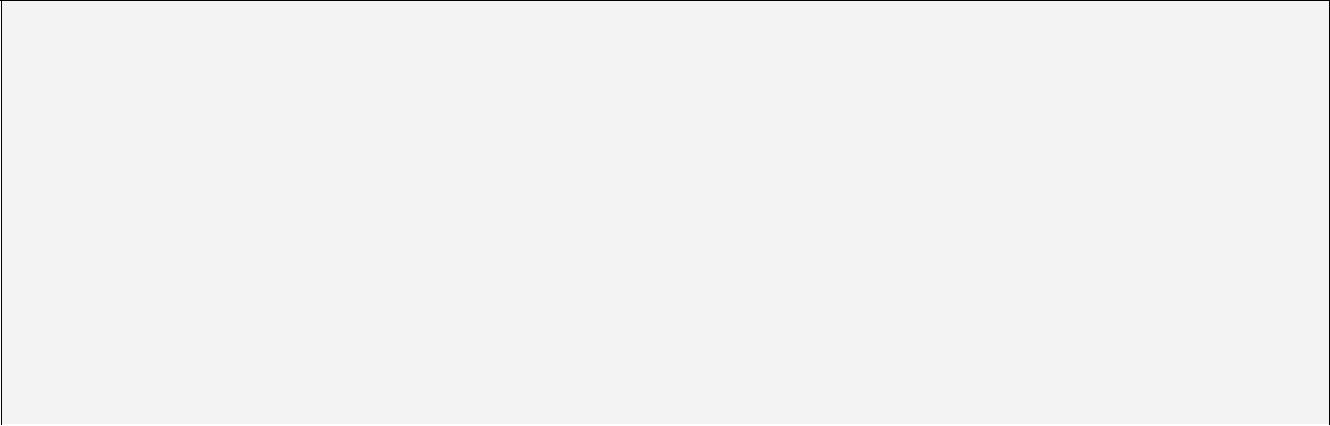
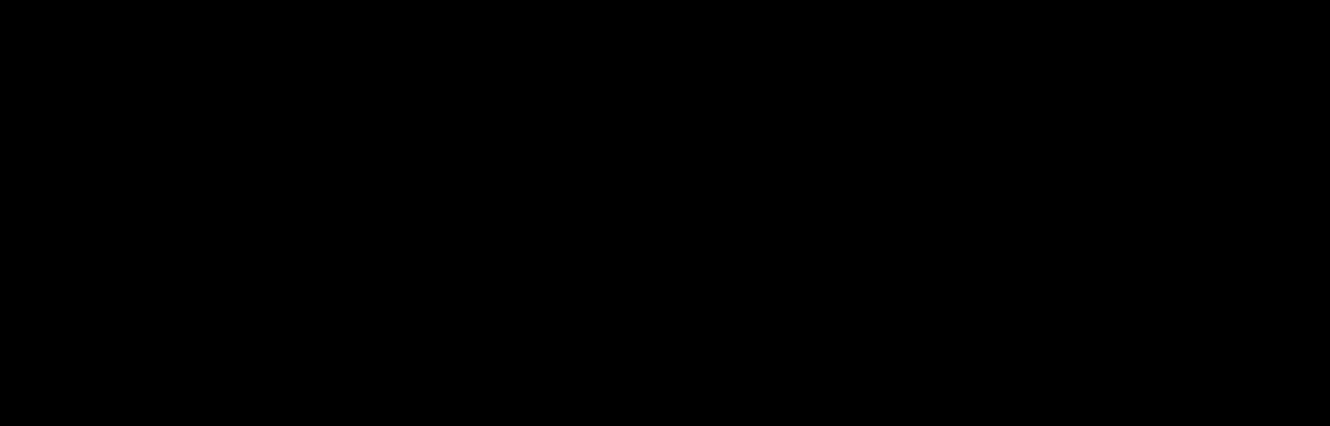
**Assembly Lab 2 Fall 2017**

**First Assembly language program:**

Let us now write a simple assembly language program, which add two numbers. (**;** means comment)



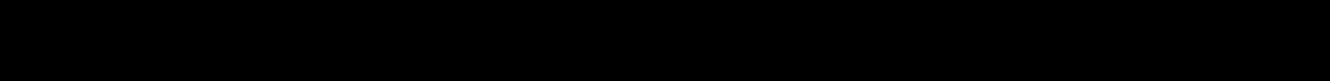
;; A PROGRAM TO ADD TWO NUMBERS

|  |  |  |
| --- | --- | --- |
| ;; USING REGISTERS  [ORG 0x0100] |  |  |
| MOV AX, 1 |  | ; AX = 1 |
| MOV BX, AX |  | ; BX = AX |
| ADD BX, 3 |  | ; BX = BX+3 |
| MOV AX, 0x4C00 | | ; |
| INT 0x21 | ; Terminate Program | |
|  |  |  |

Write it on notepad and save as **Test.asm, and place this file where you have placed nasm and afd in last lab.**

**ASSEMBLING Test.asm**

1. Switch to command prompt (dosbox). Now we use NASM to assemble this program using the command:



***nasm Test.asm –o Test.com***

This generates 1 files,

***Test.com****.*

**DEBUGGING Test.com**

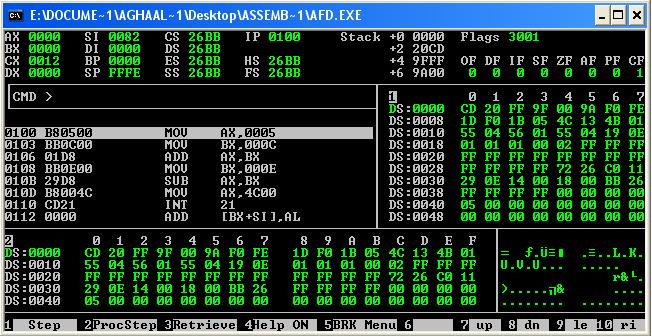
Now we will use a debugger to monitor the execution of this program. The debugger we are using is *Advanced Fullscreen Debugger (AFD).* We will load the com file by typing

***afd test.com***

As shown in the diagram, the interface of the debugger is easy to understand and use.

* + Using ***F1*** we can step through the code and see the changes taking place in the register contents.
  + Write Commands in “cmd >” to Execute AFD commands. i.e. write **quit** to terminate AFD.

**AFD OVERVIEW**



You can see registers on the top left, Flags on top right and Stack in middle. Also 2 memory panes are visible as 1 and 2.

While pressing F1 look at the registers and understand their values.

**Listing File:**

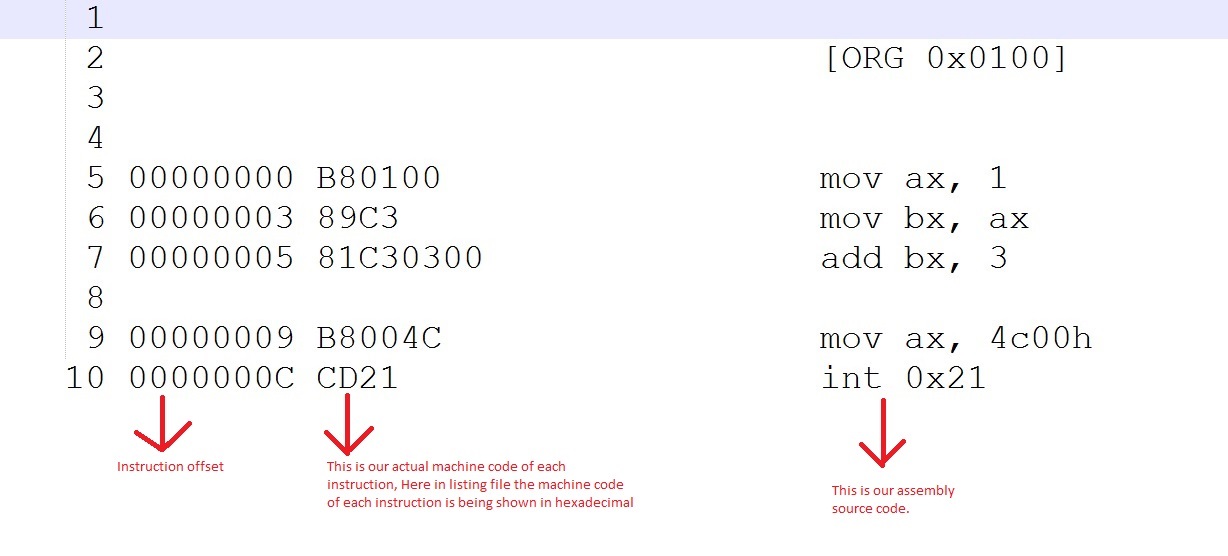
Listing file is created by nasm when you enter:



***nasm Test.asm –o Test.com –l list\_file.lst***

It will create two files **i. Test.com ii. list\_file.lst**

Listing File will look something like this:



**Registers:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **Registers** | **Range** | |
| **Unsigned** | **Signed** |
| 8 bits | AH AL BH BL CH CL DH DL | 0 to 255 (0-28-1) | − 128 to +127 (27 to 27-1) |
| 16 bits | AX BX CX DX | 0 to 65,535 (0-216-1) | − 32,768 to +32,767 (215 to 215-1) |

**Flags:**

* There are 8 flags, which you can see on AFD.
* Status flags monitor the outcome of arithmetic, logical, and related operations.
* Once a flag is set, it remains in that state until another instruction that affects the flags is executed.
* Debugger check every number as signed and unsigned number then set or unset flags.
* Not all instructions affect all status flags. **(Run all given commands of status flag type)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Flag** | **Flag Type** | **When it Enables** | **Commands or Examples** |
| CF (Carry Flag) | Status Flag | Records that the result of an arithmetic operation on **unsigned numbers** is out of range | AX = 0xFFFF  BX = 1  AX = AX + BX |
| DF (Direction Flag) | Control Flag | Index move forwards if DF =0  Index move backwards if DF = 1 | ----------------------------------- |
| IF (Interrupt Flag) | System Flag | If an interrupt is generated | Ax = 10  BX = 0  AX = AX / BX  Generate divided by zero interrupt |
| SF (Sign Flag) | Status Flag | sign flag indicates that result goes below 0, if result is –ve it will be 1, otherwise 0 | AL = 15  BL = 97  AX = AX - BX |
| ZF (Zero Flag) | Status Flag | If the result is zero, zero flag is set | AL = 0Fh  AL = AX - 0Fh |
| AF (Auxiliary flag) | Status Flag | Indicates whether an operation produced a carry or borrow in the low-order 4 bits (nibble) of 8, 16 bit operands (i.e. operand size doesn’t matter) |  |
| PF (Parity flag) | Status Flag | Indicates even parity of the low 8 bits of the result, PF is set if the lower 8 bits contain even number 1 bits |  |
| OF (Overflow Flag) | Status Flag | Indicates out-of-range result on **signed numbers** | AL = 127  AL = AL + 1 |

**In lab Problems**

**Problem 1: (MOV, ADD, SUB)**

1. Write an assembly program to sum of first ten elements of a Fibonacci series by using only **MOV and ADD** instruction. You are not allowed to use following numbers except 2 and 1, as well as maintain sum of all elements in some register.

In Fibonacci series first two elements are 0,1 then third one is sum of 1st and 2nd then forth one is sum of 2nd and 3rd and so on

Fibonacci series: **0, 1,** 1, 2, 3, 5, 8, 13, 21, 34…

Your final answer should be 58 in hex

1. Write an assembly program to subtract first 10 elements of a Reverse Lucas series by using only **MOV and SUB** instruction.

You are not allowed to use following numbers except 76 and 47, generate every next term to compute final difference.

29 = 76 – 47

18 = 47 - 29

Reverse Lucas series: 76, 47, 29, 18, 11, 7, 4, 3, 1, 2

1. **Problem 3: Logical Operations (AND, OR, XOR, NOT)**
2. Copy the value of AX into BX using a logical instruction. (You are not allowed to use mov BX, AX)
3. Set AX=0 by using XOR instruction only.
4. Set AX=0 by using AND instruction.
5. Set AX=FFFF by using OR instruction.
6. Invert the bits of AX register using a single line instruction

**Problem 3:**

Calculate code sizes of Problem 1 using their listing files.

**Good to read**

**Difference between Mnemonic and Opcode:**

**MNEMONIC:**

**Human Readable words. The assembly keywords, such as, mov, add, sub, etc are MNEMONICS for programmers because they are easily understood by them. We use these MENMONICS to write programs because we can easily remember mnemonics. Mnemonics cannot be executed by the CPU, so mnemonics are always converted into some opcode which can be executed by the CPU.**

**OPCODE**:

It is a number interpreted by the CPU that represents the operation to perform. For example in the above listing file, the opcode for moving an immediate operand into AX register is B8.

**Purpose of [ORG 0x0100]:**

It simply tells the nasm that the instructions of our program should place at the start 256th byte of code segment. (The first 256 bytes are to be skipped). Note that 0x0100 is a hexadecimal number whose decimal value is 256. The reason we skip the first 256 bytes is because these bytes have some important piece of code that we do not want to overwrite with our own. This will be further clarified in some later lab session.

***Wahi hai saahib-e-imroz (today's master) jis ne apni mehnat se***

***Zamaane ke samundar se nikaala gohar-e-fardan (tomorrows pearl)***

**[ - Dr Allama Muhammad Iqbal - ]**