NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

NUST BALOCHISTAN CAMPUS(NBC)

SUBMITTED TO: **MR. NAJEEB ULLAH**

SUBMITTED BY: **MALIK MOIZ ASGHAR (343195)**

DATE: 10-06-2022

SUBJECT: Computer Architecture and Organisation

Q.**1: A given microprocessor has words of 1 byte. What is the smallest and largest integer that can be represented in the following representations?**

a. Unsigned

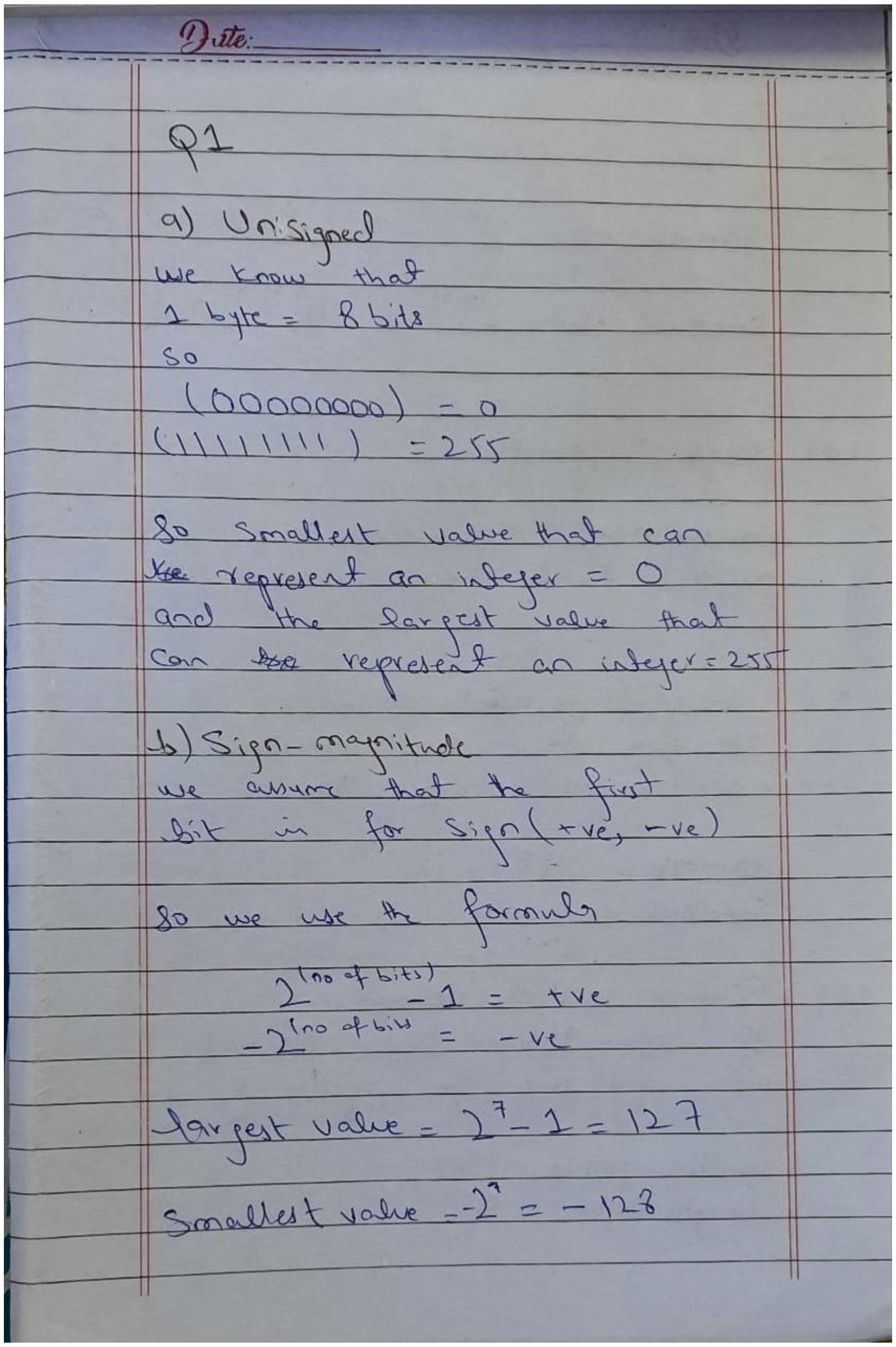
b. Sign-magnitude

c. Ones complement

d. Twos complement

e. Unsigned packed decimal

f. Signed packed decimal



Date:

Q1

a) Unsigned

we know that

1 byte = 8 bits

So

(00000000) = 0

(11111111) = 255

So smallest value that can ~~be~~ represent an integer = 0

and the largest value that can ~~be~~ represent an integer = 255

b) Sign-magnitude

we assume that the first bit is for sign (+ve, -ve)

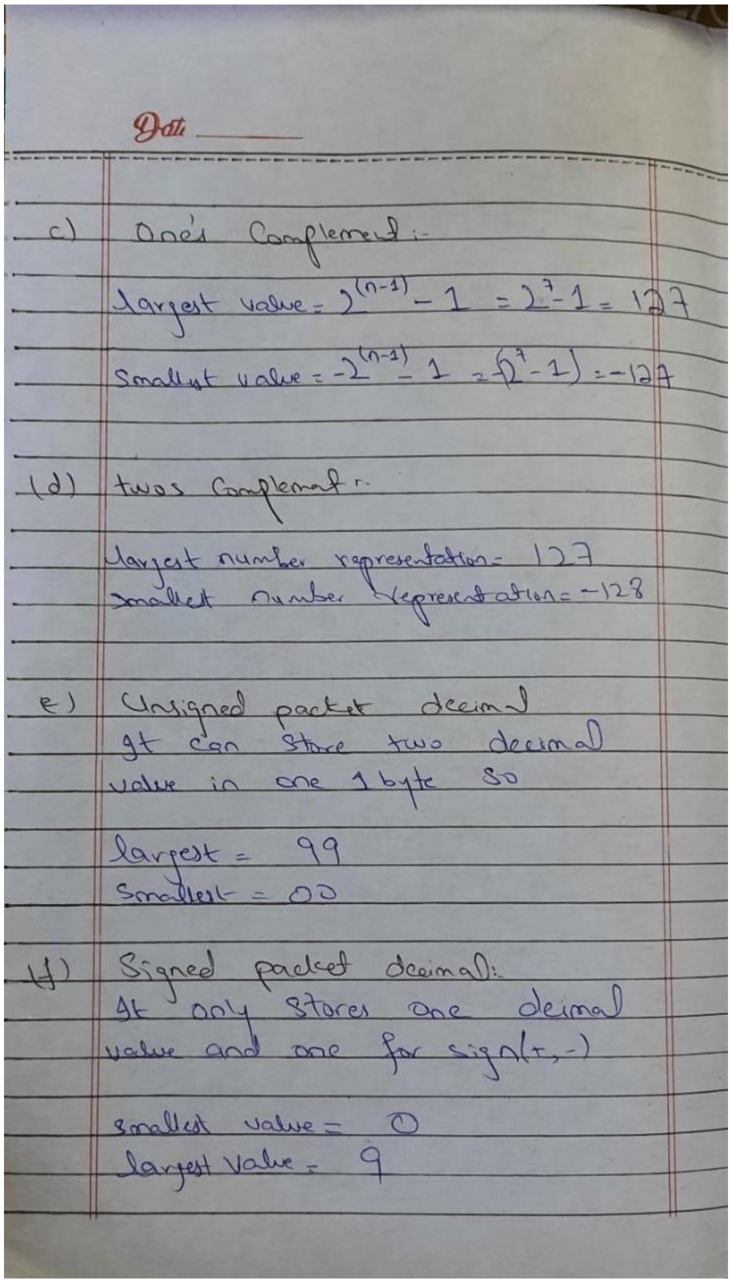
So we use the formula

$2^{(\text{no of bits})} - 1 = +ve$

$-2^{(\text{no of bits})} = -ve$

largest value = $2^7 - 1 = 127$

Smallest value = $-2^7 = -128$



Date

c) One's Complement :-

largest value = $2^{(n-1)} - 1 = 2^7 - 1 = 127$

Smallest value = $-2^{(n-1)} - 1 = -(2^7 - 1) = -127$

(d) twos Complement :-

largest number representation = 127
smallest number representation = -128

e) Unsigned packet decimal
It can store two decimal
value in one 1 byte so

largest = 99
Smallest = 00

(f) Signed packet decimal:
It only stores one deimal
value and one for sign(+,-)

smallest value = 0
largest value = 9

Q.2: Consider a hypothetical computer with an instruction set of only two n-bit instructions. The first bit specifies the

opcode, and the remaining bits specify one of the 2

n−1 n-bit words of main memory.

The two instructions are as follows:

I. SUBS X Subtract the contents of location X from the accumulator, and store the result in location X and the

accumulator.

II. JUMP X Place address X in the program counter.

A word in main memory may contain either an instruction or a binary number in twos complement notation. Demonstrate

that this instruction repertoire is reasonably complete by specifying how the following operations can be programmed:

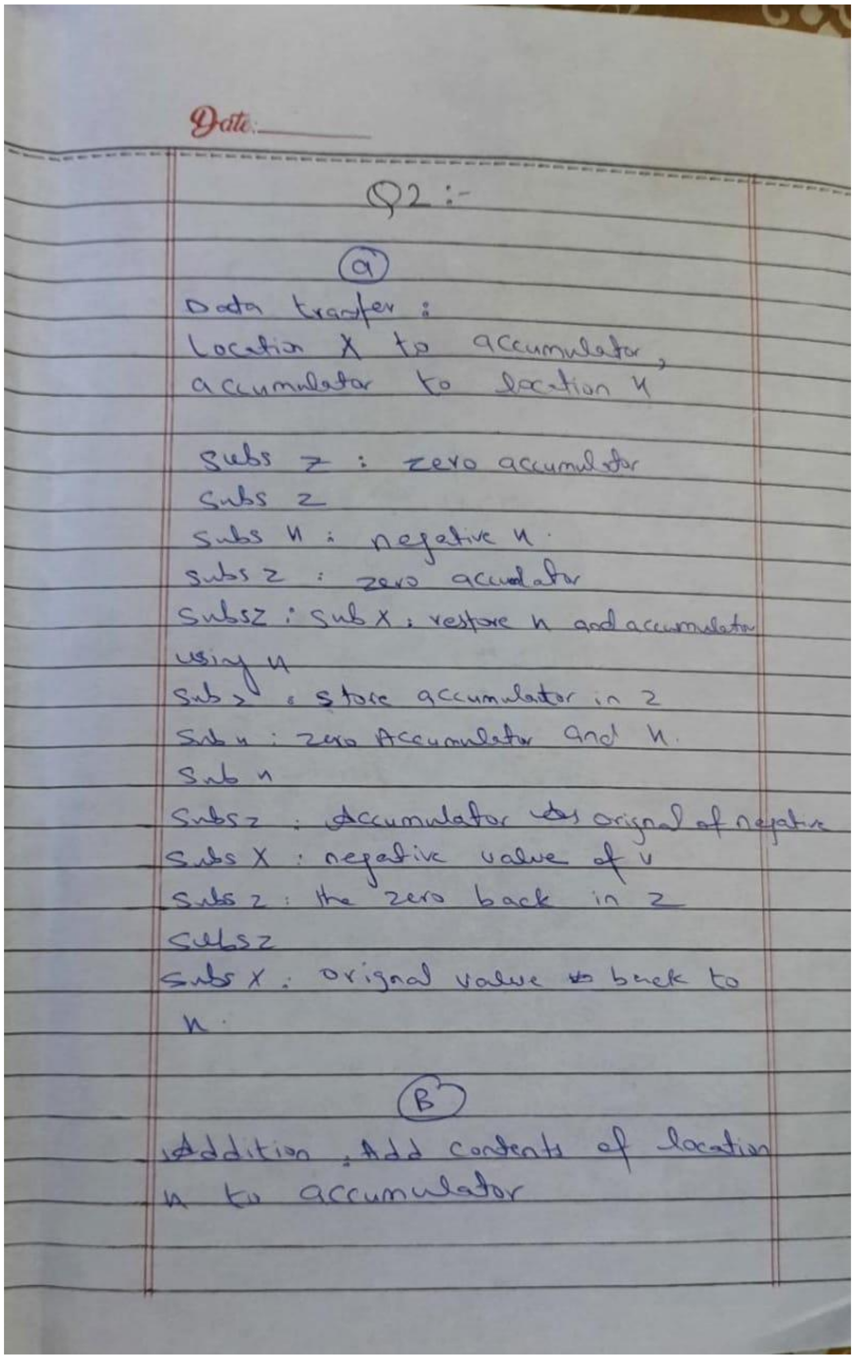
a) Data transfer: Location X to accumulator, accumulator to location X

b) Addition: Add contents of location X to accumulator

c) Conditional branch

d) Logical OR

e) I/O Operations



Date:

Q2 :-

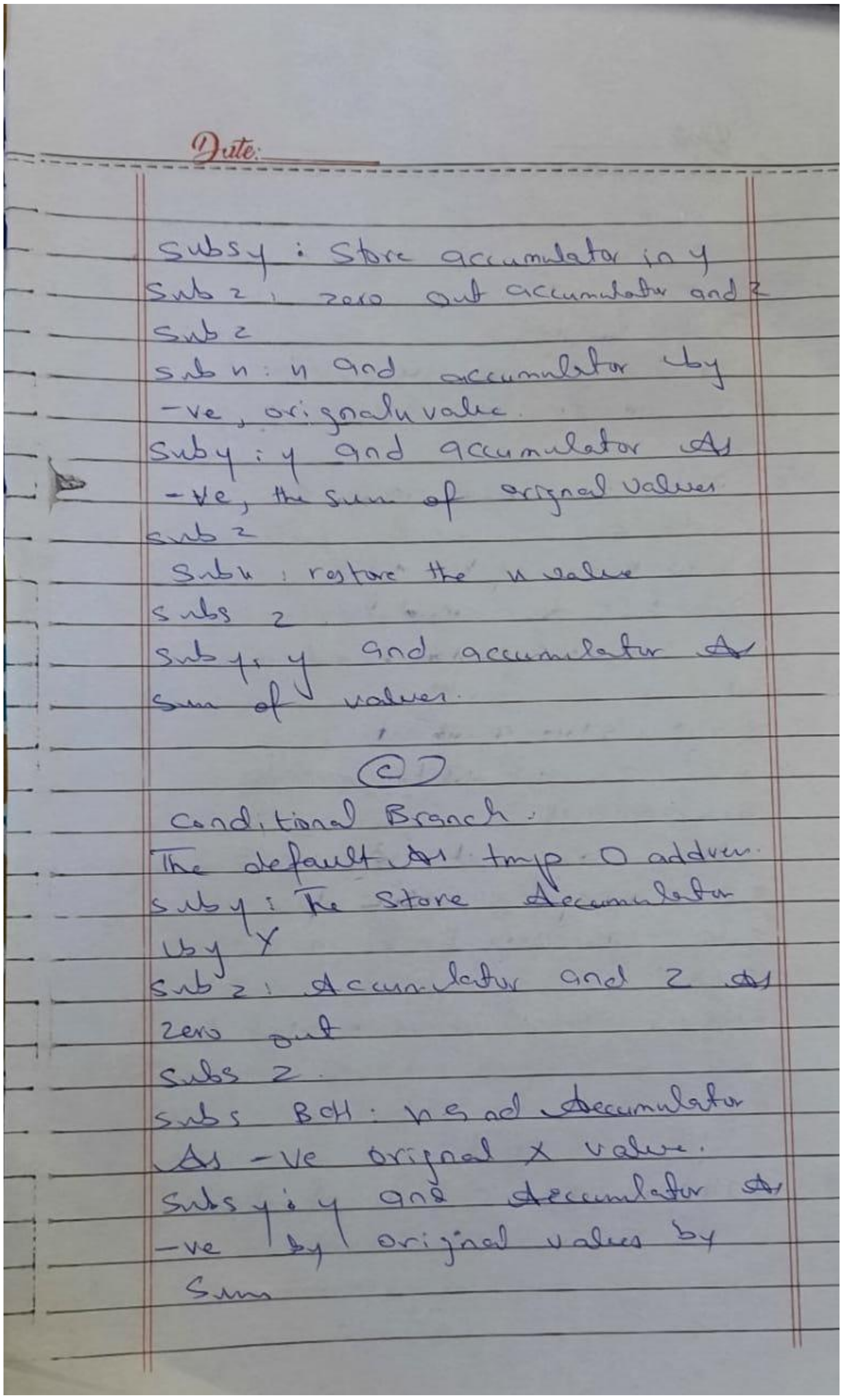
(a)

Data transfer :
Location X to accumulator,
accumulator to location y

Subs z : zero accumulator
Subs z
Subs y : negative y.
subs z : zero accumulator
SubsZ : Sub X : restore y and accumulator
using y
Sub z : store accumulator in z
Sub y : zero Accumulator and y.
Sub y
Subs z : Accumulator ~~[illegible]~~ orignal of negative
Subs X : negative value of v
Subs z : the zero back in z
Subs z
Subs X : orignal value ~~[illegible]~~ back to
y.

(B)

Addition : Add contents of location
y to accumulator

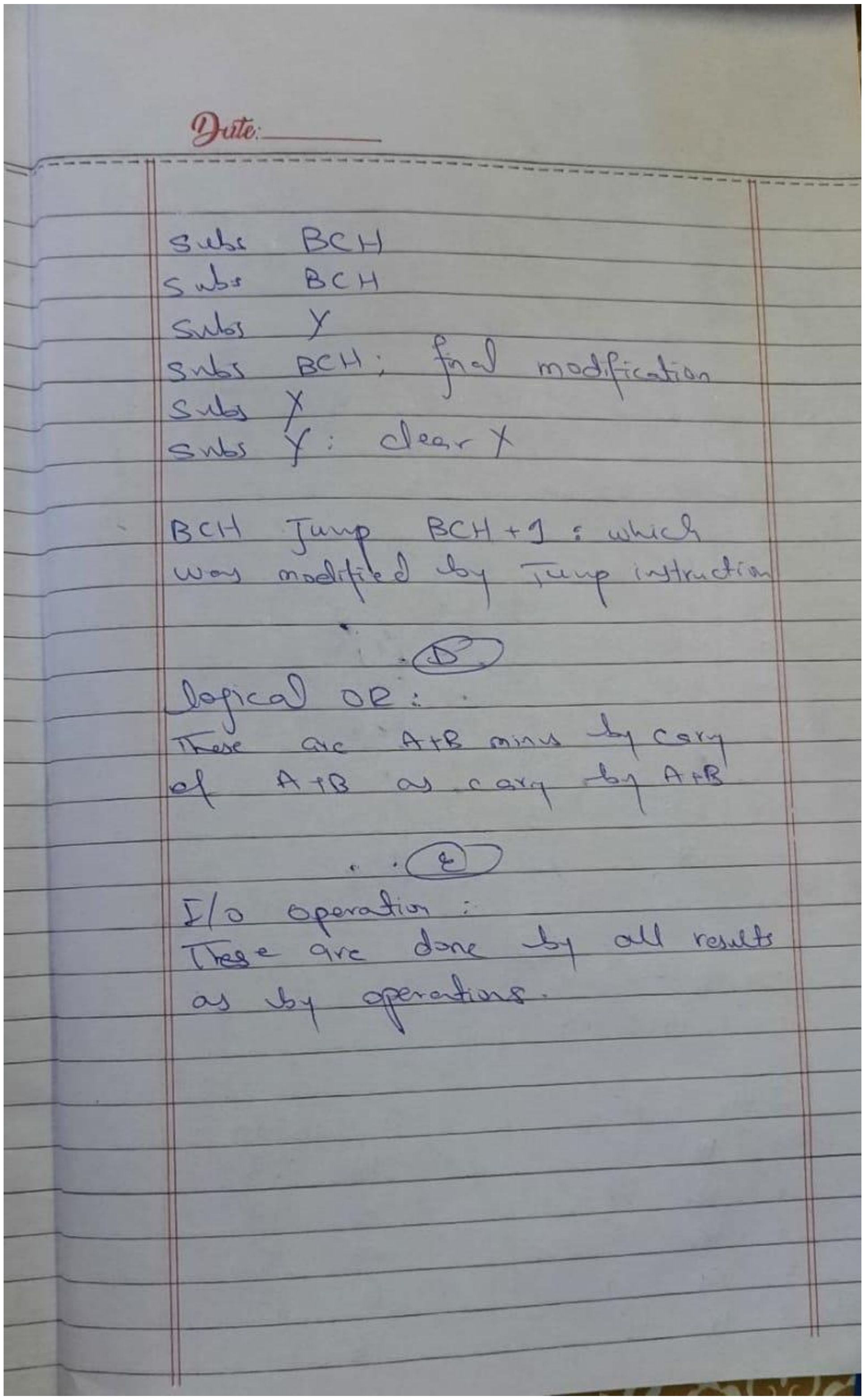


Date:

subsy : Store accumulator in y
Sub z : zero out accumulator and z
Sub z
sub n : n and accumulator by
-ve, original value.
Suby : y and accumulator As
-ve, the sum of original values
sub z
Subx : restore the x value
subs z
suby : y and accumulator As
sum of values.

(c)

Conditional Branch :
The default As tmp 0 address.
suby : The store Accumulator
by X
sub z : Accumulator and z As
zero out
subs z.
subs BCH : x and Accumulator
As -ve original x value.
subs y : y and Accumulator As
-ve by original values by
Sum


Date:

Subs BCH
Subs BCH
Subs Y
Subs BCH ; final modification
Subs X
Subs Y : clear X

BCH Jump BCH + 1 : which was modified by Jump instruction

(D)

logical OR :
These are A+B minus by cary of A+B as cary by A+B

(E)

I/o operation :
These are done by all results as by operations.

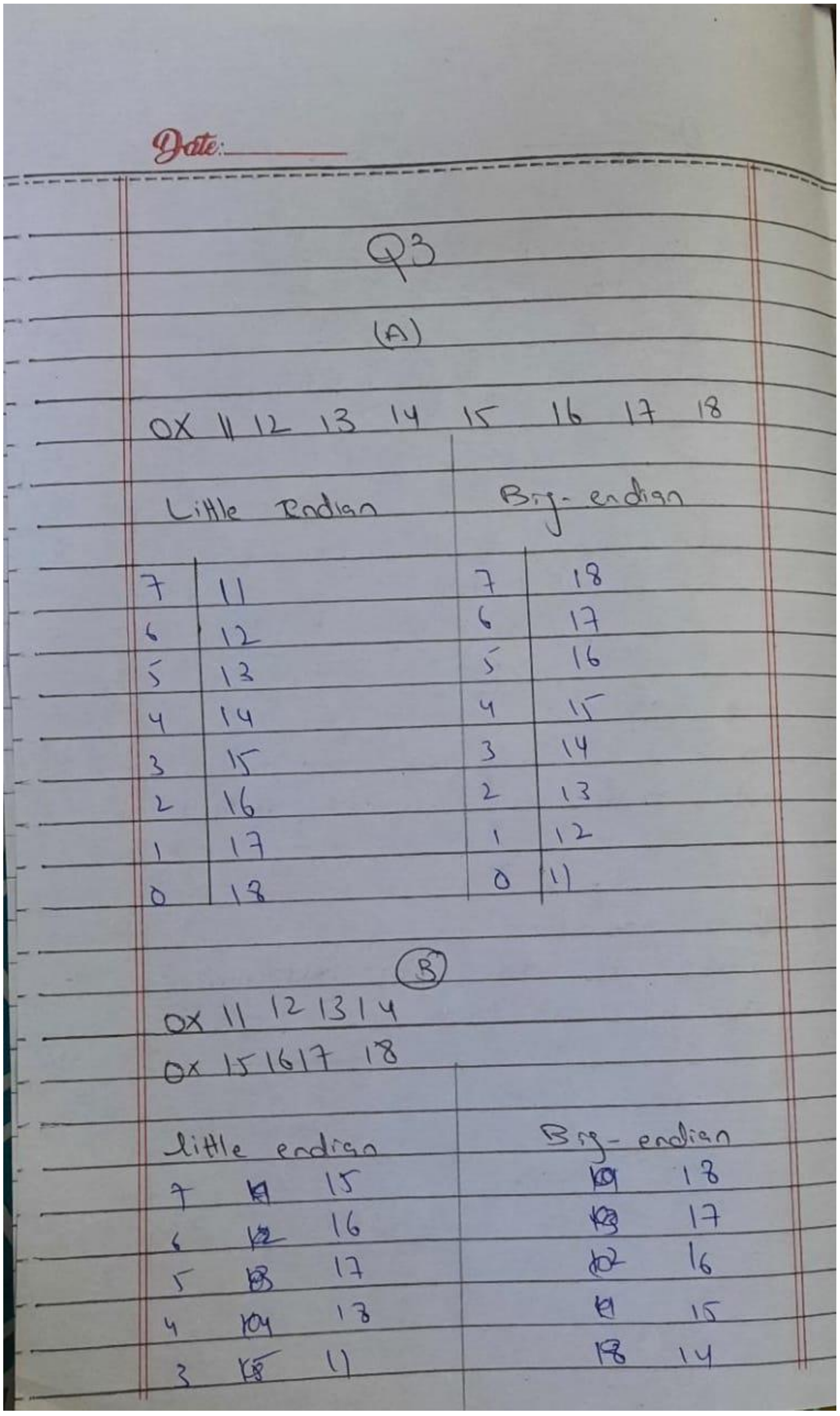
Q.3: For the following data structures, draw the big-endian and little-endian layouts, using the format of Figure given in the

lecture slides, and comment on the results.

```
A. struct {
double i; //0x1112131415161718
} s1;
```

```
B. struct {
int i; //0x11121314
int j; //0x15161718
} s2
```

```
C. struct {
short i; //0x1112
short j; //0x1314
short k; //0x1516
short l; //0x1718
} s3;
```



Date:

Q3

(A)

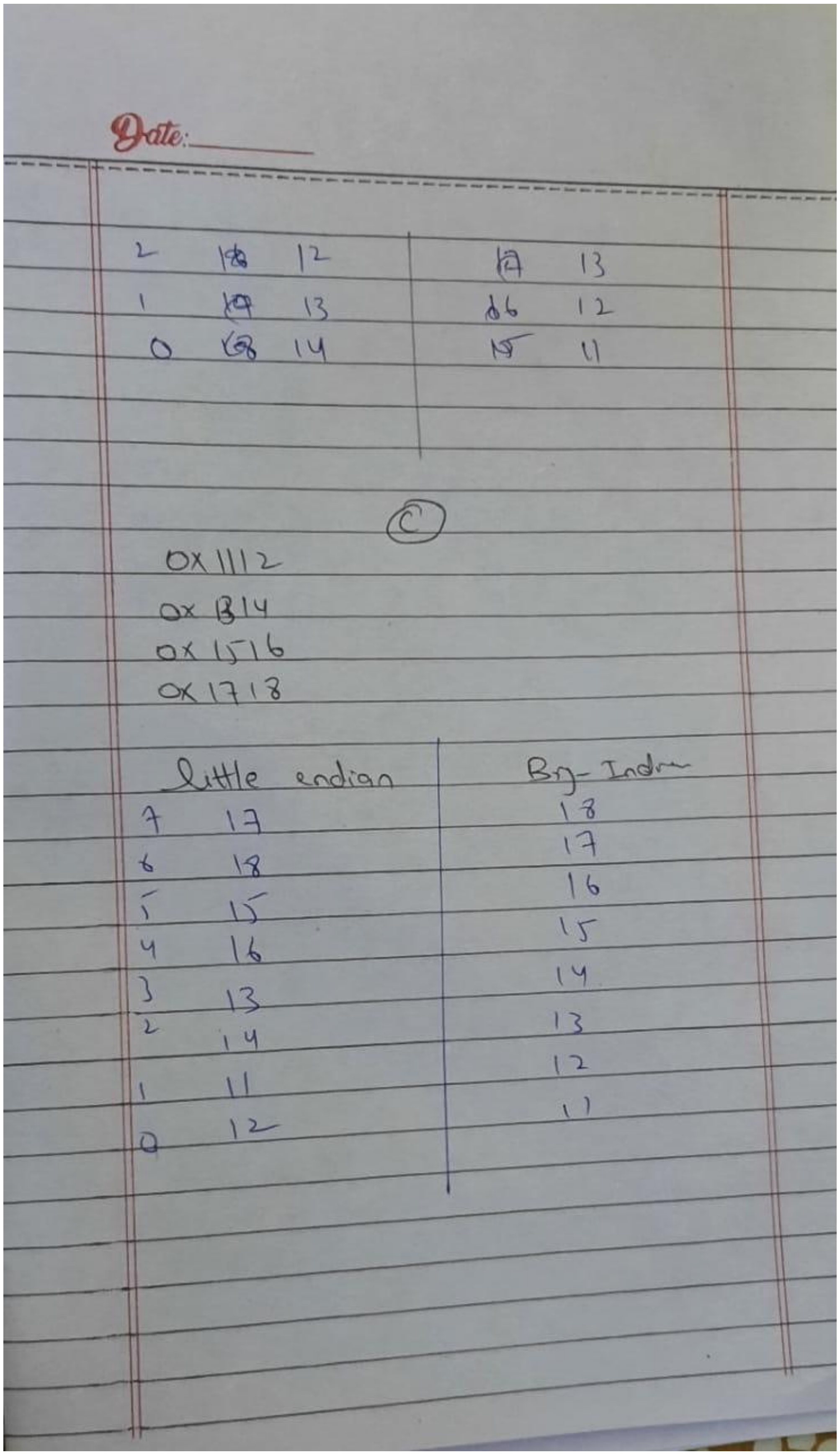
OX 11 12 13 14 15 16 17 18

Little Endian		Big-endian	
7	11	7	18
6	12	6	17
5	13	5	16
4	14	4	15
3	15	3	14
2	16	2	13
1	17	1	12
0	18	0	11

(B)

OX 11 12 13 14
OX 15 16 17 18

little endian			Big-endian	
7	~~11~~	15	~~14~~	18
6	~~12~~	16	~~13~~	17
5	~~13~~	17	~~12~~	16
4	~~14~~	18	~~11~~	15
3	~~15~~	11	~~18~~	14



Date:

2	[illegible]	12	17	13
1	[illegible]	13	16	12
0	[illegible]	14	15	11

(c)

0x1112
0x 1314
0x 1516
0x 1718

little endian		Big- Indian
7	17	18
6	18	17
5	15	16
4	16	15
3	13	14
2	14	13
1	11	12
0	12	11

Q**.4: Consider a 16-bit processor in which the following appears in main memory, starting at location 200:**

200 Load to AC Mode

201 500

202 Next Instruction

The first part of the first word indicates that this instruction loads a value into an accumulator. The Mode field specifies an

addressing mode and, if appropriate, indicates a source register; assume that when used, the source register is R1, which has

a value of 400. There is also a base register that contains the value 100. The value of 500 in location 201 may be part of the

address calculation. Assume that location 399 contains the value 999, location 400 contains the value 1000, and so on.

Determine the effective address and the operand to be loaded for the following address modes:

a. Direct

b. Immediate

c. Indirect

d. Register

e. Register indirect

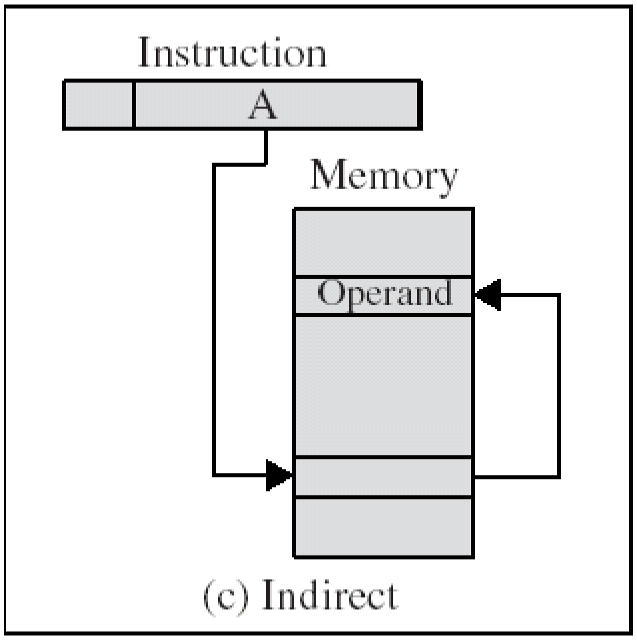
f. Displacement

Addressing mode	**Effective address**	**operand**	**Reason**
Direct addressing	500	1100	We know that in direct addressing there will be two

			addresses one address will contain the other address of operand so in this case we have the address of register 201 which contain the address of 500 finally when we reach 500 so we get the operand 1100
Immediate addressing	201	500	In immediate we know the register contain the value so in this case register 201 contain the operand 500
Indirect addressing	1100	1700	In this case we have three addresses first we have 201 address then we jump to 500 and then we again jump to 1100 which is our effective address and the value of 1100 is 1700
Register addressing	702	1302	Base address+next instruction address
Resgiter indirect	600	1200	R1+B1
Displacement	R1	400	Resgister address = effective address

Q.5: How many times does the processor need to refer to memory when it fetches and executes an indirect-address-mode

instruction if the instruction is (a) a computation requiring a single operand; (b) a branch?



(c) Indirect

a) If we have a single operand 3 times the processor needs to refer to memory first for fetch instruction, then fetch operand reference and last for fetching operand
b) If we have a branch instruction 2 times the processor needs to refer to memory first for fetch instruction, then for fetching the operand

Q.6: Design a variable-length opcode to allow all of the following to be encoded in a 36-bit instruction:

1. instructions with two 15-bit addresses and one 3-bit register number

The total we have 36-bit instruction addresses and each instruction contains 15-bit address we have 2 15-bit addresses and a 3-bit register address 30+3=33 the remaining 3-bit address will use for opcode

2. instructions with one 15-bit address and one 3-bit register number

In this case, we have one 15-bit address and 1 3bit register so a total of 15+3= 18

So reaming 18-bit address will use for opcode

3. instructions with no addresses or registers

If the instruction has no address so we have many choices in defining the opcode