The 8051 Micro-Controller 32 bit Multiplication using Assembly Language

Mitali K. Dhrangadhria, Kuldeep B. Shukla, Hetal N. Rao

Abstract— Among, the lot many of microcontroller, 8051 is one of the most popular 8-bit microcontroller. Due to it can address 128kByte of external memory and has a basic instruction time of 1 microsecond. Assembly language is the language, mixture of machine level and higher level programming language called middle language. Thus the list of specific instructions selected from those allowed by the microcontroller manufacturer and organized to control operation constitute computer software. Here we are using multiplexing instruction for two 32-Bit multi plication operation along with other necessary instruction set. This two 32-Bit data will result in (32+32) 64-Bit answer.

Keywords— Microcontroller, Programming Language, Memory, Assembler, Cross Assembler, Register, Register Bank, PSW

I. INTRODUCTION

Each of the automation needs programming according to the type of automation device. In microcontroller C++, ORACAL, JAVA, etc. type of programming language acceptable using proper assembler and cross assembler. But have the complexity and more memory space where as assembly language needs less memory space & any of the program can be generated using few steps only.

Here, two-32Bit data is selected, where any of the data can be used by user first data and second data containing 32-Bit each means number of four register will be present to have the data entered for multiplication purpose .i.e. R0, R1, R2, R3. Similarly for second data four register will be present. R4, R5, R6, R7. [1]

Thus the total one bank of register will be requisite, just to enter input data at user side. Each of second data register will be multiplied with all the register of first data registers as like a simple multiplication which we use to do in primary school days. In real time scenario when more than are performed as a multiplication then it produces a little bit complexity in the assembly language. Hence, this complexity inspires to create thirty two bit multiplication and apply into the real time world for progressive performance. [1] [2]

II. EXPERIMENTAL MATERIALS

Register Bank second is selected to enter the input data. For the multiplication purpose here, R0, R1, R2, and R3 is first data and R4, R5, R6, and R7 is second data.

The multiplication steps can be performed by dividing the whole program in 4 sections with different 4 steps in each to get the final 64-Bit result. The total number of steps will be 16 steps.

Manuscript received on August, 2013.

Mitali K. Dhrangadhria, Student of final year B.E. E.C., SAL Engineering college, Ahmedabad, Gujarat, India.

Kuldeep B. Shukla, Student of final year M.E. E.C., SAL Engineering college, Ahmedabad, Gujarat, India.

Hetal N. Rao, Student of final year M.E. E.C., RK University, Rajkot, Gujarat, India.

Those are shown,

R0 R1 R2 R3
*R4 R5 R6 R7

A. R3 Performed as a Multiplication:

a) (R3 X R7)
b) (R3 X R6)
c) (R3 X R5)
d) (R3 X R4)

TABLE: A.1.
R3 X R7 AND R6 MULTIPLICATION

R3 X R7	R3 X R6
MOV PSW,#10H	MOV PSW,#10H
MOV A,R3	MOV A,R3
MOV B,R7	MOV B,R6
MOV PSW,#00H	MOV PSW,#00H
MUL AB	MUL AB
MOV R0,A	ADD A,R1
MOV A,B	MOV R1,A
MOV R1,A	MOV A,B
	ADDC A,#00H
	MOV R2,A
	MOV A,#00H
	ADDC A,#00H
	MOV R3,A

TABLE: A.2.
R3 X R5 AND R4 MULTIPLICATION

R3 X R5	R3 X R4
MOV PSW,#10H	MOV PSW,#10H
MOV A,R3	MOV A,R3
MOV B,R5	MOV B,R4 MOV
MOV PSW,#00H	PSW,#00H
MUL AB	MUL AB
ADD A,R2	ADD A,R3
MOV R2,A	MOV R3,A
ADDC B,R3	MOV A,B
MOV A,B	ADDC A,R4
MOV R3,A	MOV R4,A
MOV A,#00H	MOV A,#00H
ADDC A,#00H	ADDC A,#00H
MOV R4,A	MOV R5,A

User spending two register banks (bank 0 and 1) where bank 1 is used to enter data from the user side, as per occupation of maximum (32 bit) data needs 4 register to store one data. For entering the data to R0-R7 register of bank 2 and this data will be engaged first 4 register for 1st data and remaining 4 register for second data. [3]

Each of the data register will be sequentially multiplied by least significant resister by transferring there contain into A & B for the multiplication purpose.

The result of multiplication will be appear on A and B register as lower and higher byte respectively where lower byte will be stand in R0, the first register of the register bank 0 and higher byte will be added to the 2nd register of the second data for the next multiplication in the sequence. Similarly R3 & R7 of bank 1 is transferred to A & B respectively and multiplied the result will be in A & B. So, a containing lower byte would be transferred to the R0 register of bank 0. Such as same process will be repeated in the multiplication of R3 X R5 and R3 X R4, where in R3 X R4 operation due to R4 is the most significant register the lower byte will be stored in R4 the next register of register bank 0.

B. R2 Performed as a Multiplication:

a) (R2 X R7)
b) (R2 X R6)
c) (R2 X R5)
d) (R2 X R4)

TABLE: B.1. R2 X R7 AND R6 MULTIPLICATION

R2 X R7	R2 X R6
MOV PSW,#10H	MOV PSW,#10H
MOV A,R2	MOV A,R2
MOV B,R7	MOV B,R6
MOV PSW,#00H	MOV PSW,#00H
MUL AB	MUL AB
ADD A,R1	ADD A,R2
MOV R1,A	MOV R2,A
MOV A,B	MOV A,B
ADDC A,#00H	ADDC A,R3
MOV R2,A	MOV R3,A
MOV A,#00H	MOV A,#00H
ADDC A,#00H	ADDC A,#00H
MOV R3,A	MOV R4,A

TABLE: B.1. R2 X R5 AND R4 MULTIPLICATION

R2 X R5	R2 X R4
MOV PSW,#10H	MOV PSW,#10H
MOV A,R2	MOV A,R2
MOV B,R5	MOV B,R4
MOV PSW,#00H	MOV PSW,#00H
MUL AB	MUL AB
ADD A,R3	ADD A,R4
MOV R3,A	MOV R4,A
MOV A,B	MOV A,B
ADDC A,R4	ADDC A,R5
MOV R4,A	MOV R5,A
MOV A,#00H	MOV A,#00H
ADDC A,#00H	ADDC A,#00H
MOV R5,A	MOV R6,A

Process will be little bit modified due to the same operation will be about to do with the additional operation of multiplication result with the register value where the answer is to store.

As per operator rule in assembly language the R2 X R7 will be multiplied using A & B register and the resultant lower byte is now not ready to store due to the destination register R1 is already containing some resultant value of before operation that it done.

Thus, this will need to add lower resultant value with the same destination resister R1, and the answer of that addition will be further stored in register R1. And then again the above frequent process the higher byte will be added in the next register of 2nd data R6 for R2 X R6 operation.

Finally, R2 X R6; R2 X R5 and R2 X R4 will be operated for the result where R4 is the most significant register so need not to move its higher resultant byte to the next register. And it will move that byte to the next register of bank 0 in addition with the value of R5 resister which is the destination register as well containing the value of above multiplication result for the safe side if there may carry is generated then that carry will be converted into 8 bit and stored to the next register of bank 0.

C. R1 and R0 Performed as a Multiplication:

a) (Rx X R7)
b) (Rx X R6)
c) (Rx X R5)
d) (Rx X R4)

Where x=1 and 0, First R1 multiply sequentially then R0 with R7, R6, R5, R4

TABLE: C.1.R1 X R7 AND R6 MULTIPLICATION

R1 X R7	R1 X R6
MOV PSW,#10H	MOV PSW,#10H
MOV A,R1	MOV A,R1
MOV B,R7	MOV B,R6
MOV PSW,#00H	MOV PSW,#00H
MUL AB	MUL AB
ADD A,R2	ADD A,R3
MOV R2,A	MOV R3,A
MOV A,B	MOV A,B
ADDC A,R3	ADDC A,R4
MOV R3,A	MOV R4,A
MOV A,#00H	MOV A,#00H
ADDC A,#00H	ADDC A,#00H
MOV R4,A	MOV R5,A

Table C.1. pointing that if the previous carry generated at the end of the multiplication of register second to register four then it will add this carry at the end of the performance of the register one with register seven.

Then processed to the multiplication of register one to register six.

TABLE: C.2. R1 X R5 AND R4 MULTIPLICATION

R1 X R5	R1 X R4
MOV PSW,#10H	MOV PSW,#10H
MOV A,R1	MOV A,R1
MOV B,R5	MOV B,R4
MOV PSW,#00H	MOV PSW,#00H
MUL AB	MUL AB
ADD A,R4	ADD A,R5
MOV R4,A	MOV R5,A
MOV A,B	MOV A,B
ADDC A,R5	ADDC A,R6
MOV R5,A	MOV R6,A
MOV A,#00H	MOV A,#00H
ADDC A,#00H	ADDC A,#00H
MOV R6,A	MOV R7,A

TABLE: C.3. R0 X R7 AND R6 MULTIPLICATION

R0 X R7	R0 X R6

MOV PSW,#10H
MOV A,R0
MOV B,R7
MOV PSW,#00H
MUL AB
ADD A,R3
MOV R3,A
MOV A,B
ADDC A,R4
MOV R4,A
MOV A,#00H
ADDC A,#00H
MOV R5,A
	MOV PSW,#10H
	MOV A,R0
	MOV B,R6
	MOV PSW,#00H
	MUL AB
	ADD A,R4
	MOV R4,A
	MOV A,B
	ADDC A,R5
	MOV R5,A
	MOV A,#00H
	ADDC A,#00H
	MOV R6,A

TABLE: C.3. R0 X R5 AND R4 MULTIPLICATION

R0 X R5	R0 X R4

MOV PSW,#10H
MOV A,R0
MOV B,R5
MOV PSW,#00H
MUL AB
ADD A,R5
MOV R5,A
MOV A,B
ADDC A,R6
MOV R6,A
MOV A,#00H
ADDC A,#00H
MOV R7,A
	MOV PSW,#10H
	MOV A,R0
	MOV B,R4
	MOV PSW,#00H
	MUL AB
	ADD A,R5
	MOV R5,A
	MOV A,B
	ADDC A,R6
	MOV R7,A
	MOV A,#00H
	ADDC A,#00

As per above discussion all the operations that the same will be followed by R1 when it has been multiplied by R7, R6, R5, R4 register bank successfully more over when the register R0 is the multiplied by R7, R6, R5, R4 the only change in last operation is that most significant register will contain one more step that is; if the last addition of multiply resultant higher value with destination register R7 value is generating carry then this carry must concerned with the 8 bit value and stored in accumulator.

Thus, all the resultant value is sequentially stored in bank 0 by their registers including accumulator and then transfer all them sequentially into memory location for the conventioneers of the resultant display where the most significant bit of the answer will be addressed first in the memory location and then starting from R7 to R0 the rest of data movement will be take place. The result and the input data that occupied in the program can be easily understand by the any user as seen in the "KEIL µ-vision program". [5] [6]

III. IMPLEMENTATION

The flow of data manipulation is as multiply first register of first data sequentially with each register of second data, multiply second register of first data sequentially with each register of second data, multiply third register of first data sequentially with each register of second data, and multiply forth registers of first data sequentially with each register of second data using,

"MUL AB" instruction

Whereas result will be stored and transformation of any immediate data is done using,

"MOV destination, source" instruction

, external memory interfacing is done using,

"MOVX destination, source" instruction

And simple addition operation and addition with carry generated in the previous operation is done by using, respectively

ADD destination, source
ADDC destination, source

In addition to because of only eight registers are there in one register bank and we need the same for data feeding as well as in result storing processes so that every time at data fetch and manipulation of data and data storing process need to select it, we are here using PSW (program status word), combination of FLAG Registers named Carry Flag, Auxiliary Flag, User Define Flag, Bank Selection Flag, Over Flow Flag, and Parity Flag, as shown below, [4]

"MOV PSW, # (No. of bank that is to select)" instruction

D7	D6	D5	D4	D3	D2	D1	D0
C	AC	F0	RS1	RS0	OV	-	P
Carry Flag	Auxiliary Flag	User Define Flag	Bank Selection Flag		Over Flow Flag		Parity Flag

Thus the actual process performs the following way to achieve the result of the multiplication operation.

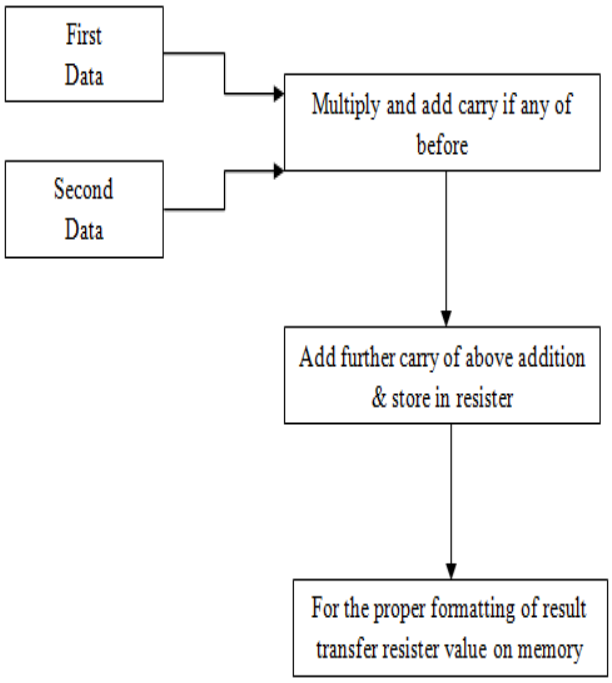

Figure 1: Flow chart of multiplication

IV. RESULT

This table will helps more to understand the process of multiplication which is divided into sections.

TABLE: D.1.
RESULT IN TERMS OF MULTIPLICATION DATA

						R4	R5	R6	R7
						R0	R1	R2	R3
Section 1				R5	R4	R3	R2	R1	R0
Section 2			R6	R5	R4	R3	R2	R1	-
Section 3		R7	R6	R5	R4	R3	R2	-	-
Section 4	A	R7	R6	R5	R4	R3	-	-	-

RESULT	M9	M8	M7	M6	M5	M4	M3	M2	M1

Each section is the result of multiplication of two input data register as well as addition of above registers in the same column that was manipulated before. Thus the result is saved in the external memory location sequentially. [7]

V. OUTPUT

Where PSW=#10H, R0=#00H, R1=0ABH, R2=#38H, R3=#82H, R4=#01H, R5=#51H, R6=#67H, R7= #0D3H.

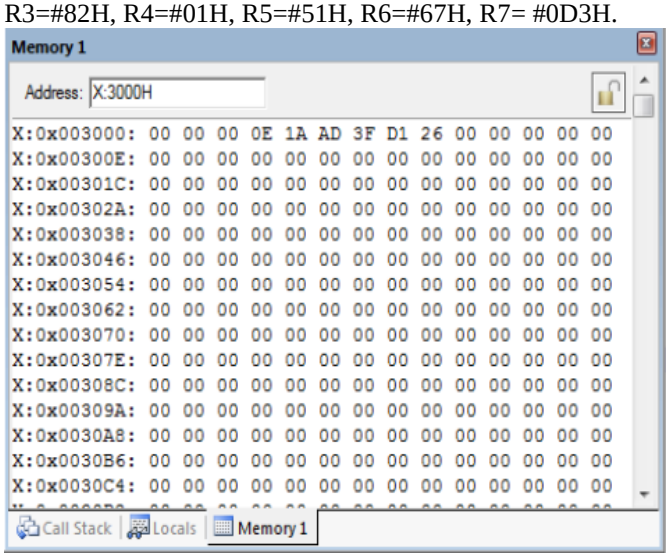

Figure 2: Output in µ-vision KEIL

VI. CONCLUSITION

The more we interact with the assembly language, the better we be familiar with internal architecture of microcontroller as well as with less memory little complex but exact solution can be achieved. Here any of the data can be used to get multiplied with, due to the program generalization. The way of understanding the path of this assembly language program is very easy and little lengthy programming while arranging it. This program facilitates as to multiply data between 8-Bit to 32-Bit of the data that user is comfortable without any modification in the same.

REFERENCES

1. Muhammad Ali & Janice Gillispie Mazidi and Rolin D. Mckinlay. "The 8051 Micro Controller and Embadded System using assembly and C", PHI Publication, 2nd Edition 2008-2009, PP No: 115-141, PP No: 153-178 PP No: 183-195.
2. Kenneth Ayala, "The 8051 Microcontroller", Cangage learing products; Canada by Nelson Education, 3rd Edition 2010, PP No: 131-144, PP No: 152-161, PP No: 170-183, PP No: 190-207.
3. Chris Braithwaite, Fred Cowan, Hassanprachzadeh, "8051 Microcontroller And Application based Introduction", 1st Edition, ELSEVIER, 2004, PP No: 38-64, 201-225.
4. Subrata Ghoshal, "Microcontroller: Internals, Instructions, Programming and Interfacing", Pearson Education, 2010.
5. Davies J H, "MICROCONTROLLER BASICS", ELSEVIER, 2011. Burkhard Kainka, "Microcontroller Basics", Tech Publications, 2006. Sampath K. Venkatesh "Microcontroller", S. K. Kataria & Sons.