Disassembling code with NASM

A compile takes as input high-level source code and outputs assembly code, which is then assembled into binary. Most assemblers come with a **disassembler** that can be used to convert binary code back to (human readable... sort of) assembly code.

With NASM, the disassembler is called **ndisasm**. Say you have a C program called `stuff.c` as follows:

```
#include <stdio.h>

int main() {
        int i;
        int sum = 0x1234;

        sum += 0xABCDEF;
        for (i=0; i < 10; i++) {
                sum += i;
        }
        printf("sum=%d\n",sum);
        sum = 0x2345;
}
```

Here is a sequence of commands to look at the assembly code generated by the compiler (in a 32-bit world):

```
% gcc –m32 cprogram.c –o cprogram
% ndisasm –b 32 cprogram > cprogram.asm
```

The file `cprogram.asm` now contains the disassembled code. On my Linux box it has 3323 lines! You note that in the C code I have put some “easy to spot once translated to assembly” constant. The relevant piece of assembly is:

```
000003ED  C744241C34120000  mov dword [esp+0x1c],0x1234
000003F5  C744241800000000  mov dword [esp+0x18],0x0
000003FD  EB0D              jmp short 0x40c
000003FF  8B442418          mov eax,[esp+0x18]
00000403  0144241C          add [esp+0x1c],eax
00000407  8344241801        add dword [esp+0x18],byte +0x1
0000040C  837C241809        cmp dword [esp+0x18],byte +0x9
00000411  7EEC              jng 0x3ff
00000413  B810850408        mov eax,0x8048510
00000418  8B54241C          mov edx,[esp+0x1c]
0000041C  89542404          mov [esp+0x4],edx
00000420  890424            mov [esp],eax
00000423  E8D8FEFFFF        call dword 0x300
00000428  C744241C45230000  mov dword [esp+0x1c],0x2345
```

On each line the disassembler conveniently prints the address of the instruction and the binary code for the instruction on each line.