Assignment 4

Assignment goals:

The assignment helps understanding concepts and mechanisms in computer architecture and assembly language. The assignment is a preparation for final exam in assembly language and SPlab. Note: This assignment is to be done **SOLO**.

1. Three numbers are stored in memory locations 70h, 71h, and 72h, which should be given the names X, Y and Z. Write an assembly language program that finds the largest even number and stores it in 73h.

2. Consider the following 80X86 code segment.

```
00000000   E800000000          call    get_my_loc
                    get_my_loc:
00000005   59                  pop     ecx            ;message length
00000006   83C11C              add     ecx, msg1-get_my_loc
00000009   BA06000000          mov     edx,6          ;message to write
0000000E   BB01000000          mov     ebx,1          ;file descriptor (stdout)
00000013   B804000000          mov     eax,4          ;system call number
00000018   CD80                int     0x80           ;call kernel
                      next:
0000001A   B801000000          mov     eax,1          ;system call number (sys_exit)
0000001F   CD80                int     0x80           ;call kernel
00000021   7468616E6B20796F7521   msg1: db "thank you!"
```

   1. What is the result of running the code?
   2. Re-write the above code segment to a “non-zeros code” (The listing file of the new code would contain no zero byte).

3. Consider the following 80X86 code segment. You cannot assume anything about content of any register.

```
xor bh, bh
mov bl, 1
mov ecx, 32
L:   rcl edx, bl
     sbb bl, 0
     or edx, bl
     mov bx,1
     loop L, ecx;
```

   1. What is this code supposed to do?
   2. There is an error in the code. Add (at most) two code lines to fix the error.
   3. Re-write the above code for better runtime efficiency if possible.

4. List 5 different ways to add 1 to the EAX register on an 80X86 with exactly 1 80X86 instruction. Assume the following initial state.

eax = 1

5. List the shortest possible code for adding 5 to y, subtracting 2 from x, and adding 1 to z, which are defined as follows (consecutive locations):

```
   z db 1
   x dw 330
   y dw 7
```

Is there a shorter way to do that? How?

6. On Intel 80X86, suppose that AX = 0000000011010111, BX = 0000000001110010, CX = 0000000010111001, and Carry Flag = 1. What is the result of each of the following operations run in the given state (describe the state of the carry and overflow flags after execution of each)

a. call ECX
b. ADC AL, 0xF9
c. ADD AX, 0x003A
d. SUB BL, 0x73
e. JMP ECX

7. Write instruction sequences to perform some common set operations, for 80X86.
Each set is a subset of [1..16] is represented by corresponding bits in the register (e.g., AX=0100001000100101 represents {1,3,6,10,15}).
Use the following table. Each entry contains a single bit. The index into the table selects which bit is set (e.g., the value at index zero has bit zero set).

```
BitTbl dw 1, 2, 4, 8
       dw 10h, 20h, 40h, 80h
       dw 100h, 200h, 400h, 800h
       dw 1000h, 2000h, 4000h, 8000h
```

a. Delete - deletes the specified item from the set.

BX contains a value in the range 0..15.
AX contains a set.

b. Odd - Even sets the zero flag if AX contains a set of numbers that are all odd.

c. Member - Member clears the zero flag if BX is an element
of the AX set, it sets the zero flag otherwise.

BX contains a value in the range 0..15.
AX contains a set.

d. UnionSets - Union computes AX := AX union BX.

AX and BX contain the sets.

e. Intersection - Intersection computes AX := AX intersect BX.

AX and BX contain the sets.

f. Complement - Complement computes AX := - AX, that is, all elements in the set are removed, and all elements not in the set are added.

8. Codes and Hamming distances:

Given the following two 5-bit code words, we would like to extend this code to 4 code words of the same length (without changing the two given code words).

```
    | Code word
-------------
 00 | 10000
 11 | 11111
```

(a) What is the maximum hamming distance that we can get for the resulting code?
(b) How many errors can it detect?
(c) How many errors can it fix?
(d) How many erasures can it fix?
(e) Is it possible to change only a single bit in one of the two given code words above in order to extend it to 4 code words with better hamming distance? If yes, what is the needed change?

9. The following macros definitions are given

```
%define ctrl 0x1F &
%define param(a, b) ((a)+(a)*(b))
%xdefine c1 ctrl
%xdefine ctrl c1 0x02
```

Which result code line would be generated by assembler for the following source code line? Will it pass the assembler compilation?

```
mov byte [param(ctrl, ebx)], c1 'D'
```

10. Execute the position independent code from the lecture (use also gdb) :

    1. What are the addresses of main, my_func, my_pic_func, my_strict_pic_func, printf in each call to one of the functions (while running)?

    2. What would happen if we define “str1” at the beginning of section text?

    3. How can we use the “Functions” jump table in my_pic_func? Explain.

11. Given the following C code

```
z = foo(&x, y);
```

Given the following assembly code that implements foo using the C calling convention:

```
foo:
push ebp
mov ebp, esp
push ebx
mov ebx, [ebp+12]
mov eax, [ebp+8]
mov eax, [eax]
```

```
sub eax, ebx
pop ebx
ret
```

What is a return value of foo (using x and y to state the answer)?

12. Consider the following code for Motorola 68000 (comments state what each instruction does, according to the 68000 instruction manual (no, we did not teach you that machine, you should learn the relevant details on your own from the exercise). Note that the 68000 has 32-bit registers D0-D7 and A0-A7, where A7 is also the STACK POINTER.

```
F: MOVE.L   D0, -(A7)     ; D0 to memory - predecrement mode
   SUBQ.L   #1, D0        ; Subtract immediate - long (32 bit) operand
   BMI      N             ; Branch (jump) if result was negative
   JSR      F             ; CALL subroutine F (push PC then jump to F)
   SUB.L   (A7)+, D2      ; Signed subtract memory (postincrement)
                          ; with D2, result in D2
   RTS                    ; Return from procedure/subroutine
N: MOVE.L   (A7)+, D0     ; Move memory to D0 - postincrement mode
   MOVEQ.L  #-1, D2        ; Move immediate to D2
   RTS
```

What happens if we execute an instruction JSR F, with D0=0? D0=1? Other values of D0 (call that value k)?

13. (SPlab part) Consider the following hexedit display of an ELF file.

```
00000000   7F 45 4C 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010   02 00 03 00  01 00 00 00  62 80 04 08  34 00 00 00  ........b...4...
00000020   D0 00 00 00  00 00 00 00  34 00 20 00  01 00 28 00  ........4. ...(.
00000030   07 00 04 00  01 00 00 00  00 00 00 00  00 80 04 08  ................
00000040   00 80 04 08  9E 00 00 00  9E 00 00 00  05 00 00 00  ................
00000050   00 10 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000060   90 90 B8 04  00 00 00 BB  01 00 00 00  8B 0D 9A 80  ................
00000070   04 08 BA 08  00 00 00 CD  80 E8 01 00  00 00 90 B8  ................
00000080   01 00 00 00  BB 00 00 00  00 CD 80 00  FF FF FF FF  ................
00000090   41 68 6D 65  64 69 21 0A  00 00 90 80  04 08 00 2E  Ahmedi!.........
000000A0   73 79 6D 74  61 62 00 2E  73 74 72 74  61 62 00 2E  symtab..strtab..
000000B0   73 68 73 74  72 74 61 62  00 2E 74 65  78 74 00 2E  shstrtab..text..
000000C0   72 6F 64 61  74 61 00 53  79 72 69 61  00 00 00 00  rodata.Syria....
000000D0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000E0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000F0   00 00 00 00  00 00 00 00  1B 00 00 00  01 00 00 00  ................
00000100   06 00 00 00  60 80 04 08  60 00 00 00  2B 00 00 00  ....`...`...+...
00000110   00 00 00 00  00 00 00 00  10 00 00 00  00 00 00 00  ................
00000120   21 00 00 00  01 00 00 00  02 00 00 00  8C 80 04 08  !...............
00000130   8C 00 00 00  0D 00 00 00  00 00 00 00  00 00 00 00  ................
00000140   04 00 00 00  00 00 00 00  29 00 00 00  01 00 00 00  ........).......
00000150   02 00 00 00  99 80 04 08  99 00 00 00  05 00 00 00  ................
00000160   00 00 00 00  00 00 00 00  01 00 00 00  00 00 00 00  ................
00000170   11 00 00 00  03 00 00 00  00 00 00 00  00 00 00 00  ................
00000180   9E 00 00 00  2F 00 00 00  00 00 00 00  00 00 00 00  ..../...........
00000190   01 00 00 00  00 00 00 00  01 00 00 00  02 00 00 00  ................
000001A0   00 00 00 00  00 00 00 00  E8 01 00 00  E0 00 00 00  ................
000001B0   06 00 00 00  0A 00 00 00  04 00 00 00  10 00 00 00  ................
000001C0   09 00 00 00  03 00 00 00  00 00 00 00  00 00 00 00  ................
000001D0   C8 02 00 00  44 00 00 00  00 00 00 00  00 00 00 00  ....D...........
000001E0   01 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000001F0   00 00 00 00  00 00 00 00  00 00 00 00  60 80 04 08  ............`...
00000200   00 00 00 00  03 00 01 00  00 00 00 00  8C 80 04 08  ................
```

```
00000210   00 00 00 00  03 00 02 00  00 00 00 00  99 80 04 08  ................
00000220   00 00 00 00  03 00 03 00  01 00 00 00  00 00 00 00  ................
00000230   00 00 00 00  04 00 F1 FF  06 00 00 00  8C 80 04 08  ................
00000240   00 00 00 00  00 00 02 00  0D 00 00 00  90 80 04 08  ................
00000250   00 00 00 00  00 00 02 00  15 00 00 00  7F 80 04 08  ................
00000260   00 00 00 00  00 00 01 00  1A 00 00 00  99 80 04 08  ................
00000270   00 00 00 00  00 00 03 00  20 00 00 00  9A 80 04 08  ........ .......
00000280   00 00 00 00  00 00 03 00  25 00 00 00  62 80 04 08  ........%...b...
00000290   00 00 00 00  10 00 01 00  2C 00 00 00  9E 90 04 08  ........,.......
000002A0   00 00 00 00  10 00 F1 FF  38 00 00 00  9E 90 04 08  ........8.......
000002B0   00 00 00 00  10 00 F1 FF  3F 00 00 00  A0 90 04 08  ........?.......
000002C0   00 00 00 00  10 00 F1 FF  00 65 32 2E  73 00 54 75  .........e2.s.Tu
000002D0   72 6B 65 79  00 4C 65 62  61 6E 6F 6E  00 65 78 69  rkey.Lebanon.exi
000002E0   74 00 41 73  73 61 64 00  48 6F 6D 73  00 5F 73 74  t.Assad.Homs._st
000002F0   61 72 74 00  5F 5F 62 73  73 5F 73 74  61 72 74 00  art.__bss_start.
00000300   5F 65 64 61  74 61 00 5F  65 6E 64 00               _edata._end.
```

(a) How many section headers does it have?
(b) Is it an object file or an executable file?
(c) How many program headers does it have?
(d) If there are any program headers, what does the first program header do?
(e) If there are any section headers, at what offset is the section header table?
(f) What are the names of all the sections in this file, if any?

14. (SPlab part) What does the run of the following program print:

```
main() {
  int i = 3, pid;

  while(--i) {
    pid = fork();
    if(pid || (i&3))
      printf("NUKE %d!\n", i);
    }
 }
```

Is the answer unique?

What would happen if we dropped the "\n" from the format string in the call to "printf"? (a bit tricky).