A Report

on

**Microprocessor based designing**

of

14-pin ZIF Socket IC Tester

As a part of the course

**CS F241-MICROPROCESSOR PROGRAMMING AND INTERFACING**

**GROUP-74**

**TEAM MEMBER ID**

BHARAT BHARGAVA 2017A7PS0025P

ABHINAVA ARASADA 2017A7PS0028P

SWADESH VAIBHAV 2017A7PS0030P

ANIRUDH ANILKUMAR GOYAL 2017A7PS0031P

****

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**25th APRIL 2019**

**DESIGN QUESTION:**

Design a Microprocessor based tester to test the logical functioning of the following chips:

1. 7408
2. 7486
3. 7432

The IC to be tested will be inserted in a 14 pin ZIF socket. The IC number is to be entered via a keyboard. The results of the test along with the IC number are to be displayed on LCD as "74xy PASS" or "74xy FAIL". Design the necessary hardware and write the necessary ALP for implementing the above-mentioned task.

**SYSTEM DESCRIPTION:**

This system uses the ZIF socket to test the logical functioning of the ICs. The user inserts the IC to be tested in a ZIF socket and enters the IC number using a push button array interface. Corresponding to the value of ICNO entered the IC is tested by comparing the truth table stored in memory with the outputs from IC. Then, the IC no along with pass or fail is printed on a LCD Display.

The main function of the digital IC tester is to test a digital IC for correct logical functioning as described in the truth table. Since it is programmable, we can test any number of ICs, provided we have that much memory. A fail indication results when there is a discrepancy in the functioning of the IC while we are checking for the outputs at each stage.

**HARDWARE DESCRIPTION:**

Following are the chips used in the system: (For each IC)

1. Intel 8086: Microprocessor (1 nos.)
2. Intel 8255A: Programmable Peripheral Interface device (PPI) (2 nos.)
3. 74LS373 octal latches (3 nos.)
4. 74LS245 bidirectional buffers (4 nos.)
5. 74LS138 single 3:8 DeMultiplexer (1 nos.)
6. 6116 SRAM chips (2 nos.)
7. 2732 EPROM chips (2 nos.)
8. ZIF SOCKET (1 nos.)
9. LCD Display (1 nos.)
10. Switch (12 nos.)
11. 74154 (4:16 decoder) (1 nos.)
12. 4078 8 i/p NOR gate (2 nos.)
13. 4075 3 i/p OR gate (1 nos.)
14. 2 i/p OR gate (6 nos.)
15. NOT gate (2 nos.)

Following are the ICs to be tested:

1. 7408 Quadruple 2-Input Positive-AND Gates
2. 7432 Quad 2-input positive-OR gates
3. 7486 Quadruple 2-Input Exclusive-OR Gates

**ASSUMPTIONS:**

1. There is no chance of any power loss during the operation of the system.
2. The 8086 microprocessor has been pre-programmed with the given code.
3. A jump instruction is present at the address generated by the microprocessor to the place where the code is stored.
4. All the chips are inserted in the right sockets and the appropriate buttons are present
5. Sufficient power is supplied to the LCD to keep it on.

**INPUT/OUTPUT PORT ADDRESS OF PPI:**

|  |  |
| --- | --- |
| **NAME OF THE PORT IN 8255A(KEYBOARD)** | **ADDRESS TO ACCESS THE PORT** |
| Port A | 10H |
| Port B | 12H |
| Port C | 14H |
| Control register | 16H |

|  |  |
| --- | --- |
| **NAME OF THE PORT IN 8255A(LCD DISPLAY)** | **ADDRESS TO ACCESS THE PORT** |
| Port A | 20H |
| Port B | 22H |
| Port C | 24H |
| Control register | 26H |

**FLOWCHARTS:**





**ASM CODE:**

.MODEL TINY

.DATA

default\_message DB "ENTER IC "

ch\_enter DW 00H ; number of characters entered

ch\_print DW 0000H ; number of characters to be printed

INVALID DB "INVALID IC" ; for incorrect input

FOUND DB 00H ; 00H-IC not found, 01H- correct IC entered

; kb table for keyboard input

KEY\_TABLE DB 0EEh,0EDH,0EBH,0E7H

DB 0DEH,0DDH,0DBH,0D7H

DB 0BEH,0BDH,0BBH,0B7H

DB 07EH,07DH,07BH,077H

; Test values for checking correctness

in7408 DB 00H,55H,0AAH,0FFH ; input for and gate

out7408 DB 00H,00H,00H,0FH

in7432 DB 00H,55H,0AAH,0FFH ;input for or gate

out7432 DB 00H,0FH,0FH,0FH

in7486 DB 00H,55H,0AAH,0FFH ; input for xor gate

out7486 DB 00H,0FH,0FH,00H

;display table for LCD

DISPLAY\_TABLE DW 00H, 01H, 02H, 03H

DW 04H, 05H, 06H, 07H

DW 08H, 09H, 0AH, 0BH

DW 0CH, 0DH, 0EH, 0FH

IC7408 DB "7408"

IC7432 DB "7432"

IC7486 DB "7486"

ICBAD DB " FAIL"

ICGOOD DB " PASS"

; First 8255

porta1 equ 10h

portb1 equ 12h

portc1 equ 14h

creg1 equ 16h

; Second 8255

porta2 equ 20h

portb2 equ 22h

portc2 equ 24h

creg2 equ 26h

keyboard\_keys DB "0123456789SRABCD” ; keyboard\_keys: R for reset and S for ;backspace

entered\_IC DB "XXXX" ;stores the IC number enetered by user

.CODE

.STARTUP

mov al,8AH ; portb2 and upper portc2 as input and porta2 and lower portc2 as output

out creg2,al

call INITIAL ;initialises LCD

Start:

mov ch\_enter,0

mov FOUND,00H

call clear ; clears display

lea DI,default\_message ;

mov ch\_print,9

call disp\_str ; prints input message to LCD

ReadKey:

mov ch\_print,01H

call KEYBRD\_PRESS ; reads keypush. offset from kb table is stored in AX

lea DI,keyboard\_keys ; list of keypad keyboard\_keys

add DI,AX ;

mov SI,ch\_enter

mov AL,[DI] ; moves pushed key to al

cmp AL,"R" ; reset go to start of program

je Start

cmp AL,"S" ; if user has entered a character and then presses backspace then call procedure

jne StoreKey

cmp ch\_enter,00H ; if no character is entered, goes back to ReadKey

je ReadKey

call BACKSPACE

dec ch\_enter

jmp ReadKey

; AL contains the last key pressed

StoreKey:

mov entered\_IC[SI],AL ; pressed key is stored. entered\_IC[SI] = [SI + entered\_IC]

call disp\_strNext ; writes character in AL to LCD

inc ch\_enter

; IC ch\_enterber has 4 characters

cmp ch\_enter,04H

jz DispICName

jmp ReadKey

; writes ic ch\_enterber entered by user onto LCD again after clearing it

DispICName:

lea DI,entered\_IC

mov ch\_print,04H

call disp\_str

; Now checking which IC is it, or if input is wrong

isIt7408:

lea BX,IC7408

call compare\_IC\_enter ; checks if user has entered 4 ch\_enterbers

cmp FOUND,01H

jne isIt7486 ; if FOUND then check if IC is good or bad

call Check7408

jmp S4

isIt7486:

lea BX,IC7486

call compare\_IC\_enter

cmp FOUND,01H

jne isIt7432

call Check7486

jmp S4

isIt7432:

lea BX,IC7432

call compare\_IC\_enter

cmp FOUND,01H

jne NO\_IC

call Check7432

jmp S4

; invalid input

NO\_IC:

lea DI,inVALID

mov ch\_print,10 ; if no ic found then writes entered\_IC "not found" on LCD

call disp\_strNext

S4:

call KEYBRD\_PRESS

lea DI,keyboard\_keys

add DI,AX ; Take the key pressed and put it in AL

mov AL,[DI]

cmp AL,"R" ; If reset is pressed go back to start

je Start

jmp S4

.EXIT

; generates delay\_20ms of 0.25 secs

D20MS:

mov cx,2220

xn:

loop xn

ret

;generates delay\_20ms of 20ms

delay\_20ms PROC

mov CX, 1325 ;1325\*15.085 usec = 20 msec

WasteTime:

NOP

NOP

LOOP WasteTime

RET

delay\_20ms ENDP

; LCD initially displays "IC ch\_enter - "

INITIAL PROC NEAR

; initializing LCD for 2 lines & 5\*7 matrix

mov AL, 38H

call Execute\_command ;write the command to LCD

call delay\_20ms ;delay\_20ms before next command

call delay\_20ms

call delay\_20ms

; LCD ON, Show cursor

mov AL, 0EH

call Execute\_command

call delay\_20ms

; clear LCD

mov AL, 01

call Execute\_command

call delay\_20ms

; command for shifting cursor right

mov AL, 06

call Execute\_command

call delay\_20ms

RET

INITIAL ENDP

; clear the display

clear PROC

mov AL, 01

call Execute\_command

call delay\_20ms

call delay\_20ms

RET

clear ENDP

;read key pressed

KEYBRD\_PRESS PROC NEAR

pushf

push BX

push CX

push DX ; SAVING THE REGISTERS USED

mov AL,0FFH

out portc2,AL

; Checking all keyboard\_keys are open

X0: mov AL,00H

out portc2,AL

Open: in AL, portc2

and AL,0F0H

cmp AL,0F0H

jnz Open ; Means the key is still pressed, go back to X1

call D20MS ; debounce check

mov AL,00H

out portc2 ,AL ; provide column values as output through lower port C

; BL has 0 on col no. Al has 0 on row no.

findRC: in AL, portc2

and AL,0F0H

cmp AL,0F0H

jz findRC

call D20MS ;key debounce check

mov AL,00H

out portc2 ,AL ;provide column values as output through lower port C

in AL, portc2

and AL,0F0H

cmp AL,0F0H

jz findRC ;debounce check

; Checking the first column

mov AL, 0EH ;E = 1110

mov BL,AL

out portc2,AL

in AL, portc2

and AL,0F0H

cmp AL,0F0H

jnz RC

; Checking the second column

mov AL,0DH ; D = 1101

mov BL,AL

out portc2 ,AL

in AL, portc2

and AL,0F0H

cmp AL,0F0H

jnz RC

;Checking the third column

mov AL, 0BH ; B = 1011

mov BL,AL

out portc2,AL

in AL, portc2

and AL,0F0H

cmp AL,0F0H

jnz RC

; Checking the fourth column

mov AL, 07H ; 7 = 0111

mov BL,AL

out portc2,AL

in AL, portc2

and AL,0F0H

cmp AL,0F0H

jz findRC

; lower nibble of BL col no. upper nibble of AL row no.

; This converts into Row Column format to be checked from kb table and stores into Al.

RC: or AL,BL

mov CX,0FH

mov DI,00H

; compares preesed key with all keyboard\_keys in keyboard.

FindKey: cmp AL,KEY\_TABLE[DI]

jz Over

inc DI

LOOP FindKey

Over:mov AX,DI ; move the offset of key pressed to AX

pop DX

pop CX

pop BX

popf

RET

KEYBRD\_PRESS ENDP

; writes a string with starting add at DI having ch\_print characters to display

disp\_str PROC NEAR

call clear

LoopOver:

mov AL, [DI]

call DispChar ;issue it to LCD

call delay\_20ms ; delay\_20ms before next character

call delay\_20ms

inc DI ; move to next character

dec ch\_print

jnz LoopOver

RET

disp\_str ENDP

; writes command in AL to LCD

; sends the commands to port A which is connected to D0-D7 of LCD

; then enables pin from high to low with RS=0 for selecting command register and R/W = 0 for write operation.

Execute\_command PROC

mov DX, porta1

out DX, AL ; AL contains the command

mov DX, portb1

; Enable High

mov AL, 00000100B

out DX, AL

; A small pause

NOP

NOP

; Enable Low

mov AL, 00000000B

out DX, AL

RET

Execute\_command ENDP

; disp\_str without CLS.

disp\_strNext PROC NEAR

LoopOver2:

mov AL, [DI]

call DispChar ;issue it to LCD

call delay\_20ms ; delay\_20ms before next character

call delay\_20ms

inc DI ; move to next character

dec ch\_print

jnz LoopOver2

RET

disp\_strNext ENDP

BACKSPACE PROC NEAR

push DX

push AX

mov AL,00010000B ; shifts cursor to one space left

call Execute\_command

call delay\_20ms ; wait before next command

call delay\_20ms

mov AL,' '

call DispChar ; overwrite " "

call delay\_20ms

call delay\_20ms ; wait before issuing next command

mov AL,00010000B ; shifting cursor to left

call Execute\_command

pop AX ;retrive registers

pop DX

RET

BACKSPACE ENDP

; Write single character in AL to LCD

; R/W = 0 because we are writing. RS is 1 because data register is to be selected.

DispChar PROC

push DX

mov DX,porta1 ; DX=port A address

out DX, AL ; issue the char to LCD

mov AL, 00000101B

mov DX, portb1 ;port B address

out DX, AL

mov AL, 00000001B

out DX, AL

pop DX

RET

DispChar ENDP

; Compares BX and entered\_IC for equality

compare\_IC\_enter PROC NEAR

mov SI,0000H

cmp\_ch\_enter:

mov AL,entered\_IC[SI]

cmp AL,[BX+SI]

je NXT\_ch\_enter

jmp EP\_cmp\_IC

NXT\_ch\_enter:

cmp SI,03H

je PASS\_cmp\_IC ; If all chars equal set FOUND to 1 before return

inc SI

jmp cmp\_ch\_enter

PASS\_cmp\_IC:

mov FOUND,01H

EP\_cmp\_IC:

RET

compare\_IC\_enter ENDP

Check7408 PROC NEAR ;checks if 7408 is good or bad[pass or fail]

mov DI,00H

Testing7408:

mov AL,in7408[DI]

out porta2,AL ; input for the 7408 IC

in AL,portb2 ; output value of 7408 goes to AL

and AL,0FH ; Masking. Only lower nibble needed.

cmp AL,out7408[DI] ; Verify by comparing with expected output.

je Next7408

call FAIL

jmp Ret7408

Next7408:

cmp DI,03H ; All four chars are same

je Pass7408 ; Means Pass.

inc DI ; If all chars are not read yet, proceed to next char

jmp Testing7408

Pass7408:

call PASS

Ret7408:

RET

Check7408 ENDP

Check7432 PROC NEAR ;checks if 7432 is good or bad[pass or fail]

mov DI,00H

Testing7432:

mov AL,in7432[DI]

out porta2,AL ; input for the 7432 IC

in AL,portb2 ; output value of 7432 goes to AL

and AL,0FH ; Masking. Only lower nibble needed.

cmp AL,out7432[DI] ; Verify by comparing with expected output.

je Next7432

call FAIL

jmp Ret7432

Next7432:

cmp DI,03H ; All four chars are same

je Pass7432 ; Means Pass.

inc DI ; If all chars are not read yet, proceed to next char

jmp Testing7432

Pass7432:

call PASS

Ret7432:

RET

Check7432 ENDP

Check7486 PROC NEAR ;checks if 7486 is good or bad[pass or fail]

mov DI,00H

Testing7486:

mov AL,in7486[DI]

out porta2,AL ; input for the 7486 IC

in AL,portb2 ; output value of 7486 goes to AL

and AL,0FH ; Masking. Only lower nibble needed.

cmp AL,out7486[DI] ; Verify by comparing with expected output.

je Next7486

call FAIL

jmp Ret7486

Next7486:

cmp DI,03H ; All four chars are same

je Pass7486 ; Means Pass.

inc DI ; If all chars are not read yet, proceed to next char

jmp Testing7486

Pass7486:

call PASS

Ret7486:

RET

Check7486 ENDP

; When IC check fails

FAIL PROC NEAR

pushf

push DI

mov ch\_print,05

lea DI,ICBAD ; ' FAIL'

call disp\_strNext ; Writes next to the IC ch\_enterber

pop DI

popf

RET

FAIL ENDP

; When the IC passes the test

PASS PROC NEAR

pushf

push DI

mov ch\_print,05 ; ch\_enterber of letters

lea DI,ICGOOD ; ' PASS'

call disp\_strNext ; Writes next to the IC ch\_enterber

pop DI

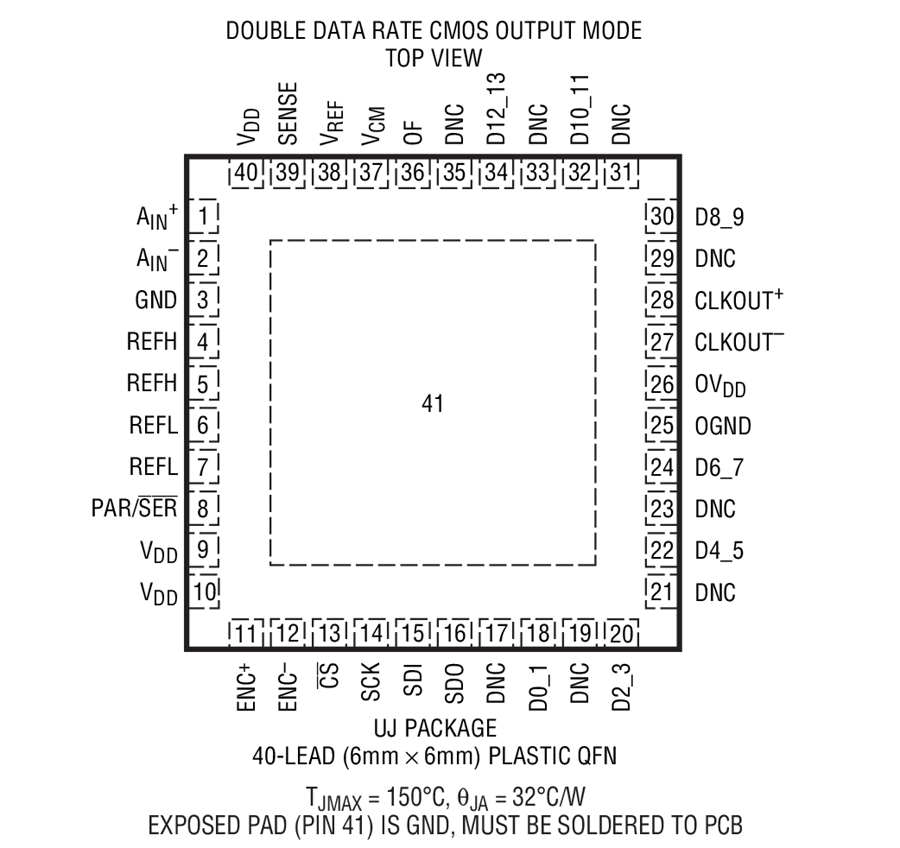
popf

RET

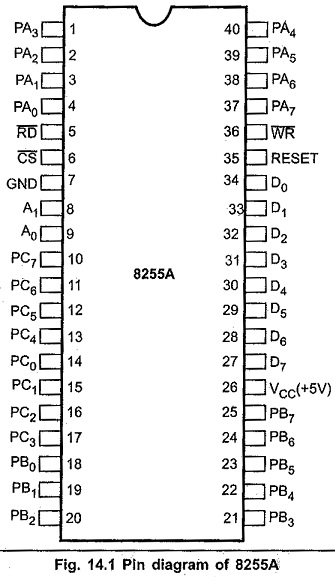
PASS ENDP

END

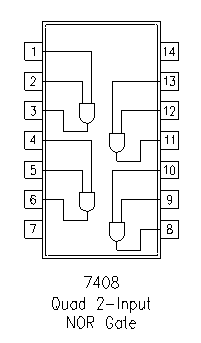
**DATASHEETS**

**8206** 

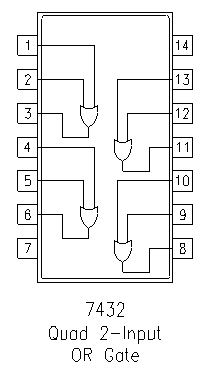
**8255**



**7408**



**7432**



**7486**

