.Net platform

Runtime → it is the software used to execute a program.

* Provides LOAD/LINK mecanisms

Langage :

* Types → int, char, float etc… → pas indispensable tho
* Control flow → relatif au cheminement du programme (ex: fonctions)
* Calcul (expressions) → computers only do maths. if/else… relate to calculus
* Mémoire → LA valeur ajoutée, 100% indispensable

Compilé Vs Interprété:

* S’il y a des ; et des blocs ({}) → éléments relatifs à la compilation (le + souvent)
  + Dans les langages interprétés, c’est le \n qui fait foi
* Dans certaines conditions, certains langages sont “compilés” au lieu d’interpréter chaque fois (ex: une boucle de 10k itérations)

|  |  |
| --- | --- |
| Compilé | Interprété |
| * Traduction offline (rapide à l’exe) | * Traduction online (lent à l’exe) |
| * Source non portable | * Source portable |
|  |  |
| **RAPIDE** | **PORTABLE** |

3 types d’instructions

* JUMP → permettent de contrôler le reste des opérations
  + int , call jump, jnz, jne
* TRANSFER → son intérêt dans l’ordi c’est de pouvoir aller dans les deux sens (depuis ou vers la mémoire.)
  + mov, push, pop, in out
* CALCULATION → tout ce qui est relatif à l’arithmétique
  + Add, sub, mul, div, and, or, asr, asl, cmp

ALU est la partie intéressante : c’est l’unité d’arithmétique logique.

Runtime :

* Program pour le développeur
* API vers l’OS

Visual Studio c’est une machine virtuelle, une plateforme, multi-langages, ...

Assembly plutot que Binary

Les programmes ne contiennent pas de code natif. Les mêmes fichiers sont portables et exécutables sur différentes plateforme

A l’execution utilise le CLR : Common Language Runtime

Il y a toujours une compilation vers le binaire JIT : Just In Time avant de run. Ce n’est pas un bytecode qui est interprété à chaque fois mais une version spécifique à chaque système.

Permet :

- Garbage collecting

- Threading

- Exception Handling

- Interoperability

- Security

- Debugging