MIT World Peace University

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

Write an assembly language program (ALP) to implement addition and subtraction of 8-bit numbers (Using user input, macro and procedure)

Practical Report
Assignment 7

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

December 3, 2022

Contents

1 Objectives

Write an assembly language program (ALP) to implement addition and subtraction of 8-bit numbers (Using user input, macro and procedure)

- To understand assembly language programming.
- To study instruction set of 8086.

2 Platform Used

Operating System : Arch Linux
Editor – Visual Studio Code
Assembler – NASM (Netwide Assembler)
LINKER – LD, a GNU linker

3 Theory:

3.1 Assembly language program basic structure:

A ALP is series of statements which are either assembly language instruction such as ADD and MOV or statements called directives. A program language instruction consists of following 4 fields.

```
[Label] mnemonic [operands] [;comment]
```

A square bracket([]) indicates that the field is optional
A Label field allows the program to refer to a line of code by name. The label Fields cannot exceed a certain no.of characters.
The mnemonics and operands fields together perform the real work of the program and accomplish the tasks statements like ADD A,C and MOV C,#68 where ADD adn MOV are the mnemonics, which produce opcodes, "AC" and "C,#68" are operands. These two fields could contain directives. Directives do not generates machine code and are used only by the assembler, whereas instructions are translated into machine code for the CPU to execute.
The comment fields begins with a semicolon which is a comment indicator
Notice the label "HERE" in the program. Any lable which refers to an instruction should be followed by a colour.

3.2 System calls to read, write and Exit.

Reading From a file:

- Put the system call sys_oread in EAX register
- Put file descriptors in the EBX register.
- Put the pointer to input buffer in ECX register.
- Put the pointer buffer size i.e. number of bytes to read in EDX register.

Writing to a file:

- Put system call sys_write() in EAX register
- Put file descriptive in EBX register
- Put buffer size i.e. the number of bytes to write in EDX.

Exit From File:

- Put system call sys_exit in EAX register.

3.3 Describe the instruction used (e.g MOV, ADD, SUB)

1. INC - used for incrementing and operand by one works on single operand that can be a memory or in a register
2. DEC - used for documenting an operand by one work on a single operand.
3. ADD and SUB - used for performing simple additional subtraction of binary data in byte, word adn double word size.
4. MUL/IMUL - used for multiplying data both affect the carry and overflow flag.

4 Code

```
; display 2 digit hexx numbers
section .data
    msg db "Addition of 2 numbers ", 10
    msglen equ $-msg
    num1 db 3AH ; the number to be printed. h is for hex
    num2 db 22H ; the number to be printed. h is for hex

section .bss
; temp data assignment
    sum resb 1
    temp resb 1

section .text
global _start

_start:
    ; printing the first message
    mov rax, 1
    mov rdi, 1
    mov rsi, msg
    mov rdx, msglen
    syscall

    ; assign one byte of num1 to al
    mov al, byte[num1]
    add al, byte[num2]
    ;assign the value of al to sum
    ;mov byte[sum], al
    ;assign 2 to bp
    mov bp, 2; bp = 2
    ; shift all binary bits 4 times to right, this flips the nibbles
    ; so rn its 0010 0011 after flipping it becomes 0011 0010
```

```
33
34 up:rol al, 4
35     ; assign al to bl
36     mov bl, al; al = 32H
37     ; and with 0FH, so 0000 1111 anded with 0000 0010 so youll end up with the
       0010
38     and al, 0FH ; al = 02H at this point
39     ; this would trigger some flag
40     cmp al, 09
41     ; goto down label if the above cmp statement gives less than or equal to
42     jbe down
43     add al, 07H
44
45 down: Add al, 30H; al = 32H
46     mov byte[temp], al
47     mov rax, 1
48     mov rdi, 1
49     mov rsi, temp
50     mov rdx, 1
51     syscall
52     mov al, bl ; bl = 23H
53     dec bp ; this is the loop register which we decrement if its 0 then we stop
       the loop
54     jnz up; now go to up again, and this time you would use bls value to al coz
       you would rotate it again.
55
56 mov rax, 60
57 mov rdi, 0
58 syscall
```

5 Output

```
Addition of 2 numbers
5C
```

6 Conclusion

Thus learnt how to add numbers using Assembly Language and Display name.

7 FAQs:

7.1 Explain assembler directives. List the assembler directives in your program.

These are the statements that direct the assembles to do something. As the name sya it directs the assembles to do a task. They are classified into the following categories based on the function performed by them.

- CODE - This assembles directives indicates the beginning of the code segment.
- Data - Indicates beginning of data segment
- Mode - Is used for selecting a standard memory. model for the assembly program.
- Stack - The directives is used for displaying the stack.

7.2 Explain why 30H 137H is added to convert the digit to ASCII?

30H is the ASCII code for a digit '0' then 31H,32H,39H, corresponds 1,2,3,.....,9. 41H is the ASCII code for a letter 'A'. As in hexadecimal a value of 10 would be responded represented by the 'A'. Basically that function segmented in two possible additions convert the internal value into a printable character than properly represents value.

7.3 Define Macro and Procedure:

- Macro - A macro in computer science is a set of rules or programmable patience which decrypts a specific sequence of output.
- Procedure - Procedure are used for a large set of rules. They help make a large program more readable. It indicates a set of instructions that executes a particular task.