Imperial College London
Department of Computing

Computer Systems (M2)
Pentium programming lab exercise for *Intel Macs*

You can try the following version of the program on Intel Macs running OS X (Tiger, Leopard, Snow Leopard). nasm is included on Intel Macs.

* Type the program in the box below into a file called `hello.s` **Do not make any mistakes!!!!**

```
segment .data
msg     db 'Hello world!',0xA
len     equ $-msg

segment .text
global start

start:
        mov  eax, 5
outer:
        mov  ebx, 1000000000

inner:
        dec  ebx
        jg   inner
        dec  eax
        jg   outer

        mov  eax, 4
        push dword len
        push dword msg
        push dword 1
        push dword 0
        int  0x80
        add  esp, 16

        mov  eax, 1
        push dword 0
        push dword 0
        int  0x80
```

```
switch to data segment
declare and initialise variable msg
set constant len = number of bytes in msg

switch to text (i.e. code) segment
make start visible outside of this file

program starts here
number of times to repeat outer loop

repeat inner loop 1 billion times. type 1 followed
    by 9 zeros - do not type any more zeros!

execute this & next instruction 1 billion times
jump if ebx greater than zero to label 'inner'
decrement eax outer loop counter
jump if eax greater than zero to label 'outer'

OSX system call 4, i.e. write ()
number of bytes in message to write
address of variable 'msg'
file descriptor 1, i.e. standard output
not used - compatibility for C return addresses
interrupt OSX, i.e. OSX will write the message
remove parameters from stack

OSX system call 1  i.e. exit ()
error code 0, i.e. no errors
not used - compatibility for C return addresses
interrupt OSX, i.e. OSX will exit the program
```

* Assemble into an object file version with:
`nasm -f macho hello.s`
`nasm` is the Netwide assembler. The command will produce an object file named `hello.o` if there are no errors in file `hello.s`

* Then link into an executable program with:
`ld -o hello hello.o`
`ld` is the OS object file 'linker' which can (amongst other things) link several object files into one executable program.

Run the program with:
`hello` or `./hello`
The program executes over 10 billion Intel machine instructions! 5 billion `dec` instructions and 5 billion `jg` instructions.

* Find the size of the executable file with:
`wc -c hello`

* Find the size of the code and data with:
`size -m hello`
The code size (in bytes) is given after '__text'

Why is there a difference between the sum of these sizes and the size of the executable program file?
Try the command `file hello` also.

* Run and time the program with:
`/usr/bin/time -p hello`
If you run the program several times, the times may differ. Why?
View the cpu type using
`/usr/sbin/system_profiler SPHardwareDataType`