Section of an assembly program

section.data

The data section is used for declaring initialized data or constants.

section.bss

The bss section is used for declaring variables.

section .text

The text section is used for keeping the actual code. This section must begin with the declaration global _start, which tells the kernel where the program execution begins.

```
section.text
global _start
_start:
```

Statemets

In assembly we can have three kind of statemets: istructions, directives or pseudo- ops. The syntax is the follwing:

```
[label] mnemonic [operands] [;comment]
```

The mnemonic is the name of the istruction, this is the central point of the istruction. The operands are the parameters for the mnemonic. Examples:

```
INC COUNT        ; Increment the memory variable COUNT

MOV TOTAL, 48    ; Transfer the value 48 in the
                ; memory variable TOTAL

ADD AH, BH       ; Add the content of the
                ; BH register into the AH register

AND MASK1, 128   ; Perform AND operation on the
                ; variable MASK1 and 128

ADD MARKS, 10    ; Add 10 to the variable MARKS
MOV AL, 10       ; Transfer the value 10 to the AL register
```

Istructions

To tell the processor what to do. Each instruction consists of an operation code (opcode). Each executable instruction generates one machine language instruction.

Directives

To tell the assembler about the various aspects of the assembly process.

Macros

Basically a text substitution mechanism.

Registers

To speed up the processor operations, the processor includes some internal memory storage locations, called registers. We have three categories of registers:

- General registers
- Control registers
- Segment registers

General registres

The general registers are further divided into the following groups:

- Data registers,
- Pointer registers
- Index registers

Data registers

Used for arithmetic, logical, and other operations. We have the following registers:

- AX primary accumulator
- BX base register
- CX count register
- DX data register

Pointer registers

- IP istruction pointer: offset in code segment, current istruction.
- SP stack pointer: offset value in the stack.
- BP base pointer: referencing variables passed to a subroutine.

Index registers

- SI Source Index: source index for string operations
- DI Destination Index: destination index for string operations.

Control Registers

- Overflow Flag (OF)
- Direction Flag (DF)
- Interrupt Flag (IF)
- Trap Flag (TF)
- Sign Flag (SF)
- Zero Flag (ZF)
- Auxiliary Carry Flag (AF)
- Parity Flag (PF)
- Carry Flag (CF)

Linux System Calls

To interact with the operative system we have to use the seo called system calls in our assembly code.

Passages:

1. Put the system call number in the EAX register.
2. Store the arguments to the system call in the registers EBX, ECX, etc.
3. Call the relevant interrupt (80h).
4. The result is usually returned in the EAX register.

All the syscalls are listed in /usr/include/asm/unistd.h, together with their numbers (the value to put in EAX before you call int 80h).