Arquitetura do SET de instruções
Instruction SET

CISC vs RISC

What’s assembly as to do with it?

Low-level - high-level programming language

Assambley CODE

```
section             .text
  global _start     ;must be declared for using gcc

_start:                         ;tell linker entry point
  mov               eax,'3'
  sub    eax, '0'

  mov               ebx, '4'
  sub    ebx, '0'
  add               eax, ebx
  add               eax, '0'

  mov               [sum], eax
  mov               ecx,msg
  mov               edx, len
  mov               ebx,1
  mov               eax,4
(sys_write)
  int               0x80
  mov               ecx,sum
  mov               edx, 1
  mov               ebx,1
  mov               eax,4
(sys_write)
  int               0x80

  mov               eax,1
(sys_exit)
  int               0x80

section .data
msg db "The sum is:", 0xA,0xD
len equ $ - msg
segment .bss
sum resb 1
```

Visual Básic

Fortan; C; Pascal

Uma linguagem entendida pelo programador – semelhante à linguagem corrente

high-level programming language
linguagem de programação de alto nível

Linguagem assambley ADD; SUB; JUMP

Linguagem do Instruction SET – entendível para o programador

Código Máquina / *machine code*

01010101 A única linguagem entendida pelo computador

Low-level programming language
linguagens de baixo nível

Hardware “CPU”

Ponto prévio

What's assembly as to do with it?

Existem dois tipos de linguagens de programação:

Low-level programming language (linguagens de baixo nível) - é uma linguagem de programação que muito relacionada com o SET de instruções da máquina. É uma linguagem que está "perto do hardware". Programas escritos em linguagens de baixo nível tendem a ser relativamente não-portáteis, principalmente por causa da estreita relação entre a linguagem e a arquitetura de hardware.

Para esta programação pode ser usada:

- linguagem máquina *machine code* - A única linguagem entendida pelo computador. (apenas entendível pelo computador, constituída por zeros e uns 01010101, não entendível para humanos)

ou

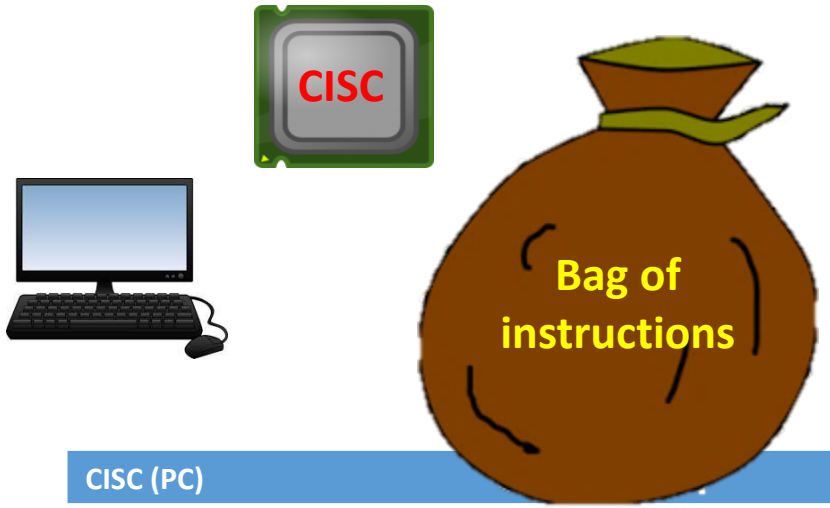
- linguagem assambley ou assembler que utiliza a já referida linguagem do Instruction SET (ADD; SUB; JUMP; etc..).

Contudo, e apesar de ser uma linguagem de baixo nível é necessário traduzi-la para linguagem máquina, para que seja entendível pelo computador, para esse efeito é utilizado um tradutor ou assembler.
ex: ADD; SUB; JUMP >>> ASEMBLER >>>>> Computador

high-level programming language (linguagem de programação de alto nível) é uma linguagem de programação distante do código máquina em comparação com linguagens de programação de baixo nível. Podem ser usados elementos da linguagem natural tornando-se mais fácil aos humanos, alias foi com esse propósito que foi inventada. Exemplo C++ que utiliza caracteres como +*-/ sem recorrer a palavras do SET de instruções ou a código binário.

CISC vs RISC

Complex Instruction Set Computer



***Reduced** Instruction Set Computer*

CISC (PC)		RISC (Telemóvel)
Instruções complexas 1 instrução = vários ciclos		Instruções simples 1 instrução = 1 ciclo
Qualquer instrução pode fazer referencia à memória		Apenas LOAD e STORE faz referência à memória
Instruções executadas pelo **Microcode** (exige um tradutor (Assembler) de programação para código máquina) ex: HTML word		Instruções executadas pelo hardware Ex: HTML notepad
Baixa utilização de Pepelining		Alta utilização de Pepelining
Número reduzido de registos		Grande número de registos
Muitas instruções e modos de endereçamento		Poucas instruções e modos de endereçamento
Programação simples – alto nível (Contudo a complexidade está no microcode)		Compiladores complexos (programação complexa – baixo nível)

CISC

Microprocessadores **CISC** (*Complex Instruction Set Computer*) **são fáceis de programar e permitem um uso eficiente de memória.**

Porquê? - **A filosofia CISC surgiu devido à necessidade de criar linguagens de programação mais próximas da linguagem humana**. Noutros tempos as máquinas eram programadas única e exclusivamente em linguagem *Assembly* (linguagem máquina), e as memórias eram lentas e caras, o que justificou a filosofia CISC. Projetos de microprocessadores clássicos, tais como o *Intel 80x86* e o *Motorola 68K series*, seguiram a filosofia CISC.

Mudança recente - Mudanças recentes na tecnologia de software e hardware forçou uma reavaliação em termos de arquitetura. **Assim, muitos processadores CISC mais modernos têm implementado alguns princípios RISC** (*Reduced Instruction Set Computer*), **tornando-se arquiteturas híbridas** mais convenientes.

Características:
Instruções de tamanho variável de acordo com o modo de endereçamento
Instruções que requerem múltiplos ciclos de clock para executar

RISC

- Microprocessadores **RISC** (*Reduced Instruction Set Computer*) são aqueles que **utilizam um pequeno conjunto de instruções altamente otimizado**.
- Os primeiros projetos RISC foram desenvolvidos nos anos 70 e 80 pelas universidades de Stanford e Berkeley, respectivamente.
- Algumas características RISC importantes são:
  - **Execução em apenas um ciclo de *clock***. Esta característica é resultado da otimização de cada instrução, aliada a uma técnica chamada de *Pipelining*;
  - ***Pepelining*** é uma técnica que permite execução simultanea de partes de instruções, tornando o processo mais eficiente;
  - **Grande número de registos** para evitar uma quantidade elevada de interações com a memória.

Pipelining – permite ao processador **guardar num buffer instruções** enquanto uma outra instrução está a ser executada na ALU.

Memory buffer register(MBR) é um registo no processador, CPU, que **armazena os dados que estão ser transferidos de e para a memória**.