IMPACT COLLEGE OF ENGINEERING & APPLIED SCIENCES

Kodigehalli Post, Bangalore – 560 092

MICROCONTROLLER LABORATORY (IPCC-BCS402)

(As per Visvesvaraya Technological University Syllabus)

Compiled by:

Prof. VIJENDRA SN
Asst. Professor
Dept. of CS & E

Dr. DHANANJAYA V
Prof & Head of Department
Dept. of CS & E

NAME : ______________________________

USN : ______________________________

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2025

PROJECT CREATION IN KEILUV4 IDE:

Create a project folder before creating NEW project.

- Open Keil uVision4 IDE software by double clicking on “Keil Uvision4” icon.
- Go to “Project” then to “New uVision Project” and save it with a name in the respective project folder, already you created.
- Select the device as “NXP” In that “LPC2148” then press OK and then press “YES” Button to add “startup.s” file.
- In startup file go to Configuration Wizard. In Configuration Wizard window uncheck PLL Setup and check VPBDIV Setup.
- Go to “File” In that “New” to open an editor window. Create your source file and use the header file “lpc21xx.h” in the source file and save the file. Color syntax highlighting will be enabled once the file is saved with a extension such as “.C “.
- Right click on “Source Group 1” and select the option “Add Existing Files to Group
- Source Group 1“add the *.C source file(s) to the group. After adding the source file you can see the file in Project Window. Then go to “Project” in that “Translate” to compile the File (s). Check out the Build output window.

Right click on Target1 and select options for Target Target1. Then go to option “Target” in that

- Xtal 12.0MHz
- Select “Use MicroLIB”.
- Select IROM1 (starting 0x0 size 0x80000).
- Select IRAM1 (starting 0x40000000 size 0x8000).

Then go to option “Output”

- Select “Create Hex file”

Then go to option “Linker”

- Select “Use Memory Layout for Target Dialog”.

To come out of this window press OK.

Go to “Project” in that “Build Target” for building all source files such as “.C”,”.ASM”, “.h”, files, etc…This will create the *.HEX file if no warnings & no Errors. Check out the Build output window.

FLASH MAGIC VERSION 6.1:

Installation of Flash Magic as follows.

Go to EXE folder and then Flash Magic 6.1 folder in the CD & run FlashMagic.exe

- Next
- Click on the option “I Accept the Agreement” and then give Next
- Then it asks the Destination location, Click Next.
- Further Select start menu folder, Click Next.
- Select “Create a desktop icon” then Next • It prompts “Ready to Install” Click INSTALL.
- Click Finish to complete the installation.

ISP PROGRAMMING:

FLASH MAGIC software can be used to download the HEX files to the Flash memory of controller.

SETTINGS IN FLASH MAGIC:

Step 1. Communications:

• Device	: LPC2148
• Com Port	: COM1(Check and connect)
• Baud Rate	: 9600
• Interface	: None(ISP)
• Oscillator	: 12MHz

Step 2. ERASE:

- Select “Erase Blocks Used By Hex File”.

Step 3. Hex file:

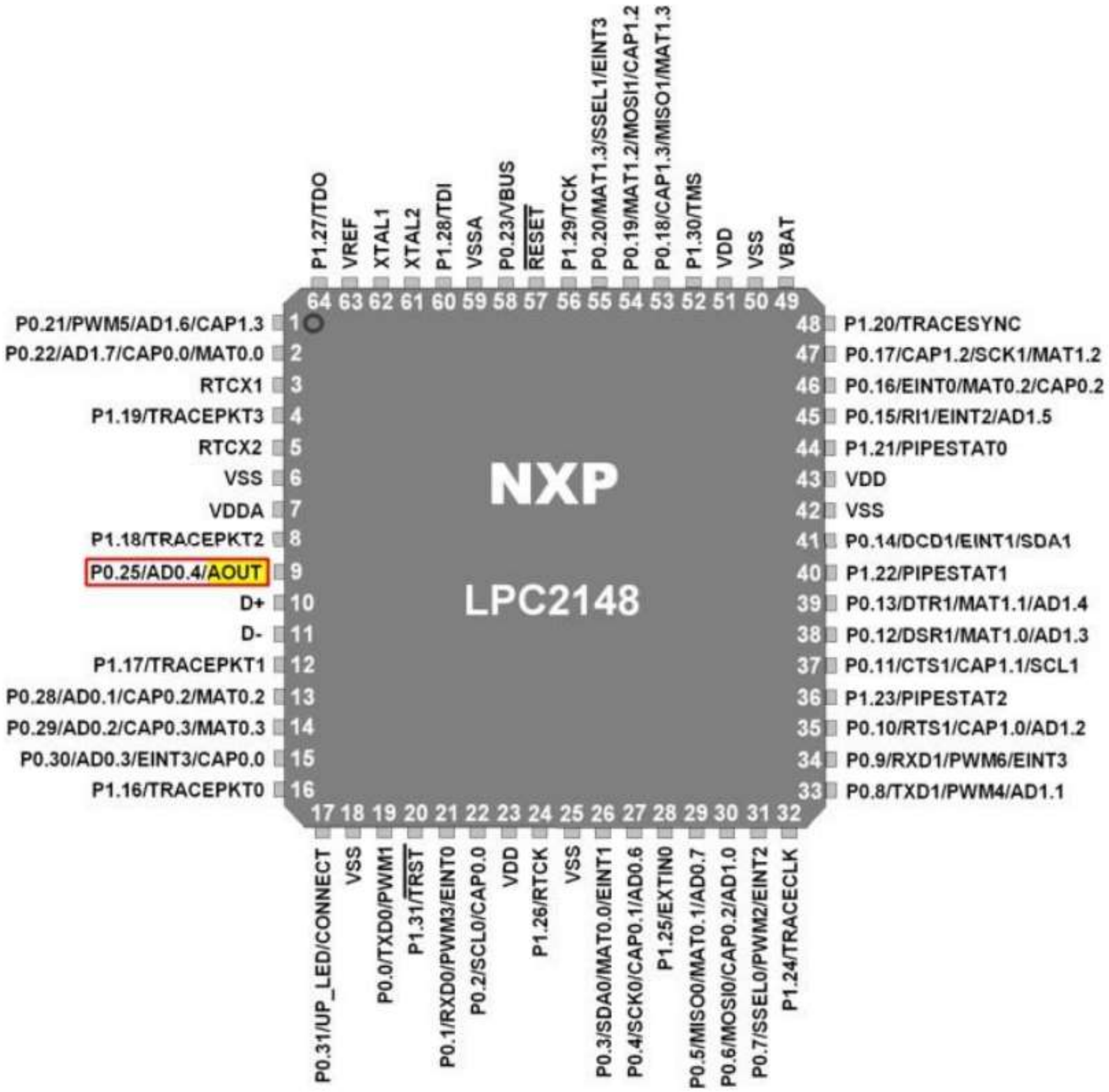
- Browse and select the Hex file which you want to download.

Step 4. Options

- Select “Verify after programming”.

Step 5. Start:

- Click Start to download the hex file to the controller. After downloading the code, the program starts executing in the hardware, then remove the ISP jumper JP7.


NXP
LPC2148
64 P1.27/TDO
63 VREF
62 XTAL1
61 XTAL2
60 P1.28/TDI
59 VSSA
58 P0.23/VBUS
57 RESET
56 P1.29/TCK
55 P0.20/MAT1.3/SSEL1/EINT3
54 P0.19/MAT1.2/MOSI1/CAP1.2
53 P0.18/CAP1.3/MISO1/MAT1.3
52 P1.30/TMS
51 VDD
50 VSS
49 VBAT
1 P0.21/PWM5/AD1.6/CAP1.3
2 P0.22/AD1.7/CAP0.0/MAT0.0
3 RTCX1
4 P1.19/TRACEPKT3
5 RTCX2
6 VSS
7 VDDA
8 P1.18/TRACEPKT2
9 P0.25/AD0.4/AOUT
10 D+
11 D-
12 P1.17/TRACEPKT1
13 P0.28/AD0.1/CAP0.2/MAT0.2
14 P0.29/AD0.2/CAP0.3/MAT0.3
15 P0.30/AD0.3/EINT3/CAP0.0
16 P1.16/TRACEPKT0
17 P0.31/UP_LED/CONNECT
18 VSS
19 P0.0/TXD0/PWM1
20 P1.31/TRST
21 P0.1/RXD0/PWM3/EINT0
22 P0.2/SCL0/CAP0.0
23 VDD
24 P1.26/RTCK
25 VSS
26 P0.3/SDA0/MAT0.0/EINT1
27 P0.4/SCK0/CAP0.1/AD0.6
28 P1.25/EXTIN0
29 P0.5/MISO0/MAT0.1/AD0.7
30 P0.6/MOSI0/CAP0.2/AD1.0
31 P0.7/SSEL0/PWM2/EINT2
32 P1.24/TRACECLK
48 P1.20/TRACESYNC
47 P0.17/CAP1.2/SCK1/MAT1.2
46 P0.16/EINT0/MAT0.2/CAP0.2
45 P0.15/RI1/EINT2/AD1.5
44 P1.21/PIPESTAT0
43 VDD
42 VSS
41 P0.14/DCD1/EINT1/SDA1
40 P1.22/PIPESTAT1
39 P0.13/DTR1/MAT1.1/AD1.4
38 P0.12/DSR1/MAT1.0/AD1.3
37 P0.11/CTS1/CAP1.1/SCL1
36 P1.23/PIPESTAT2
35 P0.10/RTS1/CAP1.0/AD1.2
34 P0.9/RXD1/PWM6/EINT3
33 P0.8/TXD1/PWM4/AD1.1

1. DEVELOP AND SIMULATE ARM ALP FOR DATA TRANSFER, ARITHMETIC AND LOGICAL OPERATIONS.

A. Develop Assembly Language Programs to test Data Transfer instructions.

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP1,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R5,=5	
	LDR R7,=8	
	MOV R7,R5	
	MVN R7,R5	
STOP	B STOP	
	END	

B. Develop Assembly Language Programs to test Arithmetic Operations.

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP2,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R1,=0x00000002	
	LDR R2,=0x00000001	
	ADD R0,R1,R2	
	ADD R0,R1,R1,LSL #1	
	SUB R0,R1,R2	
STOP	B STOP	
	END	

C. DEVELOP ASSEMBLY LANGUAGE PROGRAMS TO TEST REVERSE OPERATIONS.

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP3,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R2,=0x00000077	
	RSB R0,R2,#0	; RSB Subtracts R1 from constant value #0; R0=-R2 (Performs 2's Complement)
STOP	B STOP	
	END	

D. DEVELOP ASSEMBLY LANGUAGE PROGRAMS TO TEST LOGICAL OPERATIONS INSTRUCTIONS

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP4,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R1,=0x02040608	
	LDR R2,=0x10305070	
	ORR R0,R1,R2	
	AND R0,R1,R2	
	EOR R0,R1,R2	
	BIC R0,R1,R2	
STOP	B STOP	
	END	

E. DEVELOP ASSEMBLY LANGUAGE PROGRAMS TO TEST LOAD AND STORE INSTRUCTIONS.

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP5,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x40000000	
	LDR R1,[R0]	
	LDR R2,=0x40000050	
	STR R1,[R2]	
STOP	B STOP	
	END	

F. DEVELOP ASSEMBLY LANGUAGE PROGRAMS TO TEST FOR CARRY FLAG C IN CPSR.

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP6,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R1,=0x80000004	
	LDR R3,=0x00000001	
	MOVS R0,R1,LSL #1	
	SUBS R1,R3,#1	
STOP	B STOP	
	END	

G. DEVELOP AN ASSEMBLY LANGUAGE PROGRAMS TO TEST MOVS AND MOVCS INSTRUCTIONS.

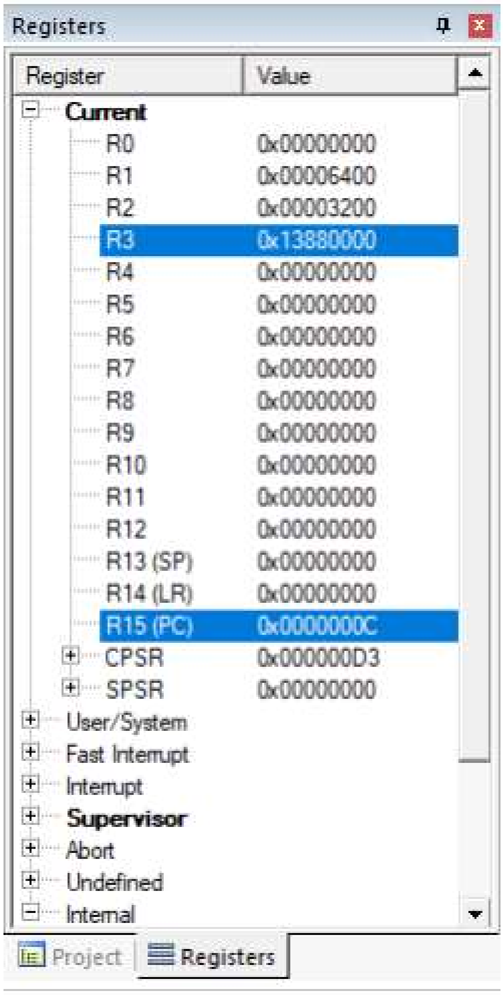
LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP7,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R1,=0x80000004	
	MOVS R0,R1,LSL #1	
	MOVCS R0,R1	; if Carry is set then R0=R1
STOP	B STOP	
	END	

H. DEVELOP AN ASSEMBLY LANGUAGE PROGRAMS TO TEST ADDCS INSTRUCTIONS

LABEL FIELD	MNEMONIC FIELD	COMMENTS FIELD
	AREA ALP8,CODE,READONLY	
	ENTRY	; Mark first instruction to execute
	LDR R0,=0x00000000	
	LDR R1,=0x80000004	
	MOVS R0,R1,LSL #1	
	ADDCS R2, R0,R1	
STOP	B STOP	
	END	

2. DEVELOP AN ALP TO MULTIPLY TWO 16 BIT NUMBERS

LABEL FIELD	MNEMONIC FIELD	COMMENTS FILED
	AREA MULTIPLY, CODE, READONLY	
	ENTRY	; Mark first instruction to execute
	MOV R1,#0X6400	; STORE FIRST NUMBER IN R0
	MOV R2,#0X3200	; STORE SECOND NUMBER IN R1
	MUL R3,R1,R2	; MULTIPLICATION
HERE	B HERE	
	END	; MARK END OF FILE

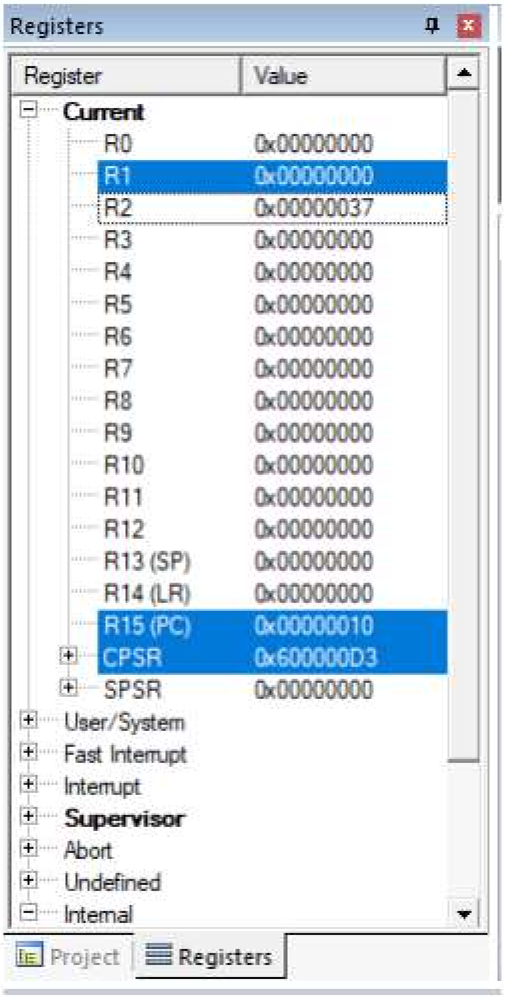


Registers

Register	Value
Current	
R0	0x00000000
R1	0x00006400
R2	0x00003200
R3	0x13880000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	

3. DEVELOP AN ALP TO FIND THE SUM OF FIRST 10 INTEGER NUMBERS

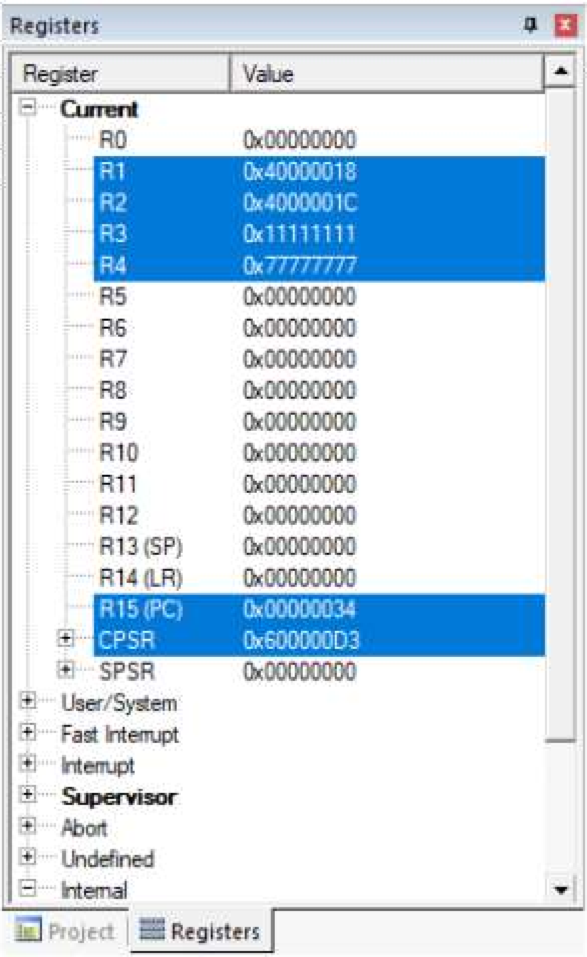
LABEL FIELD	MNEMONIC FIELD	COMMENTS FILED
	AREA INTSUM, CODE, READONLY	
	ENTRY	; Mark first instruction to execute
	MOV R1,#10	; LOAD 10 TO REGISTER
	MOV R2,#0	; EMPTY R2 REGISTER TO STORE RESULT
LOOP	ADD R2,R2,R1	; ADD THE CONTERNT OF R1 WITH RESULT AT R2
	SUBS R1,#0X01	; DECREMENT R1 BY 1
	BNE LOOP	; REPEAT TILL R1 GOES TO ZERO
HERE	B HERE	
	END	



Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000037
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	

4. DEVELOP AN ALP TO FIND THE SMALLEST NUMBER IN AN ARRAY

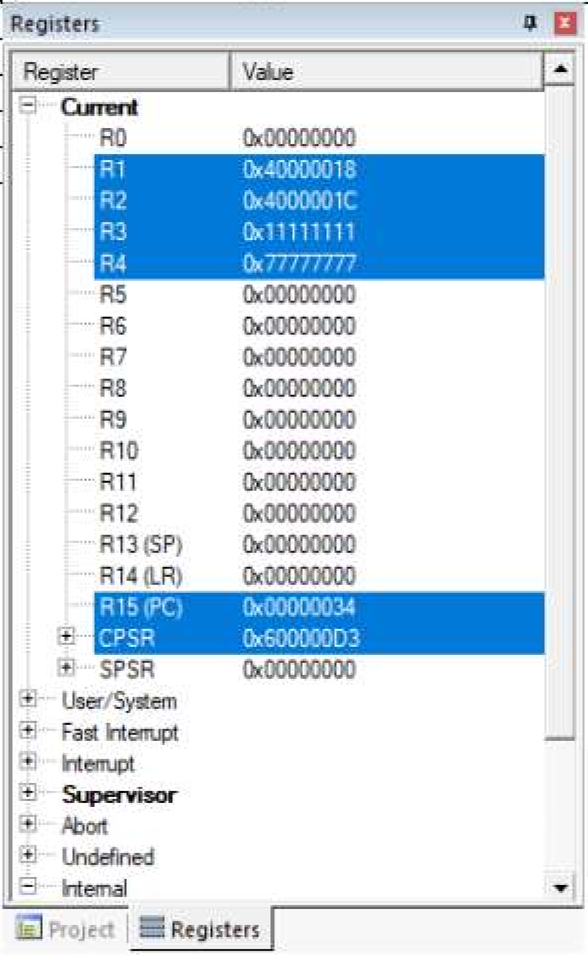
LABEL FIELD	MNEMONIC FIELD	COMMENT FIELD
	AREA LAR_SMAL, CODE, READONLY	
	ENTRY	
	MOV R5,#06	; COUNTER VALUE E.G 7 NUMBERS
	MOV R1,#0X40000000	; START OF THE DATA MEMORY
	MOV R2,#0X4000001C	; RESULT LOCATION
	LDR R3,[R1]	; GET THE FIRST DATA
LOOP	ADD R1,R1,#04	; MEMORY POINTER UPDATED TO FETCH 2ND DATA
	LDR R4,[R1]	; GET SECOND NUMBER
	CMP R3,R4	; COMPARE BOTH NUMBERS
	BLS LOOP1	;BHI → for large; IF 1ST> 2ND THAN LOOP1
	MOV R3,R4	
LOOP1	SUBS R5,R5,#01	; DECREMENT THE COUNTER
	CMP R5,#00	
	BNE LOOP	
	STR R3,[R2]	
STOP	B STOP	
	END	



Register	Value
Current	
R0	0x00000000
R1	0x40000018
R2	0x4000001C
R3	0x11111111
R4	0x77777777
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000034
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	

5.DEVELOP AN ALP TO FIND THE LARGEST NUMBER IN AN ARRAY

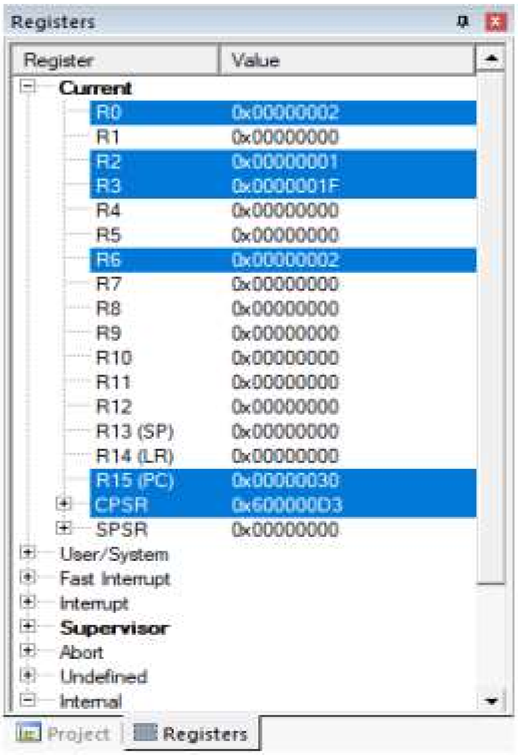
LABEL FIELD	MNEMONIC FIELD	COMMENT FIELD
	AREA LAR_SMAL,CODE, READONLY	
	ENTRY	
	MOV R5,#06	; COUNTER VALUE E.G 7 NUMBERS
	MOV R1,#0X40000000	; START OF THE DATA MEMORY
	MOV R2,#0X4000001C	; RESULT LOCATION
	LDR R3,[R1]	; GET THE FIRST DATA
LOOP	ADD R1,R1,#04	; MEMORY POINTER UPDATED TO FETCH 2ND DATA
	LDR R4,[R1]	; GET SECOND NUMBER
	CMP R3,R4	; COMPARE BOTH NUMBERS
	BHI LOOP1	;BHI → for large; IF 1ST> 2ND THAN LOOP1
	MOV R3,R4	
LOOP1	SUBS R5,R5,#01	; DECREMENT THE COUNTER
	CMP R5,#00	
	BNE LOOP	
	STR R3,[R2]	
STOP	B STOP	
	END	



```
Register        Value
Current
  R0            0x00000000
  R1            0x40000018
  R2            0x4000001C
  R3            0x11111111
  R4            0x77777777
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x00000034
  CPSR          0x600000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
```

6. DEVELOP AN ALP TO COUNT THE NUMBER OF ONES AND ZEROS AND STORE RESULT IN TWO CONSECUTIVE MEMORY LOCATIONS.

LABEL FIELD	MNEMONIC FIELD	COMMENT FIELD
	AREA ONEZERO , CODE, READONLY	
	ENTRY	;MARK FIRST INSTRUCTION TO EXECUTE
	MOV R2,#0	; COUNTER FOR ONES
	MOV R3,#0	; COUNTER FOR ZEROS
	MOV R6,#0X00000002	; LOADS THE VALUE
	MOV R1,#32	; 32 BITS COUNTER
	MOV R0,R6	; GET THE 32 BIT VALUE
	MOV R0,R6	; GET THE 32 BIT VALUE
LOOP0	MOVS R0,R0,ROR #1	; RIGHT SHIFT TO CHECK CARRY BIT (1'S/0'S)
	BHI ONES	; IF C=1 GOTO ONES BRANCH OTHERWISE NEXT
ZEROS	ADD R3,R3,#1	; IF C= 0 THEN INCREMENT THE COUNTER BY 1(R3)
	B LOOP1	; BRANCH TO LOOP1
ONES	ADD R2,R2,#1	; IF C=1 THEN INCREMENT THE COUNTER BY 1(R2)
LOOP1	SUBS R1,R1,#1	; COUNTER VALUE DECREMENTED BY 1
	BNE LOOP0	; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT
STOP	B STOP	
	END	

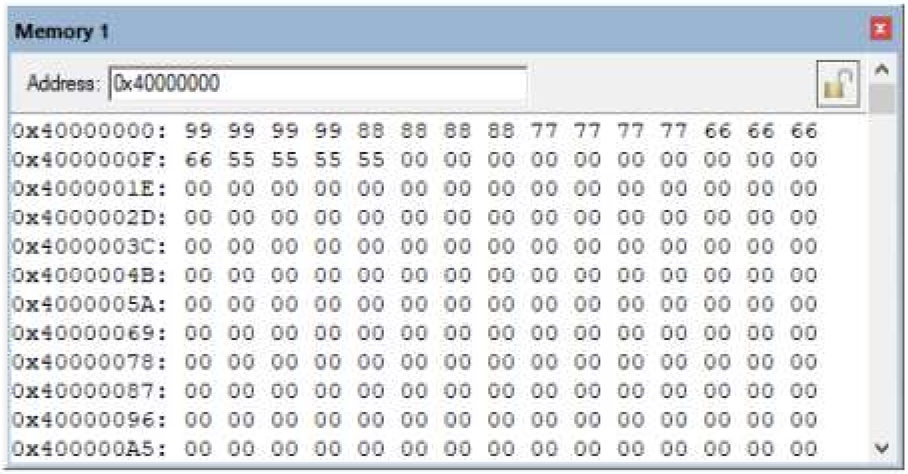


Register	Value
Current	
R0	0x00000002
R1	0x00000000
R2	0x00000001
R3	0x0000001F
R4	0x00000000
R5	0x00000000
R6	0x00000002
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	

7. DEVELOP AN ALP TO ARRANGE A SERIES OF 32 BIT NUMBERS IN ASCENDING ORDER.

LABEL FIELD	MNEMONIC FIELD	COMMENT FIELD
	AREA ASCENDING , CODE, READONLY	
	ENTRY	
	MOV R0,#05	; OUTER LOOP
OUTTERLOOP	MOV R5,#0X40000000	; DATA ADDRESS
	ADD R6,R5,#4	; INC TO CMP WITH NEXT DATA
	MOV R3,#4	; INNER LOOP
INNERLOOP	LDR R1,[R5]	; GET 1ST DATA
	LDR R2,[R6]	; GET 2ND DATA
	CMP R1,R2	; COMPARE 2 NO'S
	BLO LOOP3	; IF 1>2 THEN NO NEED TO EXCHANGE; BHI
	MOV R4,R2	; IF 1<2 THEN EXCHANGE
	MOV R2,R1	
	MOV R1,R4	
LOOP3	STR R1,[R5]	
	STR R2,[R6]	
	ADD R5,R5,#04	; INC 4 TIMES TO GET NEXT DATA FOR CMP
	ADD R6,R6,#04	
	SUBS R3,R3,#01	; DECREMENT INNER LOOP
	BNE INNERLOOP	
	SUBS R0,R0,#1	
	BNE OUTTERLOOP	; DECREMENT OUTTER LOOP
STOP	B STOP	
	END	

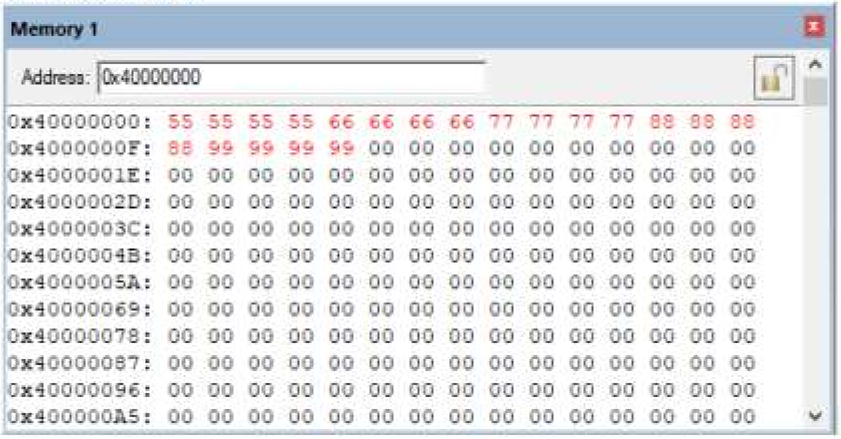
Before Sorting for ascending:



```
Address: 0x40000000

0x40000000: 99 99 99 99 88 88 88 88 77 77 77 77 66 66 66
0x4000000F: 66 55 55 55 55 00 00 00 00 00 00 00 00 00 00
0x4000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000002D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000087: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After excution for ascending:



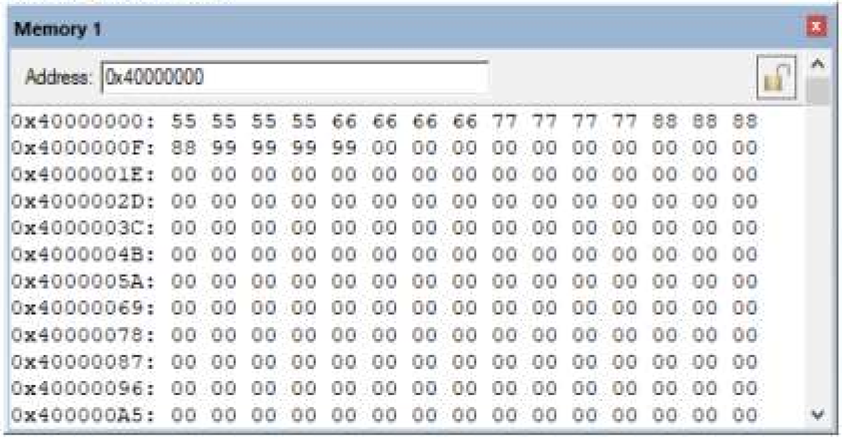
Address: 0x40000000

```
0x40000000: 55 55 55 55 66 66 66 66 77 77 77 77 88 88 88
0x4000000F: 88 99 99 99 99 00 00 00 00 00 00 00 00 00 00
0x4000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000002D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000087: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

8. DEVELOP AN ALP TO ARRANGE A SERIES OF 32 BIT NUMBERS IN DESCENDING ORDER

LABEL FIELD	MNEMONIC FIELD	COMMENT FIELD
	AREA ASCENDING , CODE, READONLY	
	ENTRY	
	MOV R0,#05	; OUTER LOOP
OUTTERLOOP	MOV R5,#0X40000000	; DATA ADDRESS
	ADD R6,R5,#4	; INC TO CMP WITH NEXT DATA
	MOV R3,#4	; INNER LOOP
INNERLOOP	LDR R1,[R5]	; GET 1ST DATA
	LDR R2,[R6]	; GET 2ND DATA
	CMP R1,R2	; COMPARE 2 NO'S
	BHI LOOP3	; IF 1>2 THEN NO NEED TO EXCHANGE;
	MOV R4,R2	; IF 1<2 THEN EXCHANGE
	MOV R2,R1	
	MOV R1,R4	
LOOP3	STR R1,[R5]	
	STR R2,[R6]	
	ADD R5,R5,#04	; INC 4 TIMES TO GET NEXT DATA FOR CMP
	ADD R6,R6,#04	
	SUBS R3,R3,#01	; DECREMENT INNER LOOP
	BNE INNERLOOP	
	SUBS R0,R0,#1	
	BNE OUTTERLOOP	; DECREMENT OUTTER LOOP
STOP	B STOP	
	END	

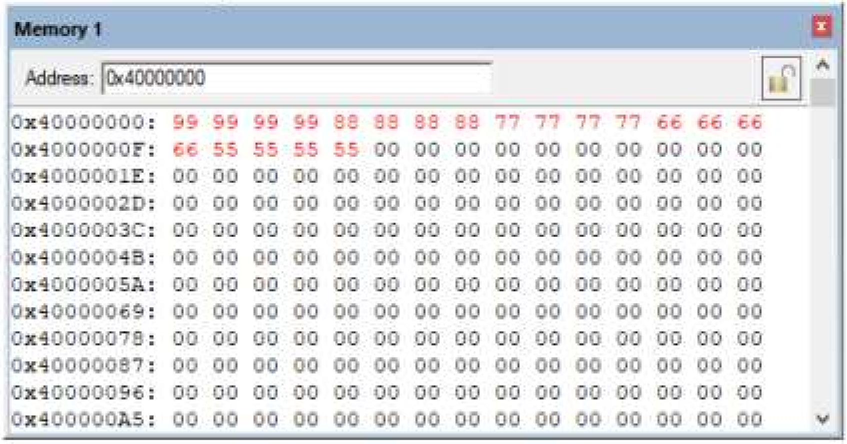
Before Exection for Descending:



Address: 0x40000000

```
0x40000000: 55 55 55 55 66 66 66 66 77 77 77 77 88 88 88
0x4000000F: 88 99 99 99 99 00 00 00 00 00 00 00 00 00 00
0x4000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000002D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000087: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution for Descending:



Address: 0x40000000

```
0x40000000: 99 99 99 99 88 88 88 88 77 77 77 77 66 66 66
0x4000000F: 66 55 55 55 55 00 00 00 00 00 00 00 00 00 00
0x4000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000002D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000087: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

NOTE: BLO instruction for Descending
BHI instruction for Ascending

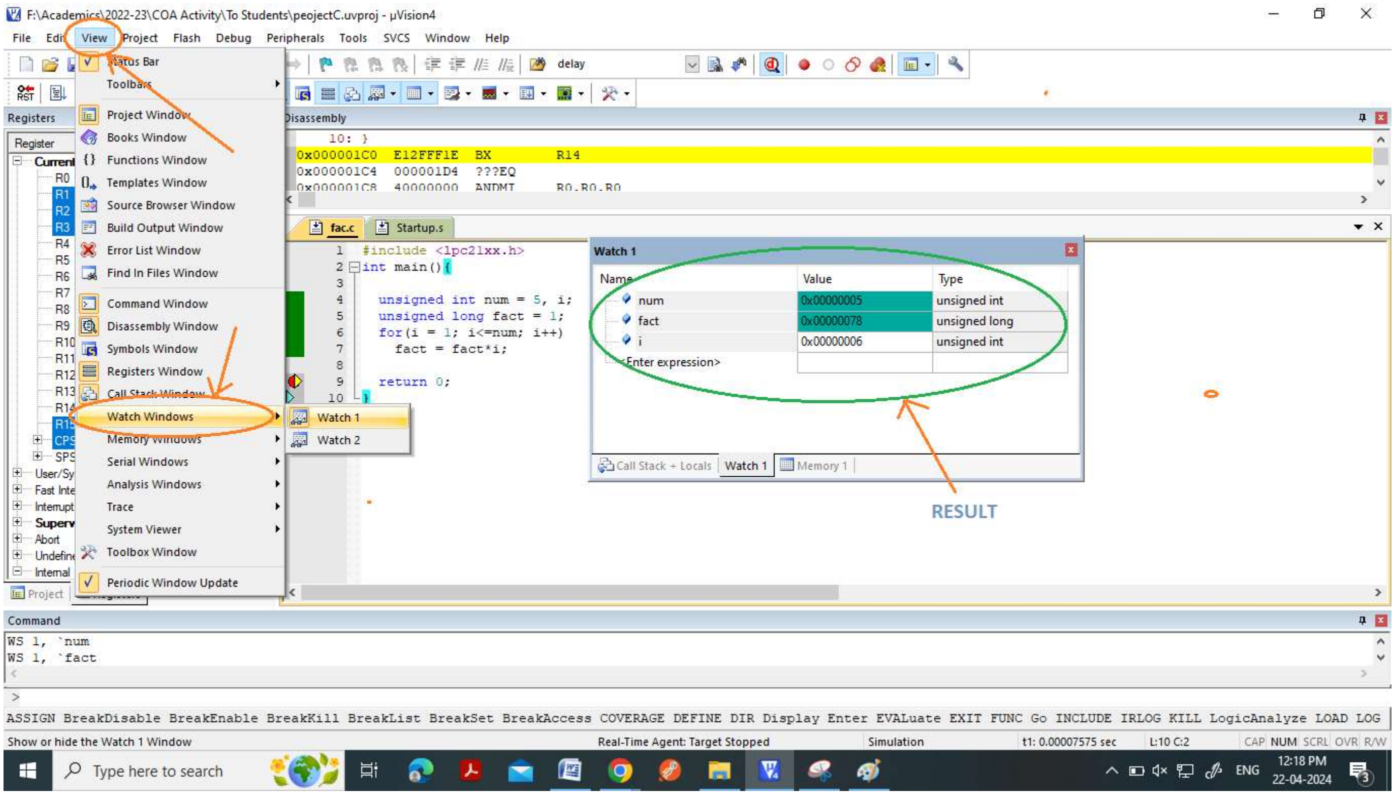
9. SIMULATE A PROGRAM IN C FOR ARM MICROCONTROLLER TO FIND FACTORIAL OF A NUMBER.

PROGRAM:

```
#include <lpc21xx.h>
int main()
{
        unsigned int num = 5, i;
        unsigned long fact = 1;
        for(i = 1; i<=num; i++)
                fact = fact * i;

        return 0;
}
```

OUTPUT:



F:\Academics\2022-23\COA Activity\To Students\peojectC.uvproj - µVision4

```
    10: }
0x000001C0  E12FFF1E  BX        R14
0x000001C4  000001D4  ???EQ
0x000001C8  40000000  ANDMI     R0,R0,R0
```

```
#include <lpc21xx.h>
int main(){

  unsigned int num = 5, i;
  unsigned long fact = 1;
  for(i = 1; i<=num; i++)
    fact = fact*i;

  return 0;
}
```

Name	Value	Type
num	0x00000005	unsigned int
fact	0x00000078	unsigned long
i	0x00000006	unsigned int
<Enter expression>		

```
WS 1, `num
WS 1, `fact
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG KILL LogicAnalyze LOAD LOG
```

Show or hide the Watch 1 Window   Real-Time Agent: Target Stopped   Simulation   t1: 0.00007575 sec   L:10 C:2   CAP NUM SCRL OVR R/W

10. SIMULATE A PROGRAM IN C FOR ARM MICROCONTROLLER TO DEMONSTRATE CASE CONVERSION OF CHARACTERS FROM UPPER TO LOWERCASE AND LOWER TO UPPERCASE.

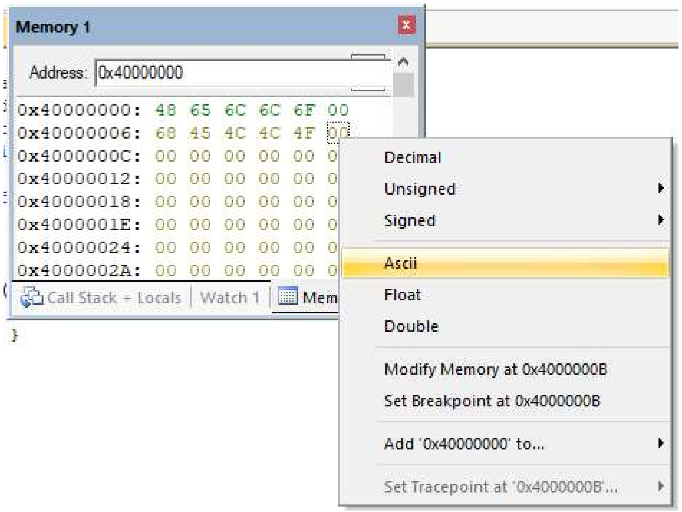
PROGRAM:

```
#include <lpc21xx.h>
char src[ ] = "Hello";
char dest[ ] = "";
void caseConvert()
{
        unsigned int i;
        for (i = 0; src[i]!='\0'; i++)
        {
                if(src[i] >= 'a' && src[i] <= 'z')
                 {
                   dest[i] = src[i] - 32;
                  }
                if(src[i] >= 'A' && src[i] <= 'Z')
                {
                        dest[i] = src[i] + 32;
                }
        }
}
int main()
{
        caseConvert();
        return 0;
}
```

In Keil uVision, when we perform a memory dump and the values are displayed in hexadecimal format, we can easily see their equivalent ASCII characters by looking at the ASCII representation of each hexadecimal value.

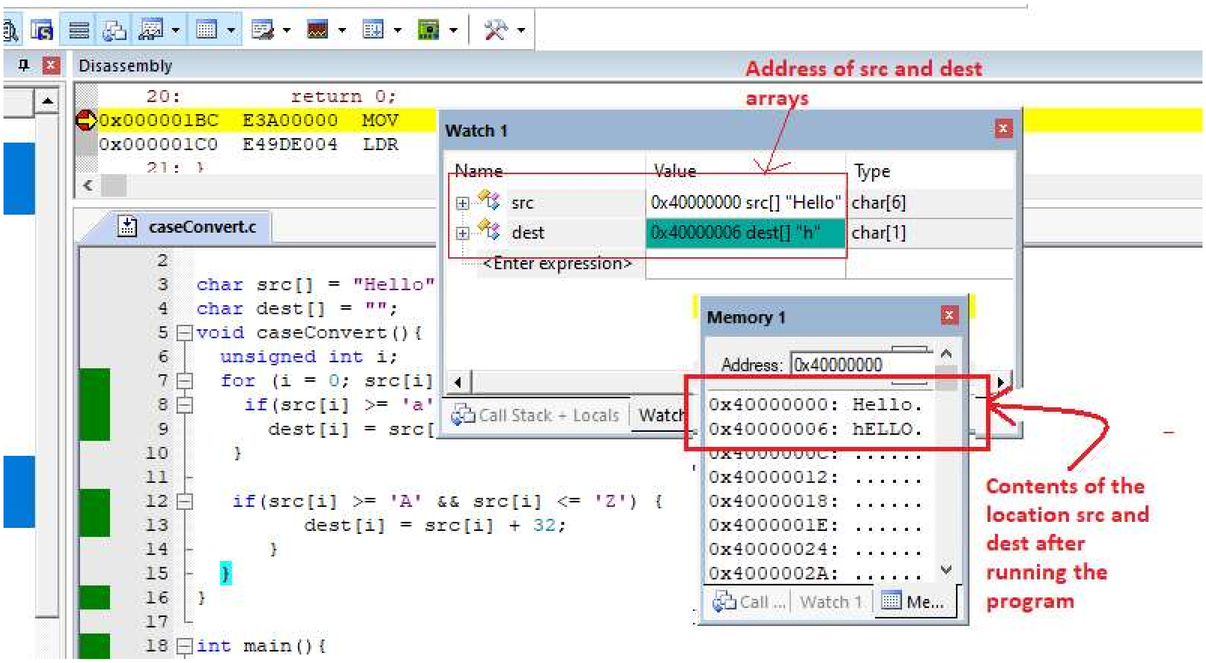
Typically, in the memory dump window, there will be two columns - one displaying the memory address and the other displaying the memory contents in hexadecimal format. To see the ASCII representation:

1. Locate the column displaying the hexadecimal values.
2. For each byte of memory content displayed in hexadecimal, you can refer to the ASCII representation of that byte by **right clicking on it and changing to "ASCII"**



```
0x40000000: 48 65 6C 6C 6F 00
0x40000006: 68 45 4C 4C 4F 00
0x4000000C: 00 00 00 00 00 0
0x40000012: 00 00 00 00 00 0
0x40000018: 00 00 00 00 00 0
0x4000001E: 00 00 00 00 00 0
0x40000024: 00 00 00 00 00 0
0x4000002A: 00 00 00 00 00 0
```

OUTPUT:



Address of src and dest arrays

```
    20:         return 0;
0x000001BC  E3A00000  MOV
0x000001C0  E49DE004  LDR
    21: }
```

```

char src[] = "Hello"
char dest[] = "";
void caseConvert(){
  unsigned int i;
  for (i = 0; src[i]
    if(src[i] >= 'a'
      dest[i] = src[
   }

   if(src[i] >= 'A' && src[i] <= 'Z') {
         dest[i] = src[i] + 32;
      }
  }
}

int main(){
```

Name	Value	Type
src	0x40000000 src[] "Hello"	char[6]
dest	0x40000006 dest[] "h"	char[1]
<Enter expression>		

```
0x40000000: Hello.
0x40000006: hELLO.
0x4000000C: ......
0x40000012: ......
0x40000018: ......
0x4000001E: ......
0x40000024: ......
0x4000002A: ......
```

Contents of the location src and dest after running the program

11. DEMONSTRATE ENABLING AND DISABLING OF INTERRUPTS IN ARM.

```
        AREA Program, CODE, READONLY

        ENTRY
 ; Enabling IRQ Interrupt
        MRS r1, cpsr
        BIC r1, r1, #0x80
        MSR cpsr_c, r1
; Enabling FIQ Interrupt
        MRS r1, cpsr
        BIC r1, r1, #0x40
        MSR cpsr_c, r1
; Disabling IRQ Interrupt
        MRS r1, cpsr
        ORR r1, r1, #0x80
        MSR cpsr_c, r1
; Disabling FIQ Interrupt
        MRS r1, cpsr
        ORR r1, r1, #0x40
        MSR cpsr_c, r1
B1      B    B1
        END
```