Jaipur Engineering College & Research Centre

Unit test-1

Subject: Microprocessor and Interface (4IT1A)

Semesters & Section-4IT1 (A+B)

UNIT 1 & 2 - THE 8085 MICROPROCESSORS

1. What is Microprocessor? Give the power supply & clock frequency of 8085
2. What are the functions of an accumulator?
3. Mention the purpose of SID and SOD lines
4. What is the function of IO/M signal in the 8085?
5. What is an Opcode and Operand?
6. List out the five categories of the 8085 instructions. Give examples of the instructions for each group?
7. What is a microcomputer?
8. How many interrupts does 8085 have, mention them.
9. Define instruction cycle, ma chine cycle and T-state
10. What is an instruction?

11. What is the use of ALE?

12. How many machine cycles does 8085 have, mention them

13. Explain the signals HOLD, READY and SID

14. Explain LDA, STA and DAD instructions

15. Explain the different instruction formats with examples

16. What is the use of addressing modes, mention the different types?

17. Give the register organization of 8085.

18. What is Microprocessor, Microcontroller and Microcomputer?

19. Define Flags

20. Define the terms T-state, instruction cycle and machine cycle?

APPENDIX-B

INSTRUCTION SET FOR 8085 MICROPROCESSOR

HEX	MNEMONIC	
CE	ACI	8-Bit
8F	ADC	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
87	ADD	A
80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
C6	ADI	8-Bit
A7	ANA	A
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
E6	ANI	8-Bit
CD	CALL	16-Bit
DC	CC	16-Bit
FC	CM	16-Bit
2F	CMA	

HEX	MNEMONIC	
3F	CMC	
BF	CMP	A
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
D4	CNC	16-Bit
C4	CNZ	16-Bit
F4	CP	16-Bit
EC	CPE	16-Bit
FE	CPI	8-Bit
E4	CPO	16-Bit
CC	CZ	16-Bit
27	DAA	
09	DAD	B
19	DAD	D
29	DAD	H
39	DAD	SP
3D	DCR	A
05	DCR	B
0D	DCR	C
15	DCR	D
1D	DCR	E
25	DCR	H
2D	DCR	L
35	DCR	M
0B	DCX	B
1B	DCX	D

HEX	MNEMONIC	
2B	DCX	H
3B	DCX	SP
F3	DI	
FB	EI	
76	HLT	
DB	IN	8-Bit
3C	INR	A
04	INR	B
0C	INR	C
14	INR	D
1C	INR	E
24	INR	H
2C	INR	L
34	INR	M
03	INX	B
13	INX	D
23	INX	H
33	INX	SP
DA	JC	16-Bit
FA	JM	16-Bit
C3	JMP	16-Bit
D2	JNC	16-Bit
C2	JNZ	16-Bit
F2	JP	16-Bit
EA	JPE	16-Bit
E2	JPO	16-Bit
CA	JZ	16-Bit
3A	LDA	16-Bit
0A	LDAX	B
1A	LDAX	D
2A	LHLD	16-Bit

HEX	MNEMONIC	
01	LXI	B, 16-Bit
11	LXI	D, 16-Bit
21	LXI	H, 16-Bit
31	LXI	SP, 16-Bit
7F	MOV	A,A
78	MOV	A,B
79	MOV	A,C
7A	MOV	A,D
7B	MOV	A,E
7C	MOV	A,H
7D	MOV	A,L
7E	MOV	A,M
47	MOV	B,A
40	MOV	B,B
41	MOV	B,C
42	MOV	B,D
43	MOV	B,E
44	MOV	B,H
45	MOV	B,L
46	MOV	B,M
4F	MOV	C,A
48	MOV	C,B
49	MOV	C,C
4A	MOV	C,D
4B	MOV	C,E
4C	MOV	C,H
4D	MOV	C,L
4E	MOV	C,M
57	MOV	D,A
50	MOV	D,B
51	MOV	D,C

HEX	MNEMONIC	
52	MOV	D,D
53	MOV	D,E
54	MOV	D,H
55	MOV	D,L
56	MOV	D,M
5F	MOV	E,A
58	MOV	E,B
59	MOV	E,C
5A	MOV	E,D
5B	MOV	E,E
5C	MOV	E,H
5D	MOV	E,L
5E	MOV	E,M
67	MOV	H,A
60	MOV	H,B
61	MOV	H,C
62	MOV	H,D
63	MOV	H,E
64	MOV	H,H
65	MOV	H,L
66	MOV	H,M
6F	MOV	L,A
68	MOV	L,B
69	MOV	L,C
6A	MOV	L,D
6B	MOV	L,E
6C	MOV	L,H
6D	MOV	L,L
6E	MOV	L,M
77	MOV	M,A
70	MOV	M,B

HEX	MNEMONIC	
71	MOV	M,C
72	MOV	M,D
73	MOV	M,E
74	MOV	M,H
75	MOV	M,L
3E	MVI	A, 8-Bit
06	MVI	B, 8-Bit
0E	MVI	C, 8-Bit
16	MVI	D, 8-Bit
1E	MVI	E, 8-Bit
26	MVI	H, 8-Bit
2E	MVI	L, 8-Bit
36	MVI	M, 8-Bit
00	NOP	
B7	ORA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
F6	ORI	8-Bit
D3	OUT	8-Bit
E9	PCHL	
C1	POP	B
D1	POP	D
E1	POP	H
F1	POP	PSW
C5	PUSH	B
D5	PUSH	D

HEX	MNEMONIC	
E5	PUSH	H
F5	PUSH	PSW
17	RAL	
1F	RAR	
D8	RC	
C9	RET	
20	RIM	
07	RLC	
F8	RM	
D0	RNC	
C0	RNZ	
F0	RP	
E8	RPE	
E0	RPO	
0F	RRC	
C7	RST	0
CF	RST	1
D7	RST	2
DF	RST	3
E7	RST	4
EF	RST	5
F7	RST	6
FF	RST	7
C8	RZ	
9F	SBB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H

HEX	MNEMONIC	
9D	SBB	L
9E	SBB	M
DE	SBI	8-Bit
22	SHLD	16-Bit
30	SIM	
F9	SPHL	
32	STA	16-Bit
02	STAX	B
12	STAX	D
37	STC	
97	SUB	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
D6	SUI	8-Bit
EB	XCHG	
AF	XRA	A
A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
EE	XRI	8-Bit
E3	XTHL	

NOTE: 1-Byte Instructions: Operand R,M or implicit
2-Byte Instructions: Operand 8-Bit
3-Byte Instructions: Operand 16-Bit

How To Write An Assembly language Program

29/8/2022

Algorithm:- Before writing the program the programmer must understand the problem for which the program is to be written. Then Algorithm can be written.

Algorithm is the step-wise defination written in simple and precise language, in which the large problem is broken into unbreakable steps, so that each step can be easily converted into programming statement.

For Example if we want to WAP to add Two number. The Algorithm is written as below

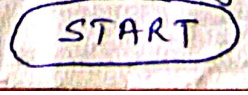
Algorithm to add two 8 bit Number

1) Read first Number
2) Read second Number
3) Add Numbers and save the sum
4) Display result.

1) A ← first No
2) B ← second No.
3) A ← A+B
4) Display Port ← A (SUM)

Flow chart- Flow chart is the graphical Representation of an Algorithm, which shows the flow of program execution.

1) **Terminal** - Oval symbol shown below is used to indicate terminal (start/end) of the Program words like START/BEGIN, STOP/END are written in Terminal symbol



ii) Input/output :-

The Parallelogram symbol is used to Represent the I/O process as shown below

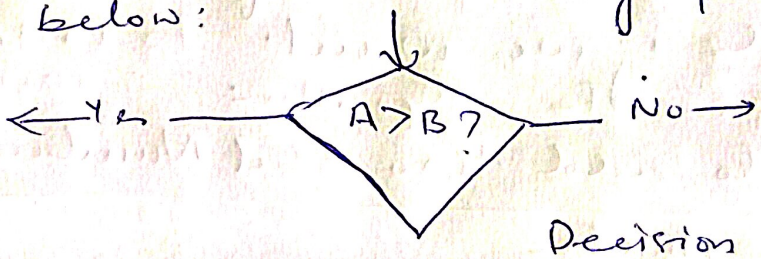
IN OUT — INPUT/OUTPUT SYMBOL

iii) Process - the Rectangle symbol is used to Represent the process operation.

MOV A,B — Process symbol

(iv) Decision Making

the rhombus / diamond symbol is used to Represent the decision making process as shown below:

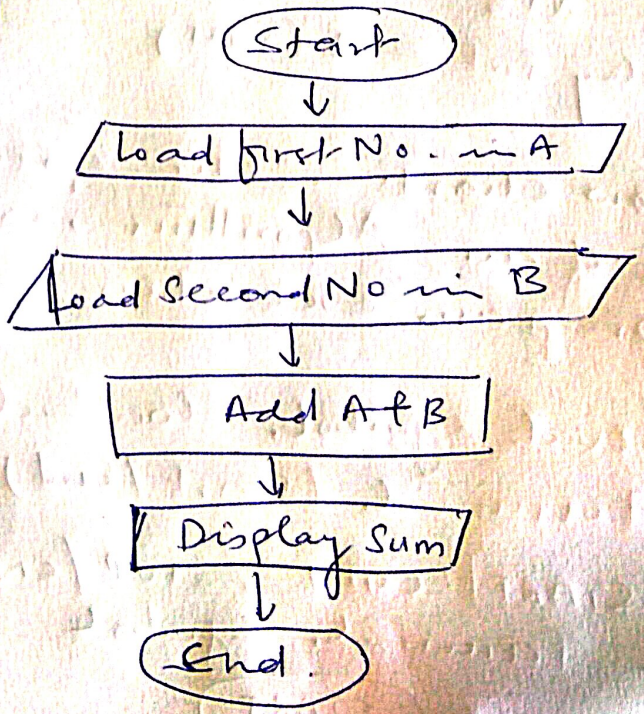


Decision Making Symbol

(v) Flow-Direction :-

Flow directions are Represented by arrows.

For Example - Flow chart of Adding two numbers may be drawn as shown below.



Writing and Executing an Assembly / Machine Language Program

a) Program to add Two 8bit Numbers

AIM :- WAP to add two 8bit Numbers stored in Register A and Register B. Display the Result at O/P Address 2050H.

Solution

Memory Address	Mnemonics	Hex Code	Remarks.
2000H 2001H	MVI A, 20H	3EH 20H	Load 20H in Register A
2002H 2003H	MVI B, 30H	06H 30H	Load 30H in Register B
2004H	ADD B	80H	Add Content of Reg B with A and Store the result in A
2005H 2006H 2007H	STA 2050H	32H 50H 20H	Display the Content of A in Memory location 2050H.
2008H	HLT	76H	STOP the program

(2) WAP two Add 16 bits Numbers 2570H and 32A1H using DAD Instructions. Store the result in HL memory.

Ans:- DAD Instruction in 8085, which adds the Content of any Register pair with Content of HL Pair and Result Stored in memory location 3050H and 3051H.

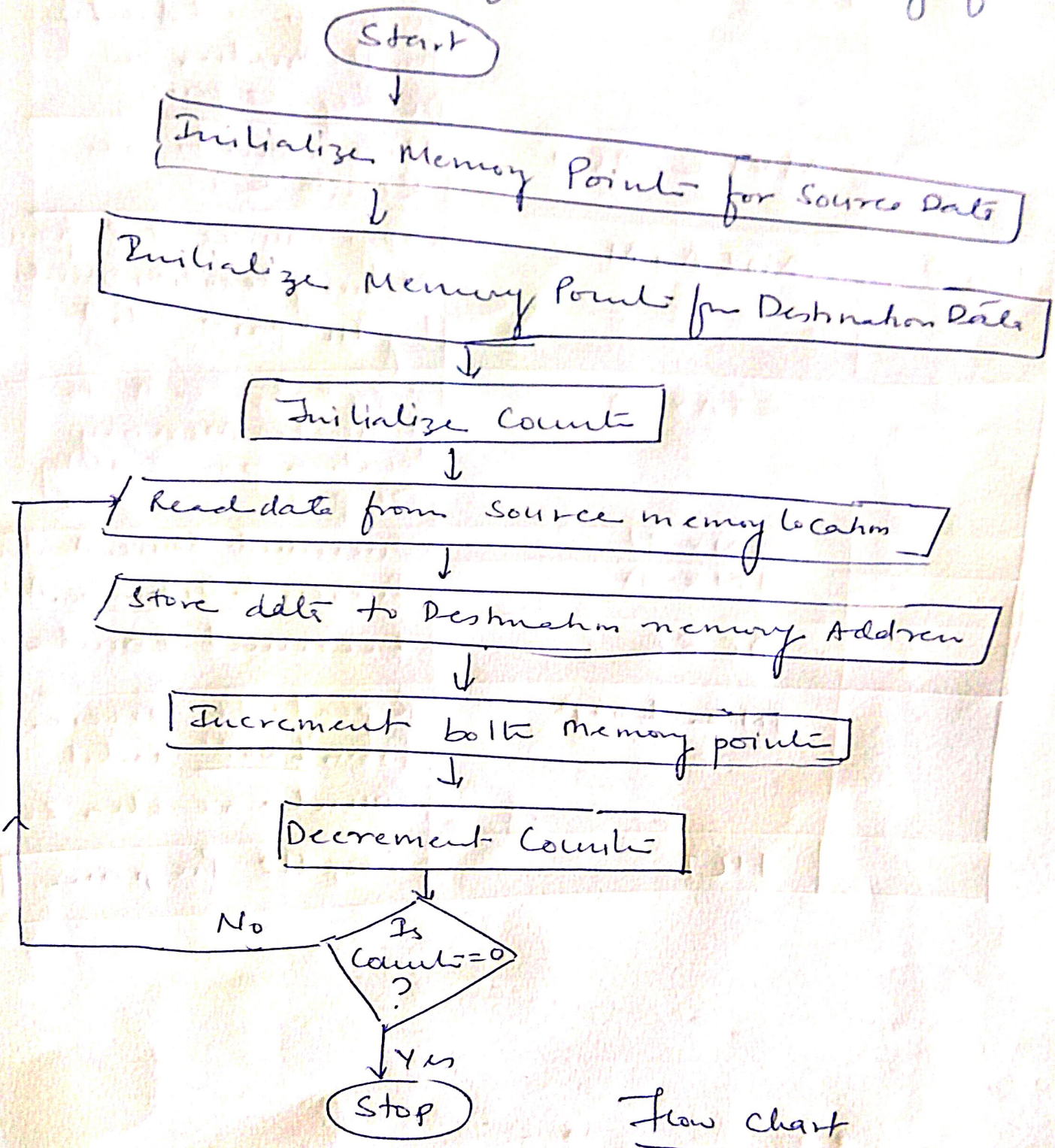
Memory Address	Mnemonics	Hex code	Remarks
2000H 2001H 2002H	LXI B, 257DH	01H 7DH 25H	Load first Number in BC Register Pair
2003H 2004H 2005H	LXI H, 32A1H	21H A1H 32H	load Second Number in HL Register pair
2006H	DAD B	09H	Add the Content of BC with HL Result stored HL
2007H	MOV A, H	7CH	Copy higher order byte of the Result into A
2008H 2009H 200AH	STA 3050H	32H 50H 30H	Store higher order byte in Memory location 3050H
200BH	MOV A, L	7DH	Copy lower order byte of the Result into A.
200CH 200DH 200EH	STA 3051H	32H 51H 30H	Store lower order byte in Memory location 3051H.
200FH	HLT	76H	Stop the Program

Result - BC - 257DH
HL - 32A1H

HL - 5812H

3. Copy Data byte from One Memory Block to Another Memory block.

Question WAP to copy a block of data byte from One Memory location from 2051H to another memory location starting from 3051H.



Flow Chart

Label	Mnemonics	Remarks
	LXI H, 2050H	Initialize Memory pointer with Address of LSB where the Data
	LXI D, 3050H	Store the Starting memory address of the Destination in DE Register pair
	MVI C, 05H	Store count in Register C.
LOOP:	MOV A, M	Copy source Data from Memory location pointed by HL pair into A
	STAX D	Copy Data in A to Destination Memory location pointed by DE pair
	INX H	Increment Source Address
	INX D	Increment Destination Address
	DCR C	Decrement Count
	JNZ LOOP	Repeat the process From 'LOOP' until Count reaches Zero
	HLT	Stop the Program.

5.2 DEBUGGING A PROGRAM :-

Debugging a program is similar to troubleshooting H/W, In program the result binary, either it works or it does not work.

Two type of Debugging are :-

1) Static Debugging - To visual Inspection of ckt board; it is done by a penciled paper check of a flow chart and m/c code.
2) Dynamic Debugging - observing I/O/P, or vaious Control following the execution of each instruction (the Single Step Technique) or of a group of instructions (the break point Technique).

Debugging a M/C code :-

The m/c code will have errors just like ckt board.

1) Selecting a wrong code.
2) Forgetting the second or third byte of an Inst.
3) Specifying the wrong jump location.
4) Not reversing the order of high and low byte in JMP Inst
5) Writing memory address in decimal, thus specifying wrong Jump location.

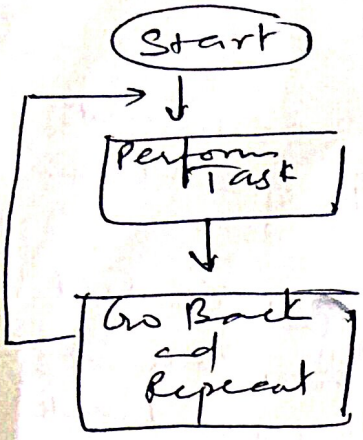
PROGRAMING TECHNIQUES :-

LOOPING, COUNTING AND INDEXING :-

PT used to instruct the MP to repeat task is called Looping. A loop is set by MP to change the sequence of execution and perform task again. This process by using JMP Inst.

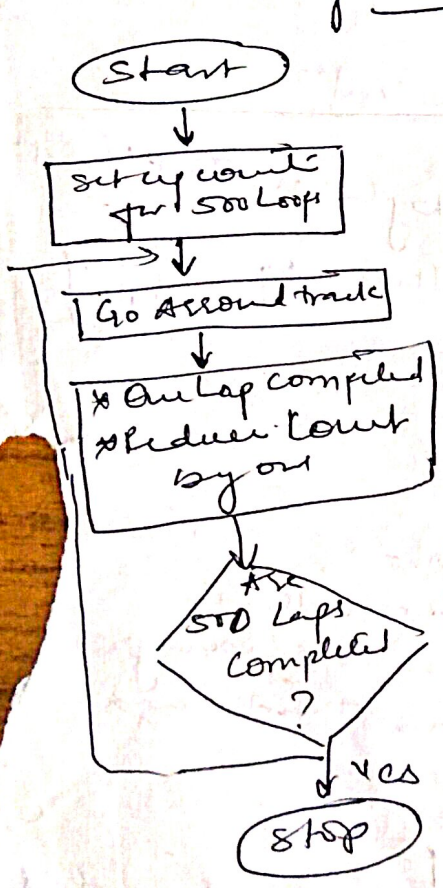
* Loop are classified into two groups.
1) Conditional loop - Repeat a task until Certain data Condition are met.
2) Continous Loop - Repeat a task continously.

1) Continuous Loop - A CL set up by using the unconditional JMP instn.

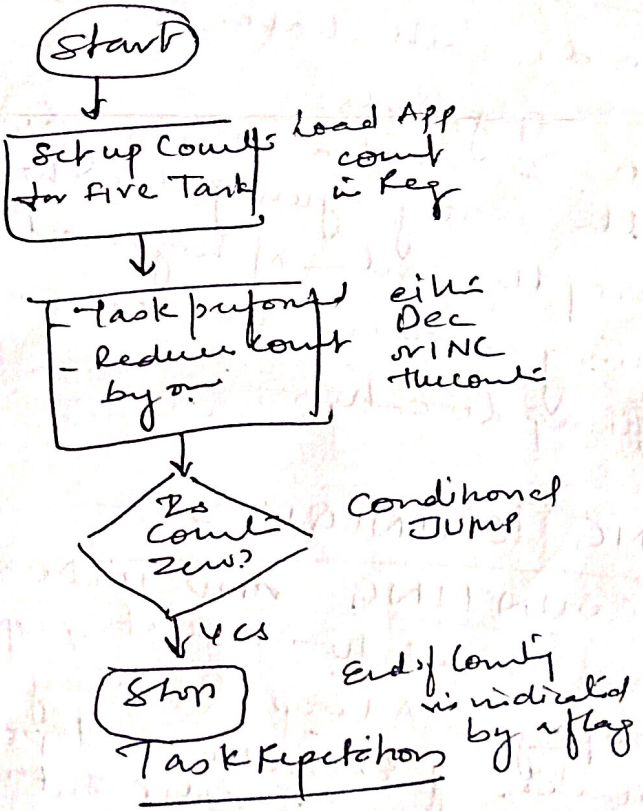


A program with a CL does not stop repeating the task until the system is reset - Ex - C counter or a continuous monitor system.

(2) Conditional Loop - CL is set by conditional JMP instn that check flags (zero, carry etc). and repeat the specified task if condition is satisfied. Three loop usually counter and Indexing



Laps in a Car Race



Task Repetitions

Block of data transfer Example

XX00 to YY00

Date

Expt. No. Page No.

DEBUGGING A PROGRAM :-

Debugging a program is similar to troubleshooting H/w. The Debugging program can be divided into 2 parts = static debugging.
= Dynamic debugging

Static debugging :- is similar to visual inspection of a ckt board it is done by a paper and pencil. check of flowchart and mdelwn code.

Dynamic Debugging - involves observing the o/p, or register content, following the execution of each Instn or of a group of Instn.

The following errors are common are :-

1) Stealing a wrong code.
2) Forgetting the second or third byte of an Instruction
3) Specifying the wrong jump location.
4) No reversing the order of high and low byte in Jump Instruction.
5) Writing memory adelresses in decimal than specifying wrong Jump location.

Teacher's Signature : _____