Computer Systems Architecture

C is the lowest high level language C is the only language that can do pointer arithmetic. The bit is the most basic unit of representing the information (0, 1). The byte is the smallest addressable unit of information.

Every single one of the 8 general registers has a special meaning

The "E" before the register name stands for "Extended"

AX - Accumulator register

A[i] ; i = INDEX

A[7] = \*(A+7) \* = the DEREFERENCING operator in C

1. - the name of an array in C is its starting address - it's a pointer = THE BASE (the starting address)

Every name that you write in a programming language is a title for a memory cell

Every name that you use is, in fact, an address used for accessing that memory cell

Byte + byte (ADD) = byte

MULTIPLY op1 (M positions) \* op2 (N positions) -> M+N positions

1. \* B -> W (Word)

W \* W -> DW (Doubleword)

(DX:AX)

DW \* DW -> DQ (Quadword)

CX - Count register

In 16 bits programming, usage of 32 bits values are allowed in a limited way (results of a multiplication or expressing the dividend of a division) - (same goes for 32 bits to 64 bits)

SP - Stack pointer (points to the stack)

BP - Base pointer (points to the stack)

Data structures - array, list, queue (FIFO), stack, (LIFO) WHY is the stack SO important?????

RUN-TIME Mechanism of ANY program in Computer Science ALWAYS FOLLOWS

THE LIFO ORDER of activating and running the involved programming units

(subroutines = functions + procedures)

SS - Address register

EBP & ESP - will point you to the base, respectively to the top, of the currently executing stack frame

A user defined type in C is defined by TYPEDEF (which is INCORRECT, 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 INSTRUCTION

DATA TYPE = structure + associated OPERATIONS!!!

(essential in this definition is ASSOCIATED - we did not have until OOP AN

ENCAPSULATION mechanism)

You also have in OOP inheritance + polymorphism

OOP = DATA ORIENTED PROGRAMMING (everything is built having as the central figure the notion of DATA)

From the point of view of the mP - which is its understanding of DATA TYPE notion?

For the mP, the notion of DATA TYPE has a very primitive meaning, limited only to the size of representation of that element

On 32 bits, these can be:

byte word dword qword (these are the assembly language DATA TYPES); You can define variables/operands in the RAM memory by using DATA DEFINITION DIRECTIVES:

DB

DW

DD

DQ

RAM (Random Access Memory) - who is RANDOM?

The access time at any given location from the RAM is THE SAME independently of the position (randomly far from the beginning of the memory...)

In contrast with ROM (read only memories), a RAM supports/allows any number of R/W and in any ORDER (Randomly... reads and writes in a random order... The order in which R/W appears is RANDOM...)