University of Nottingham Ningbo China

School of Computer Science

A Level 1 Module, Spring Semester 2022–2023

Systems and Architecture (AE1SYS)

Time Allowed: TWO Hours

Final Examination

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer ALL questions

Total Marks: 100

No calculators are permitted in this examination

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject-specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so.

APPENDIX: MIPS Reference Card

INFORMATION FOR INVIGILATORS:
Collect both the exam papers and the answer booklets at the end of the exam.

Question 1 (16 marks)

(a) Convert the decimal number -79 to 10-bit two's complement binary representation. (4)

(b) Convert the decimal number 385 to its corresponding single-precision IEEE 754 floating point number, and present the result as 8 hexadecimal digits. (6)

(c) Suppose there exists a 12-bit IEEE 754 floating point format, with 1 sign bit, 6 exponent bits, and 5 fraction bits. Answer the following 3 sub-questions below. (6)

(i) How would $-\infty$ be represented in this 12-bit format? Show its 12-bit representation.

(ii) How would the smallest positive normalised number be represented? Show its 12-bit representation.

(iii) Give the nearest representation n of 5.612 in this format.

Question 2 (14 marks)

(a) Describe the procedure calling conventions for MIPS, specifically: (6)

(1) What is the instruction used to invoke a callee, and what is the instruction for returning to the caller?

(2) How are arguments passed and results returned?

(3) Which of the user registers need to be preserved by the callee?

(b) If we execute the action: `addu $s0, $s1, $s2`, the MIPS processor will ignore overflows. Implement in MIPS assembly a test following the `addu` instruction which jumps to a label named overflow, if an overflow has indeed occurred. You can use any instruction that appears in the MIPS reference card. You will need to restrict your solution within 10 instructions. Comment your code properly. The code template is like below: (8)

```
addu $s0, $s1, $s2
#Your code here to check the overflow.

No_overflow:
#Continue normal processing - not your part.
Overflow:
#Code handling overflow - not your part.
Exit:
```

Question 3 (11 marks)

(a) Give the full names of the following CPU executions stages: (*i*) IF; (*ii*) ID; (*iii*) WB. (3)

(b) Given what you understand by the MIPS instruction format, do you think the current MIPS ISA (32-bit) can support a total of 150 different R-, I-, and J-type instructions? Give a [yes/no/others] answer, and provide your brief explanation, with calculations if necessary. (4)

(c) What is the result of interpreting 0x82984000 as a MIPS instruction? Give your answer as an assembly language instruction, use numeric register names, and show intermediate steps. Hint: Use the MIPS reference card if necessary. (4)

Question 4 **(19 marks)**

(a) Design a 5-stage pipelined MIPS CPU for an instruction set that contains only the following two instructions: `lw` and `add`. Assume that the instruction formats are the same as in the 32-bit MIPS architecture that we learnt in this module. Draw the CPU design, by showing all the hardware functional blocks, all the data links, but you don't have to show the control signals in the datapath. You must show only the minimal hardware required to implement these two instructions. (6)

(b) Suppose you need to add one more instruction, named `add21`, which has the same format as an `add`, namely `add21 rd, rs, rt`. When this instruction executes, it adds `2 * rs` and `1 * rt` together, and saves the result into `rd`. For example, in `add21 $s0, $t0, $t1`. If `$t0` = 1 and `$t1` = 3, then `$s0` = 5. Modifying from the CPU design in question 4(a) to add this instruction into the ISA that this CPU can execute. Note that you cannot alter the current design in question 4(a), but can only add circuit gadget into the current design. Assume you have only one extra circuit gadget, which is a 2-input 32-bit ALU, and unlimited number of multiplexers, how do you implement this new CPU? Your design must remain in 5-stage pipeline. Does this addition cause any hazard(s) and why? If so, (1) what types of hazard(s)? (2) How do you handle the hazard(s) using a software solution? (6)

(c) Now, assume that you have only unlimited number of multiplexers, but no extra ALU or any other gadgets. How do you implement the abovementioned CPU with `add21`? Your design must remain in 5-stage pipeline. Does this addition cause any hazard(s) and why? If so, (1) what types of hazard(s)? (2) How do you handle the hazard(s) using a software solution? (7)

Question 5 (27 marks)

Questions (1)-(6) are Single Choice Questions. Identify exactly one correct answer. (9)

(1) MAC Address is the example of

(a) Application Layer.
(b) Physical Layer.
(c) Data Link Layer.
(d) Transport Layer.

(2) Which of the following IP address class is Multicast:

(a) Class D.
(b) Class C.
(c) Class B.
(d) Class A.

(3) What is the Demilitarized Zone?

(a) The area between ISP to Military area.
(b) The area surrounded by secured servers.
(c) The area surrounded by the Military.
(d) The area between firewall and connection to an external network.

(4) What is the use of Ping command?

(a) To test a hard disk fault.
(b) To test a device on the network is reachable.
(c) To test a bug in an application.
(d) To test a printer quality.

(5) DHCP is the abbreviation of:

(a) Domain Host Configuration Protocol.

(b) Domain Host Control Protocol.

(c) Dynamic Host Configuration Protocol.

(d) Dynamic Host Control Protocol.

(6) What does the port number in a TCP connection specify?

(a) It specifies the quality of the data and connection.

(b) It specifies the communication process on the two end systems.

(c) It specifies the size of data.

(d) All of the above.

Questions (7)-(12) are Multiple Choice Questions. Identify all correct answers. Each incorrect answer causes mark deduction. (18)

(7) Which of the following are guided media?

(a) Bluetooth.

(b) Cat-5 Cable.

(c) Coaxial cable.

(d) WiFi.

(e) Optical Fiber.

(8) Which of the following protocols are defined in Transport Layer?

(a) TCP.

(b) IP.

(c) UDP.

(d) FTP.

(e) SMTP.

(9) What is a Firewall in Computer Network?

(a) A physical boundary of computer network.

(b) An operating system of computer network.

(c) A computer network management system.
(d) A system designed to prevent unauthorized access.
(e) A web browsing software.

(10) Optical fiber possesses the following properties:

(a) Immune to electromagnetic interference.
(b) Small diameter.
(c) Long distance.
(d) Big signal attenuation.
(e) Hard to tap

(11) Which of the following are not the Networking Devices?

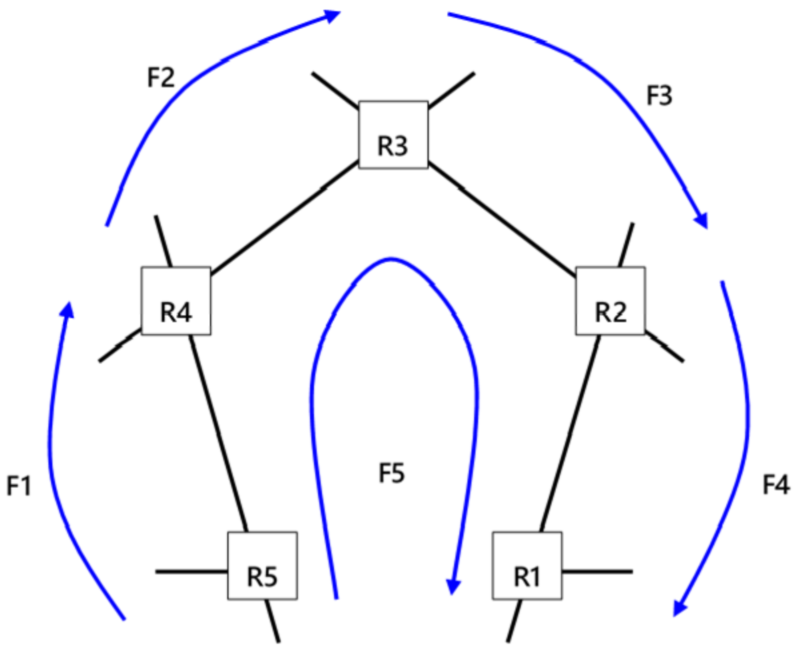
(a) Modem.
(b) Router.
(c) Linux.
(d) Email.
(e) Firewall.

(12) Which of the following are applicable for IPV6?

(a) Large address space.
(b) Better and more compact header format.
(c) No support to quality of service.
(d) Built-in security.
(e) Extensibility.

Question 6 **(13 marks)**

Consider the network of five routers shown in Figure . Each link has a capacity of 38Mbps. Assume there is no contention on the access links or for router backplane resources, i.e., the only constraints are the link capacities between routers. There are five flows in the network, labelled *F1*, *F2*, *F3*, *F4* and *F5*, which pass through the routers indicated. *F1* traverses *R5* → *R4*, *F2* traverses *R4* → *R3*, *F3* traverses *R3* → *R2*, *F4* traverses *R2* → *R1*, and *F5* shares the links with every other flow and traverses *R5* → *R4* → *R3* → *R2* → *R1*.



(a) Assume each router implements: First In First Out (FIFO) queuing. If each flow consists of an identical and constant bit rate UDP flow with equal packet sizes, what is the resulting rate for each flow? Assume that FIFO drops packets with a uniform probability. (10)

(b) All devices need to know the destination IP address. If the destination IP address is on a local network, what is the next step? If the destination IP address is not on a local network, what is the next step? (3)

MIPS/SPIM Reference Card

CORE INSTRUCTION SET (INCLUDING PSEUDO INSTRUCTIONS)

NAME	MNE-MON-IC	FOR-MAT	OPERATION (in Verilog)		OPCODE/ FUNCT (Hex)
Add	`add`	R	R[rd]=R[rs]+R[rt]	(1)	0/20
Add Immediate	`addi`	I	R[rt]=R[rs]+SignExtImm	(1)(2)	8
Add Imm. Unsigned	`addiu`	I	R[rt]=R[rs]+SignExtImm	(2)	9
Add Unsigned	`addu`	R	R[rd]=R[rs]+R[rt]	(2)	0/21
Subtract	`sub`	R	R[rd]=R[rs]-R[rt]	(1)	0/22
Subtract Unsigned	`subu`	R	R[rd]=R[rs]-R[rt]		0/23
And	`and`	R	R[rd]=R[rs]&R[rt]		0/24
And Immediate	`andi`	I	R[rt]=R[rs]&ZeroExtImm	(3)	c
Nor	`nor`	R	R[rd]=∼(R[rs]|R[rt])		0/27
Or	`or`	R	R[rd]=R[rs]|R[rt]		0/25
Or Immediate	`ori`	I	R[rt]=R[rs]|ZeroExtImm	(3)	d
Xor	`xor`	R	R[rd]=R[rs]ˆR[rt]		0/26
Xor Immediate	`xori`	I	R[rt]=R[rs]ˆZeroExtImm		e
Shift Left Logical	`sll`	R	R[rd]=R[rs]≪shamt		0/00
Shift Right Logical	`srl`	R	R[rd]=R[rs]≫shamt		0/02
Shift Right Arithmetic	`sra`	R	R[rd]=R[rs]≫>shamt		0/03
Shift Left Logical Var.	`sllv`	R	R[rd]=R[rs]≪R[rt]		0/04
Shift Right Logical Var.	`srlv`	R	R[rd]=R[rs]≫R[rt]		0/06
Shift Right Arithmetic Var.	`srav`	R	R[rd]=R[rs]≫>R[rt]		0/07
Set Less Than	`slt`	R	R[rd]=(R[rs]<R[rt])?1:0		0/2a
Set Less Than Imm.	`slti`	I	R[rt]=(R[rs]<SignExtImm)?1:0	(2)	a
Set Less Than Imm. Unsign.	`sltiu`	I	R[rt]=(R[rs]<SignExtImm)?1:0	(2)(6)	b
Set Less Than Unsigned	`sltu`	R	R[rd]=(R[rs]<R[rt])?1:0	(6)	0/2b
Branch On Equal	`beq`	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4
Branch On Not Equal	`bne`	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5
Branch Less Than	`blt`	P	if(R[rs]<R[rt]) PC=PC+4+BranchAddr		
Branch Greater Than	`bgt`	P	if(R[rs]>R[rt]) PC=PC+4+BranchAddr		
Branch Less Than Or Equal	`ble`	P	if(R[rs]<=R[rt]) PC=PC+4+BranchAddr		
Branch Greater Than Or Equal	`bge`	P	if(R[rs]>=R[rt]) PC=PC+4+BranchAddr		
Jump	`j`	J	PC=JumpAddr	(5)	2
Jump And Link	`jal`	J	R[31]=PC+4; PC=JumpAddr	(5)	2
Jump Register	`jr`	R	PC=R[rs]		0/08
Jump And Link Register	`jalr`	R	R[31]=PC+4; PC=R[rs]		0/09
Move	`move`	P	R[rd]=R[rs]		
Load Byte	`lb`	I	R[rt]={24'b0, M[R[rs]+ZeroExtImm](7:0)}	(3)	20
Load Byte Unsigned	`lbu`	I	R[rt]={24'b0, M[R[rs]+SignExtImm](7:0)}	(2)	24
Load Halfword	`lh`	I	R[rt]={16'b0, M[R[rs]+ZeroExtImm](15:0)}	(3)	25
Load Halfword Unsigned	`lhu`	I	R[rt]={16'b0, M[R[rs]+SignExtImm](15:0)}	(2)	25
Load Upper Imm.	`lui`	I	R[rt]={imm,16'b0}		f
Load Word	`lw`	I	R[rt]=M[R[rs]+SignExtImm]	(2)	23
Load Immediate	`li`	P	R[rd]=immediate		
Load Address	`la`	P	R[rd]=immediate		
Store Byte	`sb`	I	M[R[rs]+SignExtImm] (7:0)=R[rt](7:0)	(2)	28
Store Halfword	`sh`	I	M[R[rs]+SignExtImm] (15:0)=R[rt](15:0)	(2)	29
Store Word	`sw`	I	M[R[rs]+SignExtImm]=R[rt]	(2)	2b

REGISTERS

NAME	NMBR	USE	STORE?
$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
$f0-$f31	0-31	Floating Point Registers	Yes

(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 }
(4) JumpAddr = {PC[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2 s comp.)

BASIC INSTRUCTION FORMATS, FLOATING POINT INSTRUCTION FORMATS

R	31 opcode 26	25 rs 21	20 rt 16	15 rd 11	10 shamt 6	5 funct 0
I	31 opcode 26	25 rs 21	20 rt 16	15 immediate 0		
J	31 opcode 26	25 immediate 0				
FR	31 opcode 26	25 fmt 21	20 ft 16	15 fs 11	10 fd 6	5 funct 0
FI	31 opcode 26	25 fmt 21	20 rt 16	15 immediate 0		

ARITHMETIC CORE INSTRUCTION SET

NAME	MNE-MON-IC	FOR-MAT	OPERATION (in Verilog)		OPCODE/ FMT/FT/ FUNCT
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
Multiply	`mult`	R	{Hi,Lo}=R[rs]$*$R[rt]		0/–/–/18
Multiply Unsigned	`multu`	R	{Hi,Lo}=R[rs]$*$R[rt]	(6)	0/–/–/19
Branch On FP True	`bc1t`	FI	if(FPCond) PC=PC+4+BranchAddr	(4)	11/8/1/–
Branch On FP False	`bc1f`	FR	if(!FPCond) PC=PC+4+BranchAddr	(4)	11/8/0/–
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 Add Single	`add.s`	FR	F[fd]=F[fs]+F[ft]		11/10/–/0
FP Divide Single	`div.s`	FR	F[fd]=F[fs]/F[ft]		11/10/–/3
FP Multiply Single	`mul.s`	FR	F[fd]=F[fs]$*$F[ft]		11/10/–/2
FP Subtract Single	`sub.s`	FR	F[fd]=F[fs]-F[ft]		11/10/–/1
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 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 Double	`mul.d`	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}$*${F[ft],F[ft+1]}		11/11/–/2
FP Subtract Double	`sub.d`	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}-{F[ft],F[ft+1]}		11/11/–/1
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]		16/0/–/0
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/–/–/–
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/–/–/–

ASSEMBLER DIRECTIVES

`.data` [*addr*]*	Subsequent items are stored in the data segment
`.kdata` [*addr*]*	Subsequent items are stored in the kernel data segment
`.ktext` [*addr*]*	Subsequent items are stored in the kernel text segment
`.text` [*addr*]*	Subsequent items are stored in the text
	* starting at [*addr*] if specified
`.ascii` *str*	Store string *str* in memory, but do not null-terminate it
`.asciiz` *str*	Store string *str* in memory and null-terminate it
`.byte` $b_1, \ldots, b_n$	Store the n values in successive bytes of memory
`.double` $d_1, \ldots, d_n$	Store the n floating-point double precision numbers in successive memory locations
`.float` $f_1, \ldots, f_1$	Store the n floating-point single precision numbers in successive memory locations
`.half` $h_1, \ldots, h_n$	Store the n 16-bit quantities in successive memory halfwords
`.word` $w_1, \ldots, w_n$	Store the n 32-bit quantities in successive memory words
`.space` n	Allocate n bytes of space in the current segment
`.extern` *symsize*	Declare that the datum stored at *sym* is *size* bytes large and is a global label
`.globl` *sym*	Declare that label *sym* is global and can be referenced from other files
`.align` n	Align the next datum on a 2^n byte boundary, until the next `.data` or `.kdata` directive
`.set at`	Tells SPIM to complain if subsequent instructions use `$at`
`.set noat`	prevents SPIM from complaining if subsequent instructions use `$at`

SYSCALLS

SERVICE	`$v0`	ARGS	RESULT
`print_int`	1	integer `$a0`	
`print_float`	2	float `$f12`	
`print_double`	3	double `$f12/$f13`	
`print_string`	4	string `$a0`	
`read_int`	5		integer (in `$v0`)
`read_float`	6		float (in `$f0`)
`read_double`	7		double (in `$f0`)
`read_string`	8	buf `$a0`, buflen `$a1`	
`sbrk`	9	amount `$a`	address (in `$v0`)
`exit`	10		

EXCEPTION CODES

Number	Name	Cause of Exception
0	Int	Interrupt (hardware)
4	AdEL	Address Error Exception (load or instruction fetch)
5	AdES	Address Error Exception (store)
6	IBE	Bus Error on Instruction Fetch
7	DBE	Bus Error on Load or Store
8	Sys	Syscall Exception
9	Bp	Breakpoint Exception
10	RI	Reserved Instruction Exception
11	CpU	Coprocessor Unimplemented
12	Ov	Arithmetic Overflow Exception
13	Tr	Trap
15	FPE	Floating Point Exception

[1] Patterson, David A; Hennessy, John J.: Computer Organization and Design, 3rd Edition. Morgan Kaufmann Publishers. San Francisco, 2005.