Question No. 1

9(a)	`LDM #500`: Immediate 500 `LDD 500`: Direct 100 `LDI 500`: Indirect 20	3
9(b)		7

Label	Instruction	
	Opcode	Operand
	`LDM`	`#20`
	`STO`	`Twenty`
	`LDI`	`Y`
	`ADD`	`Twenty`
	`STO`	`Z`
`Twenty:`	`#20`	
`Y:`		
`Z:`		

One mark for `LDM #20` seen
One mark for storing `20` at any address
One mark for labelling that address e.g. `Twenty` away from the program code
One mark for labelling addresses away from the program code as `Y` and `Z`
One mark for correct use of `LDI Y`
One mark for correct use of `STO Z`
One mark for correct use of `ADD` with labelled address

Question No. 2

START:	LDR	#0	// initialise index register to zero	[1]
	LDM	#0	// initialise COUNT to zero	[1]
	STO	COUNT		
LOOP1:	LDX	NAME	// load character from indexed address NAME	[1]
	OUT		// output character to screen	[1]
	INC	IX	// increment index register	[1]
	LDD	COUNT	// increment COUNT starts here	
	INC	ACC		[1]
	STO	COUNT		
	CMP	MAX	// is COUNT = MAX?	[1]
	JPN	LOOP1	// if FALSE, jump to LOOP1	[1]
REVERSE:	DEC	IX	// decrement index register	[1]
	LDM	#0	// set ACC to zero	[1]
	STO	COUNT	// store in COUNT	
LOOP2:	LDX	NAME	// load character from indexed address NAME	[1]
	OUT		// output character to screen	
	DEC	IX	// decrement index register	[1]
	LDD	COUNT	// increment COUNT starts here	
	INC	ACC	//	[1]
	STO	COUNT	//	
	CMP	MAX	// is COUNT = MAX?	[1]
	JPN	LOOP2	// if FALSE, jump to LOOP2	
	END		// end of program	[1]
COUNT:				
MAX:	4			
NAME:	B01000110		// ASCII code in binary for 'F'	
	B01010010		// ASCII code in binary for 'R'	
	B01000101		// ASCII code in binary for 'E'	
	B01000100		// ASCII code in binary for 'D'	

[Max 15]

Question No. 3

	Label	Op code	Operand	Comment		
1(a)						8
	START:	IN		// INPUT character		
		STO	CHAR	// store in CHAR	1	
		LDM	#65	// Initialise ACC (ASCII value for 'A' is 65)	1	
	LOOP:	OUT		// OUTPUT ACC	1 + 1	
		CMP	CHAR	// compare ACC with CHAR	1	
		JPE	ENDFOR	// if equal jump to end of FOR loop	1	
		INC	ACC	// increment ACC	1	
		JMP	LOOP	// jump to LOOP	1	
	ENDFOR:	END				
	CHAR:					

Question No. 4

Question	Label	Op code	Operand	Comment	Marks	Total
1(a)	START:	IN		// INPUT character	1	9
		STO	CHAR1	// store in CHAR1		
		IN		// INPUT character	1	
		STO	CHAR2	// store in CHAR2		
		LDD	CHAR1	// initialise ACC to ASCII value of CHAR1	1	
	LOOP:	OUT		//output contents of ACC	1+1	
		CMP	CHAR2	// compare ACC with CHAR2	1	
		JPE	ENDFOR	// if equal jump to end of FOR loop	1	
		INC	ACC	// increment ACC	1	
		JMP	LOOP	// jump to LOOP	1	
	ENDFOR:	END				
	CHAR1:					
	CHAR2:					

Question No. 5

	Label	Op code	Operand	Comment	Marks	
4(a)	START:	LDM	#63	// load ASCII value for '?'		11
		OUT		// OUTPUT '?'	1	
		IN		// input GUESS	1	
		CMP	LETTERTOGUESS	// compare with stored letter	1	
		JPE	GUESSED	// if correct guess, go to GUESSED	1	
		LDD	ATTEMPTS	// increment ATTEMPTS	1	
		INC	ACC		1	
		STO	ATTEMPTS		1	
		CMP	#9	// is ATTEMPTS = 9 ?	1	
		JPE	ENDP	// if out of guesses, go to ENDP	1	
		JMP	START	// go back to beginning of loop	1	
	GUESSED:	LDM	#42	// load ASCII for '*'		
		OUT		// OUTPUT '*'	1	
	ENDP:	END		// end program		
	ATTEMPTS:	0				
	LETTERTOGUESS:	'a'				

Question No. 6

5(a)	Max 10	10

Label	Op code	Operand	Comment	Marks
START:	LDR	#0	// initialise Index Register	
LOOP:	LDX	LETTERS	// load LETTERS	1
	CMP	LETTERTOFIND	// is LETTERS = LETTERTOFIND ?	1
	JPN	NOTFOUND	// if not, go to NOTFOUND	1
	LDD	FOUND		1
	INC	ACC	// increment FOUND	1
	STO	FOUND		1
NOTFOUND:	LDD	COUNT		
	INC	ACC	//increment COUNT	1
	STO	COUNT		
	CMP	#6	// is COUNT = 6 ?	1
	JPE	ENDP	// if yes, end	1
	INC	IX	// increment Index Register	1
	JMP	LOOP	// go back to beginning of loop	1
ENDP:	END		// end program	
LETTERTOFIND:		'x'		
LETTERS:		'd'		
		'u'		
		'p'		
		'l'		
		'e'		
		'x'		
COUNT:		0		
FOUND:		0		

Question No. 7

1(b)	START:	LDD	NUMBER		1	7
		AND	MASK	// set to zero all bits except sign bit	1	
		CMP	#0	// compare with 0	1	
		JPN	ELSE	// if not equal jump to ELSE	1	
	THEN:	LDM	#80	// load ACC with 'P' (ASCII value 80)	1	
		JMP	ENDIF			
	ELSE:	LDM	#78	// load ACC with 'N' (ASCII value 78)		
	ENDIF:	OUT		//output character	1	
		END				
	NUMBER:	B00000101		// integer to be tested		
	MASK:	B10000000		// show value of mask in binary here	1	

Question No. 8

	Label	Op code	Operand	Comment		
1(b)	START:	LDD	NUMBER1		1	6
		XOR	MASK	// convert to one's complement	1	
		INC	ACC	// convert to two's complement	1	
		STO	NUMBER2		1	
		END				
	MASK:	B11111111		// show value of mask in binary here	1	
	NUMBER1:	B00000101		// positive integer		
	NUMBER2:	B11111011		// show value of negative equivalent	1	

Question No. 9

4(b) | 10

Label	Opcode	Operand	Comment	Mark
START:	LDR	#0	// initialise the Index Register	1
LOOP:	LDX	NUMBERS	// load the value from NUMBERS	1 (LOOP) + 1(LDX NUMBERS)
	LSL	#2	// multiply by 4	1 (LSL) + 1 (#2)
	STX	NUMBERS	// store the new value in NUMBERS	1
	INC	IX	// increment the Index Register	1
	LDD	COUNT		
	INC	ACC	// increment COUNT	1
	STO	COUNT		
	CMP	#5	// is COUNT = 5 ?	1
	JPN	LOOP	// repeat for next number	1
ENDP:	END			
COUNT:	0			
NUMBERS:	22			
	13			
	5			
	46			
	12			

Question No. 10

5(b) | 10

Label	Op Code	Operand		Comment
START:	LDR	#0	// initialise the Index Register	1
LOOP:	LDX	VALUES	// load the value from VALUES	1(loop) + 1(LDX Values)
	LSR	#3	// divide by 8	1 (LSR) + 1 (#3)
	STX	VALUES	// store the new value in VALUES	1
	INC	IX	// increment the Index Register	1
	LDD	REPS	// increment REPS	1
	INC	ACC		
	STO	REPS		
	CMP	#6	// is REPS = 6 ?	1
	JPN	LOOP	// repeat for next value	1
	END			
REPS:	0			
VALUES:	22			
	13			
	5			
	46			
	12			
	33			

Question No. 11

(a) (i) 1 mark per bullet to max 2: [2]

- `11011111`
- `AND`

(ii) 1 mark per bullet to max 2: [2]

- `00100000`
- `OR`

(b) 1 mark per line

START:	LDR	#0	// initialise index register to zero	1
	LDX	WORD	// get first character of WORD	1
	AND	MASK1	// ensure it is in upper case using MASK1	1
	OUT		// output character to screen	
	INC	IX	// increment index register	1
	LDM	#1	// load 1 into ACC	1
	STO	COUNT	// store in COUNT	1
LOOP:	LDX	WORD	// load next character from indexed address WORD	1
	OR	MASK2	// make lower case using MASK2	1
	OUT		// output character to screen	
	LDD	COUNT	// increment COUNT	1
	INC	ACC	//	
	STO	COUNT	//	
	CMP	LENGTH	// is COUNT = LENGTH?	1
	JPN	LOOP	// if FALSE – jump to LOOP	1
	END		// end of program	1
COUNT:	0			
MASK1:	B11011111		// bit pattern for upper case	1
MASK2:	B00100000		// bit pattern for lower case	
LENGTH:	4			
WORD:	B01100110		//ASCII code in binary for 'f'	
	B01101000		//ASCII code in binary for 'r'	
	B01000101		//ASCII code in binary for 'E'	
	B01000100		//ASCII code in binary for 'D'	

[max 12]