2

CPU

Objectives

The exercises in this chapter will cover the topics of the lectures of unit 2: performance improvement, multitasking OSs and virtualization support.

Exercise 10.

A CPU implementing the AMD-64 architecture will be designed. First, a sequential version of this CPU is designed executing 200 million instructions per second when running at its highest clock frequency. The execution of every instruction is divided into ten stages working sequentially, all of them with the same duration: one clock cycle.

❑ **10.1** What is the required clock frequency of the CPU to achieve this number of instructions per second?

```
2 GHz
```

❑ **10.2** What is the minimum time to execute any instruction in this CPU? Give your answer in nanoseconds.

```
5 ns
```

❑ **10.3** How many instructions can this CPU execute per clock cycle (IPC)? Example of answer: 1.2 instructions per clock cycle.

```
0.1 instructions/cycle
```

Once the initial version of the CPU has been designed, it is modified to use pipelining. The control unit is modified for the execution of the ten stages of the pipeline to happen concurrently. You must ignore the delay introduced by pipeline registers.

❑ **10.4** What is the minimum time to execute any instruction? How many instructions, at most, can be executed per second?

```
5 ns
2000 MIPS
```

❑ **10.5** What modification could be made in the internal organization of the CPU in order to achieve 3000 MIPS?

```
For example, the number of stages in the pipeline may be in-
creased, also increasing the clock frequency. There are other
possibilities such as converting the CPU in superscalar.
```

Exercise 11.

Let A and B be two CPU manufacturers. A has designed the A10 CPU that executes 3000 MIPS at 1.5 GHz. B will release the B20-PRO CPU to compete with A. B20-PRO runs at 2 GHz, implements the same instruction set as A10 and is an enhanced version of the sequental B20 CPU, which runs at 500 MHz and requires 4 clock cycles to execute any instruction.

❑ **11.1** What is the average time to execute an instruction in the B20 CPU?

```
4 cycles × 2 ns/cycle = 8 ns
```

❑ **11.2** The B20-PRO CPU is obtained by dividing the execution of instructions in N balanced stages of one cycle (of 2 GHz) running concurrently. What is the value of N?

```
N = 8 ns / 0.5 ns = 16
```

❑ **11.3** Which CPU provides a better performance, A10 @ 1.5 GHz or B20-PRO @ 2 GHz? Why?

```
The B20-PRO CPU can execute one instruction per cycle since
it is a pipelined CPU. Its clock frequency is 2 Ghz; thus, it
achieves 2000 MIPS, less than 3000 MIPS achieved by A10.
Therefore, the A10 CPU provides a better performance even
though its clock frequency is lower than the B20-PRO clock fre-
quency.
```

Exercise 12.

A CPU can execute 200 million instructions per second. The execution of every instruction requires completing 15 stages sequentially. Each stage consumes one clock cycle. Giving these features of the CPU, answer the following questions.

❑ **12.1** What is the maximum frequency of this sequential CPU?

200 MIPS × 15 cycles/instruction = 3 GHz

❑ **12.2** How long does it take to execute a single instruction? Answer in nanoseconds.

1 s / 200 MIPS = 5 ns/instruction

❑ **12.3** How many instructions can be executed per clock cycle?

1 instruction / 15 clock cycles = 0.0667 instructions/cycle

A new CPU with the same ISA but with a clock frequency of 1.5 GHz is designed. This CPU uses a 10-stage pipeline, where the stages work in parallel. Each stage consumes one clock cycle.

❑ **12.4** What is the maximum number of instructions per second that this CPU can execute?

1500 MIPS

❑ **12.5** How long does it take to execute a single instruction in this new CPU? Answer in nanoseconds.

10 clock cycles $\times(1/(1.5 \times 10^9))$ ns/clock cycle = 6,67 ns

❑ **12.6** How many instructions per clock cycle can this CPU execute?

1 instruction per clock cycle

Exercise 13.

Given a sequential CPU implementing the MIPS64 ISA where each instruction execution is divided into five stages: 50 ns, 50 ns, 60 ns, 50 ns and 40 ns.

❑ **13.1** How long does it take to execute an instruction in this CPU?

$50 + 50 + 60 + 50 + 40 = 250$ ns

❑ **13.2** How many instructions per second can this CPU execute?

4 MIPS

The organization of the CPU is modified to turn it into a pipelined CPU. The above 5 stages are the pipeline stages.

❑ **13.3** How long does it take to execute an instruction in the pipelined version of the CPU?

$60 \times 5 = 300\,\text{ns}$

❑ **13.4** What is the frequency of the pipelined CPU?

$16.67\,\text{MHz}$

❑ **13.5** How many instructions per second will be able to execute the pipelined CPU in an ideal scenario?

$16.67\,\text{MIPS}$

❑ **13.6** What is the throughput speedup in the execution of instructions in the pipelined CPU using the ideal scenario (compared to the sequential version)?

$250/60 = 4.17$

Exercise 14.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) Running a program in a pipelined CPU generates control hazards when control flow instructions such as branches or function calls appear in the program.

B) In a pipelined CPU a structural hazard may appear when several instructions try to access the same piece of data in different stages at the same time.

C) Data hazards in pipelined CPUs refer to several instructions needing to wait for a functional unit to be available.

D) A pipelined CPU is able to finish the execution of one instruction per clock cycle at most.

E) In a pipelined CPU including single-cycle and multi-cycle execution units the time required to complete any instruction is always the same.

`A and D`

Exercise 15.

A MIPS64 CPU has the following latencies:

- `IF` stage: 0.25 ns.
- `ID` stage: 0.12 ns.
- `EX` stage for arithmetic and logic instructions: 0.15 ns.
- `MEM` stage: 0.25 ns.
- `WB` stage: 0.15 ns.
- Integer multiplication unit: 0.7 ns.

❑ **15.1** What is the maximum throughput in MIPS that the sequential version of this CPU can achieve? Round to avoid decimals.

1087 MIPS

❑ **15.2** What is the frequency of the pipelined CPU?

4 GHz

❑ **15.3** Taking into account these temporal parameters and ideal conditions, what is the speedup of the pipelined CPU compared to the sequential CPU? Round to two decimals.

3.68

The following code is executed on the pipelined CPU. The pipeline does not include any improvement, but it is possible to retire instructions out of order.

```
1         daddi r1, r0, 200
2 loop:
3         ld    r2, 0(r2)
4         dmul  r2, r3, r1
5         daddi r1, r1, -1
6         dmul  r2, r3, r2
7         bnez  r1, loop
8         dadd  r9, r2, r3
```

❑ **15.4** Fill the timeline with the execution of the above code until clock cycle #15.

Instruction \ Cycle	4	5	6	7	8	9	10	11	12	13	14	15
daddi r1, r0, 200	MEM	WB										
ld r2, 0(r2)	EX	MEM	WB									
dmul r2, r3, r1	ID	ID	EX	EX	EX	MEM	WB					
daddi r1, r1, -1	IF	IF	ID	EX	MEM	WB						
dmul r2, r3, r2			IF	ID	ID	ID	ID	EX	EX	EX	MEM	WB
bnez r1, loop				IF	IF	IF	IF	ID	EX	MEM	WB	
dadd r9, r2, r3								IF	IF	IF		
ld r2, 0(r2)											IF	ID
dmul r2, r3, r1												IF

Exercise 16.

The following code snippet is executed in a MIPS64 microarchitecture that implements, in addition to the general purpose execution unit, a specialized mult/div execution unit for integers, with a 4-clock cycle latency. Furthermore, out-of-order execution and retirement are allowed.

```
1 dmul r2, r5,  r6
2 xor  r1, r10, r3
3 dmul r4, r6,  r8
4 ld   r3, 100(r2)
```

❑ **16.1** Write down the pipeline stalls, identifying their duration.

Structural: 2 clock cycles
Data (RAW): 1 clock cycle

	1	2	3	4	5	6	7	8	9	10	11	12
dmul r2, r5, r6	IF	ID	EX	EX	EX	EX	MEM	WB				
xor r1, r10, r3		IF	ID	EX	MEM	WB						
dmul r4, r6, r8			IF	ID	X	X	EX	EX	EX	EX	MEM	WB
ld r3, 100(r2)				IF	X	X	ID	ID	EX	MEM	WB	

❑ **16.2** How many clock cycles does the above code snippet require? What is the CPI excluding the 4-clock cycle initial transient period?

12 clock cycles; CPI $= (12-4)/4 = 2$

❑ **16.3** If the system is required to support precise exceptions, how many clock cycles are required to run the program?

xor and ld instructions must be retired in order: 13 cycles.

	1	2	3	4	5	6	7	8	9	10	11	12	13
dmul r2, r5, r6	IF	ID	EX	EX	EX	EX	MEM	WB					
xor r1, r10, r3		IF	ID	EX				MEM	WB				
dmul r4, r6, r8			IF	ID	X	X	EX	EX	EX	EX	MEM	WB	
ld r3, 100(r2)				IF	X	X	ID	ID	EX			MEM	WB

❑ **16.4** If the mul/div execution unit is pipelined and precise exceptions are supported, what would happen with the structural stall? How many clock cycles does it require to run the program? What is the speedup compared to the previous question excluding the initial transient period?

The structural stall is avoided: 11 clock cycles.
Speedup is $(13-4)/(11-4) = 1.29$

	1	2	3	4	5	6	7	8	9	10	11
dmul r2, r5, r6	IF	ID	EX1	EX2	EX3	EX4	MEM	WB			
xor r1, r10, r3		IF	ID	EX				MEM	WB		
dmul r4, r6, r8			IF	ID	EX1	EX2	EX3	EX4	MEM	WB	
ld r3, 100(r2)				IF	ID	ID	ID	ID	EX	MEM	WB

Exercise 17.

The following code snippet stalls the pipeline when it is run.

```
1 dadd r9, r5, r7
2 xor  r4, r6, r7
3 ori  r9, r6, 1
4 ld   r3, 100(r4)
```

❑ **17.1** Which instruction does stall the pipeline? How many clock cycles does this stall last? What is the type of the hazard that generates the stall?

```
ld r3, 100(r4). Stall lasts for 1 clock cycle. Data hazard: RAW
on r4.
```

❑ **17.2** How many clock cycles does the execution of the previous snippet require?

```
9
```

❑ **17.3** If the compiler were able to re-order the instructions to remove the penalties introduced by data hazards, what would this order be?

```
xor r4, r6, r7
dadd r9, r5, r7
ori r9, r6, 1
ld r3, 100(r4)
```

❑ **17.4** How many clock cycles does the execution of the snippet require in this case?

```
The pipeline is not stalled, so it takes 8 clock cycles.
```

❑ **17.5** Would it be possible to move the dadd instruction just after ori to obtain the same result? Why?

```
No, it is not possible. There is a WAW data dependency between
dadd and ori and this movement would change the semantic of the
program.
```

Exercise 18.

A MIPS64-based CPU implements two forwarding paths: `Output EX`→`Input EX` and `Output MEM`→`Input EX`. In this CPU the following code snippet is run.

```
1 dadd r9, r5, r7
2 xor  r2, r6, r7
3 ld   r4, 100(r9)
4 dsub r2, r4, r9
```

❑ **18.1** Draw the pipeline evolution during the execution of the program.

```
                  1   2   3   4    5    6    7   8    9
dadd r9, r5, r7   IF  ID  EX  MEM  WB
xor r2, r6, r7        IF  ID  EX   MEM  WB
ld r4, 100(r9)            IF  ID   EX   MEM  WB
dsub r2, r4, r9               IF   ID   ID   EX  MEM  WB
```

❑ **18.2** How many clock cycles are saved thanks to forwarding?

```
The forwarding path Output MEM→Input EX saves 1 clock cycle,
since it provides the operand 1 clock cycle before WB writes
the result.
There are two forwardings of this type in the program, so 2
clock cycles are saved.
```

Exercise 19.

A MIPS64-based CPU works at 2 GHz. It implements early branches in the ID stage and all required forwarding paths. The following code snippet is run, loaded into memory from 0000 0000 0000 A200h.

```
1 ori r1, r0, 2      ; r1 = 2
2 startf:
3         beqz  r1, endf    ; jump to endf if r1 zero
4         daddi r1, r1, -1  ; r1 = r1-1
5         j startf          ; jump to startf
6 endf:
7         ld r5, 100(r3)
```

❑ **19.1** How long does it take to run the above code snippet?

```
19 clock cycles => 9.5 ns.
Nine instructions are run: ori, beqz, daddi, j, beqz, daddi, j,
beqz y ld. Thus, the number of clock cycles needed to run them
matches the initial transient period plus the number of in-
structions, plus the number of clock cycles stalled. The first
time beqz is executed, a data hazard on r1 appears. In addi-
tion, each branch and jump introduces one stall cycle. There-
fore, the number of clock cycles is: (4+9+1+5) = 19.
```

❑ **19.2** What forwarding paths are activated and what values are transferred?

```
Output EX→Input ID, value 2 (first execution of beqz)
This forwarding path is activated in the first execution of the
beqz instruction due to the data dependency on r1.
```

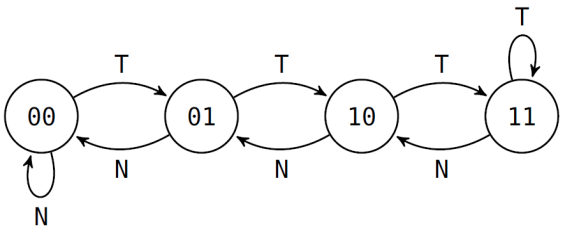

Figure 2.1: State machine of the 2-bit predictor

❑ **19.3** If an always-not-taken branch prediction is used, what is the new response time?

```
17 clock cycles => 8.5 ns.
When the prediction of a branch is a hit, the pipeline is not
stalled. When the prediction misses, the pipeline is stalled
one clock cycle. In the first two executions of beqz the pre-
diction is a hit; the last one is a miss. Thus, the number of
clock cycles results: (4+9+1+3) = 17.
```

❑ **19.4** If using a 2-bit branch predictor, what is the response time? What is the final state of the BTB table? The state machine used is represented in figure 2.1 and the prediction for branch instructions starts in state 01 (weak not taken).

```
16 clock cycles => 8 ns.
A204h A210h 01 (beqz)
A20Ch A204h 11 (j)



The first time that the jump is executed the pipeline is
stalled one clock cycle in the IF stage: the target of the jump
is computed in ID and it is stored in the BTB. The pipeline is
not stalled in subsequent executions of the jump since the tar-
get of the jump is obtained in IF from the BTB. On the other
hand, the first time that the branch is executed, the predic-
tion is a hit, since the initial state is 01 weak not taken
(predict not-taken) in the IF stage. However, the pipeline is
stalled one clock cycle due to the data hazard on r1. The tar-
get of the branch is computed in ID and stored in the BTB. The
second time that the branch is executed, a hit is also obtained
in the prediction (in IF due to the state 00) and the pipeline
is not stalled. The third time the prediction misses (not-taken
due to state 00) since the branch is taken, and the state is
updated to 01. Therefore, the pipeline is stalled one clock cy-
cle due to the jump, one clock cycle due to data hazard and one
clock cycle due to the branch. The number of clock cycles is:
(4+9+3) = 16.
```

Exercise 20.

A MIPS64 CPU with the following features is used: no forwarding paths, always-not-taken branch prediction, early branches at the ID stage, 3-cycle pipelined multiplication, out-of-order execution (when using different execution units) and out-of-order retirement. The program below is executed in this CPU:

```
 1         ori    r1, r0, 8 ; r1 = 8
 2         daddi  r2, r0, 2 ; r2 = 2
 3 loop:
 4         beqz   r1, dest
 5         dmul   r3, r2, r7
 6         xor    r6, r4, r5
 7         dadd   r6, r7, r6
 8         dsub   r1, r1, r2
 9         j      loop
10 dest:
11         dmul   r5, r4, r2
```

❑ **20.1** What is the execution time of the program in clock cycles in a sequential (non-pipelined) version of this CPU?

$10 + 4 \times (5 \times 6 + 2) + 5 + 7 = 150$ cycles

❑ **20.2** Indicate the first time that each type of the data dependecy (RAW, WAW and WAR) appears by identifying the type of dependency, the instructions involved and the register causing the dependency. Example of answer: `RAW: dsub and dadd, r5`.

```
WAW: xor and dadd, r6
RAW: ori and beqz, r1
WAR: beqz and dsub, r1
```

❑ **20.3** Identify the first three pipeline stalls while running the program. Answer with the type of stall (**(D)**ata, **(S)**tructural or **(C)**ontrol), the instruction suffering the stall and the duration of the stall in clock cycles. Example of answer: `(D) - dsub - 3 cycles`.

```
(D) - beqz - 1 cycle
(D) - dadd - 2 cycles
(C) - j / dmul - 1 cycle
```

The microarchitecture is improved by implementing all the forwarding paths.

❑ **20.4** What fowarding paths are activated while executing the program? Example of answer: `Output MEM dmul` → `Input EX dsub`.

```
Output EX ori → Input ID beqz
Output MEM daddi → Input EX dmul
Output EX xor → Input EX dadd
```

❑ **20.5** Fill the following table with the evolution of the pipeline from cycle #1 to cycle #13 assuming all forwarding paths available.

Instr. \ Etapa	1	2	3	4	5	6	7	8	9	10	11	12	13
ori r1, r0, 8	IF	ID	EX	MEM	WB								
daddi r2, r0, 2		IF	ID	EX	MEM	WB							
beqz r1, dest			IF	ID	EX	MEM	WB						
dmul r3, r2, r7				IF	ID	EX	EX	EX	MEM	WB			
xor r6, r4, r5					IF	ID	EX	MEM	WB				
dadd r6, r7, r6						IF	ID	EX	X	MEM	WB		
dsub r1, r1, r2							IF	ID	X	EX	MEM	WB	
j loop								IF	X	ID	EX	MEM	WB
dmul r5, r4, r2										IF			
beqz r1, dest											IF	ID	EX

Exercise 21.

The following code snippet is run in a MIPS64 CPU. This CPU features an integer multiplication and division unit that requires 4 clock cycles. The pipelined microarchitecture includes all necessary forwarding paths and does not implement precise exceptions.

```
1 daddi r3, r0, 5
2 ddiv  r2, r4, r5
3 xor   r3, r6, r8
4 dsub  r2, r9, r10
```

❑ **21.1** Draw the timeline of the execution.

```
                  1   2   3   4    5   6    7   8    9    10
daddi r3, r0, 5   IF  ID  EX  MEM  WB
ddiv r2, r4, r5       IF  ID  EX   EX  EX   EX  MEM  WB
xor r3, r6, r8            IF  ID   EX  MEM  WB
dsub r2, r9, r10              IF   ID  EX             MEM  WB
```

❑ **21.2** Now, assume that register renaming is used. For this purpose, 64 physical registers for integers are used, named rr0 to rr63. In order to select a physical register for renaming, a FIFO queue is available which originally contains registers rr32 to rr63. Registers are added to this queue whenever they go available. What is the speedup compared to the previous execution ignoring the intital transient period? What architectonic registers change of physical register at the end of the execution of the code snippet, and what physical registers have them assigned?

```
                  1   2   3   4    5   6    7    8    9
daddi r3, r0, 5   IF  ID  EX  MEM  WB
ddiv r2, r4, r5       IF  ID  EX   EX  EX   EX   MEM  WB
xor r3, r6, r8            IF  ID   EX  MEM  WB
dsub r2, r9, r10              IF   ID  EX   MEM  WB

A = (10-4)/(9-4)= 1.2
Registers r3 and r2 are assigned to physical registers rr34 and
rr35, respectively.
```

Exercise 22.

A multiple-issue MIPS64 CPU with an issue width of 3 and register renaming is used. The CPU includes the following execution units:

- 1 load/store unit (2 clock cycles).
- 3 general purpose integer units (1 clock cycle).
- 1 multiplication and division unit for integers (4 clock cycles).
- 1 branch unit (1 clock cycle).
- 1 floating point unit (6 clock cycles).

❑ **22.1** What is the minimum CPI that can be achieved in this CPU?

```
1/3 = 0.33
```

This CPU runs the following code snippet:

```
1 ld   r2, 120(r1)
2 dadd r2, r2, r5
3 beqz r2, label
```

The three instructions are distributed in the same clock cycle.

❑ **22.2** What is the minimun time that the dadd instruction needs to wait before being issued to its execution unit? Where is the instruction stored while it is waiting?

```
It must wait until the ld instruction goes out of the execu-
tion unit and forwards the value of r2 => 2 clock cycles. It
is stored in the reservation station of the general purpose
integer unit.
```

In order to avoid the previous stall, the program is modified. Instruction `dadd r2, r2, r5` is substituted by instruction `dadd r2, r1, r5`.

❑ **22.3** How long does the dadd instruction need to wait before being issued to its execution unit?

```
0 clock cycles. The RAW dependency with ld is not present now.
The WAW dependency on register r2 is avoided using register
renaming.
```

❑ **22.4** What instruction will publish first the value of the computed register, ld or dadd? Why?

```
The dadd instruction, since it is issued to its execution unit
at the same time as ld and its execution unit requires one
clock cycle, while ld requires two clock cycles.
```

❑ **22.5** The beqz instruction waits for the value of r2 in the branch reservation station. What instruction will provide the value of this architectural register? Why?

```
The instruction that modifies r2, that is, dadd r2, r1, r5.
When the dadd instruction goes through the ID stage, register
r2 is renamed to rr33; thus, the r2 architectural register is
now linked to rr33. The next instruction, beqz r2, label is
equivalent to beqz rr33, label; thus, the dadd instruction is
the one providing the value for architectural register r2. The
value published by the ld r2, 120(r1) instruction is rr32, that
is associated now with architectural register r2.
```

❑ **22.6** What instruction is retired first, ld or dadd? Why?

```
ld, because instructions are retired in order.
```

Exercise 23.

Many modern CPUs are superscalar. Thus, in ideal scenarios they execute as many instructions per clock cycle as their issue width. As the issue width increases, the number of transistors required to implement the CPU rises. Nowadays, superscalar CPUs feature an issue width equal or less than 4, although current technology would allow CPUs with larger issue width.

❑ **23.1** Why do you think superscalar CPUs are not manufactured with larger issue width?

```
Because programs feature a limited ILP and superscalar microar-
chitectures with larger issue width would not imply performance
improvements.
```

❑ **23.2** Why do modern CPUs feature clock frequencies under 4 GHz?

```
Current manufacturing technologies of integrated circuits re-
quire high amounts of engery to increase clock frequency. Thus,
higher frequencies would generate so much heat that would not
be possible to dissipate.
```

❑ **23.3** What is the implication in cost and energy dissipation when the same microarchitecture is implemented using a more modern technology?

```
Dissipated energy and cost are reduced. Dissipated energy in-
creases with surface and frequency. When a more modern technol-
ogy is used, surface is reduced, also reducing cost. Using the
same frequency, dissipated energy decreases.
```

❑ **23.4** Modern processors feature a large number of transistors compared with older ones. What are these transistors used for?

```
Modern CPUs implement multiple-issue microarchitectures with
replicated units, extensions of the architecture such as ad-
vanced instructions sets (SIMD, for instance), multiple cores
in the same chip, multi-level cache memories and other func-
tional units of the computer.
```

Exercise 24.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) Multi-threaded programming does not provide performance benefits in a single-core CPU with no simultaneous multi-threading capabilities.

B) Simultaneous multi-threading technology is equivalent to multi-core.

C) SIMD extensions allow improving the performance of multi-thread programs, but not for single-threaded programs.

D) Multi-core technology allows improving throughput, but response time is not improved in the case of single-threaded programs.

E) The memory wall prevents the number of cores of a CPU from going beyond a given threshold.

```
D and E
```

Exercise 25.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) According to Flynn's taxonomy, a single-processor running a single-task operating system is classified as an SIMD system.

B) According to Flynn's taxonomy, GPUs of modern computers are examples of SIMD devices.

C) According to Flynn's taxonomy, MIMD is a generalization of von Neumann's architecture in modern computers with several cores or several processors.

D) A multi-threaded application provides better performance when its threads run in parallel.

E) A single-threaded application takes more advantage of a high-performance pipeline (instruction-level parallelism) than exploiting some type of thread-level parallelism.

F) *Simultaneous Multithreading* technology provides the ability for a CPU to run several threads simultaneously without replicating any functional unit.

```
B, C, D and E
```

Exercise 26.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) The pipeline technique divides the execution of instructions in several stages that run sequentially to increase the performance.

B) When a service call, an interrupt or an exception occurs, a handler stored in the address space of the task is executed.

C) Non-maskable interrupts are always served.

D) Superscalar is a term used to describe CPUs that replicate internal components, and thus, are able to work on several instructions in each stage of the pipeline.

E) A CPU running in user privilege level can gain access to the memory addresses of the operating system.

```
C and D
```

Exercise 27.

Using VirtualBox as a hypervisor allows running a Linux-based virtual machine in the lab. What type of hypervisor is VirtualBox? Why?

```
Type 2. It runs on top of a host operating system (Windows),
but not directly on hardware.
```