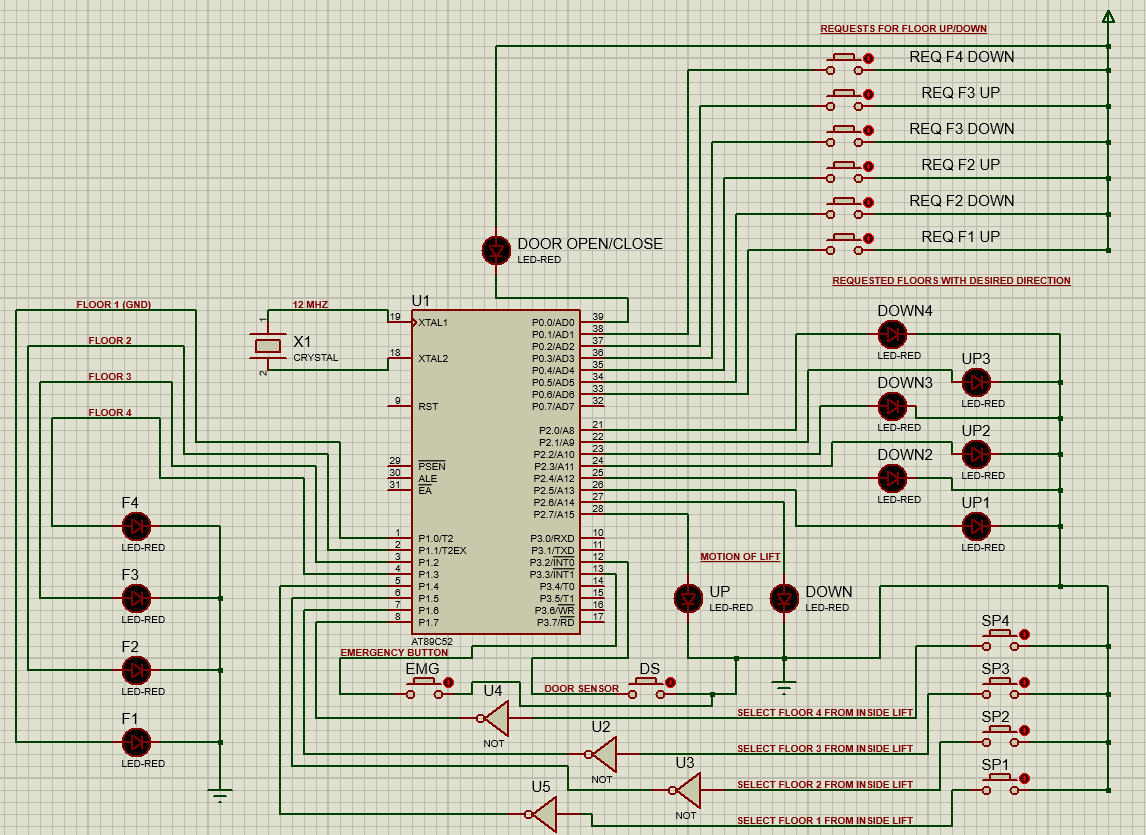
MI – ELEVATOR SYSTEM

* CIRCUIT DIAGRAM:



* DESCRIPTION:
  + Two external interrupts INT0 and INT1 will be used. Input at INT0 will be given by a DOOR SENSOR which is a button. It will be used for when the lift is in motion to let the lift know that it has reached the next floor. Input at INT1 will be given by a button that will act as an emergency switch to stop the lift at the nearest floor.
  + Using a register R2, the value of the register will determine which location LED will be ON and the rest will be OFF. E.g., if the value of R2 = #02H, it would mean that the LED is on the 2nd floor and the LED of FLOOR 2 will be ON. The register R2 will be updated as the lift moves from floor to floor and the LEDs will turn OFF and ON respectively.
  + Using registers R4, R5, R6 and R7, we will store the rest for the desired floor which is selected from inside the lift. Another register R1 will be used to store the current direction of the lift for example R1 = #01H means lift is moving UP. Certain conditions will be applied and if the requested stop is falling in the direction of the lift saved in R1, it will stop or else it will stop after the direction has changed. The emergency button for immediate stop will be done by using an external interrupt at INT1 using a button as explained before.
  + The register R1 will be used for storing the current lift direction. Requests from outside lift will be entertained according to the value of register R1. The 6 floor requests from outside lift namely, Floor 1 UP, Floor 2 Down, Floor 2 Up and so on will stored in the accumulator A by setting bits ACC.0, ACC.1, ACC.2 and so on as:

ACC.0 = 1 🡪 Request from floor 1 - UP

ACC.1 = 1 🡪 Request from floor 2 - UP

ACC.2 = 1 🡪 Request from floor 2 - DOWN

ACC.3 = 1 🡪 Request from floor 3 - UP

ACC.4 = 1 🡪 Request from floor 3 - DOWN

ACC.5 = 1 🡪 Request from floor 4 – DOWN

The ACC bits if set will be compared with the floor direction stored in register R1, which is updated after every stop, and will decide if the lift should stop or not.

* Proteus Simulation: 🡪 Files are attached.
  + Code:

ORG 0

LJMP 0030H

ORG 0003H

SETB P1.3

FLOOR1UP:

JNB ACC.0,FLOOR2UP

GOTO1:

CJNE R1,#02H,FLOOR2UP

DEC R2

CALL MOVELIFT

STOP1:

JNB P1.0,FLOOR2UP

CLR P2.5

CLR ACC.0

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

//----

FLOOR2UP:

JNB ACC.1,FLOOR3UP

CJNE R2,#04H,CCC

DEC R2

CALL MOVELIFT

CCC:

CJNE R2,#03H,GO2\_1

DEC R2

CALL MOVELIFT

SJMP STOP2UP

GO2\_1:

CJNE R2,#02H,GOTO2UP

SJMP FLOOR3UP

GOTO2UP:

CJNE R1,#01H,FLOOR3UP

INC R2

CALL MOVELIFT

STOP2UP:

JNB P1.1,FLOOR3UP

CLR P2.3

CLR ACC.1

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

//-------

FLOOR3UP:

JNB ACC.3,FLOOR2DOWN

CJNE R2,#04H,GO3

DEC R2

CALL MOVELIFT

SJMP STOP3UP

GO3:

CJNE R2,#03H,GOTO3UP

SJMP FLOOR2DOWN

GOTO3UP:

CJNE R1,#01H,FLOOR2DOWN

INC R2

CALL MOVELIFT

STOP3UP:

JNB P1.2,FLOOR2DOWN

CLR P2.1

CLR ACC.3

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

//----------

FLOOR2DOWN:

JNB ACC.2,FLOOR3DOWN

CJNE R2,#01H,GO2\_3

INC R2

CALL MOVELIFT

SJMP STOP2DOWN

GO2\_3:

CJNE R2,#02H,GO2DOWN

SJMP FLOOR3DOWN

GO2DOWN:

CJNE R1,#02H,FLOOR3DOWN

DEC R2

CALL MOVELIFT

STOP2DOWN:

JNB P1.2,FLOOR3DOWN

CLR P2.4

CLR ACC.2

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

//----------

FLOOR3DOWN:

JNB ACC.4,FLOOR4DOWN

CJNE R2,#04H,GO2\_4

DEC R2

CALL MOVELIFT

SJMP STOP3DOWN

GO2\_4:

CJNE R2,#02H,GO3DOWN

SJMP FLOOR4DOWN

GO3DOWN:

CJNE R1,#02H,FLOOR4DOWN

DEC R2

CALL MOVELIFT

STOP3DOWN:

JNB P1.2,FLOOR4DOWN

CLR P2.2

CLR ACC.4

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

//------------

FLOOR4DOWN:

JNB ACC.5,RETURN

GOTO4DOWN:

CJNE R1,#01H,RETURN

INC R2

CALL MOVELIFT

STOP4DOWN:

JNB P1.3,RETURN

CLR P2.0

CLR ACC.5

CALL DELAY ; 5 secs delay x 4 = 20 secs delay

CALL DELAY

CALL DELAY

CALL DELAY

RETURN:

RETI

//-------------------------------------------------------------------------

DELAY:

CLR P0.0

MOV R7, #200

AGAIN1:

MOV TH0, #0D8H

MOV TL0, #0F0H

SETB TR0

WAIT1:

JNB TF0, WAIT1

CLR TF0

CLR TR0

DJNZ R7, AGAIN1

MOV R7, #200

AGAIN2:

MOV TH0, #0D8H

MOV TL0, #0F0H

SETB TR0

WAIT2:

JNB TF0, WAIT2

CLR TF0

CLR TR0

DJNZ R7, AGAIN2

MOV R7, #100

AGAIN3:

MOV TH0, #0D8H

MOV TL0, #0F0H

SETB TR0

WAIT3:

JNB TF0, WAIT3; jump if bit not set

CLR TF0

CLR TR0

DJNZ R7, AGAIN3

SETB P0.0

RET

//--------------------------------------------------------------------------------------

MOVELIFT:

FLOOR1:

CJNE R2,#01H,FLOOR2

CPL P1.0

CPL P1.1

CLR P1.2

CLR P1.3

FLOOR2:

CJNE R2,#02H,FLOOR3

CPL P1.0

CPL P1.1

CLR P1.2

CLR P1.3

FLOOR3:

CJNE R2,#03H,FLOOR4

CLR P1.0

CPL P1.1

CPL P1.2

CLR P1.3

FLOOR4:

CJNE R2,#04H,GOBACK1

CLR P1.0

CLR P1.1

CPL P1.2

CPL P1.3

GOBACK1:

RET

//---------------------------------------------------------------------------------------------

UPDATE\_DIRECTION:

GOINGUP:

CJNE R1,#01H,GOINGDOWN

SETB P2.7

CLR P2.6

SJMP CON\_RET

GOINGDOWN:CJNE R1,#02H,GOINGNOWHERE

SETB P2.6

CLR P2.7

SJMP CON\_RET

GOINGNOWHERE:CJNE R1,#00H,CON\_RET

CLR P2.6

CLR P2.7

CON\_RET:

RET

//----------------------------------------------------------------------------------------------

CHECK\_DIRECTION:

//--

CHECKNEXT1:

JNB ACC.0,CHECKNEXT2

CJNE R2,#01H,SET1

MOV R1,#00H

CALL UPDATE\_DIRECTION

SJMP CHECKNEXT2

SET1:

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

//--

CHECKNEXT2:

JNB ACC.1,CHECKNEXT3

CJNE R2,#02H,SET2

MOV R1,#00H

CALL UPDATE\_DIRECTION

SJMP CHECKNEXT3

SET2:

CJNE R2,#01H,SET3

MOV R1,#01H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

SET3:

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

//--

CHECKNEXT3:

JNB ACC.2,CHECKNEXT4

CJNE R2,#02H,SET4

MOV R1,#00H

CALL UPDATE\_DIRECTION

SJMP CHECKNEXT4

SET4:

CJNE R2,#01H,SET5

MOV R1,#01H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

SET5:

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

//--

CHECKNEXT4:

JNB ACC.3,CHECKNEXT5

CJNE R2,#03H,SET6

MOV R1,#00H

CALL UPDATE\_DIRECTION

SJMP CHECKNEXT5

SET6:

CJNE R2,#04H,SET7

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

SET7:

MOV R1,#01H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

//-----

CHECKNEXT5:

JNB ACC.4,CHECKNEXT6

CJNE R2,#03H,SET8

MOV R1,#00H

CALL UPDATE\_DIRECTION

SJMP CHECKNEXT6

SET8:

CJNE R2,#04H,SET9

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

SET9:

MOV R1,#01H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

//-------

CHECKNEXT6:

JNB ACC.5,NOW\_BACK

CJNE R2,#04H,SET10

MOV R1,#02H

CALL UPDATE\_DIRECTION

SJMP NOW\_BACK

SET10:

MOV R1,#01H

CALL UPDATE\_DIRECTION

NOW\_BACK:

RET

//--

ORG 0030H

SETB P0.0

MOV P1,#0F0H

MOV P2,#0H

MOV R1,#00H ;CURRENT DIRECTION OF THE LIFT

MOV R2,#01H ;CURRENT POSTION OF THE LIFT

MOV A,#00H

SETB P1.0

CLR IT0

CLR IT1

MAIN:

MOV IE,#81H

CLR IT0

CLR IT1

UP1: ; BUTTONS OUTSIDE THE ELEVATOR

JNB P0.6, UP2

SETB P2.5

SETB ACC.0

UP2:

JNB P0.4, DOWN2

SETB P2.3

SETB ACC.1

DOWN2:

JNB P0.5, UP3

SETB P2.4

SETB ACC.2

UP3:

JNB P0.2,DOWN3

SETB P2.1

SETB ACC.3

DOWN3:

JNB P0.3,DOWN4

SETB P2.2

SETB ACC.4

DOWN4:

JNB P0.1,REQ1

SETB P2.0

SETB ACC.5

REQ1: ; BUTTONS INSIDE THE ELEVATOR

JNB P1.4,REQ2

MOV R4,#01H

REQ2:

JNB P1.5,REQ3

MOV R5,#01H

REQ3:

JNB P1.6,REQ4

MOV R6,#01H

REQ4:

JNB P1.7,CONN

MOV R7,#01H

CONN:

CALL CHECK\_DIRECTION

SJMP MAIN

END