BIRLA INSTITUTE OF TECHNOLGY AND SCIENCE – PILANI
GOA CAMPUS

A
REPORT ON

Microprocessor Based
RAM Tester

In partial fulfillment of the course MICROPROCESSOR PROGRAMMING AND INTERFACING

Prepared for:

Dr. K.R. Anupama
INSTRUCTOR-IN-CHARGE
MICROPROCESSOR PROGRAMMING AND INTERFACING

Prepared By:

ADITYA BHANDARI	**2018A7PS0805G**
JASH MEHTA	**2018A8PS0746G**
MANHAL RAHMAN	**2018A3PS0310G**
NILESH RAGHUVANSHI	**2018A8PS0415G**
SHOBHIT MEHTA	**2018A8PS0417G**

Table of Contents

ACKNOWLEDGEMENT

We would like to express our sincere thanks and gratitude to Dr. K. R. Anupama, instructor-in-charge of the course, for providing such an opportunity to students, to hone their programming skills and apply it in a real time situation. Such an application has helped us gain knowledge of the principles of microprocessor interfacing and hardware programming, which has been applied at various stages of development of this project.

We are also indebted to other instructors involved in this course for guiding us during the whole project. The project has given us a great insight into the depths of Microprocessor Programming and Interfacing. It helped us in practically applying the major principles of interfacing viz. memory interfacing, and I/O interfacing. Finally we extend our thanks to friends for their continuous support and encouragement.

PROBLEM STATEMENT

RAM Tester

Design a microprocessor-based RAM tester. The tester should be able to test 6164 RAM chips. The tester tests each bit of the RAM individually. For a byte of RAM, the first bit (D0) is written as zero and read back, now a one is written into the bit and again it is read back. If the two read operations result in bit D0 to contain a zero and one respectively then the bit is inferred as good. Any other result indicates a faulty bit. The test is repeated for all bits of a byte and for all bytes of the RAM. The summary result, PASS/FAIL should be displayed. User will place the 6164 chip in the zip socket, then press a switch labelled TEST. The RAM is tested and the result is displayed on the 7-segemnt display as PASS/FAIL.[Pl note RAM to be tested cannot be directly connected to system bus of 8086 in case the RAM chip is damaged – the RAM here has to be treated as an IC under test- and corresponding interfacing has to be done].

ASSUMPTION

- Once the switch is pressed for testing it should remain high until PASS/FAIL is displayed on the 7 Segment LED'S. If a new RAM chip needs to be tested then the test switch should be made LOW and new RAM chip should be placed in the zip socket. After this, the switch is pressed again to begin the testing for the new chip.

I/O MAPPING

8255$_{(1)}$	00-06$_{H}$
8255$_{(2)}$	08-0E$_{H}$
8255$_{(3)}$	10-16$_{H}$

1. 8255 :Interfacing the I/O devices

Base Address: 00H

The addresses of the ports are as follows

NAME OF THE PORTS OF 8255	ADDRESS
Port A	00H
Port B	02H
Port C	04H
Control register	06H

Data lines: D0-D7 data lines of the microprocessor(as it is connected in even bank)

Port Specification:

Group A: Mode 0

Group B: Mode 0

Port A: Input

Port C: Output

Port B: Output

Hence, the control word is

1	0	0	1	0	0	0	0

i.e. 90H which is written to the control register.

Port A is used for the input (POLLING).

Port B is used to give the data for the output of the status message on the seven segment displays. It gives the code for the LED’s to be on or off.

Port C is used for controlling the latches. By setting and resetting its pins using Bit Set Reset(BSR) mode latches are selected, and corresponding display is given its data and starts glowing.

2. 8255 Interfacing the RAM to be checked

Base Address: 08H

The addresses of the ports are as follows

NAME OF THE PORTS OF 8255	ADDRESS
Port A	08H
Port B	0AH
Port C	0CH
Control register	0EH

Data lines: D0-D7 data lines of the microprocessor(as it is connected in even bank)

Port Specification:

Group A: Mode 0

Group B: Mode 0

Port A: Output

Port C: Output

Port B: ISN'T CONNECTED

Hence, the control word is

1	0	0	0	0	0	0	0

i.e. 80H which is written to the control register.

Port A specifies the UPPER bits of the address

Port B specifies the LOWER bits of the address, including CE,WR,RD.

3. 8255 Interfacing the RAM to be checked

Base Address: 10H

The addresses of the ports are as follows

NAME OF THE PORTS OF 8255	ADDRESS
Port A	10H
Port B	12H
Port C	14H
Control register	16H

Data lines: D0-D7 data lines of the microprocessor(as it is connected in even bank)

Port Specification:

a) **Writing to the memory**

Group A: Mode 0

Group B: Mode 0

Port A: : ISN’T CONNECTED

Port B: : ISN’T CONNECTED

Port C : : OUTPUT

Hence, the control word is

1	0	0	1	0	0	1	0

i.e. 92H which is written to the control register.

Port A specifies the data that is to be written into the memory location.

b) Reading to the memory

Group A: Mode 0

Group B: Mode 0

Port A: : ISN’T CONNECTED

Port B: : ISN’T CONNECTED

Port C : : INPUT

Hence, the control word is

1	0	0	1	0	0	1	1

i.e. 93H which is written to the control register.

Port A specifies the data that is to be READ into the memory location.

MEMORY MAPPING

RAM1 (even)	01000 - 01FFFE
RAM1 (odd)	01001 - 01FFFF
ROM1 (even)	00000 - 007FE
ROM1 (odd)	00001 - 007Ff
ROM2 (even)	FF000 - FF7FE
ROM1 (odd)	FF001 - FF7FF

List of Components used

CHIP NO.	QUANTITY	CHIP	PURPOSE
8086	1	Microprocessor	Central Processing Unit
8255	3	Programmable Peripheral Interface (PPI) with 24 I/O lines	To control the I/O devices and interfaces the CPU to the outside world
6164	1	8K SRAM	For storing samples
6116	2	2K SRAM	For storing samples and memory
2716	4	2K EPROM	For IVT and RESET
74LS138	2	3 Line to 8 Line Decoder	One to select between 8255's and other one to select between RAM and ROM for memory
74LS373	3	Octal D-type Transparent Latches with 3 state outputs	To latch outputs
74LS245	2	Octal Bus Transceivers with tristate output/Bidirectional Buffer	To strengthen signals
8284	1	5MHz Clock	Input clock to 8086

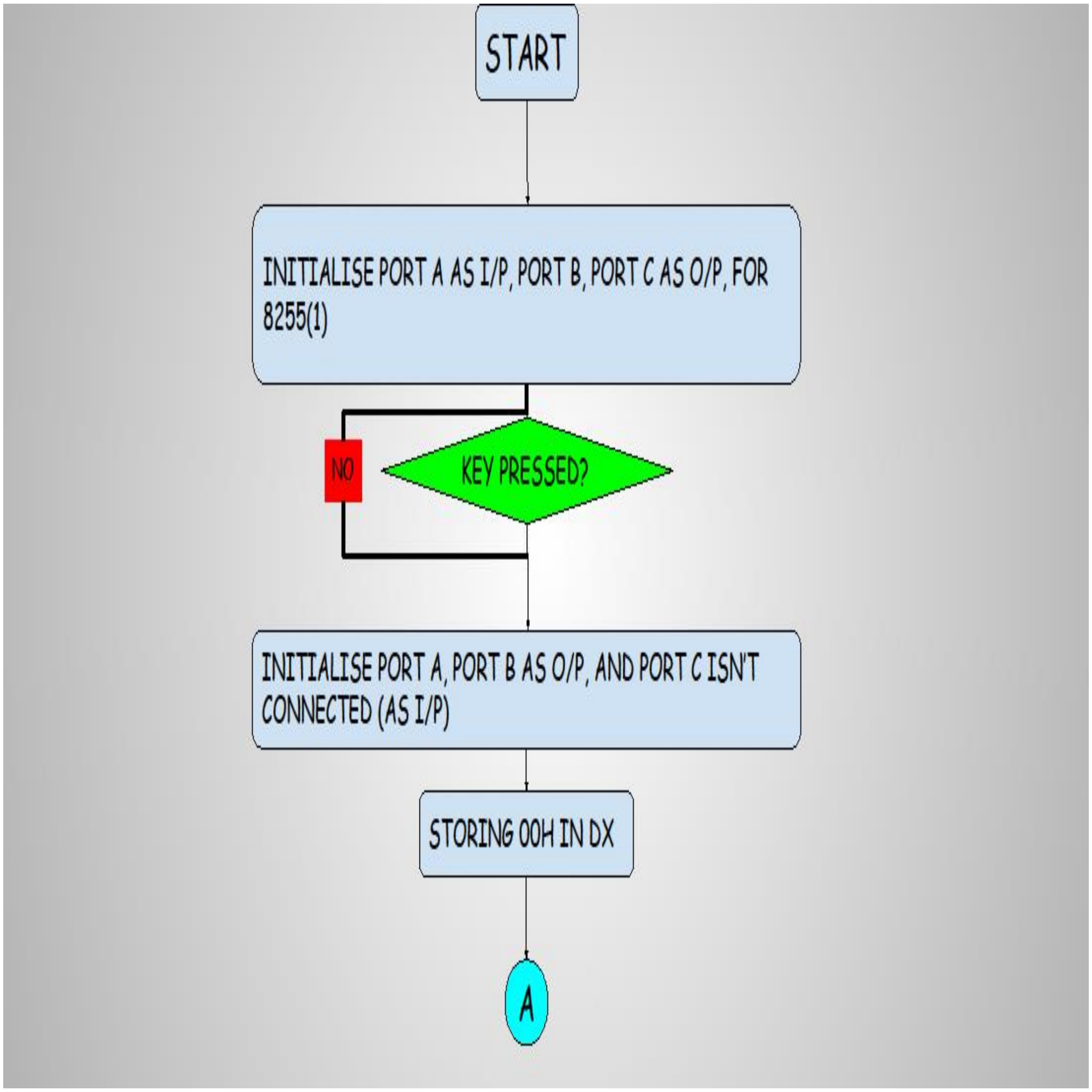
Other Hardware

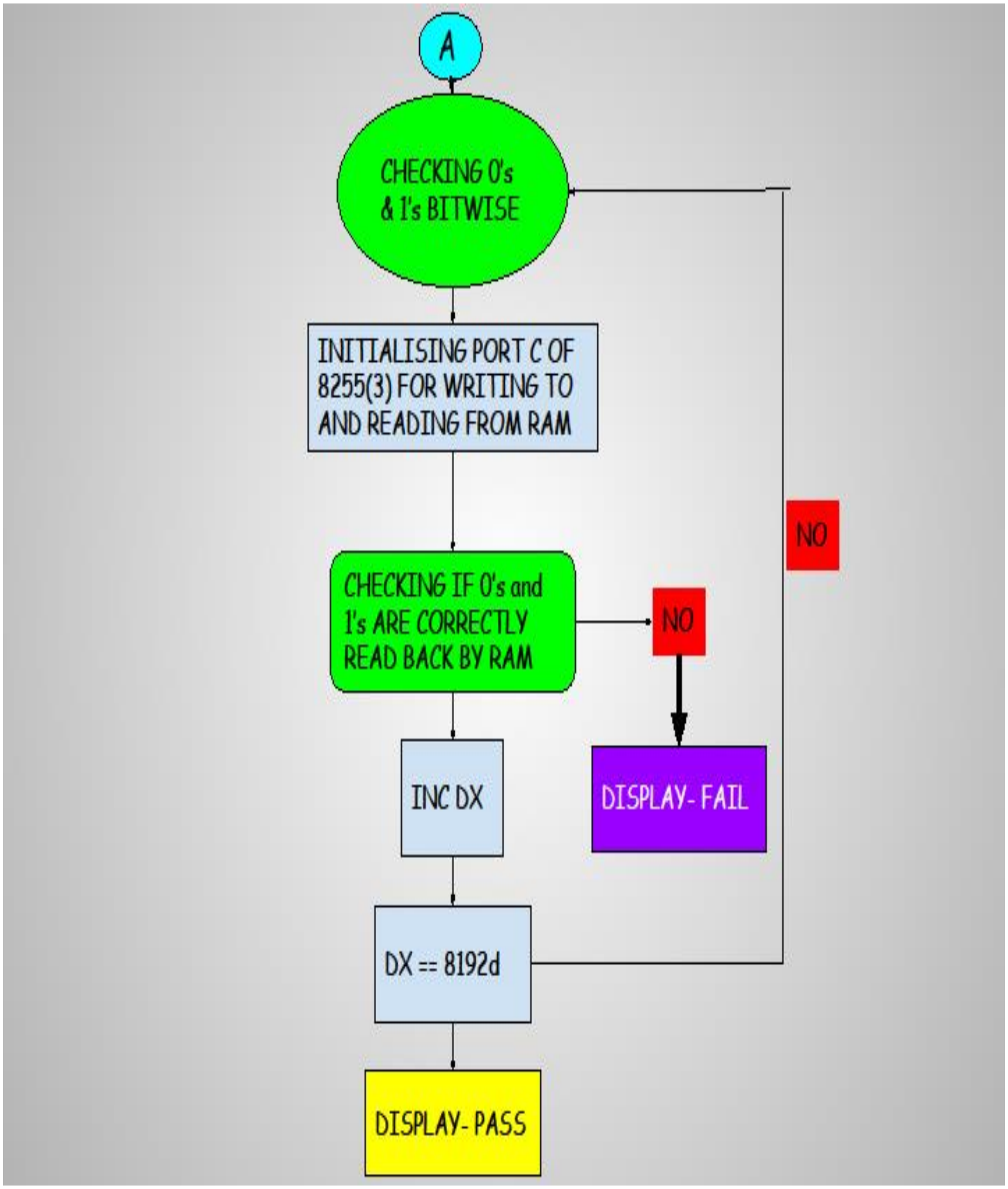
1. **Logic Gates**: Simple 2-input OR gates (7432) (Quantity-6)
   NOT gates (7404) (Simple Digital Inverter) (Quantity-3)
   They are primarily used in decoding logic for memory and I/O interfacing.

2. **Switches**: Interactive SW-SPDT Latched Action (Single Pole Double Throw Switch)
   (Quantity-2)
   One input terminal and two output terminals used to turn ON/OFF the circuit.

3. **LED**: 7 segment-MPX1-CA
   Used Red,1-digit, Common Anode, 7-segment display (Quantity-4).

FLOWCHART


START
INITIALISE PORT A AS I/P, PORT B, PORT C AS O/P, FOR 8255(1)
NO
KEY PRESSED?
INITIALISE PORT A, PORT B AS O/P, AND PORT C ISN'T CONNECTED (AS I/P)
STORING 00H IN DX
A


A
CHECKING 0's & 1's BITWISE
INITIALISING PORT C OF 8255(3) FOR WRITING TO AND READING FROM RAM
CHECKING IF 0's and 1's ARE CORRECTLY READ BACK BY RAM
NO
DISPLAY- FAIL
INC DX
DX == 8192d
NO
DISPLAY- PASS

DESIGN

Complete design shown with proper labeling(Design attached).

VARIATION IN PROTEUS IMPLEMENTATION

- EROM in proteus is present only at 00000h as proteus allows to change the reset address.
- 2732 is used in proteus is 6264 as 6164 is unavailable. The difference here is the chip enable in proteus is CE’ while in the On paper design it is CS1’, other connections are the same.
- Clock is at 2MHz as the clock generated for 8086 requires a long rise and time of clock.

CODE

```
#LOAD_SEGMENT=FFFFh#

#LOAD_OFFSET=0000h#

#CS=0000h#

#IP=0000h#

#DS=0000h#

#ES=0000h#

#SS=0000h#

#SP=FFFEh#

#AX=0000h#

#BX=0000h#

#CX=0000h#

#DX=0000h#

#SI=0000h#

#DI=0000h#

#BP=0000h#

; this header has to remain
; add your code here

                  jmp    st1

;proteus allows you to change the reset address - hence changing it to
00000H - so every time
;system is reset it will go and execute the instruction at address 00000H -
which is
```

```
                jmp st1

               db     1021 dup(0)

;jmp st1 will take up 3 bytes in memory - another 509 bytes are filled with
'0s'
;1021 + 3 bytes = 1024 bytes
;first 1 k of memory is IVT - 00000 -00002H will now have the jmp
instruction. 00003H - 001FFH will
;have 00000 - as vector number 0 to 79H are unused
;IVT entry for 80H - address for entry is 80H x 4 is 00200H
;code segment will be in ROM

st1:            cli

; intialize ds, es,ss to start of RAM - that is 020000H - as you need r/w
capability for DS,ES & SS
; pl note you cannot use db to store data in the RAM you have to use a
MOV instruction.
; so if you want to do dat1 db 78H - you have to say something like
; dat1 equ 0002h
; mov al,78h
; mov dat1,al
;0002H is the offset in data segmnet where you are storing the data.
;db can be used only to store data in code segment

                mov      ax,01000h

                mov      ds,ax

                 mov      es,ax

                mov      ss,ax

                mov      sp,0FFFEH

; intialise portA as input portB,portC as output for the first 8255

                mov      al,90h     ;1st

                out     06h,al

;Keep polling port A until you get 1 from the switch
```

```
POLLING :       in   al, 00h

                mov       bl,01h

                cmp       bl,al

                jnz POLLING

 TRUE:

; initialize portA,B as output and port C isnt connected

                mov       al,10000000b       ;2nd

                out       0Eh,al

                mov dx,00h

START:

               mov ch,11111110b

               mov bh,00000001b

               mov cl,08h

 ;for reading and writing 0's

 START1:

               mov al,dl

               out 08h,al

              mov al,dh

              and al,10111111b      ;Turn on write enable

              or  al,00100000b      ;Turn off read enable!

              and  al,01111111b      ;Turn on CE!

              out 0Ah,al
```

```
                mov      al,10010010b
;Initialize 3rd 8255 to write data ;So port C in output mode
                 out      16h,al
                 mov al,ch    ; Write from Port C of 3rd 8255
                 out 14h,al
                 mov al,dh
                 and al,11011111b      ;Turn on read enable!
                  or  al,01000000b      ;Turn off write enable dash!
                 and  al,01111111b     ;Turn on CE!
                  out 0Ah,al
                 mov      al,10011011b
;Initialize 3rd 8255 to read data : So port C in input mode
                out      16h,al
                in al,14h
                mov bl,ch
                cmp al,bl
                jnz FAIL
  ;for reading and writing 1's
               mov al,dl
               out 08h,al
               mov al,dh
               and al,10111111b      ;Turn on write enable dash!
               or  al,00100000b      ;Turn off read enable!
```

```
                 and  al,01111111b     ;Turn on CE!
              out 0Ah,al
              mov      al,10010010b
;Initialize 3rd 8255 to write data ;So port C in output mode
               out      16h,al
               mov al,bh     ;Write from Port C of 3rd 8255
               out 14h,al
               mov al,dh
               and al,11011111b      ;Turn on read enable!
               or  al,01000000b      ;Turn off write enable dash!
               and  al,01111111b     ;Turn on CE!
               out 0Ah,al
               mov      al,10011011b
;Initialize 3rd 8255 to read data : So port C in input mode
              out      16h,al
              in al,14h
              mov bl,bh
              cmp al,bl
              jnz FAIL

              ROL ch,1
              ROL bh,1
              dec cl
              jnz START1
```

```
inc dx
cmp dx,8192d
jz PASS
jmp  START
;Fail on LED
FAIL:           mov al,0FFh
                out 04,al
                mov al,01h
                out 02h,al
                mov al,8eh      ;For F
                out 04h,al
                mov al,02h
                out 02h,al
                mov al,88h      ;For a
                out 04h,al
                mov al,00
                out 02h,al
                mov al,0ffh
                out 04h,al
                 mov al,04h
                out 02h,al
                mov al,0F9h      ;For I
```

```
            out 04h,al
            mov al,08h
            out 02h,al
            mov al,0c7h       ;For L
            out 04h,al
            mov al,00
           out 02h,al
            mov al,0ffh
            out 04h,al
;this for the recurssion of the program
            in      al,00h
            mov     bl,01h
            cmp     bl,al
            jz TRUE
            jmp FAIL
;Pass on LED
PASS:       mov al,0FFh
            out 04,al
            mov al,01h
            out 02h,al
            mov al,8ch      ;For P
            out 04,al
```

```
    mov al,02h

    out 02h,al

    mov al,88h      ;For A

    out 04,al

    mov al,04h

    out 02h,al

    mov al,92h            ;for S

    out 04,al

    mov al,08h

    out 02h,al

    mov al,92h            ;for S

    out 04,al

    mov al,00

    out 02h,al

    mov al,0ffh

    out 04h,al

;this for the recurssion of the program

    in      al,00h

    mov     bl,01h

    cmp     bl,al

    jz TRUE

    jmp PASS
```