THE MICROPROCESSORS & MICROCONTROLLERS

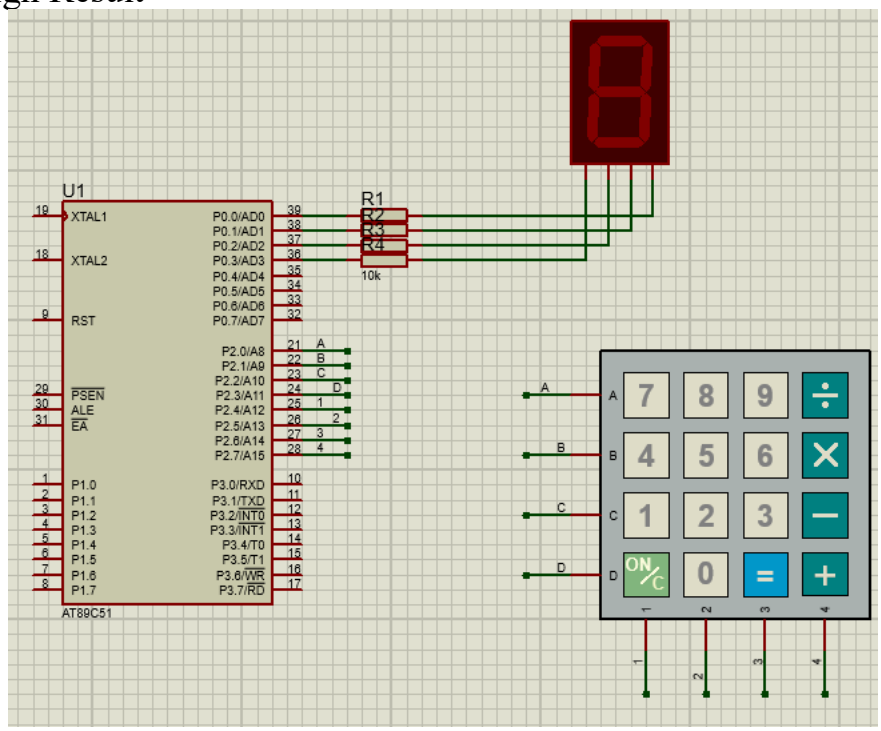
InsSUBctor : The Tung Than
Student: Bach Hoang Phan Thanh
ID: 21520599

PRATICE EXERCISE #4:
USING UART

I. Practice content

1. Design a 4x4 keyboard set including the following buttons:
    - o From 0 to 9
    - o The signs + - * /
    - o Sign =
    - o Reset button

1.1. Design Result



1.2Explain the operating principle of the effects

Link result:
https://drive.google.com/drive/folders/1PIaJGq8aQnYFuXUv5PnCj0xuxdZq_9IY?usp=sharing

Code	Explaination

ORG 0000H
MAIN:
MOV P2, #11110111B
JNB P2.4, OFF
JNB P2.5, NUM0
JNB P2.6, EQUAL
JNB P2.7, PLUS
MOV P2, #11111011B
JNB P2.4, NUM1
JNB P2.5, NUM2
JNB P2.6, NUM3
	
	Check the first row of 4x4 keyboard
	- Check ON/C button
	- Check “0” button
	- Check “=” button
	- Check “+” button
	Check the second row of 4x4 keyboard
	- Check “1” button
	- Check “2” button
	- Check “3” button

```
        JNB P2.7, SUB
        MOV P2, #11111101B
        JNB P2.4, NUM4
        JNB P2.5, NUM5
        JNB P2.6, NUM6
        JNB P2.7, MULTI
        MOV P2, #11111110B
        JNB P2.4, NUM7
        JNB P2.5, NUM8
        JNB P2.6, NUM9
        JNB P2.7, DIVIDE
        JMP MAIN
NUM0:
        MOV P0, #0000B
        JMP MAIN
NUM1:
        MOV P0, #0001B
        JMP MAIN
NUM2:
        MOV P0, #0010B
        JMP MAIN
NUM3:
        MOV P0, #0011B
        JMP MAIN
NUM4:
        MOV P0, #0100B
        JMP MAIN
NUM5:
        MOV P0, #0101B
        JMP MAIN
NUM6:
        MOV P0, #0110B
        JMP MAIN
NUM7:
        MOV P0, #0111B
        JMP MAIN
NUM8:
        MOV P0, #1000B
        JMP MAIN
NUM9:
```

- Check “-“ button

Check the third row of 4x4 keyboard

- Check “4” button
- Check “5” button
- Check “6” button
- Check “x” button

Check the forth row of 4x4 keyboard

- Check “7” button
- Check “8” button
- Check “9” button
- Check “/” button

Code above is used for display each buttons with each bit value

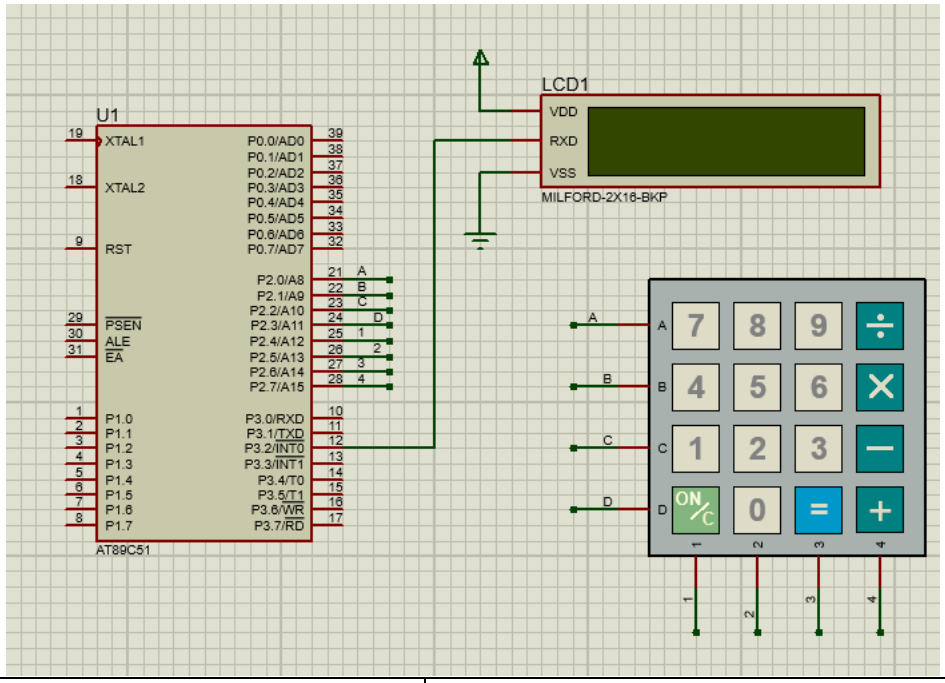
- 0000 display “0”
- 0001 display “1”
- 0010 display “2”
- 0011 display “3”
- 0100 display “4”
- 0101 display “5”
- 0110 display “6”
- 0111 display “7”
- 1000 display “8”
- 1001 display “9”
- 1010 display “+”
- 1011 display “-“
- 1100 display “x”
- 1101 display “/”
- 1110 display “ON/C”
- 1111 display “=”

```
        MOV P0, #1001B
        JMP MAIN
PLUS:
        MOV P0, #1010B
        JMP MAIN
SUB:
        MOV P0, #1011B
        JMP MAIN
MULTI:
        MOV P0, #1100B
        JMP MAIN
DIVIDE:
        MOV P0, #1101B
        JMP MAIN
OFF:
        MOV P0, #1110B
        JMP MAIN
EQUAL:
        MOV P0, #1111B
        JMP MAIN
END
```

2.

Using AT89C51/AT89C52 in combination with the module just designed above to design a handheld calculator, display calculations and results on an LCD that receives data via UART.

2.1 Design Result



Code	
ORG 00h	
MOV R0, # -1	Set initial value for R0, R1 (-1)
MOV R1, # -1	
MOV TMOD, #20H	Use Timer0 in Mode 1
MOV TH1, #0FDH	Baud rate is set to 9600, when Crystal Frequency is 11.0592MHz → FDH value
MOV SCON, #50H	Set value 50H for SCON → UART mode 1
SETB TR1	Turn on Timer0
JMP MAIN	
RESULT:	Result is used to have “two-digit” number
MOV B, #10	Ex: If result were 15, we will have A as 1 and B as 5.
DIV AB	
IF0:	
CJNE A, #0, IF1	CJNE A, #0 → to check if A equals to 0 or not. If A equals to 0, the string value ‘0’ move to SBUF register
MOV SBUF, # '0'	
JNB TI, $	
CLR TI	
JMP SHIFTB	The same with IF1.. IF9 and the unit number by SHIFTB (the value of B is the remainder of division between A and B)
IF1:	
CJNE A, #1, IF2	
MOV SBUF, # '1'	

```
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF2:
        CJNE A, #2, IF3
        MOV SBUF, # '2'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF3:
        CJNE A, #3, IF4
        MOV SBUF, # '3'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF4:
        CJNE A, #4, IF5
        MOV SBUF, # '4'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF5:
        CJNE A, #5, IF6
        MOV SBUF, # '5'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF6:
        CJNE A, #6, IF7
        MOV SBUF, # '6'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF7:
        CJNE A, #7, IF8
        MOV SBUF, # '7'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF8:
```

```
        CJNE A, #8, IF9
        MOV SBUF, # '8'
        JNB TI, $
        CLR TI
        JMP SHIFTB
IF9:
        CJNE A, #9, SHIFTB
        MOV SBUF, # '9'
        JNB TI, $
        CLR TI
        JMP SHIFTB
SHIFTB:
        MOV A, B
CHECK0:
        CJNE A, #0, CHECK1
        MOV SBUF, # '0'
        JNB TI, $
        CLR TI
        JMP DONE
CHECK1:
        CJNE A, #1, CHECK2
        MOV SBUF, # '1'
        JNB TI, $
        CLR TI
        JMP DONE
CHECK2:
        CJNE A, #2, CHECK3
        MOV SBUF, # '2'
        JNB TI, $
        CLR TI
        JMP DONE
CHECK3:
        CJNE A, #3, CHECK4
        MOV SBUF, # '3'
        JNB TI, $
        CLR TI
        JMP DONE
CHECK4:
        CJNE A, #4, CHECK5
        MOV SBUF, # '4'
```

JNB TI, $ CLR TI JMP DONE	
CHECK5: CJNE A, #5, CHECK6 MOV SBUF, # '5' JNB TI, $ CLR TI JMP DONE	
CHECK6: CJNE A, #6, CHECK7 MOV SBUF, # '6' JNB TI, $ CLR TI JMP DONE	
CHECK7: CJNE A, #7, CHECK8 MOV SBUF, # '7' JNB TI, $ CLR TI JMP DONE	
CHECK8: CJNE A, #8, CHECK9 MOV SBUF, # '8' JNB TI, $ CLR TI JMP DONE	
CHECK9: CJNE A, #9, DONE MOV SBUF, # '9' JNB TI, $ CLR TI JMP DONE	
DONE: RET	
EQUAL: MOV SBUF, # '=' JNB TI, $ CLR TI	Display the string value ‘=’
IFPLUS:	Check which function is requested

```
        CJNE A, #10000000B, IFSUB
        MOV A, #0
        ADD A, R0
        ADD A, R1
        JMP KQUA
IFSUB:
        CJNE A, #01000000B,
IFMULTIPLE
        MOV A, #0
        ADD A, R0
        SUBB A, R1
        JMP KQUA
IFMULTIPLE:
        CJNE A, #00100000B,
IFDIVIDE
        MOV A, R0
        MOV B, R1
        MUL AB
        JMP KQUA
IFDIVIDE:
        MOV A, R0
        MOV B, R1
        DIV AB
        JMP KQUA
KQUA:
        CALL RESULT
        JNB P2.6, $
        JMP MAIN
MAIN:
        MOV P2, #11110111B
        JNB P2.4, ONOFF
        JNB P2.5, NUM0
        JNB P2.6, EQUAL
        JNB P2.7, PLUS
        MOV P2, #11111011B
        JNB P2.4, NUM1
        JNB P2.5, NUM2
        JNB P2.6, NUM3
        JNB P2.7, SUB
        MOV P2, #11111101B
```

- 10000000 for add
- 01000000 for subtract
- 00100000 for multiplier
- 00010000 for division

JNB P2.4, NUM4
JNB P2.5, NUM5
JNB P2.6, NUM6
JNB P2.7, MULTIPLE
MOV P2, #11111110B
JNB P2.4, NUM7
JNB P2.5, NUM8
JNB P2.6, NUM9
JNB P2.7, DIVIDE
JMP MAIN
NUM0:
JMP Inp0
NUM1:
JMP Inp1
NUM2:
JMP Inp2
NUM3:
JMP Inp3
NUM4:
JMP Inp4
NUM5:
JMP Inp5
NUM6:
JMP Inp6
NUM7:
JMP Inp7
NUM8:
JMP Inp8
NUM9:
JMP Inp9
PLUS:
JMP PLUSFUNC
SUB:
JMP SUBFUNC
MULTIPLE:
JMP MULFUNC
DIVIDE:
JMP DIVIDEFUNC
ONOFF:
MOV SBUF, #254
	When decode, I face out an error that out of range for each "JNB", so I use NUM0..DIVIDE to shorten the range

JNB TI, $ CLR TI MOV SBUF, #1 JNB TI, $ CLR TI MOV R0, # -1 MOV R1, # -1 JNB P2.4, $ JMP MAIN	
PLUSFUNC: MOV SBUF, # '+' JNB TI, $ CLR TI JNB P2.7, $ MOV A, #10000000B JMP MAIN	Move String value '+' to SBUF register, display on LCD Set value of Plus function to A register
SUBFUNC: MOV SBUF, # '-' JNB TI, $ CLR TI JNB P2.7, $ MOV A, #01000000B JMP MAIN	 Move String value '-' to SBUF register, display on CLD Set value of Subtract function to A register
MULFUNC: MOV SBUF, # '*' JNB TI, $ CLR TI JNB P2.7, $ MOV A, #00100000B JMP MAIN	 Move String value '*' to SBUF register, display on LCD Set value of Multiple function to A register
DIVIDEFUNC: MOV SBUF, # '/' JNB TI, $ CLR TI JNB P2.7, $ MOV A, #00010000B JMP MAIN	 Move String value '/' to SBUF register, display on LCD Set value of Divide function to A register
Inp0: MOV SBUF, # '0' JNB TI, $	Inp0 use to check if R0 value is initial or not. If R0 is initial, the pressed button give data for first value – 0.

```
        CLR TI
        JNB P2.5, $
        CJNE R0, # -1, Out0
        MOV R0, #0
Out0:
        MOV R1, #0
        JMP MAIN
Inp1:
        MOV SBUF, # '1'
        JNB TI, $
        CLR TI
        JNB P2.4, $
        CJNE R0, # -1, Out1
        MOV R0, #1
Out1:
        MOV R1, #1
        JMP MAIN
Inp2:
        MOV SBUF, # '2'
        JNB TI, $
        CLR TI
        JNB P2.5, $
        CJNE R0, # -1, Out2
        MOV R0, #2
Out2:
        MOV R1, #2
        JMP MAIN
Inp3:
        MOV SBUF, # '3'
        JNB TI, $
        CLR TI
        JNB P2.6, $
        CJNE R0, # -1, Out3
        MOV R0, #3
Out3:
        MOV R1, #3
        JMP MAIN
Inp4:
        MOV SBUF, # '4'
        JNB TI, $
```

```
        CLR TI
        JNB P2.4, $
        CJNE R0, # -1, Out4
        MOV R0, #4
Out4:
        MOV R1, #4
        JMP MAIN
Inp5:
        MOV SBUF, # '5'
        JNB TI, $
        CLR TI
        JNB P2.5, $
        CJNE R0, # -1, Out5
        MOV R0, #5
Out5:
        MOV R1, #5
        JMP MAIN
Inp6:
        MOV SBUF, # '6'
        JNB TI, $
        CLR TI
        JNB P2.6, $
        CJNE R0, # -1, Out6
        MOV R0, #6
Out6:
        MOV R1, #6
        JMP MAIN
Inp7:
        MOV SBUF, # '7'
        JNB TI, $
        CLR TI
        JNB P2.4, $
        CJNE R0, # -1, Out7
        MOV R0, #7
Out7:
        MOV R1, #7
        JMP MAIN
Inp8:
        MOV SBUF, # '8'
        JNB TI, $
```

```
        CLR TI
        JNB P2.5, $
        CJNE R0, # -1, Out8
        MOV R0, #8
Out8:
        MOV R1, #8
        JMP MAIN
Inp9:
        MOV SBUF, # '9'
        JNB TI, $
        CLR TI
        JNB P2.6, $
        CJNE R0, # -1, Out9
        MOV R0, #9
Out9:
        MOV R1, #9
        JMP MAIN
END
```