Lecturer: (Date)	**Approved by:** (Date)
(Signature & Fullname)	(Signature, Position & Fullname)

(The above part must be hidden when copying for exam)

Ho Chi Minh City University of Technology **Faculty of Computer Science and Engineering**	**FINAL EXAM**	Semester/AY	1	2020-2021
		Date	January 25th, 2021	
	Course name	**Computer Architecture**		
	Course ID	CO2007		
	Duration	70 minutes	**Code**	201F01

Notes:

- Write you answers right after every question;
- Students are allowed to use materials printed/written on **TWO** A4 papers (4 pages);
- All pages of this question sheet **MUST** be returned. Any missing page will let you get 0 mark!

Student's name: ANSWERS...

Student's ID:..

Learning outcomes - Questions mapping:

- L.O.1 - Estimate performance of a computer under given parameters: 1 - 5;
- L.O.2 - Explain basic instructions in the instruction sets: 6 - 15
- L.O.3 - Describe the principles and working mechanism of a processor: 16 - 29
- L.O.4 - Describe the principles and working mechanism of memory hierarchy: 30 - 40

Questions for the final test are started from here:

1. Is "throughput" or "response time" affected when more current processors are added to the system?

Answer: Throughput

2. Which factors will affect energy consumption of a computer system (given that energy consumption is a product of power consumption and execution time)?
A - Processor architecture
B - Cache size
C - Clock frequency

Answer: A, B, and C

The following values are used for Question 3 to Question 5:

Assume that a program consists of 200×10^6 floating point (FP) instructions, 100×10^6 integer instructions, 80×10^6 data transfer instructions, and 40×10^6 branch instructions. CPIs of instruction types are 2, 1, 4, and 2, respectively.

3. What is average CPI of the above program?

Answer: $\approx \mathbf{2.14}/\frac{15}{7}$

4. By how much must we improve the CPI of FP instructions if we want the program to run 1.5× faster?

Answer: newCPI(FP) = 0.5

5. How much is the execution time of the program if CPIs of integer and floating point instructions are reduced by 40% while CPIs of data transfer and branch instructions are reduced by 30%. Assume that the processor works at a 2GHz frequency.

Answer: $\approx$ **0.29s**

6. Which following standard instructions can be used to implement the pseudo-instruction **li** `$v0, 10` (assign an integer to a register)?
   A - **addi** `$v0, $zero, 10`
   B - **ori** `$v0, $zero, 10`
   C - **xori** `$v0, $zero, 10`

Answer: A and B

7. Assume that the register `$s0` contains `0xFFFFCA16`; which instruction can be used to change this register content to `0x0000CA16`?

Answer: `andi $s0, $s0, 0xFFFF` or
`andi $s0, $s0, 0xCA16` or
`addiu $s0, $zero, 0xCA16`

8. What is the main purpose of the following sequence (the `$sp` register is the stack pointer)?

```
addi $sp, $sp,-4
sw $ra,0($sp)
```

Answer: Push `$ra` to the top of the stack

Following data is used for questions from 9 to 10:

Given the following memory locations/cells, as depicted in Figure 1, used in a standard MIPS-based system:

address:	8	9	10	11
	0x12	0x34	0x56	0x78
	0x9A	0xBC	0xDE	0xF0
address:	12	13	14	15

Figure 1: Memory map for questions from 9 to 10

Assume that the `$s0` register stores value of 8.

9. What is the value of the `$t0` register after the instruction **lb** `$t0, 6($s0)` is executed?

Answer: `0xFFFFFFDE`

10. What is the value of the `$t1` register after the instruction **lui** `$t1, 4($s0)` is executed?

Answer: Compile error

Following data is used for questions from 11 to 12:

Execute the following MIPS instructions with a MIPS processor:

```
lui $t0, 0
ori $t0, $t0, 7
lui $t1, 0
ori $t1, $t1, 2
div $t0, $t1
mfhi $s0
mflo $s1
```

11. What is the value of the `$s0` register?

Answer: 1

12. What is the value of the `$s1` register?

Answer: 3

13. Which is decimal value of the IEEE 754 single precision 0x40C80000?

Answer: 6.25

14. Which is the IEEE 754 single precision representation of 12.625?

Answer: 0x414a0000

15. Why MIPS floating point instructions do not support immediate numbers?

Answer: Cannot represent instructions in binary with only 32 bits

Following data is used for Question 16 to Question 17:

A pipeline MIPS processor consists of five stages: Instruction Decode (ID), Instruction Fetch (IF), Instruction Execution (EX), Memory Access (MEM), and Write Back (WB). Assume that the ID, EX, and MEM stages need 200 ns while the ID and WB stage require only 150 ns.

16. Which is a correct execution order of those stages

Answer: IF, ID, EX, MEM, WB

17. What is speed-up of a program consisting of 1.000.000 instructions processed by the pipeline processor when compared to the single cycle processor?

Answer: $\approx$ 4.50

The following 32-bit datapath diagram of the standard MIPS processor (Figure 2) is used for Question 18 to Question 23:

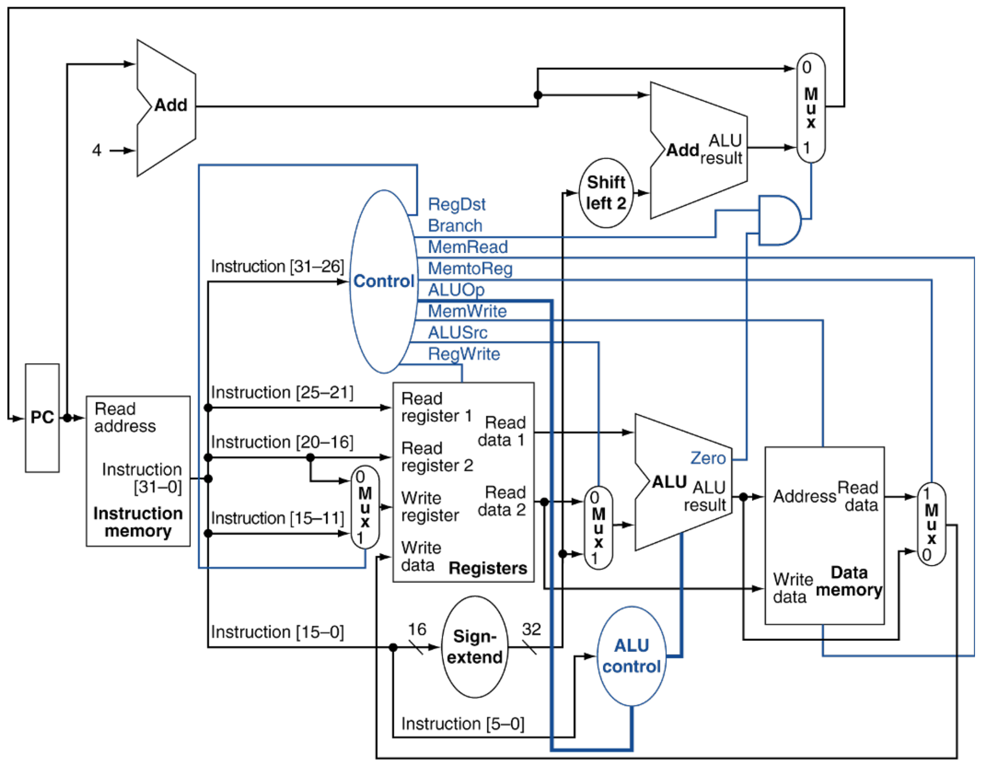

Figure 2: A single-cycle MIPS processor. Source: Computer Organization and Design: the Hardware/Software Interface, fifth edition, Morgan Kaufmann Publisher

18. Which is the size of two inputs “Read Register 1” and “Read Register 2” of the Register Block?

Answer: 5

19. How many **MUX** Block taking part in the execution of R-format arithmetic instructions?

Answer: 4

20. According this simplified MIPS implementation, which instruction is the longest instruction?

Answer: lw

21. Which instructions will use the `Zero` output?

Answer: beq and bne

22. Assume that **add** `$t0, $t1, $zero` is executing, what are the values of `RegDst`, `ALUOp` and `ALUSrc`?

Answer: RegDst = 1; ALUOp = 10; ALUSrc = 0

23. Assume that **lw** `$t0, 0($t1)` is executing, which control signals are 0?

Answer: RegDst, Branch, ALUOp, MemWrite

The following 32-bit datapath MIPS processor diagram (Figure 3) is used for Question 24 to Question 29:

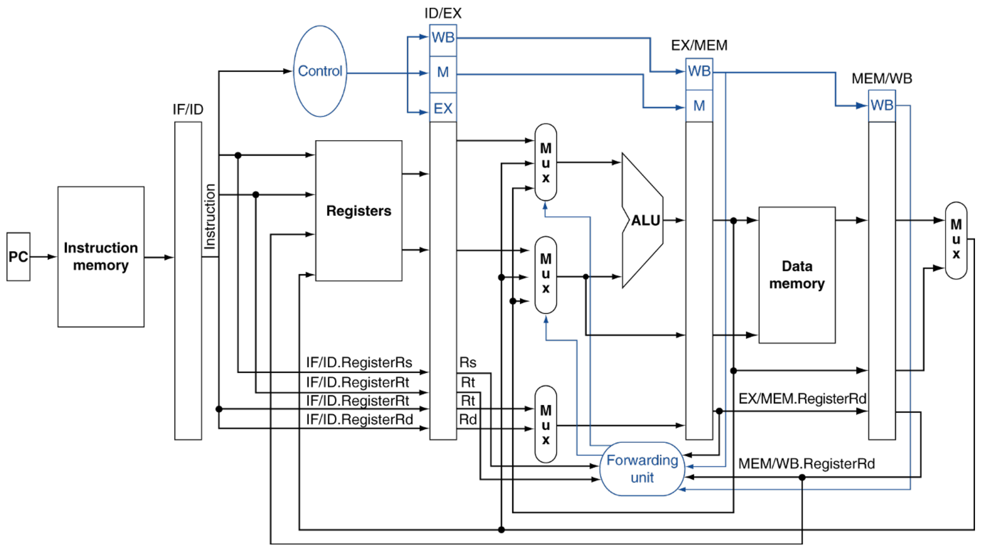

Figure 3: A pipeline MIPS processor. Source: Computer Organization and Design: the Hardware/Software Interface, fifth edition, Morgan Kaufmann Publisher

24. Which control signals (generated by the **Control** Block) transferred to the *EX/MEM* register?

Answer: MemRead, MemWrite, Branch, and RegWrite

The following MIPS instructions sequence, started executing at the first cycle, is used for Question 25 to Question 29:

```
lw      $t0, 100($s0)
add     $t0, $t0, $t1
sw      $t1, 100($t0)
add     $t0, $t1, $t0
lw      $t1, 100($t0)
add     $t0, $t0, $t1
```

25. If the Forwarding technique is **NOT** applied, how many data hazards are there in the sequence?

Answer: 5

26. If the Forwarding technique is **NOT** applied, how many cycles does the sequence need to finish?

Answer: 18

27. If the Forwarding technique is applied, how many cycles does the sequence need to finish?

Answer: 12

28. At cycle number 5, what are values for Forwarding units (ForwardA and ForwardB signals)?

Answer: ForwardA = 01, ForwardB = 00

29. At cycle number 5, which control signals at functional units are 0 if the Forwarding technique is applied for solving data hazards?

Answer: MemRead, MemWrite, Branch, ALUSrc, MemtoReg

The following values are used for Question 30 to Question 37:

Assume that a main memory has 32-bit byte address. A 64 KB cache consists of 4-word blocks.

30. Which is the size of the main memory?

Answer: 4GB

31. If the cache uses "direct mapped", how many bits are the tag field?

Answer: 16

32. How many memory bits do we need to use to build in the direct mapped cache?

Answer: 593,920 or 580Kbit

33. If the cache uses "fully associative", how many bits are the tag field?

Answer: 28

34. How many memory bits do we need to use to build the fully associative cache?

Answer: 643,072 or 628Kbit

35. If the cache uses "2-way set associative", how many sets are there in the cache?

Answer: 2,048

36. If the cache uses "2-way set associative", how many bits are the tag field?

Answer: 17

37. How many memory bits do we need to use to build the 4-way set associative cache?

Answer: 602,112 or 588Kbit

The following parameters are used for the tree last questions of the test:

A CPU functions at 4 GHz with a base CPI 1. Assume that the L1 cache miss rate per instruction is 2%. The accessing time of the main memory is 100 ns. The L2 cache miss rate per instruction is 0.5% while accessing time of L2 is 5ns.

38. What is the average CPI if the L2 cache is removed?

Answer: 9

39. What is the L2 miss penalty?

Answer: 400 cycles

40. What is the average CPI when both L1 and L2 are used?

Answer: 3.4

— The test consists of **40** questions printed on **6** pages —

MIPS Reference Data

1

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	`add`	R	R[rd] = R[rs] + R[rt]	(1)	0 / 20_{hex}
Add Immediate	`addi`	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	`addiu`	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	`addu`	R	R[rd] = R[rs] + R[rt]		0 / 21_{hex}
And	`and`	R	R[rd] = R[rs] & R[rt]		0 / 24_{hex}
And Immediate	`andi`	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	`beq`	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	`bne`	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	`j`	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	`jal`	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	`jr`	R	PC=R[rs]		0 / 08_{hex}
Load Byte Unsigned	`lbu`	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	`lhu`	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	`ll`	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	`lui`	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	`lw`	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	`nor`	R	R[rd] = ~ (R[rs] | R[rt])		0 / 27_{hex}
Or	`or`	R	R[rd] = R[rs] | R[rt]		0 / 25_{hex}
Or Immediate	`ori`	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	`slt`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0 / $2a_{hex}$
Set Less Than Imm.	`slti`	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	`sltiu`	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	`sltu`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0 / $2b_{hex}$
Shift Left Logical	`sll`	R	R[rd] = R[rt] << shamt		0 / 00_{hex}
Shift Right Logical	`srl`	R	R[rd] = R[rt] >>> shamt		0 / 02_{hex}
Store Byte	`sb`	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	`sc`	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	`sh`	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	`sw`	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	`sub`	R	R[rd] = R[rs] - R[rt]	(1)	0 / 22_{hex}
Subtract Unsigned	`subu`	R	R[rd] = R[rs] - R[rt]		0 / 23_{hex}

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1b'0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC+4[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2's comp.)
(7) Atomic test&set pair; R[rt] = 1 if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

	31–26	25–21	20–16	15–11	10–6	5–0
R	opcode	rs	rt	rd	shamt	funct

	31–26	25–21	20–16	15–0
I	opcode	rs	rt	immediate

	31–26	25–0
J	opcode	address

ARITHMETIC CORE INSTRUCTION SET

2

NAME, MNEMONIC		FOR-MAT	OPERATION		OPCODE / FMT /FT / FUNCT (Hex)
Branch On FP True	`bc1t`	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	`bc1f`	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	`div`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	`divu`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	`add.s`	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	`add.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	`c.`*x*`.s`*	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	`c.`*x*`.d`*	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*
* (*x* is `eq`, `lt`, or `le`) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)					
FP Divide Single	`div.s`	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	`div.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	`mul.s`	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	`mul.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	`sub.s`	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	`sub.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	`lwc1`	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	`ldc1`	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	`mfhi`	R	R[rd] = Hi		0 /--/--/10
Move From Lo	`mflo`	R	R[rd] = Lo		0 /--/--/12
Move From Control	`mfc0`	R	R[rd] = CR[rs]		10 /0/--/0
Multiply	`mult`	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	`multu`	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Shift Right Arith.	`sra`	R	R[rd] = R[rt] >> shamt		0/--/--/3
Store FP Single	`swc1`	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	`sdc1`	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING-POINT INSTRUCTION FORMATS

	31–26	25–21	20–16	15–11	10–6	5–0
FR	opcode	fmt	ft	fs	fd	funct

	31–26	25–21	20–16	15–0
FI	opcode	fmt	ft	immediate

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	`blt`	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	`bgt`	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	`ble`	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	`bge`	if(R[rs]>=R[rt]) PC = Label
Load Immediate	`li`	R[rd] = immediate
Move	`move`	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes