F0,F4,F6	Exec	F0	F4/F6

Cuanto más este, más bajo se va

UCLM

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

escuela superior ingeniería informática

Final written exam (January 2023)	**Exercises**	**Year 2022/23**

Name: _____

Important: Solve this exercise **on this sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 3 (2.5 points) (THREE PARTS). A compiler for a VLIW processor has generated the following output program:

Instruction 1 **loop**	L.D F1, 0(R1)	L.D F2, 0(R2)	DADDUI R1, R1, #8	DADDUI R2, R2, #8
Instruction 2	NOP	NOP	NOP	NOP
Instruction 3	DADDUI R4, R4, #-1	NOP	NOP	NOP
Instruction 4	NOP	MUL.D F3, F1, F2	NOP	NOP
Instruction 5	NOP	NOP	NOP	NOP
Instruction 6	ADD.D F4, F1, F3	BNEZ R4, loop	NOP	NOP
Instruction 7	S.D F4, -8(R1)	NOP	NOP	NOP

Assume that the code is as compact as possible, and the processor employs forwarding and delayed branch.

a. Complete the data in the following table:

FU= Functional unit	Number of cycles
Integer FU	2
Adder FP FU	0
Multiplier FP FU	1
MEM stage of the memory access instructions	2

2 DADDUI → Branch
0 ADD → SD
1 MULD → ADD
2 LD → MULD

Supongo fully Pipelined, si no lo fuese, en el caso de los primeros loads sería ciclo 1: ld F1, ciclo 3: ld F2

b. Unroll the program loop 4 times and then schedule it for the VLIW processor. Assume that the processor imposes no restrictions on the type of operations that can be issued, and the static scheduling is as aggressive and efficient as possible to optimize the resulting code as much as possible. Indicate the result in the following table (use as many rows as you need).

→ No restriction for type of instruc issued

	~~Integer FU~~	~~MEM stage~~	~~FP Mult~~	~~FP Adder~~
Instruction 1	LD F1, 0(R1)	LD F2, 0(R1)	LD F5, 8(R1)	LD F6, 8(R1)
Instruction 2	LD F9, 16(R1)	LD F10, 16(R1)	LD F13, 24(R1)	LD F14, 24(R1)
Instruction 3	DADDUI R1, R1, #32	DADDUI R2, R2, 32	DADDUI R4, R4, -4	
Instruction 4	MULD F3, F1, F2	MULD F7, F5, F6		
Instruction 5	MULD F11, F9, F10	MULD F15, F13, F14		
Instruction 6	ADD F4, F1, F3	ADD F8, F5, F7		BNEZ R4, loop
Instruction 7	ADD F12, F9, F11	ADD F16, F13, F15	SD F4, -32(R1)	SD F8, -24(R1)
Instruction 8	SD F12, -16(R1)	SD F16, -8(R1)		
Instruction 9				
Instruction 10				
Instruction11				
Instruction 12				
Instructino 13				
Instruction 14				
Instruction 15				
Instruction 16				

c. Complete the following table

Effectiveness of the VLIW program I gave you (in %)	
Effectiveness of the VLIW program obtained in b (in %)	

Code goes of

```
LD F1,0(R1)
LD F2,0(R2)
DADDUI R1,R1,#8
DADDUI R2,R2,#8
DADDUI R4,R4,#-1
MULD F3,F1,F2
ADD F4,F1,F3
BNEZ R4,loop
SD F4,-8(R1)
```

```
LD F1,0(R1)
LD F2,0(R2)
MULD F3,F1,F2
ADD F4,F1,F3
SD F4,0(R1)

LD F5,8(R1)
LD F6,8(R2)
MULD F7,F5,F6
ADD F8,F5,F7
SD F8,8(R1)

LD F9,16(R1)
LD F10,16(R2)
MULD F11,F9,F10
ADD F12,F9,F11
SD F12,16(R1)

LD F13,24(R1)
LD F14,24(R2)
MULD F15,F13,F14
ADD F16,F13,F15
SD F16,24(R1)

DADDUI R1,R1,32
DADDUI R2,R2,32
DADDUI R4,R4,-4
BNEZ R4,loop
```

Ejercicios Indio


```
loop: LD   F2, 0(R1)
      MULD F4, F2, F0
      MULD F10, F4, F8
      LD   F6, 0(R2)
      ADD.D F10, F10, F8
      SD   F10, 0(R2)
      DADDIU R1, R1, 8
      DADDIU R2, R2, 8
      CMPI R1, 800
      BEQ2 R1, loop
```

VLIW
- 2 MEM references
- 1 FP op
- 1 Int op / branch

FP → FP	Latency
FP → Store	3
LD → FP	2
LD → Store	1
	0

Fixed → F0, F8

a) Unroll 4 times & no stall

```
LD F2,0(R1)      LD F12,8(R1)     LD F22,16(R1)    LD F1,24(R1)     DADDIU R1,R1,32
MULD F4,F2,F0    MULD F14,F12,F0  MULD 24,F22,F0   MULD F3,F1,F0    DADDIU R2,R2,32
MULD F10,F4,F8   MULD F20,F14,F8  MULD F30,F24,F8  MULD F5,F3,F8    CMPI R1,800
LD F6,0(R2)      LD F16,0(R2)     LD F26,0(R2)     LD F7,0(R2)      BEQ2 R1,loop
ADD.D F10,F10,F8 ADD.D F20,F20,F8 ADDD F30,F30,F8  ADDD F9,F9,F8
SD F10,0(R2)     SD F20,8(R2)     SD F30,16(R2)    SD F9,24(R2)
```

Mem 1	Mem2	FP	Int
LD F2,0(R1)	LD F6,0(R2)		
LD F12,8(R1)	LD F16,8(R2)		
LD F22,16(R1)	LD F26,16(R2)	MULD F4,F2,F0	
LD F1,24(R1)	LD F7,24(R2)	MULD F14,F12,F0	
		MULD F24,F22,F0	
		MULD F3,F1,F0	
		MULD F10,F4,F8	
		MULD F20,F14,F8	
		MULD F30,F24,F8	
		MULD F5,F3,F8	
		ADDD F10,F10,F8	
		ADDD F20,F20,F8	
		ADDD F30,F30,F8	
SD F10,0(R2)		ADDD F9,F9,F8	DADDIU R1,R1,32
SD F20,8(R2)			DADDIU R2,R2,32
SD F30,16(R2)			CMPI R1,800
SD F9,24(R2)			BEQ2 R1,loop
-8			

↳ ¿p9? → sobrepasa el +32 de hace dos ciclos

→ Explicación. Hay dependencia. Cuando un LD crea un dato y un FP lo necesita, hay stall de 1. Por eso el hueco OBLIGATORIO entre ambas

```
loop: LD   F0,0(R1)
  ADDD   F4, F0, F2
    SD   F4,0(R1)
DADDIU   R1,R1,-8
   BNE   R1, R2, loop
```

	Latency cycles
FP → FP	3
FP → store	2
LD → FP	1
LD → store	0

```
loop: LD   F0,0(R1)
      ADD.D F4,F0,F2
      SD   F4,0(R1)
      DADDUI R1,R1,-8
      BNE  R1,R2,loop
```

Software Pipelining

Iteration 1	Iteration 2	Iteration 3
LD F0,0(R1)		
ADD.D F4,F0,F2	LD F0,-8(R1)	
SD F4,0(R1)	ADDD F4,F0,F2	LD F0,-16(R1)
	SD F4,-8(R1)	ADD.D F4,F0,F2
		SD F4,-16(R1)

→ Modified loop

```
loop: SD   F4,0(R1)
      ADD.D F4,F0,F2
      LD   F0,-16(R1)
      DADDUI R1,R1,-8
      BNE  R1,R2,loop
```

{ diff iterations

Start-up:

```
LD   F0,0(R1)
ADDD F4,F0,F2
LD   F0,-8(R1)
DADDUI R2,R2,16
```

loop:

finish up:

```
SD   F4,0(R1)
ADDD F4,F0,F2
SD   F4,-8(R1)
```

```
loop: LD   R4,0(R1)
      LD   R5,0(R2)
      SLT  R4,R4,R5
      SD   R4,0(R1)
      DADDUI R1,R1,8
      DADDUI R2,R2,8
      BNE  R1,R3,loop
```

{ independent

iteration 1			
LD R4,0(R1)			
LD R5,0(R2)			
SLT R4,R4,R5	LD R4,8(R1)		
	LD R5,8(R2)		
SD R4,0(R1)	SLT R4,R4,R5	LD R4,16(R1)	
	SD R4,8(R1)	LD R5,16(R2)	
		SLT R4,R4,R5	
		SD R4,16(R1)	

Start-up:

```
LD   R4,0(R1)
LD   R5,0(R2)
SLT  R4,R4,R5
LD   R4,8(R1)
LD   R5,8(R2)
DADDUI R3,R3,-16
```

↳ al contrario, y 0,-8,-16 por iteración

loop:

```
SD   R4,0(R1)
SLT  R4,R4,R5
LD   R4,16(R1)
LD   R5,16(R2)
DADDUI R1,R1,8
DADDUI R2,R2,8
BNE  R1,R3,loop
```

finish-up:

```
SD   R4,0(R1)
SLT  R4,R4,R5
SD   R4,8(R1)
```

```
read: LW R6,0(R1)
      DADD R3,R3,R6
       LW R7,4(R1)
      DADD R4,R4,R7
       SLT R8,R7,R6
       ADD R5,R5,R8
     DADDUI R1,R1,8
       BNE R1,R2,read
```

- Forwarding ✓
- Write and read at same cycle
- Dependences and branches solved at ID
- Delayed branch

a) Stalls (no depend.) between instrucs

```
LW R6,0(R1)
  STALL
DADD R3,R3,R6
LW R7,4(R1)
  STALL
DADD R4,R4,R7
SLT R8,R7,R6
DADD R5,R5,R8
DADDUI R1,R1,8
  STALL
BNE R1,R2,read
  STALL
```

IF ID EX ME WB
IF ID EX ME WB

IF ID EX ME WB
IF ID EX ME WB

IF ID EX ME WB
IF ID EX ME WB
→ solved

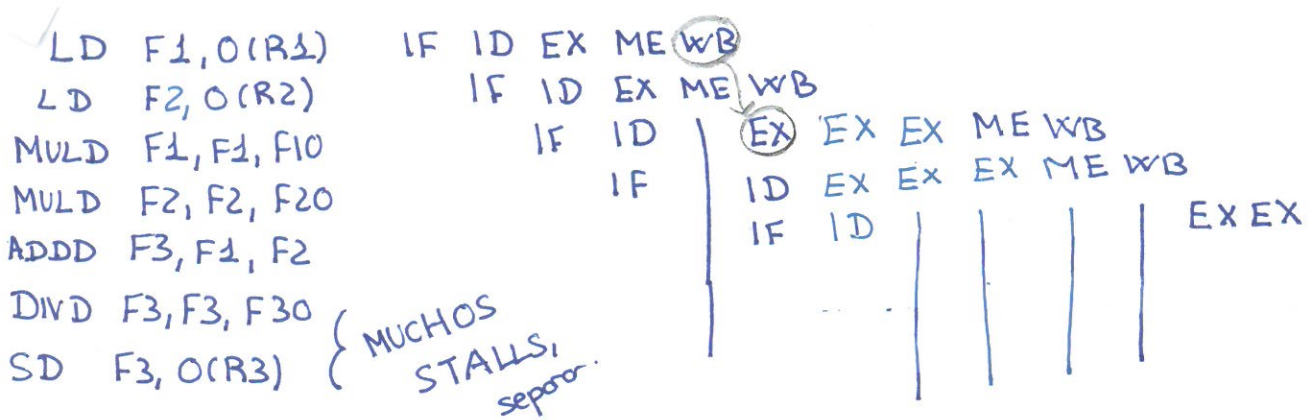
→ Como las 2 primeras

Loopunrolling

```
read: LW R6,0(R1)
      DADD R3,R3,R6
       LW R7,4(R1)
      DADD R4,R4,R7
       SLT R8,R7,R6
      DADD R5,R5,R8
       LW R9,8(R1)
      DADD R3,R3,R9
       LW R10,12(R1)
      DADD R4,R4,R10
       SLT R11,R10,R9
     DADDUI R1,R1,16
       BNE R1,R2,read
```

Renaming and haya stalls

```
LW R6,0(R1)
LW R7,4(R1)
LW R9,8(R1)
LW R10,12(R1)
DADD R3,R3,R6
DADD R4,R4,R7
DADD R3,R3,R9
DADD R4,R4,R10
SLT R8,R7,R6
DADD R5,R5,R8

SLT R11,R10,R9
BNE R1,R2,read
DADD R5,R5,R11
```


LD F1, 0(R1)
LD F2, 0(R2)
MULD F1, F1, F10
MULD F2, F2, F20
ADDD F3, F1, F2
DIVD F3, F3, F30
SD F3, 0(R3)
MUCHOS STALLS, sepoo.
IF ID EX ME WB
IF ID EX ME WB
IF ID EX EX EX ME WB
IF ID EX Ex EX ME WB
IF ID EX EX

1/ Superescalar VS VLIW

Superescalars pueden ir en desorden. VLIW se llevan a cabo en orden, salvo end port, ellas llevan su orden

```
LD   F2,0(R1)
MULD F4,F2,F0
LD   F6,0(R2)
ADDD F6,F4,F6
SD   F6,0(R2)
DADDUI R1,R1,-8
DADDUI R2,R2,-8
SLT  R3,R1,end
BEQZ R3,loop
```

- Fully pipelined
- Branch delay ignored

ALU	ALU	3
ALU	SD	2
LD	ALU	1
SD	LD	

VLIW (order)

Fixed → F0

Unroll 4 times

```
LD   F2,0(R1)      LD F8,-8(R1)      LD F14,-16(R1)    LD F20,-24(R1)
MULD F4,F2,F0      MULD F10,F8,F0    MULD F16,F14,F0   MULD F22,F20,F0
LD   F6,0(R2)      LD F12,-8(R2)     LD F18,-16(R2)    LD F24,-24(R2)
ADDD F6,F4,F6      ADD F12,F10,F12   ADD F18,F16,F18   ADD F24,F22,F24
SD   F6,0(R2)      SD F12,-8(R2      SD F18,-16(R2)    SD F24,-24(R4)

DADDUI R1,R1,-32
DADDUI R2,R2,-32
SLT  R3,R1,end
BEQZ R3,loop
```

cycle	MEM 1	MEM 2	FP 1	FP 2	Int/branch
1	LD F2,0(R1)	LD F6,0(R2)			
2	LD F8,-8(R1)	LD F12,-8(R2)	NOP LD→ALU		
3	LD F14,-16(R1)	LD F18,-16(R2)	MULD F4,F2,F0	MULD F10,F8,F0	
4	LD F20,-24(R1)	LD F24,-24(R2)	MULD F16,F14,F0	MULD F22,F20,F0	DADDUI R1,R1,-32 (*)
5			NOP ALU→ALU		
6			" "		DADDUI R2,R2,-32
7					
8			ADD F6,F4,F6	ADD F12,F10,F12	
9		NOP ALU→SD	ADD F18,F16,F18	ADD F24,F22,F24	
10	SD F6,32(R2)	NOP ALU→SD SD F12,-8(R2) 24			SLT R3,R1,end
11	SD F18,-16(R2) 16	SD F24,-24(R2) 8			BNE R3,loop

(*) Al haber restado 32, se debe compensar sumando 32 a los SD

2) Poison bits

```
        LW* R1,0(R2)
if_1:   BEQ2 R3, endif1
        DADD D1 R1, R4, 8
ndif_1: LW* R5,0(R2)
 if_2: BEQ2 R6,endif_2
     DADDI R5, R7, -1
ndif_2:DADD R8, R1,R5
  SW  R8,0(R2)
)ADD1 R9, R0, 1
```

R1	R2	R3	R4	R5	R6	R7	R8	R9
10	0	0	4	5	6	7	8	9

Access to MEM position 0 is forbidden, no LD or SD

a) Processor with no speculation help

Instruc	Produces Exception? Y/N	Terminates?	Performs
LW* R1,0(R2)	Y	Y	R1 ← [illegible]

b) HW/SW cooperation. Not distinguish between norm and spec instructs. Always returns undef.

Instruc	Produces Exception? Y/N	Terminates?	Performs
LW* R1,0(R2)	Y	N	R1 ← Undef
BEQ2 R3, end-if1	N	N	Jumps
LW* R5,0(R2)	Y	N	R5 ← Undef
BEQ2 R6,endif-2	N	N	Doesn't jumps
)ADD1 R5,R7,-1	N	N	R5 ← 7-1 = 6
)ADD R8,R1,R5	N	N	R8 ← Undef
SW R8, 0(R2)	Y	N	No storage
DADD1 R9,R0, 1	N	Y	R9 ← 10+1

c) HW/SW cooperation. Distinguish between norm and spec. Normal terminates

Instruc	Produces Exception? Y/N	Terminates?	Performs
LW*R1,0(R2)	Y	N	R1 ← Undef
BEQ2 R3,end-if1	N	N	Jumps
.W* R5,0(R2)	Y	N	R5 ← Undef
BEQ2 R6,end-if2	N	N	Doesn't jumps
)ADD1 R5,R7,-1	N	N	R5 ← 7-1
ADD R8,R1,R5	N	N	R8 ← Undef
W R8,0(R2)	Y	Y	No store

d) Speculation with poison bits

Instruc	Produces Exception? Y/N	Terminates?	Performs
W* R1,0(R2)	Y	N	R1 ← dirty
BEQ2 R3,end-if	N	N	Jumps
.W* R5,0(R2)	Y	N	R5 ← dirty
BEQ2 R6,end-if2	N	N	Doesn't jumps
)ADD1 R5,R7,-1	N	N	R5 ← 7-1 = 6
)ADD R8,R1,R5	Y	Y	Terminate

R1	R2	R3	R4	R5	R6	R7	R8	R9
Dirty	0	0	4	6	6	7	8	9

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2024)	Exercises	Year 2023/24

Name: _____

Important: Solve each exercise on **a different sheet**. Do not forget to write down your name on all the sheets you deliver.

Exercise 1 (2,5 points). The following MIPS64 code calculates the dot product of two vectors (vector1 and vector2). The dot product result is stored in $f0 at the end of the loop.

```
;Example vectors
vector1:  .float 1.0, 2.0, 3.0, 4.0, 5.0, 6.0
vector2:  .float 6.0, 5.0, 4.0, 3.0, 2.0, 1.0

;Variable initialization
          addi   $t0, $zero, 0     ; I1 (i = 0)
          addi   $t1, $zero, 6     ; I2 (n = 6)
          add.s  $f0, $zero, $f31  ; I3 ($f0=0)
          la     $t2, vector1      ; I4 (get address of vector1)
          la     $t3, vector2      ; I5 (get address of vector2)

Loop:
          l.s    $f4, ($t2)        ; I6 (read element from vector1)
          l.s    $f6, ($t3)        ; I7 (read element from vector2)
          mul.s  $f8, $f4, $f6     ; I8 (multiply the values)
          add.s  $f0, $f0, $f8     ; I9 (add the result to the dot product)
          addi   $t0, $t0, 1       ; I10 (i = i + 1)
          addi   $t2, $t2, 4       ; I11 (get next position of vector1)
          addi   $t3, $t3, 4       ; I12 (get next position of vector2)
          blt    $t0, $t1, Loop    ; I13 (if i < n, go back to the beginning)
```

The computer used has the following characteristics:

- A data value can be written to and read from a register in the same clock cycle.
- Operand forwarding is used. (*)
- All dependencies are detected and resolved in the ID stage.
- Branches are entirely resolved in the ID stage. Delayed branching is applied.
- Two WB operations in floating point are not allowed in the same cycle.
- In the same cycle, one floating point WB operation and another integer operation can be performed.
- All functional units are segmented.

Here is the table that shows the number of cycles used by each functional unit:

Functional Unit	Number of cycles used
FP adder/subtractor	2
FP multiplier	4
FP Divider	12
Other units	1

a) Mark the stalls that exist in the code from I6 to I13 (in a single iteration). Indicate between which instructions they occur and the reason why they occur.

Between which instructions	Reason
I6 → I8	Data Dependency through f4
I7 → I8	Data Dependency through f6
I8 → I9	Data Dependency through f8

b) Calculate the CPI of the program (consider all the instructions executed from I1; show the operations performed).

c) Rename and unroll the loop the minimum number of times to schedule it without stalls. The resulting code should perform the same operation as the original code. Then, answer the following questions:

a. What is the CPI of the resulting code? (show the operations performed)

b. Complete the following table with the resulting code.

```
;Example vectors
vector1:  .float 1.0, 2.0, 3.0, 4.0, 5.0, 6.0
vector2:  .float 6.0, 5.0, 4.0, 3.0, 2.0, 1.0

;Variable initialization
          addi   $t0, $zero, 0     ; I1 (i = 0)
          addi   $t1, $zero, 6     ; I2 (n = 6)
          add.s  $f0, $zero, $f31  ; I3 ($f0=0)
          la     $t2, vector1      ; I4 (get address of vector1)
          la     $t3, vector2      ; I5 (get address of vector2)
```

Loop:		

ls f4, (t2)
ls f6, (t3)
muls f8, f4, f6
adds f0, f0, f8
ls f4, 4(t2)
ls f6, 4(t3)
muls f8, f4, f6
adds f0, f0, f8
addi t0, t0, 2
addi t2, t2, 8
addi t3, t3, 8
blt t0, t1, loop
	ls f4, (t2)
	ls f6, (t3)
	ls f10, 4(t2)
	ls f12, 4(t3)
	muls f8, f4, f6
	muls f14, f10, f12
	adds f0, f0, f8
	adds f16, f16, f8
	addi ...
	addi ...
	addi ...
	blt t0, t1, loop
		ls f4, (t2)
		ls f6, (t3)
		ls f10, 4(t2)
		ls f12, 4(t3)
		muls f8, f4, f6
		muls f14, f10, f12
		addi ...
		addi ...
		addi ...
		adds f0, f0, f8
		adds f16, f16, f8
		blt t0, t1, loop

problema de ls con muls

(*) Con Forwarding, al acabar MEM, sin forwarding en WB

Si carga de Memoria, sino EX

IF ID EX ME WB
IF ID EX ME WB
IF ID EX EX EX ME WB

I6→I8 = 1 stall
I7→I8 = 1 stall
I8→I9 = 2 stalls
Stalls: (espera 2 ciclos extra por datapath)

b) Instructions = From I1 to I13 = 13; Iteration = From 0 to n = 6 = 6;
Cycles per ite: 13 + 4 = 17; Total cycles: 12 · 6 = 72; Total instruc: 8 · 6 = 48 + 5 = 53
+5 = 77
CPI = total cycles / total instruc = 77/53 = 1'4528...

CPI = Instruct = 17 Iterat = 3; Total cyc = 5 + 12 · 3
Total instruc = 5 + 12 · 3
CPI = 1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2024)	Exercises	Year 2023/24

Name: _____

Important: Solve each exercise on **a different sheet**. Do not forget to write down your name on all the sheets you deliver.

Exercise 2 (2,5 points). Let be a MIPS64 processor using hardware speculation and dynamic scheduling using the Tomasulo technique with speculation, with the following characteristics:

- The load and store execution stage takes 2 cycles.
- The floating point (FP) multiplication execution step takes 7 cycles.
- The floating-point addition (FP) execution step takes 3 cycles.
- A 4-entry reorder buffer (ROB) is provided. → Cuántas instruc. sin commitear a la vez
- 2-entry reservation stations are available for each unit in FP.
- It has different load and store buffers of 2 entries each. → Puede haber dos loads a la vez
- Any stage of an instruction that wants to write to the CDB (Common Data Bus) cannot do so if another instruction is already using it. → Solo uso del WB a la vez
- In general, any resource released at a given clock cycle cannot be used until the next one.

a) Detail the execution of the sequence of instructions shown below when using the Tomasulo algorithm with speculation. To do so, indicate in the table below the cycle number (start and end if applicable) for each stage of each instruction.

Instruction	Issue	Execution	Results Writing	Commit
L.D F0, 0(R1)	1	2-3	4	5
L.D F2, 0(R2)	2	3-4	5	6
ADD.D F4, F0, F2	3	6-8	9	10
L.D F6, 0(R3)	5	6-7	8	11
MUL.D F8, F4, F6	6	10-18	17	18
ADD.D F10, F6, F12	7	9-11	12	19
S.D F8, 8(R1)	11	18-19	20	21
S.D F10, 8(R2)	12	13-14	15	22

b) Fill the table below to show the contents of the reorder buffer (ROB) at cycle 10. Use only the necessary table entries. → "Aquellas en ejecución o WB en el ciclo 10"

Entry #	Instruction	Stage	Destination Register	Value
1	ADDF4, F0, F2	COMMIT	F4	F0 + F2
2	LD F6, 0(R3)	Wait commit	F6	MEM 0(R3)
3	MUL.D F8, F4, F6	Exec	F8	F4 * F6
4	ADD.D F10, F6, F12	Exec	F10	F6 + F12

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2024)	Exercises	Year 2023/24

Name: _____

Important: Solve each exercise on **a different sheet**. Do not forget to write down your name on all the sheets you deliver.

Exercise 3 (2 points) Consider a MIPS64-based processor, and various mechanisms for handling exceptions produced by speculative instructions. From the following code:

```
        DADDI R1, R1, #1

        BEQ   R1, R2, then

else:   DDIV  R5, R1, R5
        J     end

then:   DDIV  R5, R2, R1
        DADDI R3, R3, #1

end:    DADD  R4, R4, R3
```

Considering that there is a high probability that the branch condition will not be met, the compiler has generated the next version of the same code, with one speculative instruction (highlighted):

```
        DADDI R1, R1, #1

        DDIV* R5, R1, R5     ; speculative division

        BNE   R1, R2, end

        DDIV  R5, R2, R1
        DADDI R3, R3, #1

end:    DADD  R4, R4, R3
```

First Part:

Indicate, depending on the exception handling mechanism available in the processor, and **for the speculative version of the code**, the instructions that cause any exception, and whether or not they terminate the program, as well as the value of the R1-R5 registers at the end of the program, knowing that their initial state is:

R1	R2	R3	R4	R5
0	1	2	3	0

a) The processor has no speculation support, i.e., it does not distinguish between normal and speculative instructions. Therefore, any instruction that throws an exception will cause the program to be terminated immediately.
b) The processor has a hardware/software cooperation mechanism that distinguishes between normal and speculative instructions. If the instruction that throws the exception is normal, the exception's service routine will terminate the program, but if it is speculative, the routine will return an undefined value (without aborting the execution).
c) Speculation with *poison bits*.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (January 2024)	Test	Year 2023/24

Name: Javier García Terceño

Indicate **IN THE NEXT TABLE** the correct answer to each test question below:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	A	B	C	D	A	B	B	C	C	A	A	C	A

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
C	C	D	A	C	C	A	C	B	D	A	A	A	D	B

Important: Test counts **3 points**. Each correct question counts 0.1 points. Wrong answers subtract 0.1/3 points. Not answered questions are not considered for the mark.

1. Which of the following statements is correct?
   a) An organization can survive for many years, but its architecture changes as technology evolves
   b) Many computer manufacturers offer a family of models, all with the same organization, but with a difference in architecture
   c) The architecture of a computer must be designed to implement a specification of a particular organization
   d) All statements are false

2. The Linpack benchmark used in the TOP500 is of the type:
   a) Synthetic
   b) Real program
   c) Kernels
   d) Suites

3. From compiler X we have obtained an improved compiler M, which provides a code for floating-point instructions that is k times faster than the original. If 40% of the total time is spent on floating-point instructions when using the C compiler, what would be the maximum speed increase we could obtain with compiler M?
   a) 1.67
   b) 167%
   c) 2
   d) All are false

   $Speedup = \frac{1}{(1-f)+f/s} = \frac{1}{1-0'4+0} = \frac{1}{0'6} = 1'67$ ↳ 0'4/∞

4. We want to analyse the performance of two computers with the same instruction set. In that set, there are three types of instructions: A, B and C. Computer 1 needs 2, 4 and 6 cycles to execute instructions of type A, B and C, respectively. Computer 2 needs 2, 2 and 5 cycles. Computer 1 operates at 1.2 GHz and computer 2 at 1.8 GHz. The benchmark used is composed of 20% of type A instructions, 50% of type B and 30% of type C. What is the MIPS speed of both computers?
   a) Computer 1: 285.7 and Computer 2: 512.2
   b) Computer 1: 285.7 and Computer 2: 620.7
   c) Computer 1: 222.2 and Computer 2: 610.2
   d) All are false

   $MIPS = \frac{Clock\ Freq}{CPI \cdot 10^6}$; $CPI_1 = (0'2 \cdot 2) + (0'5 \cdot 4) + (0'3 \cdot 6) = 4'2$; $CPI_2 = (0'2 \cdot 2) + (0'5 \cdot 2) + (0'3 \cdot 5) = 2'9$

   $MIPS_1 = \frac{1'2 \cdot 10^9}{4'2 \cdot 10^6} = 285'7$; $MIPS_2 = \frac{1'8 \cdot 10^9}{2'9 \cdot 10^6} = 620'7$

5. What problem does "register renaming" solve?
   a) Control dependencies
   b) Cache misses
   c) Name dependencies
   d) All are correct

6. In a basic segmented processor with forwarding, we execute this code:
```
ADD.D F4, F4, F0 ;I1
SUB.D F4, F2, F0 ;I2
ADD.D F4, F4, F0 ;I3
```
   Knowing that the addition and subtraction functional units take 2 cycles. Indicate the correct option:
   a) There are two output dependencies
   b) There are two anti-dependencies
   c) There are two data dependencies
   d) All are correct

7. Consider the following loop:
```
Loop:   L.D F8, 0(R1)        ;I1
        ADD.D F6, F2, F8     ;I2
        S.D F6, 0(R1)        ;I3
        DADDUI R1, R1, #-8   ;I4
        BNEZ R1, Loop        ;I5
```
   Between the first and second instruction (L.D and ADD.D):
   a) There is a data dependency in the same iteration
   b) There is an output dependency in the same iteration
   c) There is a data dependency between iterations
   d) None are correct

8. In a new segmented processor of 10 stages with forwarding, all possible hardware improvements have been applied to avoid stalls. The CPI (Cycles Per Instruction) ideally achievable for the system is:
   a) 0.1
   b) 1
   c) 10
   d) All are false

9. The Scoreboard technique is used to…
   a) Dynamically rename registers
   b) Dynamically resolve data dependencies
   c) Reduce the number of instructions to execute
   d) All are false

10. What is the technique called that allows instructions to execute out of order, but ensures that the results are generated according to the program order?
    a) Speculative programming
    b) Compiler programming
    c) Dynamic programming
    d) Static programming

11. What is a reservation station in the context of the Tomasulo algorithm?
    a) A physical storage location for data that allows their speculation
    b) A physical storage location for registers that allows their renaming
    c) A physical storage location for data dependencies that allows their resolution
    d) All are correct

12. In the trace scheduling technique, what does a trace represent?
    a) Most likely sequence of instructions to execute
    b) Sequence of instructions to execute in a loop iteration
    c) Basic block within a loop
    d) All are false

13. Given a 5-stage pipeline in which 2 out of every 9 instructions is a taken branch, and 2 delay slots happen for each taken branch, indicate which of the following is correct:

a)	CPI = 1.44	IPC = 0.69
b)	CPI = 1.22	IPC = 0.81
c)	CPI = 1.66	IPC = 0.6
d)	CPI = 1	IPC = 1

   Additional cycles: 2 x 2 = 4 (2 instr ↳ delay per instr)

   $CPI = \frac{Total\ cycle}{Instruc} = \frac{9+4}{9} = 1'44$

   $IPC = \frac{1}{CPI} = \frac{1}{1'44} = 0'69$

IF | ID | EX | MEM | WB

14. Indicate which of the following statements is correct:
    a) If the branch condition is evaluated in the EX stage and the branch target address is computed in the MEM stage, we will have a scheme with 1 delay slot
    b) If the branch condition and address jump are both computed in the ID stage, we will have a scheme with 3 delay slots
    c) If the branch condition is evaluated in the EX stage and the branch target address is computed in the ID stage, we will have a scheme with 2 delay slots
    d) All of the above are correct

↳ Depende del pipeline

15. What is the latency of a branch instruction?
    a) The number of additional cycles between the IF stage and the cycle of its complete resolution
    b) The number of cycles between the IF stage and the ID stage
    c) The number of additional cycles between the IF stage and the cycle where its condition is computed
    d) The number of cycles between the IF stage and the MEM stage

↳ Falta + compute log address

16. Dynamic branch prediction techniques focus on
    a) the behavior through the statistics of all the branch instructions
    b) the use of software compilers that predict the behavior of a program
    c) speculating through poison bits which branch instructions are to be taken
    d) the behavior of the last executions of branch instructions

17. A "speculative" instruction:
    a) Is a conditional branch whose condition must be predicted
    b) Is an instruction that causes a non-recoverable exception
    c) It is an instruction executed before knowing whether it should be executed → To exploit ILP
    d) An instruction that calculates a probable result of an arithmetic-logical operation

18. When a branch instruction that has been mispredicted reaches the top of the ROB:
    a) The data structures used for the prediction are updated
    b) The fetch of the correct instruction is initiated
    c) The ROB is flushed
    d) All are correct

19. The problem of memory "disambiguation":
    a) Refers to the occurrence of potential RAW hazards in memory
    b) Refers to the occurrence of potential WAR hazards in memory
    c) Refers to the occurrence of potential in-memory WAW hazards
    d) Refers to the occurrence of potential structural hazards in memory

20. We need to schedule a sequential code consisting of 40 instructions for a VLIW processor with 4 operations per instruction. In the best case scenario (assuming an "ideal" code and processor), we would have:
    a) 40 long instructions
    b) 20 long instructions
    c) 10 long instructions
    d) 4 long instructions

21. A characteristic of superscalar processors is...
    a) They use long instructions and dynamic scheduling
    b) They always employ static prediction
    c) They execute multiple instructions per cycle
    d) Instructions are always retired out-of-order

22. In a superscalar processor, in the event of an exception in the *Execution* stage...
    a) The Reorder Buffer (ROB) is immediately emptied, preventing the execution of more program instructions
    b) The resources (ROB and others) used by the instruction that caused the exception are released, but the program execution continues normally
    c) It immediately jumps to the exception handler
    d) None of the above

23. Each iteration of a given loop in a scalar processor requires 15 clock cycles to be issued. We unrolled such a loop 5 times and scheduled it for a 4-way superscalar obtaining that we now perform 5 iterations of the original loop in 15 cycles. What is the speedup obtained?
    a) 3
    b) 4
    c) 5
    d) None of the above.

$$\text{Speedup} = \frac{\text{Time before}}{\text{Time after}} = \frac{15 \times 5}{15} = 5$$

24. Intel's first processor not compatible with the x86 family was:
    a. XEON
    b. Itanium
    c. i7
    d. All are false

25. Which of the following is true?
    a. ARM processors use a load/store architecture
    b. RISC-V processors use a load/store architecture
    c. Alpha processors use a load/store architecture
    d. All statements are true

26. The slowdown or loss of performance in an SMT multicore generates:
    a. Unpredictable execution time
    b. Possibility of WCET calculation (Worst Case Execution Time)
    c. Decrease in the execution time of a thread
    d. All statements are true

27. A difference between the P6 and P7 architecture is:
    a. P7 has more decoders
    b. That there is no barrel shifter in P6
    c. That the L1 of the P6 data cache is larger
    d. All statements are true

28. Thread Level Parallelism is used to:
    a. Improve the utilization of a single processor
    b. Improve the execution time of a program
    c. Allow a processor's segmentation to be deeper
    d. None of the statements are true

29. CUDA defines:
    a. An architecture model
    b. A programming model
    c. A memory management model
    d. All statements are true

30. TPUs:
    a. These are FPGAs specifically for neural computing
    b. They achieve much higher computing power by using tens of thousands of low-precision arithmetic units on each chip
    c. Their main disadvantage is their low performance in matrix multiplication
    d. All statements are false

Use the tables below to present the results of this part. Use only the rows you need in each case. In a) and b) use the label "undefined" to indicate that the value of a register is undefined, and in c) use the label "dirty" to indicate that the *poison bit* associated to a register is set (equal to 1).

a) Processor without speculation support.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	Yes	Program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	1	2	3	0

b) Hardware/software cooperation mechanism.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	No	R5←undefined
BNE R1, R2, end	No	No	Doesn't jump
DDIV R5, R2, R1	No	No	R5←1
DADDI R3, R3, #1	No	No	R3←3
DADD R4, R4, R3	No	Yes	R4←6; program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	1	3	6	1

c) Speculation with *poison bits*.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	No	R5←dirty (poison bit=1)
BNE R1, R2, end	No	No	Doesn't jump
DDIV R5, R2, R1	No	No	R5←1 (poison bit=0)
DADDI R3, R3, #1	No	No	R3←3
DADD R4, R4, R3	No	Yes	R4←6; program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	1	3	6	1

Second Part:

Repeat the previous part (use the tables below), but now starting from these initial values for registers R1-R5:

R1	**R2**	R3	R4	R5
0	**0**	2	3	0

a) Processor without speculation support.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	Yes	Program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	0	2	3	0

b) Hardware/software cooperation mechanism.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	No	R5←undefined
BNE R1, R2, end	No	No	Jumps
DADD R4, R4, R3	No	Yes	R4←5; program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	0	2	5	undef.

c) Speculation with *poison bits*.

Instruction executed	¿Exception raised? (Yes/No)	¿Program termination? (Yes/No)	Result (jump/don't jump, dest. reg. value...)
DADDI R1, R1, #1	No	No	R1←1
DDIV* R5, R1, R5	Yes	No	R5←dirty (poison bit=1)
BNE R1, R2, end	No	No	Jumps
DADD R4, R4, R3	No	Yes	R4←5; program terminates

Register file content after program termination:

R1	R2	R3	R4	R5
1	0	2	5	dirty

Jan 24

Exer 3.

```
        DADDI  R1,R1,#1
        DDIV*  R5,R1,R5     ;especulative division
        BNE    R1,R2, end
        DDIV   R5,R2,R1
        DADDI  R3,R3,#1
end:    DADD   R4,R4,R3
```

1st Part

R1	R2	R3	R4	R5
0	1	2	3	0

a) No spec. support. If except = terminate

Instruc exec.	Excep Raised? Y/N	Program Term? Y/N	Result
DADDI R1,R1,#1	N	N	R1 ← 1
DDIV* R5,R1,R5 (R1/R5)	Y	Y	Program terminates due 0/0

b) HW + SW cooperation, distinghises between normal and speculative instruc, if except normal → terminates if specul → undef value

Instruc exec	Excep Raised? Y/N	Program Term? Y/N	Result
DADDI R1,R1,#1	N	N	R1 ← 1
DDIV* R5,R1,R5	Y	N	R5 ← undef
BNE R1,R2,end	N	N	No jump; R1 = R2 (1 : 1)
DDIV R5,R2,R1	N	N	R5 ← 1; R2/R1 (1/1)
DADDI R3,R3,#1	N	N	R3 ← 3; 2+1
DADD R4,R4,R3	N	Y	R4 ← 6; 3+3; end program

c) Speculative with poison bits

Instruc	Excep Raised? Y/N	Program Term? Y/N	Result
Daddi R1,R1,#1	N	N	R1 ← 1
DDIV* R5,R1,R5	Y	N	R5 ← dirty (poison bit = 1)
BNE R1,R2,end	N	N	R1 = R2; doesn't jump
DDIV R5,R2,R1	N	N	R5 ← 1 (poison bit = 0)
DADDI R3,R3,#1	N	N	R3 ← 3
DADD R4,R4,R3	N	Y	R4 ← 6

NEW	R1	R2	R3	R4	R5
	0	0	2	3	0

b) HW + SW cooperation

Instruc Exec	Excep Raised? Y/N	Program Term? Y/N	Result
DADDI R1,R1,#1	N	N	R1 ← 1
DDIV* R5,R1,R5	Y	N	R5 ← Undef
BNE R1,R2,end	N	N	Jumps
DADD R4,R4,R3	N	Y	R4 ← 5

R1	1
R2	0
R3	2
R4	5
R5	undef

c) speculative with poison bits

DADDI R1,R1,#1	N	N	R1 ← 1
DDIV R5,R1,R5	Y	N	R5 ← dirty (poison bit: 1)
BNE R1,R2,end	N	N	Jumps
DADD R4,R4,R3	N	Y	R5 ← 5

Branch Based on the exec history

1-Bit predict

Bit stores behaviar of last exec.

NT (0)
T (1)

2-Bit predict

If predict 2 times wrong, changes

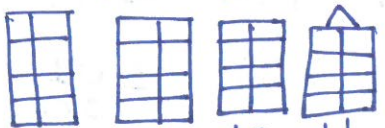
SNT (00)
WNT (01)
WT (10)
ST (11)

normally, predictor n=2 is equal or better than n>2

(m,n) predictor
Uses info from m branches
Used among 2^m branch predictions, being each a n-bit predictor for a single branch

BHR: Branch History Register → 2^m BHT tables
BHT: Branch History Table → n bits per entry

Example: (2,2) predictor



→ 2^m bits BHT

Exer 8

Hybrid. 2 bit local predictor. Entry per branch. Global (1,2), 2 entry BHT entries indexed by the least significant bit of branch address. Selector 2-bit saturating (↑ Taken, ↓ NT). Local when selector = 0,1, global = 2,3

```
loop: LW   R2,0(R1)   6
      ANDI R4,R2,#1   0
      BEQZ R4,next    T   (B1, branch B1 = 0x654)
      DADDUI R1,R1,#4 9
      DADDUI R2,R2,#1
      ...
next: BNE  R1,R3,loop (B2, branch B2 = 0x623)
      SW   R2,-4(R1)  5
```

R1 and R3 initialized with: 3, 2, 5, 4, 6

11	10	0101	0100	0110
01	01	0001	0001	0001
01	00	0001	0000	0000

WNT

Branch	ACTUAL	Predictor	Selector	Local Predictor	Global Predictor	
Initial state	–	–	00	01 / 01	BHR=0	01 / 01
B1 (v=3)	NT	NT(h)	00	00 / 01	BHR=0	01 / 01
B2 (v=3)	T	NT(m)	01	00 / 10	BHR=0	01 / 01
B1 (v=2)	T	NT(m)	10	01 / 10	BHR=0	01 / 01
B2 (v=2)	T	NT(m)	11	01 / 10	BHR=1	01 / 01
						10 / 01
B1 (v=5)	NT	NT(h)	10	01 / 10	BHR=0	01 / 00
						10 / 01
B2 (v=5)	T	T(h)	11	01 / 10	BHR=1	01 / 00
						11 / 01
B1 (v=4)	T	NT(m)	11	01 / 10	BHR=1	01 / 01
						11 / 01
B2 (v=4)	T	NT(m)	11	01 / 10	BHR=1	01 / 01
						11 / 10
B1 (v=6)	T	NT(m)	11 (Miro aquí GP)	01 / 10	BHR=1 (*)	01 / 10
						11 / 10
B2 (v=6)	NT	T(m)	10	01 / 10	BHR=0	01 / 10
						11 / 01

(*) Miro anterior tabla 1, B1 → Esto tmb depende anterior tabla

6 First iterations

```
    DADD R1,R0,R0
L1: BEQZ R1,L2      #B1
    ...
L2: BEQZ R1,L3      #B2
    ...
L3: XORI R1,R1,#1   #(<=> NOT R1)
    J L1
```

0101 + 0001 = 0100

0000 + 0001 = 0001

"Estoy 1 en pasado"

T↑ NT↓ — 2bit per branch

2^m tables = 4 — (2,2) → n bits

a) Actual behavior

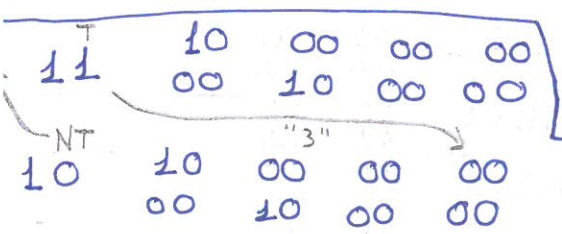
B1/B2	Actual T/NT
initial state	-
B1 (R1=0)	T
B2 (R1=0)	T
B1 (R1=1)	NT
B2 (R1=1)	NT
B1 (R1=0)	T
B2 (R1=0)	T
B1 (R1=1)	NT
B2 (R1=1)	NT
B1 (R1=0)	T
B2 (R1=0)	T
B1 (R1=1)	NT
B2 (R1=1)	NT
B1 (R1=0)	T
B2 (R1=0)	T
B1 (R1=1)	NT

b) One-level predict. behavior

Predict bits for B1 y B2	Predicted T/NT	Hit Y/N
00 / 00	-	-
01 / 00	NT	N(m)
01 / 01	NT	N(m)
00 / 01	NT	Y(h)
00 / 00	NT	Y(h)
01 / 00	NT	N(m)
01 / 01	NT	N(m)
00 / 01	NT	Y(h)
00 / 00	NT	Y(h)
01 / 00	NT	N(m)
01 / 01	NT	N(m)
00 / 01	NT	Y(h)
00 / 00	NT	Y(h)
01 / 00	NT	N(m)
01 / 01	NT	N(m)
00 / 01	NT	Y(h)

c) Two-level predictor behavior

Prediction bits (BHR and BTHs)	Predicted outcome (T/NT)	Hit (Y/N)
00 00 / 00 00 / 00 00 / 00 00	-	-
00 00 / 00 00 / 00 00 / 00 00	NT	N(m)
01 00 / 00 00 / 00 00 / 00 00	NT	N(m)
11 01 / 00 01 / 00 00 / 00 00	NT	Y(h)
10 01 / 00 01 / 00 00 / 00 00	NT	Y(h)
01 00 / 00 01 / 00 00 / 00 00	NT	N(m)
10 00 / 00 01 / 00 00 / 00 00	NT	N(m)
11 10 / 00 10 / 00 00 / 00 00	NT	Y(h)
10 10 / 00 10 / 00 00 / 00 00	NT	Y(h)
10 00 / 00 10 / 00 00 / 00 00	NT	N(m)
11 00 / 00 10 / 00 00 / 00 00	NT	N(m)
11 11 / 00 11 / 00 00 / 00 00	NT	N(m)



Por el BHR entero apuntamos a la tabla 3, Restoriamos 1 por ser NT el actual pero como es ya, ahi se queda

El BHR sirve para saber qué tabla actualizaríamos en el momento. Se suma 1 si el actual fue taken. Se resta 1 si fue not taken

Nov 2015

- 2Bit local predictor, entry por branch instruc → 2, B1 y B2
- Global (2,1) predictor, each BHT table contains on entry por branch instruc
- Selector of 2-bit saturating counter. Taken ↑ ; Not Taken ↓. Local when selector = 0 or 1, global = 2,
- Selector initialized to 11, rest = 0

R1 and R3 initialized to the sequence: 0, 0, 1, 0, 0, 2

```
loop: LW   R2, 0(R1)
      BEQZ R2, skip     #B1
      SW   R0, 0(R1)
      DADDUI R4, R4, #1
skip: BEQ  R1, R3, end  #B2
      DADDUI R1, R1, #4
      J loop
end: ...
```

Branch exec. (B1 or B2)	Actual T/NT	Predictor T/NT (h/m)	Selector	Local Predictor	Global Predictor
—	—	—	11	0000	BHR = 00 BHTs: 00 00 00 00
B1 (R2 = 0)	T	NT (m)	11	0000	BHR: 01 por miss ↙

Jan 23

initial State → 01 / 01 — b) 2-bit pred — c) (2,2) — 00 / 00 — — —

step	B1 or B2	R1	R4	T/NT	Predictor	HIT	BHR	Table 1	Table 2	Table 3	Table 4	Predict	HIT
1	B1	0	?	NT	01	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
2	B2	0	0	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
3	B2	0	1	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
4	B2	0	2	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
5	B2	0	3	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
6	B2	0	4	T	00 / 01	N	01	00 / 01	00 / 00	00 / 00	00 / 00	NT	N
7	B1	1	4	NT	00 / 01	Y	10	00 / 01	00 / 00	00 / 00	00 / 00	NT	Y
8	B2	1	0	NT	00 / 00	Y	00	00 / 01	00 / 00	00 / 00	00 / 00	NT	Y
9	B2	1	1	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
10	B2	1	2	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
11	B2	1	3	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
12	B2	1	4	T	00 / 01	N	01	00 / 01	00 / 00	00 / 00	00 / 00	NT	N
13	B1	2	4	NT	00 / 01	Y	10	00 / 01	00 / 00	00 / 00	00 / 00	NT	Y
14	B2	2	0	NT	00 / 00	Y	00	00 / 01	00 / 00	00 / 00	00 / 00	NT	Y
15	B2	2	1	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
16	B2	2	2	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
17	B2	2	3	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
18	B2	2	4	T	00 / 01	N	01	00 / 01	00 / 00	00 / 00	00 / 00	NT	N
19	B1	3	4	T	01 / 01	N	11	00 / 01	01 / 00	00 / 00	00 / 00	NT	N

hit rate: $15/19 \cdot 100 = 78'94\%$

hit rate = $15/19 \cdot 100 = 78'94\%$

$$\frac{15 \text{ hits}}{19 \text{ hits} \cdot \text{miss}} = 0'78 \cdot \frac{1}{\text{miss}} = 0'78 \text{ hits}$$

BHR = Branch History Register

(1,2)

```
slt rd, rs, rt
    if rs < rt
      rd = 1
    if rs > rt
      rd = 0
```

```
loop: LS   F2,0(R2)
      MULS F4,F2,F2
      MULS F6,F4,F1
      S.S  F6,0(R4)
      DADDUI R2,R2,4
      DADDUI R4,R4,4
      DADDUI R3,R3,1
      BNE  R3,R1,loop
```

ALU	ALU	3	
ALU	SF	2	
Int	Int	0	
LD	ALU	1	
LD	SD	0	

Latencia → Huecos entre instruc. tal cual

↳ Si fuese Tiempo de ejecución, sería los huecos + la propia instrucción contado

- **NO** delay
  ↳ No hay que poner una instrucción debajo del BNE

ciclo	MEM	MEM	FP	FP	Int/Branch
1	LS F2,0(R2)				
2		—			DADDUI R2,R2,4
3			MULS F4,F2,F2		
4				—	DADUI R4,R4,4
5				—	DADDUI R3,R3,1
6				—	
7				MULS F6,F4,F1	
8			—		
9					
10	S.S F6,-4(R4)		—		BNE R3,R1,loop

Instruc	Issue	Execution	Write (→ solo 1 a la vez)	Commit
MULD F0,F4,F12	1	2-5	6	7
SUBD F4,F0,F10	2	7-8	9	10
LD F0,0(R1)	3	4	5	11
DIVD F0,F8,F14	4	6-25	26	27
SUBD F12,F8,F0	8	27-28	29	30
ADDD F10,F14,F24	11	12-13	14	31
LD F0,0(R1)	12	13	15	32
MULD F18,F14,F24	28	29-32	33	34

4 ROB

2+ 4 x 20 %

1 FU pa cada uno

CDB+1 mínimo

ROB en ciclo que finaliza 1º LD (11)

0	DIVD F0 F8 F4	Exec	F8/F14
1	SUBD F12 F8 F0	Wait issue	F8-F0
2	LD F0 0(R1)	Commit	MEM F0↷R1
3	ADD F10 F14F24	Issue	F14+F24

Local 2 bits global of (1,2) Selector 2 bit saturated T↑ NT↓
Local when 0 or 1 global 2 or 3. SOLO ACTUALIZA el predictor empleado

```
loop: LW R2,0(R1)
      ANDI R4,R2,1
      BEQZ R4,next
      DADDUI R1 R1 4
      DADDUI R2 R2 1
next: BNE R1,R3,loop
      SW R2,-4(R1)
```

B1

B2 → cuando 1 ya no avanza más, habrá llegado al final y nos dará 3

3, 2, 5, 4, 6

011 010
001 001
001 000

2^m n bits
2^1 2 bits

weak NT

Branch	T/NT?	Predicts	Local predictor	selector	Global Predictor BHR	BHT0	BHT1
—	—	—	01 01	00	0	01 01	01 01
B1(3)	NT	NT	00 01	00	0	01 01	01 01
B2(3)	T	NT	00 10	01	0	01 01	01 01
B1(2)	T	NT	01 10	10	0	01 01	01 01
B2(2)	T	NT	01 10	11	1	01 10	01 01
B1(5)	NT	NT	01 10	10	0	01 10	00 01
B2(5)	T	T	01 10	11	1	01 11	00 01
B1(4)	T	NT	01 10	11	1	01 11	01 01
B2(4)	T	NT	01 10	11	1	01 11	01 10
B1(6)	T	NT	01 10	11	1	01 11	10 10
B2(6)	NT	T	01 10	10	0	01 11	10 01

Fórmulas

$$Speedup = \frac{1}{(1-f) + f/s}$$

$$MIPS = \frac{clock\ freq}{CPI \cdot 10^6}$$

$$speedup = \frac{before}{after}$$

$$CPI = \frac{total\ cycles}{total\ instruc}$$

$$IPC = 1/CPI$$

Flynn Taxonomy

Unproc ← SISD

MISD → No use TPU

SIMD ← Vector, multimedia GPU

MIMD → SMP, NUMA, cluster, clad

UCLM	SCHOOL OF COMPUTER SCIENCE AND ENGINEERING COMPUTING SYSTEMS DEPARTMENT COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE	eviT escuela superior de ingeniería informática
Final written exam (June 2023)	**Test**	**Year 2022/23**

Name: _____

Indicate the correct answer to each test question **IN THE NEXT TABLE**:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Important: Test counts **3 points**. Each correct question counts 0.1 points. Wrong answers subtract 0.1/3 points. Not answered questions are not considered for the mark.

1. The peak performance of an architecture is:
   a. The performance of a machine is guaranteed not to exceed
   b. The performance of a machine when running the Linpack benchmark
   c. The maximum performance a machine achieves with the most favourable benchmark
   d. All of them are false

2. The CPI depends on:
   a. The architecture of the instruction set
   b. The CPU organisation
   c. On the complexity of the set of instructions
   d. All of them are true

3. Suppose you have two machine realisations of the same instruction set architecture. Machine A has a clock cycle of 1 ns and a CPI of 3 for a particular program, while machine B has a cycle time of 2 ns and a CPI of 1.5 for the same program. Which machine is faster for this program, how much faster?
   a. A is 20% faster than B
   b. B is 20% faster than A
   c. A is as fast as B
   d. All of them are false

4. We want to analyse the performance of a machine. In its instruction set there are three types of instructions, A, B, and C. and it needs 2, 6, and 4 clock cycles to execute instructions of type A, B, and C. The machine has a clock frequency of 800 MHz. The test program we are going to use has the following mix of instructions: 40% type A, 50% type B, and 10% type C. What is the peak MIPS speed of the machine?
   a. 190,5
   b. 200
   c. 400
   d. None of the above

5. Renaming floating point records allows to avoid.
   a. Data dependencies.
   b. Name dependencies.
   c. Control Dependencies.
   d. None of the above.

6. Mark which of the following statements is correct for MIPS64:
   a. Pipelining decreases the latency of a task.
   b. It uses one cache memory for instructions and another for data.
   c. The PC is updated at the final stage of each instruction (WB).
   d. None of the above.

7. Select the correct option for the following loop and the ADD.D and S.D instructions:

```
Loop: L.D F1, 0(R4)
      ADD.D F2, F1, F0
      S.D F2, 0(R1)
      DADDUI R4, R4, #-8
      BNEZ R4, Loop
```

   a. There is a data dependency, through F2, in the same iteration.
   b. There is an anti-dependence, through F2, in the same iteration.
   c. There is an output dependence, through F2, between iterations.
   d. There is a data dependence, through F2, between iterations.

8. The following code shows the modified loop that resulted from applying the software pipelining technique to an initially given code:

```
Loop: S.D F4, 16(R1)  ; 1 store M[i]
      ADD.D F4, F0, F2      ; 2 add a M[i-1]
      L.D F0, 0(R1)   ; 3 load M[i-2]
      DADDUI R1, R1, #-8    ; 4 update counter
      BNEZ R1, Loop  ; 5 jump if R1 = 0
```

   a. The initial code has the DADDUI instruction with an increment of #+8.
   b. The initial code has the DADDUI instruction with an increment of #-8.
   c. The initial code has the DADDUI instruction with an increment of #-4.
   d. The initial code does not have the DADDUI instruction.

9. In the Scoreboard technique, the results of the functional units:
   a. They are sent to the instructions that are waiting for them through the CDB.
   b. They are sent to the instructions that are waiting for them through the register bank.
   c. They are sent to instructions that are waiting for them through the load buffers.
   d. All are false.

10. Mark the correct option for dynamic instruction scheduling without speculation:
    a. Issue in order, execution out of order and completion in order.
    b. Issue in order, execution in order, and completion out of order.
    c. Issue in order, execution out of order and completion out of order.
    d. All are false.

11. Given a 5-stage pipeline in which 2 out of every 8 instructions are taken branches and there are 3 delay slots for each taken branch, indicate which of the following options is correct:
    a. CPI = 1.2, IPC = 0.9
    b. CPI = 1.33, IPC = 0.75
    c. CPI = 1.75, IPC = 0.57
    d. CPI = 1.1, IPC = 0.95

12. A code is executed on a 5-stage pipelined processor with 25% conditional branches, 80% of which are taken. It is assumed that, for conditional branches, the branch direction and target address are resolved in the MEM stage (CPI is 1 for all other instructions), and there is no branch prediction. Indicate the correct calculation of CPI:
    a. CPI = 1 + 0.8 * 0.25 * 1
    b. CPI = 1 + 0.8 * 0.25 * 2
    c. CPI = 1 + 0.25 * 0.8 * 3
    d. CPI = 1 + 0.25 * 0.8 * 4

13. What is the latency of a branch instruction?
    a. The number of cycles between the IF stage and the MEM stage.
    b. The number of additional cycles between the IF stage and the cycle of its complete resolution.
    c. The number of additional cycles between the ID stage and the cycle of its complete resolution.
    d. The number of cycles between the IF stage and the ID stage.

14. Consider the following source code after applying software speculation:

```
LD     R1, 1(R3)          ; load A
LD     R1, 0(R2)          ; speculative load of B
BEQZ   R1, L3             ; branch
DADDI  R1, R1, #4         ; else clause
L3:    SD     R1, 0(R3)   ; speculative store of A
```

    Indicate which of the following statements is correct:
    a. Software speculation is incorrect because the registers used by the speculative load of B have not been renamed.
    b. Software speculation is correct because the speculative load of B may not succeed.
    c. Software speculation is incorrect because the "else" clause should be executed before the speculative load of B.
    d. Software speculation is correct since the branch instruction will execute before the speculative load of B succeeds or fails.

15. Consider the following source code and assume that exception handling is performed using poisoned bits:

```
LD     R1, 1(R3)          ; load A
LD     R1, 0(R2)          ; speculative load of B
BEQZ   R1, L3             ; branch
DADDI  R1, R1, #4         ; else clause
L3:    SD     R1, 0(R3)   ; speculative store of A
```

The state of the registers before executing this code is:

R1	R2	R3	R4	R5	R6	R7	R8	R9
10	0	5	4	5	6	7	8	9

Furthermore, access to memory address 0 is prohibited, so any attempt to read from or write to that position will raise an exception for illegal access. Indicate which of the following statements is correct:

a. The load instruction for A generates an exception, and the program terminates.
b. The speculative load of B generates an exception, sets the poisoned bit in register R1 to 1, and the program continues correctly.
c. The speculative load of B generates an exception, sets the poisoned bit in register R1 to 1, but the program terminates when the branch instruction tries to access that register.
d. The speculative load of B generates an exception, sets the poisoned bit in register R1 to 1, and the program executes correctly, but the program terminates when the speculative store instruction tries to use register R1.

16. In a processor that uses the Tomasulo technique with speculation, indicate which of the following statements is correct:
    a. In the results' write stage, operations send their results to the reservation stations, and from there, they are sent to the CDB.
    b. In the execution stage, the operation is executed even if the operands are not available, as the CDB will provide them.
    c. In the commit stage, any out-of-order instruction can be completed as long as it is in the reorder buffer.
    d. In the issue stage, if a reservation station is available and there is space in the reorder buffer, the processor continues the execution of an instruction.

17. A difference between a superscalar processor and a VLIW one is...
    a. Conceptually there is no difference. "VLIW" is the name Intel gives to its superscalar processors.
    b. Unlike VLIWs, superscalars can have multiple instructions in the execution stage (EX).
    c. In VLIWs, instruction execution is always in order, while in superscalars it can also be out of order.
    d. None of the above.

18. A 4-way superscalar processor...
    a. Can issue 4 instructions per cycle.
    b. It has a speedup of 1/4 with respect to a scalar processor.
    c. It has a CPI of 4 cycles.
    d. None of the above.

19. In a typical superscalar processor, branch instructions are predicted...
    e. In the *Fetch* step
    f. In the *Decode* step
    g. In the *Execution* step
    h. In the *Commit* step

20. In a typical superscalar processor, in the case of a misprediction...
    a. An exception occurs that stops program execution.
    b. Program execution continues as normal along the predicted path.
    c. The results of the instructions (stored temporarily in the ROB) are copied to the corresponding registers and the execution of the program continues along the alternative path to the one predicted.
    d. None of the above.

21. We have the following optimized scheduling for a 2-way superscalar processor:

Cycle	Integer instruction	FP instruction
1	L.D F2, 0(R1)	
2	L.D F8, -8(R1)	
3	L.D F14, -16(R1)	MUL.D F4, F2, F0
4	L.D F6, 0(R2)	MUL.D F10, F8, F0
5	L.D F12, -8(R2)	MUL.D F16, F14, F0
6	L.D F18, -16(R2)	
7	DADDUI R1, R1, #-24	
8	DADDUI R2, R2, #-24	ADD.D F6, F4, F6
9	SLTI R3, R1, end	ADD.D F12, F10, F12
10		ADD.D F18, F16, F18
11	S.D F6, 24(R2)	
12	S.D F12, 16(R2)	
13	BEQZ R3, Loop	
14	S.D F18, 8(R2)	

    a. We can state that the execution step of the multiplication instruction takes 2 cycles.
    b. We can state that the execution step of the multiplication instruction takes 3 cycles.
    c. We can state that the execution step of the multiplication instruction takes 4 cycles.
    d. We can state that the execution step of the multiplication instruction takes 5 cycles.

22. We must schedule a sequential code consisting of 12 instructions for a VLIW processor with 6 operations per instruction. In the best case (assuming an ideal code and processor) we would have
    a. 2 long instructions.
    b. 3 long instructions.
    c. 4 long instructions.
    d. 6 long instructions.

23. When we improve integration technology:
    a. We quadratically increase the number of transistors/cm2
    b. We increase the heat dissipated quadratically
    c. We increase the power dissipation quadratically
    d. All of them are true

24. Neoverse are ARM processors that apply to:
    a. Low power consumption for microcontrollers and IoT
    b. Real-time for critical applications
    c. HPC
    d. Applications for computationally complex systems

25. The Big.Little ARM configuration is for:
    a. Having an integrated system for 32-bit and 64-bit applications on the same chip
    b. Have the possibility to address memory in two different formats
    c. Having two types of cores on the same chip
    d. All of them are false

26. The M2 processor has
    a. 8 cores
    b. RISC-V architecture
    c. Tri-Cluster configuration
    d. All of them are false

27. Which of the following is true?
    a. CLOUD is a collection of computers that belong to several owners in different locations and are connected together so that users can share the combined power of the resources.
    b. GRID is a collection of computers usually owned by a single party.
    c. Facebook is an example of CLOUD defined as PaaS (Platform as a Service)
    d. None are true

28. A message passing environment that provides tools for communicating processes is:
    a. CUDA
    b. OpenMP
    c. MPI
    d. None of them are true

29. If a thread is running until some event generates a delay, this means that the processor is multi-threading of:
    a. Fine-grained or interleaved
    b. Coarse-grained or blocked
    c. Simultaneous o SMT
    d. None of the above is true

30. Today in the TOP500 most computers are:
    a. Clúster
    b. MIMD
    c. SIMD
    d. MISD

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (June 2023)	Exercises	Year 2022/23

Name: _____

Important: Solve each exercise **in a different sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 1 (2 points). Let's consider the execution of the first 6 iterations of the following loop on a MIPS64 processor:

```
        DADD R1, R0, R0
    L1: BEQZ R1, L2       ; (B1)
        ...
    L2: BEQZ R1, L3       ; (B2)
        ...
    L3: XORI R1, R1, #1   ; (<-> NOT R1)
        J    L1
```

a) Indicate the sequence of branch instructions (B1/B2) executed and their actual outcomes (T/NT). Use the first and second columns in the table below.

b) Detail the behavior of a one-level branch predictor using 2 bits per branch instruction. Consider the initial state indicated in the table. Use columns 3, 4, and 5. Indicate at the end the hit rate for this predictor.

c) Detail the behavior of a two-level (2, 2) correlating branch predictor, assuming the same input vector, and considering the initial state indicated in the table. Use columns 6, 7, and 8. Indicate at the end the hit rate for this predictor.

a) Actual behavior		b) One-level predictor behavior			c) Two-level predictor behavior						
Branch instruction executed (B1/B2)	Actual outcome (T/NT)	Prediction bits for B1 and B2	Predicted outcome (T/NT)	Hit (Y/N)	Prediction bits (BHR and BTHs)					Predicted outcome (T/NT)	Hit (Y/N)
(Initial state)	-	00 00	-	-	00	00 00	00 00	00 00	00 00	-	-
B1	T	01 00	NT	N	01	01 00	"	"	"	NT	N
B2	T	01 01	NT	N	11	"	00 01	"	"	NT	N
B1	NT	00 01	NT	Y	10	"	"	"	00 00	NT	Y
B2	NT	00 00	NT	Y	00	"	"	00 00	"	NT	Y
B1	T	01 00	NT	N	01	10 00	"	"	"	NT	N
B2	T	01 01	NT	N	11	"	00 10	"	"	NT	N
B1	NT	00 01	NT	Y	10	"	"	"	00 00	NT	Y
B2	NT	00 00	NT	Y	00	"	"	00 00	"	NT	Y
B1	T	01 00	NT	N	01	11 00	"	"	"	T	Y
B2	T	01 01	NT	N	11	"	00 11	"	"	T	Y
B1	NT	00 01	NT	Y	10	"	"	"	00 00	NT	Y
B2	NT	00 00	NT	Y	00	"	"	00 00	"	NT	Y
			Hit rate:	50%						Hit rate:	66'66%

6/12 =
= 0'5 · 100 =
= 50%

8/12 =
= 0'6666 · 100
= 66'66%

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final written exam (June 2023)	Exercises	Year 2022/23

Name: _____

Important: Solve each exercise **in a different sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

→ Pa branchs

Exercise 2 (3 points). The following MIPS64 code is going to be executed on a processor without speculation but with dynamic scheduling using the Tomasulo technique:

↳ Out-of-order

```
L.D     F2, 0(R1)      /* I1 */
L.D     F4, 8(R1)      /* I2 */
L.D     F6, 16(R1)     /* I3 */
ADD.D   F8, F2, F4     /* I4 */
MUL.D   F10, F2, F6    /* I5 */
DIV.D   F0, F12, F8    /* I6 */
S.D     F0, 32(R1)     /* I7 */
```

IF ID EX MEM WB
IF ID EX MEM WB
IF ID EX MEM WB
IF ID EX EX EX MEM WB
IF ID EX EX ...

The mentioned processor has the following characteristics:

- 1 cycle for loads and stores.
- 4 cycles for floating-point multiplication (FP).
- 3 cycles for floating-point addition (FP).
- 6 cycles for floating-point division (FP).
- Reservation stations per FP unit: 2 entries.
- Load buffer: 2 entries.
- Store buffer: 1 entry.
- The execution stage of an instruction waiting for a result cannot start in the same cycle in which the result is written to the CDB (Common Data Bus).

Please answer the following questions for the given code and the provided processor configuration.

RAW, WAR, WAW

a) Identify the dependencies between instructions and indicate their type in the following table:

Instruction 1	Instruction 2	Type of dependency
I1	I4	RAW
I2	I4	RAW
I3	I5	RAW
I4	I6	RAW
I6	I7	RAW (En R0)
I1	I5	RAW

b) Indicate the stall cycles between instructions, if any, in the gaps in the previous code that appear in the following table:

L.D F2, 0(R1)	
	0
L.D F4, 8(R1)	
	0
L.D F6, 16(R1)	
	0
ADD.D F8, F2, F4	
	2
MUL.D F10, F2, F6	
	0
DIV.D F0, F12, F8	
	5
S.D F0, 32(R1)	
	8

c) Fill in the following table using the Tomasulo technique:

Instruction	Issue	Execution	Commit
L.D F2, 0(R1)	1	2	3
L.D F4, 8(R1)	2	3	4
L.D F6, 16(R1)	3	4	5
ADD.D F8, F2, F4	4	5-7	8
MUL.D F10, F2, F6	5	6-9	10
DIV.D F0, F12, F8	6	9-14	15
S.D F0, 32(R1)	7	16	17

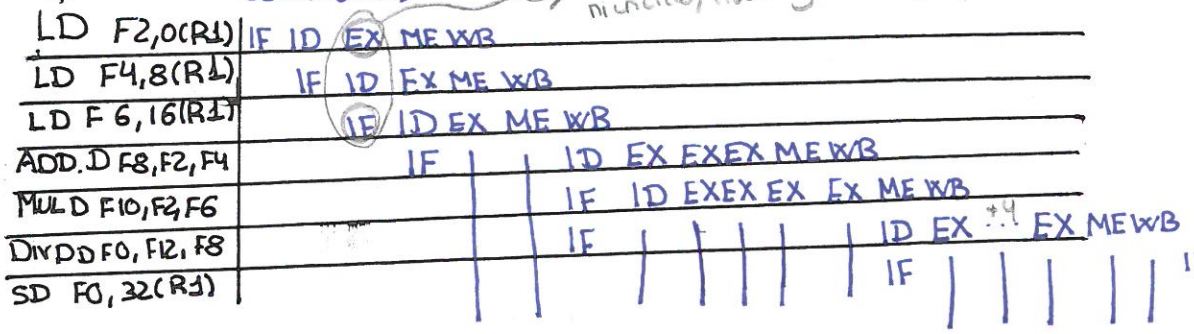
Sies in order puede no ir justo después de su exec

Sólo se salta ciclos si las RS (Reservation Stations) están llenas

Necesita la info de los registros

Un load no toma nunca, libera y entra en 3 aunque haya sólo 2 RS

b)



Name: _____

Important: Solve this exercise **on this sheet**. You can use more sheets if you need, but do not forget to write down your name in all the sheets you deliver.

Exercise 3 (2 points) A compiler for a VLIW processor with the following features:

- Two memory operations, two FP operations, and one integer (or jump) can be issued in each cycle.
- All FP units are fully segmented
- The jump delay is ignored

has generated the following output program, making the code as compact as possible:

ycle	Mem 1	Mem 2	FP 1	FP 2	integer/branch
1	L.D F2, 0(R1)	L.D F8, -8(R1)	NOP	NOP	NOP
2	L.D F14, -16(R1)	L.D F20, -24(R1)	NOP	NOP	NOP
3	L.D F6, 0(R2)	L.D F12, -8(R2)	MUL.D F4, F2, F0	MUL.D F10, F8, F0	NOP
4	L.D F18, -16(R2)	L.DF24, -24(R2)	MUL.D F16, F14, F0	MUL.D F22, F20, F0	NOP
5	NOP	NOP	NOP	NOP	NOP
6	NOP	NOP	NOP	NOP	DADDUI R1, R1, #-32
7	NOP	NOP	ADD.D F6, F4, F6	ADD.D F12, F10, F12	NOP
8	NOP	NOP	ADD.D F18, F18, F16	ADD.D F24, F22, F24	NOP
9	NOP	NOP	NOP	NOP	NOP
10	NOP	NOP	NOP	NOP	SLTI R3, R1, fin
11	S.D F6, 0(R2)	S.D F12, -8(R2)	NOP	NOP	DADDUI R2, R2, #-32
12	S.D F18, -16(R2)	S.D F24, 8(R2)	NOP	NOP	BEQZ R3, Loop

Assume that the code is as compact as possible, and the processor employs forwarding and delayed branch.

a. Indicate the cycles used by the functional units (FU) of the processor and complete the following table:

FU= Functional unit	Number of cycles
Integer FU	1
Adder FP FU	3
Multiplier FP FU	4
MEM stage of the memory access instructions	2

Lo que yo quiera!!!

b. Show the initial code for the MIPS64 processor segmented such that at each turn of the loop a result is calculated. Write the code in this table (use as many rows as you need):

Inst. 1	LD F2, 0(R1)
Inst. 2	LD F6, 0(R2)
Inst. 3	MUL.D F4, F2, F0
Inst. 4	ADD.D F6, F4, F6
Inst. 5	DADDI R1, R1, +8
Inst. 6	SLTI R3, R1, fin
Inst. 7	DADDUI R2, R2, #8

Pues no, subnormal, mira la tabla que dan

Inst. 8	SD F6, 0 (R2)
Inst. 9	BEQZR3, loop
Inst. 10	
Inst. 11	
Inst. 12	
Inst. 13	
Inst. 14	
Inst. 15	

c. What is the speedup obtained by the VLIW processor with respect to the code in section b) of the segmented MIPS64 processor? Explain the calculations made and consider that the code is scheduled to obtain the best performance (i.e. the lowest number of stalls).

12 cdos/itoación VLIW

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final exam (January 2021)	Test	Year 2020/21

Name: _____ **Aula 1.5. Mesa número:** ____

Indicate the correct answer to each test question in the next table:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Important: Test counts **3 points**. Each correct question counts 0.1 points. Wrong answers subtract 0.1/3 points. Not answered questions are not considered for the mark.

1. Two examples of processors with the same organization but different architecture are:
   a) Itanium and Pentium
   b) AMD Opteron and Pentium
   c) ARM and Alpha
   d) None of the above

2. MIPS allow us to compare the performance of:
   a) Computers with the same instruction set
   b) Computers with different instruction set
   c) Compilers
   d) All the above

3. We want to compare the performance of two different systems: S1 and S2. S1 costs 1000 € and S2 costs 1200 €. The following times have been obtained in these systems:

Program	S1 time	S2 time
1	30 s	26 s
2	8 s	7 s

   Which machine is more cost effective and how much?
   a) S1 is 1.042 more cost effective than S2
   b) S1 is 1.14 more cost effective than S2
   c) S2 is 1.14 more cost effective than S1
   d) None of the above

4. We have a computer with a clock frequency of 1000MHz, requiring the following number of clock cycles to execute each type of instruction:
   - Floating point multiplication: 6 cycles
   - Floating point addition: 4 cycles
   - Floating point division: 20 cycles
   - Integer instructions: 2 cycles

   Consider a program (P), with the next instruction mix:
   - Floating point multiplication: 10%
   - Floating point addition: 15%
   - Floating point division: 5%
   - Integer instructions: 70%

   The native MIPS value of this computer is:
   a) 277.78
   b) 321.3
   c) 211.1
   d) None of the above

5. We are looking for a benchmark to measure the performance of the new floating-point unit of a computer. This unit runs 4 times faster than the old one. We want to get a speedup of 2 for the benchmark we are considering. If the execution time of this benchmark in the old floating-point hardware is 100 seconds, how much of that initial execution time should the floating-point instructions count to get this speedup?
   a) 80 s
   b) 66.6 s
   c) 50 s
   d) None of the above

6. Given two instructions *i* and *j* (*i* precedes *j*), there is a data dependence between them if:
   a) Both instructions update the same register or memory position.
   b) Instruction *i* produces a result that may be used by instruction *j*.
   c) Instruction *j* produces a result that may be used by instruction *i*.
   d) Instruction *j* writes a register or memory position that instruction *i* reads.

7. In this loop:

```
Loop:  L.D    F0, 0(R1)    ;I1
       MUL.D  F2, F0, F1   ;I2
       SUB.D  F3, F2, F0   ;I3
       S.D    F3, 0(R1)    ;I4
       DADDUI R1, R1, #-8  ;I5
       BNEZ   R1, Loop     ;I6
```

   Between I1 and I5 there is...
   a) A data dependence between iterations
   b) A data dependence in the same iteration
   c) An antidependence between iterations
   d) An output dependence in the same iteration

8. Consider the next high-level code (A and B are two previously defined arrays):

```
for(i=0; i<100; i++) {
      A[i+1] = A[i] + B[i];    // S1
      B[i]   = A[i] * 2;       // S2
}
```

   a) There is an antidependence, between S1 and S2, through B (between iterations)
   b) There is a data dependence, between S1 and S2, through A (in the same iteration)
   c) There is a antidependende, between S1 and S1, through A (between iterations)
   d) None of the above

9. Output dependences:
   a) Are removed by software pipelining
   b) Are removed by register renaming
   c) Are removed by trace scheduling
   d) None of the above

10. We unroll 3 times this loop (assuming that the number of iterations is divisible by 3):

```
Loop:  L.D    F0, 0(R1)    ;I1
       ADD.D  F2, F0, F1   ;I2
       SUB.D  F3, F2, F0   ;I3
       S.D    F3, 0(R1)    ;I4
       DADDUI R1, R1, #-8  ;I5
       BNEZ   R1, Loop     ;I6
```

    a) The resulting loop will be composed of 6 instructions.
    b) The resulting loop will be composed of 9 instructions.
    c) The resulting loop will be composed of 14 instructions.
    d) The resulting loop will be composed of 18 instructions.

11. After unrolling 3 times the loop of the previous question, the immediate value of the `DADDUI` instruction will be:
    a) `#8`
    b) `#-8`
    c) `#24`
    d) `#-24`

12. In Scoreboard, the order in which the issue step is performed is given by:
    a) The availability of functional units.
    b) The order of the instructions in the program.
    c) The antidependences in the program.
    d) None of the above.

13. To implement the BHR of a (2,2) predictor we use:
    a) A binary adder
    b) A saturating counter
    c) A shift register
    d) None of the above

14. A (2,4) correlating predictor uses exactly:
    a) 2 BHTs with 4 entries
    b) 2 BHRs and 4 BHTs
    c) 4 BHRs and 2 BHTs
    d) None of the above

15. Our processor uses a 2-bit predictor, initialized to 11, for each branch. What is the number of mispredictions for this code?

```
for (i=0; i<10; i++)            ; branch B1
     for (j=0; j<10; j++) {     ; branch B2
          . . .
     }
```

    a) 9
    b) 10
    c) 11
    d) None of the above

16. Compute the CPI of a code in which 30% of the instructions are conditional branches. The code is executed in a 5-stage pipelined processor without BTB, in which the branch target address is known at the end of the MEM stage and the branch outcome (taken or not taken) is known at the end of the EX stage. Assume that 75% of the conditional branch instructions are taken, and a CPI=1 in all cases, except for the taken branches.
    a) CPI=1.675
    b) CPI=1.6
    c) CPI=1.45
    d) CPI=1.225

17. A speculative instruction:
    a) Is an instruction that removes stalls due to control hazards.
    b) Is an instruction whose execution depends on a previously established condition.
    c) Is an instruction causing a terminating or resumable exception.
    d) None of the above.

18. What will be the behavior of a MIPS64 processor that executes the speculative instruction "`ADD.D* F6,F7,F8`", assuming the "poison bits" technique and knowing that the poison bits of F7 and F8 are initially 0 and 1, respectively?
    a) The poison bit of F8 will be reset (poison bit=0)
    b) The poison bit of F7 will be set (poison bit=1)
    c) The poison bit of F6 will be reset (poison bit=0)
    d) None of the above

19. A superscalar processor of degree *n*:
    a) Issues a variable number of instructions per cycle, which is always less or equal to *n*.
    b) Limits to *n/2* the number of memory accesses that can be simultaneously issued.
    c) Improves the performance, by reducing the number of instructions executed.
    d) Has a reduced hardware cost.

20. In the *issue* step of a un superscalar processor:
    a) The instruction is decoded and sent to the instructions window.
    b) The processor determines which instructions in the window can be sent to execution.
    c) The execution of the instruction is carried out in the corresponding functional unit.
    d) Resources and operands are checked.

21. A critical aspect to consider in the *fetch* stage of a superscalar processor is:
    a) Dynamic renaming.
    b) Recovery in case of misprediction.
    c) Instruction reading bandwidth.
    d) Resources and operands checking.

22. Among the advantages of VLIW processors, we could highlight:
    a) They perform a dynamic scheduling.
    b) Hardware detects dependencies, since complier is very simple.
    c) Hardware is very simple, since complexity resides in the compiler.
    d) They do not have to unroll the code.

23. The power dissipated by a processor is proportional to:
    a) Frequency
    b) Voltage squared
    c) Transistor count
    d) All the above

24. A difference between the P6 and P7 architectures is:
    a) P6 uses more decoders
    b) P7 does not use a barrel shifter
    c) L1 data cache in P6 is bigger
    d) All the above

25. One of the advantages of a multicore over a single core is:
    a) Less PCB than a SMP, which implies less energy
    b) A better use of chip area
    c) The share the same bus and memory bandwidth
    d) All of them are disadvantages

26. Indicate the correct statement:
    a) Quantum computers will replace general purpose processors in the medium term.
    b) Heterogeneous computing is present in many of today's mobile phones.
    c) Heterogeneous computing is only applicable to large supercomputers.
    d) None of the above.

27. In a multi-threading processor, several instructions from different threads are issued to be executed at the same time when the type of multi-threading is:
    a) Fine-grained multi-threading
    b) Coarse-grained multi-threading
    c) Simultaneous multi-threading
    d) Chip multiprocessing

28. GRID services include:
    a) Resource discovering and monitoring
    b) Resource allocation and management
    c) Security infrastructure and file transfer
    d) All the above

29. Indicate the correct statement:
    a) A SIMD architecture is less energy efficient than a MIMD
    b) MMX extensions of Intel processors are MISD extensions
    c) In GPUs there is not penalty due to changes of context
    d) None of the above

30. Skype or Gmail are examples of:
    a) Cloud SaaS
    b) Cloud PaaS
    c) Cloud IaaS
    d) Cloud JaaS

UCLM

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final exam (January 2021)	Exercises	Year 2020/21

Name: _____

Important: Solve this exercise **on this sheet**, and do not forget to write down your name in all the sheets you deliver.

Exercise 2 (2 points). We have a MIPS64 processor equipped with hardware speculation according to the "Tomasulo" method. The processor has the following characteristics:

- There is an 8-entry ROB (initially EMPTY), 1 load buffer, 1 reservation station (RE) for FP addition, 2 reservation stations for FP multiplication, and 2 reservation stations for FP division.

Functional Unit	Amount	Cycles	Pipelined
FP Adder	1	2	Yes
FP Multiplier	1	4	Yes
FP Divider	1	8	No
LOAD/STORE	1	2	Yes
INT ALU	1	1	No

puede nueva op cada ciclo
solo una op a la vez

- In the same clock cycle in which a result is written in the Common Data Bus (CDB) the execution of any instruction that was waiting for that result is initiated.
- A resource that is released/freed in a clock cycle cannot be used until the next cycle.

a) Complete the following table, indicating the cycles in which each stage is carried out.

Repeto el orden por ROB

Un commit es por instrucción en el ROB

Instruction	Issue	Execute	Write result	Commit
1.- MUL.D F2, F1, F0	1	2-5	6 (Lib)	7
2.- DIV.D F0, F2, F4	2	6-13	14	15
3.- ADD.D F2, F4, F5	3	9-5	6	16
4.- S.D F2, 0(R1)	4	6-7	8	17
5.- MUL.D F4, F2, F6	(Lib) 7	8-11	12	18
6.- DIV.D F6, F1, F2	15	16-23	24	25
7.- L.D F6, 0(R2)	25	26-27	28	29
8.- ADD.D F2, F0, F6	28	29-30	31	32
9.- MUL.D F3, F0, F2	31	32-35	36	37
10.- ADD.D F3, F0, F6	36	37-38	39	40

b) Complete the following table, indicating the reason for each of the stalls when running the above code (for example, **Stall due to:** "structural hazard due to the RE", **Where?:** "execution of instruction number 3").

Stall due to	Where?
Data hazard due to F2	Exec of instruc 2
Struc hazard due to ROB	Commit instruc 3
Data hazard due to F2	Exec instruc 4
Struc hazard due RS	Issue instruc 5

Stall due to	Where?
Struc hazard due FU	Occupied by DIV.D2

```
LW RS,0(R1)
SLT RS,R4,RS
DADD R3,R3,RS
SW RS,0(R2)
DADDUI R2,R2,4
DADDUI R2,R2,-1
BNE R2,R0,read
```

```
LW RS,0(R1)
SLT RS,R4,RS LW RS,4(R1)
DADD R3,R3,RS SLT RS,R4,RS LW RS,8(R1)
SW RS,0(R1) DADD R3,R3,RS SLT RS,R4,RS LW RS,12(R1)
SW RS,4(R1) DADD R3,R3,RS SLT RS,R4,RS
SW RS,8(R1) DADD R3,R3,RS
SW RS,12(R1)
```

```
stat-up: LW RS,0(R1)
SLT RS,R4,RS
LW RS,4(R1)
DADD R3,R3,RS
SLT " " "
LW RS,8(R1)
DADDUI R0,R0,-12
0,4,8,12
```

```
loop: SW RS,0(R1)
DADD R3,R3,RS
SLT RS,R4,RS
LW RS,12(R2)
```

```
finish-up:
SW RS,0(R1)
DADD R3,R3,RS
SLT RS,R4,RS
SW RS,4(R1)
DADD R3,R3,RS
SW RS,8(R1)
```

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE

Final exam (January 2021)	Exercises	Year 2020/21

Name: _____

Important: Solve this exercise **on this sheet**, and do not forget to write down your name in all the sheets you deliver.

Exercise 1 (2.5 points). We have installed a CO_2 monitor in our classroom. Every 5 minutes, it gets a new sample of the CO_2 level. Each sample is a 32-bit positive value. All these samples are stored in a set of consecutive memory positions (starting from the address in R1). Assuming that R2 contains the number of samples taken during a whole day, the following MIPS64 code counts (in R3) the number of times the CO_2 level has exceeded a certain threshold (in R4), resets all the samples below that threshold and writes 1 in the rest of them. Assume that R1, R2, R3 and R4 have been previously initialized.

```
read: LW     R5, 0(R1)      #I1 R5 = next CO2 sample
      SLT    R5, R4, R5     #I2 checks if it exceeds the threshold (R4)
      DADD   R3, R3, R5     #I3 updates the counter (R3)
      SW     R5, 0(R1)      #I4 overwrites the sample
      DADDUI R1, R1, #4     #I5 R1 = address of the next CO2 sample
      DADDUI R2, R2, -1     #I6 updates R2
      BNE    R2, R0, read   #I7 If R2<>0 read the next CO2 sample
```

Assume that we have a pipelined processor with following characteristics:

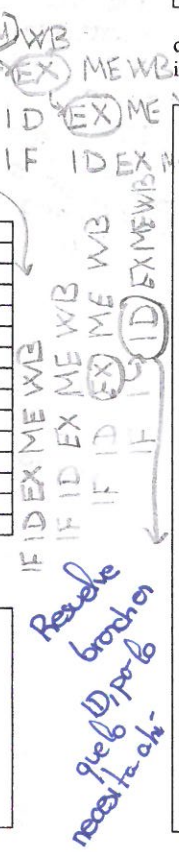
- Operand forwarding is supported
- A value can be written and read from a register in the same clock cycle
- All the dependences are detected and solved in the ID stage
- Branches are completely solved in the ID stage
- Delayed branch is NOT supported

a) Indicate the stalls in the code between each pair of instructions:

```
LW     R5, 0(R1)      #I1 R5 = next CO2 sample
stalls → NO
SLT    R5, R4, R5     #I2 checks if it exceeds the threshold (R4)
stalls → LW, data. 1 stall
DADD   R3, R3, R5     #I3 updates the counter (R3)
stalls → NO, Forwarding EX→EX
SW     R5, 0(R1)      #I4 overwrites the sample
stalls → NO MODIFICA
DADDUI R1, R1, #4     #I5 R1 = address of the next CO2 sample
stalls → NO MODIFICA
DADDUI R2, R2, -1     #I6 updates R2
stalls → NO MODIFICA
BNE    R2, R0, read   #I7 If R2<>0 read the next CO2 sample
stalls → 1, necesita en ID
Next instruction after the BNE
```

b) Apply software pipelining and show the modified loop body.

read:



c) Show the start-up and finish-up blocks (do not schedule them):

Start-up:

Finish-up:

d) Show the complete code (do not schedule it, but remember that the new code must behave exactly like the initial one):

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING SYSTEMS DEPARTMENT
UCLM — **COMPUTER SCIENCE DEGREE. COMPUTER ARCHITECTURE**

Final exam (January 2021)	**Exercises**	**Year 2020/21**

Name: _____

Important: Solve this exercise **on this sheet**, and do not forget to write down your name in all the sheets you deliver.

Exercise 3 (2.5 points). Given the following code fragment, where R2 and R3 contain the initial and final memory addresses where the data are stored:

```
Bucle: L.D    F10, 0(R2)
       ADD.D  F8, F10, F12
       S.D    F8, 0(R2)
       DADDUI R2, R2, -8
       BNE    R2, R3, Bucle
```

We want to compare the number of instructions issued per cycle by processors A and B, with the following common characteristics:

- We assume forwarding and delayed branch.
- All the units are fully pipelined.
- The next latencies:

Instruction producing the result	Instruction using the result	Latency in cycles
ALU FP	ALU FP	3
ALU FP	S.D	1
L.D	ALU FP	1
L.D	S.D	0

Each processor presents the following specific characteristics:

- **Processor A**. It is a MIPS64 5-stage superscalar processor, capable of issuing 4 instructions per cycle: two integer instructions (including memory accesses and branches) and two floating-point instructions.
- **Processor B**. It is a VLIW processor, capable of issuing in each cycle four operations per instruction. It does not impose restrictions on the type of operations that can be issued.

a) For processor A, unroll 8 times the code and schedule it as compact as possible, by indicating the instructions issued in each cycle. Use as rows as you need in the next table:

Cycle	Instr. 1 (MEM)	Instr. 2 (MEM)	Instr. 3 (FP)	Instr. 4 (FP)
1	LD F10,0	LDF16,-8		
2	LD F20,-16	LDF24,-24	ADD F8,F10	ADD F14,F16
3	LD F28,-32	LD F31,-40	ADD F18,F20	ADD F22,F24
4	LD F4,-48	LDF7,-56	ADD F26,F28	ADD F30,F31
5	SD F8,0	SD F14,-8	ADD F2,F4	ADD F6,F7
6	SD F18,-16	SD F22,-24		
7	SD F26,-32	SD F30-40		
8	SD F2,-48	SD F6,-56		
9	DADDUI R2,R2			
10		BNE R2,R36uc		

Fixed: F12

→ Por la data dependency, espera 1 más

Todo arriba
4 LD
4 LD
4 ADD
4 ADD
4 SD
4 SD
2 BNE

b) For processor B, schedule the unrolled code, assuming that static scheduling is as aggressive as possible and as efficient and possible to optimize the resulting code. You can use register renaming and out-of-order scheduling. Use as rows as you need in the next table:

Cycle	Operation 1	Operation 2	Operation 3	Operation 4
1				
2				
3				
4				
5				
6				
7				

c) Compare the number of operations per cycle for each processor. Which processor gets the best efficiency in operations per cycle?

a) $\frac{26 \text{ instruc}}{10 \text{ ciclos}} = 2{,}6$ inst/ciclo b) $\frac{26 \text{ instruc}}{7 \text{ ciclos}} = 3{,}70$ inst/ciclo

↑ ↑ mejor ↑↑

LD F10,0(R2) LD F16,-8(R2) LD F20,-16(R2) LD F24,-24(R2)
ADD.D F8,F10,F12 ADDD F14,F16,F12 ADDD F18,F20,F12 ADDD F22,F24,F12
SD F8,0(R2) SD F14,-8(R2) SD F18,-16(R2) SD F22,-24(R2)

LD F28,-32(R2) LD F31,-40(R2) LD F4,-48(R2) LD F7,-56(R2)
ADDD F26,F28,F12 ADDD F30,F31,F12 ADDD F2,F4,F12 ADDD F6,F7,F12
SD F26,-32(R2) SD F30,-40(R2) SD F2,-48(R2) SD F6,-56(R2)

DADDUI R2,R2,-
BNE R2,R3,Bucle

```
loop: LB   R2, 0(R1)
      DADD R3, R2, R0
      BNE  R2, R4, not_a
DADDUI  R5, R5, 1
DADDUI  R6, R6, 1   <-- not_a:
DADDUI  R1, R1, 1
 BNE2   R3, loop
```

Carga "exam18"

(m,n)
(2,1)

2^m branches of n bits

Branch executed	Actual outcome	Predicted outcome	Hit/miss	BHR	BHTs
(initial state)	-	-	-	00	00 00 00 00
B1 (e)	T	NT	miss	01	10 00 00 00
B2 (e)	T	NT	miss	11	10 01 00 00
B1 (x)	T	NT	miss	11	10 01 00 10
B2 (x)	T	NT	miss	11	10 01 00 11
B1 (a)	NT	T	miss	10	10 01 00 01
B2 (a)	T	NT	miss	01	10 01 01 01
B1 (m)	T	NT	miss	11	10 11 01 01
B2 (m)	T	T	hit	11	10 11 01 01
B1 (1)	T	NT	miss	11	10 11 01 11
B2 (1)	T	T	hit	11	10 11 01 11
B1 (8)	T	T	hit	11	10 11 01 11
B2 (8)	T	T	hit	11	10 11 01 11
B1 (.)	T	T	hit	11	10 11 01 11
B2 ( )	NT	T	miss	10	10 11 01 10

hit rate: 5/14 · 100 = 35'75%

```
loop: LW   R3, 0(R1)
   BEQZ2   R3, next      B1
   DMUL    R3, R3, R5
     SW    R3, 0(R1)
next: DADDUI R1, R1, 4
   DADDUI  R2, R2, -1
   BNEZ2   R2, loop
```

Carga: 0,1,1,2,3,5

R2 → Cantor of numbe

Most significat bit = el que + pesoticne
2-bit saturating → 00, 01, 10, 11

Branch executed	T/NT	B2 Predictor used (P1/P2)	Predicted T/NT	Hit/miss	Cantors B1	Cantors B2	Predictor 1 B1	Predictor 1 B2	Predictor 2 B1	Predictor 2 B2
					00	00	0	0	000	000
B1 0	T	P2	NT	miss	00	00	1	0	001	000
B2	T	P2	NT	miss	00	00	1	1	001	001
B1 1	NT	P2	NT	hit	00	00	0	1	000	001
B2	T	P2	NT	miss	00	01	0	1	000	010
B1 1	NT	P2	NT	hit	00	01	0	1	000	010
B2	T	P2	NT	miss	00	10	0	1	000	011
B1 2	NT	P2	NT	hit	00	10	0	1	000	011
B2	T	P1	T	hit	00	11	0	1	000	100
B1 3	NT	P2	NT	hit	00	11	0	1	000	100
B2	T	P1	T	hit	00	11	0	1	000	101
B1 5	NT	P2	NT	hit	00	11	0	1	000	101
B2	NT	P1	T	miss	00	11	0	0	001	100

Instruc	Issue	Execute	Write	Commit
LD F0, 0(R1)	1	2-3	4	5
LD F2, 0(R1)	2	3-4	5	6
LD F4, 0(R3)	5	6-7	8	9

Pq este salto? pq hay 2 RS (Reservation stations) por FU. Así se vería al llegar a la 3ª instruc ⇒ RS1 | Busy / RS2 | Busy. Ambas ocupadas

MULD F6, F0, F2	6	7-12	13	14

Hay data hazards pero no afecta pq la instruc 3 hizo stalls por su problema. Si llega a haber 3 RS de Mem (LD o SD), MULD debería haber esperado igualmente al ciclo 6 para ejecutar ya que es cuando F2 tiene el resultado. Esperaría al 6 ya que tiene que esperar al siguiente clock cycle del que escribe

ADD F8, F0, F4	7	9-10	11	15 → por el ROB debe acabar en orden

¿y el ciclo 8? → No puede ejecutar pq es ahí cuando LD F4, 0(R3) escribe y hay que esperar un ciclo más

DIV.D F10, F8, F6	8	14-23	24	25

¿Este salto? → por data dependency con MULD F6, F0, F2 en F6

SD F10, 0(R4)	9	25-26	27??	28

¿salto? Data dependency con DIV D F10, F8, F6 en F10

↳ pq las "???" los stores no escriben en memoria, ACCEDEN, pero en clase lo tomamos como escritura.

→ Pipeladas!!

FU	FU	Cycles
ADD	1	4
MULT	1	6
DIV	1	8
LD/SD	1	1
Int	1	1

Instruc	Issue	Read	Exec	Write
LD F0, 0(R1)	1	2	3	4
LD F2, 4(R1)	5 (Estruc)	6	7	8
LD F4, 8(R1)	9 (Estruc)	10	11	12
MULD F6, F0, F2	10	11	12-17	18
MULD F4, F8, F6	19 (Estruc)	20	21-26	27
ADDD F8, F0, F8	20	21	22-25	28 (order)
SD F4, 0(R4)	21	28 (RAW)	29??	30

→ Regla de bloqueo: cycle+1

→ Para no sobreescribir

MAL, por esta pipelada podrían estar ambas a la vez, siempre y cuando NO sea en la misma etapa

Tomasulo

Reservation Station

Name	Q_j	Q_K	V_j	V_K	Address	Data	Operation

DIVD F4, F2, F0
ADD F6, F4, F3
↓ K

(Supongo fin in-order)

Si añado un Register Result Status

FU	F0	F2	F4	...	F30

puede ser fuera de orden

Instruc	ISSUE	Execute	Write	Commit
LD F6,34(R2)	1	2 - 3	4	5
LD F2, 45(R3)	2	3 - 4	5	6
MULT F0, F2, F4	3	6 - 15	16	17
SUB F8, F6, F2	4	6 - 7	8	18
DIV F10, F0, F6	5	17 - 56	57	58
ADD F6, F8, F2	6	19 - 20	21	59

3 LD stat.
3 ADD stat.
2 Mult stat
LD → 2 cycle
ADD → 2 "
MUL → 10 "
DIV → 40 "
} exec.

RS

Time	Name	Busy	OP	VJ	VK	QJ	QK
	ADD1	Yes	SUB 34(R2)				Load2
	ADD2	Yes	ADD		LD2	ADD1	
	ADD3						
	MULT1	Yes	MULT		F4	Load2	
	MULT2	Yes	DIV		F6	Mult1	

{ Para en ciclo 4
{ Para en cycle 3, QJ: esperando de LD
{ Para en ciclo 5

Mini-exercises

2) Benchmark executed on 40MHz processor → F: clock rate

Instruc type	inst Cant	clock cycle
Arith	45000	1
Branch	32000	2
LD/SD	15000	2
FP	8000	2

clock speed = 40MHz = $40 \cdot 10^6$ Hz

a) Avg CPI?

$$\frac{\text{Cycles to execute}}{\text{Instruc. Cant}} = \frac{45000 \cdot 1 + 32000 \cdot 2 + 15000 \cdot 2 + 8000 \cdot 2}{45000 + 32000 + 15000 + 8000} = 1'55$$

(≈ 155 000 / ↳ 100 000)

b) Exec. time?

$$\frac{\text{Total cycles}}{\text{freq}} = \frac{155000}{40 \cdot 10^6} = 0'003875\,s$$

c) MIPS rate?

$$\frac{\text{Instruction Cant}}{\text{Exec time}} = \frac{100\,000}{0'003875} = 25'8\ \text{MIPS}$$

c) Throughput?

$$\frac{\text{MIPS} \cdot 10^6}{\text{Instruc Cant}} = \frac{25'8 \cdot 10^6}{100\,000} = 258$$

$$\frac{f}{\text{InstrucCant} \cdot \text{CPI}} = \frac{40 \cdot 10^6}{100\,000 \cdot 1'55} = 258'06\ldots$$

3) 100000 instruc on 500 MHz processor

Instruc type	CPI	instruc %
Arith	1	60%
FP	2	20%
LD/SD	4	10%
MEM	6	10%

a) AVG CPI = $1 \cdot 0'6 + 2 \cdot 0'2 + 4 \cdot 0'1 + 6 \cdot 0'1 = 2 \frac{\text{cycles}}{\text{inst}}$

b) MIPS = $\frac{f}{\text{CPI}} = \frac{500 \cdot \text{MHz}}{2} = 250$

VLIW and Superscalars Exercises

Int = 1 FPDiv = 16
FP mult = 5

1) Superscalar issues 4 instruc per cycle. 2 integer, 2 FP. FPADD ⇒ 3 cycles per exec

```
loop: LD   F0,0(R1)
      ADD.D F4,F0,F2
      SD   F4,0(R1)
      DADDUI R1,R1,-8
      BNEZ R1,loop
```

a) Unroll 8 times, schedule as compact as possible

```
      LD   F0,0(R1)
      ADD  F0,F0,F2
      SD   F0,0(R1)
      LD   F4,-8(R1)
      ADD  F4,F4,F2
      SD   F4,-8(R1)
      LD   F6,-16(R1)
      ADD  F6,F6,F2
      SD   F6,-16(R1)
      ...
      DADDUI R1,R1,-64
      BNEZ R1,loop
```

→ Hasta F16

3/ DAXPY loop

```
loop: LD F2, 0(R1)        ALU FP      ALU FP      3
      MULD F4, F2, F0     ALU FP      SD          2
      LD F6, 0(R2)        LD          ALU FP      1
      ADDD F6, F4, F6     LD          SD          0
      SD F6, 0(R2)
      DADDUI R1, R1, -8
      DADDUI R2, R1, -8
      SLTI R3, R1, end
      BEQZ R3, loop
```

Fixed F0

a) Unroll 4 times

```
LD F2, 0(R1)     LD F8, 8(R1)     LD F14, 16(R1)    LD F20, 24(R1)    DADDUI R2, R1, -32
MULD F4, F2, F0  MULD F10, F8, F0 MULD F16, F14, F0 MULD F22, F20, F0 DADDUI R2, R2, -32
LD F6, 0(R2)     LD F12, 8(R2)    LD F18, 16(R2)    LD F24, 24(R2     SLTI R3, R1, end
DDD F6, F4, F6   ADD F12, F10, F12 ADD F18, F16, F18 ADD F24, F22, F24 BEQZ R3, loop
SD F6, 0(R2)     SD F12, 8(R2)    SD F18, 16(R2)    SD F24, 24(R2)
```

cycle	Mem 1	Mem 2	FP1	FP2	Int / Branch
1	LD F2, 0(R1)	LD F8, 8(R1)			
	LD F14, 16(R1)	LD F20, 24(R1)			
	LD F6, 0(R2)	LD F12, 8(R2)	MULD F4, F2, F0	MULD F10, F8, F0	DADDUI R1, R1, -32
	LD F18, 16(R2)	LD F24, 24(R2)	MULD F16, F14, F0	MULD F22, F20, F0	DADDUI R2, R2, -32
			ADD F6, F4, F6	ADD F12, F10, F12	SLT R3, R1, end
			ADD F18, F16, F18	ADD F24, F22, F24	
	SD F6, 32(R2)	SD F12, 24(R2)			Beqz R3 loop
	SD F18, 16(R2)	SD F24, 8(R2)			

Branch Predictors, BHR, BHTs...

1-Solo global

	T/NT	Predictor	HIT	BHR	Table1	Table2	Table3	Table4	Predict	HIT
B2	NT	00 / 00	Y	00	00 / 00	00 / 00	00 / 00	00 / 00	NT	Y
B2	T	00 / 01	N	01	00 / 01	00 / 00	00 / 00	00 / 00	NT	N
B1	NT	00 / 01	Y	10	00 / 01	00 / 00	00 / 00	00 / 00	NT	Y

- Las Tables se llenan con lo que pasó en la línea nueva PERO modificando la BHT que indique el anterior
- El predictor es para el primer hit/miss
  - 00 → SNT
  - 01 → WNT
  - 10 → WT
  - 11 → ST

(June 2014)

B1	T	NT(m)	01 / 10	11	1	01 / 11	10 / 10	← anterior
B2	NT	T(m)	01 / 10	10	0	01 / 11	10 / 01	(*) Actual

- Para saber si es predictor T/NT, miramos BHR anterior, y en la línea anterior buscamos la branch actual
- Nos fijamos al selector anterior a la línea actual pa saber si toca global o local
- El BHR va fluyendo con entradas de ceros y unos

LAB ASIGNMENT 4
DYNAMIC CODE SCHEDULING

Objectives

- Understand the behavior of the *Scoreboard* and *Tomasulo* schemes.
- Study the differences between both approaches.

Procedure

Execute the following MIPS program in the *Simula3MS* simulator (version 4.12!), using the Tomasulo's Algorithm:

```
# data segment
        .data
array:  .float 3.0, 4.0, 9.0

# text segment
        .text
        .globl main
main:
        la $t0, array
        lwc1 $f4,0($t0)
        lwc1 $f2,4($t0)
        lwc1 $f0,8($t0)
        mul.s $f6,$f0,$f0
        div.s $f0,$f6,$f2
        add.s $f6,$f4,$f4
        addi $v0, $0, 10      # end
        syscall
```

Information to collect

During the lab session, you should answer these questions:

1. Provide the instruction status table when the `add.s` instruction is in the *write result* step.
2. Give the same table for the Scoreboard method.
3. What are the hazards solved in this example by Tomasulo, but not solved by Scoreboard?

Note: for the simulation runs, leave the default latencies for the functional units.

1. Status Table (Tomasulo)

la → 3 primeros ciclos

	Emisión	Ejecución	Escritura
lwc1 f4,0($t0)	4	5-6	7
lwc1 f2,4($t0)	8	9-10	11
lwc1 f0,8($t0)	12	13-14	15
mul.s f6,f0,f0	13	16-19	20
div.s f0,f6,f2	14	21-27	28
add.s f6,f4,f4	15	16-17	18

WAW

6 structural hazards

2. Scoreboard

la → 3 primeros ciclos

	Emisión	Lectura	Ejecución	Escritura
lwc1 f4,0($t0)	4	5	6	7
lwc1 f2,4($t0)	8	9	10	11
lwc1 f0,8($t0)	12	13	14	15
mul.s f6,f0,f0	13	16	17-20	21
div.s f0,f6,f2	16	22	23-29	30
add.s f6,f4,f4	22	20	24-25	26

MUST WAIT!!

Debe esperar a que escriba en f0

Esperando f6 de muls, ya en proceso

WAR

3. Tomasulo solves WAR WAW

mul.s f6,f0,f0
div.s f0,f6,f2

→ mul.s f6 (solved by TOMASULO
add.s f6

Not solved by scoreboard

Unit 3

Tomasulo

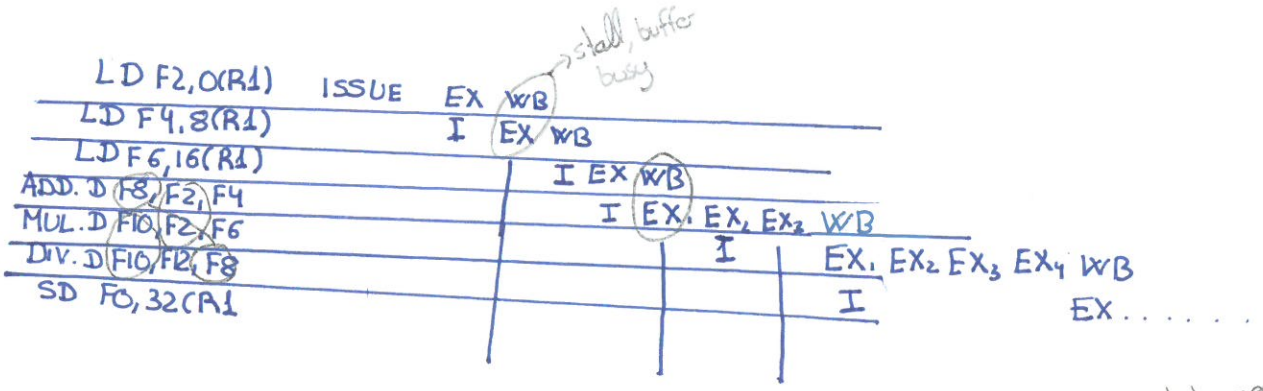
LD/SD: 1 cycle
FPMult: 4 cycles
FPAdd: 3 "
FPDiv: 6 "

Reserv Station per FP unit: 2 entries
LD Buffer: 2 entries
SD Buffer: 1 entry

EX stage can't be in the same cycle in which the result is written into the CBD

Instruc	Emisión	Ejecuc.	Escritur.
LD F2,0(R1)	1	2	3
LD F4,8(R1)	2	3	4
LD F6,16(R1)	4	5	6
ADD.D F8,F2,F4	5	6-8	9
MUL.D F10,F2,F6	7	9-12	13
DIV.D F10,F12,F8	9	13-18	19
SD F0,32(R1)	20	21	22

LD BUFF √ + 1 lat. + write



Unit 4

```
DADDIU R1,R0,#2
...
1: DADDIU R1,R0,#4
...
2: DADDIU R2,R2,#-1
   BNEZ R2,L2        → Run 8 times
...
   DADDIU R1,R1,#-1
   BNEZ R1,L1        → Run 2 times
```

a) 1 bit predict not taken

(*) Takes what happens before

1 = Taken
0 = Not Taken

BRANCH 2

Times runned	Prediction	Real	State	
1	NOT TAKEN	TAKEN	1	#3
2	TAKEN	TAKEN	1	#2
3	TAKEN	TAKEN (*)	1	#1
4	TAKEN	NOT TAKEN	0	#0 → Cumple condition, no toma
5	NOT TAKEN	TAKEN	1	#3
6	SAME		1	#2
7	SAME		1	#1
8			0	#0 → NO TOMA

BRANCH 1

1	NOT TAKEN	TAKEN	1 #1
2	TAKEN	NOT TAKEN	0 #0

CONCLUSION: 1 BIT predict is not enough for branch 1

b) 4 bit predict, saturizing casters initialized at 7 → 0111

BRANCH 2

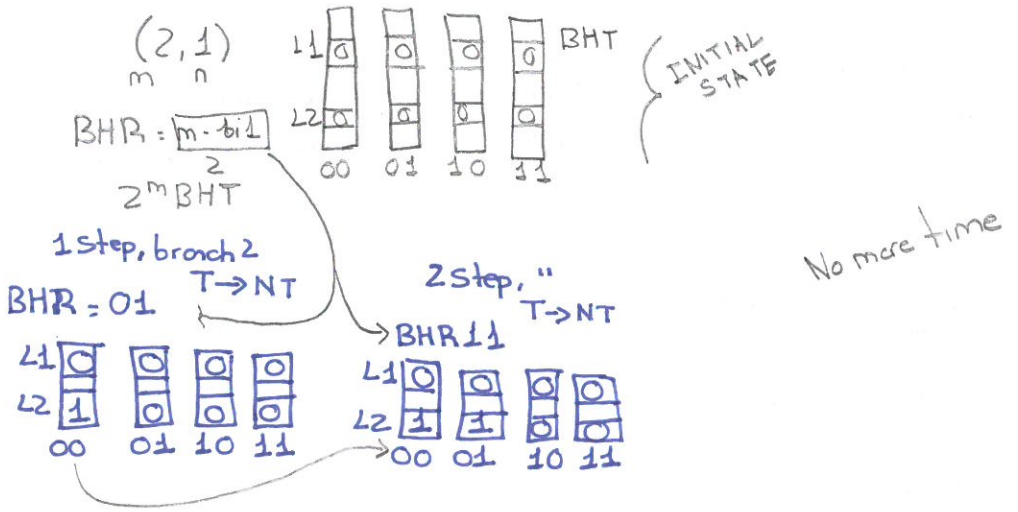
Times runned	Prediction	Real	STATE
1	NOT TAKEN	TAKEN	0111 +1 → 1000
2	TAKEN	TAKEN	1000 +1 → 1001
3	TAKEN	TAKEN	1001 +1 → 1010
4	TAKEN	NOT TAKEN	1010 -1 → 1001
5	TAKEN	TAKEN	1001 +1 → 1010
6	TAKEN	TAKEN	1010 +1 → 1011
7	TAKEN	TAKEN	1011 +1 → 1100
8	TAKEN	NOT TAKEN	1100 -1 → 1101

TAKES the prediction from previous action, so the predict is correct

BRANCH 1

1	NOT TAKEN	TAKEN	1000
2	TAKEN	NOT TAKEN	1001

c) (2,1) predictor, when exec. starts, 2 last branch = NT, initial is NT



1.1 Computer Architecture

Comp. designer task?

Maximize performance, energy efficiency...

Definition: Design to meet specific requirements, maximizing performance related to cost, power...
Includes ISA, microarchitecture and hardware

Implementation has two components
- Microarchitecture: high-level design, memory system...
- Hardware: Detailed logic design

→ Orgnanization: designed to meet the requirements of particular architecture. Architecture may rest, but organization evolves

Will the computer provide a multiply instruction? ← Architectural VS Organizational → Will the multiply instruc be implemented by a special unit or by a mechanism?

1.2 Performance

Latency (response time): time between start and completition of an event
Bandwith (throughput): total work in a given time

Performance = 1/Exec. time

"X is n times faster than Y"
$\frac{Perf_X}{Perf_Y} = n$

"X is f% faster than Y"
$f = (n-1) \times 100$

CPU exec. time = CPU clock cyc. · clock cycle time
↳ instr. count · CPI · clock cycle time
↳ $\frac{\text{CPU clock cycles}}{\text{clock freq}}$

clock freq = 1/clock cycle time

	Instr. count	CPI	clock rate
Program	X		
Compiler	X		
ISA	X	X	
Organization		X	X
Technology			X

CPI: cycles per instruc. Average num of clock cycles for each instruc.
CPU clock cycles = $\sum_{i=1}^{n} (CPI_i \cdot IC_i)$

1) Comp 1: 1ns, CPI of 2 Faster? $\frac{Exec.time_2}{Exec.time_1} = \frac{Instr.num \cdot CPI_2 \cdot clock_2}{Instr.num \cdot CPI_1 \cdot clock_1} = \frac{1.2 \cdot 2}{2 \cdot 1} = 1'2$
Comp 2: 2ns, CPI of 1.2
1 is 1'2 times faster/A is 20% faster

Speed-up: $\frac{\text{Perform. after}}{\text{Perform. before}}$

Amdahl's Law: Performance is improved when using some faster exec. mode is limited by the time fraction the faster mode can be used

Exect. time after enhancement = exect. time$_{\text{frac not enhan}}$ + exect. time$_{\text{fraction enhanced}}$

2) Program runs in 100 seconds, using 80 sec. in multiplications. How much should be improve for 5 times faster?

Speed-up: $\frac{\text{Exec time no enhacement}}{\text{Exec time with enhacement}} = 5$; New exec time: Exec. time$_{\text{not enhanced}}$ + Exec. time$_{\text{with enhanced}}$:
$= 20 + \frac{80}{\text{factor of enhan for mult}} \Rightarrow 5 = \frac{100}{20 + \frac{80}{\text{factor}}}$; IMPOSSIBLE SPEED UP OF 5

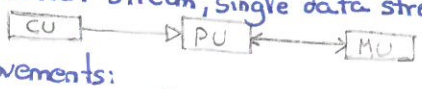
MIPS: $\frac{\text{instruc. count}}{\text{CPU exec. time} \cdot 10^6} = \frac{\text{clock freq}}{\text{CPI} \cdot 10^6}$

MFLOPS: $\frac{\text{num. floating-point opera.}}{\text{CPU exec. time} \cdot 10^6}$

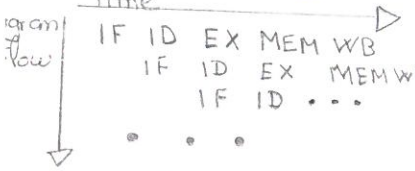
Benchmark: program used to assess the performance of the whole computing system or any part
↳ Types: Real programs, kernels, toy programs
They increase predictability, set by computer vendors and periodically updated
• SPEC • TPC

1.3 Classes of computers

Internet of Things (IoT), Mobile, Desktop, Servers, Clusters (software)
Trends in AC:
- Instruction-Level Parallelism (ILP): Exploits the implicit parallelism in a code/loop
- Thread-level Parallelism (TLP): Multiple instruction data flows that are parallel
- Data-level Parallelism (DLP): Operates on many data items at the same time
- CPU-level Parallelism
- Request-level Parallelism

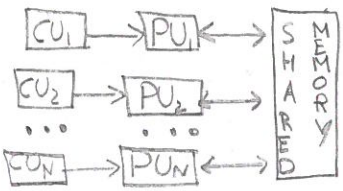
Flynn's Taxonomy: very popular but outdated. Criteria: number of instruc & data streams
- SISD, single instruction stream, single data stream
  Uniprocessor 

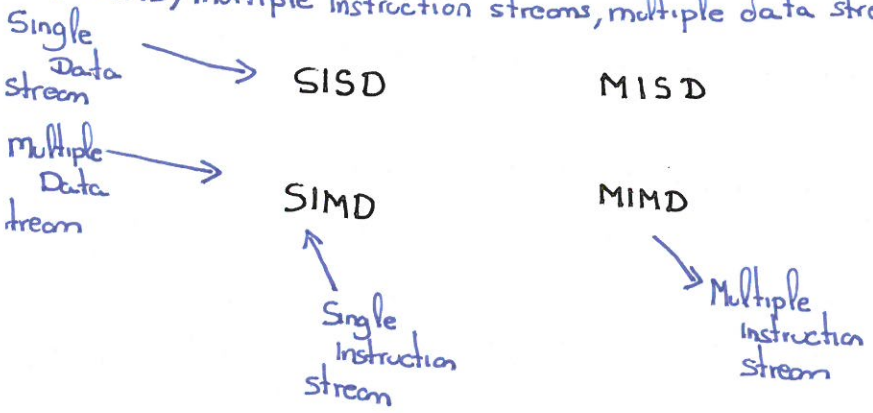
  Performance improvements:
  - Pipelining: Consists in overlapping the execution of several instruc. so they use different resources



Impact in throughput
Limited by the slowest step
Not visible for the programmer

  - Superscalar principles
    Issue n instructions in each clock cycle CPI = 1/n
    2-way superscalar, n = 2.
    IF ID EX MEM WB
    IF ID EX MEM WB CPI = 0'5
    IF ID EX MEM WB
    ...
  - VLIW (Very long instruction word): pack multiple operations in one instruction
- SIMD, single instruction stream, multiple data streams
  Vector architectures, multimedia extensions, graphic processor units
- MISD, multiple instruction streams, single data stream
  No commercial application
- MIMD, multiple instruction streams, multiple data streams





Increasing the ILP

15% branch, 6-7 instruc per basic block → Conclusion: Parallelism with basic block is limited
↳ Solution: Exploit ILP across multiple basic blocks
↳ How?: Overlap basic blocks and independent loop iterations

2.3 Study of dependences

- Dependences:
  - Property of programs
  - Indicate possible hazards
  - Impose an upper bound for ILP
- Types of dependences
  - Data
  - Name → Antidependences
  - Control → Output dependence

Data dependences

Instructions i and j (i precedes j), j is data dependent of i if:
- i produces a result used by j
- j is dependent of k, and k of i

```
Loop: LD F0,0(R1)
      ADD F4,F0,F2
      SD F4,0(R1)
      ADDI R1,R1,#-8
      BNE R1,R2,Loop
```

Name dependences

When two instruc use the same register or mem. location. Exists dependences if:
- j writes a register or mem location that i reads (Antidependence)
- i and j write the same register or mem location (Output dependence)

```
LD F0,0(R1)
ADD F4,F0,F2
SD F4,0(R1)
LD F0,-8(R1)
ADD F4,F0,F2
...
```

Control dependences

Ordering of an instruction i respect to a branch instruction
Every instruct except first block is control dependences

```
Loop: LD F2,0(R1)
      ...
      BEQ R1,exit
      LD F6,0(R1)
      ADD F8,F6,F2
      SD F8,0(R1)
```

↓ILP → Dependent instruc → possible hazards → ↑CPI

- Removing dependences

Name dependences
- Register renaming
  - Writing on registers or mem locations different than the used
  - Can be done statically (by the compiler) or dynamically (by the hardware)

Control dependences
- Removing intermediate branches
- Pending

Data dependences
- Scheduling → maintain the dependence, but avoid the hazard
- Transform the code

2.1 Pipelining basics

Pipelining: Overlap the execution of several instructions to use different resources

Two instructions can't use the same resource at the same time
- Dif cache memories
- Register file in ID and WB

Ideal speed-up = pipeline deth (number of steps)
- Ideally: 5 stage pipeline is 5 times faster
- Actually: steps unbalanced + overhead

{ but... sometimes next instruction cannot be executed = hazard

Types of hazards:
- Structural hazards: Hardware cannot support a combination of instructions
- Data hazards: An instruction depends on the result of a prior one
- Control hazards: Delay

$CPI_{pipelined} = 1 + \text{pipeline stall cycles per instruc.}$ $\quad$ Speed-up: $\frac{CPI_{unpip}}{1 + \text{pipeline stall...}}$

✯ Structural hazards

Ex: two instruc. try to write in a register with only one port

↳ Solution: stall one cycle

✯ Data hazards

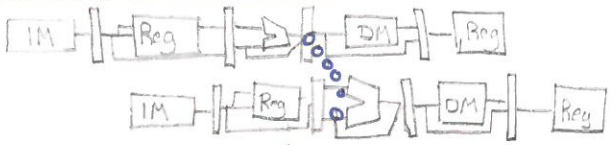
Changes the order of read/write accesses to operands so that the order differs from the normal execution

Types:
- RAW (read after write): Read before it has been written
- WAW (write after write): Only in pipelines that write in more than one stage
- WAR (write after read): Not happening in MIPS

Forwarding: Passing the ALU result from who has it to who needs it

Not all data hazards can be solve by forwarding. Some load instructions produce that problem

↳ We need a hardware to detect it and produce a stall



✯ Control hazards

Stopping the pipe: after detecting that there's a branch (after ID)

Branch	IF	ID	EX	MEM	WB
+1		IF	stall	stall	IF	ID	EX
+2					IF	ID	EX

FP instructions: Multiple functional units. Some instruc. faster

	Latency	Initiation interval
Integer ALU	0	1
Data mem	1	1
FP add	3	1
FP multiply	6	1
FP divide	24	25

- Latency: Cycles between an instruc. that produces a result and an instruc. that uses it
- Initiation interval: nº of cycles that must elapse between issuing two operations of a type

2.2 Reducing stalls

Techniques:
- Pipelining: Overlapping instructions → The potential overlap is called instruction-level parallelism (ILP)
  - Goal: Reduce the CPI
  - How? Compilation-time analysis and Run-time analysis
- Basic (static) techniques:
  - Segmentation → Structural
  - Forwarding → RAW data
  - Delayed branch → Control
- Advanced (static and dynamic)
  - WAR, WAW, RAW
  - Loop unrolling → Control hazards
  - Branch prediction → Control hazards
  - Multiple instruc. per cycle → Ideal CPI

ILP related to CPI

ILP: Overlapping in a sequence of instructions, depends on the degree of dependency between instruc.

↑ILP → few issues → few stalls → ↓CPI

Independ instructions if they can be executed simultaneously

31) New floating point? 4 times faster than old one. Speedup of 5. Benchmark takes 100s with old one
↳time to archive that speedup?

Exec time (New) = Exec time not enhanced + Exec time enhanced

$$\text{Speedup} = \frac{\text{Exec time NE}}{\text{Exec time E}} \; ; \; 5 = \frac{100}{100 - t + t/4} \; ; \; 5 \cdot (100 - t + t/4) = 100 \; ; \; 500 - 5t + 5t/4 =$$

$$2000 - 20t + 5t = 400$$

$$-15t = -1600$$

$$t = 106'66\ldots s$$

c) H's not possible

Type	CPI in M1	CPI in M2
A	1	2
B	2	2
C	3	3
D	4	4

Evenly distributed among four types of instruc.
What freq for M1 to equal performance of M2 with 1000MHz

$$CPI_{1\,AVG} = \frac{1+2+3+4}{4} = \frac{10}{4} = 2'5$$

$$CPI_{2\,AVG} = \frac{2+2+3+4}{4} = \frac{11}{4} = 2'75$$

$$\frac{\text{Freq } M_1}{2'5} = \frac{1000}{2'75} \; ; \; \frac{1000 \cdot 2'5}{2'75} = \text{Freq M1} = 909'09\,09\ldots \text{ MHz}$$

a) 900

Reduces CPI for FP to 35% of initial value. Improved clock cycle by 20%. Before the improvement FP was 4 times the CPI for integer operations. Speed-up for a program of 6M int. instruc and 2M FP?

		R1	R2
FP add	16%	6	20
FP mult	10%	8	32
FP div	8%	10	66
Non-FP	66%	3	3

a) MIPS for R1 and R2, clock freq of 400 MHz

$R_1 = (6 \cdot 0'16) + (8 \cdot 0'1) + (10 \cdot 0'08) + (3 \cdot 0'66) = 4'54$

$= 0'96 + 0'8 + 0'8 + 1'98 = 4'54$

$$\text{MIPS} = \frac{\text{freq}}{CPI \cdot 10^6} = \frac{400 \cdot 10^6}{4'54 \cdot 10^6} = 88'10 \text{ MIPS}$$

$R_2 = (20 \cdot 0'16) + (32 \cdot 0'1) + (66 \cdot 0'8) + (3 \cdot 0'66) =$

$= 3'2 + 3'2 + 5'28 + 1'98 = 13'66$

$$\text{MIPS} = \frac{\text{freq}}{CPI \cdot 10^6} = \frac{400 \cdot 10^6}{13'66 \cdot 10^6} = 29'28 \text{ MIPS}$$

b) CPU program for R1 and R2. 12000 instruc

$$\text{freq} = \frac{1}{\text{tiempo}} \; ; \; \text{tiempo} = 1/\text{freq} = 1/400 \cdot 10^6 = \ldots$$

R1 CPU exec = instruc count · CPI · time =

$= 12000 \cdot 4'54 \cdot \ldots = 0'00013625$

↳iPhone: 2'5e-9

R2 CPU exec = instruc count · CPI · time =

$= 12000 \cdot 13'66 \cdot \ldots = 0'00040985$

c) What mixture of instruc R1 and R2 will be equally fast?

When it does not have the FP since they would have the same CPI $0'66 \cdot 3 = 0'66 \cdot 3$ (R_1 R_2)

Unit 2.

9) LB R2, 0(R1) 1 → loop:
DADD R3, R2, R0 2
DADDUI R2, R2, S 3
SB R2, 0(R1) 4
DADDUI R1, R1, 1 S
BNE2 R3, loop 6

Data depend:
R2 → 1 and 2
3 and 4
1 and 3
R3 → 2 and 6
R1 → none

Output depend:
R2 → 1 and 3
R1 → None
R3 → None

Antidependences:
R1 → 1 and S
4 and S
R2 → 2 and 3

P2) loop: LS F0, 0(R1)
LS F2, 80(R1)
MULS F4, F0, F2
Read → SS F4, 0(R1)
DADDUI R1, R1, #4
BNE R1, R3, loop

P1) loop: LD R4, 0(R1)
LD R5, 0(R2)
SLT R4, R4, R5
SD R4, 0(R1)
DADDUI R1, R1, #8
DADDUI R2, R2, #8
BNE R1, R3, loop

Unit 3

Ex 13/Jan 2014

Vectors x and y containing n 64 bits integer elem. $x(i) = \begin{cases} 1 & \text{if } x(i) < y(i) \\ 0 & \text{if } x(i) \geq y(i) \end{cases}$

a) List all dependences (some iterat)
b) Apply software pipelining and show mod. body
c) Add start-up and finish-up blocks

loop:
LD R4, 0(R1) A
LD R5, 0(R2) A
SLT R4, R4, R5 B
SD R4, 0(R1) C
DADDUI R1, R1, 8
DADDUI R2, R2, 8
BNE R1, R3, loop

R1, R2 → starting addresses of x and y
R3 → next element after last x elem.

SLT R6, R4, R5
SD R6, 0(R1)

1st	2nd	3rd	
A			start
B	A		
C	B	A	loop
	C	B	finish
		C	

start-up:
LD R7, 0(R1)
LD R8, 0(R2)
SLT R6, R7, R8
LD R4, 8(R1)
LD R5, 8(R2)

loop:
SD
SLT
LD
LD
DADDUI
DADDUI
BNE

finish up:
SD R6, 0(R1)
SLT R6, R4, R5
SD R6, 8(R1)

3.1 Static scheduling

Compiler modifies the code to increase ILP, reducing dependencies

- loop unrolling → Separate dependent instruc
- Software pipelining → Reorder dependent instruc
- Trace scheduling → Reorder dependent instruc

✧ Loop Unrolling

Static/software scheduling technique, reduce pipeline stalls due control hazards

Consists in repeating the loop code several times (reduces iterations), increase the basic block to move away dependences. In first compiling stages

```
LD F0,0(R1)
ADD.D F4,F0,F2
SD F4,0(R1)
LD F0,-8(R1)
ADD.D F4,F0,F2
SD F4,-8(R1)
LD F0,-16(R1)
ADD.D F4,F0,F2
SD F4,-16(R1)
LD F0,-24(R1)
ADD.D F4,F0,F2
SD F4,-24(R1)
DADDUI R1,R1,#-32
BNE R1,R2,loop
```

stalls ←

Name / Rename ⇒

```
LD F0,0(R1)
ADD.D F4,F0,F2
SD F4,0(R1)
LD F6,-8(R1)
ADD.D F8,F6,F2
SD F8,-8(R1)
LD F10,-16(R1)
ADD.D F12,F10,F2
SD F12,-16(R1)
LD F14,-24(R1)
ADD.D F16,F14,F2
SD F16,-24(R1)
DADDUI R1,R2,#-32
BNE R1,R2,loop
```

7 cycles per array element

```
LD F0,0(R1)
LD F6,-8(R1)
LD F10,-16(R1)
LD F14,-24(R1)
ADD.D F4,F0,F2
ADD.D F8,F6,F2
ADD.D F12,F10,F2
ADD.D F16,F14,F2
SD F4,0(R1)
SD F8,-8(R1)
DADDUI R1,R1,#32 → -32
SD F12,16(R1) → -16+32
BNE R1,R2,loop
SD F16,8(R1) → -24+32
```

3.5 cycles per array element

Limitations:

- Decrease in the amount of overhead amortized with each unroll
- Growth in code size (+ cache miss rate)
- Register pressure, may not be possible to allocate all the values to registers

✧ Software pipelining

Scheduling performed by the compiler. Reduce stalls due to data hazards and CPI

Consists in transform a loop with dependent instruc and independent iterations into independent instruc and dependent iter

Iteration 1	Iteration 2	Iteration 3
LD F0,0(R1)		
ADD.D F4,F0,F2	LD F0,-8(R1)	
SD F4,0(R1)	ADD.D F4,F0,F2	LD F0,-16(R1) →
	SD F4,-8(R1)	ADD.D F4,F0,F2
		SD F4,-16(R1)

Each iteration is composed by those 3 instructions, in this case it can be able to

Práctica 5

Cargamos el código

```
ADDI R2, R0, 10 //Dir x
ADDI R3, R0, 30 //Dir y
ADDI R4, R0, 5 //Dir k
LF   F0, (R4)
ADDI R5, R2, 10
Buc: LF   F1 (R2)
LF   F2 (R3)
MultF F1,F1,F2
ADDF F1, F1, F0
SF F1,(R3)
ADDI R2,R2,1
ADDI R3,R3,1
BNE R2,R5, buc
SW R5, 0(R0)
SW R4, 1(R0)
```

{ 1st loop

Load mem

#GPR

#FPR

#MEM

[5] 2

[10] 1 2 3 4 5 6 7 8 9 10

[31] 1

[33] 1

[35] 1

[37] 1

[39] 1

a) Behavior of branch predict. used by the simulator. State in every branch instruc

- Uses dynamic predictor of two bits. Start at 00 then 01 11 10 00.
  ↳ 00 means (predicts not taken, but is taken (10 times)
- First loop fails so 00 ← 1; 01
- Second loop is not taken, but should have been taken, so 01 ← 1; 11
- Third, four ... loops are not taken, all Ok until last, that is not taken so 11 ← 0; 10

00
01
11
⋮
10

b) Predictor hit rate

Fails: 1st, 2nd, last $Hit_R = \frac{7 \text{ hit}}{11 \text{ instruc}} = 63'60$

Total: 1 + 10 = 11

c) Prefetch, decoder and reorder in cycles 26 and 27

26 Prefetch

Decoder

— ? ?

?

Entry	Instruc	Dest.	Value	Address	State
11	11	3	31	-1	write
12	12	-1	1	-1	write
13	13	-1	20	0	write
14	14	-1	5	1	write

No commit bcs fails predict

Fail of first predict. Prefetch and decoder empty bcs when fails it doesn't keep exec. next instruc of next loop until firsts end. Empty bcs only left last 4 writing stages of last iterat. (Nothing in deco or prefetch)

27 Prefetch Decoder Entry Instruc Dest. Value Address State

5
6
7
↝ 8

Vacio de

System starting next iteration for second loop. It will end as first loop in cycle 26. Some in last

First Part, Superescalar

a) CPI for degree of 2

$$CPI = \frac{141 \text{ cycles}}{6+(8\cdot 16)} = \frac{141}{134} = 1'0597$$

	Num	Lat.
Suma	1	2
Mult	1	3
Suma F	1	3
Mult F	1	4
Memoria	1	2
Fallo caché:		9
Salto.	1	2

b) CPI for degrees of 4, 6, 8, 16

$$CPI_4 = \frac{137}{6+(8\cdot 16)} = \frac{138}{134} = 1'02985$$

$$CPI_6 = \frac{134}{6+(8\cdot 16)} = \frac{134}{134} = 1 \qquad CPI_8 = CPI_{16} = CPI_6$$

c) HW aspect that is limiting performance? Improvement to best speedup?

Seen before, as higher degree, a better CPI (closer to 0)

Degree 2: 0'5; Degree 4: 0'25... ...

↳ Cannot happen due to program dependencies

Best option is degree 6, since we obtain the ideal CPI of 1.

d)

GWR

6'282 → R3

25'12

56'52

100

157 ✓

226

307

402

508

0

Second part, VLIW

↳ Debe de ser 13

3 0
4 2
5 1
6
7
8
10
12
11 9 13

152

```
LF F1, 10(R0)      en dir 10 valor π
ADDF F1, F1, F1    2·π
LW R1, 11(R0)      dir 11, num elem de v
ADDI R2, R0, #100  dir. vector v
ADDI R12, R0, #200 dir dest. vector w
ADDI R3, R0, #0    Contador elem. a 0
Bucle:
LF F6, (R2)        lectura elem. v
MULTF F6, F6, F6   elem al cuadrado
MULTF F6, F6, F1   2·π. elem al cuadrado
SF F6, (R12)
ADDI R2, R2, #1
ADDI R12, R12, #1
ADDI R3, R3, #1
BNE R3, R1, Bucle
```

d)

```
LF F6, 0(R2)
LF F1, 1(R2)
LF F4, 2(R2)
LF F5, 3(R2)
MULTF F6, F6, F6
MULTF F1, F1, F1
MULTF F4, F4, F4
MULTF F5, F5, F5
ADDI R3, R3, #4
MULTF F6, F6, F8
MULTF F1, F1, F2
MULTF F4, F4, F3
MUTF F5, F5, F1
SF F6, -4(R12)
SF F1, -3(R12)
SF F4, -2(R12)
SF F5, -1(R12)
ADDI R2, R2, #4
ADDI R12, R12, #4
BNE R3, R1, bucle
```

$$CPI = \frac{98 \text{ cycles}}{6 + (4 \cdot 20)} = 1.1$$

(4 ↙ Loops; 20 ↘ instruc)

Initially: 16 loops
After 4 unrolling: 4 loops } 16/4 = 4

Instruc BF loop mtra: 6
Instruc AF loop unroll: 6

Instruc per loop with NO LU: 8
Instruc per loop with LU: 20

Total before LU: 134
Total after LU: 86

	Instruction	Issue	Execute	Write Result	Commit
0	ADDI R2 R0 #10	3	4	5	6 | 4
1	ADDI R3 R0 #30	3	5	6	7 | 4
2	ADDI R4 R0 #5	5	6	7	8 | 5
3	LF F0 (R4)	5	7		| 8
4	ADDI R5 R2 #10	6	7	8	9 | 6
5	LF F1 (R2)	6	8		| 9
6	LF F2 (R3)	7	8		| 10
7	MULTF F1 F1 F2	7			| 10
8	ADDF F1 F1 F0	8			| 11
9	SF F1 (R3)	8			| 11
10	ADDI R2 R2 #1	9			| 9
11	ADDI R3 R3 #1	9			| 9
	BNE R2 R5				| 10
	SW R5 0(R0)				| 13
	SW R4 1(R0)				| 14

F0 ↑ R4=5

0	3	4	5	6
1	3	5	6	7
2	5	6	7	8
3	5	7	13	14
4	6	7	8	14
5	6	7	13	15
6	7	8	14 (F2)	15
7	7	14	20 (F1)	21
8	8	20	24	25 (F1 to store)
9	8	9	25	26
10	9	10	11	26
11	9	11	12	27
12	10	11	13	27
13	10	11	13	27
14	11	12	14	27

Todo ejecuta