Assignment 4 - Eden Keshet - 305156531

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 smallest number and stores it in 73h.

Answer:

```
y equ 71h
z equ 72h
mov eax, o
mov ebx, 0
mov al, word[X]
cmp al, word[Y]
jg case1 ;X < Y
mov bl, word[Y] ; X > Y
case1:
  cmp bl, word[Z]
  jle finish ;X or Y < Z
  mov bl, word[Z] ;X or Y > Z
finish:
  mov [73h], bl
```

2. Consider the following 80X86 code segment. Assume that when the code is run, CX is non-zero.

```
L: ADD AX, AX
  ADC DX, 0
  LOOP L
```

1. **What is the result of running the code?**

2. **Re-write the above code segment for better efficiency.**

Answer:

1. The code suppose to do shift left on AX, CX times.

2.
```
mov edx, 1
  shl edx, cx
  mul
```

3. What is the output of the following code.

```
mov bh, 0
push 0x000a2164
push 0x6f6f6720
push 0x79726556
push esp
call printf
add esp, 16
```

Answer:
The output of the code will be:
print to stdout "Very good!".

4. List 5 different ways to add 1 to the EAX register on an 80X86 with exactly 2 80X86 instructions, where both instructions should be meaningful for a solution (i.e. if delete one instruction of the two, a solution would not work). Assume the following initial state.

eax = 3 ebx = 3

Answer:

```
mov ebx, 1
add eax, ebx
sub ebx, -4 ;ebx=-1
sub eax, ebx
inc ebx
mov eax, ebx
shr ebx, 1
add eax, ebx
mov eax, 0
add eax, 4
```

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):

x: resb 1
z: resw 1
y: resb 1

If we know that subtracting 2 from x and adding 1 to z cause no overflow, is there a shorter way to do that? How?

Answer:

Yes, if we can assume there isn't overflow, we can take x, y and z for one block start at the addrsess of X. for sub 2 from x we can add (ff - 2 = fe), also add 5 to y.

add dword[X], 0x050001fe

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

a. ADD CL, AL
b. ADC AL, 0xF9
c. ADD AX, 0x003A
d. SUB BL, 0x73
e. RCR CX, 3

Answer:

a. CX <- 0000000010010000, CF <- 1, OF <- 0
b. AX <- 0000000011010001, CF <- 1, OF <- 0
c. AX <- 0000000100010001, CF <- 0, OF <- 0
d. BL <- 0000000011111111, CF <- 1, OF <- 0
e. CX <- 0110000000010111, CF <- 0, OF <- 0

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=1000000000000101 represents {1,3,16}). 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.

Answer:

a. xor ax, word[BitTbl + bx]
b. mov cx, 1010101010101010b
   and cx, ax
   cmp cx, 0
c. mov cx, [BitTbl + bx]
and cx, ax
cmp cx, 0
d. or ax, bx
e. and ax, bx
f. not ax

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).

(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?

Answer:

I will add 11000, 00111.

(a) The maximum hamming distance is 2. beacuse 11000 is diffrent from 11111 with 2 bits.

(b) By the formole from Practical sessions it's $d - 1 = 2 - 1 = 1$.

(c) By the formole from Practical sessions it's $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{2-1}{2} \rfloor = 0$.

(d) By the formole from Practical sessions it's $d - 1 = 2 - 1 = 1$.

(e) Yes, for those set of words:

00001

11111

10010

01100

9. The following macros definitions are given %define ctrl 0x1F & %define param(a, b) ((a)+(a)*(b)) %xdefine c1 ctrl %xdefine ctrl 2

Which result code line would be generated by assembler for the following source code line?

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

Answer:

mov byte[2 + 2*ebx], 0x1F & 'D'

10. The following PIC is given

to_printf: dd printf get_my_loc: call next_i next_i: pop edx ret
call get_my_loc push edx add edx, (to-print – next_i)
Which of the answers below are right?

1. a position of to_printf cannot be calculated since this code is PIC
2. this code is not PIC since it uses printf
3. after this code execution EDX contains an offset of printf in the section that contains printf definition
4. after this code execution EDX contains a difference of addresses of to_printf and next_i
5. after this code execution EDX contains an absolute address of printf
6. after this code execution EDX contains an absolute address of to_printf

Answer:
The correct answer is 6.

11. Given the following C code

z = foo(&x, &y);
What is a return value of foo (using x and y to state the answer)?

Answer:
Can't know, Segmentation Fault

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.

Answer:
Case 1: $D_0 = 0$, push 0, subtract 1, and then jump to N and return D_2.
Case 2: $D_0 = 1$, push 1,recursive do subtract 1 from D_0, and returns $D_2 = -1$.

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

(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?

Answer:
(a) 7, from the ELF header offset 0x30
(b) It's executable file, from the ELF header magic number.
(c) There is 1 program headers.
(d) The program header is load from offset 0 in the in the file t0 vaddr 0x08048000, size of 0x009E bytes.
(e) 0xD0 or 208 (dec), , from the ELF header offset 0x20.
(f) .symtab, .strtab, .shstrtab, .text, .rodata, .Syria

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

Answer: No, the answer isn't uniqe because every child depends on the scheduler.
NUKE 3!
NUKE 2!
NUKE 1!
NUKE 1!
NUKE 1!
NUKE 1!
NUKE 3!
NUKE 1!
NUKE 1!
NUKE 2!
NUKE 1!
NUKE 1!