LAB 2 - Reference Material

[label:] opcode [operand] [;comment]

Data movement:

MOV dst, src

XCH A, <R[0-7] / addr / @R0 / @R1> ; exchange values

Mathematical Operations:

INC <A / R[0-7] / addr / @R0 / @R1 / DPTR> ; increments the value by 1

DEC <A / R[0-7] / addr / @R0 / @R1 / DPTR> ; decrements the value by 1

ADD A, <A / R[0-7] / addr / @R0 / @R1 / #imm> ; A = A + second operand

SUBB A, <A / R[0-7] / addr / @R0 / @R1 / #imm> ; A = A - second operand - CY (in PSW)

; to perform normal subtraction, CY to be set to 0 before using SUBB

MUL AB ; A * B . result low byte in A and high byte in B

DIV AB ; A / B . quotient in A and remainder in B

Bit Manipulation:

SETB P1.0 ; sets a bit to 1 - turns of LED

CLR P1.0 ; sets a bit to 0 - turns on LED

CPL P1.0 ; complements a bit - here inverts Port 1, Pin 0 to turn the LED on/off

RL A ; circular left rotate - if A = `10101100` , it becomes `01011001`

RR A ; circular right rotate - if A = `10101100` , it becomes `01010110`

ANL <A/addr/C>, <A / R[0-7] / addr / @R0 / @R1 / bit / #imm> ; logical/Bitwise AND

ORL <A/addr/C>, <A / R[0-7] / addr / @R0 / @R1 / bit / #imm> ; logical/Bitwise OR

XRL <A/addr/C>, <A / R[0-7] / addr / @R0 / @R1 / bit / #imm> ; logical/Bitwise EXOR

Unconditional Jump

JMP <label>

```
Blinking light
      LOOP: CPL P1.0 ; Invert the state of the LED pin
            JMP LOOP ; Jump right back to invert it again
                                                instantly
```

Conditional Jump

DJNZ (Decrement and Jump if Not Zero)

```
            MOV R5, #50
AGAIN:      INC A
            DJNZ R5, AGAIN    ; Decrements R5 by 1. If R5 isn't 0,
                                      jump to LOOP_BACK.
```

The following jump statements have one operand: **label**

JC (Jump if Carry is Set)

JNC (Jump if Carry is Not Set)

JZ (Jump if Accumulator is Zero)

JNZ (Jump if Accumulator is Not Zero)

```
      JC loop
```

The following jump statements have two operands: **bit, label**

JB (Jump if Bit is Set)

JNB (Jump if Bit is Not Set)

JBC (Jump if Bit is Set and Clear Bit)

```
      JB P3.2, loop
```

CJNE (Compare and Jump if Not Equal)

```
CJNE A, #0Dh, loop
```