SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

Department of Electronics and Communication Engineering
(Accredited by NBA)

22EC505 MICROCONTROLLERS LABORATORY

PROGRAM MANUAL

Faculty Incharge **HoD**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

Department of Electronics and Communication Engineering
(Accredited by NBA)

22EC505 -MICROCONTROLLERS LABORATORY

2024-2025

NAME :

REGISTER NUMBER :

CLASS & SECTION :

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

Department of Electronics and Communication Engineering (Accredited by NBA)

22EC505 MICROCONTROLLERS LABORATORY

PRACTICAL RECORD

Name : …… **Reg. no: …….**

Class : …… **Branch: ……..**

BONAFIDE CERTIFICATE

Certified bonafide record of work done by Mr /Ms …….....
during the academic year 2024-25.

Staff-In Charge

Submitted for the Autonomous Practical Examination held on …….....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Department of Electronics and Communication Engineering
(Accredited by NBA)

VISION OF THE INSTITUTION

- To Produce Globally Competitive Engineers with High Ethical Values and Social Responsibilities.

MISSION OF THE INSTITUTION

- To impart highest quality state-of-the-art technical education by providing impetus to innovation, Research and Development and empowering students with Entrepreneurship skills.
- To instill ethical values, imbibe a sense of social responsibility and strive for societal wellbeing.
- To identify needs of society and offer sustainable solutions through outreach programs.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE DEPARTMENT

- To equip future engineers with high academic knowledge, ethical values, leadership skills and a passion to contribute to the society.

MISSION OF THE DEPARTMENT

- To provide quality and contemporary education in Electronics and Communication Engineering through continuous upgradation of Curriculum and laboratory facilities, industrial collaboration and effective teaching learning process.
- To facilitate research activities and entrepreneurship skills to cope up with the changes in industrial demand and meet the global and societal needs.
- To inculcate professional attitude and ethical values.

Department of Electronics and Communication Engineering
(Accredited by NBA)

PROGRAMME OUTCOMES (POs)

At the time of their graduation students of Electronics and Communication Engineering Programme should be in possession of the following Programme Outcomes

PO**1.**Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO**2. Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO**3. Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO**4. Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO**5. Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO**6. The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Department of Electronics and Communication Engineering
(Accredited by NBA)

PO**7. Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO**8. Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO**9**. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO**10**. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO**11. Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Department of Electronics and Communication Engineering
(Accredited by NBA)

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

The following Programme Educational Objectives are designed based on the department mission

PEO1: Exhibit technical competence in Electronics and Communication Engineering by providing innovative engineering solutions and excel in professional career.

PEO2: Indulge in problem identification, analysis and formulation to provide technically superior, economically feasible, environmentally compatible and socially acceptable design solutions.

PEO3: Contribute towards entrepreneurship and research, and exercise leadership through effective communication, teamwork and knowledge upgradation through lifelong learning.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

On successful completion of Bachelor of Engineering in Electronics and Communication Engineering Programme from Sri Krishna College of Engineering and Technology, the graduate will demonstrate:

PSO1: Potential to analyse, design, synthesize and provide technical solutions in the field of VLSI, Embedded Systems and Communication Networks.

PSO2: Emerge as ethical leaders, excel in research, engage in lifelong learning, pursue entrepreneurship and contribute towards the field of Electronics and Communication Engineering.

Department of Electronics and Communication Engineering
(Accredited by NBA)

22EC505	MICROCONTROLLERS LABORATORY	0/0/2/1
Nature of Course: M (Practical application)		
Course Objectives:		
1.	To write and execute the programming for various application using Microcontrollers.	
Course Outcomes		
C505.1	Understand the addressing modes of 8051 to perform basic arithmetic, Data Transfer and Conditional operations	[U]
C505.2	Apply the interfacing procedures of 8051 with hardware and peripheral devices	[AP]
C505.3	Evaluate the command word format for interfacing with peripherals	[E]
C505.4	Apply the Embedded C codes in microcontroller to perform Interfacing with MSP430	[AP]
C505.5	Understand the Interfacing procedure of ARM	[AP]
C505.6	Analyze the use of Keil Micro vision and perform ARM interfacing	[AN]

Course Content:

S.No	List of Experiments	CO Mapping	RBT
	Programming using 8051 microcontroller		
1	Arithmetic Instructions - Addition, subtraction, multiplication and division.	C505.1	[U]
2	Data Transfer - Block move, Exchange, Sorting, Finding largest element	C505.1	[U]
3	Programs using Conditional CALL & RETURN.	C505.1	[U]
	INTERFACING with 8051 microcontroller		
4	Stepper motor control interface to 8051 microcontroller.	C505.2	[AP]
5	Programmable peripheral interface to 8051 microcontroller.	C505.3	[E]
6	Keyboard and seven segment display interface with 8051 microcontroller.	C505.3	[E]
7	ADC and DAC using 8051 microcontroller.	C505.3	[E]
	Programming using MSP430		
8	LED blinking	C505.3	[AN]
9	Seven segment display interfacing	C505.5	[AP]
	Programming using ARM		
10	LED blinking	C505.6	[AN]
11	Seven segment display interfacing	C505.6	[AN]
12	LCD Interfacing	C505.6	[AN]

Total Hours:	**45**

Reference Books:	
1	Muhammad Ali Mazidi, "The 8051 Microcontroller and Embedded systems", Prentice Hall India, New Delhi, 2013
2	Kenneth J Ayala, "The 8051 Microcontroller – Architecture, Programming and Applications", Penram International Publications, Mumbai, India, 2005
3	MSP430 Microcontrollers Basics, John H Devis, 1stEdition, Newnes Publisher.2008
4	ARM on-chip architecture, Pearson Edition,second edition 2009.
Web References:	
1	https://exploreembedded.com
2	https://www.elprocus.com/peripherals-interfacing-to-the-microcontroller-8051-in-electronics/
Online Resources:	
1	*processors.wiki.ti.com/index.php/MSP430_LaunchPad_Tutorials*
2	*https://www.electronicshub.org › ARM*

Continuous Assessment				End Semester Examination	Total
Formative Assessment	**Summative Assessment**	**Total**	**Total Continuous Assessment**		
75	**25**	**100**	**60**	**40**	**100**

Assessment based on Continuous and End Semester Examination

Bloom's Level	Continuous Assessment (60%) [100 Marks]		End Semester Practical Examination (40%) [100 Marks]
	FA (75 Marks)	**SA (25 Marks)**	
Remember	-	-	-
Understand	-	-	-
Apply	60	60	40
Analyse	30	30	30
Evaluate	10	10	30
Create	-	-	-

Course Articulation Matrix														
CO	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**
1	2	1	1	1	1	-	-	-	3	-	-	-	1	-
2	2	1	2	2	2	-	-	-	3	-	-	-	2	-
3	2	1	1	1	1	-	-	-	2	-	-	-	1	-
4	2	1	2	1	2	-	-	-	3	-	-	1	2	1
5	2	1	2	2	3	-	-	-	3	-	-	1	3	1
6	2	1	3	2	3	-	-	-	3	-	-	1	3	1

1	Reasonably agreed	2	Moderately agreed	3	Strongly agreed

Department of Electronics and Communication Engineering
(Accredited by NBA)

Name of Lab Course	**22EC505 MICROCONTROLLERS LABORATORY**
Semester & Year	**5th & 3rd year**
Name of the student	
Class	
Name of the Evaluator	
Marks scored out of 100	

RUBRIC ASSESSMENT FOR MICROCONTROLLERS LABORATORY

NAME OF THE LABORATORY : 22EC505 – MICROCONTROLLERS LABORATORY						
Sl. No.	Criteria	Maximum Marks	Excellent	Good	Average	Poor
			91% - 100%	71% - 90%	50% - 70%	<50%
1	Objective & Componentsrequired	10	The aim and purpose of the experiment is clearly defined. The components required are clearly listed (9 – 10)	Few Preliminary questions regarding objective are notanswered (6-8)	Aim and purposeof the experimentare defined with less details and unable to explain (3-5)	Not clear about the objectives and explanation (<3)
2	Algorithm & Source Code	30	The required algorithm & coding is writtencorrectly according to the logic, commentsand explanations given appropriately (25-30)	The required algorithm & coding iswritten with very fewerrors. Able to list most comments and give explanation to some extend (20-24)	The required algorithm & coding is writtenwith errors (10-19)	The required algorithm& coding just attempted(<10)
3	Compiling and Debugging	30	Able to performthe required compilation andinterpret correctresults (25-30)	Almost performed the compilation and incorrectly interpreted (20-24)	Performed the compilation to some extend and not able to interpret(10-19)	Not able to understand what is compilation and implementation (<10)

4	Results and discussion	20	The program isexecuted and the required output is obtained .Clearly summarize the results of the experiment andrelate them back to the objectives (15-20)	The program is executed and a partialoutput is obtained. Moderately summarize the resultsof the experiment and relate them back to the objectives (10-14)	Irrelevant output and the inferenceof the experimentis not done. (9-4)	Output was not obtained (<4)
5	Documentation	5	Printouts and Documentation fully completed. Neatly documented. On date Submission (4-5)	Printouts and Documentation mostly completed . . . Relevant documentation (5-3)	Incomplete work with meager presentation (3-2)	Late submission(1-0)
6	Viva	5	Understoodthe program clearly. Answered allquestions (4-5)	Understood the program clearly. Answered partialquestions (5-3)	Not able to understand thelogic of the program. Answered partial questions (3-2)	Not able to understand thelogic of the program. Not able to answer(1-0)
Signature of the Faculty Member						

TABLE OF CONTENTS

Microcontroller Basics

A microcontroller is a computer-on-a-chip, or a single-chip computer. *Micro*suggests that the device is small, and *controller* tells that the device might be used to control objects, processes, or events. Another term to describe a microcontroller is *embedded controller,* because the microcontroller and its support circuits are often built into, orembedded in, the devices they control.

About the 8051

The Intel 8051 is an 8-bit microcontroller which means that most available operations are limited to 8 bits. All the ICs from a generic model in general are called 8051 because they can all be programmed using 8051 assembly language, and they all share certain features (although the different models all have their own special features).

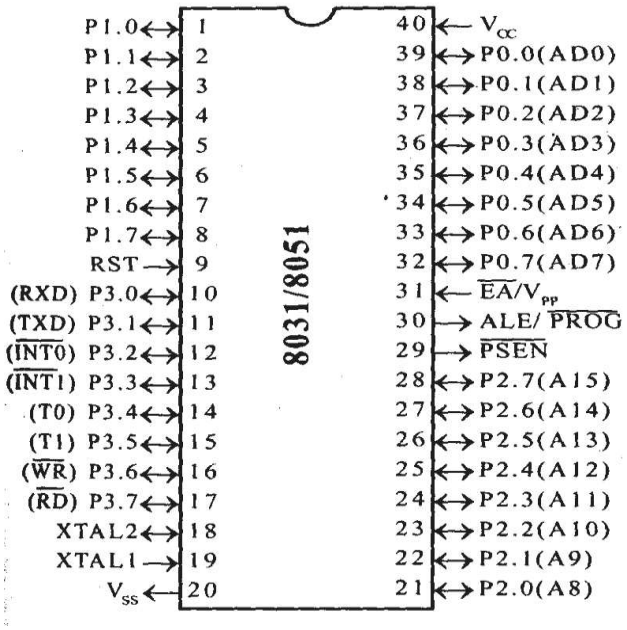
Some of the features that have made the 8051 popular are:

- 8-bit data bus
- 16-bit address bus
- 34 general purpose registers each of 8 bits
- 16 bit timers (usually 2, but may have more, or less).
- 3 internal and 2 external interrupts.
- Bit as well as byte addressable RAM area of 16 bytes.
- 4 8-bit ports, (short models have 2 8-bit ports).
- 16-bit program counter and data pointer

8051 models may also have a number of special, model-specific features, such as UARTs, ADC, Op-Amps, etc...

Typical applications

8051 chips are used in a wide variety of control systems, telecom applications, robotics as well as in the automotive industry. By some estimations, 8051 family chips make up over 50% of the embedded chip market.



Basic Pins

PIN 9: PIN 9 is the reset pin which is used reset the microcontroller's internal registers and ports upon starting up.

PINS 18 & 19: The 8051 has a built-in oscillator amplifier hence we need to only connect a crystal at these pins to provide clock pulses to the circuit.

PIN 40 and 20: Pins 40 and 20 are VCC and ground respectively. The 8051 chip needs +5V 500mA to function properly, although there are lower powered versions like the Atmel 2051 which is a scaled down version of the 8051 which runs on +3V.

PINS 29, 30 & 31: As described in the features of the 8051, this chip contains a built-in flash memory. In order to program this we need to supply a voltage of +12V at pin 31. If external memory is connected then PIN 31, also called EA/VPP, should be connected to ground to indicate the presence of external memory. PIN 30 is called ALE (address latch enable), which is used when multiple memory chips are connected to the controller and only one of them needs to be selected. We will deal with this in depth in the later chapters. PIN 29 is called PSEN. This is "program select enable". In order to use the external memory it is required to provide the low voltage (0) on both PSEN and EA pins.

Ports

There are 4 8-bit ports: P0, P1, P2 and P3.

PORT P1 (Pins 1 to 8): The port P1 is a general purpose input/output port which can be used for a variety of interfacing tasks. The other ports P0, P2 and P3 have dual roles or additional functions associated with them based upon the context of their usage.

PORT P3 (Pins 10 to 17): PORT P3 acts as a normal IO port, but Port P3 has additional functions such as, serial transmit and receive pins, 2 external interrupt pins, 2 external counter inputs, read and write pins for memory access.

PORT P2 (pins 21 to 28): PORT P2 can also be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P2 will act as an address bus in conjunction with PORT P0 to access external memory. PORT P2 acts as A8-A15, as can be seen from fig 1.1

PORT P0 (pins 32 to 39) PORT P0 can be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P0 acts as a multiplexed address and data bus that can be used to access external memory in conjunction with PORT P2. P0 acts as AD0-AD7.

Oscillator Circuits

The 8051 requires the existence of an external oscillator circuit. The oscillator circuit usually runs around 12MHz, although the 8051 (depending on which specific model) is capable of running at a maximum of 40MHz. Each machine cycle in the 8051 is 12 clock cycles, giving an effective cycle rate at 1MHz (for a 12KHz clock) to 3.33MHz (for the maximum 40MHz clock).

Data and Program Memory

The 8051 Microprocessor can be programmed in PL/M, 8051 Assembly, C and a number of other high-level languages. Many compilers even have support for compiling C++ for an 8051.

Program memory in the 8051 is read-only, while the data memory is considered to be read/write accessible. When stored on EEPROM or Flash, the program memory can be rewritten when the microcontroller is in the special programmer circuit.

Program Start Address

The 8051 starts executing program instructions from address 0x00 in the program memory.

Direct Memory

The 8051 has 256 bytes of internal addressable RAM, although only the first 128 bytes are available for general use by the programmer. The first 128 bytes of RAM (from 0x00 to 0x7F) are called the **Direct Memory**, and can be used to store data.

Special Function Register

The **Special Function Register** (SFR) is the upper area of addressable memory, from address 0x80 to 0xFF. This area of memory cannot be used for data or program storage, but is instead a series of memory-mapped ports and registers. All port input and output can therefore be performed by memory **mov** operations on specified addresses in the SFR. Also, different status registers are mapped into the SFR, for use in checking the status of the 8051, and changing some operational parameters of the 8051.

General Purpose Registers

The 8051 has 4 selectable banks of 8 addressable 8-bit registers, R0 to R7. This means that there are essentially 32 available general purpose registers, although only 8 (one bank) can be directly accessed at a time. To access the other banks, we need to change the current bank number in the flag status register.

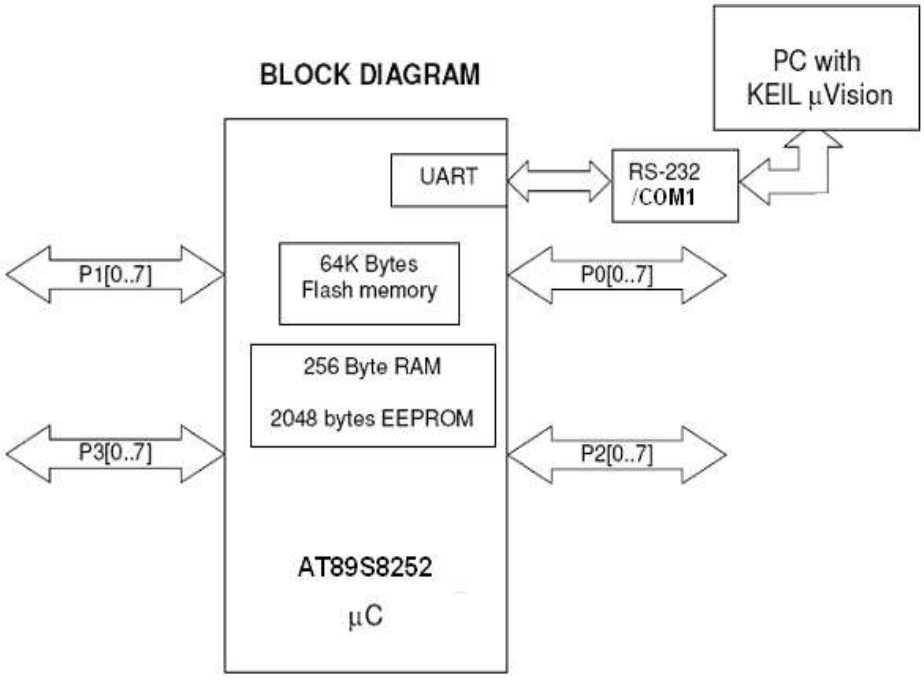
A and B Registers

The A register is located in the SFR at memory location 0xE0. The A register works in a similar fashion to the AX register of x86 processors. The A register is called the **accumulator**, and by default it receives the result of all arithmetic operations. The B register is used in a similar manner, except that it can receive the extended answers from the multiply and divide operations. When not being used for multiplication and Division, the B register is available as an extra general-purpose register.

Introduction

Microcontroller used is Atmel AT89S8252

8051 based Full Static CMOS controller with Three-Level Program, Memory Lock, 32 I/O lines, 3 Timers/Counters, 9 Interrupts Sources, SPI, Watchdog Timer, 2 DPTRs, 8K Flash Memory, 2k EEPROM, 256 Bytes On-chip RAM



INTEL 8051 MICRO CONTROLLER KIT INTERFACED WITH KEYBOARD

Exp: 1
Date:

8051 ARITHMETIC OPERATIONS

AIM:

To perform arithmetic operations and store the result in memory using 8051 microcontrollers.

16-BIT ADDITION WITHOUT CARRY

ALGORITHM

1. Start the program.
2. Get the first 8 bit data (MSB) of the 16 bit in accumulator.
3. Add another 8 bit data (MSB) with the accumulator content.
4. Move the address location to Data Pointer.
5. Store the added result (MSB) in the Data Pointer.
6. Get the first 8 bit data (LSB) of the 16 bit in accumulator.
7. Add another 8 bit data (LSB) with the accumulator content.
8. Increment the Data Pointer.
9. Store the added result (LSB) in the Data Pointer.
10. Stop the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		Clr C	Clear carry
		MOV A,# DATA 1	
		ADD A,# DATA 2	
		MOV DPTR,#4500	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,#DATA 3	
		ADDC A,# DATA 4	
		MOVX @DPTR,A	
		HERE: SJMP HERE	

Input:

Output:

16-BIT ADDITION WITH CARRY

ALGORITHM

1. Start the program.
2. Get the first 8 bit data (MSB) of the 16 bit in accumulator.
3. Add another 8 bit data (MSB) with the accumulator content.
4. Move the address location to Data Pointer.
5. Store the added result (MSB) in the Data Pointer.
6. Get the first 8 bit data (LSB) of the 16 bit in accumulator.
7. Add another 8 bit data (LSB) with the accumulator content.
8. Increment the Data Pointer.
9. Store the added result (LSB) in the Data Pointer.
10. Check with carry
11. Display output as 01 as carry else 00
12. Stop the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		Clr C	
		MOV A,# DATA 1	
		ADD A,# DATA 2	
		MOV DPTR,#4500	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,#DATA 3	
		ADDC A,# DATA 4	
		MOVX @DPTR,A	
		INC DPTR	
		MOV R0, #00	
		JNC L1	
		INC R0	
		L1: MOV A,R0	
		L2: MOV @DPTR, A	
		HERE : SJMP HERE	

Input:

Output:

16-BIT SUBTRACTION WITHOUT BORROW

ALGORITHM

1. Start the program.
2. Get the first 8 bit data (MSB) of the 16 bit in accumulator.
3. Sub another 8 bit data (MSB) with the accumulator content.
4. Move the address location to Data Pointer.
5. Store the added result (MSB) in the Data Pointer.
6. Get the first 8 bit data (LSB) of the 16 bit in accumulator.
7. Sub another 8 bit data (LSB) with the accumulator content.
8. Increment the Data Pointer.
9. Store the added result (LSB) in the Data Pointer.
10. Stop the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		Clr C	
		MOV A,# DATA 1	
		SUBB A,# DATA 2	
		MOV DPTR,#4500	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,#DATA 3	
		SUBB A,# DATA 4	
		MOVX @DPTR,A	
		HERE: SJMP HERE	

Input:

Output:

16-BIT SUBTRACTIONWITH BORROW

ALGORITHM

1. Start the program.
2. Get the first 8 bit data (MSB) of the 16 bit in accumulator.
3. Subtract another 8 bit data (MSB) with the accumulator content.
4. Move the address location to Data Pointer.
5. Store the added result (MSB) in the Data Pointer.
6. Get the first 8 bit data (LSB) of the 16 bit in accumulator.
7. Subtract another 8 bit data (LSB) with the accumulator content.
8. Increment the Data Pointer.
9. Store the added result (LSB) in the Data Pointer.
10. Check with carry
11. Display output as 01 as borrow else 00
12. Stop the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		Clr C	
		MOV A,# DATA 1	
		SUBBA,# DATA 2	
		MOV DPTR,#4500	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,#DATA 3	
		SUBBA,# DATA 4	
		MOVX @DPTR,A	
		INC DPTR	
		MOV R0, #00	
		JNC L1	
		INC R0	
		L1: MOV A,R0	
		L2: MOV @DPTR, A	
		HERE: SJMP HERE	

16-BIT MULTIPLICATION

ALGORITHM:

1. Get the data in A reg. Get the value to be multiplied in D reg.
2. Multiply the data. The high and low nibble of result is stored in A & B reg.
3. Terminate the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		MOV DPTR, #4500	
		MOV R1, #MSB1	
		MOV R2, #MSB2	
		MOV R3, #LSB1	
		MOV R4, #LSB2	
		MOV A,R3	
		MOV B,R4	
		MUL AB	
		MOVX @DPTR,A	
		INC DPTR	
		MOV R6,B	
		MOV A,R3	
		MOV B,R2	
		MUL AB	
		MOV R7,B	
		ADDC A,R6	
		MOV R6,A	
		MOV A,R1	

		MOV B,R4	
		MUL AB	
		ADDC A,R6	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,B	
		ADDC A,R7	
		MOV R7,A	
		MOV A,R1	
		MOV B, R2	
		MUL AB	
		ADDC A,R7	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,B	
		ADDC A,#00	
		MOVX @DPTR,A	
		HERE: SJMP HERE	

Input:

Output:

8-BIT DIVISION

ALGORITHM:

1. Get the data in A reg.
2. Get the value to be divided in B reg.
3. Divide the data.
4. The Quotient is in Areg and remainder is in B reg.
5. Terminate the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		MOV A,# DATA 1	
		MOV **B**,# DATA 2	
		DIVA,B	
		MOV DPTR,#4500	
		MOVX @DPTR,A	
		INC DPTR	
		MOV A,B	
		MOVX @DPTR,A	
		HERE: SJMP HERE	

Input:

Output:

BIT MANIPULATION

ALGORITHM:

1. Get the data bitwise in memory.
2. Get each bit in A.
3. Set and Reset A according to requirement.
4. Save in Memory.
5. Terminate the program.

PROGRAM

Memory Address	Object code	Mnemonics	Comment
		MOV DPTR, #4500	
		MOVX A,@DPTR	
		CLR A	
		MOVX @DPTR,A	
		INC DPTR	
		MOVX A,@DPTR	
		CLR A	
		MOVX @DPTR,A	
		INC DPTR	
		INC DPTR	
		INC DPTR	
		MOVX A,@DPTR	
		SETB A	
		MOVX @DPTR, A	
		L1:SJMP L1	

RESULT:

Exp: 2
Date:

DATA TRANSFER, DATA EXCHANGE AND SORTING USING 8051

AIM:

To perform the
(A) Data transfer using Block Move
(B) Data transfer using Exchange
(C) Largest number in an array of data
(D) Smallest number in an array of data
(E) Ascending and descending order in an array of data using 8051microcontroller.

Data Transfer (BlockMove):

ALP to move block of data bytes present in internal memory with starting address 10h and ending address20h to the destination memory with starting address 30h(Without overlap).

Algorithm:

1) Start the program.
2) Load the values in a block of memory.
3) Move the values to another block of memory using LOOP conditions.
4) End.

Program:

MemoryAddress	Opcode	Label	Mnemonics	Comments
			MOV R3, #0A	
			MOV DPH,#45	
			MOV R0,#35H	
			MOV R1,#41H	
		BACK:	MOV DPL,R0	
			MOVX A,@DPTR	
			MOV DPL,R1	
			MOVX @DPTR,A	
			INC R0	
			INC R1	

			DJNZ R3,BACK	
		HERE:	SJMP HERE	

Input:

4535:
4536:
4537:
4538:
4539:
453A:
453B:
453C:
453D:
453E:

Output:

4541:
4542:
4543:
4544:
4545:
4546:
4547:
4548:
4549:
454A:

Block Exchange (Exchange of data):

Algorithm:

1. Start
2. Get the first number in Accumulator
3. Get the second number in R_0
4. Swap A and exchange with R_0
5. Display the result
6. Stop

MemoryAddress	Opcode	Label	Mnemonics	Comments
			MOV R0,#05	
			MOV DPH,#45H	
			MOV R1,#35H	

			MOV R2,#41H	
		LOOP1;	MOV DPL,R1	
			MOVX A,@DPTR	
			MOV R3,A	
			MOV DPL,R2	
			MOVX A,@DPTR	
			XCH A,R3	
			MOVX @DPTR,A	
			MOV DPL,R1	
			MOV A,R3	
			MOVX @DPTR,A	
			INC R1	
			INC R2	
			DJNZ R0,LOOP1	
		HERE	SJMP HERE	

Input:
Set 1:
4535:
4536:
4537:
4538:
4539:
453A:
453B:
453C:
453D:
453E:

Set 2:

4541:
4542:
4543:
4544:
4545:
4546:
4547:
4548:
4549:
454A:

Output:

Set 1:
4535:
4536:
4537:
4538:
4539:
453A:
453B:
453C:
453D:
453E:

Set 2:

4541:
4542:
4543:
4544:
4545:
4546:
4547:
4548:
4549:
454A:

LARGEST OF AN ARRAY

1) Start the program.
2) Initialize the counter and pointer.
3) Load the internal memory location40h with zero.
4) Store the value 10 in the register.
5) Get the input value and check if it isequalto10.
6) Use if loop to increment the pointer and decrement the counter.
7) Store and display the final result in 40h.
8) Stop the program.

PROGRAM

(a) Largest Number

MemoryAddress	Opcode	Mnemonics	Comments
		MOVDPTR,#4200	
		MOV40H,#00	
		MOVR5,#0AH	
		LOOP2: MOVXA,@DPTR	
		CJNEA,40H, LOOP1	
		LOOP3:INCDPTR	
		DJNZ R5, LOOP2	
		MOVA,40H	
		MOVX@DPTR, A	
		HLT:SJMPHLT	
		LOOP1: JCLOOP3	
		MOV40H, A	
		SJMP LOOP3	

Input:
4200-
4201-
4202-
4203-
4204-
4205-
4206-
4207-
4208-
4209-

Output:
420A-

(b) Smallest number

MemoryAddress	Opcode	Mnemonics	Comments
		MOVDPTR,#4200	
		MOV40H,#FF	
		MOVR5,#0AH	
		LOOP2: MOVXA,@DPTR	
		CJNEA,40H, LOOP1	
		LOOP3:INCDPTR	
		DJNZ R5, LOOP2	
		MOVA,40H	
		MOVX@DPTR, A	
		HLT:SJMPHLT	
		LOOP1: JNCLOOP3	
		MOV40H, A	
		SJMP LOOP3	

Input:
4200-
4201-
4202-
4203-
4204-
4205-
4206-
4207-
4208-
4209-

Output:
420A-

(iii) Sorting – Ascending :

Algorithm

1) Start the program.
2) Initialize the counter and pointer.
3) Point the data pointer low to R5 and data pointer high to R6.
4) Get first element in B and second element in A.
5) Check if A is greater than B.
6) Display the stored result.
7) Stop the program.

Program:

MemoryAddress	Opcode	Mnemonics	Comments
		MOV R0, #05	
		DEC R0	
		LOOP 1:MOV DPTR, #4500	
		MOV A, R0	
		MOV R1, A	
		LOOP 2; MOVX A, @DPTR	
		MOV R2, A	
		INC DPTR	
		MOVX A, @DPTR	
		SUBB A, R2	
		JC LOOP3	
		MOVX A,@ DPTR	
		XCH A,R2	
		MOVX @ DPTR,A	
		DEC DPL	
		MOV A,R2	
		MOVX A,@ DPTR,	
		INC DPR	

		LOOP3;DJNZ R0,LOOP1	
		L4:SJMP L4	

(iv) Sorting – Descending :

Algorithm

1) Start the program.
2) Initialize the counter and pointer.
3) Point the data pointer low to R5 and data pointer high to R6.
4) Get first element in B and second element in A.
5) Check if A is smaller than B.
6) Display the stored result.
7) Stop the program.

Program:

MemoryAddress	Opcode	Mnemonics	Comments
		MOV R0, #05	
		DEC R0	
		LOOP 1:MOV DPTR, #4500	
		MOV A, R0	
		MOV R1, A	
		LOOP 2; MOVX A, @DPTR	
		MOV R2, A	
		INC DPTR	
		MOVX A, @DPTR	
		SUBB A, R2	
		JNC LOOP3	
		MOVX A,@ DPTR	
		XCH A,R2	
		MOVX @ DPTR,A	
		DEC DPL	
		MOV A,R2	
		MOVX A,@ DPTR,	
		INC DPR	
		LOOP3;DJNZ R0,LOOP1	
		L4:SJMP L4	

Result:

Exp: 3
Date:

PROGRAM USING CONDITIONAL CALL AND RETURN

AIM:

To perform Factorial of a number using conditional call and return.

ALGORITHM:

1) Start the program.
2) Input the number whose factorial to be found as a counter.
3) Call the subroutine FACT.
4) End the main program
5) Continue the programming in subroutine.
6) Until value of counter is 0, Continue to Decrement counter and Multiply Counter with Accumulator
7) Return to the main program when the counter is zero.
8) Store and display the final result in 4500h.

PROGRAM:

MemoryAddress	Opcode	Label	Mnemonics	Comments
			MOV R0, #05	
			MOV A,R0	
			LCALL L2	
			LCALL 4500	
		L2	CJNE R0,#01, L1	
			RET	
		L1	DEC R0	
			MOV B, R0	
			MUL AB	
			LJMP L2	
		4500	MOV DPTR,#5000	
			MOVX @DPTR,A	
		HERE	SJMP HERE	

Input:

Output:

Result:

Exp: 4
Date:

STEPPER MOTOR INTERFACE WITH 8051

Algorithm:

1) Start the program.
2) Initialize the data pointer to 4500H.
3) Store the R0 with 04H.
4) Store the input at accumulator.
5) Push value DPH and DPL.
6) Move the accumulator value to DPH.
7) POP the DPH and DPL values.
8) Give the input for clockwise and anticlockwise direction.
9) Stop the program.

Program:

Memory Address	Opcode	Mneumonics	Comments

		START : MOV DPTR,#4500
		
		MOV R0,#04
		
		J0: MOVX A,@DPTR
		PUSH DPH
		
		PUSH DPL
		
		MOV DPTR,#FFC0
		
		MOV R2,#04H
		
		MOV R1,#0FH
		
		DLY 1: MOV R3,#0FH
		
		DLY : DJNZ R3,DLY
		
		DJNZ R1,DLY1
		
		DJNZ R2,DLY1
		
		MOVX @DPTR,A
		POP DPL	

		POP DPH INC DPTR DJNZ R0,J0 SJMP START 09,05,06,0A	

Result:

Thus the stepper motor is interfaced with 8051 and rotated in both clockwise and anticlockwise direction.

Exp: 5
Date:

PROGRAMMABLE PERIPHERAL INTERFACE WITH 8051 MICROCONTROLLER

AIM:
To perform various operations by interfacing programmable peripheral interface with 8051 microcontrollers.

ALGORITHM-1 (Port A)

1.Move FFC6 to DPTR (Initialize Control Register)

2.Move 90 to A register

3.Move A value to DPTR

4.Move FFC0 to DPTR (Initialize Port A)

5.DPTR Value is moved to A register

6.Output is seen at 4500.

ALGORITHM-2 (Port B)

1.Move FFC6 to DPTR (Initialize Control Register)

2.Move 90 to A register

3.Move A value to DPTR

4.Move FFC0 to DPTR (Initialize Port A to give input using SPDT Switch)

5.DPTR Value is moved to A register

6. Move FFC0 to DPTR (Initialize Port A)

7.Output is seen at Port B.

ALGORITHM-3 (Port C)

1.Move FFC6 to DPTR (Initialize Control Register)

2.Move 80 to A register

3.Move A value to DPTR

4.Move FFC4 to DPTR (Initialize Port C)

5.DPTR Value is moved to A register

6.Output is seen at PORT C.

ALGORITHM-4

Port C Set Mode

1.Move FFC6 to DPTR (Initialize Control Register)

2.Move 80 to A register

3.Move A value to DPTR

4.Move FFC4 to DPTR (Initialize Port C)

5. Move A with 01 (To SET A = 01, & for RESET A=00)

6.DPTR Value is moved to A register

7. Move FFC6 to DPTR (Initialize Control Register)

8. Move A with a data (eg.011)

9.DPTR Value is moved to A register

10.Output is seen at PORT C.

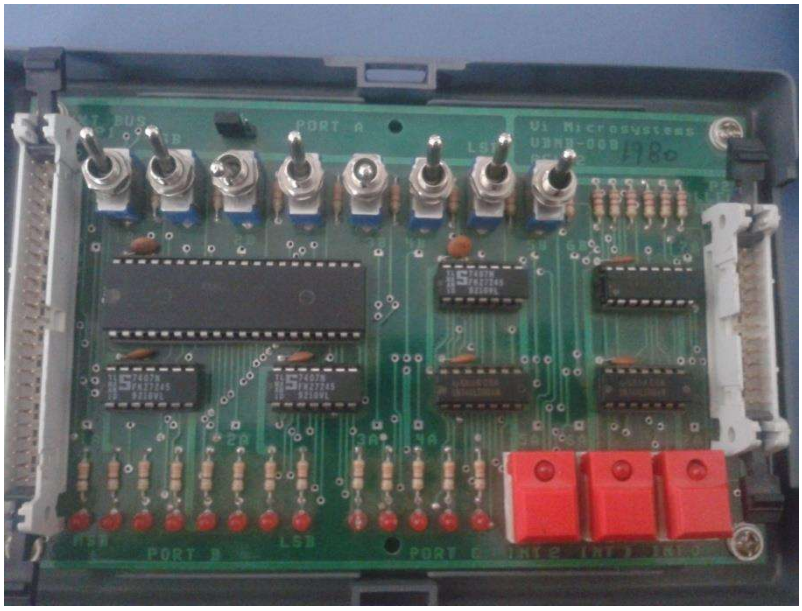
Theory:

The 8255 has been decided as general purpose programmable interface I/O device,compatible with Intel Microprocessors.It contains three 8 bit ports which can be configured by software means to provide any one of the three programmable data transfer modes available with 8255 .

PORT-A
One 8-bit data output latch/buffer and one 8-bit data input latch.port A can functions as input or output ports in two modes.

PORT-B
One 8-bit data output latch/buffer and one 8-bit data input buffer.Port B can functions as input or output port in two modes.

PORT-C
One 8-bit output latch/buffer and one 8-bit data input buffer .Port C can function as simple input or output port.This port can be divided into two 4-bit ports,which in turn can function as simple input or output port.In addition the Port C lines can be used for the control signal outputs and ststus signal inputs in conjunction with Port A and Port B.



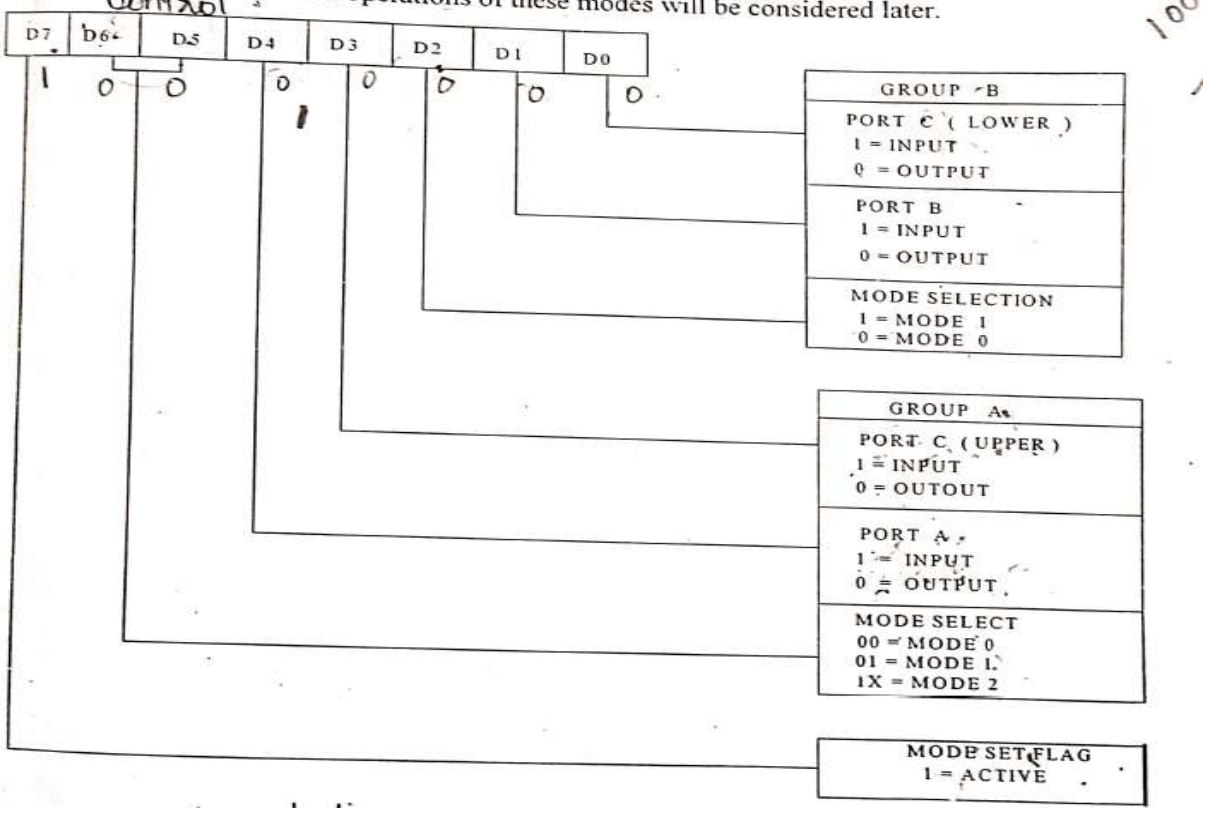
4

8255 PROGRAMMABLE PERIPHERAL INTERFACE

CONTROL WORD :

The control word is of 8-bit wide. Bit 7 (MSB) decides whether the mode set operation or bit set/reset operation is selected.

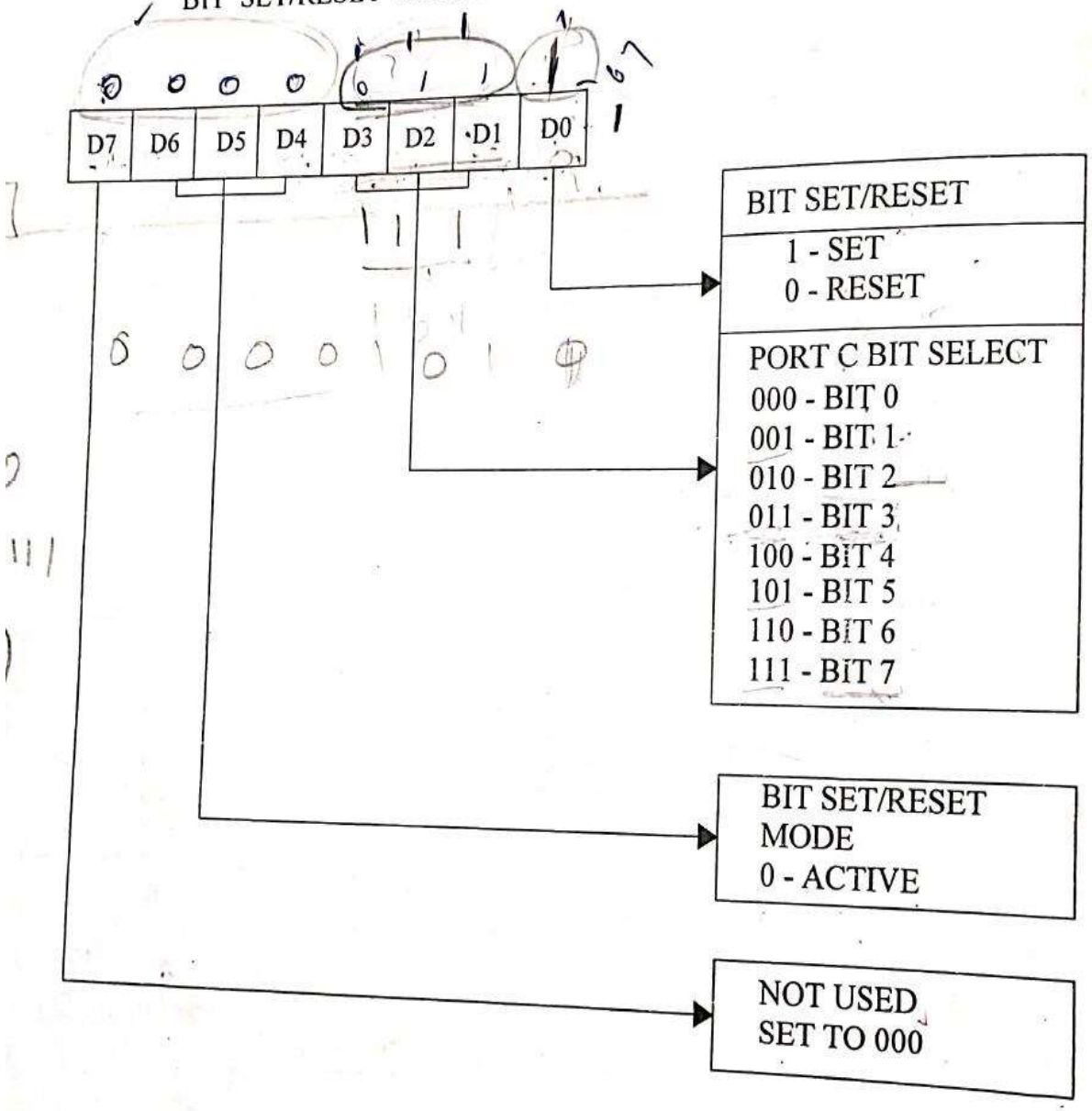
With Bit 7 =1, Bit 6,5,4 and 3 are to set the mode of Group A while Bit 2,1 and 0 are to set the mode of group B. Detailed operations of these modes will be considered later.



.0

	A7	A6	A5	A4	A3	A2	A1	A0	HEX
CONTROL REG	1	1	0	0	0	1	1	x	C6
PORT A	1	1	0	0	0	0	0	x	C0
PORT B	1	1	0	0	0	0	1	x	C2
PORT C	1	1	0	0	0	1	0	x	C4

BIT SET/RESET CONTROL WORD FORMAT OF THE 8255



PROGRAM:

1. To initialize Port A as an input port in Mode 0 & to input data set by the SPDT switches through Port A and output at location specified at DPTR.

ADDRESS	OPCODE	MNEMONICS	COMMENT
		PORT A	
		MOV DPTR,#FFC6H	
		MOV A,#90	
		MOVX @DPTR,A	
		MOV DPTR ,#FFC0H	
		MOVX A,@DPTR	
		MOV DPTR, #4500	
		MOVX @DPTR,A	
		SJMP 410E	

2. To initialize Port A as input Port and Port B as output Port in Mode 0. To input data at Port as set by SPDT switch & to output the same data to Port B to Glow the LEDs accordingly.

ADDRESS	OPCODE	MNEMONICS	COMMENT
		PORT-B	
		MOV DPTR,#FFC6H	
		MOV A,#90	
		MOV X@DPTR,A	
		MOV DPTR,#FFC0	
		MOVX A,@DPTR	
		MOV DPTR,#FFC2	
		MOVX@DPTR,A	
		SJMP 410E	

3. To initialze Port C as an output Port in Mode 0

ADDRESS	OPCODE	MNEMONICS	COMMENT
		PORT C	
		MOV DPTR,#FFC6H	
		MOV A,#80	
		MOVX @DPTR,A	
		MOV DPTR ,# FFC4H	
		MOV A,#80	
		MOVX @DPTR,A	
		HLT:SJMP HLT	

4. To initialize Port C as Output Port in Mode 0 and to access the SET/RESET feature of Port C

a. SET

ADDRESS	OPCODE	MNEMONICS	COMMENT
		PORT-C (SET MODE)	
		MOV DPTR,#FFC6H	
		MOV A,#80	
		MOVX @DPTR,A	
		MOV DPTR,#FFC4H	
		MOV A ,#01	
		MOV X@DPTR,A	
		MOV DPTR,#FFC6H	
		MOV A,#07	
		MOVX @DPTR,A	
		HLT:SJMP HLT	

b. RESET

ADDRESS	OPCODE	MNEMONICS	COMMENT
		PORT-C (SET MODE)	
		MOV DPTR,#FFC6H	
		MOV A,#80	
		MOVX @DPTR,A	
		MOV DPTR,#FFC4H	
		MOV A ,#00	
		MOV X@DPTR,A	
		MOV DPTR,#FFC6H	
		MOV A,#07	
		MOVX @DPTR,A	
		HLT:SJMP HLT	

RESULT:

Thus, the port programming of 8255 processor with 8051 controllers has been performed.

Exp: 6
Date:

KEYBOARD AND SEVEN SEGMENT INTERFACING WITH 8051 MICROCONTROLLER

AIM:

To interface keyboard and Seven Segment with 8051.

DISPLAY CHARACTER A

ALGORITHM:

1) Start the program.
2) Initialize the data pointer to FFC2H to upload control word.
3) Send Control word 00, CC, 90 to return home, clear, write command .
4) Initialize the data pointer to FFC0H to upload data.
5) Send value 88 which is seven segment value of A to dataregister
6) Introduce a delay
7) Stop the program

PROGRAM

ADDRESS	OPCODE	MNEMONICS	COMMENT
		MOV DPTR, #FFC2	
		MOV A, #00	
		MOVX @DPTR, A	
		MOV A, #CC	
		MOVX @DPTR, A	
		MOV A, #90	
		MOVX @DPTR,A	
		MOV DPTR, #FFC0	
		MOV A, #88	
		MOVX @DPTR, A	
		MOV R0, #05	
		MOV A, #FF	
		LOOP: MOVX @DPTR, A	
		DJNZ R0, LOOP	
		HERE: SJMP HERE	

ROLLING DISPLAY:

ALGORITHM:

1) Start the program.
2) Initialize the data pointer to FFC2H to upload control word.
3) Send Control word 00, CC, 90 to return home, clear, write command.
4) Initialize the data pointer to 4400 (look up table address).
5) Send data from look up table (contains seven segment value of the letters to be displayed)one by one with a delay
6) Go back to Step 2

PROGRAM:

ADDRESS	OPCODE	MNEMONICS	COMMENT
		START: MOV DPTR, #FFC2	
		MOV R0, #00	
		MOV R1, #44	
		MOV A, #10	
		MOVX @DPTR,A	
		MOV A, #CC	
		MOVX @DPTR,A	
		MOV A, #90	
		MOVX @DPTR,A	
		LOOP: MOV DPH, R1	
		MOV DPL,R0	
		MOVX A, @DPTR	
		MOV DPTR, #FFC0	
		MOVX @DPTR,A	
		LCALL DELAY	
		INC R0	
		CJNE R0, #0F, LOOP	
		LJMP START	
DELAY			
		MOV R4,#A0	
		LOOP2: MOV R5,#FF	
		LOOP1: NOP	
		DJNZ R5, LOOP1	
		DJNZ R4, LOOP2	
		RET	
LOOK UP TABLE			
4400	FF FF FF FF		
4404	FF FF FF FF		
4408	98 68 7C C8		
440C	FF 1C 29 FF		

Result:
Thus, keyboard and seven segment are interfaced successfully using 8051.

Exp: 7
Date:

ADC & DAC INTERFACING WITH 8051 MICROCONTROLLER

AIM:

To interface ADC and DAC with 8051.

ALGORITHM:

1) Start the program.
2) Set the control word for selecting channel 0 and make ALE Low
3) Make ALE High.
4) Make the start of conversion low and high to start conversion.
5) Check for End of conversion by checking no bit in E0 and remain in wait state.
6) Once there is bit in E0 then read the analog to digital converted data.
7) Store the converted digital data to memory location.
8) End the Program.

PROGRAM:

ADDRESS	OPCODE	MNEMONICS	COMMENT
		MOV DPTR,#FFC8	
		MOV A,#10	
		MOVX @DPTR,A	
		MOV A,#18	
		MOVX @DPTR,A	
		MOV DPTR,#FFD0	
		MOV A,#01	
		MOVX @DPTR,A	
		MOV A,#00	
		MOVX @DPTR,A	
		MOV DPTR,#FFE0	
		WAIT: MOVX A,@DPTR	
		JNB E0,WAIT	
		MOV DPTR,#FFC0	
		MOVX A,@DPTR	
		MOV DPTR,#4150	
		MOVX @DPTR,A	
		HERE : SJMP HERE	

Output:

Vin(v)	Vout (V)		
	Practical Value		Theoretical Value
	Memory (4150 location)	LED Output	Calculated Formula value

DAC SQUARE WAVE

ALGORITHM:

1) Start the program.
2) Set the control word for interfacing DAC interface with 8051.
3) Get the hex value inputs for generating the square, triangular and saw tooth waveforms and see the output in CRO display and note the values of magnitude and time period of wave.
4) End the Program.

PROGRAM:

ADDRESS	OPCODE	MNEMONICS	COMMENT
		MOV DPTR, #E0C8	
		START : MOV A, #00	
		MOVX DPTR, A	
		LCALL DELAY	
		MOV A, #FF	
		MOVX @DPTR, A	
		LCALL DELAY	
		LJMP START	
		DELAY : MOV R1, #05	
		LOOP : MOV R2, #FF	
		DJNZ R2, HERE	
		DJNZ R1, LOOP	
		RET	
		SJMP START	

SAW TOOTH

ADDRESS	OPCODE	MNEMONICS	COMMENT
		MOV DPTR, #E0C0	
		MOV A, #00	
		LOOP : MOVX @DPTR, A	
		INC A	
		SJMP LOOP	

TRIANGULAR WAVE FORM

ADDRESS	OPCODE	MNEMONICS	COMMENT
		MOV DPTR, #E0C8	
		START : MOV A, #00	
		LOOP1 : MOVX @DPTR, A	
		INC A	
		JNZ LOOP1	
		MOVX A, #FF	
		LOOP2 : MOVX @DPTR, A	
		DEC A	
		JNZ LOOP2	
		LJMP START	

Output:

S.No.	Waveform	Magnitude	Time Period
1.	Square		
2.	Triangular		
3.	Sawtooth		

Result:

Thus the ADC and DAC are interfaced successfully using 8051.

STUDY OF IAR EMBEDDED WORKBENCH

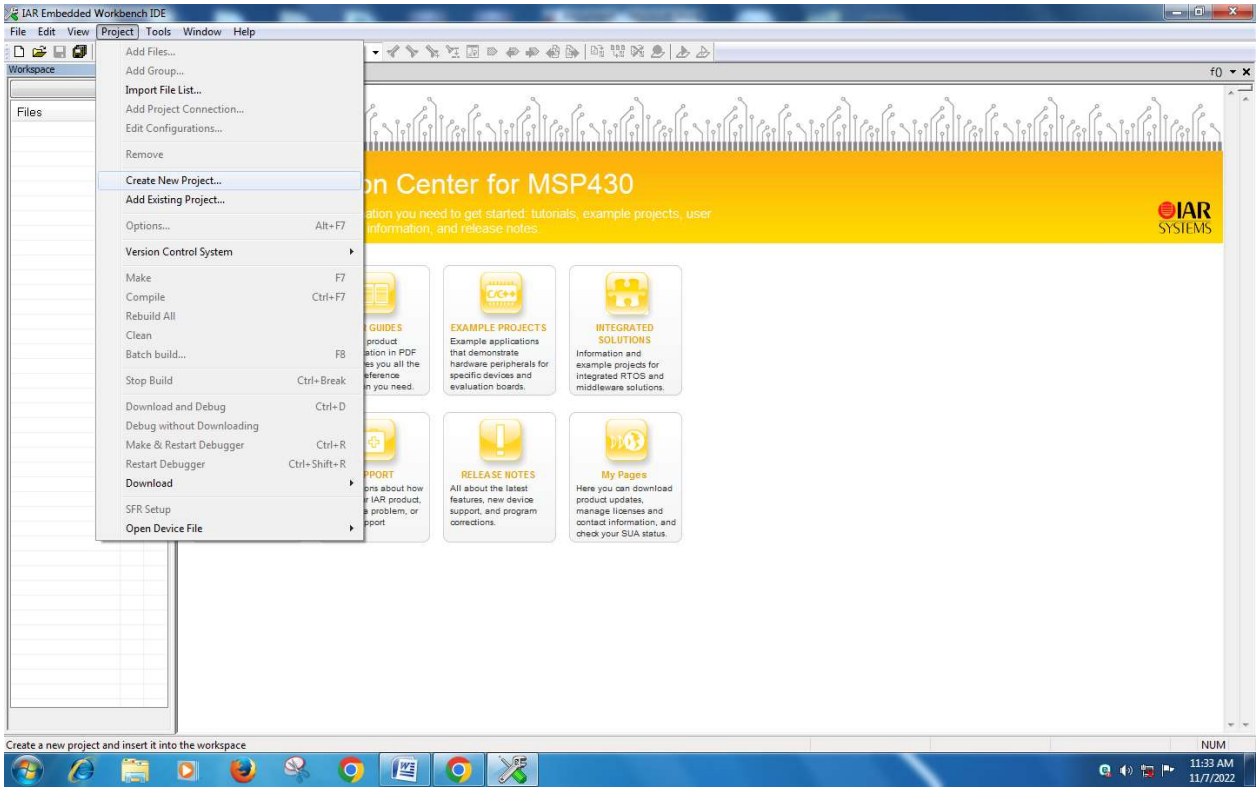
AIM:

To study software IAR Embedded Workbench

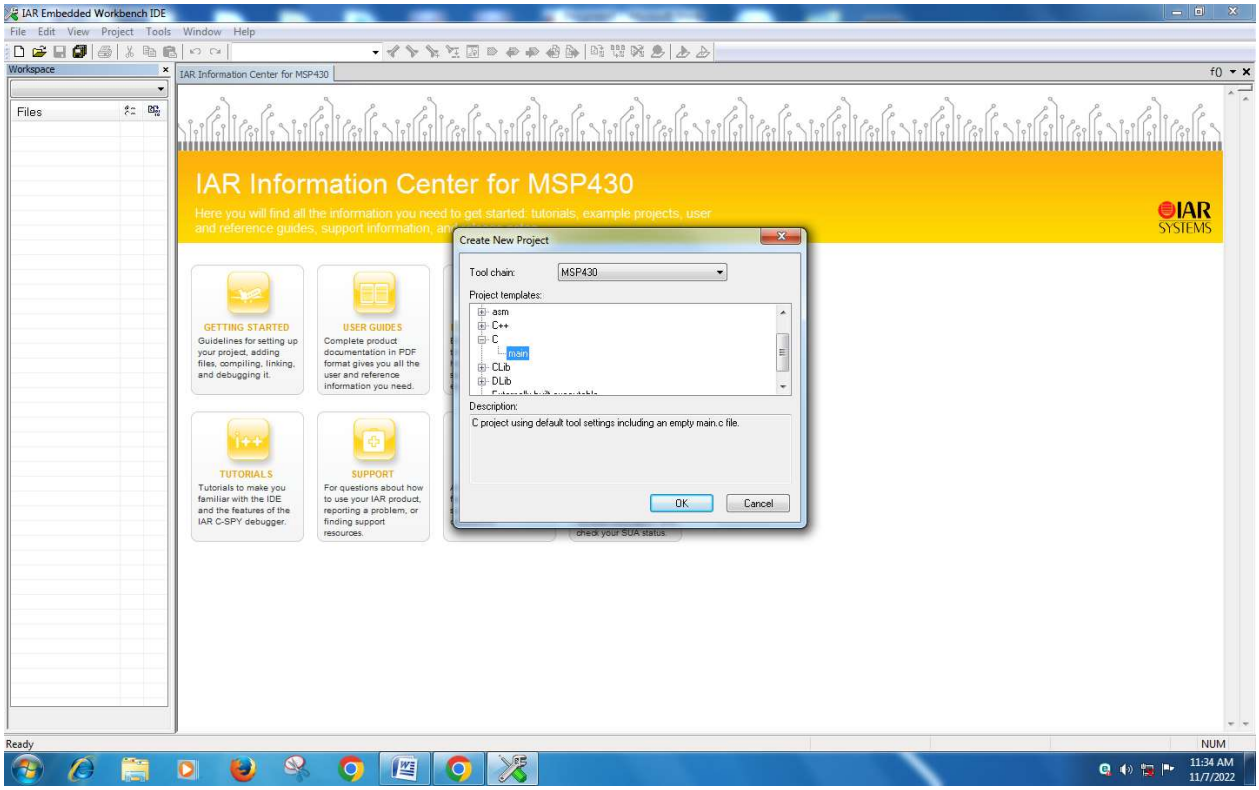
PROCEDURE:

Step1: Open IAR Embedded Workbench

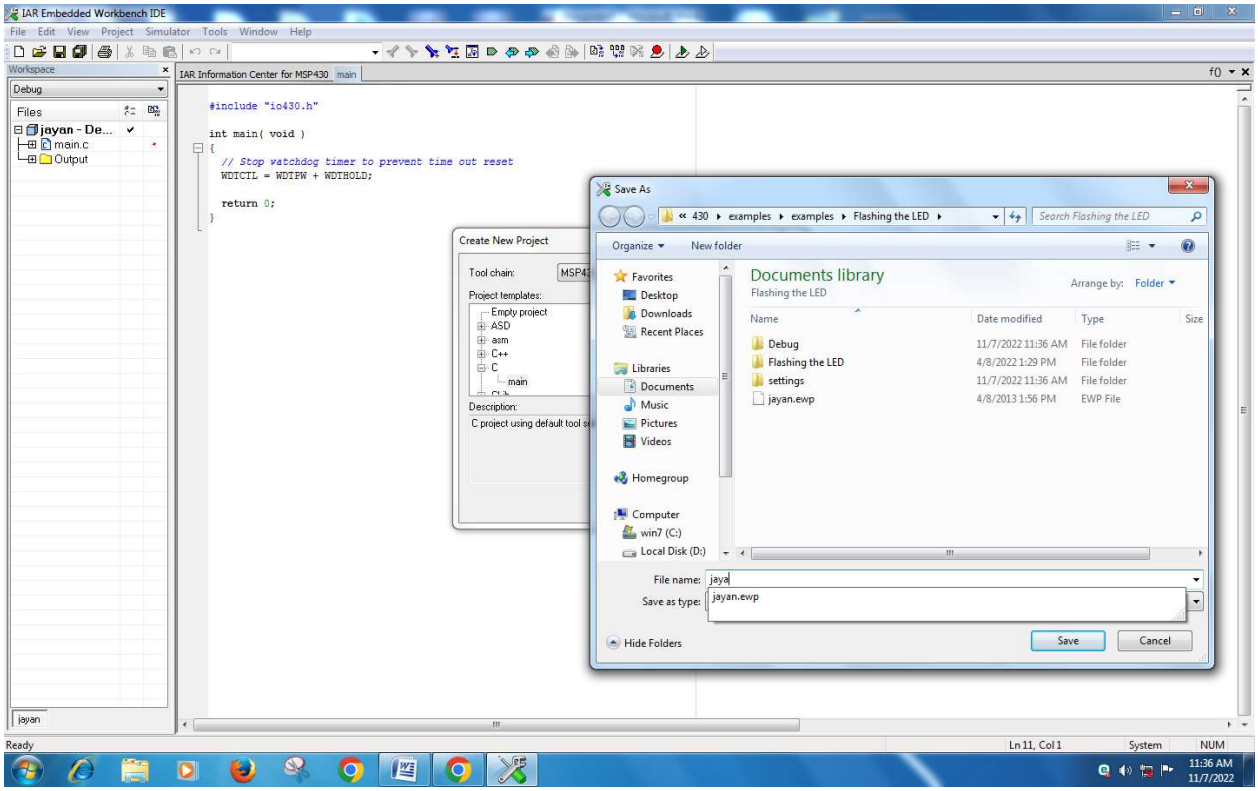
Step-2: Create New Project under Project Tab.



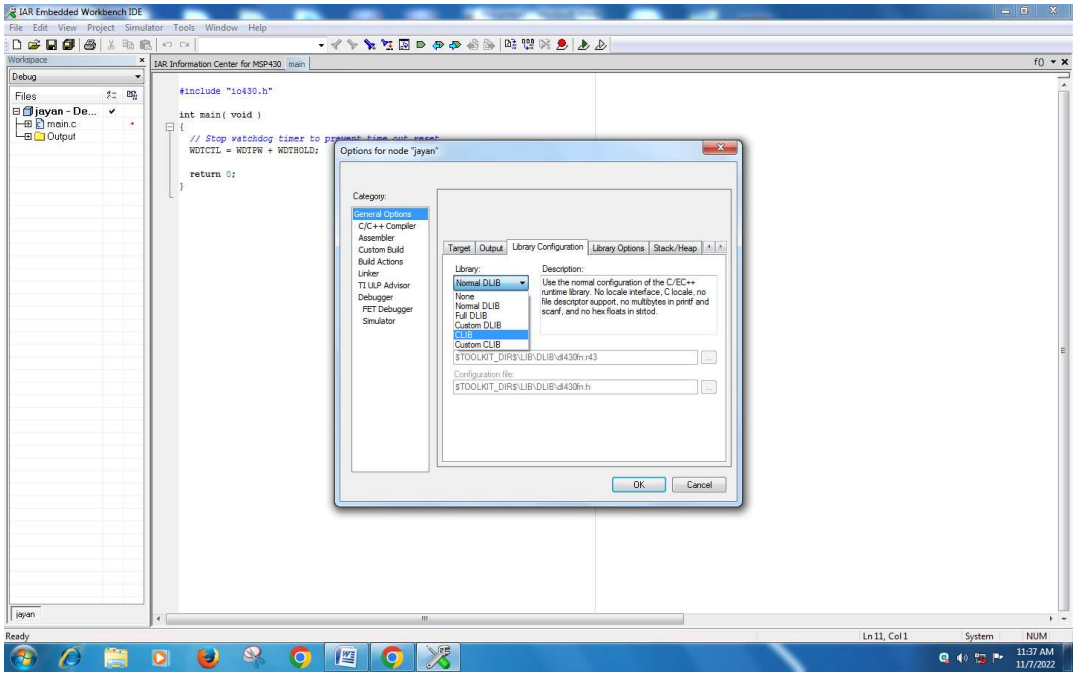
Step-3: When creating New Project, Select mainoption under C

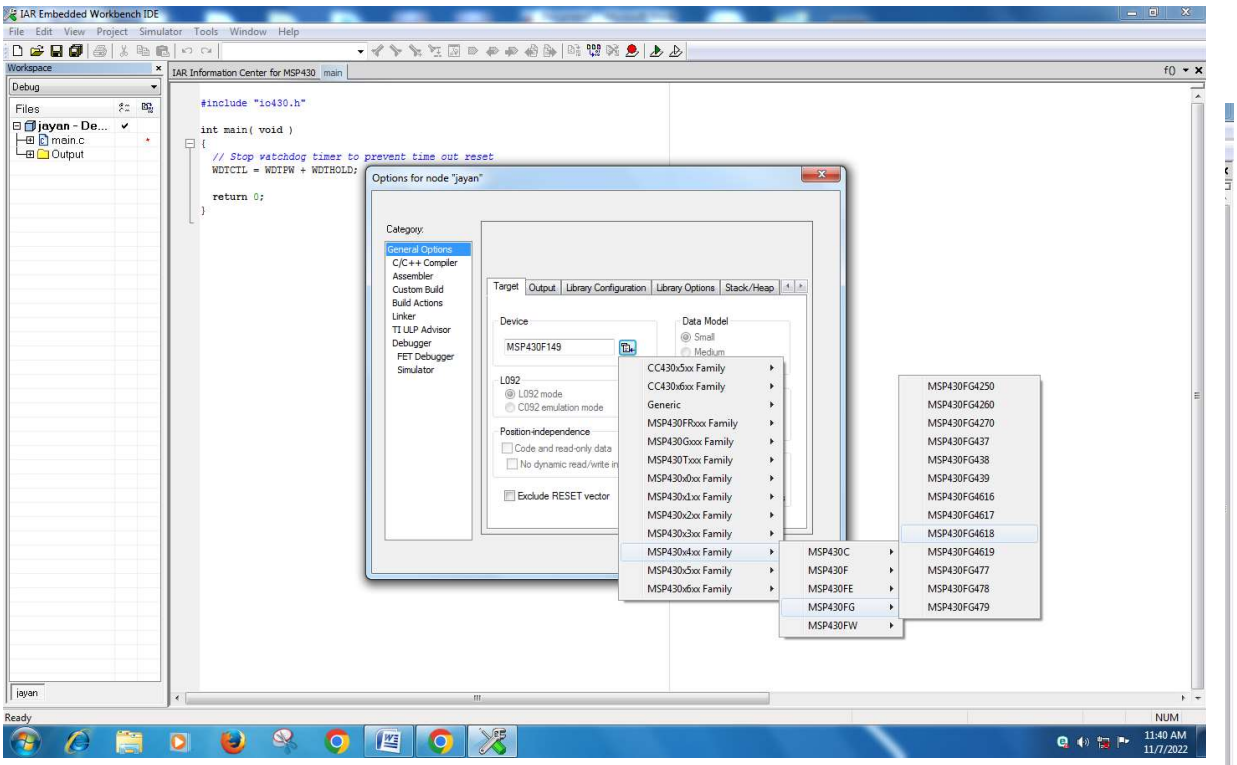


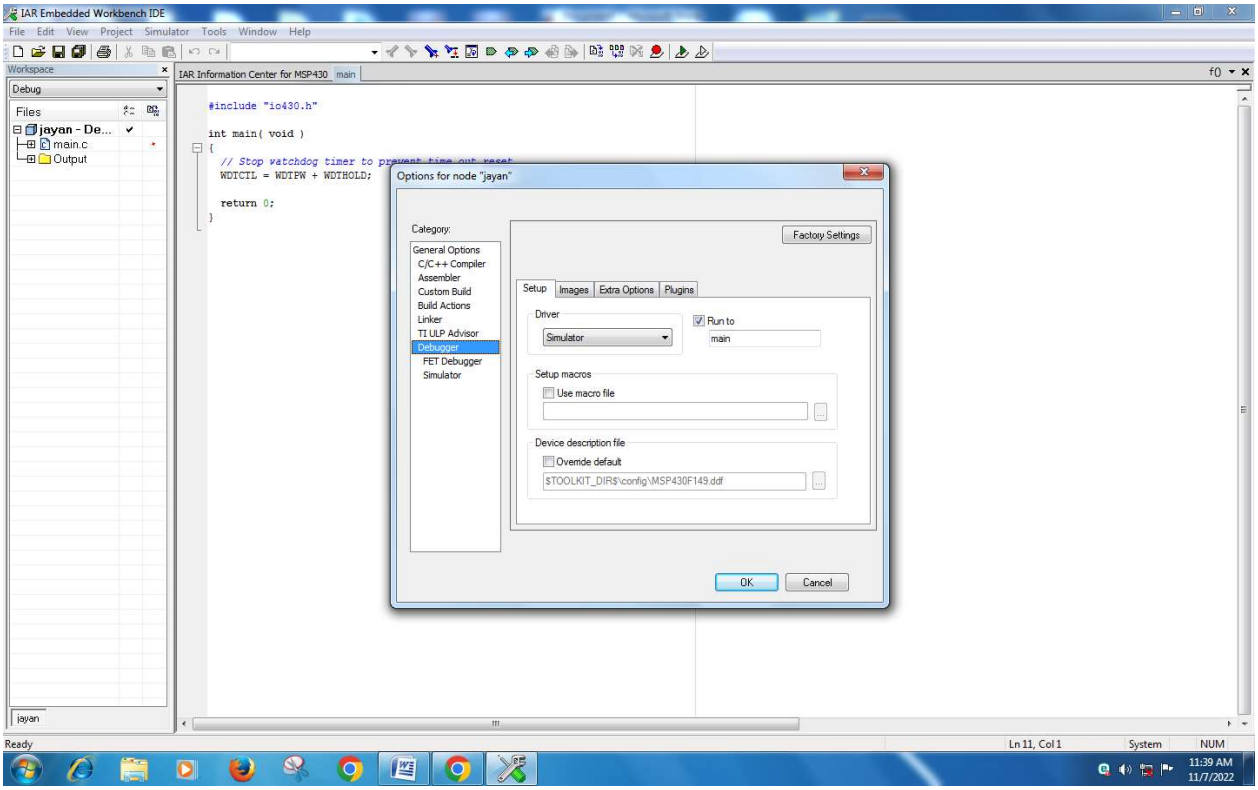
Step-4: Give a name and create a Project



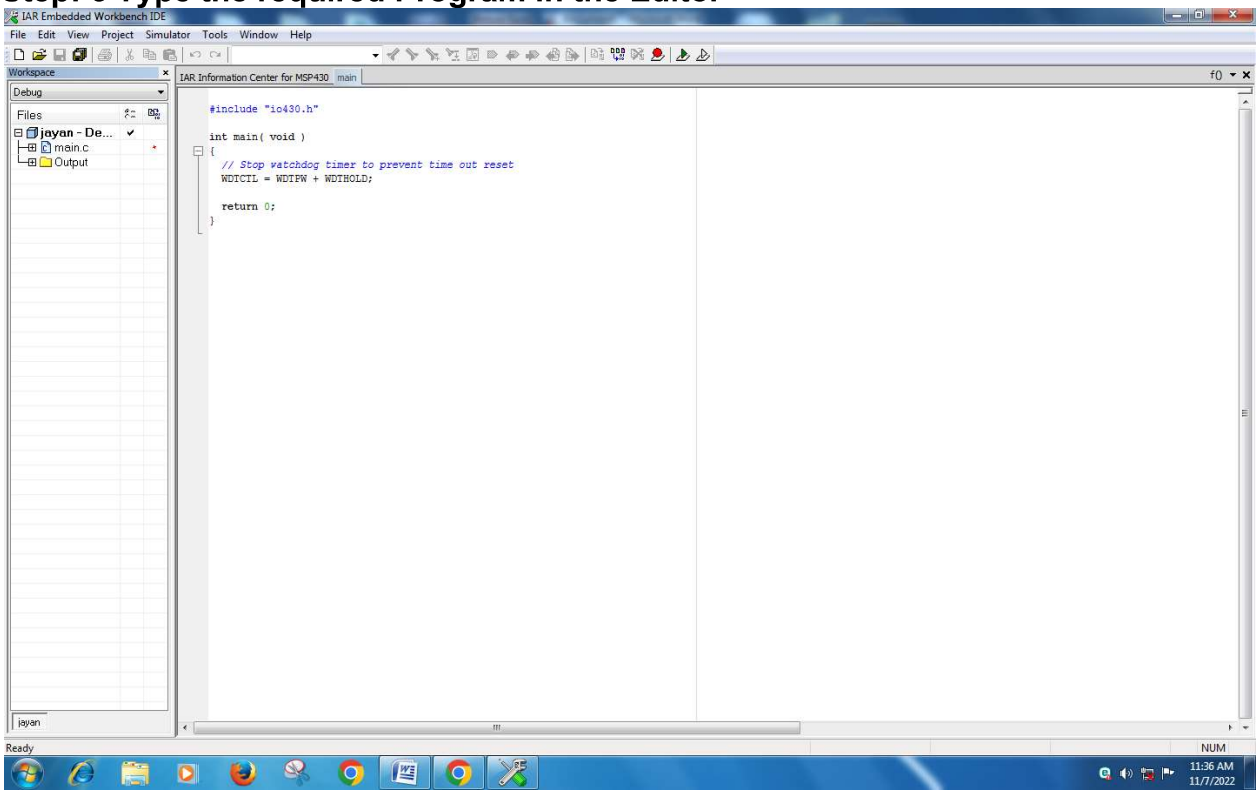
Step-5: Under Project Tab, click option, give the corresponding options for general options (under library configuration give CLIB) (under target give, 4x family->FG->4168), C++ compiler (optimization give none), and debugger (check for simulation) option



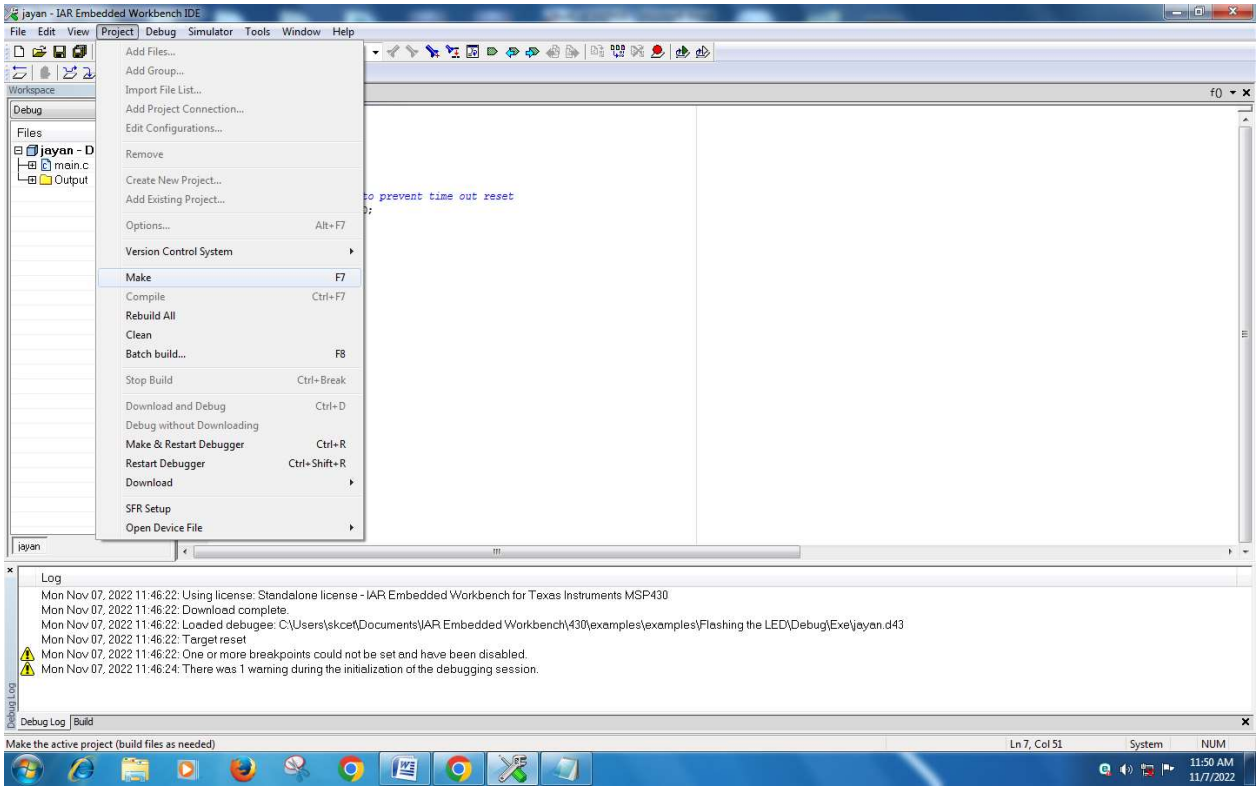

IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
Debug
Files
jayan - De...
main.c
Output
IAR Information Center for MSP430
main
#include "io430.h"
int main(void)
{
// Stop watchdog timer to prevent time out reset
WDTCTL = WDTPW + WDTHOLD;
return 0;
}
Options for node "jayan"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
TI ULP Advisor
Debugger
FET Debugger
Simulator
Target
Output
Library Configuration
Library Options
Stack/Heap
Device
MSP430F149
Data Model
Small
Medium
L092
L092 mode
C092 emulation mode
Position-independence
Code and read-only data
Exclude RESET vector
CC430x5xx Family
CC430x6xx Family
Generic
MSP430FRxxx Family
MSP430Gxxx Family
MSP430Txxx Family
MSP430x0xx Family
MSP430x1xx Family
MSP430x2xx Family
MSP430x3xx Family
MSP430x4xx Family
MSP430x5xx Family
MSP430x6xx Family
MSP430C
MSP430F
MSP430FE
MSP430FG
MSP430FW
MSP430FG4250
MSP430FG4260
MSP430FG4270
MSP430FG437
MSP430FG438
MSP430FG439
MSP430FG4616
MSP430FG4617
MSP430FG4618
MSP430FG4619
MSP430FG477
MSP430FG478
MSP430FG479
jayan
Ready
NUM
11:40 AM
11/7/2022


IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
Debug
Files
jayan - De...
main.c
Output
IAR Information Center for MSP430
main
#include "io430.h"
int main(void)
{
// Stop watchdog timer to prevent time out reset
WDTCTL = WDTPW + WDTHOLD;
return 0;
}
Options for node "jayan"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
TI ULP Advisor
Debugger
FET Debugger
Simulator
Factory Settings
Setup
Images
Extra Options
Plugins
Driver
Simulator
Run to
main
Setup macros
Use macro file
Device description file
Override default
$TOOLKIT_DIR$\config\MSP430F149.ddf
OK
Cancel
jayan
Ready
Ln 11, Col 1
System
NUM
11:39 AM
11/7/2022

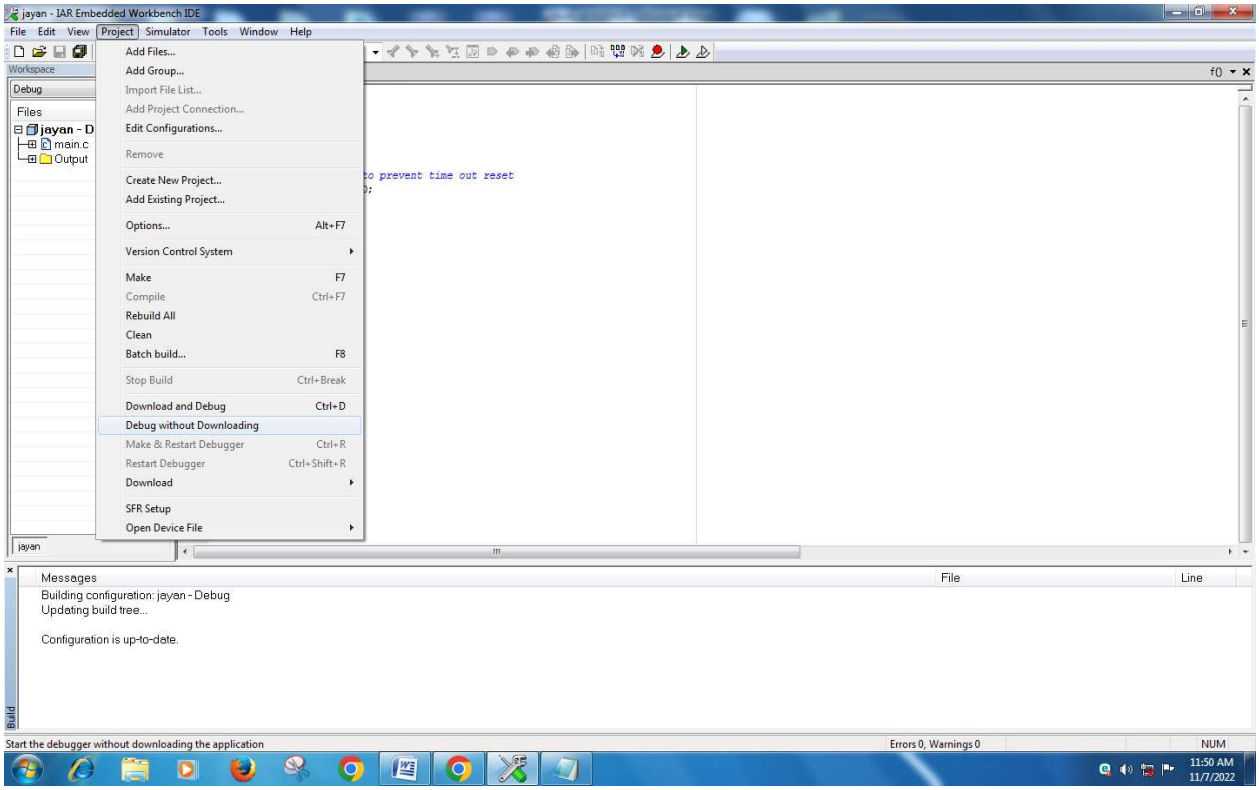
Step: 6 Type the required Program in the Editor

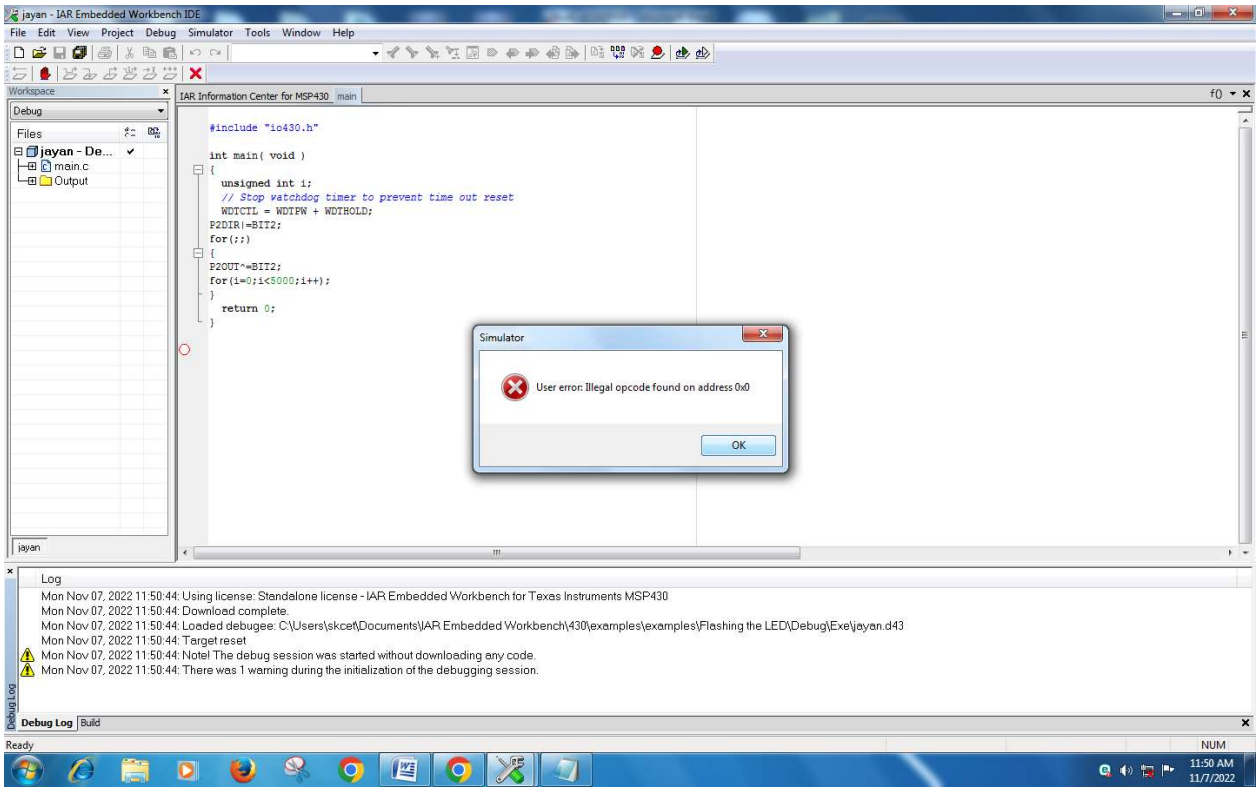


Step-7 : After Typing program click Project Make

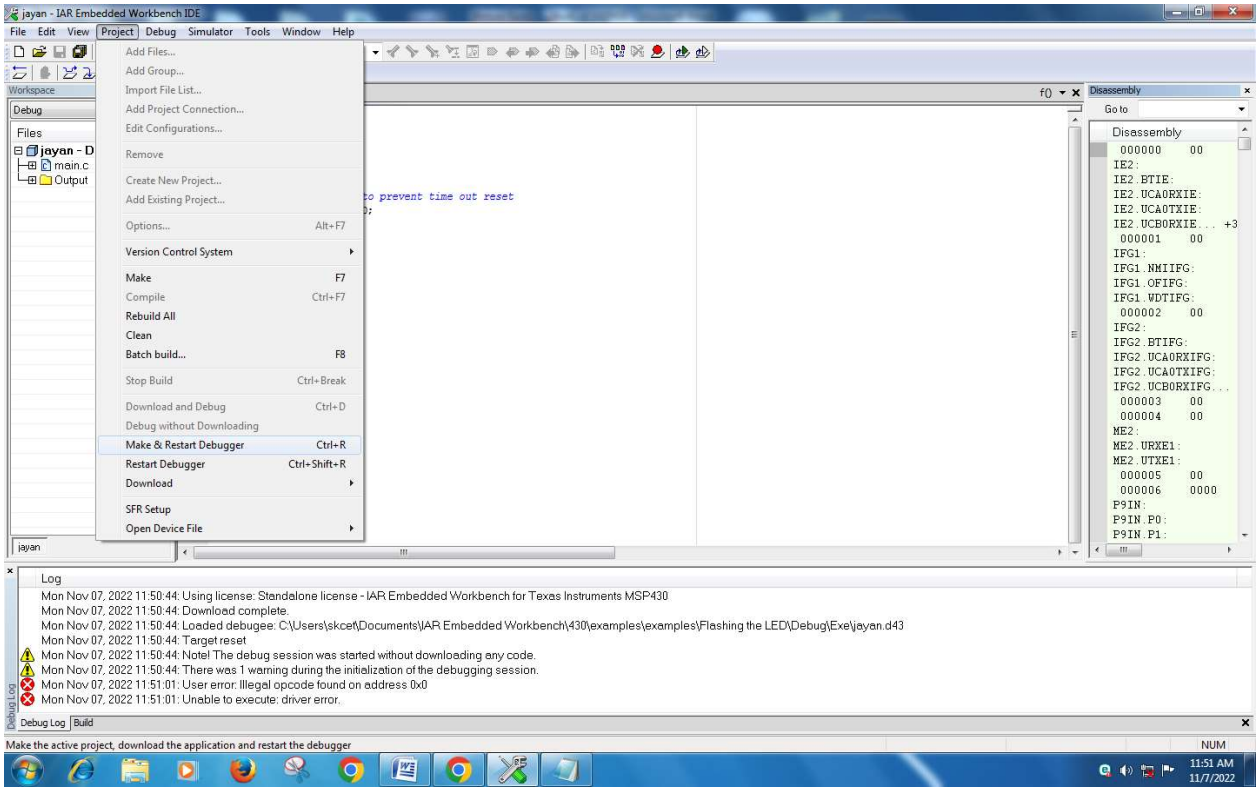


Step-8: Click Project->Debug without downloading

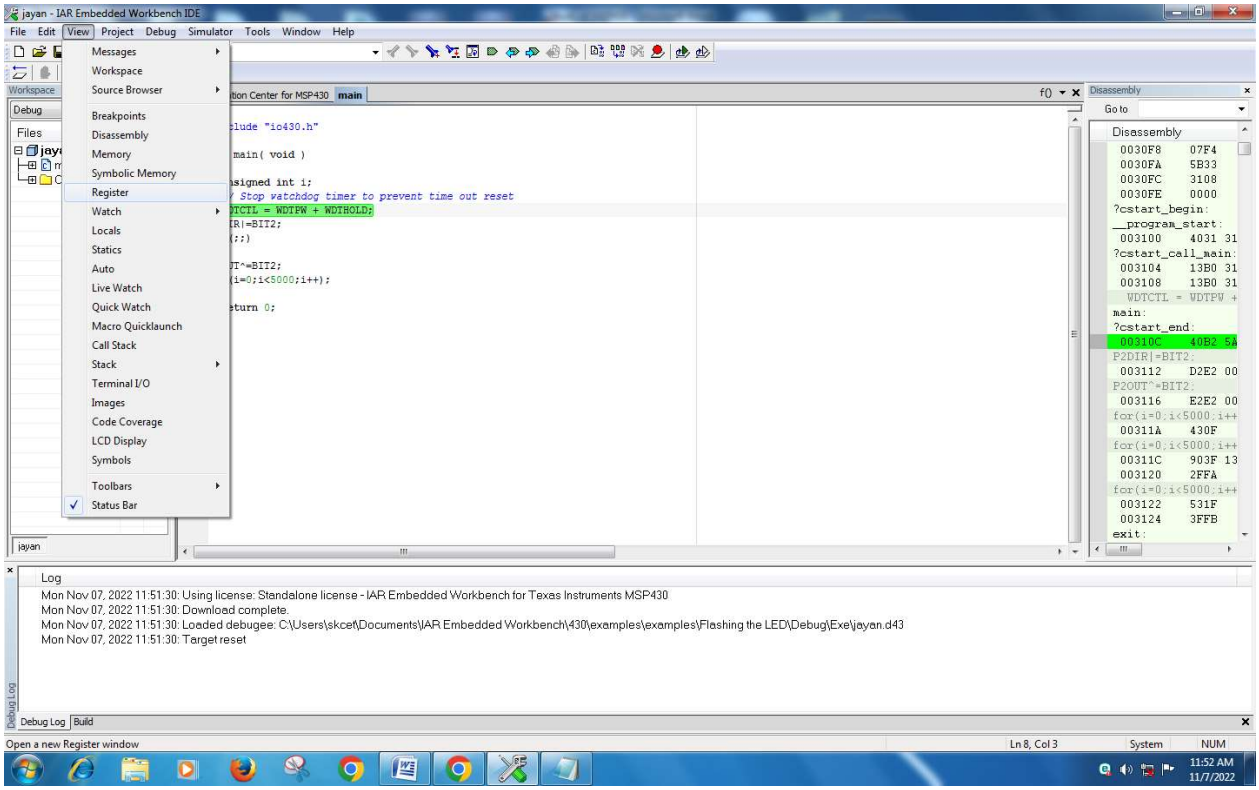




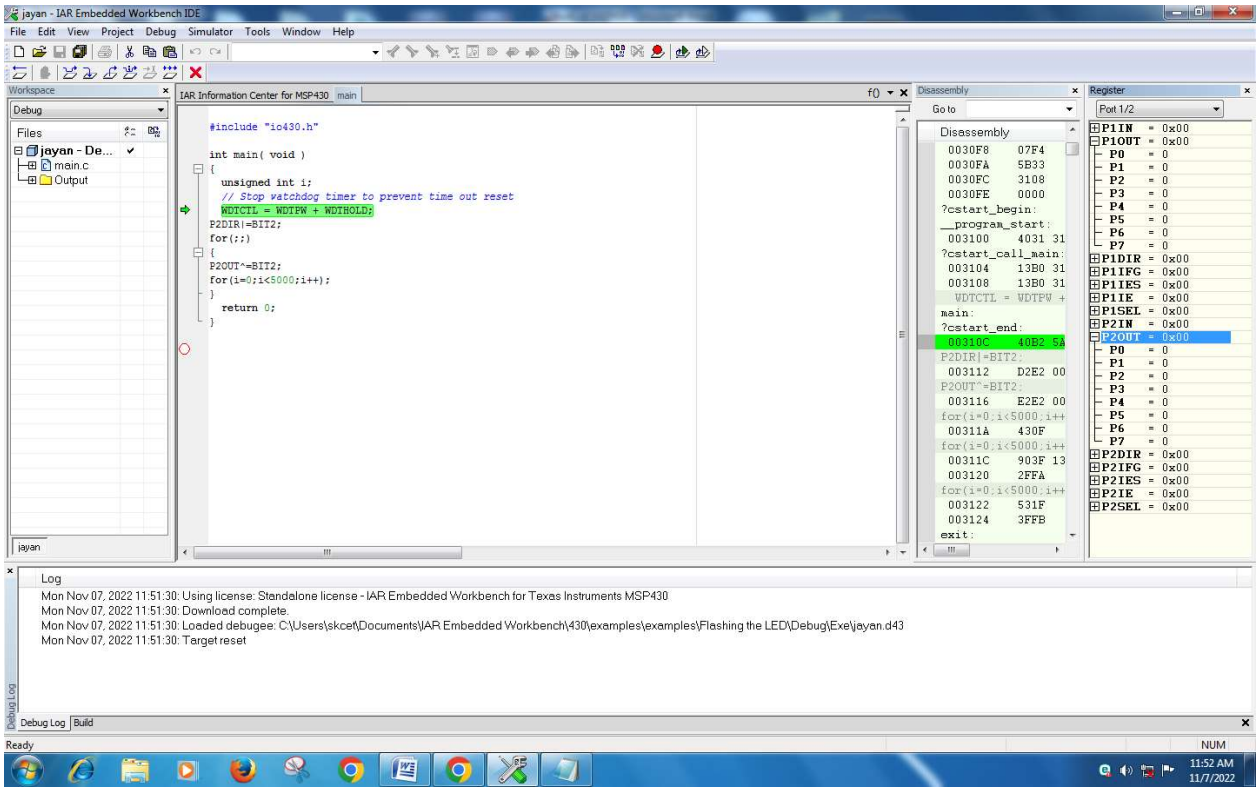
Step-9: Click Project-> Make and start Debugger/ Project-> Debugger



Step- 10: Click View-> Register



Step-11: Click Port 1/2



Result:

Thus IAR Embedded Workbench is studied.

Exp: 8
Date:

LED BLINKING USING MSP430 PROGRAMMING

AIM:

To perform LED blinking using MSP430 Programming

ALGORITHM:

Step 1: Start the program
Step 2: Enable Input and Output Ports
Step 3: Initialize watch dog timer
Step 4: Select the port 2 for display of the led blinking
Step 5: Set the port 2 output
Step 6: Use infinite loops for the continuous blinking of LEDs
Step 7: Stop the program

PROGRAM:

```
#include <msp430x14x.h>
int main(void)
{
        unsigned int i;
        WDTCTL = WDTPW + WDTHOLD;
        P2DIR|=BIT2;
        for( ; ;)
        {
                P2OUT^=BIT2;
              for(i=0;i<5000;i++);

        }
}
```

RESULT:

Thus the LED blinking was performed through MSP430 using IAR work bench.

Exp: 9
Date:

SEVEN SEGMENT DISPLAY USING MSP430 PROGRAMMING

AIM:

To perform Seven Segment display using MSP430 Programming

ALGORITHM:

Step 1: Start the program
Step 2: Enable Input and Output Ports
Step 3: Initialize watch dog timer
Step 4: Select the port 1 for display of the bits for enabling 7 segment
Step 5: Set the port 1 output
Step 6: Use infinite loops for the continuous working of the seven segment.
Step 7: Stop the program

PROGRAM:

```
#include <msp430g2553.h>

#define a BIT1
#define b BIT2
#define c BIT3
#define d BIT4
#define e BIT5
#define f BIT6
#define g BIT7

void main()
{
   WDTCTL = WDTPW + WDTHOLD;
   P1DIR = a+b+c+d+e+f+g;

   while(1)
   {
      P1OUT = a+b+c+d+e+f;
      __delay_cycles(500000);

      P1OUT = b+c;
      __delay_cycles(500000);

      P1OUT = a+b+g+e+d;
      __delay_cycles(500000);

      P1OUT = a+b+g+c+d;
      __delay_cycles(500000);
```

```
        P1OUT = f+g+b+c;
        __delay_cycles(500000);

        P1OUT = a+f+g+c+d;
        __delay_cycles(500000);

        P1OUT = a+f+e+d+c+g;
        __delay_cycles(500000);

        P1OUT = a+b+c;
        __delay_cycles(500000);

        P1OUT = a+b+c+d+e+f+g;
        __delay_cycles(500000);

        P1OUT = a+b+c+d+f+g;
        __delay_cycles(500000);
    }
}
```

Result:

Thus the Seven Segment Display was performed through MSP430 using IAR work bench.

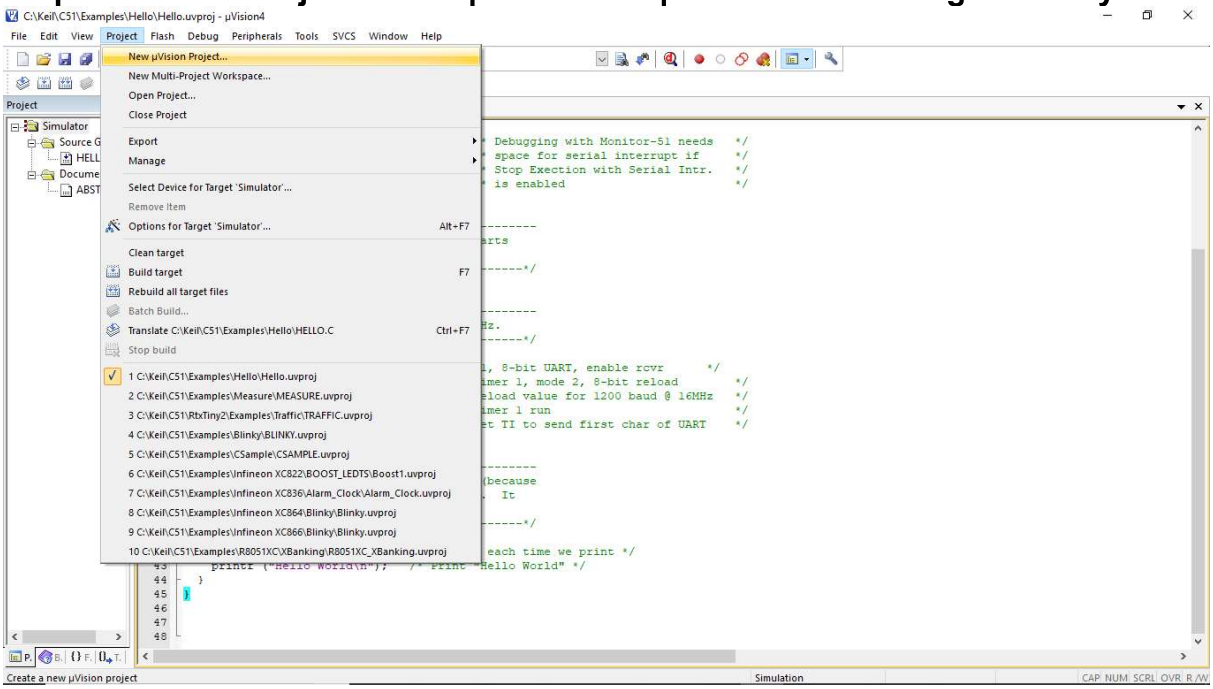
STUDY OF KEIL

AIM:

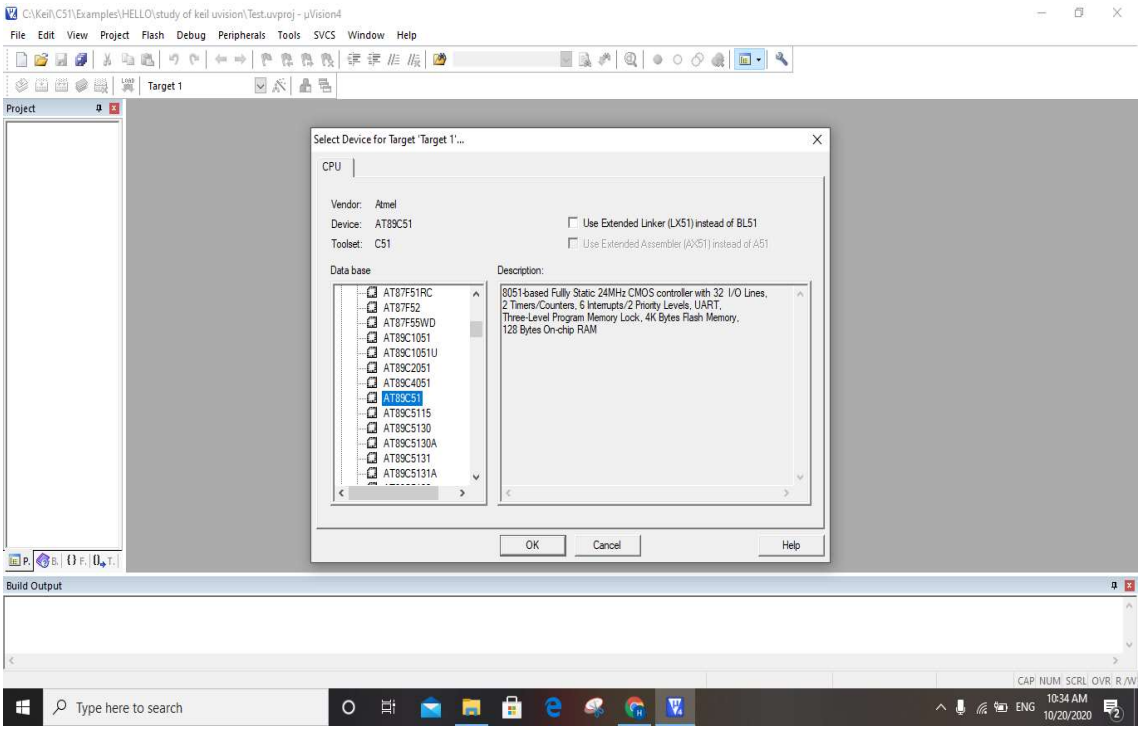
To study Keil Software

PROCEDURE:

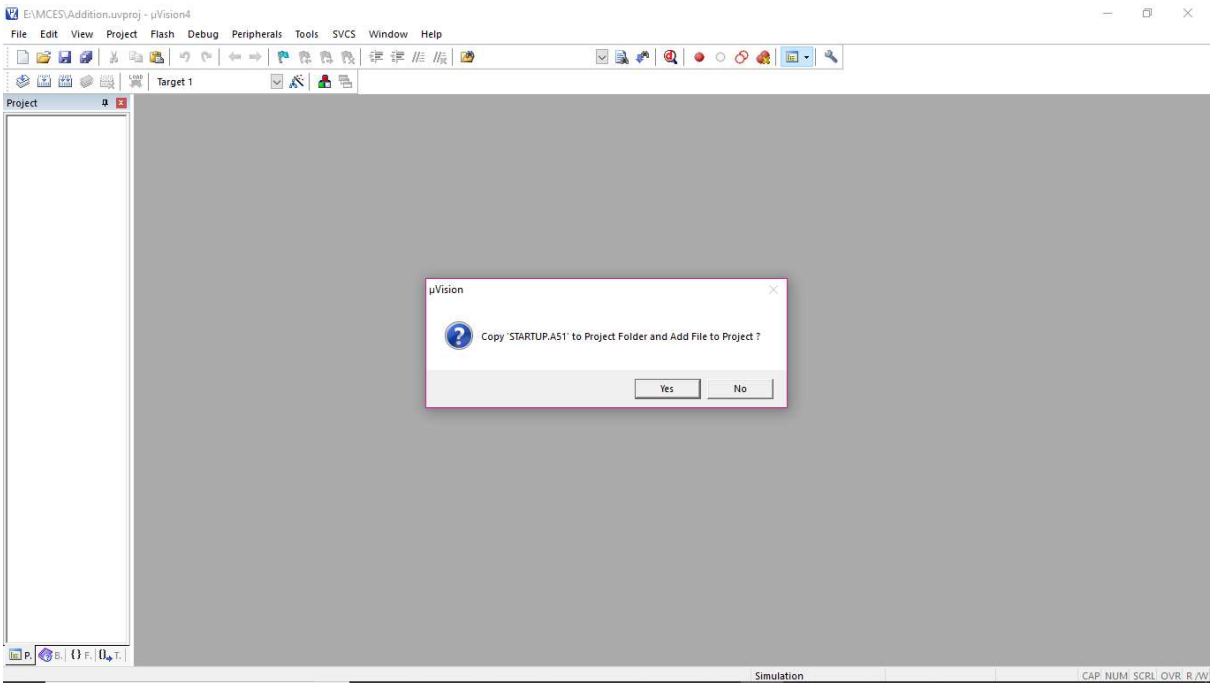
Step1: Create a Project in Keil µvision Compiler Under Working Directory



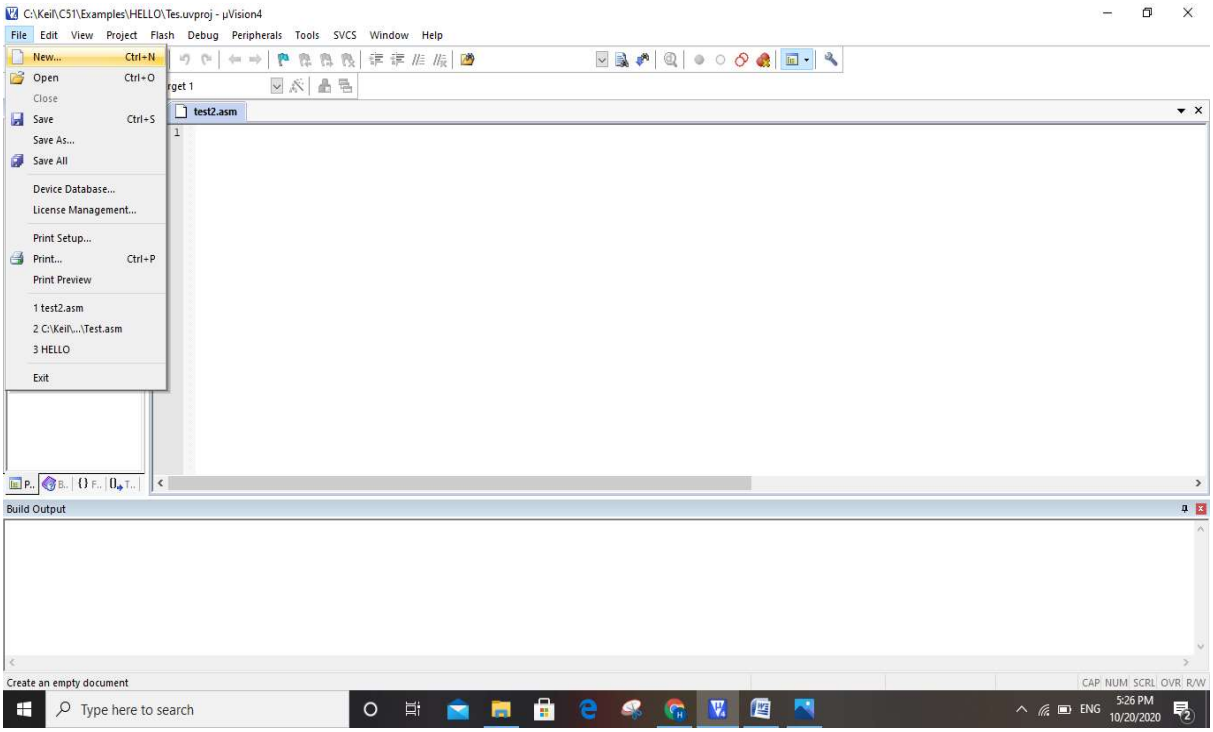
Step-2: Select Device for Selected Target in Keil µvision Compiler (Atmel - AT89C51).

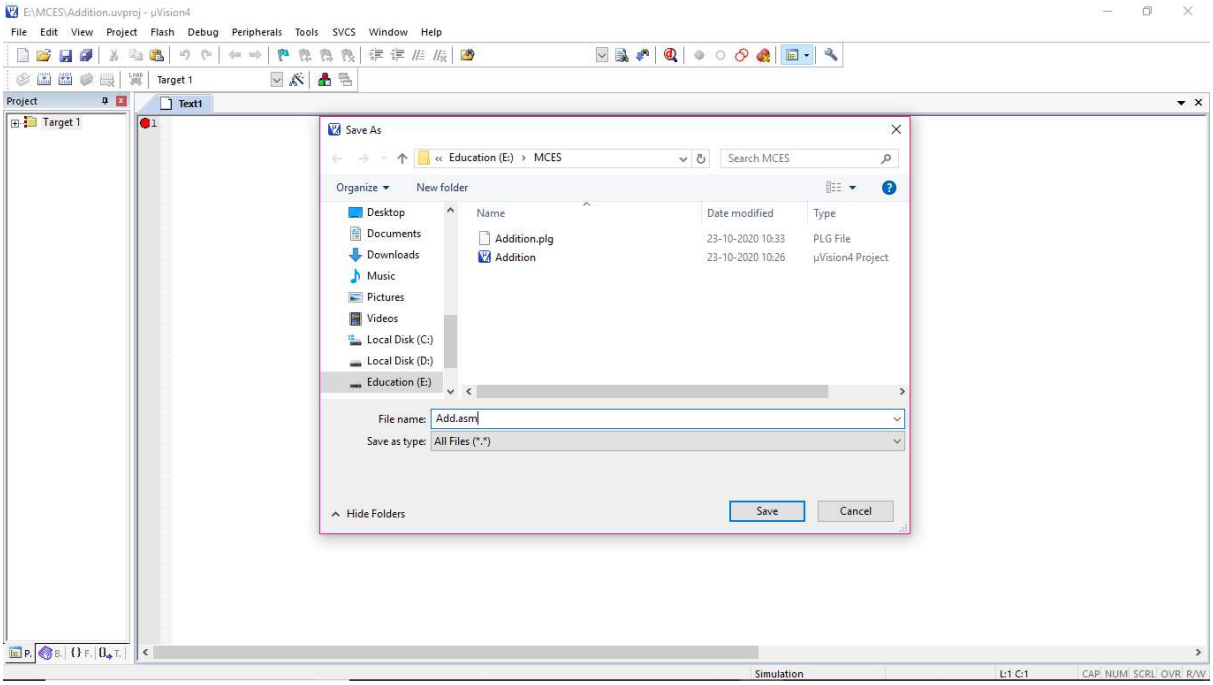


Step-3: Select No. for Adding ‘Startup.A51’ in Source Folder because it is Assembely Language.

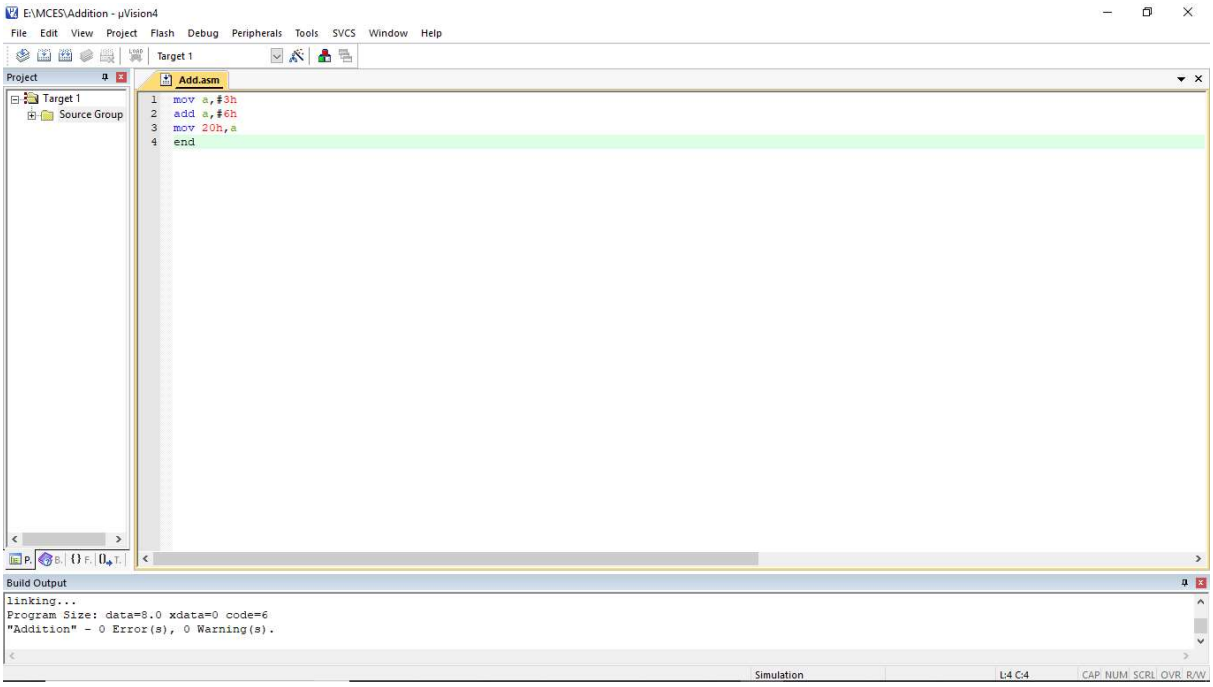


Step-4: Create a New File with (.asm) Extension in Keil µvision Compiler.

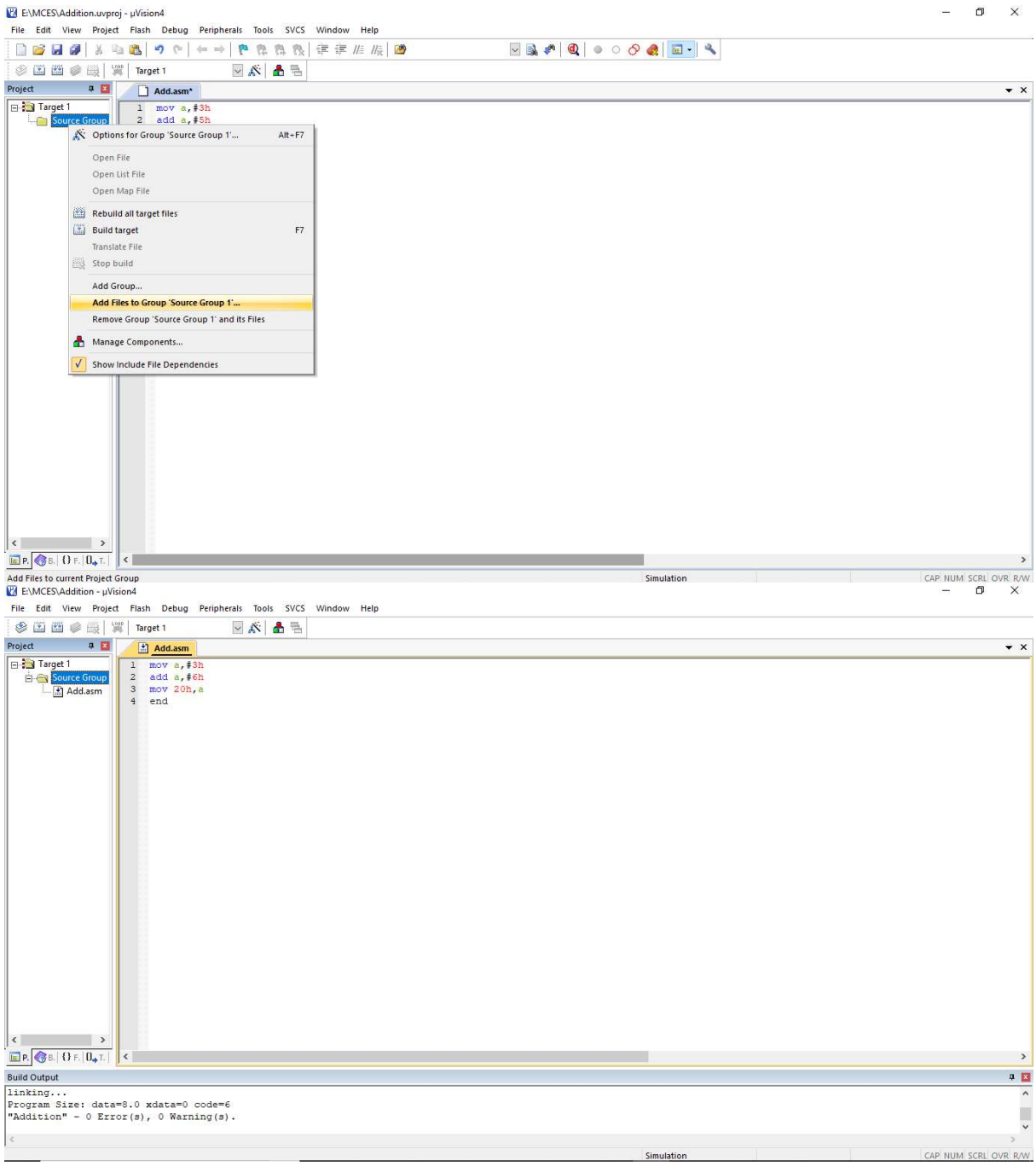




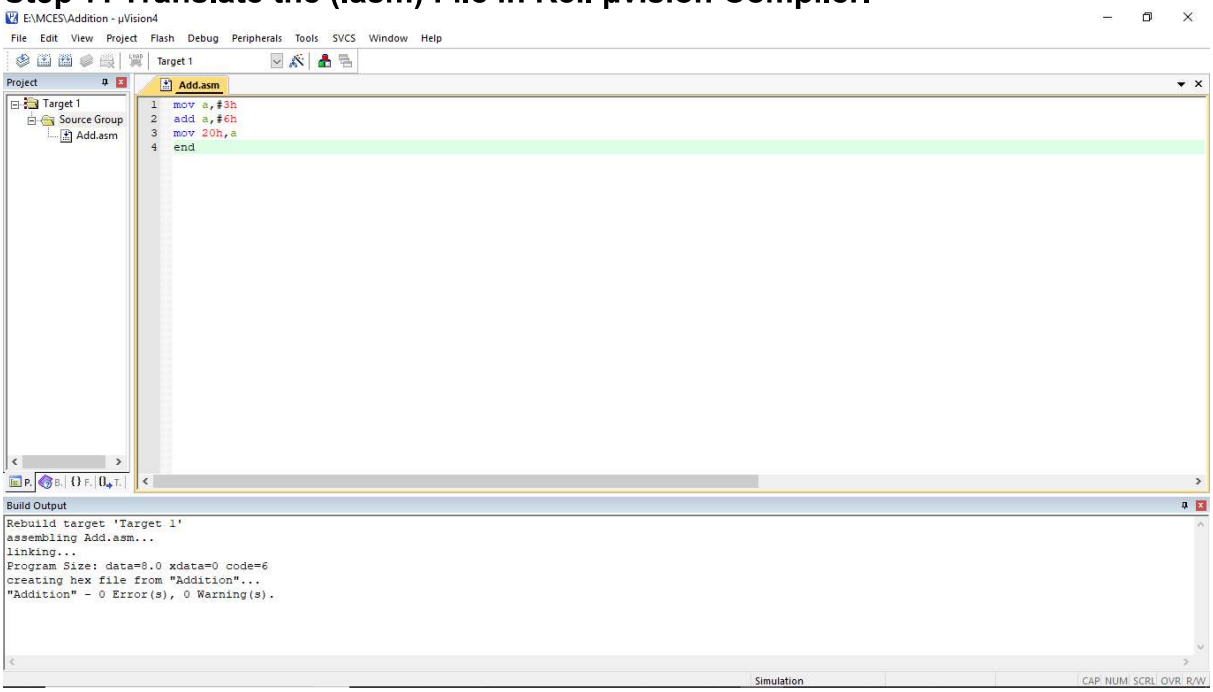
Step-5: Build The Source Code in (.asm) File and Save it in Keil µvision Compiler.



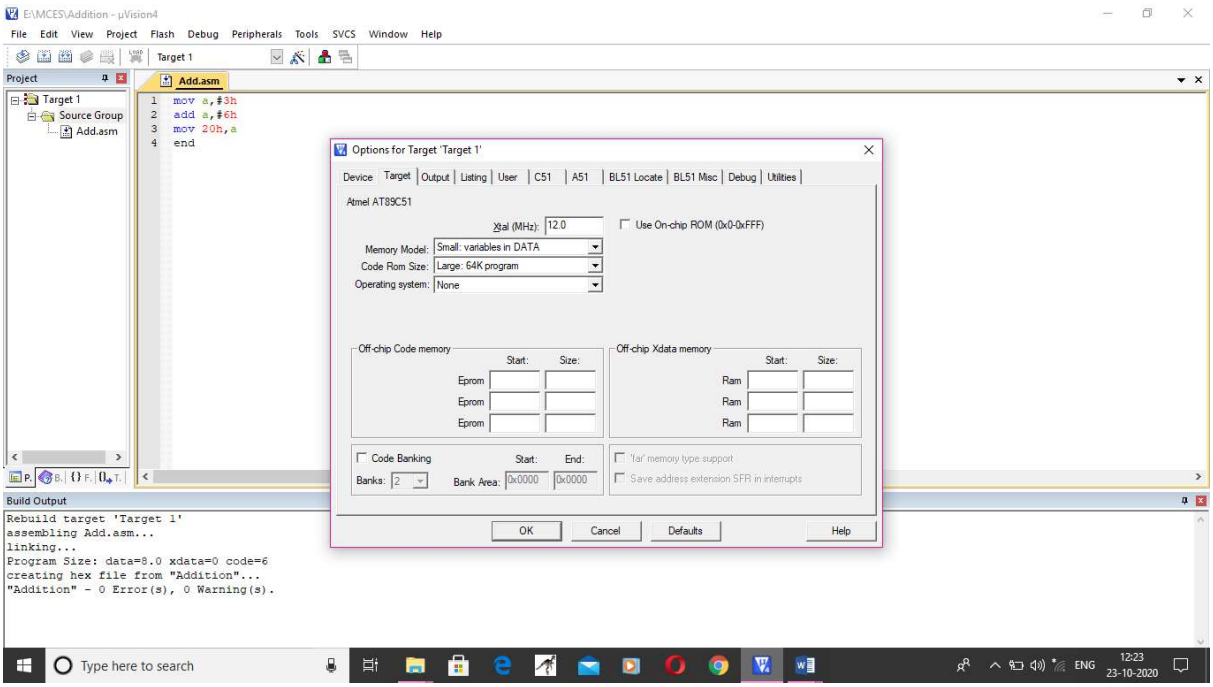
Step-6: Add (.asm) File to Source Group Under Project Directory in Keil µvision Compiler.

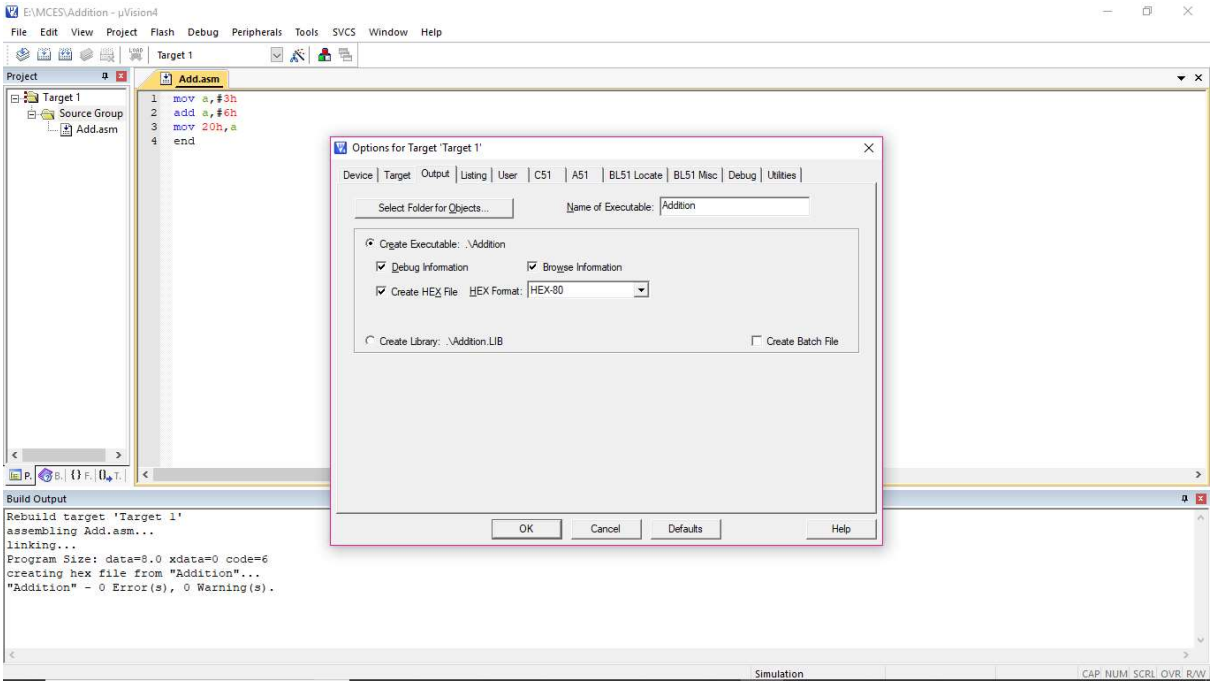


Step-7: Translate the (.asm) File in Keil µvision Compiler.

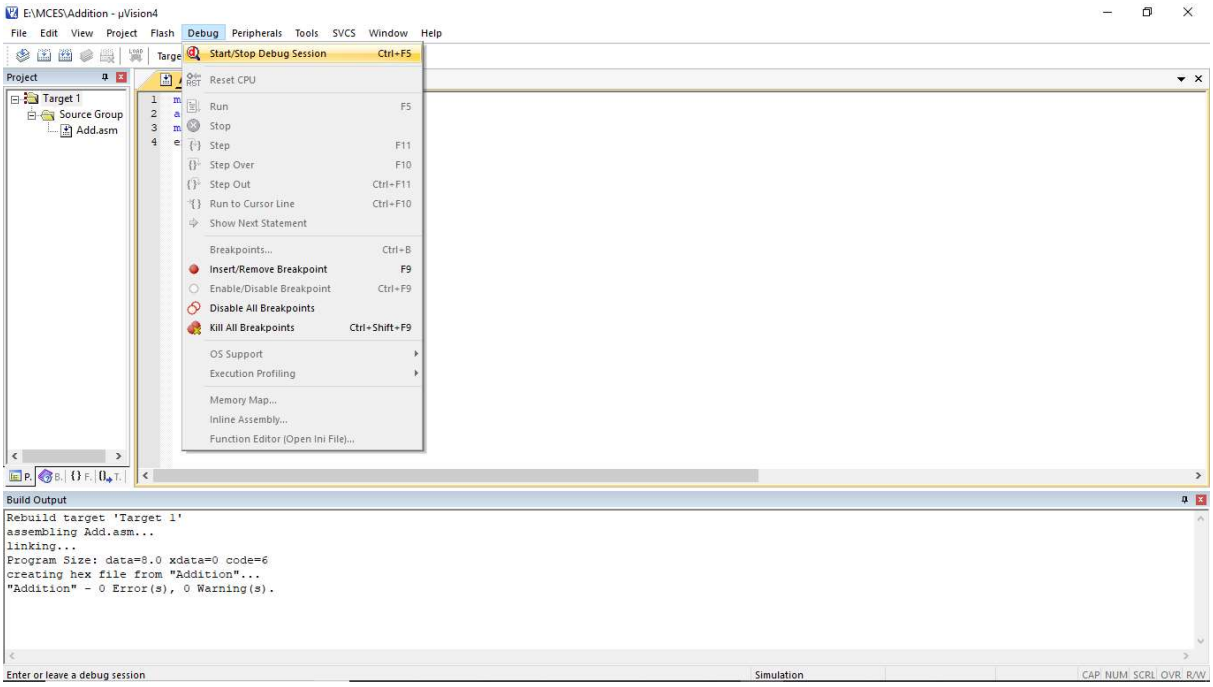


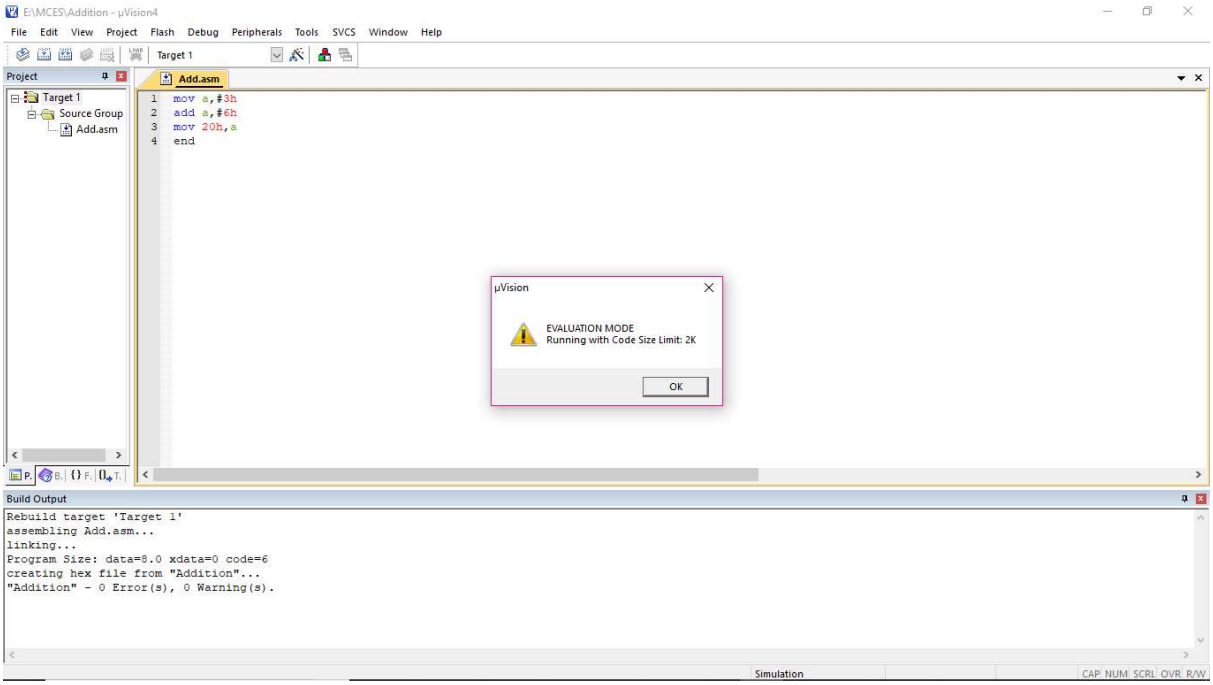
Step-8: Build the (.asm) File in Keil µvision Compiler.





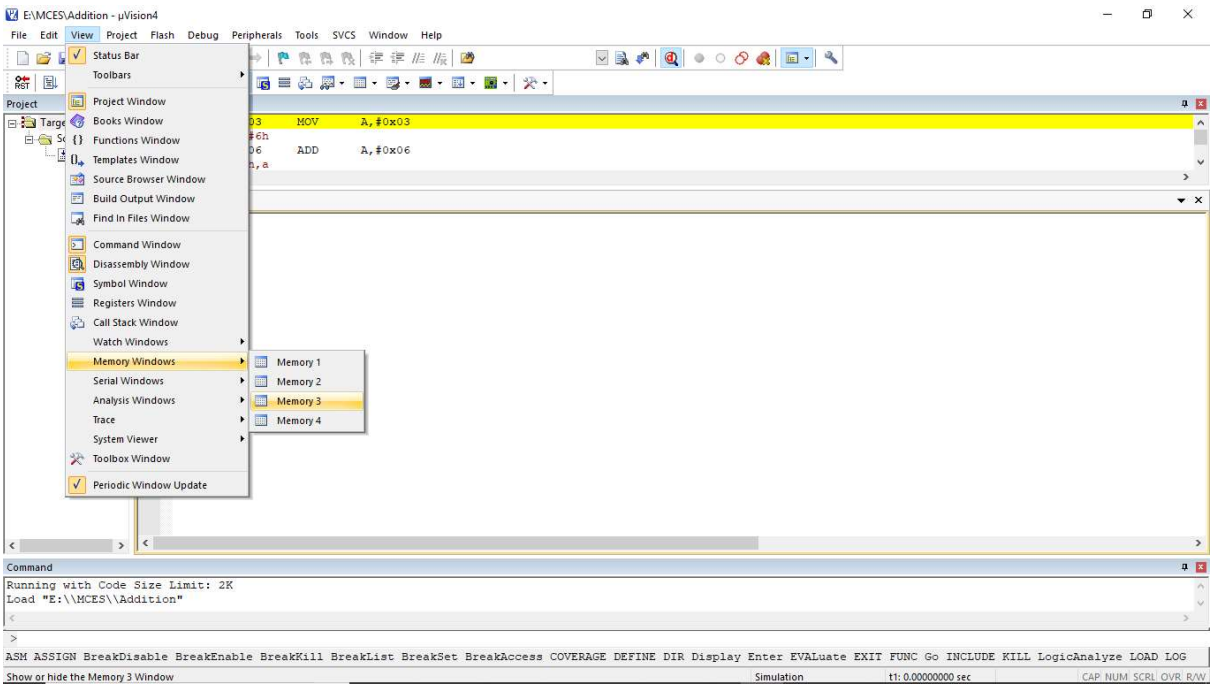
Step-9: Rebuild the (.asm) File in Keil µvision Compiler.

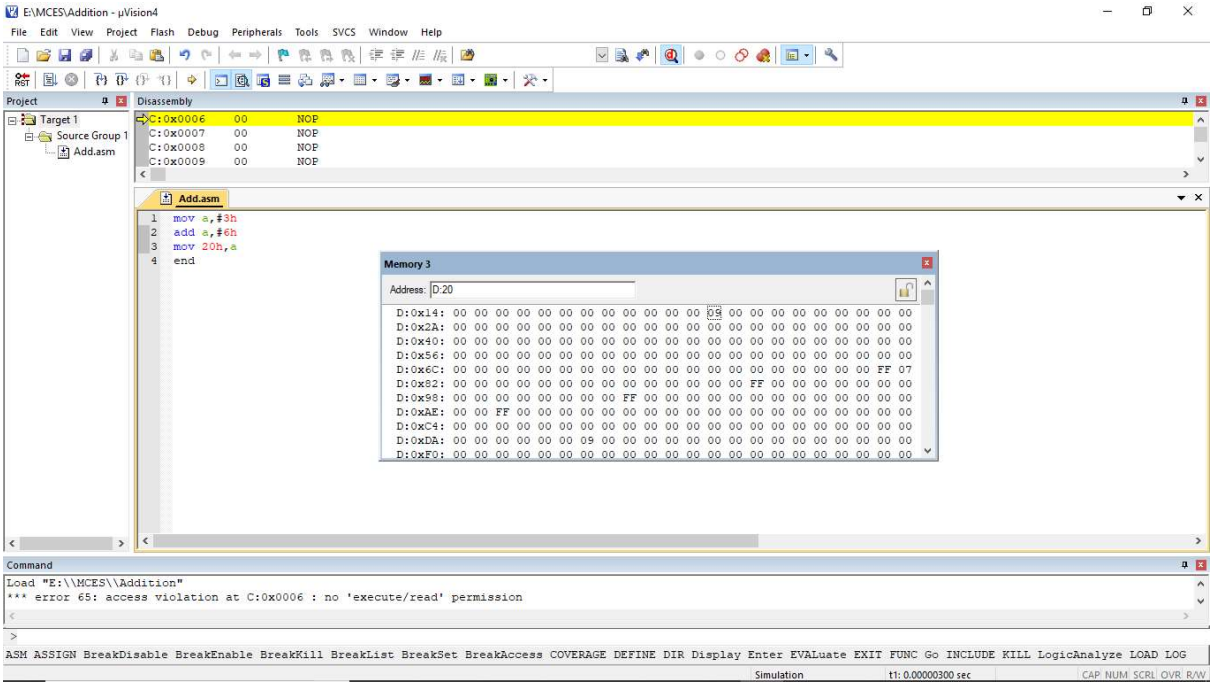




Output: Should Be 0 Errors and 0 Warnings in Output Build Window.

Step-10: To Change the Target Device, Right Click on Target in Project Window in Keil µvision Compiler.





RESULT:

Thus Keil Software is studied

Exp: 10
Date:

LED BLINKING USING ARM

AIM:

To write an embedded C program to simulate LED Blinking and to verify the output in the keil compiler.

Hardware & Software Required:

- ARM Development tools
- LED
- Keil C software

THEORY

NXP's ARM7 (LPC2148), ARM Primer Kit is proposed to smooth the progress of developing and debugging of various designs encompassing of High speed 32-bit Microcontrollers.

LED (LIGHT EMITTING DIODES)

Light Emitting Diodes (LED) is the most commonly used components, usually for displaying pins digital states. Typical uses of LEDs include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

INTERFACING LED

Fig. shows how to interface the LED to microcontroller. As you can see the Anode is connected through a resistor to GND & the Cathode is connected to the Microcontroller pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the LED is turned ON.

ALGORITHM

Step 1: Start the program
Step 2: Enable Input and Output Ports
Step 3: Initialize timer
Step 4: Place the data for the LED in the data bus
Step 5: Call delay
Step 6: Rotate the data for the LEDs to switch on next LED
Step 7: Use infinite loops for the continuous blinking of LEDs
Step 8: Stop the program

PROGRAM:

```
#include <LPC21xx.H>                    /* LPC21xx definitions */

void wait (void)  {                     /* wait function */
int  d;

for (d = 0; d < 1000000; d++);          /* only to delay for LED flashes */
}

int main (void) {
unsignedinti;                           /* LED var */

  IODIR1 = 0xFFFFFFFF;                  /* P1.16..23 defined as Outputs */

while (1)  {                            /* Loop forever */
     IOSET1 = 0XFFFFFFFF;               /* Turn on LED */
wait ();                                /* call wait function */
     IOCLR1 = 0xFFFFFFFF;               /* Turn off LED */
   }}
```

PROGRAM:

```
#include <LPC21xx.H>                    /* LPC21xx definitions */

void wait (void)  {                     /* wait function */
int  d;

for (d = 0; d < 1000000; d++);          /* only to delay for LED flashes */
}

int main (void) {
unsignedinti;                           /* LED var */

  IODIR1 = 0x00FF0000;                  /* P1.16..23 defined as Outputs */

while (1)  {                            /* Loop forever */
for (i = 1<<16; i< 1<<23; i<<= 1) {  /* Blink LED 0,1,2,3,4,5,6 */
     IOSET1 = i;                        /* Turn on LED */
wait ();                                /* call wait function */
     IOCLR1 = i;                        /* Turn off LED */
   }
for (i = 1<<23; i> 1<<16; i>>=1 ) {  /* Blink LED 7,6,5,4,3,2,1 */
     IOSET1 = i;                        /* Turn on LED */
wait ();                                /* call wait function */
     IOCLR1 = i;                        /* Turn off LED */
   }}}
```

Result:

Thus the simulation of LED blinking has been done and the output has been verified using the keil compiler.

Exp: 11
Date:

SEVEN SEGMENT DISPLAY INTERFACING USING ARM

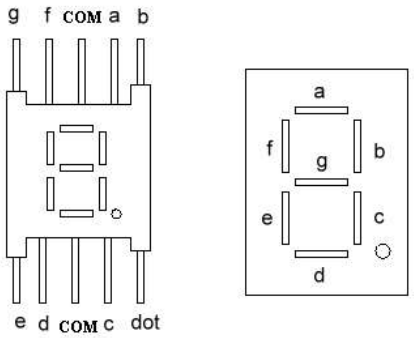
AIM:

To write an embedded C program for interfacing Seven Segment Display and to verify the output in the ARM kit.

HARDWARE & SOFTWARE REQUIRED:

- Keil uVision4 compiler
- ARM Development Kit
- Seven Segment Display

Segment Display is an array of 8 LEDs connected or arranged in a special pattern together to form or display the digits (0-9) and Dot functions. The 7-segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle, and bottom. Additionally, the seventh segment bisects the rectangle horizontally. The 7 LEDs assigning specific alphabet to each and 1 LED acts as a dot in the display. Every LED is assigned a name from 'a' to 'h' and is identified by its name. Seven LEDs 'a' to 'g' are used to display the numerals while eighth LED 'h' is used to display the dot/decimal. =A 7-segment is generally available in ten pin package. While eight pins correspond to the eight LEDs, the remaining two pins (at middle) are common and internally shorted. These segments come in two configurations, namely, Common cathode (CC) and Common anode (CA). In CC configuration, the negative terminals of all LEDs are connected to the common pins. The common is connected to ground and a particular LED glows when its corresponding pin is given high. In CA arrangement, the common pin is given a high logic and the LED pins are given low to display a number. There are also fourteen-segment displays and sixteen-segment displays.



Seven-Segment Display

Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

Conversion of Hex number into seven segment equivalent

ALGORITHMS

Step1: Include a header file to assign bits for ports of the microcontroller

Step2: Use segment registers for initializing the display

Step3: Use loops for defining bits continuously

Step4: Use sub-routines for invoking delay – if necessary

PROGRAM

```
#include<LPC21XX.H>
unsigned int i,j,k;
unsigned int ar[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};

void delay();

int main()
{
PINSEL0=0x00000000;
IO0DIR=0x000000FF;
while(1)
{
for(i=0;i<10;i++)
{
IO0SET=ar[i];
delay();
IO0CLR=ar[i];
```

```
}
}
return 0;
}
void delay()
{
for(j=0;j<4000;j++)
        for(k=0;k<600;k++);
}
```

RESULT:

Thus the interfacing of Seven Segment Display has been and output has been verified using ARM Kit.

Exp: 12
Date:

LCD INTERFACING WITH ARM

AIM

To study the LCD interfacing with ARM

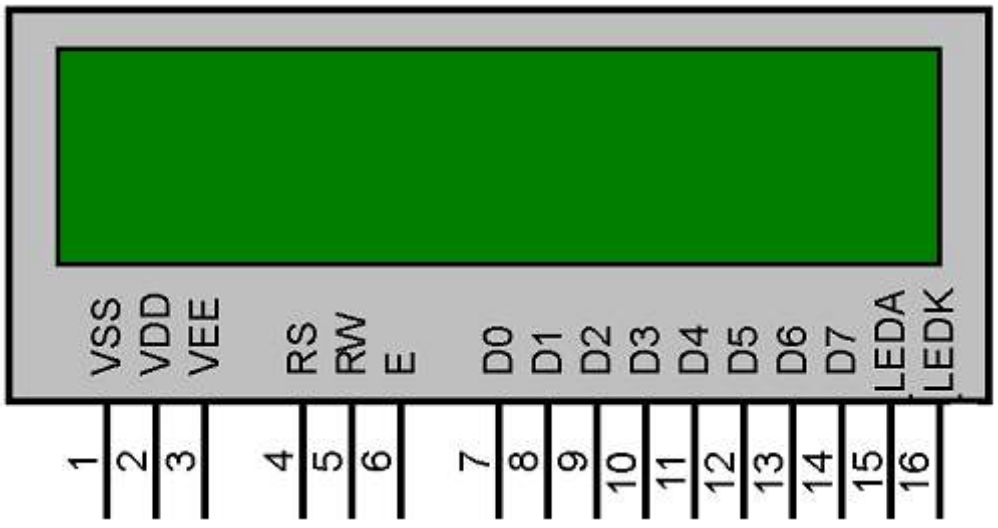
HARDWARE & SOFTWARE REQUIRED:

Hardware: ARM Development Kit
Software: Keil µVision and LaunchPad

THEORY

LCD is abbreviated as liquid crystal display. It is also known from the name of alpha numeric display. It uses small grid of lights to display numbers, letter and digits. 16X2 alphanumeric display which we are using in this tutorial has 32, 5×8 grids and It can display up to 16 characters in one line. So in total it can display 32 character and 16 in each line. So to display 16 character in one line, we need to control 5 x 8 x 16 grids or dot. But fortunately, this liquid display has build in controller which take care of all these grids control and it provide us pin out to send text and control the display. So we do not need to worry about controlling these light grids to display characters.

First pin and second pin feature for power supply of the device. So with the help of these 2 pins only you are going to provide *the* power supply. Coming to third pin which is contrast and this contrast pin can connect with *the* help of one variable resistor. Variable resistor having three pins generally the first pin is vcc and another pin is ground. Third variable pin you are going to connect with your third pin of the device . By varying the resistance for variable resistor is going to increase or decrease the brightness of screen. Pin number four five six these are the three control lines of lcd device.



No	HEX Value	COMMAND TO LCD
1	0x01	Clear Display Screen
2	0x30	Function Set: 8-bit, 1 Line, 5x7 Dots
3	0x38	Function Set: 8-bit, 2 Line, 5x7 Dots
4	0x20	Function Set: 4-bit, 1 Line, 5x7 Dots
5	0x28	Function Set: 4-bit, 2 Line, 5x7 Dots
6	0x06	Entry Mode
7	0x08	Display off, Cursor off
8	0x0E	Display on, Cursor on
9	0x0C	Display on, Cursor off
10	0x0F	Display on, Cursor blinking
11	0x18	Shift entire display left
12	0x1C	Shift entire display right
13	0x10	Move cursor left by one character
14	0x14	Move cursor right by one character
15	0x80	Force cursor to beginning of 1st row
16	0xC0	Force cursor to beginning of 2nd row

GPIO function is the most frequently used functionality of the microcontroller. The GPIO function in both the Ports are controlled by a set of 4 registers: IOPIN, IODIR, IOSET and IOCLR.

IOPIN: It is a GPIO Port Pin Value register and can be used to read or write values directly to the pin. The status of the Pins that are configured as GPIO can always be read from this register irrespective of the direction set on the pin (Input or Output).

The syntax for this register is IOxPIN, where 'x' is the port number i.e. IO0PIN for PORT0 and IO1PIN for PORT1.

IODIR: It is a GPIO Port Direction Control register and is used to set the direction i.e. either input or output of individual pins. When a bit in this register is set to '0', the corresponding pin in the microcontroller is configured as Input. Similarly, when a bit is set as '1', the corresponding pin is configured as Output.

The syntax for this register is IOxDIR, where 'x' is the port number i.e. IO0DIR for PORT0 and IO1DIR for PORT1.

IOSET: It is a GPIO Port Output Set Register and can be used to set the value of a GPIO pin that is configured as output to High (Logic 1). When a bit in the IOSET register is set to '1', the corresponding pin is set to Logic 1. Setting a bit '0' in this register has no effect on the pin.

The syntax for this register is IOxSET, where 'x' is the port number i.e. IO0SET for PORT0 and IO1SET for PORT1.

IOCLR: It is a GPIO Port Output Clear Register and can be used to set the value of a GPIO pin that is configured as output to Low (Logic 0). When a bit in the IOCLR register is set to '1', the corresponding pin in the respective Port is set to Logic 0 and at the same time clears the corresponding bit in the IOSET register. Setting '0' in the IOCLR has no effect on the pin.

The syntax for this register is IOxCLR, where 'x' is the port number i.e. IO0CLR for PORT0 and IO1CLR for PORT1.

PROGRAM:

```
#include<lpc214x.h>
#define bit(x) (1<<x)
#define delay for(i=0;i<10000;i++);
unsigned int i;
void lcd_int();
void dat(unsigned char);
void cmd(unsigned char);
void string(unsigned char* );
void main()
{
IO0DIR|=0XFFF;
 lcd_int();
 cmd(0x8a);
 string("19EUEC502Darshan");
  while(1) {
cmd(0x18);
 delay;
  }
}
void lcd_int()
{
  cmd(0x30);
            cmd(0x0c);
  cmd(0x06);
 cmd(0x01);
 cmd(0x80);
}
void cmd(unsigned char a)
{
  IO0PIN&=0x00;
 IO0PIN|=(a<<0);
 IO0CLR|=bit(8);              //rs=0
 IO0CLR|=bit(9);              //rw=0
  IO0SET|=bit(10);             //en=1
 delay;
  IO0CLR|=bit(10);             //en=0
}
void dat(unsigned char b)
```

```
{
  IO0PIN&=0x00;
 IO0PIN|=(b<<0);
 IO0SET|=bit(8);               //rs=1
 IO0CLR|=bit(9);               //rw=0
  IO0SET|=bit(10);              //en=1
 delay;
 IO0CLR|=bit(10);              //en=0
}
void string(unsigned char *p)
{
   while(*p!='\0') {
   dat(*p++);
 }
}
```

RESULT:

Thus LCD interfacing with ARM is studied.

ADDITIONAL EXPERIMENTS

8051 Based Traffic Light Controller

AIM

To perform the Traffic Light controller operation using 8051 Microcontroller

PROGRAM:

4100			org	4100h
FF0F	contrl		equ	0ff0fh
FF0C	porta		equ	0ff0ch
FF0D	portb		equ	0ff0dh
FF0E	portc		equ	0ff0eh
4100	7480		MOV	A, #80h
4102	90FF0F		MOV	DPTR, #contrl
4105	F0		MOVX	@DPTR, A
4106	7C04	START	MOV	R4, #04H
4108	90419B		MOV	DPTR, #LOOK1
410B	AA83		MOV	R2, DPH
410D	AB82		MOV	R3, DPL
410F	90418F		MOV	DPTR, #LOOK
4112	A883		MOV	R0, DPH
4114	A982		MOV	R1, DPL
4116		GO		
4116	E0		MOVX	A, @DPTR
4117	A883		MOV	R0, DPH
4119	A982		MOV	R1, DPL

411B	90FF0C		MOV	DPTR, #porta
411E	F0		MOVX	@DPTR, A
411F	09		INC	R1
4120	8883		MOV	DPH, R0
4122	8982		MOV	DPL, R1
4124	E0		MOVX	A, @DPTR
4125	A883		MOV	R0, DPH
4127	A982		MOV	R1, DPL
4129	90FF0D		MOV	DPTR, #portb
412C	F0		movx	@dptr. a
412D	09		INC	R1
412E	8883		MOV	DPH, R0
4130	8982		MOV	DPL, R1
4132	E0		MOVX	A, @DPTR
4133	A883		MOV	R0, DPH
4135	A982		MOV	R1, DPL
4137	90FF0E		MOV	DPTR, #portc
413A	F0		MOVX	@DPTR, A
413B	09		INC	R1
413C	124175		LCALL	DELAY

413F	8A83		MOV	DPH, R2
4141	8B82		MOV	DPL, R3
4143	E0		MOVX	A, @DPTR
4144	AA83		MOV	R2, DPH
4146	AB82		MOV	R3, DPL
4148	90FF0C		MOV	DPTR, #porta
414B	F0		MOVX	@DPTR, A
414C	0B		INC	R3
414D	8A83		MOV	DPH, R2
414F	8B82		MOV	DPL, R3
4151	E0		MOVX	A, @DPTR
4152	AA83		MOV	R2, DPH
4154	AB82		MOV	R3, DPL
4156	90FF0D		MOV	DPTR, #portb
4159	F0		MOVX	@DPTR, A
415A	0B		INC	R3
415B	8A83		MOV	DPH, R2
415D	8B82		MOV	DPL, R3
415F	E0		MOVX	A, @DPTR
4160	AA83		MOV	R2, DPH

4162	AB82		MOV	R3, DPL
4164	90FF0E		MOV	DPTR, @portc
4167	F0		MOVX	A, @DPTR
4168	0B		INC	R3
4169	124182		LCALL	DELAY1
416C	8883		MOV	DPH, R0
416E	8982		MOV	DPL, R1
4170	DCA4		DJNZ	R4, GO
4172	124106		LCALL	START
4175	7D12	DELAY1:	MOV	R5, #12H
4177	7EFF	L3:	MOV	R6, #0ffH
4179	7FFF	L2:	MOV	R7, #0ffH
417B	DFFE	L1:	DJNZ	R7, L1
417D	DEFA		DJNZ	R6, L2
417F	DDF6		DJNZ	R5, L3
4181	22	DELAY1:	RET	
4182	7D12	L6:	MOV	R5, #12H
4184	7EFF	L5:	MOV	R6, #0ffH
4186	7FFF	L4:	MOV	R7, #0ffH
4188	DFFE		DJNZ	R7, L4

418A	DEFA		DJNZ	R6, L5
418C	DDF6		DJNZ	R5, L6
418E	22		RET	
418F	44 27 12	LOOK:	DB	44rl, 27H, 12H
4192	92 2B 10		DB	92H, 2BH, 10H
4195	84 9D 10		DB	84H, 9DH, 10H
4198	84 2E 48		DB	84H, 2EH, 48H
419B	48 27 12	LOOK1:	DB	48H, 27H, 12H
419E	92 4B 10		DB	92H, 4BH, 10H
41A1	84 9D 20		DB	84H, 9DH, 20H
41A4	04 2E 49		DB END	04H, 2EH, 49H

Result:

Thus the Traffic Light program is performed using 8051 and the outputs are verified.

CODE CONVERSION USING 8051

AIM

To perform the Code conversion from

(i) BCD - ASCII
(ii) ASCII – Decimal
(iii) Decimal – ASCII
(iv) HEX – Decimal
(v) Decimal – HEX
(vi) HEX-Binary

(i) BCD-ASCII

PROGRAM

LABEL	MNEMONICS	COMMENT
	MOV R0, #23	LCALL CONV
	LCALL 9000	
	LCALL 0003	

(ii) ASCII- DECIMAL

PROGRAM

LABEL	MNEMONICS	COMMENT
	MOV R1, #39	Input char
	MOV A, R1	
	LCALL 9000	LCALL CONV
	LCALL 0003	
CONV	CLR C	
	SUBB A, #41	
	MOV A, R1	
	JC DGT	
	CLR C	
	SUBB A, #37	ASCII -> CHAR
	RET	
DGT	CLR C	
	SUBB A, #30	ASCII -> NUMBER
	RET	

(iii) DECIMAL - ASCII

PROGRAM

LABEL	MNEMONICS	COMMENT
	MOV R1, #0B	Input Char
	MOV A, R1	
	LCALL 9000	LCALL CONV
	LCALL 0003	
CONV	CLR C	
	SUBB A, #0A	
	MOV A, R1	
	JC 9009	JC DGT
	ADD A, #37	ASCII (CHAR)
	RET	
DGT	ADD A, #30	ASCII (NUMBER)
	RET	

(iv) HEXADECIMAL- DECIMAL

Program:

Memory Address	Opcode	Mneumonics
4100	90	MOV DPTR,#4500
4101	45	
4102	00	
4103	E0	MOVX A,@DPTR
4104	75	MOV B,#64
4105	F0	
4106	64	
4107	84	DIV AB
4108	90	MOV DPTR,#4501
4109	45	
410A	01	
410B	F0	MOVX @DPTR,A
410C	E5	MOV A,B
410D	F0	
410E	75	MOV B,#0A
410F	F0	
4110	0A	
4111	84	DIV AB
4112	A3	INC DPTR
4113	F0	MOVX @DPTR,A
4114	A3	INC DPTR
4115	E5	MOV A,B
4116	F0	
4117	F0	
4118	80	MOVX @DPTR,A
4119	FE	HLT: SJMP HLT

Input:
4500-87
Output:
4501-01
4502-03
4503-05

(v) DECIMAL- HEXADECIMAL

Program:

Memory Address	Opcode	Mneumonics
4100	90	MOV DPTR,#4200
4101	42	
4102	00	
4103	E0	MOVX A,@DPTR

(vi) HEX- Binary

4100	MOV	DPTR, #4500
4103	MOV	B, #08
4106	MOV	A, #5A
4108	CLR	C
4109	MOV	R0, #00
410B	MOV	R1, #01
410D	RRC	A
410E	MOV	R3, A
410F	JC	LOOP 1 [4115]
4111	MOV	A, R0
4112	LCALL	LOOP 2 [4116]
LOOP1: 4115	MOV	A, R1

LOOP2: 4116	MOVX	@DPTR, A
4117	MOV	A, R3
4118	DEC	B
411A	INC	DPTR
411B	JNZ	4100 (LOOP 3)
411D	SJMP	411D

4100	00	MOV DPTR,4200
4101	42	
4102	00	MOVX A,@DPTR
4103	E0	
4104	75	MOV B,#0A
4105	F0	
4106	0A	
4107	A4	MUL AB
4108	F5	MOV B,A
4109	F0	
410A	A3	INC DPTR
410B	E0	MOVX A,@DPTR
410C	25	ADD A,B
410D	F0	
410E	A3	INC DPTR
410F	F0	MOVX @DPTR,A
4110	80	SJMP 4111
4111	FE	

Input:

Output:

Result:

Thus the code conversions are performed using 8051 and the outputs are verified.