Curs 2

joi, 7 octombrie 2021
18:45

Every of this 8 registers has a special meaning in a certain context

- AX - accumulator - used by explicit operands (in multiplication or division is the target or result, in division is the dividend)
- BX - base register - used in address computation (in C if we have an array A then A - it's the pointer = THE BASE (the starting address))
- CX - counter (put 100 in it, use a loop, the loop will repeat 100 times) - loop expects to find the value in ECX
- DX - data register - when data doesn't fit where it's supposed to in EDX will be made space for that thing (as an extension)
    - Ex: byte + byte (ADD) = byte;
    - MULTIPLY op1(M positions)*op2(N positions) = M+N positions
    - Byte*byte = word; w*w = dw; dw*dw = qw
    - In 1975 when we multiplied 2 words we used DX since there were only 16 bit data types (word), the result was put on DX:AX
    - Nowadays data can be put on EDX:EAX
    - In 16 bit programming usage of 32 bit values is allowed in a limited way (results of a multiplication/expressing the dividend of a division)
    - The same is in 32 bit programming where quad words are allowed in a limited way.

Every name we write in a programming language is a title (denomination) of a memory cell.
Every name we use is in fact an address used accessing that memory cell.
* in C is the DEREFERENCING operator

A[i]; i = index

- DI - destination index
- SI - source index

Data structures: array, list, queue (FIFO), stack (LIFO)
Why is the stack so important????

RUN-TIME Mechanism of any program in CS follows always the LIFO order of activating an running the involved programming units (subroutines = functions + procedures)
The stack is the main data structure used by the processor to run all its programs

- SP - stack pointer
- BP - base pointer
- SS - (nu e dintre cele 8 registre generale) starting of the stack area

EBP and ESP will point to the base and to the top of the currently executing stack frame, which is the active part of code that is running at that moment.

A user defined type in C is defined by TYPEDEF (which is INCORECT because typedef is in fact defining only the structure)

C, Java, VB, Pascal, Fortran - were IMPERATIVE languages, because they rely as a central element on the INSTRUCTIONS (commands).

DATA TYPE = structure + associated OPERATIONS !!!!
(essential in this definition is ASSOCIATED - we did not have until OOP AN ENCAPSULATION MECHANISM)

OOP = DATA ORIENTED PROGRAMMING (everything is build having as the central figure the notion of DATA).
In OOP we also have inheritance an polymorphism.

For the microprocessor the notion of data type has a very primitive meaning limited only to the size of representation.
On 32 bits these can be: byte, word, dword and qword(limited). You can define variables/operands in the RAM memory by using DATA DEFINITION DIRECTORIES: DB, DW, DD, DQ.