Name: Aryan Unhale Class: TE Div.: B Roll No.: 48
Subject: SPCC Topic: Assignment 1 Date: 14/3 Page No.:

Q1) Difference between application program & system program.

	Application program	System program.
a)	software designed to perform specific tasks for the user.	a) Software designed to manage & control comp. hardware.
b)	To help users perform tasks & solve specific problems.	b) To manage system resources & provide an environment for apps to run.
c)	Depends on system programs to function.	c) works independently.
d)	High level interaction with user.	d) Low-level interaction with hardware
e)	simple	e) complex.
e)	ex→ MS word etc.	ex→ OS, compilers etc.

Q2) short note on Editor & types of editors.

→ Editor is a software program used to create, modify & manage text files - In the context of SPCC, they are essential tools for writing source code, config files & docs. They provide features like insertion, deletion, copying, pasting, searching & replacing

Types→

a) Text editors
b) line editors.
c) Stream editors
d) Structure editors
e) word processors
f) source code editors
g) Hex editors.

Page No.
Date

Q3) Compare compiler & Interpreter.

	Compiler	Interpreter.
a)	Translates entire source code in machine before execution	a) Translates & executes code line-by-line or statement-by-statement.
b)	Faster execution speed.	b) Slower.
c)	Errors are detected & reported after entire code is analyzed.	c) Errors are detected & reported instantly.
d)	Generates an independent executable file.	d) No separate executable files
e)	Difficult to debug.	e) Easy.
f)	Requires more memory ex→ C, C++ etc.	f) Less memory required. ex→ python, PHP etc.

Q4) short note on software tools

- They are computer programs designed to assist developers & users in creating, maintaining, analyzing & improving software systems. These tools enhance productivity, improve software quality & simplify complex tasks.

Categories

a) Programming tools
b) Version control tools
c) Build tools.
d) Testing tools
e) Project management tools.
f) IDEs
g) Modeling & Design Tools.
h) Documentation tools.
i) Configuration management tools.

Name: _____ Class: _____ Div.: _____ Roll No.: _____

Subject: _____ Topic: _____ Date: _____ Page No.: _____

Q5) Discuss with ex. forward reference & how is it handled in assembler design.

→ A forward reference occurs when a program references a label or symbol before it is defined. Assemblers process code sequentially, so encountering an undefined symbol during the first pass presents a problem.

How its handled →

Pass 1 →

a) Symbol Table creation.

Scans the code & records all symbol definitions with their addresses. leaves undefined symbols as placeholders.

b) Location counter calculation.

Pass 2

a) code generation -

Replaces all symbolic references with their corresponding addresses from the symbol table.

b) Handling forward references.

Ex →

Symbol	Address.
BETA	1003
ALPHA	1004
NEXT	1005

generated machine code (After pass 2)

Instruction	Address	machine code.
LDA ALPHA	1000	20 1004
STA BETA	1001	30 1003
JMP NEXT	1002	40 1005
BETA RESW	1003	Reserved
ALPHA RESW 1	1004	Reserved
NEXT END	1005	End of program

Page No.
Date

Q8) Explain design of 2 pass assembler with flowchart & databases.

→ A 2 pass Assembler processes the source code in 2 separate passes to handle forward references & generate the machine code accurately.

Pass1 → creates symbol table & calculate addresses

Analysis phase

Tasks → 1) Read source code line-by-line.

2) generate sym table.

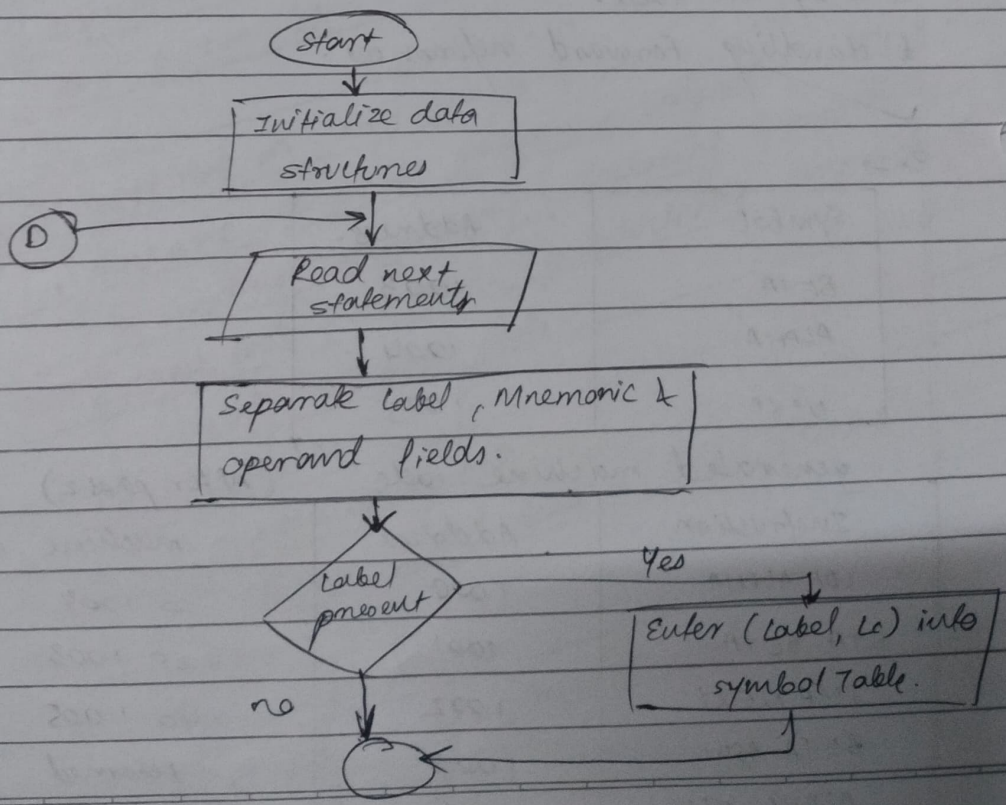
3) calculate addr.

DS → symbol Table, Operation Table.

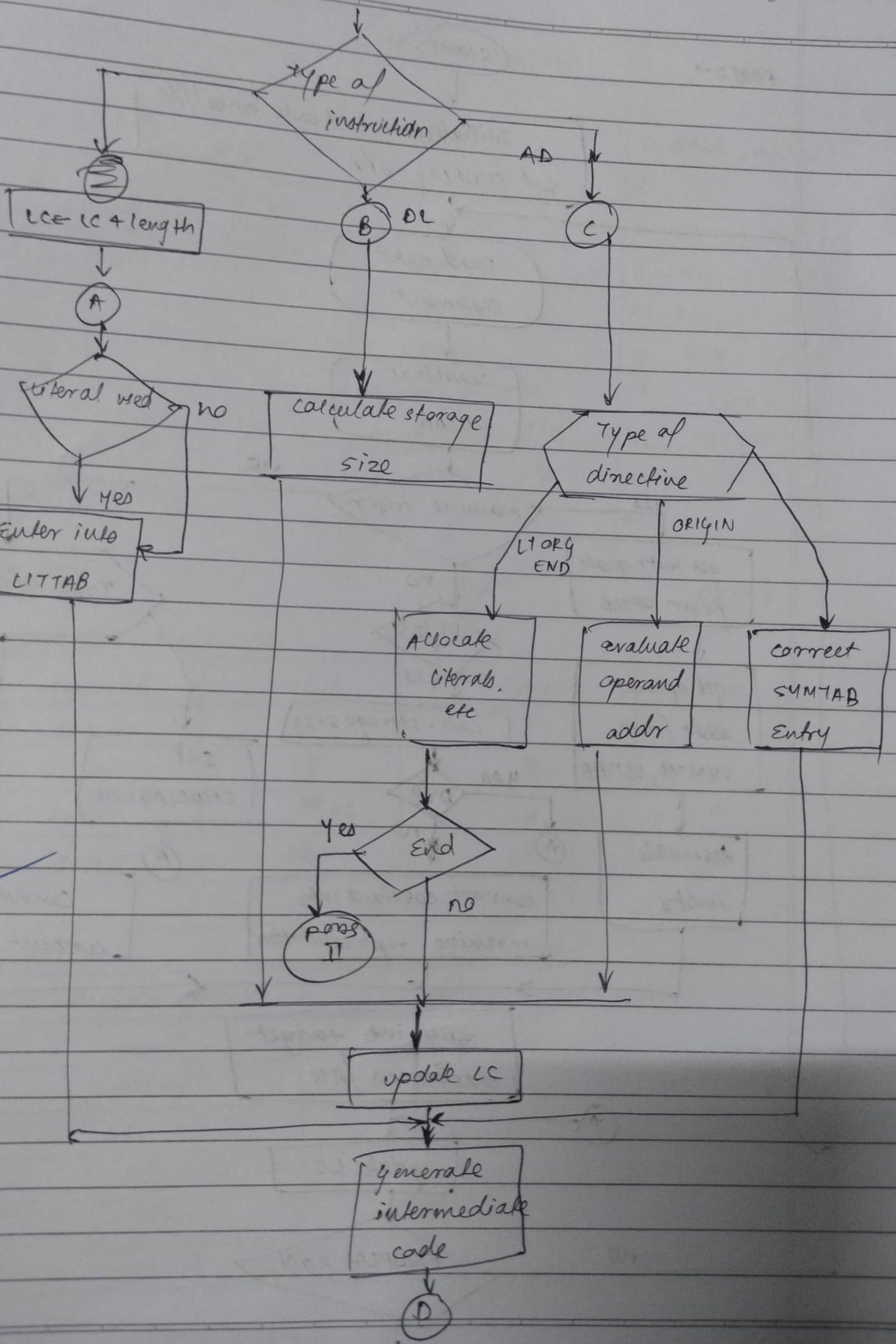
Sym Table after pass1.

symbol	Addr.
Beta	1003
Alpha	1004
Next	1005.

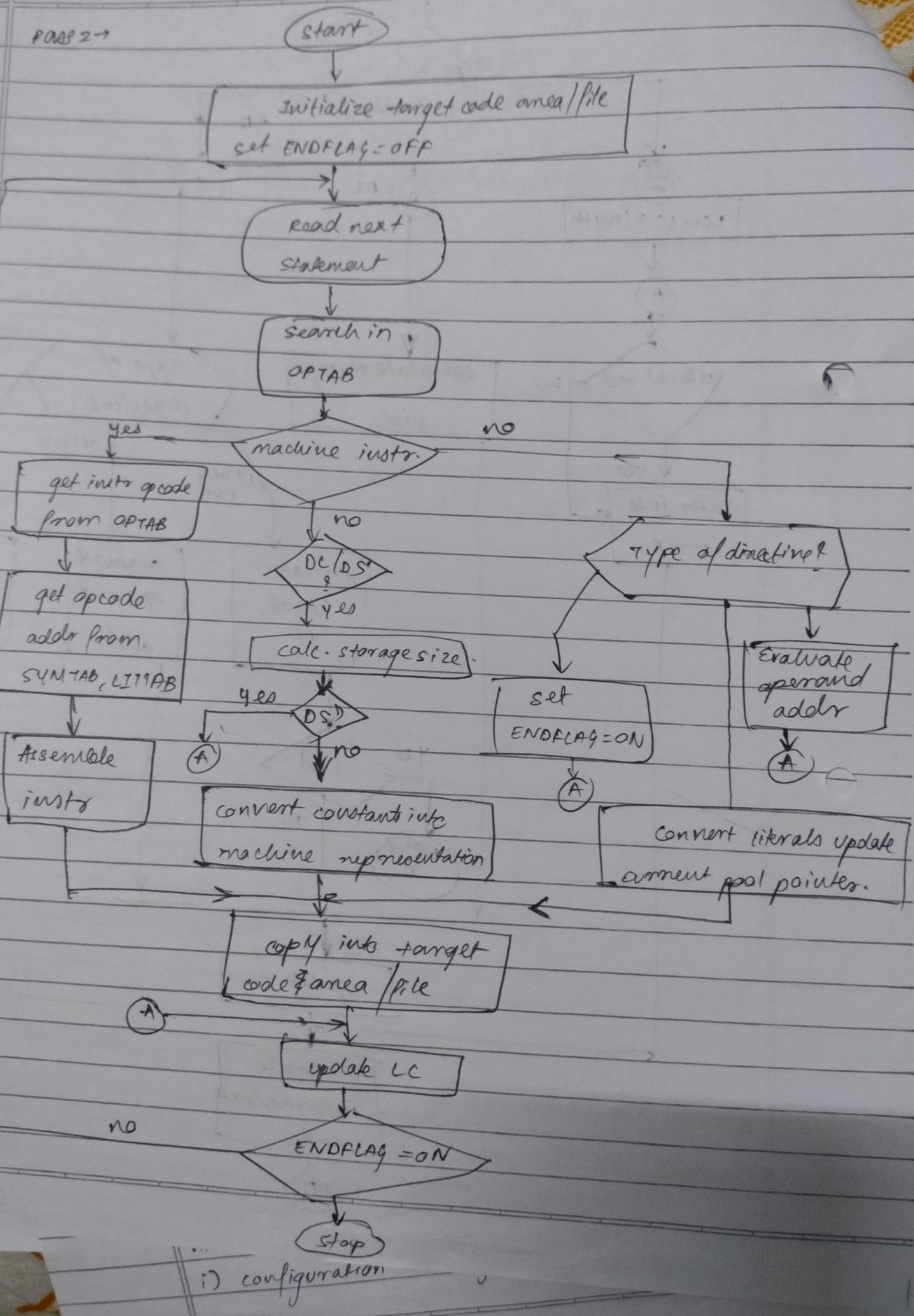
Pass1



Name : Class: Div. : Roll No.:
Subject : Class: Div. : Roll No.:
Topic : Date : Page No. :



PASS 2→



i) configuration

Name : _____ Roll No.: _____

Subject : _____ Class: _____ Div. : _____ Roll No.: _____

Topic : _____ Date : _____ Page No. : _____

	Location counter	LC
START 0	0	0
BEGIN BALR 15,0	0 BALR	0 BALR, 0(0,15)
USING *,15	2	2
L 3,F'34'	2 L3,F'34'	2 L3, 2(0,15)
A 3,OLDOH	6 A3, —	6 A3, 18(0,15)
S 3,RECPT	10 S3, —	10 S3, 22(0,15)
ST 3,ISSUE	14 ST3, —	14 ST3, 26(0,15)
OLDOH DC F'9'	18 OLDOH = F'9'	18 1001
RECPT DC F'4'	22 RECPT = F'4'	22 0100
ISSUE DS 1F	26 ISSUE = '1'F	26 —
END	30	30

Pass 1 Pass 2.

1) Symbol Table - (ST)

Symbol	value	length	A/R
Begin	0	1	R
OLDOH	18	4	R
RECPT	22	4	R
ISSUE	26	4	R

2) POT

Pseudo opcode	addr.
start	0
using	2
DC	18, 22
DS	26
END	30

3) Base Table (BT)

Register no.	Availability indicator	Contents of BR.
0	no	-
1	no	-
:	:	-
:	:	-
15	yes	0

4) MOT

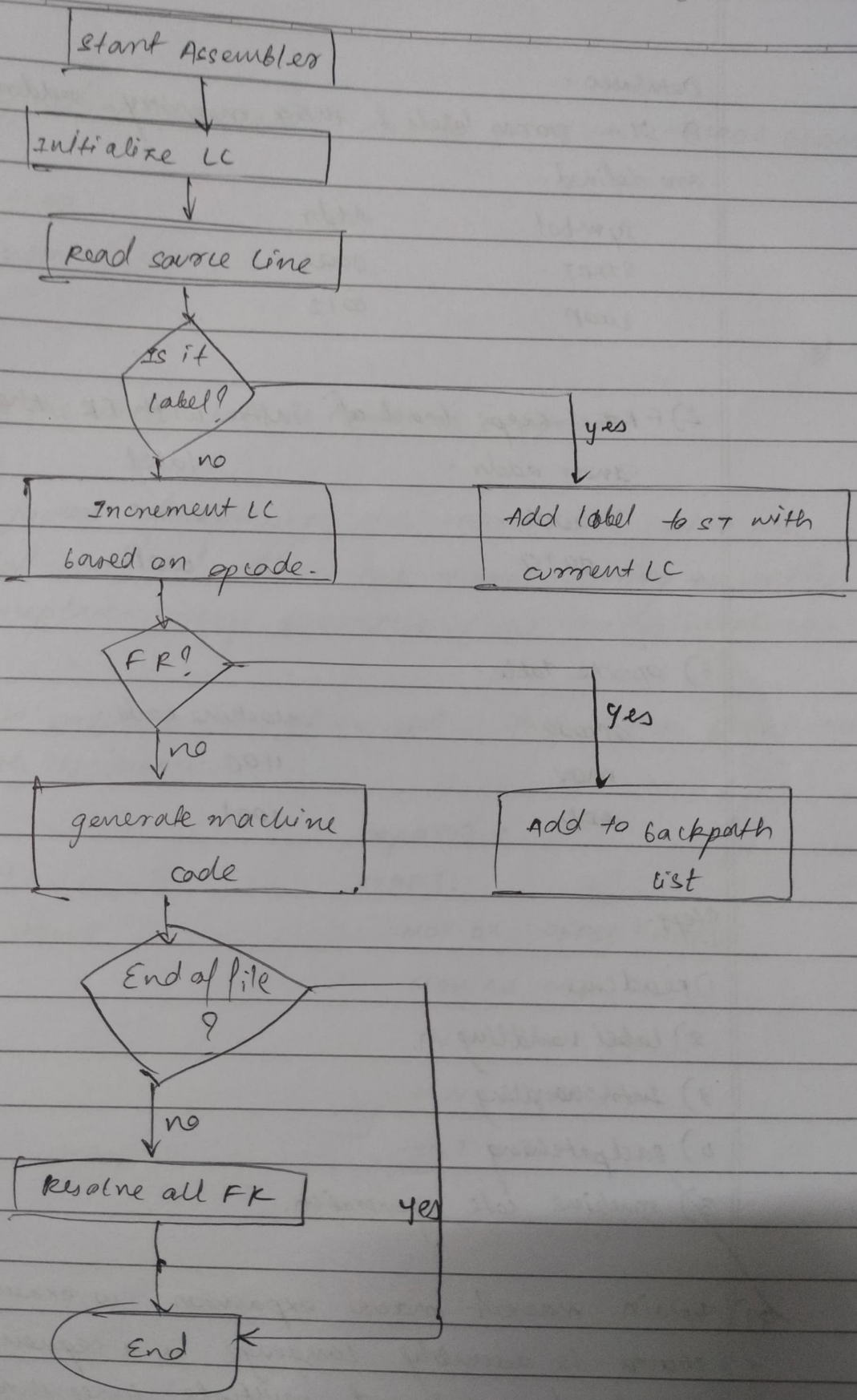
mnemonic opcode	Binary opcode	instr length	instr format
BALR	✓	2	RR
L	✓	4	RR
A	✓	4	RX
S	✓	4	RX
ST	✓	4	RX.

Q8) Explain design of single pass assembler.

→ A single pass assembler scans the source code only once to translate assembly instructions into machine code. It handles forward references using techniques like backpatching, where placeholders are used until labels or symbols are defined. The design reduces processing time but is more complex to implement compared to a 2-pass assembler.

Key tasks are

1) symbol Table management.
2) forward reference resolution.
3) Machine code generation.


start Assembler
Initialize LC
Read source line
Is it label?
yes
no
Add label to ST with current LC
Increment LC based on opcode.
FR?
yes
no
Add to backpath list
generate machine code
End of file ?
no
yes
Resolve all FR
End

Page No.
Date

Databases.

1) ST → stores labels & their memory addresses as soon as they are defined.

symbol	Addr
START	0000
LOOP	0012

2) FR → Keeps track of instr. with FR that need to be patched later.

Instr addr.	label
0004	START
0010	LOOP

3) opcode Table.

opcode	machine code.
MOV	1100
ADD	1001

Steps -

1) Read line
2) Label handling
3) Instr. handling
4) Backpatching
5) machine code generation.

Q9) Explain macro & macro expansion w example.

→ Macro is assembly language is a sequence of instructions that can be defined once & used multiple times throughout the program. Macros help reduce repititive code & improve readability by allowing the user to define reusable code blocks with a single name.

Name: Class: Div.: Roll No.:
Subject: Topic: Date: Page No.:

Definition -
The macro is defined with a label, usually using the MACRO & ENDM directives.

```
<macroname> MACRO
    ; Assembly instr.
ENDM.
```

Macro call → To use the macro, call its name in program.

Macro expansion →
It refers to the process of replacing the macro call with the actual sequence of instr. defined in the macro. During assembly the assembler replaces every occurence of the macro with the instr.
When the assembler processes the above code, the macro print_hello is replaced with its instr.

Eg →

```
PRINT_HELLO MACRO
    MOV DX, OFFSET HELLO
    MOV AH, 09h
    INT 21h
ENDM
START:
    PRINT_HELLO
    MOV AH, 4CH
    INT 21h
```

Expand →

```
START:
    MOV DX, OFFSET HELLO
    MOV AH, 09h
    INT 21h
    MOV AH, 4CH
    INT 21h
```

Q10) Features of macro →
→ a) Reusability →
Macro allows reusing a set of instructions multiple times by invoking a macro name.

b) Parameter passing →
Macros can accept parameters, allowing them to be flexible & adaptable to different use cases.
c) Text substitution -
when a macro is invoked, the macro processor replaces it with the corresponding set of instrs.
d) No runtime overhead
since macros are expanded at assembly time, there is no runtime overhead for macro calls.
e) Local label handling
some assemblers allow macros to use local labels, which prevent label conflicts when a macro is expanded multiple times.
f) conditional assembly
Macros may have conditional assembly capabilities using if, else & endif directives to control which parts of macro are expanded.

Q11) Draw fc & explain 2 pass macro processor with its db.
→ A 2 pass macro processor is a system that processes macros in 2 phases or passes.
- pass 1 → Analyzes macro defns.
pass 2 → expands all macro calls.

Databases
1) macro name Table

macro name	MDT index.
PRINT_HELLO	1
ADD_NUMBERS	5

Subject : _____ Topic : _____ Class : _____ Div. : _____ Roll No. : _____ Date : _____ Page No. : _____

2) MDT

MDT index	macro instr.
1	MOV DX, OFFSET HELLO
2	MOV AH, 09h
3	INT 21h
4	ENDM

3)

3) Argument list Array.

Parameter	Actual value.
&ARG1	HELLO
&ARG2	WORLD.

Q12)

```
MACRO
ADD1    &arg1, &arg2
LOAD    &arg1
ADD     &arg2
STORE   &arg1
MEND
MACRO
MUL1    &arg3, &arg4
MOV     A, 00
MOV     C, &arg4
ADD     &arg3
DEC     C
JNC     Repeat
MEND
START   0
ADD1    N1, N2
MUL1    N3, N4
N1      DB      1
N2      DB      2
N3      DB      3
N4      DB      4
        END
```

MDT

Index	Def	
1		ADD1 &arg1, &arg2
2		LOAD #1
3		ADD #2
4		STORE #1
5		MEND
6		MUL1 &arg3, &arg4
7		MOV A, 00
8		MOV C, #2
9	Repeat	ADD #1
10		DEC C
11		INC Repeat
12		MEND

ALA

1) ADD1 &arg1, &arg2
&arg1 → #1
&arg2 → #2

2) MUL1 &arg3, &arg4
&arg3 → #1
&arg4 → #2

MNT

Index	Name	MN Index
1	ADD1	1
2	MUL1	6

ALA.

1) ADD1
#1 → N1
#2 → N2

2) MUL1
#1 → N3
#2 → N4.

10/13

A

RD 1/2/3/12

Expansion →

~~STAR~~	START 0			JNC Repeat
	LOAD N1	N1	DB	1
	ADD N2	N2	DB	2
	STORE N1	N3	DB	3
	MOV A, 00	N4	DB	4
	MOV C, N4			End.
Repeat	ADD N3			
	DEC C			