

Imperial College London
Department of Computing

Computer Systems (113) / Architecture (110)
Intel IA-32 exercise for *Linux*

You'll be writing an Intel IA-32 assembly language program soon as part of the PPT lab. In the meantime you're encouraged to try the following program. To do so, you'll need to login to one of the CSG machines running **Linux** (not Windows!).

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

<pre>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 mov ebx, 1 mov ecx, msg mov edx, len int 0x80 mov eax, 1 mov ebx, 0 int 0x80</pre>	<pre>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' linux system call 4, i.e. write () file descriptor 1, i.e. standard output address of variable 'msg' number of bytes in message to write interrupt Linux, i.e. Linux will write the message linux system call 1 i.e. exit () error code 0, i.e. no errors interrupt Linux, i.e. Linux will exit the program</pre>
<p>* Assemble into an object file version with:</p> <pre>nasm -f elf hello.s or nasm -f elf64 hello.s</pre>	<p><code>nasm</code> is the Netwide assembler. The command will produce an object file named <code>hello.o</code> if there are no errors in file <code>hello.s</code>. Use <code>nasm -f elf64 hello.s</code> on a 64-bit machine.</p>
<p>* Then link into an executable program with:</p> <pre>ld -s -o hello hello.o</pre>	<p><code>ld</code> is the Linux object file 'linker' which can (amongst other things) link several object files into one executable program.</p>
<p>Run the program with:</p> <pre>hello or ./hello</pre>	<p>The program executes over 10 billion Intel IA-32 instructions! 5 billion <code>dec</code> instructions and 5 billion <code>jg</code> instructions.</p>
<p>* Find the size of the executable file with:</p> <pre>wc -c hello</pre>	<p>This is probably the smallest complete program you'll ever create during your degree!</p>
<p>* Find the size of the code and data with:</p> <pre>size hello</pre> <p>Why is there a difference between the sum of these sizes and the size of the executable program file?</p>	<p>The code size is given in the 'text' column!</p> <p>Try the command <code>file hello</code> for fun also.</p>
<p>* Run and time the program with:</p> <pre>/usr/bin/time -p hello</pre> <p>If you run the program several times, the times may differ. Why?</p>	<p>For a more verbose timing output use the command:</p> <pre>/usr/bin/time -v hello</pre> <p>View the cpu type using <code>more /proc/cpuinfo</code></p>
<p>* Login to other CSG Linux machines with different processors and compare the speed.</p>	<p>For names of machines, go to the old csg home page and click on Computer List on the panel.</p>