Software Básico

Aula #01

Introdução à Disciplina

Quais são os objetivos da disciplina? Qual é o nosso contexto de estudo? Como se dará o andamento da disciplina?

Ciência da Computação - BCC2 - 2023/02
Prof. Vinícius Fülber Garcia

Começando do Começo!

O que é **SOFTWARE BÁSICO**?

Palpites?

Começando do Começo!

Possíveis respostas no contexto da computação:

- Conjunto de programas que definem padrões comportamentais que tornam um equipamento computacional útil
- Conjunto de programas que dão fornecem funcionalidades-base para a utilização e funcionamento de um computador, o tornando operável

SOFTWARE BÁSICO É, ENTÃO, DIFERENTE DE SOFTWARE APLICATIVO

Escopo da Disciplina

Para começar...

- Exercitar e visualizar, na prática, conceitos de organização de computadores e sistemas computacionais
- Compreender o que é uma linguagem de montagem (*assembly*) e como ela relaciona linguagens de alto nível e linguagem de máquina

FIM DA PRIMEIRA PARTE DA DISCIPLINA

Escopo da Disciplina

Em seguida...

- Compreender as características de
  - programas de base (compilador, montador, ligador, interpretador)
  - formatos de arquivo (fonte, objeto e executáveis)
  - tipos de bibliotecas (estáticas, compartilhadas e dinâmicas)
  - funcionamento de programas em tempo de execução

FIM DA SEGUNDA PARTE DA DISCIPLINA

Escopo da Disciplina

Tecnicamente, no decorrer da primeira parte da disciplina iremos:

- Traduzir programas em C para programas em *assembly* (AMD64)
- Montar e ligar programas em *assembly*
- Analisar a execução dos programas gerados em memória

Da mesma forma, na segunda parte da disciplina vamos:

- Analisar o formato de arquivos fonte
- Analisar o formato de arquivos objeto
- Analisar o formato de arquivos executáveis

Relembrando Fundamentos

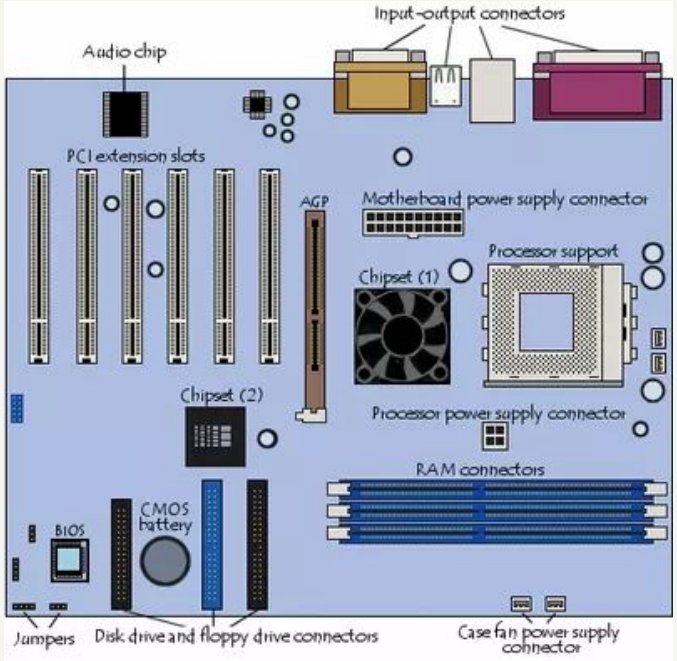
O que é um **COMPUTADOR**?
(de maneira geral)

O que é um **COMPUTADOR MODERNO**?
(quais suas necessidades fundamentais?)

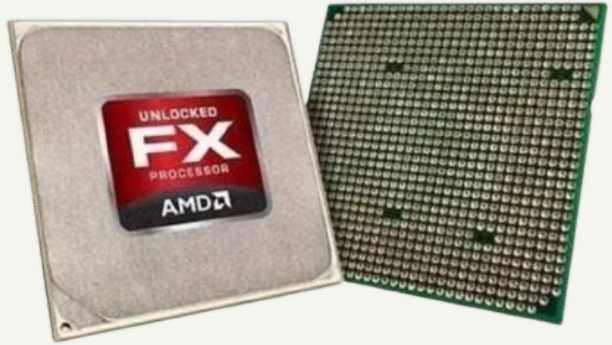
Relembrando Fundamentos

O computador moderno:

- Processador
- Memória
- Dispositivos de E/S
- Placa mãe
  - Organização
  - Barramento



Relembrando Fundamentos



Os pinos da CPU se conectam aos *plugs* do *slot* de CPU da placa mãe...

O QUE PODEMOS CONCLUIR DISSO?

Relembrando Fundamentos

As conexões da memória devem encaixar no *slot* da placa mãe...

O QUE PODEMOS CONCLUIR DISSO?

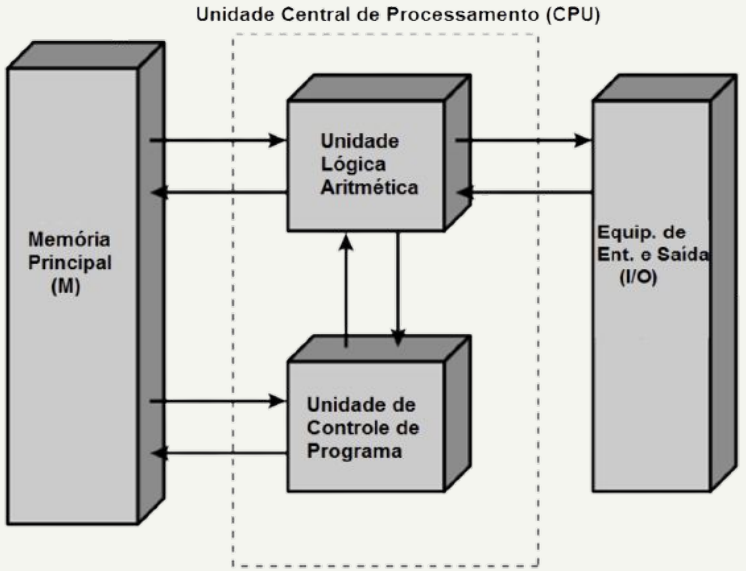
Relembrando Fundamentos

As conexões dos dispositivos externos se conectam em diferentes *slots* da placa mãe...

O QUE PODEMOS CONCLUIR DISSO?

Relembrando Fundamentos

Porém, para simplificar a representação de um computador moderno, podemos adotar uma abstração:



Qual é o nome dessa abstração?

Relembrando Fundamentos

O modelo Von Neumman é bastante simples em termos de blocos de construção; mas é isso que o torna genial!

Mas quem é Von Neumman? O que significa esse modelo?

Para o nosso momento atual, o que é mais importante saber é:

- Um programa em execução deve estar carregado em memória
- Existe um ponteiro que aponta a próxima instrução a ser executada

Relembrando Fundamentos

Mas, existem outros modelos de computadores além do Von Neumman?

Sim! Existem. Mas são praticados apenas em cenários específicos ou experimentais, portanto não estudaremos eles

Apenas por curiosidade:

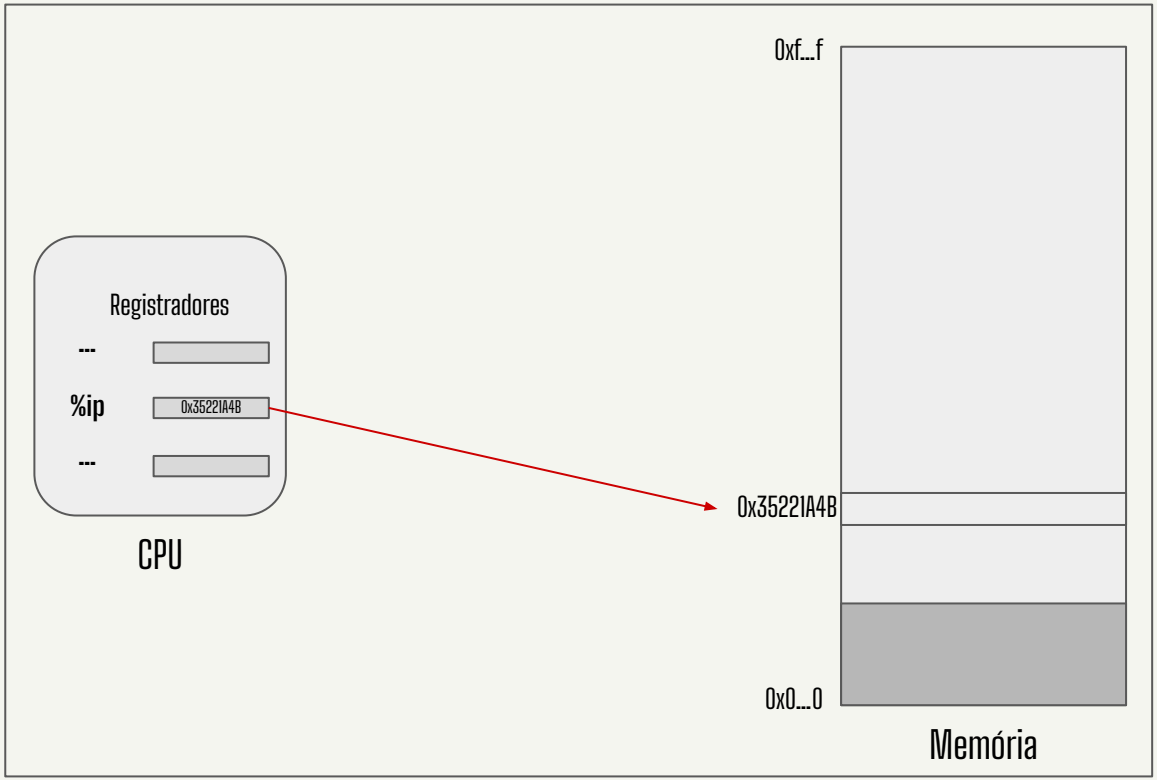
- Modelo de Harvard
- Modelo SIMD/MIMD
- Modelos Neuromórficos

Relembrando Fundamentos

O registrador de instrução, na arquitetura x86_64 é chamado de "ip" (*instruction pointer*)!

Em outras arquiteturas pode ser conhecido como "pc" (*program pointer*).

Este registrador armazena um endereço de memória com a próxima instrução a ser executada.

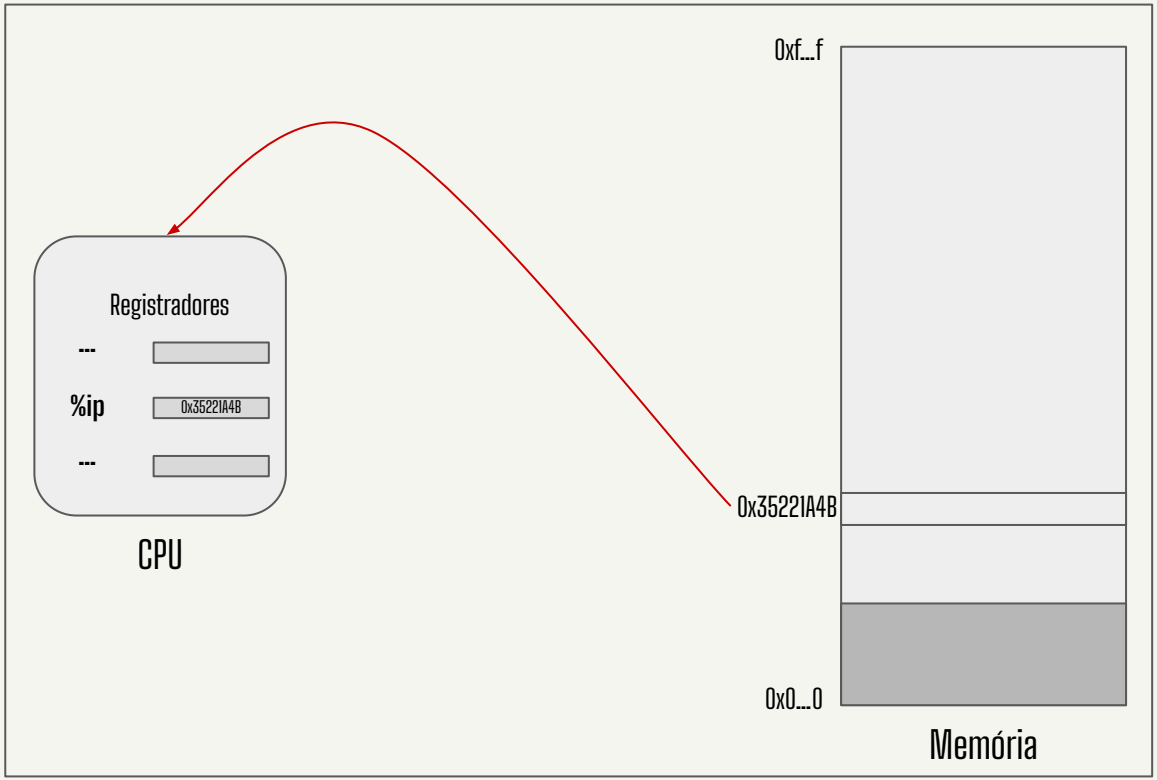


Relembrando Fundamentos

A instrução apontada pelo registrador “ip” é buscada e carregada na CPU.

A instrução é, então, decodificada:

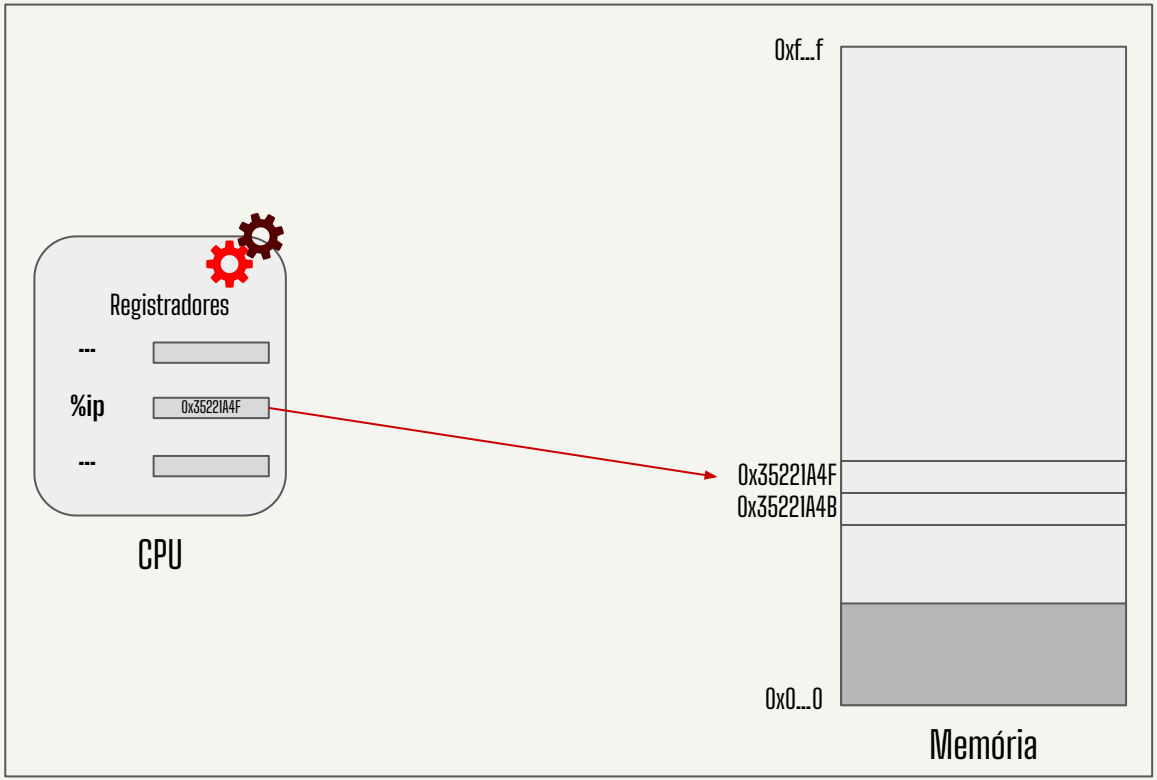
- Cód. de operação
- Operandos
  - Parâmetros



Relembrando Fundamentos

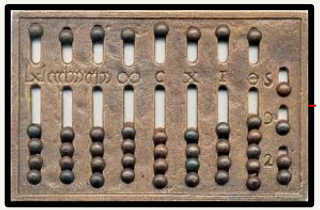
A operação carregada e decodificada é então **executada pelo processador**.

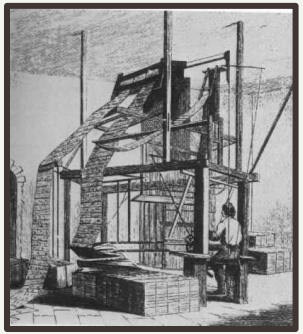
Após a conclusão da execução, o registrador "ip" é **atualizado com o endereço da próxima instrução**.



Relembrando Fundamentos

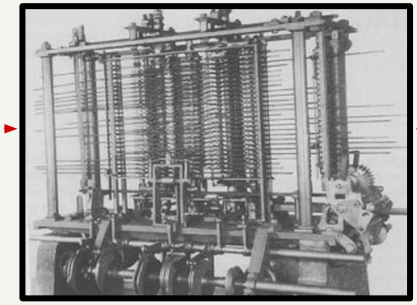
Ábaco (3500 A.C.)





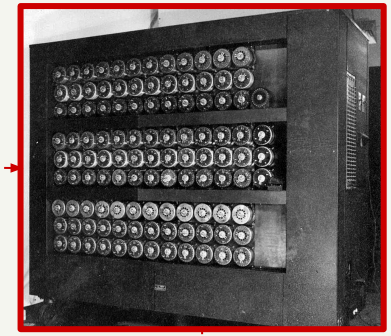
Tear de Jacquard
(1801)

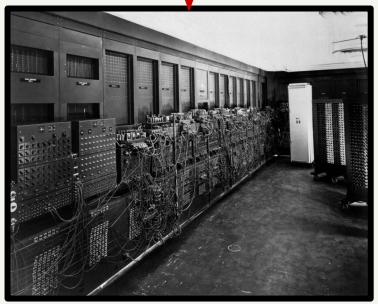
Máquina Análitica de
Babbage (1834)



Relembrando Fundamentos

Bombe (1940)





ENIAC (1945)

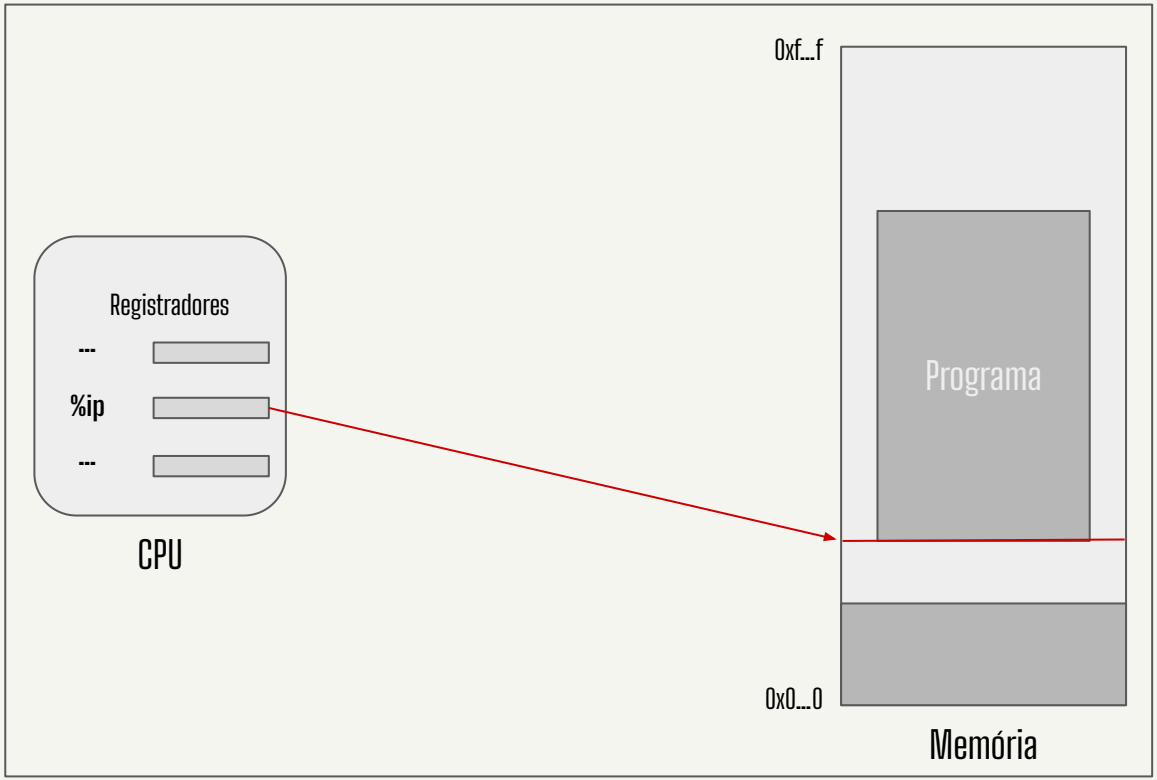
- O computador chamado **Bombe** é considerado o primeiro a ser construído seguindo o modelo Von Neumann
  - Decifrava o Enigma, código de comunicação das forças alemãs durante a Segunda Guerra Mundial
  - Eletromecânico
- O **ENIAC** foi o primeiro modelo “““popular”””

Relembrando Fundamentos

Na geração do Bombe e do ENIAC, **apenas um programa** poderia estar em memória por vez (*batch*).

A distinção de entre instruções e dados era difusa.

Toda a espera por **I/O era OCIOSA**!



Relembrando Fundamentos

A próxima geração de computadores então solucionou alguns desafios da geração anterior. Sendo esta conhecida como **GERAÇÃO MULTIPROCESSO**

A espera ociosa, na geração anterior, era tão danosa, que um programa inteiro poderia ser executado no meio tempo.

A solução? Criar um “gerente” de processos para executar programas com aparente simultaneidade!

SISTEMA OPERACIONAL

Relembrando Fundamentos

Uma linha histórica de importantes sistemas operacionais:

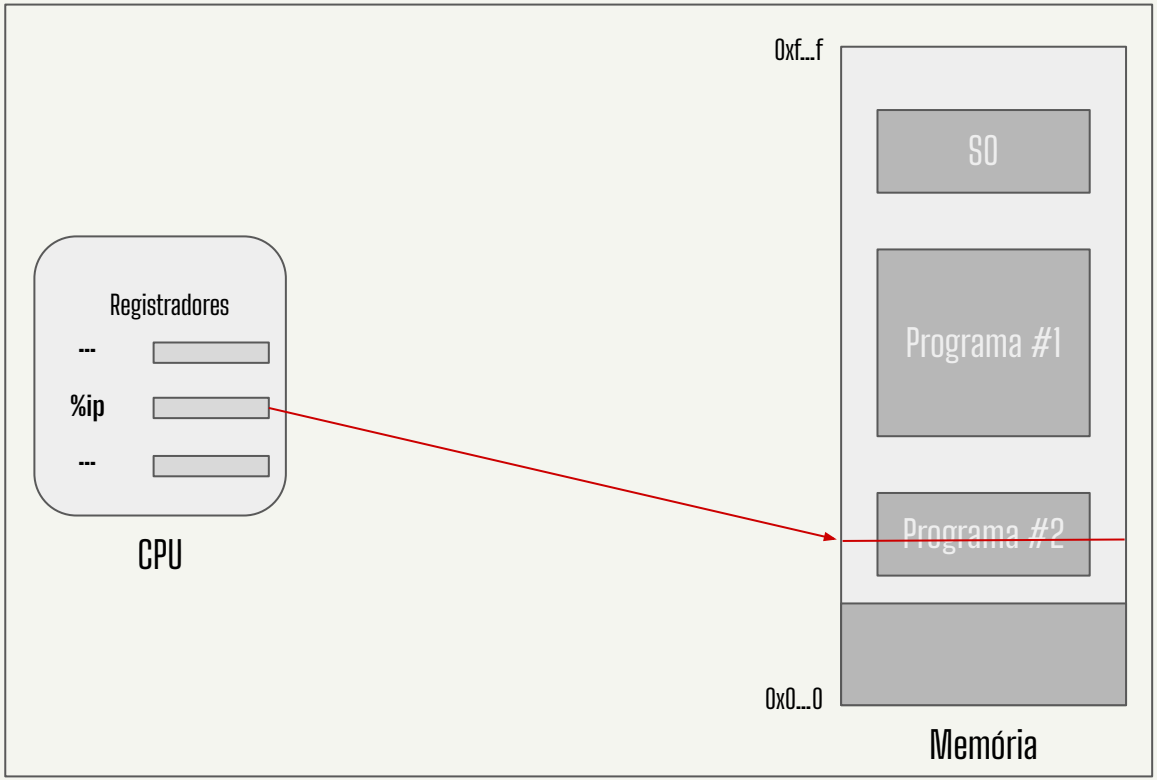
- 1956: GM-NAA I/O (gerenciador de memória IBM)
- 1960: OS-360 (sistema operacional do IBM-360)
- 1967: Multics (primeiro sistema de tempo compartilhado nativo)
- 1969: Unics (precursor do Unix)
- 1981: MS-DOS (precursor do Windows)

Relembrando Fundamentos

Apenas um programa está em execução em um dado momento.

Cada programa ganha um determinado tempo de execução na CPU.

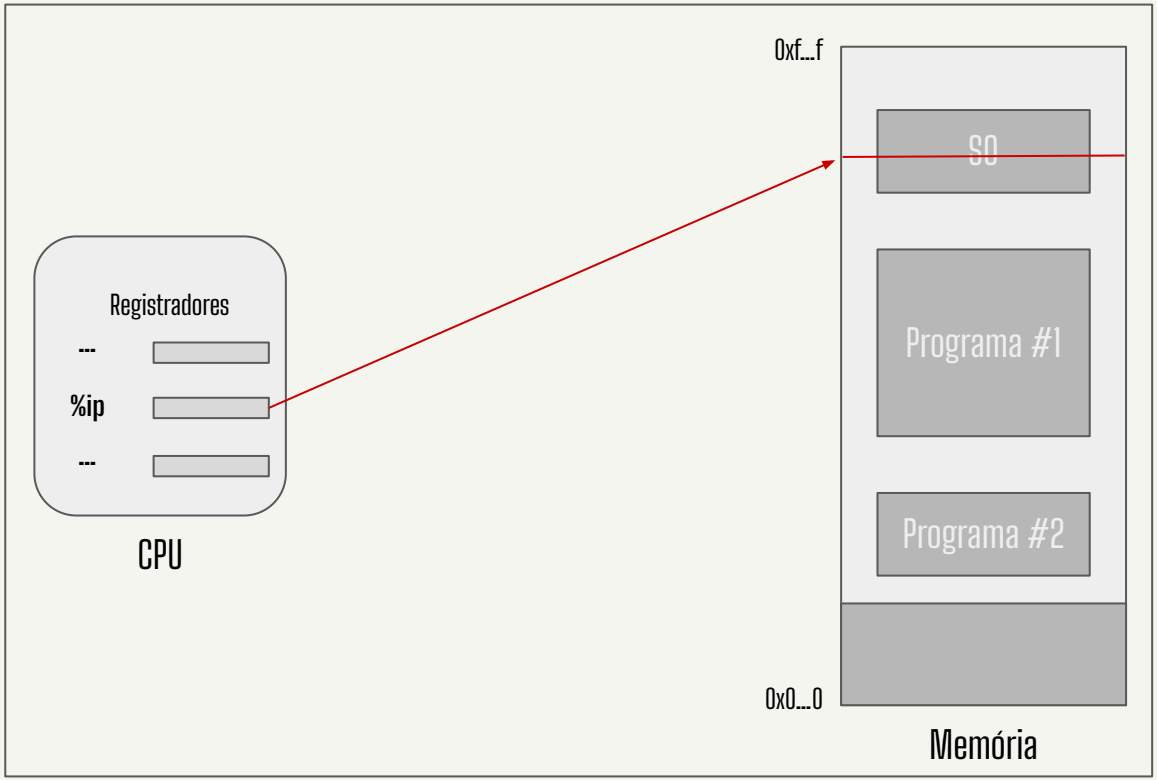
Durante tal tempo, o programa é normalmente executado, instrução a instrução.



Relembrando Fundamentos

Quando o tempo dedicado ao programa acaba, a CPU executa o **sistema operacional**.

O sistema operacional então **decide qual é o próximo programa** a ser executado.

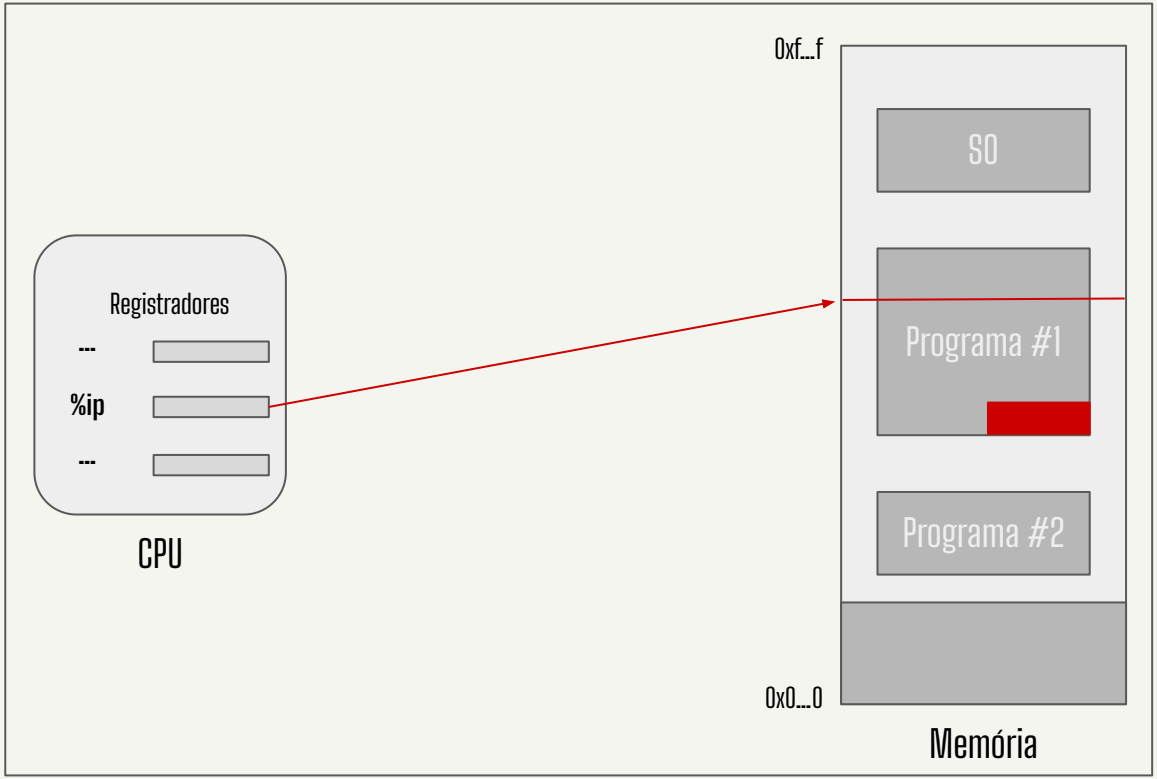


Relembrando Fundamentos

Uma vez decidido **o novo programa a ser executado**, um tempo de CPU é dedicado ao mesmo.

Metadados do programa selecionado são lidos e processados.

O novo programa é colocado em execução.

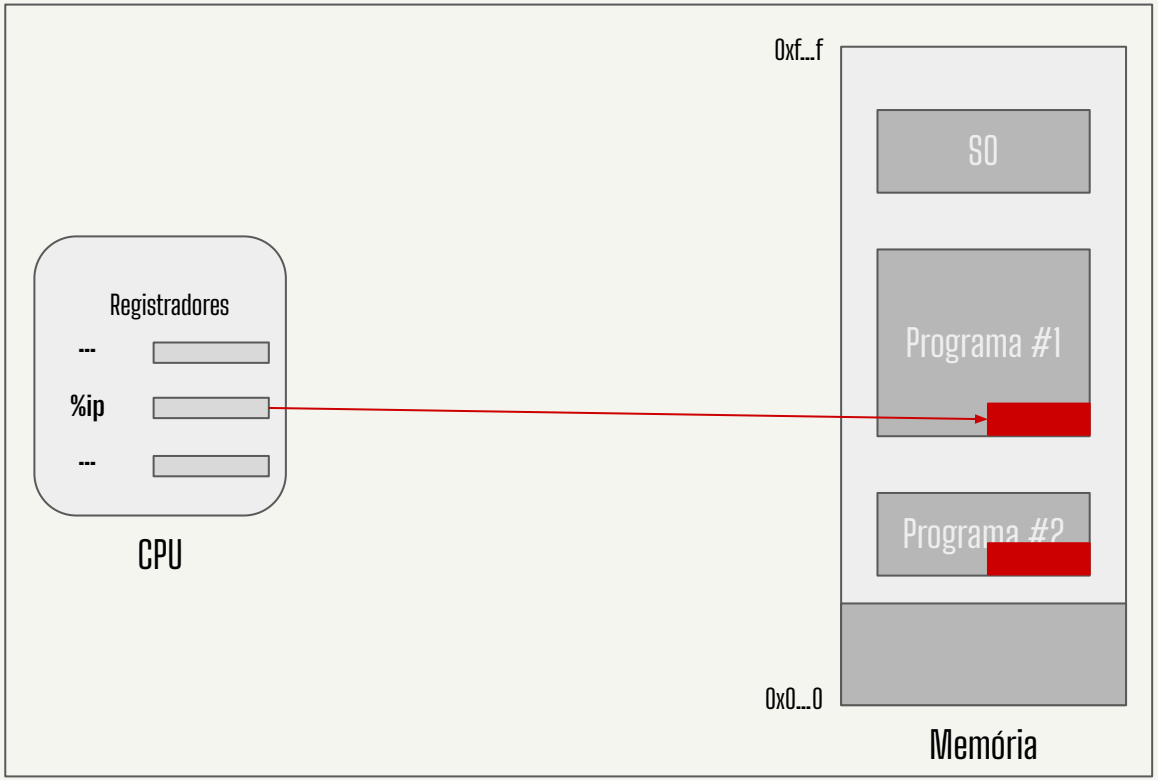


Relembrando Fundamentos

Os metadados do programa agregam uma série de informações importantes que permitiam resumir a sua execução.

Por exemplo, é necessário determinar a próxima linha do programa a ser executada.

Todos os programas guardam tais metadados.

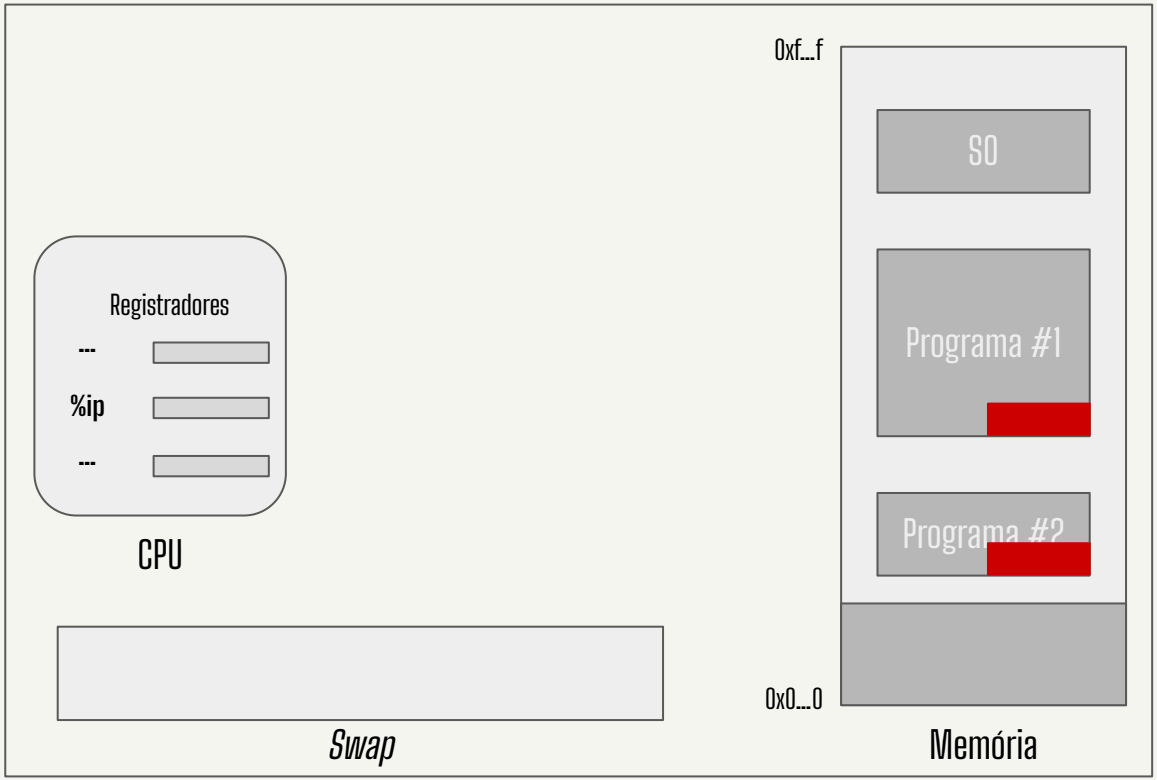


Relembrando Fundamentos

Esses dados estão organizados em seções.

Com o tempo, uma série de seções foram alocadas nos programas de forma a permitir o SO a realizarem processo complexos.

Por exemplo, a troca de processos utilizando uma região de memória secundária (*swap*).

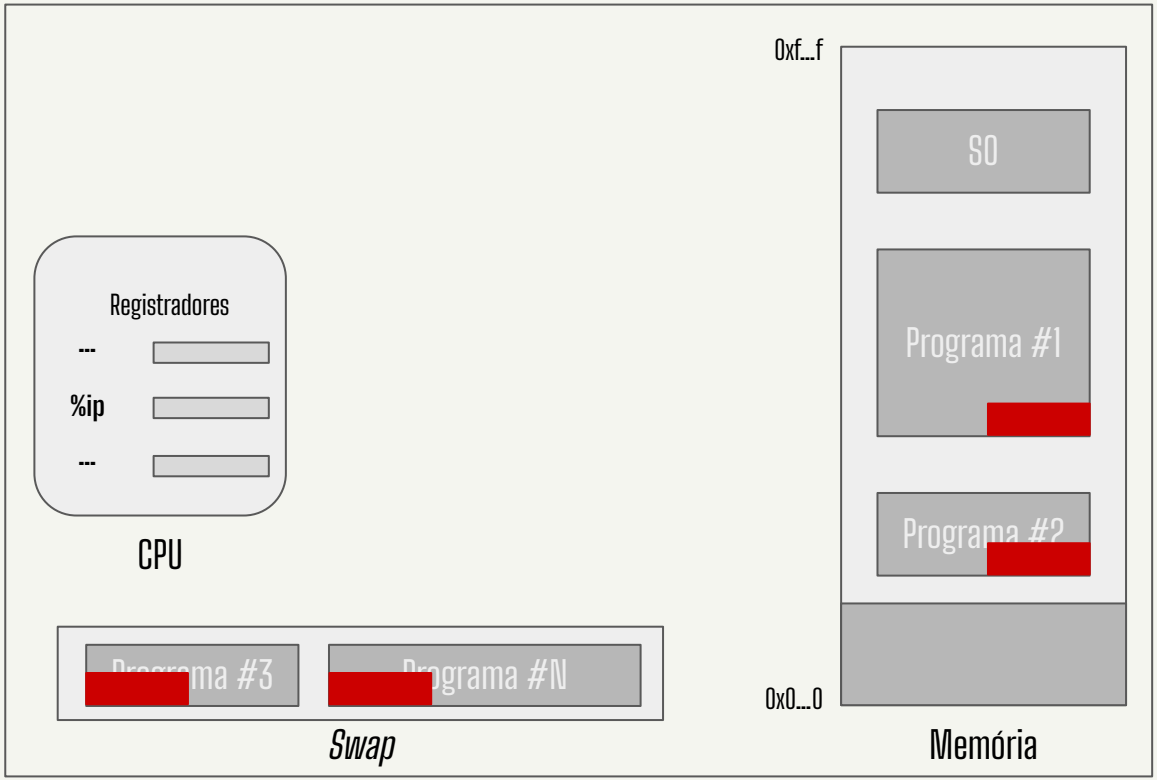


Relembrando Fundamentos

Agora, programas que excediam a memória principal poderiam ser alocados, **enquanto não estavam em execução**, na área de *swap* (memória secundária).

Alguns metadados:

- Endereços de cópia
- Valor de registradores
- Etc...



Relembrando Fundamentos

Porém, a existência de programas está intimamente relacionada com a possibilidade de expressão das operações que devem ser executadas em um computador.

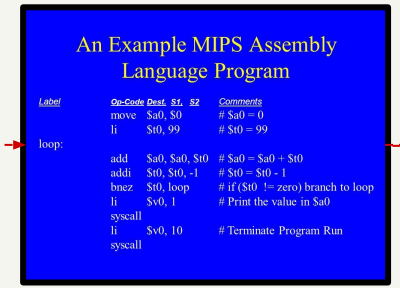
Para isso, se faz necessária uma ferramenta de comunicação entre programadores e computador:

LINGUAGENS DE PROGRAMAÇÃO

Relembrando Fundamentos

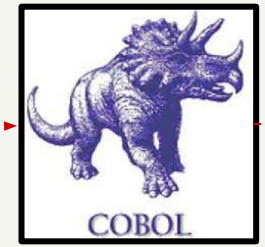
Plugboards
(mapa de bits)

Assembly
(símbolo mnemônicos)



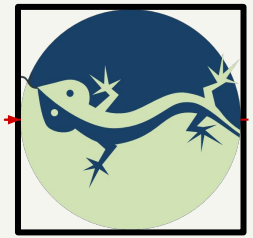
Código e variáveis globais

FORTRAN e COBOL



Código e variáveis globais

LISP



Alocação dinâmica
De memória

Algol

Procedimentos

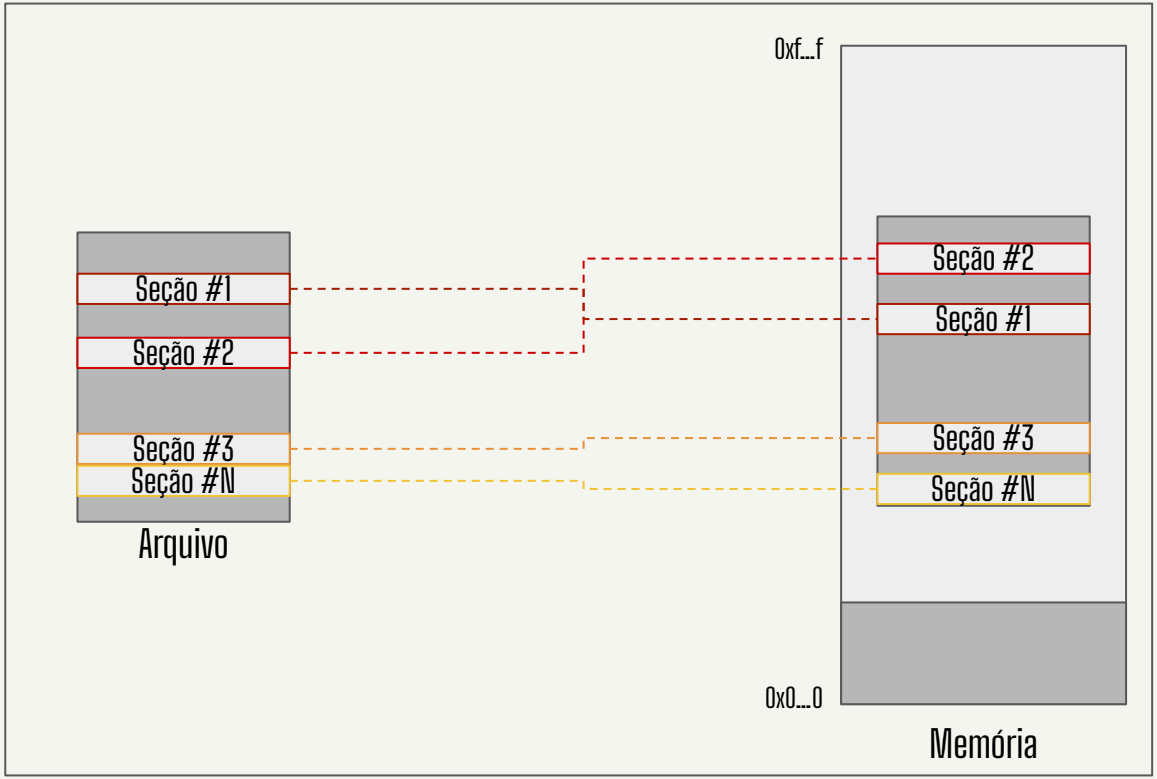
Relembrando Fundamentos

Em um arquivo de programa, existem:

- Seções mapeadas em memória
- Seções apenas do arquivo
- Seções apenas da memória

Também:

- Carregador: mapeamento
- Processo: programa em execução



Materiais

- Livro de referência
  - Execução de Programas: Como funcionam carregadores, ligadores e interpretadores. Bruno Müller Junior. Atualização de 2019. (https://www.inf.ufpr.br/bmuller/assets/material/livroSB.pdf).
- Recursos técnicos
  - Ambiente Linux
  - GCC
    - as

Avaliação

- Duas provas e um trabalho prático
  - Prova #01: primeira parte da disciplina
    - *Assembly*
  - Prova #02: segunda parte da disciplina
    - Formatos de arquivos
  - Trabalho: ????
    - Programação e apresentação
- Média aritmética
- Prova final (exame)

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius/
viniciusfulber@ufpr.br