ROLL. NO - 2
Aaron Philip
1032210163

MAIOT ASSIGNMENT 7

AARON PHILIP
1032210163

PROBLEM STATEMENT :
Write X86/64 ALP to add an array of N hexadecimal numbers

OBJECTIVES :
① Understand the concept of unpacked and packed HEX number & the need of packing the accepted number from user
② Repetitive addition

THEORY :

Explain unpacked & packed number with an example (HEX)
In computer science, hexadecimal (or simply "hex") numbers are commonly used to represent binary data in a more human-readable format.
Packed hex numbers are a string of hexadecimal digits that represent a single value.
Eg) The packed hex "0x1234" represents 4660
Unpacked hex numbers are used to represent multiple values packed together in a single string of hexadecimal digits.
Eg) "0x12 0x34) represents 4660 & 4660

Why is packing required ?

Packing of hexadecimal numbers is required in computing to save memory space. computers store & process data in binary form, which can be efficiently represented using hexadecimal numbers. By packing multiple hexadecimal numbers into a single value, the data can be stored in a more compact form, which reduces the amount of memory required to store the data.

INSTRUCTIONS

1. MOV
- used to transfer data from one memory location to the other

2. ADD
- used to add two operands

3. SUB
- used to subtract two operands

4. CMP
- used to compare two operands

5. JMP
- used to jump to a specific memory location

★ In 8086, a syscall is a request made by a user program ~~to~~ which transfers control to the operating system's system call handler. It is used to perform operations such as ~~and~~ read & write.

Algorithm Implementation

Packing 2 digit Hex

```
mov rsi, NumArray
mov ax, 00h ; initialize accumulator to 0
mov bx, 00h ; initialize b-register to 0
mov cx, 5
```

Array addition

```
up2: mov bl, byte [rsi] ; loading content of rsi to bl register
add ax, bx ; adding contents of b register to a register
jnc skip ; go to skip label if no carry
inc ah ; if carry the increment accumulator
skip: inc rsi ; go to next number in array
dec cx ; decrement the counter
jnz up2 ; loop back to the up2 label
```

displaying the result

```
mov word [Result], ax
mov bp, 4
up: rol ax, 4 ; rotate
mov bx, ax
and ax, 0Fh
cmp al, 09
jbe down
add 07h ; add 37 to get ASCII value of alphabets
else add 30 for numbers
```

Platform
Editor - gedit
Assembler = NASM
Linker - LD

Input
Array elements

Output
Sum of array elements

Conclusion
Understood implementation of adding an array of numbers in 8086

FAQs

Q1) Explain flag register of 8086 with a neat diagram.
A1) The flag register is a special purpose register. Depending upon the value of the result after any arithmetic & logical operations, the flag bit becomes set (1) or reset (0)

1) STATUS flags
1. Sign flag (S)
2. Zero flag (Z)
3. Auxilliary Carry flag (AC)
4. Parity flag (P)
5. Carry flag (CY)
6. Overflow flag (O).

CONTROL FLAGS
Directional flag (D)
Interrupt flag (I)
Trap flag (T)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

8086 Register format

State the difference between ROL & SHL instructions.
SHL (Shift left)
The shift left instruction performs a left shift on the destinations operand, filling the lowest bit with 0. The highest bit is moved into the carry flag.
eg) SHL destination, bits shifted

ROL
The ROL instruction shifts each bit to the left, with the highest bit copied into the carry flag into the lowest bit.

★ Shifting means to move bits inside an operand

CODE :

```
%macro rw 4
mov rax,%1
mov rdi,%2
mov rsi,%3
mov rdx,%4
syscall
%endmacro



section .data
Num_Array db 11h,12h,13h,14h,15h

msg db "Result of array addition is:",10
msgLen equ $-msg

section .bss
Result resw 1 ; 2bytes=16bit=1word
temp resw 2
temp1 resb 1

section .text

global _start

_start: ; label - indicates that execution of code starts from this label

mov rsi,Num_Array
mov ax,00h ; initializing to 0
mov bx,0h ; initializing to 0
mov cx,5 ; initializing counter to 5 to decrement it later

up2: mov bl,byte[rsi] ; loading content of rsi to the bl register, []- called as mem addressing mode
indicates value at that address
add ax,bx ; adding contents of b register to accumulator which is initially 0
jnc skip ; go to the skip label if there is no carry
inc ah ; if there is a carry then increment the accumulator

skip: inc rsi ; go to the next number in the array
dec cx ; decrement the counter
jnz up2 ; loop back to the up2 label
```

```
mov word[Result],ax
mov bp,4
up: rol ax,4 ; after rotation we will get 05F0
mov bx,ax ; ax=005F therefore bx=005F
and ax,0Fh ; andding with 000F (05F0 & 00F0 = 0000)
cmp al,09
jbe down
add al,07h

down: add al,30h ; add 30
mov byte[temp],al ; move al value to temp
rw 1,1,temp,1 ; on output screen we will first get 0
mov ax,bx ; load 005F into ax again (it had become 05F0)
dec bp
jnz up
rw 60,0,0,0 ; if we dont write this termination code we will get a segmentation fault error
```

OUTPUT :