Exam No : Y3874734

Q1)

For each new instruction, define its operation and determine an opcode value and an instruction coding compatible with the existing instruction set. [4 marks]

New Instructions/operations/opcode:

Program Code Line	Operation	Instructions	OPCODE
temp ← data and mask;	Rt<=Ra AND Rb	and Rt, Ra, Rb	01100
parity ← temp xor parity;	Rt<=Ra XOR Rb	xor Rt, Ra, Rb	01001

Convert the program into assembler and then to machine language (in binary and hex notation), using the instruction set and coding of architecture B and the additional instructions you have defined. [10 marks]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OpCode					Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Rb			Φ	Φ	Φ	Φ	Φ	Ra			Φ	Φ	Φ	Φ	Φ	Rt		
and Rt, Ra, Rb; xor Rt, Ra, Rb;																															
OpCode					Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Ra			Φ	Φ	Φ	Φ	Φ	Rt		
loadr Rt, Ra; dec Rt, Ra; move Rt, Ra;																															
OpCode					Φ	Φ	Φ	Φ	Φ	Φ	Φ	Shift N				Φ	Φ	Φ	Φ	Φ	Ra			Φ	Φ	Φ	Φ	Φ	Rt		
shr Rt, Ra, n;																															
OpCode					Φ	Φ	Φ	Immediate																Φ	Φ	Φ	Φ	Φ	Rt		
move Rt, imm;																															
OpCode					offset											Φ	Φ	Φ	Φ	Φ	Ra			Φ	Φ	Φ	Φ	Φ	Rs		
storo Rs, Ra, off;																															
OpCode					offset											Φ	Φ	Φ	Φ	Φ	Ra			Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ
br Ra, cond, off;																															

Exam No : Y3874734

instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	machine language in hex	
loadr R2, R7	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	A0000702	Rt <= DMEM[Ra] {register indirect addressing}
move R3, #0000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	80000003	Rt <= imm
move R4, R1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	88000104	Rt <= Ra
move R5, #0001	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	80000105	Rt <= imm
br R4, R4 = 0, 6	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	C0060400	If condition then jump to IMEM[PC+off], else continue
and R6, R2, R5	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	60050206	Rt<=Ra & Rb
xor R3, R6, R3	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	68030603	Rt<=Ra XOR Rb
shr R2, R2,1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	48010202	Rt <= Ra shifted right by n bits
dec R4, R4	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	48010202	Rt <= Ra – 1
br R4, R4 ≠ 0, -4	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	CFFC0400	If condition then jump to IMEM[PC+off], else continue
storo R7, R3, 2	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	B8020703	DMEM[Ra+off] <= Rs {base plus offset addressing}

*Brown = new Instructions

Exam No : Y3874734

For every instruction within the program, define the values of all the necessary control signals for the execution of the program on architecture B. Identify “don’t care” values with ‘Φ’. [10 marks]

	RA	RB	WA	MA (Hex)	IMM (Hex)	OEN	S1,2,3	AL	SH	WEN
loadr R2, R7	111	000	010	X’ΦΦΦΦ	X’ΦΦΦΦ	0	001	101	ΦΦ0000	1
move R3, #0000	000	ΦΦΦ	011	X’ΦΦΦΦ	X’0000	0	1Φ0	101	ΦΦ0000	1
move R4, R1	001	000	100	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0Φ0	101	ΦΦ0000	1
move R5, #0001	000	ΦΦΦ	101	X’ΦΦΦΦ	X’0001	0	1Φ0	101	ΦΦ0000	1
br R4, R4 = 0, 6	100	000	ΦΦΦ	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0ΦΦ	101	ΦΦΦΦΦΦ	0
and R6, R2, R5	010	101	110	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0Φ0	001	ΦΦ0000	1
xor R3, R6, R3	110	011	011	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0Φ0	010	ΦΦ0000	1
shr R2, R2,1	010	000	010	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0Φ0	101	100001	1
dec R4, R4	100	ΦΦΦ	100	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0Φ0	100	ΦΦ0000	1
br R4, R4 ≠ 0, -4	100	000	ΦΦΦ	X’ΦΦΦΦ	X’ΦΦΦΦ	0	0ΦΦ	101	ΦΦΦΦΦΦ	0
storo R7, R3, 2	111	011	ΦΦΦ	X’ΦΦΦΦ	X’0002	1	10Φ	101	ΦΦ0000	0

*Brown = new Instructions

Exam No : Y3874734

Q2)

Using the multi-cycle architecture C, define (in a table) the operations to be carried out and the values of all the control signals required for each cycle in the execution of the following instructions. Identify "don't care" values. [4 marks each]:

[shift the contents of R1 right by 11 bits and store the result in R3]

SHR R3, R1, x'b;	RA[2:0]	RB[2:0]	WA[2:0]	MA[15:0]	IMM[15:0]	OEN	S[1:4]	AL[2:0]	SH[5:0]	WEN
S1 - Fetch	ØØØ	ØØØ	ØØØ	x'ØØØØ	x'ØØØØ	0	Ø1ØØ	111	ØØ0000	0
S2 - Register Read	001	000	ØØØ	x'ØØØØ	x'ØØØØ	0	ØØØØ	ØØØ	ØØØØØØ	0
S3 - ALU	ØØØ	ØØØ	ØØØ	x'ØØØØ	x'ØØØØ	0	00ØØ	101	10,1011	0
S4 - Memory Read/Write										
S5 - Register Write	ØØØ	ØØØ	011	x'ØØØØ	x'ØØØØ	0	ØØØ0	ØØØ	ØØØØØØ	1

[load into R5 the contents of the memory at address AF1F]

loadi R5, x'af1f;	RA[2:0]	RB[2:0]	WA[2:0]	MA[15:0]	IMM[15:0]	OEN	S[1:4]	AL[2:0]	SH[5:0]	WEN
S1 - Fetch	ØØØ	ØØØ	ØØØ	x'ØØØØ	x'ØØØØ	0	Ø1ØØ	111	ØØ0000	0
S2 - Register Read										
S3 - ALU										
S4 - Memory Read/Write	ØØØ	ØØØ	ØØØ	x'AF1F	x'ØØØØ	0	ØØ1Ø	ØØØ	ØØØØØØ	0
S5 - Register Write	ØØØ	ØØØ	101	x'ØØØØ	x'ØØØØ	0	ØØØ1	ØØØ	ØØØØØØ	1

[if R3 is not equal to 0, then jump to PC+11A, else continue]

brneq R3, x'11A	RA[2:0]	RB[2:0]	WA[2:0]	MA[15:0]	IMM[15:0]	OEN	S[1:4]	AL[2:0]	SH[5:0]	WEN
S1 - Fetch	ØØØ	ØØØ	ØØØ	x'ØØØØ	x'ØØØØ	0	Ø1ØØ	111	ØØ0000	0
S2 - Register Read	011	000	ØØØ	x'ØØØØ	x'011A	0	11ØØ	101	ØØ0000	0
S3 - ALU	ØØØ	ØØØ	ØØØ	x'ØØØØ	x'ØØØØ	0	00ØØ	101	ØØ0000	0
S4 - Memory Read/Write										
S5 - Register Write										

Exam No : Y3874734

Q3)

Determine a coding for each of the instructions in the set. Assign OPCODE values (6 bits) and specify the position of every field within each kind of instruction. [14 marks]

	AR/LO/TR /CO		CONTROLS THE REST OF THE OPERATION				5 BIT RT ADDRESS VALUE					5 BIT RA ADDRESS VALUE & 5 BITS OF IMMEADIATE VALUES					16 BIT IMMEDIATE VALUES x 2, RB VALUES,SHIFTED BY N BIT VALUES, ADDRESS OFF SET VALUES, PROGRAM COUNTEROFFSET VALUES, FLAGS																
NO OPERATION	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	nop (no op code) = 000000
ARITHMATIC	0	0	1	0	0	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					rt <= ra + rb
	0	0	0	0	0	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					rt <= ra – rb
	0	0	1	0	1	0	5 bit RT address value					5 bit RA address value					16 bit immediate value number between 65535-0																rt <= ra + immediate value
	0	0	0	0	1	0	5 bit RT address value					5 bit RA address value					16 bit immediate value number between 65535-0																rt <= ra – immediate value
	0	0	1	0	1	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	rt <= ra + 1
	0	0	0	0	1	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	rt <= ra – 1
LOGIC	0	1	0	0	0	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					rt <= ra AND rb
	0	1	0	0	0	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					rt <= ra OR rb
	0	1	0	0	1	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					rt <= ra XOR rb
	0	1	0	0	1	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	rt <= NOT ra
	0	1	0	1	0	0	5 bit RT address value					5 bit RA address value					16 bit immediate value number between 65535-0																rt <= ra AND immediate value
	0	1	0	1	0	1	5 bit RT address value					5 bit RA address value					16 bit immediate value number between 65535-0																rt <= ra OR immediate value
	0	1	0	1	1	0	5 bit RT address value					5 bit RA address value					16 bit immediate value number between 65535-0																rt <= ra XOR immediate value
	0	1	1	0	0	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	Binary N bit value				0	0	0	0	0	rt <= ra shifted left (LS)by n bits
	0	1	1	0	0	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	Binary N bit value				0	0	0	0	0	rt <= ra shifted right (RS) by n bits
	0	1	1	0	1	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	Binary N bit value				0	0	0	0	0	rt <= ra rotated left (LR) by n bits
	0	1	1	0	1	1	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	Binary N bit value				0	0	0	0	0	rt <= ra rotated right (RR)by n bits
TRANSFER	1	0	0	0	0	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	rt <= ra
	1	0	0	0	0	1	5 bit RT address value					16 bit immediate value number should be between 0-511 dec for address range unused bits should be 0s																0	0	0	0	0	rt <= DMEM[imm] {direct addressing}
	1	0	0	0	1	0	5 bit RT address value					5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	rt <= DMEM[ra] {register indirect addressing}
	1	0	0	1	0	0	5 bit RT address value					5 bit RA address value					10 bit address value **offset** +- number 0-511										0	0	0	0	0	0	rt <= DMEM[ra+off] {base plus offset addressing}
	1	0	1	0	0	1	0	0	0	0	0	16 bit immediate value number should be between 0-511 dec for address range unused bits should be 0s																5 bit RB address value					DMEM[imm] <= rb {direct addressing}
	1	0	1	0	1	0	0	0	0	0	0	5 bit RA address value					0	0	0	0	0	0	0	0	0	0	0	5 bit RB address value					DMEM[ra] <= rb {register indirect addressing}
	1	0	1	1	0	0	0	0	0	0	0	5 bit RA address value					10 bit address value **offset +-** number 0-511										0	5 bit RB address value					DMEM[ra+off] <= rb {base plus offset addressing}
CONTROL	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9 bit Program Counter **offset** +- value for PC + offset value									0	0	0	0	0	0	0	jump control opcode and + - 9 bits for PC+offset
	1	1	0	0	0	1	0	0	0	0	0	5 bit RA address value					9 bit Program Counter **offset +-** value for PC + offset value									0	0	0	FLAGS				4 bits for flag(cond), + - 9 bits or (PC) offset, 5 bits for RA,