ACTIVITY 1A

X86 MACHINE LANGUAGE PROGRAMMING (I)

An x86 machine language program consists of a sequence of bytes, where each byte is a sequence of 8 bits.

All microprocessors have *registers,* which hold one or more bytes of data and store intermediate values during computations. The Intel x86 processors have a register called AL, which holds a single byte. The contents of AL when a program begins executing are unspecified.

The table below shows four x86 machine language instructions. The first instruction is one byte long; the others are two bytes.

Instruction	Meaning
10010000	Does nothing.
10110000 *value*	Places a constant value in the AL register. The second byte of the instruction determines the value (see the table to the right).
00000100 *value*	Adds a constant value to the value currently stored in AL. The sum is placed in AL.
00101100 *value*	Subtracts a constant value from the value currently stored in AL. The difference is placed in AL.

Binary Representations

0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
10	00001010
11	00001011
12	00001100
13	00001101
14	00001110
15	00001111
16	00010000
17	00010001
18	00010010
19	00010011
20	00010100

1. What does the following program do? What value is in AL when execution completes?

   10110000 00000111 10010000 00101100 00000010

2. Write a machine language program that computes

   $$17 + 12 - 7$$

   and places the result in the AL register.

ACTIVITY 1B

x86 MACHINE LANGUAGE PROGRAMMING (II)

Here are the four x86 machine language instructions again, this time represented in hexadecimal rather than binary.

Instruction	**Meaning**
90h	Does nothing.
B0h *value*	Places a constant value in the AL register. The second byte of the instruction determines the value (see the table to the right).
04h *value*	Adds a constant value to the value currently stored in AL. The sum is placed in AL.
2Ch *value*	Subtracts a constant value from the value currently stored in AL. The difference is placed in AL.

(The *h* suffix is used to emphasize that the numbers are hexadecimal rather than decimal.)

3. What does the following program do?

   90h B0h 14h 04h 0Dh

4. Write a machine language program that computes

   14 – 3

   and places the difference in the AL register.

Hexadecimal Representations

0	00h
1	01h
2	02h
3	03h
4	04h
5	05h
6	06h
7	07h
8	08h
9	09h
10	0Ah
11	0Bh
12	0Ch
13	0Dh
14	0Eh
15	0Fh
16	10h
17	11h
18	12h
19	13h
20	14h

ACTIVITY 1C

X86 MACHINE LANGUAGE PROGRAMMING (III)

Here are the four x86 machine language instructions again, along with their equivalent assembly language instructions.

In assembly language, values can be represented in several different ways.

- Decimal numbers are written as-is. 20 represents the value 20.
- Hexadecimal numbers are written with an h suffix. 0Eh represents the value 14.
- Binary numbers are written with a b suffix. 00001101b represents the value 13.

Machine Language Instruction	**Assembly Language Representation**
90h	`nop`
B0h *value*	`mov al,` *value*
04h *value*	`add al,` *value*
2Ch *value*	`sub al,` *value*

5. What does the following program do?

```
mov al, 14h
add al, 10
sub al, 00000111b
nop
```

6. Disassemble the following machine language program (i.e., translate it into assembly language).

   90h B0h 14h 04h 0Dh

7. Write an assembly language program that computes

$$10 + 4 - 3$$

   and places the result in the AL register. (Keep the program as simple as possible; use decimal representations of all numbers.)

8. Assemble your program from #7 (i.e., translate it into machine language). You can write the machine language code in either binary or hexadecimal.