LABORATORY MANUAL

MICROPROCESSORS AND MICROCONTROLLERS

S.E (I.T) (Sem. IV)

Index

Lab Session 1

Title: To add two 8 bit numbers.

Objective:

Program involves storing the two 8-bit nos in memory locations and adding them taking into consideration the carry generated. The objective of this program is to give an overview of arithmetic instructions of 8086 for 8-bit operands.

References:

1.8086/8088 Family, Architecture, Programming and Design - Yu-Cheng Lui, lenn Gibsons

2.8086/8088 Interfacing, Programming and Design - John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of Arithmetic Instructions of 8086 for 8-bit operands.

Theory :

The addressing modes used in program are:

1) Direct addressing mode: in this mode address of operand is directly Specified instruction that address is offset address of the segment being indicated by instruction

e.g. MOV CL, [4521H]

assume DS = 5000 H therefore PA = 50000+4321=54321

therefore CL = [54321 H]

2) Register Addressing Mode:

In this mode operand are specified using registers. Instructions are shorter but operations cannot be identified looking at instruction.

e.g. MOV CL,DL

3) Based Indexed Addressing Mode:

The operand address is calculated as base register plus index register.

e.g. MOV DX, [BX + SI] moves word from address pointed by BX + SI in data segment to DS.

EA = DS x 10H + BX + SI

4) Base index plus displacement

In this mode address of operand is calculated base register plus index register plus displacement. e.g. MOV CX, [BX+DI+DS]

This moves a word from address pointed by BX + DI +DS in data segment to CX.

Algorithm:

1. Initialize the data segment.
2. Store two 8-bit numbers in memory locations.
3. Move the 1st number in any one of the general purpose register.
4. Move the 2nd number in any other general purpose register.
5. Add the 2 numbers.

6. Store the result in memory location.
7. Check for carry flag. If carry flag is set then store '1' as MSB of result.
8. Stop

Post Lab Assignments:

1. Explain arithmetic instructions of 8086.
2. Explain TASM directives.
3. Explain the instances when the carry flag is set.

Lab Session 2

Title: TO ADD TWO 16 BIT NUMBERS

Objective: Program involves storing the two 16 bit numbers in memory locations and adding them taking into consideration the carry generated. The objective of this program is to give an overview of arithmetic instructions of 8086 for 16 bit operation.

Pre-Requisites:

1. Instructions of microprocessor 8086
2. Addressing mode of microprocessor 8086.
3. Flag register of microprocessor 8086
4. Knowledge of tasm directories.

.

Theory: Base index displacement addressing mode is used for 16 bit addition. Here the effective address is obtained by adding displacement value with base register and index register..

Algorithm:

1. Initialize the data segment.
2. Store two 16 bit numbers in memory locations.
3. Move the 1st number in any one of the general purpose register.
4. Move the 2nd number in any other general purpose register.
5. Add the 2 numbers.
6. Store the result in memory location.
7. Check for carry flag. If carry flag is set then store '1' as MSB of result.
8. Stop

Observations:

1. Fill the table of registers with it's contents after execution of specific instructions.
2. Fill the table of Memory with the address and data of the numbers on which operation is performed and final result\s obtained.

Post Lab Assignments:

1. List of condition of the flag in the status register after ADD instructions has been executed.
2. Some 8086 mnemonics which is used for addition and storing a DATA

Lab Session 3

Title : TO PERFORM 8 BIT MULTIPLICATION.

Objective: Program involves storing the two 8 bit numbers in memory locations and performing multiplication operation.

Pre-Requisites:

1. Instructions of microprocessor 8086
2. Addressing mode of microprocessor 8086.
3. Flag register of microprocessor 8086
4. Knowledge of tasm directories.

Theory:

MUL instruction

This instruction multiplies byte/word present in source with AL/Ax.

After 8 bit multiplication AH stores the higher byte of result. After 16 bit multiplication DX stores higher word of result

e.g. MUL BL

This instruction multiplies data in BL register with data in AL register. Result is stored in AX register where AH store higher byte of result.

Algorithm:

1. Initialize the data segment.
2. Initialize general purpose register.
3. Move the content of DS to AL register.
4. Increment general purpose register by 1.
5. Move content of DS to any other 8 bit register.
6. Perform multiplication operation using MUL instruction
7. Stop

Observations:

1. Fill the table of registers with its contents after execution of specific instructions.
2. Fill the table of Memory with the address and data of the numbers on which operation is performed and final result\s obtained.

Post Lab Assignments:

1. List of condition of the flag in the status register after MUL instructions has been executed.
2. Some 8086 mnemonics which is used for multiplication and storing a DATA

Lab Session 4

Title : TO MULTIPLY TWO 16 BIT NUMBERS

Objective: Program involves storing the two 16 bit numbers in memory locations and multiplying them the objective of this program is to give an overview of arithmetic instructions of 8086 for 16 bit operation.

Pre-Requisites:

1. Instructions of microprocessor 8086
2. Addressing mode of microprocessor 8086.
3. Flag register of microprocessor 8086
4. Knowledge of tasm directories.

Theory : This can be perform by using MUL instruction. Out of two 16 bit operands, one 16 bit operand is stored in AX register and the other operand is stored in a general purpose register or a memory location which mentioned in instruction.

e.g. MUL BX

Multiplies content of AX and BX register. The result is 32 bit data. The higher 16bit of result is stored in DX register and the lower 16 bits of result is stored in AX register.

MUL Word PTR [1150]:

Multiplies content of memory location pointed by 1150 & 1151 & AX register & stored the higher 16 bit in DX register & lower 16 bit in Ax register.

Algorithm:

1. Initialize the data segment.
2. Store two16 bit numbers in memory locations.
3. Move the 1st number in any one of the general purpose register.
4. Move the 2nd number in any other general purpose register.
5. Multiply the 2 numbers.
6. Store the result in memory location.
7. Stop.

Observations:

1. Fill the table of registers with its contents after execution of specific instructions.
2. Fill the table of Memory with the address and data of the numbers on Which operation is performed and final results obtained.

Post Lab Assignments:

1. List of condition of the flag in the status register after MUL instructions has been executed.
2. Some 8086 mnemonics which is used for multiplication and storing a DATA

Lab Session 5

Title: TO IMPLEMENT BLOCK TRANSFER.

Objective:

Program involves transferring source string from a particular location in source segment (Data Segment) to the desired location in destination segment (Extra Segment). The objective of this program is to give an overview of the String instructions of 8086.

References:

1.8086/8088 Family, Architecture, Programming and Design
-Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing, Programming and Design
- John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of String Instructions of 8086.

Theory:

A string is a series of bytes stored sequentially in the memory string instruction operate on such strings. The sequence element is taken from the data segment using the SI register the destination element is in the extra segment pointed by DI register. SI/DI are incremented/ decremented after each operation depending upon the destination flag 'OF' in the flag register.

MOVS: MOVSB/MOVSW (move string)

It is used to transfer a word/byte from data segment to extra segment the offset of the source in data segment is in SI the offset of the destination is in ES in DI. SI & DI is incremented / Decremented after the transfer depending upon the direction flag.

REP (Repeat):

This is an instruction prefix in which can be used in string instruction. It causes the instruction to be repeated (X number of time)

e.g. MOV CX,0023H
CLD
REP MOVSB.

Algorithm:

1. Initialize the data segment.
2. Store the source string in consecutive memory location.
3. Initialize the extra segment.
4. Allocate consecutive memory locations for transfer.
5. Load the effective address of source string in SI register.
6. Load the effective address of destination string in DI register.
7. Initialize the Direction flag for Auto increment or Auto decrement.
8. Store number of bytes to be transferred in any of the general purpose registers.
9. Transfer the source string using appropriate string instructions (MOVSB / MOVSW)

10. Decrement count.
11. Check if count = 0.If yes then stop else repeat steps 9 - 11.
12. Stop

Post Lab Assignments:

1. Explain string instructions.
2. Explain the purpose of direction flag.
3. Explain the REP prefix.
4. Explain the difference between CMP and CMPSB

Lab Session 6

Title: TO CLEARING DATA IN MEMORY LOCATION.

Objective:

Program involves clearing data from a particular location in source segment (Data Segment). The objective of this program is to give an overview of the String instructions of 8086.

References:

1.8086/8088 Family, Architecture, Programming and Design - Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing, Programming and Design - John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of String Instructions of 8086.

Theory:

The clearing of 10 bytes of data in memory location can be done using string instruction.
STOSB : this string instruction is repeated 10 times using repeat (REP)prefix.
e.g. STOSB
This instruction copies the content of AX register and stores it in the memory location provided by ES: DI. Thus by string zero in AL register we can clear 10 bytes of data using STOSB. The ES & DI should point to the memory location which has to be cleared.

Algorithm:

1. Initially ES & DS such that it points to the memory location which has to be cleared first.
2. Set the counter register (CX) to 10.
3. Set AL register to value zero.
4. Perform STOSB operation to clear the memory.
5. Increment DI.
6. Repeat STOSB using REP prefix, count number of time.
7. Thus 10 consecutive memory locations are cleared.
8. Stop.

Post Lab Assignments:

1. Explain string instructions.
2. Explain the REP prefix.

Lab Session 7

Title: TO COUNT A NUMBER OF 1'S IN A GIVEN STRING.

Objective:

Program involves counting of 1's in a given string. The objective of this program is to give an overview of the String instructions of 8086.

References:

1.8086/8088 Family, Architecture, Programming and Design - Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing, Programming and Design - John Uffenback

Pre-Requisites:

1. .Knowledge of TASM directives.
2. Knowledge of String Instructions of 8086.

Theory: The program involves counting the number of 1's in a given 16 bit/ 8 bit number.
The instruction used in this program is ROL.
ROL dest, count
Left shift the bit of destination. MSB shifted to the carry. MSB goes to LSB. Bits are used shifted count number of times. If count = 1. It is specified in instruction if count > 1 load to CL register and CL gives count in the instruction.
Destination: Register as memory location.
e.g. ROL AX,1 left shift X bits once.

Algorithm:

1. Initialize the data segment.
2. Initialize BL = 0
3. Load given number.
4. Perform ROL and Al
5. If CF = 1, increment else decrement CX and Check zero flag.
6. Repeat till zero
12. Stop

Post Lab Assignments:

1. Explain Logical instruction in 8086.

Lab Session 8

Title: TO FIND EVEN AND ODD NUMBERS IN A GIVEN STRING.

Objective:

Program involves counting the even and odd numbers in a given array. The objective of this program is to give an overview of the String instructions of 8086.

References:

1.8086/8088 Family, Architecture, Programming and Design
- Yu-Cheng Lui, Glenn Gibsons
2.8086/8088 Interfacing, Programming and Design
- John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of String Instructions of 8086.

Theory:

A string is a series of bytes stored sequentially in the memory string instruction
Operate on such 'strings'. The SRC element is taken from the data segment using and S1 register. The destination element is in extra segment pointed by D1 register. These registers are incremented or decremented after each operation depending upon the direction flag in flag register.
CLC: - the instruction clears the carry flag.
RCR: - destination, count
Right shift the bits of destination. LSB is shifted into CF. CF goes to MSB. Bits Are shifted counts no of times.
JC: jump to specified location.
INC/DFC destination: add 1 to the specified location. If CF is set to 1.No CF is affected for DFC.
JMP: Depending upon the location the control is shifted there (intra/inter seg)
JNZ: jump to location if ZF = 0.

Algorithm:

1. Initialize the data segment.
2. Initialize the array.
3. Load the effective address of array in any general purpose register.
4. Load total number of elements of the array in any register.
5. Test LSB of CX register. If LSB =1then no. is odd else no. is even.
6. Store result in DX register.
7. Stop.

Post Lab Assignments:

1. Explain shift & rotate instructions.

Lab Session 9

Title: ARRANGING NUMBER IN ASCENDING ORDER.

Objective:

Program involves sorting an array in ascending order using Bubble sort algorithm. The objective of this program is to give an overview of the Compare and Jump instructions. Use of Indirect Addressing mode for array addressing is expected.

References:

1.8086/8088 Family,Architecture,Programming and Design
- Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing,Programming and Design
- John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of CMP and Jump Instructions of 8086.

Theory:

1) compare instruction:
CMP dest, source
The instruction compares the source with destination. The source and destination must be of same size. Comparison is done by internally subtracting source from destination. The result of this is not stored anywhere instead flag bits are affected.
2) JMP instruction: if condition is true then it is similar to then it is similar to an intra segment branch and if false then branch does not take place & next instruction is executed. The destination must be in range of -128 to 127 from address of instruction.
INC next: Short jump to next only if CF = 0

Algorithm:

1. Initialize the data segment.
2. Initialize the array to be sorted.
3. Store the count of numbers in a register.
4. Store count-1 in another register.
5. Load the effective address of array in any general purpose register.
6. Load the first element of the array in a register.
7. Compare with the next element of the array.
8. Check for carry flag.
9. If carry=0 first number > second number. Swap the 2 numbers.
10. Increment to the contents of the SI register so that it points to the next element of the array.
11. Decrement (count-1) by 1.
12. Check if (count-1) =0. If no then repeat steps 7 to 11.
13. Decrement count by 1.
14. Check if count = 0.If no then repeat steps 6 through
15. Stop.

Post Lab Assignments:

1.Explain CMP and Jump Instructions.
2.Explain Indirect Addressing Modes.

Lab Session 10

Title: TO FIND AVERAGE OF 10 NUMBERS.

Objective:

Program involves averaging of a ten numbers.

References:

1.8086/8088 Family,Architecture,Programming and Design - Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing,Programming and Design - John Uffenback

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of CMP and Jump Instructions of 8086.

Theory:

CLC: Clears carry flag. No other flags affected.
LEA: load effective address of source into given register.
E.g. LEA BX, total: BX – effective address of total.
ADC dest, source:
Adds source to destination along previous carry & stores it in destination. Both source & destination have to be of same size.
E.g. ADC BX, CX; BX- BX+CX+CF
JNZ: jump to location if ZF= 0
DEC destination: subtracts 1 from specified destination.
INC destination:
Add 1 to destination.

Algorithm:

1. Initialize the data segment.
2. Clear carry flag.
3. Store initial address of list in a register.
4. Move 0A to register AL.
5. Store count in register AL.
6. Reset AL registers (content of AX)
7. Add one number from array to contents of AL
8. Decrement count
9. Increment SI
10. If count $\neq 0$ go to step 8
11. Divide content of AX with that of BL.
12. Store answer in register.
13. Stop.

Post Lab Assignments:

1. Write different instructions used.

Lab Session 11

Title: **Display Character from keyboard until 0 is entered.**

Objective:

To Read Character from Keyboard and display on screen until 0 is pressed.

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of DOS Interrupt.

Theory: **Instruction used in program are:**

- **MOV AH,08H**
  INT 21H

Read Input From Keyboard without echo and store at AL.

- **MOV AH,02H**
  INT 21H
  Display Character on screen. Character should be in DL register.

Algorithm:

1. Initialize the data segment.
2. Read input from keyboard.
3. Compare input with ASCII value of ZERO.
4. If result is 0, go to step 7.
5. Move content of AL to DL, to display it on screen.
6. Display character on screen.
7. Stop

Lab Session 12

Title: Password Verification

Objective:

The objective is to make use of string instruction and MACRO,to check whether the entered password by the user is correct or not.

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of DOS interrupt.
3. Knowledge of string instruction and MACRO.

Algorithm:

1. Store Initial password into Array.
2. Write Macro for printing output message.
3. Write Macro to display '*'.
4. Initialize the data segment.
5. Set the counter value=no. of character present in password.
6. Load Effective address of stored password in BX.
7. Take input from the keyboard.
8. Compare input with the password string.
9. If zero=0, both value are equal. Go to step 10.
   If zero is not equal to 0.Go to step 15.
10. display '*' Macro.
11. Increment BX.
12. Decrement counter by 1.
13. Check if counter=0.If not, Repeat step 7 to 12.
14. Display Macro message for correct password, Go to step 16.
15. Display '*' macro and Macro message for wrong password.
16. End

.

Post Lab Assignments:

1. Explain Macro.

Lab Session 12

Title: To write a program for 3 X 3 matrix Addition

Objective:

The objective is to multiply 3 X 3 matrix .

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of DOS interrupt.
3.Knowledge of string instruction and MACRO

Algorithm:

1. Initialize the data segment.
2. Initialise counter = 9
3. Initialize pointer DI to matrix 1.
4. Initialize pointer Bx to matrix 2.
5. Initialize pointer SI to result matrix 3.
6. Get the number from matrix 1.
7. Add number from matrix 1 with matrix 1 number.
8. Save the carry if any.
9. Save the result in result matrix 3.
10. Increment DI,BX,SI to point to next element.
11. Decrement count.
12. Check if count = 0,if not goto step VI else goto step XIII
13. Display the result.
14. Stop.

Lab Session 11

Title: TO CHECK IF THE EXTERNAL STRING IS A PALINDROME OR NOT.

Objective: To search the given string is palindrome.

Pre-Requisites:
1. Knowledge of string instruction.
2. Knowledge of TASM directive.
3. Addressing mode of microprocessor 8086.
4. Flag register of microprocessor 8086

Reference:
1.8086/8088 Family,Architecture,Programming and Design - Yu-Cheng Lui,Glenn Gibsons
2.8086/8088 Interfacing,Programming and Design - John Uffenback

Theory:
String instruction/ data transfer
LEA register, Source
Load effective address (offset Address) of source in given register.
e.g. LEA BX, Total; BX – Effective address of Total DS.
STOSB/STOSW (Store string) – It is used to store AL(or AX) into a memory location pointed by ES:DI DI is incremented (decremented after transfer depending upon DF)
LODSB/LODSW: Used to copy the contents of memory location pointed by DS:SI & store it in AL register SI is incremented /decremented after transfer depending upon DF.

Algorithm:
1. Start
2. Define Data Segment (DS)
3. Move data segment address to register AX.
4. Move content of AX to DX
5. Move content of AX to ES.
6. Move 00 to DL.
7. Move 05 to CX
8. Load effective address of source to SI.
9. Load effective address of destination to DI.
10. Clear DF & load string in byte format.
11. Set DF.
12. Store string in by format pointed by DI.
13. Loop to the step 9.
14. Move 0005 to CX.
16. Clear DF.
17. Repeat if zero when comparing string in byte format.
18. Jump if not zero to 21.
19. Move 01h to data byte initialized.
20. Move 41H to Register AH.
21. Stop.

Post Lab Assignments: 1. Explain 8086 programming in assembler.

Lab Session 13

Title: To compute power of number x^N using dos intrrupt.

Objective:

The objective is to make use of string instruction and MACRO,to check whether the entered password by the user is correct or not.

Pre-Requisites:

1. Knowledge of TASM directives.
2. Knowledge of DOS interrupt.
3.Knowledge of string instruction and MACRO.

Algorithm:

1. Initialize data segment
2. Write a macro for reading a value from the keyboard
3. Write a macro for printing input message and result
4. Initialize the code segment.
5. Call the macro to input the number X & N.
6. Write a loop for calculating x^N .
7. Convert the above computed value obtained in step 5 to ASCII value.
8. Call the macro to point the result.

Lab Session 12

Title: TRANSFER A BLOCK OF MEMORY FROM ONE LOCATION TO THE OTHER IN 8051

Objective:

a) To familiarize with data transfer instruction of 8051
b) General Assembly Language Programming concepts of 8051

Pre-requisites: 8051 data transfer instruction

Reference: Kenneth Ayala - Penram International (II edition)

Theory:

Data transfer instructions are used as follows.
MOV < dest - byte>, < src – byte>
This instruction copies the content of the source location to the destination location. The contents of source location are unchanged. It allows fifteen combinations of source and destination addressing modes.
1) MOV A, Rn : this instruction copies the contents of the register Rn to the accumulator. MOV A, R1
2) MOV A, Direct: This instruction copies the contents of direct address given in the instruction to the accumulator. E.g. MOV a, 40H.
3) MOV A, @ Ri: This instruction will copy the contents of memory location whose address is specified in the register Ri of the selected bank of the accumulator. E.g. MOV A, @ R0.
4) MOV A, #data: this instruction will copy the immediate data to accumulator.
e.g. MOV A,#31H.
5) MOV Rn, A: this instruction will copy the contents of accumulator to the register Rn of selected register bank. E.g. MOV R5, A.
6) MOV Rn, Direct: this instruction will copy contents from direct address specified in the instruction to register Rn of selected register bank.
E.g. MOV R3, 30H.
7) MOV Rn, #data: this instruction will copy the immediate data to the register Rn of selected register bank. E.g. MOV R7, #20H.
8) MOV direct, A: this instruction will copy the contents of accumulator to the direct address specified in the instruction. E.g. MOV 80 H, A.
9) MOV direct, Rn: this instruction will copy the contents of register Rn of the selected register bank to the direct address specified in the instruction. E.g. MOV 30 H, R5.
10) MOV direct, direct: this instruction will copy the contents source direct address to the destination direct address. E.g. MOV 20 H, 30 H.

Algorithm:

a) Direct addressing mode:

1. Move the data into accumulator.
2. Move the data from accumulator to the memory location one by one.

b) Register direct addressing mode without loop.

1. Move the data into accumulator.
2. Initialize any register Ro to memory location.
3. Move data of accumulator into the address pointed by Ro.
4. Increment Ro.
5. Repeat step 3 & 5 for 5 more times.
6. End.

c) Register indirect addressing mode with loop

1. Move the data into accumulator.
2. Initialize any register Ro to memory location. Initialize R1 to Another memory address.
3. Move data of accumulator into the address pointed by Ro.
4. Increment Ro.
5. Decrement R1.
5. If R1 $\neq$ 0, Repeat step 3, 4 & 5.
6. End.

Post Lab Assignments:

1. Explain the difference between MOV, MOVX, MOVC
2. Explain register bank in 8051
3. Explain difference between DPTR and PC

Lab Session 13

Title: WRITE A PROGRAM TO TOGGLE BITS FOR PORT 1.

Objective: a) To familiarize with data transfer instruction of 8051
b) General Assembly Language Programming concepts of 8051

Pre-requisites: 8051 data port details.

Reference: Kenneth Ayala - Penram International (II edition)

Theory: 1) SJMP relative address
It is the conditional jump after execution of SJMP instruction contents of relation address with PC to for jump location to which program control is transferred.
2) A call address (Absolute call)
It calls address, subroutine located at a specific address. No flag affected.

Algorithm:

1. Start
2. Move data 00H to the accumulator.
3. Move accumulator content to port.
4. Call delay.
5. Move data FF H to the accumulator.
6. Move accumulator content to port.
7. Call delay.
8. Jump to step 2.
9. Stop.

Observations:

Observe the port pin is toggling from 0 to 1 and again 0.

Post Lab Assignments:

1. Explain the various instructions used.

Lab Session 14

Title: WRITE A PROGRAM TO SERIAL DATA TRANSFER TO TRANSMIT CHARACTER "A"

Objective: a) To familiarize with data transfer instruction of 8051
b) General Assembly Language Programming concepts of 8051

Pre-requisites: 8051 data port details.

Reference: Kenneth Ayala - Penram International (II edition)

Theory: for communication between two computer systems we need to send and receive the data bits serially. The microcontroller 8051 has serial data communication circuit which uses a register in the SFR called SBUF to hold the data. The register SCON controls the data communication; the register PCON controls the data rates.
The SBUF register comprise the two registers physically one of them is write only and is used to hold the data that is to be transmitted out from the microcontroller via the TXD pin, while the other is read only and holds the data that is receive from the external source via the RXD pin.
The serial interface can be operated in four different modes mode 0, mode 1, mode 2, and mode 3 which can be configured using SCON register.
Mode 0: Shift Register Mode 1: 8 bit UART.
Mode2: 9 bit UART with fixed baud rate Mode3:9 bit UART with variable baud rate
Mode 2 &3 has special provision for multiprocessor communication i.e. we can have master /slave configuration.
(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

M0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	Fosc/12
0	1	1	8 – bit UART	Variable
1	0	2	9 – bit UART	F_{osc} /64 or f_{osc} 32
1	1	3	9 – bit UART	Variable

Algorithm:

1. Initializes TMOD register with value.
2. Initializes TH1 register with value -6 for baud 4800
3. Initialize SCON register with value.
4. Start the timer 1.
5. Move the immediate value ‘p’ to SBUF register.
6. Wait till the TI flag is set.
7. Clear the TI flag.
8. To transmit data “A” continuously repeat 5, 6, and 7.

Post Lab Assignments:

1. Explain the various instructions used.

Lab Session 15

Title: TO GENERATE OF 50 % DUTY CYCLE USING 8051.

Objective:

a) Understand 8051 Timers modes
b) Use of TCON, TMOD and PCON SFRs to program the Timers for a particular overflow times
c) Write ISR for Timer Interrupt

Pre-requisites:

1. 8051 Timer modes
2. SFRs TCON, TMOD, IE PCON definitions
3. ISR locations

Reference:

Kenneth Ayala - Penram International (II edition)

Theory:

Timer Mode 0: Setting timer X mode bits to 00b in the TMOD register results in using the THX register as an 8 - bit counter & TLX as a 5 – bit counter; the pulse input is divided by 32d in TL so that TH counts the original oscillator frequency reduced by a total 384d. As an example, the 6 MHz oscillator frequency would result in a final frequency to TH of 15625 Hz. The timer flag is set whenever THX go from FFh to 00h, or in .0164 seconds for 6 MHz. crystal if THX starts at 00h.

Timer Mode 1: Mode 1 is similar to mode 0 except TLX is configured as a full 8-bit counter when the mode bits are set to 01b in TMOD. The timer flag would be set in 01311 seconds using a 6 MHz crystal.

Timer Mode 2: setting the mode bits to 10b in TMOD configures the timer to use only the TLX counter as an 8 bit counter. THX are used to hold a value that is loaded into TLX every time TLX overflow from FFh to 00h. The timer flag is also set when TLX overflows.

This mode exhibits an auto – reload feature: TLX will count up from the number in THX, overflow and initialized again with the contents of THX.

Timer Mode 3: timer 0, 1 may be programmed to be in mode 0, 1 or 2 independently of a similar mode for the other timer. This is not true for mode 3; the timers do not operate independently if mode 3 is chosen for timer 0. Placing timer 1 in mode 3 cause it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer 0.

Timer 0 in mode 3 becomes two completely separate 8 – bit counters.

Algorithm:

Main Program

1. Start
2. Initialize 8255 ports (A - o/p)
3. Initialize timer 0 in Auto-reload mode
4. Enable Timer 0 Interrupt
5. Load count in Timer registers
6. Reset port A
7. Enable Timer Run
8. Go to 8051 in idle mode

Interrupt Service Routine

1. Complement port A
2. Out to port A
3. Go to idle mode

Observations:

Observe the Square Wave generated on CRO and calculate the Time Period between two Interrupts ($T_{obs.}$)

T_{calc} = 12*(256 - Count in TH)/12

Compare $T_{calc.}$ with $T_{obs.}$

Lab Session 16

Title: Handshake operation of 8255

Objective:

To study handshake operation of 8255 (Port A as a input port and port B as an output port read a byte from port A and transfer to port B)

Pre-Requisites:

1. Knowledge of operating modes of 8255.
2. Handshake operation of 8255

Theory: Port A and the high part of port C may be programmed in one of three modes; port B and the lower part of port C may be programmed in one of two modes. The modes are:
Mode 0- Basic I/O: Data written to the port is latched; data read from the port is read from the input pins.(this mode is identical to 8051 port operation.)
Mode 1 – Strobed I/O: This handshaking mode uses port A & B as I/O and port C to Generate handshaking signal to the devices connected to port A & B and an interrupt signal to the host microcontroller.
Mode 2 – Strobed bidirectional I/O: this mode is similar to mode 1 with the ability to use port A as a Bidirectional data Bus.
Mode 1 & 2 require setting interrupt enable bits in the port C data register. These modes are intended to be used with intelligent peripherals such as printer.

Algorithm:

1. To initialize 8255 port A as input port & port B as output port in mode 1
2. Enable INTRA using BSR mode. i.e. Make PC4 high.
3. Enable INTRN using BSR mode i.e. Make PC2 high.
4. Apply input to port A through LED.
5. Check if INTRA is present.
6. If INTRA is present input a byte from port A & increment data counter & store the data into memory
7. Read the status word of port B
8. Check INTRB. If INTRB is present & transfer a byte from memory location to port B

Post Lab Assignments:

1. Explain BSR mode of 8255 in detail.
2. Explain handshake operations of 8255.