Page No.
Date

Devanshu Surana
1032210755, PC-23
Batch C1

MAIoT Lab Assignment-7

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 and the need of packing the accepted number from user.
2. Repetitive addition

Theory:
- Unpacked and packed numbers:

In case of unpacked numbers, the machine stores each four-bit BCD group that corresponds to a decimal digit in a separate register. If the registers are 8 bit or more than the register space is wasted. Hence packing is done to utilize the space fully and avoid wastage of memory. In packing two BCD ~~numbers~~ digits are kept in a single eight bit register. To combine the two BCD digits into one 8 bit register, the top register number must be moved 4 times to the left and then the upper and lower register numbers must be added.

Eg 98 = 10011.000

Unpacking the BCD number is separating each BCD digit. 98 can be separated as 09 and 08. So we can say 10011000 [98] is packed and 00001001 [09] & 00001000 [08] are unpacked.

Page No.
Date

Algorithm / Implementation :

1) Packing of 2 digit Hex number :
1. Take input of 2 digit Hex no.
2. Convert the first digit of the hexadecimal number from ASCII to its equivalent hexadecimal value.
3. Shift the packed hexadecimal number 4 bits to left.
4. Add converted hexadecimal value to packed hexadecimal number
5. Repeat step 2 to step 4 for second digit of Hex no.
6. The two digit Hex number is now packed into single 8-bit binary number.

Addition :
1. Declare array of 5 Hex numbers.
2. Initialize pointer to array, counter = 5 & result = 0
3. Read byte from array.
4. Perform 16 bit addition, ex. AX + BX . If carry is generated increment Alt.
5. Increment pointer
6. Decrement counter
7. Repeat steps 3 to 6 till counters becomes 0.
8. Store sum in "result" variable.

Display :
1. Load result in AX.
2. Initialize base pointer = 4
3. Rotate contents of AX to left by 4 ~~digits~~. bits.
4. Move contents of AX into BX.
5. AND of AX and OFH
6. Convert contents of AL into equivalent ASCII values.
7. Store ASCII representation into 'temp'.

8. Print temp
9. Move contents of BX back to AX.
10. Repeat step 3 to step 9 until bp = 0.

Platform:
Editor: Gedit, a gnu editor.
Assembler: NASM (Netwide Assembler
Linker: LD, a GNU Linker.

Input:- i) 01, 02, 03, 04, 05
ii) 0F, 0F, 0F, 0F, 0F

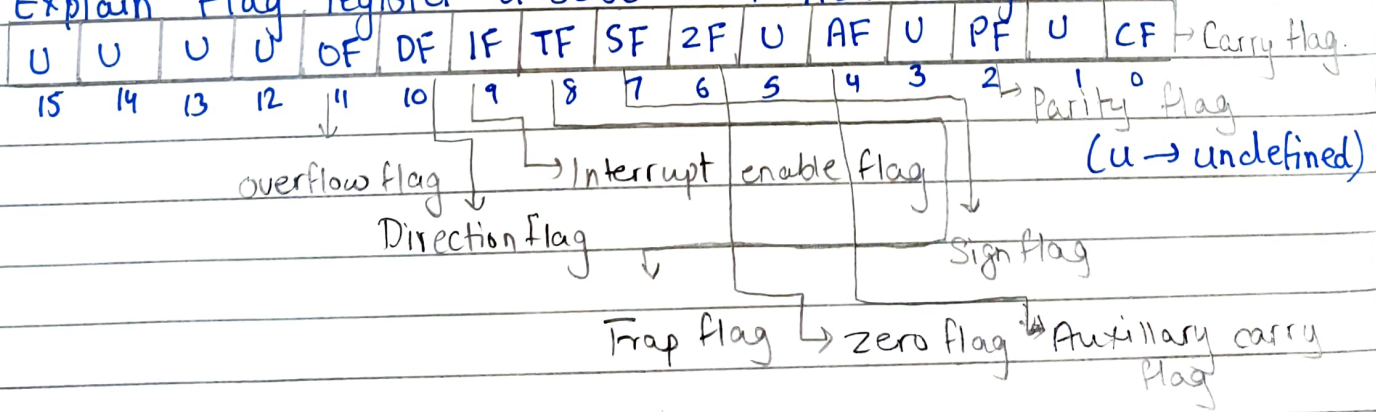
Output:- i) 000F
ii) 004B

Conclusion: We have learnt and understood the concept of packing and unpacking of Hex numbers using the 'ROL' and 'AND' instructions. Also, we have illustrated on ALP to add an array of N Hexadecimal numbers and display its result (Hardcoded and user input both).

FAQ's.

1. Explain Flag register of 8086 with neat diagram.

→



The flag register in 8086 microprocessor is a 16-bit register that stores various flags. We can divide flag bit into 2 sections :-

1. Status flags.
- Carry flag (CF) - set if there is carry out of MSB of result.
- Parity flag (PF) - set if no. of set bit is even.
- Auxillary Carry flag (AF) - set if there is carry out of bit 3.
- Zero flag (ZF) - set if the result is zero.
- Sign flag (SF) - set if MSB of result is ~~zero~~ set.
- Overflow flag (OF) - set if result of signed operation is too large to fit in destination register.

2. Control flags.
- Direction flag (DF) - If set then string data is accessed from higher memory location to lower memory location. If reset (0) then string data is accessed from lower memory location to higher memory location
- Interrupt flag (IF) - If set, recognize interrupt and if reset (0), decline any interrupts.
- Trap flag : If set (1), CPU generates an interrupt.

2. State the difference between ROL and ~~AND~~ SHL instructions.

→ ROL instruction rotates the bits of a destination operand to the left, with the bit that is rotated out of MSB is shifted to LSB. While SHL instruction shifts the bits of destination operand to the left filling the LSB with '0'.
In other words, ROL retains the original value of the bits that are shifted out, while SHL instruction discards the original value of bits that are shifted out.

Page No.
Date

3. Why 30H/37H is subtracted from the number? Explain with example of each.

→ 30H and 37H are subtracted from a number to convert the number from ASCII representation to Hex no.

ASCII representation for 0-9 is 30H to 39H resp. while for uppercase A-F it is 41H to 46H resp.

For ex) i) If ASCII of is 34 H then equivalent hex no will be:

$$34 - 30 = 04$$

ii) If ASCII reperesentation is 45H then equivalent hex no will be:

$$45H - 37H = 0E$$

Name: Devanshu Surana **Roll No.:** 23
Panel: C
Batch: C1

CODE:

MAIoT Assignment 07

```
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 temp resw 2 temp1 resb 1

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

section .text
global _start
_start:
mov rsi,Num_Array mov ax,00h

mov bx,0h
mov cx,5
up2: mov bl, byte[rsi] add ax,bx
jnc skip
inc ah
skip:
inc rsi
dec cx
jnz up2
mov word[result],ax mov ax,word[result] mov bp,4
up: rol ax,4
mov bx,ax
and ax,0Fh
cmp al,09
jbe down
add al,07h
down:Add al , 30h mov byte[temp],al

rw 1,1,temp,1 mov ax,bx dec bp
jnz up
```

rw 60,0,0,0

OUTPUT: