

ASSIGNMENT NO. 4

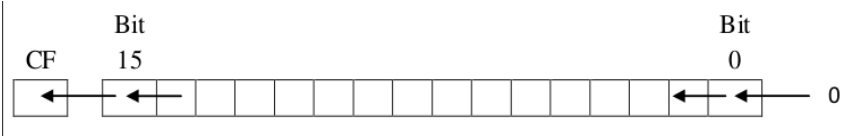
AIM :- Write X86/64 ALP to perform multiplication of two 8-bit hexadecimal numbers.
Use successive addition and add and shift method. (use of 64-bit registers is expected)

APPARATUS :

- Core 2 duo/i3/i5/i7 - 64bit processor
- OS – ubuntu 32bit/64bit OS
- Assembler used –nasm (the netwide assembler)
- Editor Used – gedit

THEORY :

SHL / SAL – Shift Left / Shift Arithmetic Left

Description	Shift Left
Mnemonic	SAL/SHL destination, count
Algorithm	Shift all bits left, the bit that goes off is set to CF and zero bit is inserted in the right most position.
Operation	<p>CF ← MSB ← LSB ← 0</p>  <ul style="list-style-type: none"> • SAL/SHL are two mnemonics for the same instruction. This instruction shifts each bit of the specifies destination, some number of bit positions to the left. As left bit is shifted out of the LSB position, 0 is put in the LSB position. The MSB will be shifted into CF. • CL is default register used for shift instruction when count is greater than 1.

Flags	OF, CF, AF, PF and SF are affected
Example	SHL AX, 01 ; This instruction shift AX by 1 bit to the left

ALGORITHMS:-**1. Accept numbers from user.**

1. Declare

Section .data of proper string messages.

Section .bss of proper variables.

Declare various macros e.g. print, read & exit macros.

Section .text as starting point of code segment.

2. Prompt message on screen & accept 2 numbers from user using accept_16 procedure.

3. Display menu on screen & accept choice from user.

Successive Addition Method

4. According to choice, display Result by Successive Addition Method message on screen.

5. Clear the contents of RAX & RDX. These are used to store final answer.

6. Load 1 st number in BX.

7. Load 2 nd number in CX.

8. (Perform $RAX + RBX = RDX$: RAX.))

Perform addition of RAX & 1 st no. i.e. RBX.

If carry then increment RDX

Decrement CX by 1.

Repeat these steps until CX becomes. Zero

9. Store result in ansL and ansH from AX and DX respectively.

10. Call display_16 procedure to display ansH on screen.

11. Call display_16 procedure to display ansL on screen.

12. Stop.

CONCLUSION:
