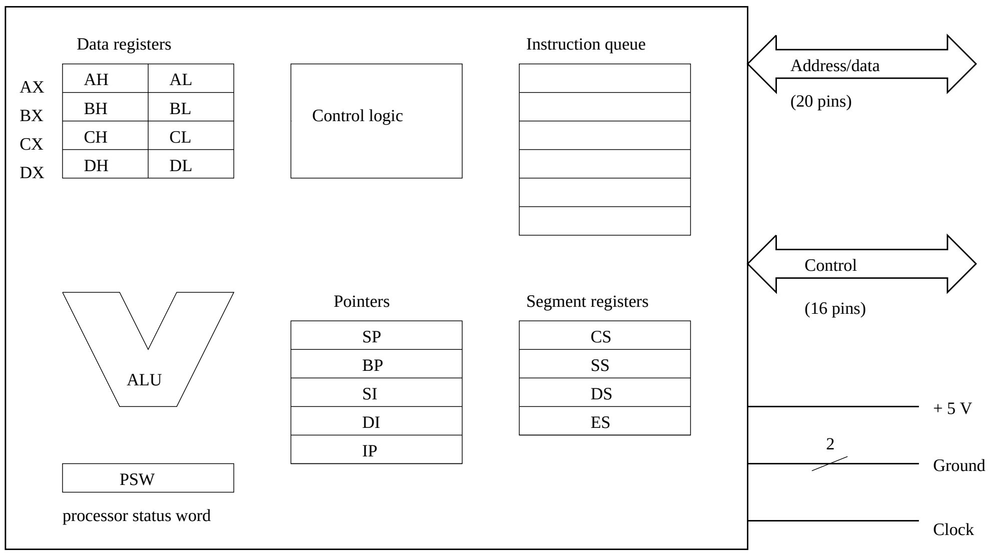
8086/8088 Assembly Language Programming: Introduction

Why assembly language? Applications that are not time-critical, or only standard input/output devices are used have minimal direct need for assembly language programming. Otherwise, assembly language is used to program time critical tasks. Moreover, some programmers must write the library routines to achieve standard interfaces and these routines are written in assembly language.

CPU Architecture.



All registers are 16-bit wide and they are categorized as follow:

Data registers: The registers are used to store both operands. Results in each of them can be accessed as a whole; or the upper and lower bytes can be accessed separately. Besides serving as arithmetic registers, BX, CX, and DX registers play special addressing, counting, and I/O roles: (i) AX – Accumulator; (ii) BX – Base register; (iii) CX – Counter register; (iv) DX – Data register.

Pointer registers. (i) IP – Instruction pointer register (program counter); (ii) SP – Stack pointer register; (iii) BP – Base register for accessing the stack; (iv) SI,DI – They are used as indexing which are often used with the BX or BP registers.

Remark: To provide flexible base addressing and indexing (e.g. array indexing in high-level language terminology), a data address may be formed by adding together a combination of the BX or BP register contents, SI or DI register, and a displacement. The result of such an address computation is called an *effective address* or *offset*.

Segment registers. (i) CS – Code segment register; (ii) SS – Stack segment register; (iii) DS – Data segment register; (iv) ES – Extra Data segment register.

Recall that all effective addresses are 16-bit wide. However, the address put on the address bus, called the *physical address*, must contain 20 bits. The extra 4 bits are obtained by adding the effective address to the contents of one of the segment registers. For example, if (CS) = 123A, and (IP) = 341B, then the next instruction will be fetched from $341B + 123A0 = 157BB$. (Note that (IP) means the contents of IP and the addresses are represented in hex).

Assembler. We shall be using the Turbo Assembler and Linker to assemble and link our assembly program. They are available via the campus-wide network. Suppose we have an assembly program `try.asm`. After booting up one of the campus machine, we can:

```
C:\> l:\bc31\bin\tasm try.asm
```

which, on successful assembling, will produce the object code `try.obj`. Then type

```
C:\> l:\bc31\bin\tlink try.obj
```

to link the object so as to produce the executable code `try.exe`. You may use the program `edit` provided by DOS or other ASCII editor as well to edit your program. In Windows, you can use notepad; but *be sure to save the file with extension* `.asm`. Note that both `tasm` and `tlink` are located in `L:\BC31\BIN`.

Sample assembly program.

```
;
; This program adds two number and store the result
; in the main memory
;
data_seg        segment                         ; data segment
        oper1   dw      12
        oper2   dw      230
        result  dw      ?
data_seg        ends
code_seg        segment                         ; code segment
        assume  cs:code_seg, ds:data_seg
start:  mov     ax,data_seg
        mov     ds,ax
        mov     ax,oper1                        ; move oper1 to Accumulator
        add     ax,oper2                        ; add with oper2
        jge     store                           ; go to store if the result >= 0
        neg     ax                              ; else negate the result
store:  mov     result,ax                       ; store the result in main memory
        mov     ax, 4c00h                       ; set up DOS terminate service
        int     21h
        hlt
code_seg        ends
        end     start
```