Chapter 1

Assembly

1.1 Registers

What are registers for?

- Store instructions
- Store result of operations
- Manipulate data in them (shift registers)

Main registers		
AH	AL	**AX** (primary accumulator)
BH	BL	**BX** (base, accumulator)
CH	CL	**CX** (counter, accumulator)
DH	DL	**DX** (accumulator, other functions)
Index registers		
SI		**S**ource **I**ndex
DI		**D**estination **I**ndex
BP		**B**ase **P**ointer
SP		**S**tack **P**ointer
Status register		
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		(bit position)
- - - - O D I T S Z - A - P - C		Flags
Segment register		
CS		**C**ode **S**egment
DS		**D**ata **S**egment
ES		**E**xtra**S**egment
SS		**S**tack **S**egment
Instruction pointer		
IP		**I**nstruction **P**ointer

Why EAX,EBX,ECX? Registers AL,AH,BL,BH,CL,CH,DL,DH are of 16 bits each and can be used in pairs: AX, BX, CX, DX

1.2 Hello world

```
section .data
        hello:     db 'Hello world!',10    ;
        helloLen:  equ \$-hello             ; Length of the 'Hello world'
                                            ; (I'll explain soon)

section .text
        global _start

_start:
        mov eax,4            ; The system call for write (sys_write)
        mov ebx,1            ; File descriptor 1 - standard output
        mov ecx,hello        ; Put the offset of hello in ecx
        mov edx,helloLen     ; helloLen is a constant, so we don't need to say
                             ;  mov edx,[helloLen] to get it's actual value
        int 80h              ; Call the kernel

        mov eax,1            ; The system call for exit (sys_exit)
        mov ebx,0            ; Exit with return code of 0 (no error)
        int 80h
```

Then in the terminal:

```
nasm -f elf hello.asm
ld -m elf_i386 -s -o hello hello.o
```

Note that the elf_i386 is necessary as we are writing 32 bit assembly code in a 64 bit architecture.

1.3 References

1. https://www.quora.com/What-is-an-intuitive-explanation-of-how-CPU-registers-work
2. http://stackoverflow.com/questions/2545192/what-does-x-mean-in-eax-ebx-ecx-in-assembly
3. http://docs.cs.up.ac.za/programming/asm/derick_tut/