FELISARTA, GERMAN III E.   16101002   CpE3104 Grp 1

I. IVT

IR1	00212H	Segment High/Low(81H)
	00208H	Offset High/Low (81H)
IR0	00204H	Segment High/Low(80H)
	00200H	Offset High/Low (80H)

II. Address Decoding

Memory Address
32k Memory = 2^32
= 32, 767
= 7FFF
= 0111 1111 1111 1111
= 0000 0000 0000 0000 0000 0xxx xxxx xxxx xxxx

I/O Address
Address Range = C0H - FEH
= 3EH
= 0011 1110
= 0000 0000 0000 xxxx xxxx

III. I/O and Memory Mapping

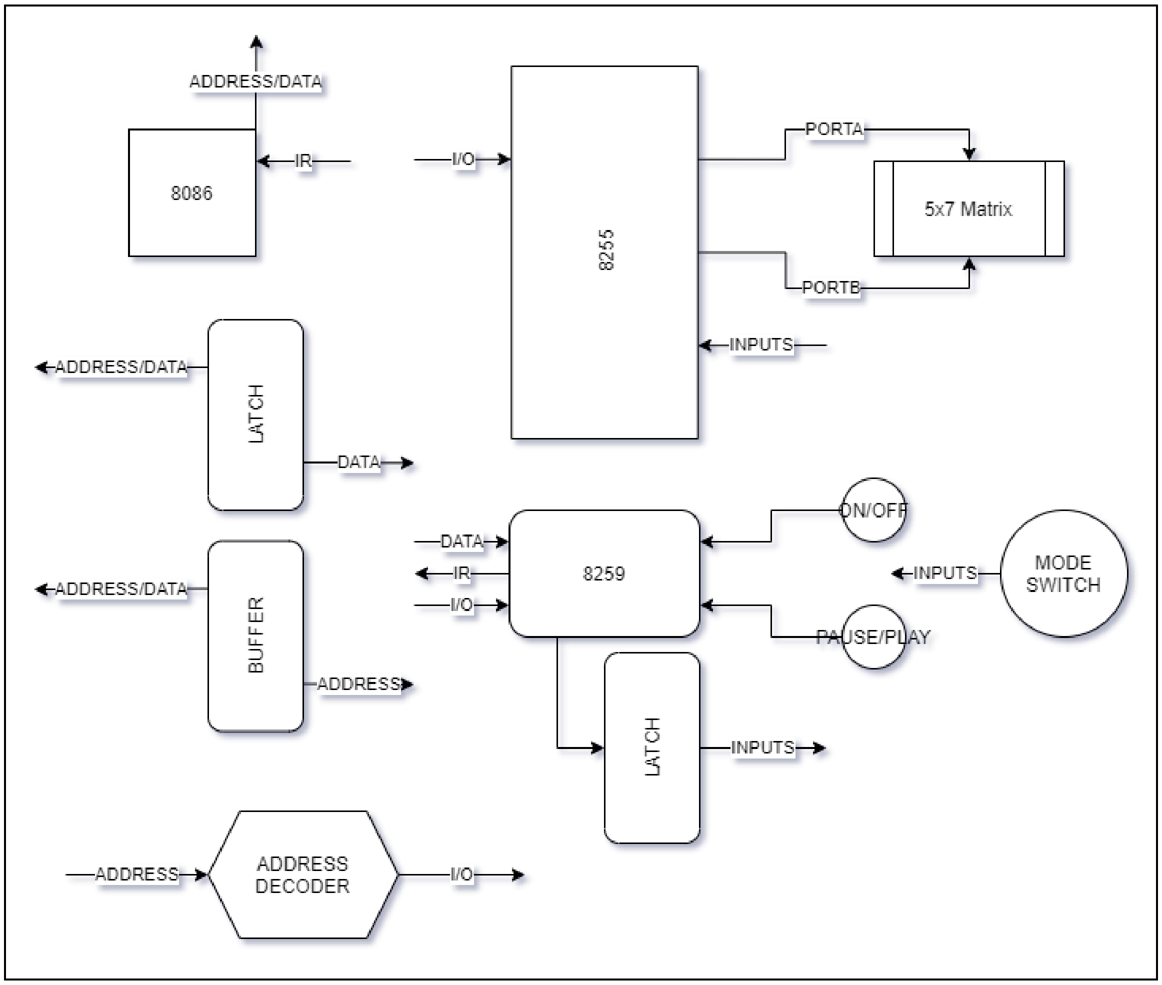
Isolated I/O = F0H - F6H

PORT A - OUTPUT	- F0H
PORT B - OUTPUT	- F2H
PORT C - INPUTS	- F4H

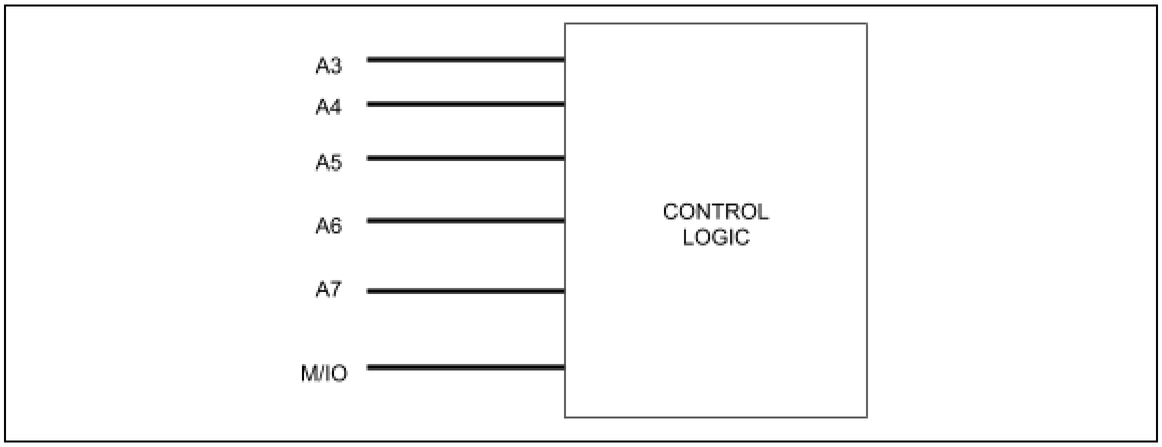
Interrupts = F8H - FEH
IR0 - 0F8H
IR1 - 0FAH

FELISARTA, GERMAN III E. 16101002 CpE3104 Grp 1

IV. Circuit Diagram

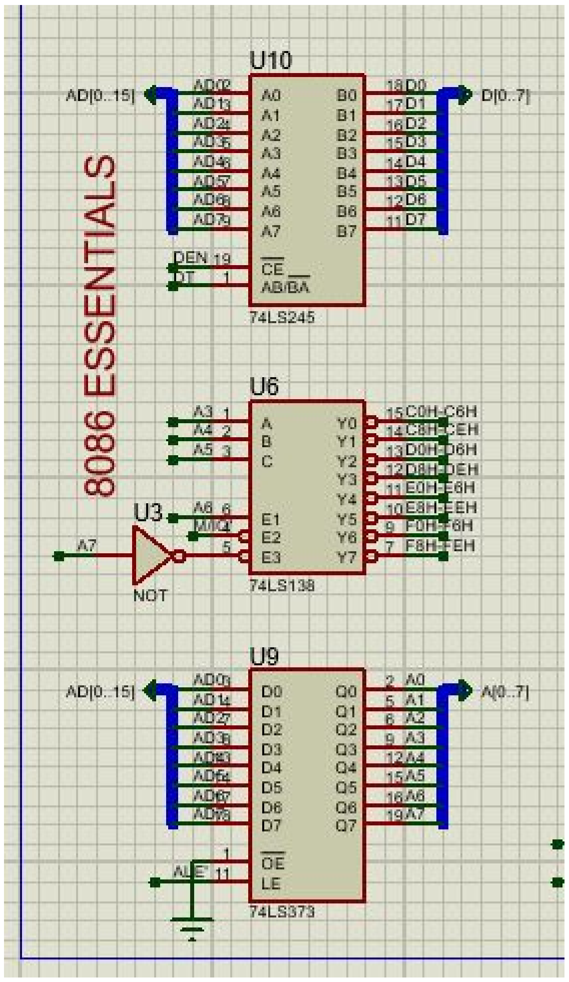


V. Control Logic Design



FELISARTA, GERMAN III E. 16101002 CpE3104 Grp 1

VI. Data Buffering and Latching

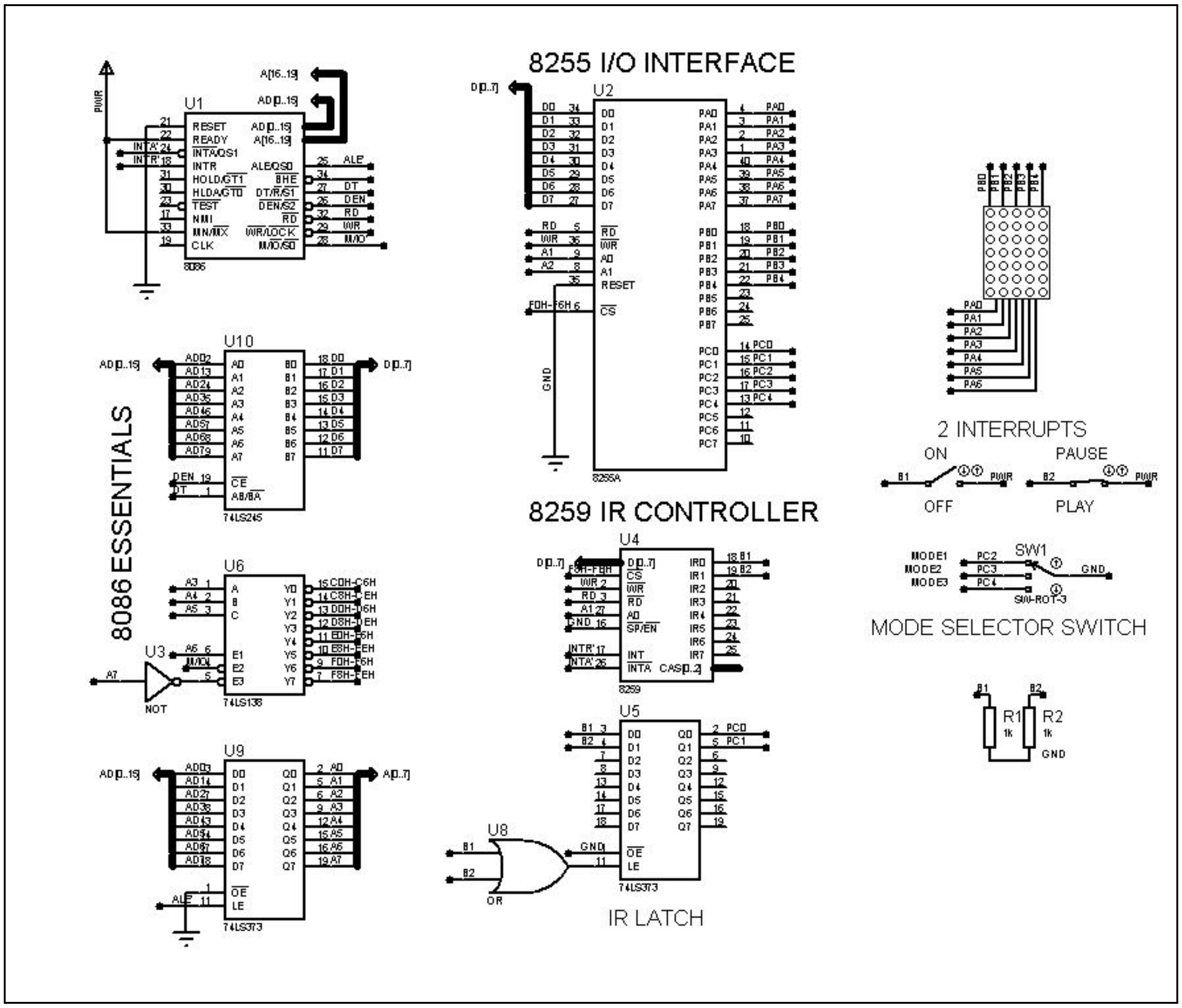


INPUT BUFFER

DECODER

ADDRESS LATCHES'

VII. Circuit Schematic



VIII. Source Code

```
;=========================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Sat Dec5 2020
; Processor: 8086
; Compiler:  MASM32
;
; Before starting simulation set Internal Memory Size
; in the 8086 model properties to 0x10000
;
;         GERMAN E FELISARTA III 16101002         CpE3104 Grp 1
```

```
;
;==========================================================================

PROCED1 SEGMENT
ISR1 PROC FAR
ASSUME CS:PROCED1, DS:DATA  ; ON/OFF INTERRUPT
                            ; TURNS OFF ONLY WHEN A MODE IS NOT ACTIVATED
ORG 08000H                  ; write code within below starting at address 08000H

    PUSHF           ; push 16-bit operands
    PUSH AX          ; save program context
    PUSH DX

      ON_OFF:
       MOV DX, PORTA        ; display '9' on the 7-segment in PORTA
       MOV AL, 00H
       OUT DX, AL

       MOV DX, PORTB        ; display '9' on the 7-segment in PORTA
       MOV AL, 00H
       OUT DX, AL

       MOV DX, PORTC
       IN AL, DX
       AND AL, 01H
       CMP AL, 01H
       JE ON_OFF

      EXIT:
       POP DX            ; retrieve program context
       POP AX
       POPF             ; pop 16-bit operands
       IRET              ; return from interrupt

ISR1 ENDP               ; end of procedure
PROCED1 ENDS

PROCED2 SEGMENT
ISR2 PROC FAR
ASSUME CS:PROCED2, DS:DATA  ; PAUSE/PLAY INTERRUPT
                            ; RETAINS THE CURRENT STATE UNTIL PRESSED AGAIN
ORG 09000H                  ; write code within below starting at address 09000H

    PUSHF           ; push 16-bit operands
    PUSH AX          ; save program context
    PUSH DX

    PAUSE:
       MOV DX, PORTC
       IN AL, DX
```

```
          AND AL, 02H
          CMP AL, 02H
          JE EXIT
         JMP PAUSE

         EXIT:
      POP DX          ; retrieve program context
      POP AX
      POPF            ; pop 16-bit operands
      IRET            ; return from interrupt

ISR2 ENDP             ; end of procedure
PROCED2 ENDS

DATA SEGMENT
ORG 0F000H

   PORTA   EQU 0F0H    ; PORTA address
   PORTB   EQU 0F2H    ; PORTB address
   PORTC   EQU 0F4H    ; PORTC address
   COM_REG EQU 0F6H    ; Command Register Address
   PIC1    EQU 0F8H    ; A1 = 0
   PIC2    EQU 0FAH    ; A1 = 1
   ICW1    EQU 013H    ; 8259 command word ICW1
   ICW2    EQU 080H    ; 8259 command word ICW2
   ICW4    EQU 03H     ; 8259 command word ICW4
   OCW1    EQU 0FCH    ; 8259 command word OCW1

   BLOCK DB 1111111B, 1111110B, 1111101B      ; blocks falling
         DB 1111011B, 1110111B, 1101111B
         DB 1011110B, 0111100B, 0111001B
         DB 0110011B, 0100110B, 0001101B
         DB 0001011B, 0000111B, 0000110B
         DB 0000101B, 0000011B, 0000010B
         DB 0000001B, 0000000B, '$'

   TEXT DB 0FFH, 0FFH, 0FFH, 0FFH, 0FFH ; SPACE
        DB 000H, 077H, 077H, 077H, 000H, 0FFH ; H
        DB 03EH, 000H, 03EH, 0FFH, 0FFH, 0FFH ; I [space]
        DB 03EH, 000H, 03EH ; I
        DB 0FFH, 0F6H, 0F8H, 0FFH ; APOSTROPHE
        DB 000H, 0FDH, 0FBH, 0FDH, 000H, 0FFH, 0FFH, 0FFH ; M SPACE
        DB 041H, 03EH, 036H, 045H, 0FFH ; G
        DB 000H, 0FDH, 0FBH, 0FDH, 000H, 0FFH ; M
        DB 003H, 0EDH, 0EEH, 0EDH, 003H, 0FFH ; A
        DB 000H, 0F9H, 0E7H, 09FH, 000H ; N
        DB 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, '$' ; SPACE

DATA ENDS
```

FELISARTA, GERMAN III E.   16101002   CpE3104 Grp 1

```
STK SEGMENT STACK
   BOS DW 64D DUP(?)   ; stack depth (bottom of stack)
   TOS LABEL WORD      ; top of stack
STK ENDS

CODE SEGMENT PUBLIC 'CODE'
ASSUME CS:CODE, DS:DATA, SS:STK

   ORG 0E000H         ; write code within below starting at address 0E000H

   START:
      MOV AX, DATA
      MOV DS, AX      ; set the data segment address
      MOV AX, STK
      MOV SS, AX      ; set the stack segment address
      LEA SP, TOS     ; set the address of SP as top of stack

      CLI             ; clears IF flag

      ; program the 8255
      MOV DX, COM_REG
      MOV AL, 10001001B
      OUT DX, AL
      MOV DX, PORTA
      MOV AL, 00H
      OUT DX, AL

      ; program the 8259
      MOV DX, PIC1    ; set the I/O address to access ICW1
      MOV AL, ICW1
      OUT DX, AL      ; send command word
      MOV DX, PIC2    ; set the I/O address to access ICW2, ICW4 and OCW1
      MOV AL, ICW2
      OUT DX, AL      ; send command word
      MOV AL, ICW4
      OUT DX, AL      ; send command word
      MOV AL, OCW1
      OUT DX, AL      ; send command word
      STI             ; enable INTR pin of 8086

      ; storing interrupt vector to interrup vector table in memory
      MOV AX, OFFSET ISR1 ; get offset address of ISR1(IP)
      MOV [ES:200H], AX   ; store offset address  to memory
      MOV AX, SEG ISR1    ; get segment address of ISR1 (CS)
      MOV [ES:202H], AX   ; store segment address to memory


      MOV AX, OFFSET ISR2 ; get offset address of ISR2 (IP)
      MOV [ES:204H], AX   ; store offset address to memory
      MOV AX, SEG ISR2    ; get segment address of ISR2 (CS)
```

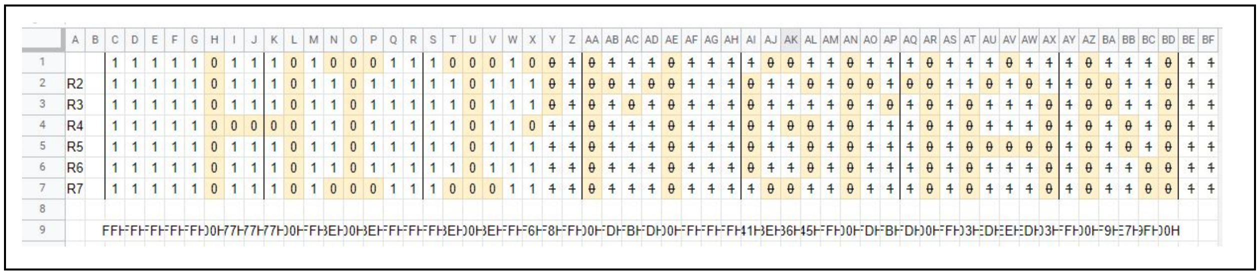
```
    MOV [ES:206H], AX   ; store segment address to memory


  ; foreground routine
  HERE:
        MOV DX, PORTC
        IN AL, DX

        AND AL, 11111010B ;checks if mod1 button is pressed and also pause is on
        CMP AL, 11111010B
        JE MOD1

        IN AL, DX
        AND AL, 11110110B ;checks if mod2 button is pressed and also pause is on
        CMP AL, 11110110B
        JE MOD2

        IN AL, DX
        AND AL, 11101110B ;checks if mod3 button is pressed and also pause is on
        CMP AL, 11101110B
        JE MOD3

        JMP RESTART

  MOD1:                   ; displays HI I'M GMAN
   LEA SI, TEXT

        DISP_LOOP:        ; loops the display algorithm (idea from hint video)
          MOV AL, [SI+4]  ; CHECKS if the fifth character is end, if it is then exits.
          CMP AL, '$'
          JE MOD1EXIT

          MOV DX, PORTB   ; prints the first column
          MOV AL, 00001B
          OUT DX, AL
          MOV DX, PORTA
          MOV AL, BH
          MOV AL, [SI]
          OUT DX, AL
          CALL DELAY2

          MOV DX, PORTB   ; prints the second column
          MOV AL, 00010B
          OUT DX, AL
          MOV DX, PORTA
          MOV AL, BH
          MOV AL, [SI+1]
          OUT DX, AL
          CALL DELAY2
```

```
            MOV DX, PORTB   ; prints the third column
            MOV AL, 00100B
            OUT DX, AL
            MOV DX, PORTA
            MOV AL, BH
            MOV AL, [SI+2]
            OUT DX, AL
            CALL DELAY2

            MOV DX, PORTB   ; prints the fourth column
            MOV AL, 01000B
            OUT DX, AL
            MOV DX, PORTA
            MOV AL, BH
            MOV AL, [SI+3]
            OUT DX, AL
            CALL DELAY2

            MOV DX, PORTB   ; prints the fifth column
            MOV AL, 10000B
            OUT DX, AL
            MOV DX, PORTA
            MOV AL, BH
            MOV AL, [SI+4]
            OUT DX, AL
            CALL DELAY2

          UPSI:
            INC SI
            JMP DISP_LOOP

     MOD1EXIT:
          ; CHECK FOR LOOP OR EXIT
            MOV DX, PORTC
            IN AL, DX
            AND AL, 11111010B              ;checks if mod1 button is pressed and also pause
is on
            CMP AL, 11111010B
            JE MOD1
            AND AL, 00000010B              ;checks if play is pressed
            CMP AL, 00000010B
            JE MOD1
            JMP RESTART

    MOD2:       ; BLOCK FALLING
          LEA SI, BLOCK
          MOV BL, '$'

          LPRINT:
            MOV DX, PORTB                  ; first row
            MOV AL, 11111B
            OUT DX, AL
            MOV DX, PORTA                        ; first row
            MOV AL, [SI]
            OUT DX, AL

            CALL DELAY

            INC SI
            CMP [SI], BL
            JE LMOD2
            JMP LPRINT

          LMOD2:
            ; CHECK FOR LOOP OR EXIT
            MOV DX, PORTC
            IN AL, DX
            AND AL, 11101110B                    ;checks if mod2 button is pressed and also pause
is on
            CMP AL, 11101110B
            JE MOD2
            AND AL, 00000010B                    ;checks if play is pressed
            CMP AL, 00000010B
            JE MOD2
            JMP RESTART

    MOD3:       ;DISCO DISCO WOOT WOOT
          MOV DX, PORTA                 ;PATTERN 1
          MOV AL, 10101010B
          OUT DX, AL
          MOV DX, PORTB
          MOV AL, 10101B
          OUT DX, AL

          CALL DELAY

          MOV DX, PORTA                 ; PATTERN 2
          MOV AL, 10110011B
          OUT DX, AL
          MOV DX, PORTB
          MOV AL, 11011B
          OUT DX, AL

          CALL DELAY

          MOV DX, PORTA                 ; PATTERN 3
          MOV AL, 01010101B
          OUT DX, AL
          MOV DX, PORTB
          MOV AL, 01010B
```

```
        OUT DX, AL

        CALL DELAY


         ; CHECK FOR LOOP OR EXIT
        MOV DX, PORTC
        IN AL, DX
        AND AL, 11101110B ;checks if mod3 button is pressed
        CMP AL, 11101110B
        JE MOD3
        AND AL, 00000010B ;checks if play is pressed
        CMP AL, 00000010B
        JE MOD3
        JMP RESTART

  RESTART:
        JMP HERE

DELAY PROC NEAR                ; TIME DELAY (optional)
  MOV CX, 0FFFFh
  DELAY_LOOP:
    DEC CX
    CMP CX, 00H
    JNZ DELAY_LOOP
  RET
DELAY ENDP

DELAY2 PROC NEAR                       ; TIME DELAY (optional)
  MOV CX, 0FFFh
  DELAY_LOOP:
    DEC CX
    CMP CX, 00H
    JNZ DELAY_LOOP
  RET
DELAY2 ENDP


CODE ENDS
END START
```

FELISARTA, GERMAN III E.   16101002   CpE3104 Grp 1

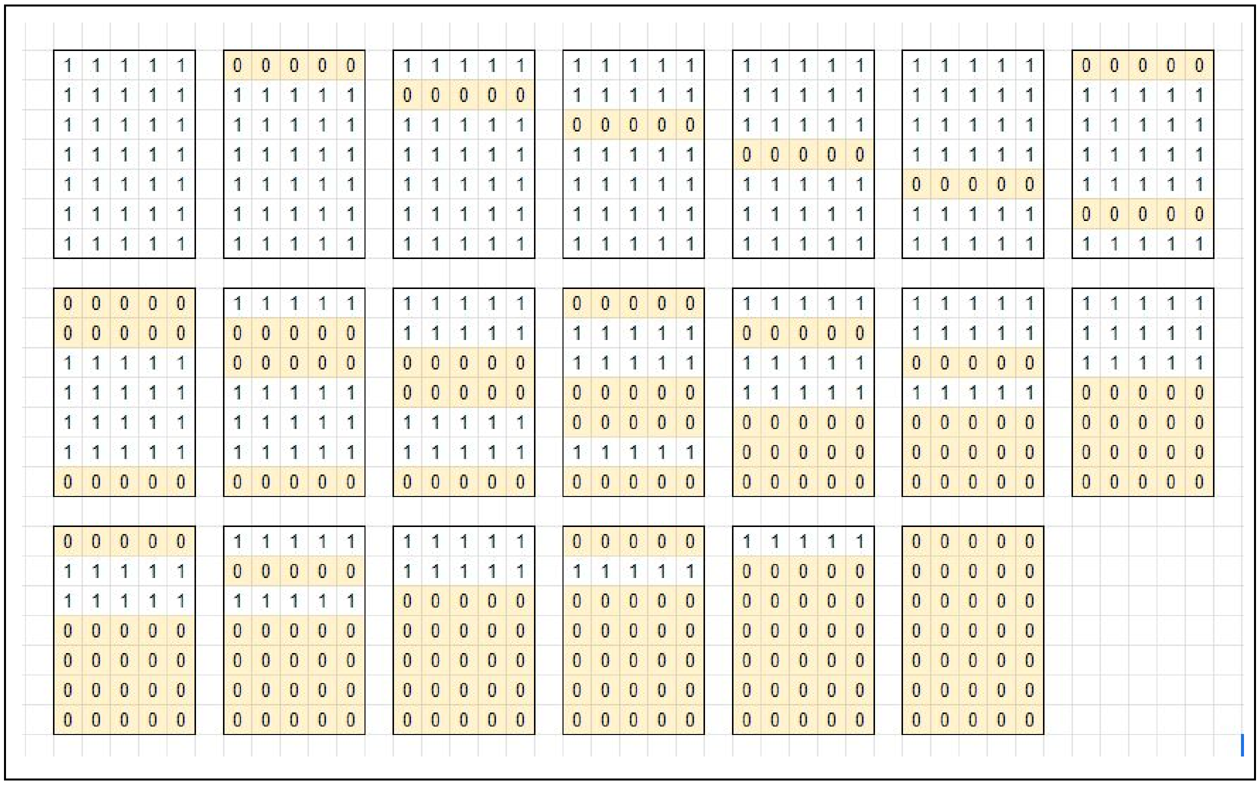
IX. MODE 1 ANIMATION GUIDE



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1			1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	1	0	~~0~~	~~1~~
2	R2		1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	~~0~~	~~1~~
3	R3		1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	~~0~~	~~1~~
4	R4		1	1	1	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	0	1	1	0	~~1~~	~~1~~
5	R5		1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	~~1~~	~~1~~
6	R6		1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	~~1~~	~~1~~
7	R7		1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	1	1	~~1~~	~~1~~
8																										
9		FFHFFHFFHFFHFFH0H77H77H77H0HFFH3EH0H3EHFFHFFHFFH3EH0H3EHFFH6HF8HFFH0HFDHFBHFDH0HFFHFFHFFH41H3EH36H45HFFH0HFDHFBHFDH0HFFH3HEDHEEHEDH3HFFH0HF9HE7H9FH0H																								

	A	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF
1		~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~
2	R2	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~
3	R3	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~
4	R4	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~
5	R5	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~
6	R6	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~
7	R7	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~
8																																	
9																																	

X. MODE 2 ANIMATION GUIDE



1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1

0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0

1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

XI. VIDEO DEMONSTRATION OF CIRCUIT

https://youtu.be/ndlWmtgqqqI