Homework 4 Answer Sheet

Please state the name, SID and email of each member of your group.

member	name	SID	email
#1 (contact person)	Oscar Hoffmann	40141036	ohoffmann2-c@cityu.edu.hk
#2	Julia Jaksic	40141024	Jsajaksic2-c@cityu.edu.hk
#3			

A. Do all members make significant contributions to this homework? If not, please specify the details.

B. Please explain how many types of instructions are supported in your processor, and explain the format of each type of instructions (e.g., which bits are used as the operation or function code, which bits are used to index the 1st, 2nd or 3rd operand, and which bits are used to store the immediate number). You can draw figures to better explain your answer.

Our CPU supports all of the 16 instructions.

R-type

op	rs	rt	rd	func
15-12	11-9	8-6	5-3	2-0

I-type

op	rs	rd	immediate
15-12	11-9	8-6	5-0

J-type

op	immediate
15-12	11-0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
li (I)	0	0	0	1	1	0	0	x	x	x	1	0	0
add (R)	0	1	1	0	0	0	0	0	0	1	0	0	1
and (R)	1	1	1	1	0	0	0	0	1	0	0	1	0
or (R)	0	0	0	0	0	0	0	0	1	1	0	1	1
load (I)	0	0	1	0	0	0	1	x	x	x	0	0	1
store (I)	0	0	1	1	0	0	1	x	x	x	0	0	1
move (I)	0	1	0	0	1	0	1	x	x	x	1	0	1
addi (I)	0	1	0	1	0	0	1	x	x	x	0	0	1
andi (I)	0	1	1	1	0	1	0	x	x	x	0	1	0
ori(I)	1	0	0	0	0	1	1	x	x	x	0	1	1
ble (I)	1	0	0	1	1	1	0	x	x	x	1	1	0
bne (I)	1	0	1	0	1	1	1	x	x	x	1	1	1
jump (J)	1	0	1	1	x	x	x	x	x	x	x	x	x
call (J)	1	1	0	0	x	x	x	x	x	x	x	x	x
rtn (J)	1	1	0	1	x	x	x	x	x	x	x	x	x
halt	1	1	1	0	x	x	x	x	x	x	x	x	x

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
li (I)	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0
add (R)	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
and (R)	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
or (R)	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
load (I)	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1
store (I)	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1
move (I)	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
addi (I)	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1
andi (I)	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0
ori(I)	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
ble (I)	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0
bne (I)	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1
jump (J)	1	0	1	1	0	0	0	0	0	0	0	1	0	0	x	x	x
call (J)	1	1	0	0	0	0	0	0	0	0	0	0	1	0	x	x	x
rtn (J)	1	1	0	1	0	0	0	0	0	0	0	0	0	1	x	x	x
halt	1	1	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x

C. Please explain the format of each instruction (including the format of this instruction and its operation codes, and other information if needed).

Clarification of the tables:

1. Op (0-3) is the opcode from the instruction that is feeded into the Control unit

2. **RegDst, ALUsrc, MemtoReg, Regwrite, MemRead, MemWrite, Branch, Jump, call, return** and **ALUOp (1-3)** are the output-pins from the control unit.

3. func (1-3) is the three last bits in the R-type instruction which together with the **ALUOp (1-3)** controls the signal sent to the ALU from the ALU control.

4. cOp (1-3) is the control signal sent to the ALU from the ALU control

li

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
li (I)	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
li (I)	0	0	0	1	1	0	0	x	x	x	1	0	0

I-Type

add

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
add (R)	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
add (R)	0	1	1	0	0	0	0	0	0	1	0	0	1

R-type

and

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
and (R)	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
and (R)	1	1	1	1	0	0	0	0	1	0	0	1	0

R-type

or

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
or (R)	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
or (R)	0	0	0	0	0	0	0	0	1	1	0	1	1

R-type

load

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
load (I)	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
load (I)	0	0	1	0	0	0	1	x	x	x	0	0	1

I-type

store

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
store (I)	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
store (I)	0	0	1	1	0	0	1	x	x	x	0	0	1

I-type

move

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
move (I)	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
move (I)	0	1	0	0	1	0	1	x	x	x	1	0	1

I-type

addi

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
addi (I)	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
addi (I)	0	1	0	1	0	0	1	x	x	x	0	0	1

I-type

andi

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
andi (I)	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
andi (I)	0	1	1	1	0	1	0	x	x	x	0	1	0

I-type

ori

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
ori(I)	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
ori(I)	1	0	0	0	0	1	1	x	x	x	0	1	1

I-type

ble

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
ble (I)	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
ble (I)	1	0	0	1	1	1	0	x	x	x	1	1	0

I-type

bne

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
bne (I)	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
bne (I)	1	0	1	0	1	1	1	x	x	x	1	1	1

I-type

jump

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
jump (J)	1	0	1	1	0	0	0	0	0	0	0	1	0	0	x	x	x

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
jump (J)	1	0	1	1	x	x	x	x	x	x	x	x	x

J-type

call

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
call (J)	1	1	0	0	0	0	0	0	0	0	0	0	1	0	x	x	x

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
call (J)	1	1	0	0	x	x	x	x	x	x	x	x	x

J-type

rtn

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
rtn (J)	1	1	0	1	0	0	0	0	0	0	0	0	0	1	x	x	x

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
rtn (J)	1	1	0	1	x	x	x	x	x	x	x	x	x

J-type

halt

Name	Op0	Op1	Op2	Op3	RegDst	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	call	return	ALUOp1	ALUOp2	ALUOp3
halt	1	1	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x

Name	Op0	Op1	Op2	Op3	ALUOp1	ALUOp2	ALUOp3	func1	func2	func3	cOp1	cOp2	cOp3
halt	1	1	1	0	x	x	x	x	x	x	x	x	x

D. Fill the following tables with the machine codes of each instruction of the testing programs:

Test program 1:

instruction	machine code (binary)	machine code (hex)
li $r1, 1	0001000001000001	1041
li $r2, 2	0001000010000010	1082
li $r3, 10	0001000011001010	10CA
add $r2, $r1, $r2	0110001010010001	6291
ble $r2, $r3, -1	1001010011111111	94FF
halt	1110000000000000	E000

Test program 2:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	0001000001000110	1046
li $r2, 5	0001000010000101	1085
andi $r3, $r1, 3	0111001011000011	72C3
ori $r4, $r3, 8	1000011100001000	8708
halt	1110000000000000	E000

Test program 3:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	0001000001000110	1046
li $r2, 5	0001000010000101	1085
and $r3, $r1, $r2	1111001010011010	F29A
li $r8, 0	0001000000000000	1000
store $r3, $r8	0011000011000000	30C0
or $r4, $r1, $r2	0000001010100011	02A3

li $r8, 1	0001000000000001	1001
store $r4, $r8	0011000100000000	3100
li $r8, 1	0001000000000001	1001
load $r7, $r8	0010000111000000	21C0
halt	1110000000000000	E000

Test program 4:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	0001000001000110	1046
li $r2, 4	0001000010000100	1084
call 7	1100000000000111	C007
move $r4, $r3	0100011100000000	4700
li $r1, 7	0001000001000111	1047
call 3	1100000000000011	C003
move $r5, $r3	0100011101000000	4740
jump 3	1011000000000011	B003
add $r3, $r1, $r2	0110001010011001	6299
rtn	1101000000000000	D000
halt	1110000000000000	E000