DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING
SRM UNIVERSITY, Kattankulathur – 603203.

Title of Experiment	: 7. INTERFACING OF DAC 8051
Name of the candidate	: GAUTAM NAG
Register Number	: RA1811005010278
Date of Experiment	: 19-03-21
Date of submission	**:19-03-21**

S.NO :	**MARKS SPLIT UP**	**MAXIMUM MARKS (50)**	**MARKS OBTAINED**
1	PRE LAB	5	
2	PROGRAM	25	
3	EXECUTION	15	
4	POST LAB	5	
TOTAL		50	

Staff Signature

PRE LAB QUESTION AND ANSWERS

1. **Explain the various steps involved when executing CALL instruction.**
   Step 1: Fetch instruction. Execution cycle starts with fetching instructions from the main memory. ...
   Step 2: Decode instruction. ...
   Step 3: Perform ALU operation. ...
   Step 4: Access memory. ...
   Step 5: Update Register File. ...
   Step 6: Update the PC (Program Counter)

2. **What is the use of PUSH and POP instruction?**
   "push" and "pop" instructions. "push" stores a constant or 64-bit register out onto the stack. The 64-bit registers are the ones like "rax" or "r8", not the 32-bit registers like "eax" or "r8d". "pop" retrieves the last value pushed from the stack.

3. **What is a subroutine program?**
   a subroutine is a sequence of program instructions that performs a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task should be performed.

7. INTERFACING OF DAC 8051

Aim:

The purpose of this experiment is to learn about the interfacing of Digital to Analog converter with 8051 microcontroller using edsim51.

Hardware Requirement:

The 8051 Microcontroller kit, Power Supply.

Software Requirement:

8051 simulator (Edsim51)

Algorithm:

1. Move the Port Address of DAC 2 FFC8 to the DPTR.
2. Move the Value of Register A to DPTR and then Call the delay.
3. Move the Value of Register A (FFh) to DPTR and the call the dalay.
4. Repeat the steps 2 and 3.

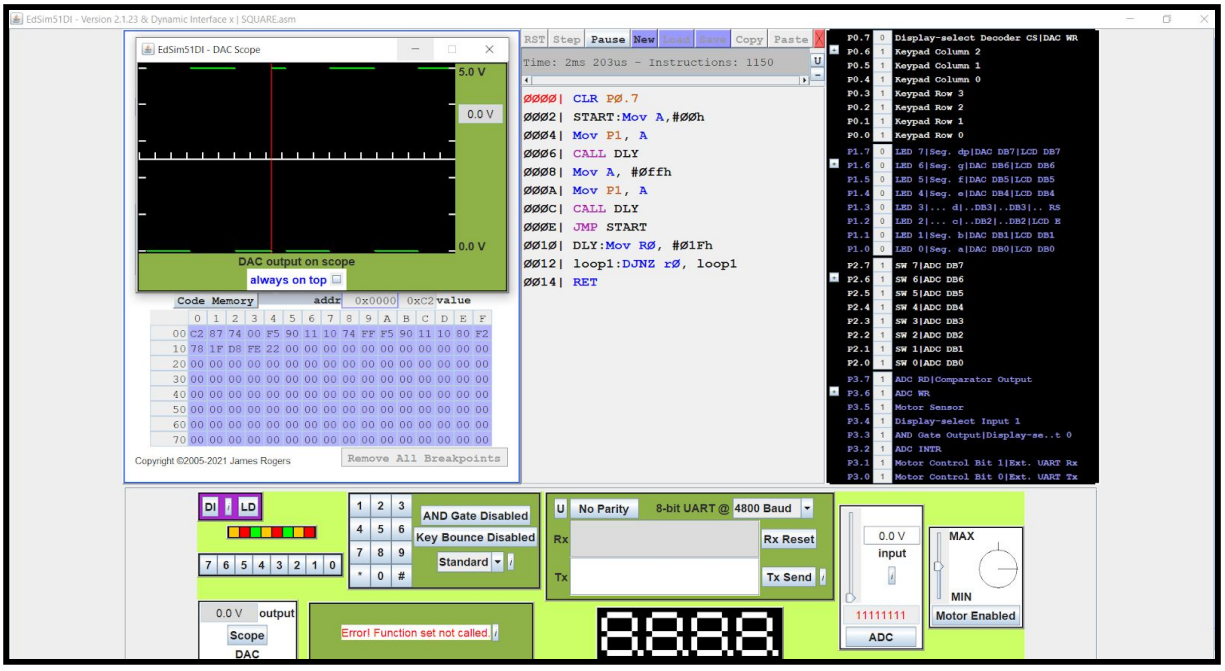
PROGRAM : SQUARE WAVE GENERATION:

Memory Location	Label	Opcode	Mnemonics	Comments
4100		90 FF C8	MOV DPTR, #FFC8H	
4103	START:	74 00	MOV A, #00H	
4105		F0	MOVX @DPTR, A	
4106		12 41 12	LCALL DELAY	
4109		74 FF	MOV A, #FFH	
410B		F0	MOVX @DPTR, A	
410C		12 41 12	LCALL DELAY	
410F		02 41 03	LJMP STTART	
4112	DELAY:	79 05	MOV R1, #05H	
4114	LOOP:	7A FF	MOV R2, #FFH	
4116	HERE:	DA FE	DJNZ R2, HERE	
4118		D9 FA	DJNZ R1, LOOP	
411A		22	RET	
411B		80 E6	SJMP START	

EDSIM 51 PROGRAM: SQUARE WAVE GENERATION

ADDRESS	MNEMONICS	OPCODE	COMMENTS
0000	CLR P0.7		**clears set to 0 and sets to 0.7**
0002	MOV A, #00H		**move value of #00h to register a**
0004	MOV P1, A		**move value of a to P**
0006	CALL DELAY		**set the delay time with freq to 50 hz**
0008	MOV A, #0FFH		**store value of 00FH to P1**
000A	MOV P1,A		**store value of a to P1**
000C	CALL DELAY		**set delay**
000E	JMP START		**Execute the program**
00010	MOV R0, #01FH		**store value of #01FH into R0**
00012	DJNZ R0, LOOP1		**decrement the bit of the first operand in loop1**
00014	RET		**pops the lower and higher order bytes**

SIMULATION:



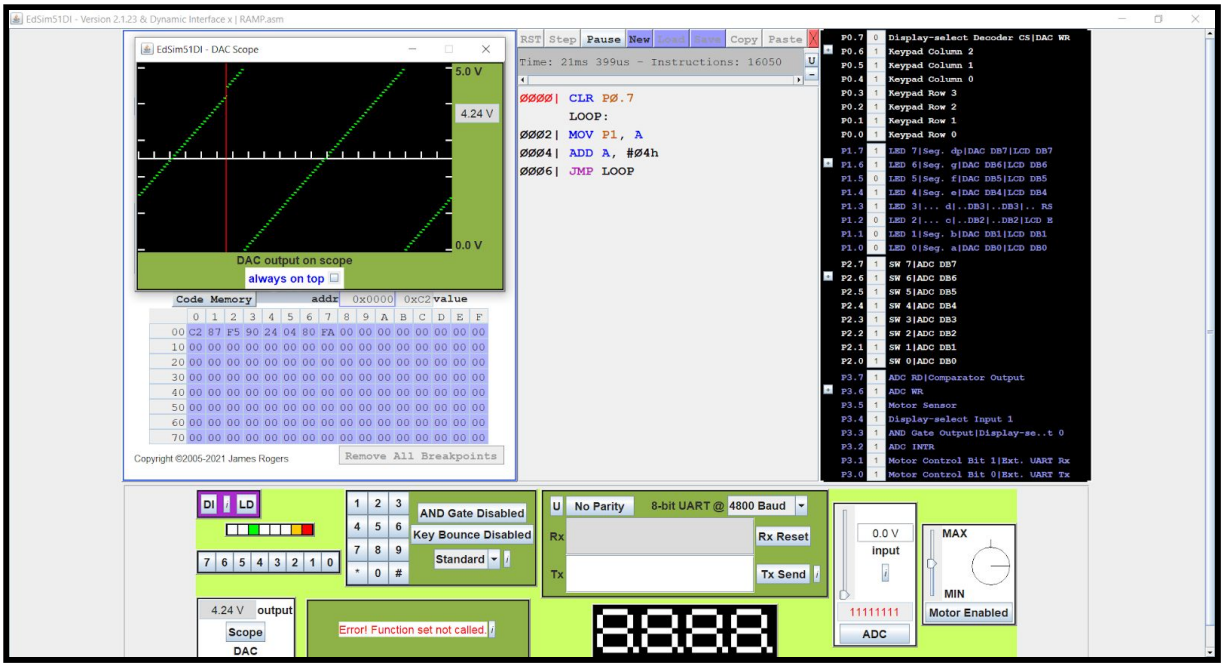
PROGRAM :

Memory Location	Label	Opcode	Mnemonics	Comments
4100		90 FF C8	MOV DPTR, #FFC8H	
4103		74 00	MOV A, #00H	
4105	LOOP:	F0	MOVX @DPTR, A	
4106		04	INC A	
4107		80 FC	SJMP LOOP	

EDSIM51 PROGRAM: RAMP WAVE GENERATION

ADDRESS	MNEMONICS	OPCODE	COMMENTS
0000	CLR P0.7		**clears set to 0 and sets to 0.7**
0002	MOV P1,A		**move value of #00h to register a**
0004	ADD A,#04h		**ADD THE VALUES OF REGISTERS**
0006	JMP LOOP		**Execute the loop program**

SIMULATION:



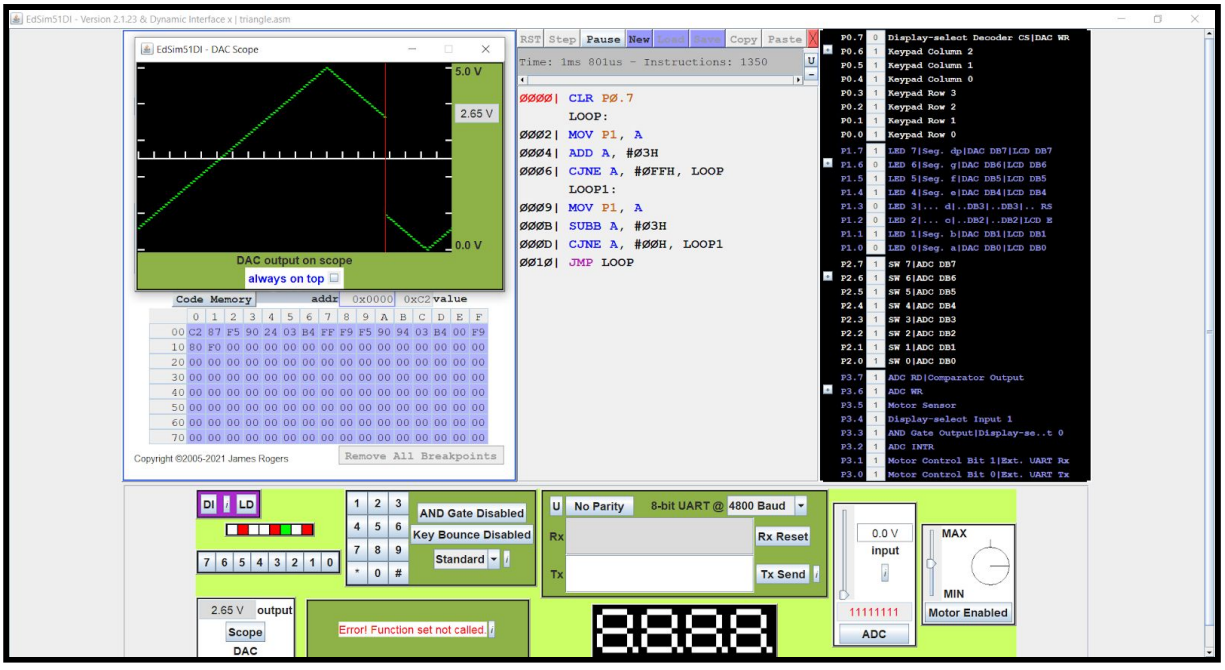
PROGRAM: TRIANGULAR WAVE GENERATION

Memory Location	Label	Opcode	Mnemonics	Comments
4100		90 FF C8	MOV DPTR, #FFC8H	
4103	START:	74 00	MOV A, #00H	
4105	LOOP1:	F0	MOVX @DPTR, A	
4106		04	INC A	
4107		70 FC	JNZ LOOP1	
4109		74 FF	MOV A, #0FFH	
410B	LOOP2:	F0	MOVX @DPTR, A	
410C		14	DEC A	
410D		70 FC	JNZ LOOP2	
410F		02 41 03	LJMP START	

EDSIM51 PROGRAM: TRIANGULAR WAVEFORM

0000	CLR P0.7		**clears set to 0 and sets to 0.7**
0002	MOV P1, A		**move value of #00h to register a**
0004	ADD A, #03H		**ADD THE VALUES OF REGISTERS**
0006	CJNE A,#0FFH, LOOP		**compares the first two operands and branches them to A (loop)**
0009	MOV P1, A		**store value of A to P1**
000B	SUBB A, #03H		**subtraction variable**
000D	CJNE A, #00H, LOOP1		**compares the first two operands and branches them to A (loop)**
00010	JMP LOOP		**Executes and loops**

SIMULATION:



Result:
Thus, an assembly language program for Digital to Analog has been executed.

POST LAB QUESTION AND ANSWERS

1. **How are the instructions classified according to word size?**
   The 8085 instruction set is classified into 3 categories by considering the length of the instructions. ... Three types of instruction are: 1-byte instruction, 2-byte instruction, and 3-byte instruction.

2. **What is mode 0 operation of 8255.**

   Mode 0 – Simple or basic I/O Mode

   In this mode all of the ports A, B and C can be used as input or output mode. The outputs are latched, but inputs are not latched. This mode has interrupt handling capability.

3. **What are the modes of operation supported by 8255?**

   Bit Set Reset (BSR) Mode.
   Input/ Output Mode.
   Mode 0 – Simple or basic I/O Mode.
   Mode 1 – Handshake or Strobed I/O.
   Mode 3 – Bidirectional I/O.