

Lab Homework Week 7 Report

Group 66: 102403015 程祥恩、102403016 邱威穎、102403020 曾子軒

Objective:

To understand how to write simple Assembly program and program structure. Use the course content to learn data in memory storage and how to use MUL instruction. Make a program that stored multiplication table from 1*1 to 9*9 as shown as in the memory.

```
exercise7.asm
1  TITLE  exercise7[exercise7.asm]
2  INCLUDE Irvine32.inc
3
4  main EQU start@0
5
6  .data
7  ninenine byte 81 DUP(?)
8
9  .code
10 start@0 PROC
11
12     mov esi, OFFSET ninenine
13     mov al, 0
14     mov bl, 0
15     mov ecx, 9 ; set outer loop count
16
17 L1: ; begin the outer loop
18     push ecx ; save outer loop count
19     inc al
20     mov ecx, 9 ; set inner loop count
21     mov bl, 0
22 L2: ; begin the inner loop
23     push ax
24     inc bl
25     mul bl ; ax = al*bl
26     mov [esi], ax
27     inc esi
28     pop ax;
29
30     loop L2 ; repeat the inner loop
31
32     pop ecx ; restore outer loop count
33     loop L1 ; repeat the outer loop
34
35 exit
36 start@0 ENDP
37 END start@0
```

The screenshot shows two windows from a debugger. The left window, titled 'D:\...\exercise7.asm', displays the assembly code for a program that calculates a 9x9 multiplication table. The code uses nested loops (L1 and L2) to iterate through the table, pushing and popping registers to maintain state. The right window, titled 'Memory(&ninenine)', shows a memory dump starting at address 0x00404000, displaying hexadecimal values and their corresponding ASCII representations.

```

D:\...\exercise7.asm
TITLE exercise7[exercise7.asm]
INCLUDE Irvine32.inc

main EQU start@0

.data
ninenine byte 81 DUP(?)

.code
start@0 PROC

    mov esi, OFFSET ninenine
    mov al, 0
    mov bl, 0
    mov ecx, 9 ; set outer loop count

L1: ; begin the outer loop
    push ecx ; save outer loop count
    inc al
    mov ecx, 9 ; set inner loop count
    mov bl, 0

L2: ; begin the inner loop
    push ax
    inc bl
    mul bl ; ax = al*bl
    mov [esi], ax
    inc esi
    pop ax

    loop L2 ; repeat the inner loop

    pop ecx ; restore outer loop count
    loop L1 ; repeat the outer loop

exit
start@0 ENDP
END start@0
  
```

Memory(&ninenine)

0x00404000	01	02	03	04	05	06	07	08	09
0x00404009	02	04	06	08	0a	0c	0e	10	12
0x00404012	03	06	09	0c	0f	12	15	18	1b
0x0040401b	04	08	0c	10	14	18	1c	20	24
0x00404024	05	0a	0f	14	19	1e	23	28	2d
0x0040402d	06	0c	12	18	1e	24	2a	30	36
0x00404036	07	0e	15	1c	23	2a	31	38	3f
0x0040403f	08	10	18	20	28	30	38	40	48
0x00404048	09	12	1b	24	2d	36	3f	48	51
0x00404051	00	00	00	00	00	00	00	00	00
0x0040405a	00	00	00	00	00	00	00	00	00
0x00404063	00	00	00	00	00	00	00	00	00
0x0040406c	00	00	00	00	00	00	00	00	00
0x00404075	00	00	00	00	00	00	00	00	00
0x0040407e	00	00	00	00	00	00	00	00	00
0x00404087	00	00	00	00	00	00	00	00	00
0x00404090	00	00	00	00	00	00	00	00	00
0x00404099	00	00	00	00	00	00	00	00	00
0x004040a2	00	00	00	00	00	00	00	00	00
0x004040ab	00	00	00	00	00	00	00	00	00
0x004040b4	00	00	00	00	00	00	00	00	00
0x004040bd	00	00	00	00	00	00	00	00	00
0x004040c6	00	00	00	00	00	00	00	00	00
0x004040cf	00	00	00	00	00	00	00	00	00
0x004040d8	00	00	00	00	00	00	00	00	00
0x004040e1	00	00	00	00	00	00	00	00	00
0x004040ea	00	00	00	00	00	00	00	00	00
0x004040f3	00	00	00	00	00	00	00	00	00
0x004040fc	00	00	00	00	00	00	00	00	00
0x00404105	00	00	00	00	00	00	00	00	00
0x0040410e	00	00	00	00	00	00	00	00	00
0x00404117	00	00	00	00	00	00	00	00	00
0x00404120	00	00	00	00	00	00	00	00	00
0x00404129	00	00	00	00	00	00	00	00	00
0x00404132	00	00	00	00	00	00	00	00	00
0x0040413b	00	00	00	00	00	00	00	00	00
0x00404144	00	00	00	00	00	00	00	00	00
0x0040414d	00	00	00	00	00	00	00	00	00
0x00404156	00	00	00	00	00	00	00	00	00
0x0040415f	00	00	00	00	00	00	00	00	00

Explanation:

We set ecx to 9 in order to loop L1 for 9 times. In L1, we pushed ecx to preserve the count of L1, and then set ecx to 9 to prepare for the L2 loop. We also used inc al to increase multiplicand by one. In L2, we pushed ax in order to preserve from the change of al so that we could implemented the 9x9 table correctly. Finally, we used inc esi to make esi point to next byte. This way, we could finish 9x9 table eventually.

Review:

The assignment of today is to code a 9x9 multiplication table which gave us a chance to review the instructions of mul, loop, push, pop and esi pointer. We have done 9*1 to 9*9 in previous class which let us be able to refer to it. At first, we didn't know how to implement nested loop so we refer to the ppt of chapter 5. After that, we searched the concept of 9x9 table by assembly from Google and adjust the code: push and pop instructions, also we clarified the idea of FIFO in stack. Finally, we finished this lab assignment.