Estrutura do tema ISC

1. Representação de informação num computador
2. Organização e estrutura interna dum computador
3. Execução de programas num computador
4. O processador e a memória num computador
5. Da comunicação de dados às redes

Níveis de abstracção:

- nível das linguagens HLL (*High Level Languages*): as linguagens convencionais de programação (puro texto)
  - » imperativas e OO (Basic, Fortran, C, Java, ...)
  - » funcionais (Lisp, Haskell, ...)
  - » lógicas (Prolog, ...)
- nível da linguagem máquina: a linguagem de comandos, específica para cada CPU ou família de CPU's (em binário)
  - » arquitecturas CISC (*Complex Instruction Set Computers*)
  - » arquitecturas RISC (*Reduced Instruction Set Computers*)
- nível da linguagem *assembly* (de "montagem"): linguagem intermédia (comandos do CPU em formato texto)


```
int x = x+y;
```

- Código C
  - somar 2 inteiros (c/ sinal)

```
addl 8(%ebp),%eax
```

Idêntico à expressão `x = x + y`

- *Assembly*
  - somar 2 inteiros de 4-*bytes*
    - operandos "*long*" em GCC
    - a mesma instrução, c/ ou s/ sinal
  - operandos:
    - x: em registo %eax
    - y: na memória M[%ebp+8]

```
0x401046:    03 45 08
```

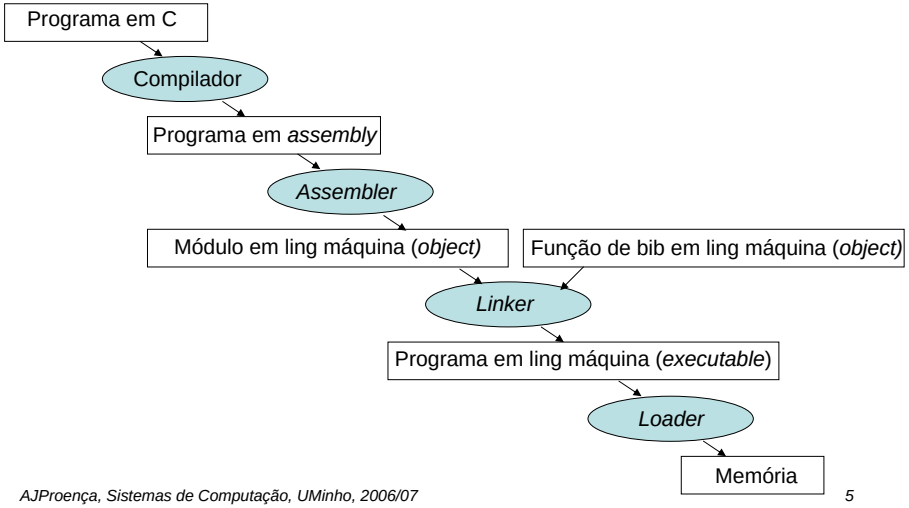
- Código *object*
  - instrução com 3-*bytes*
  - na memória em `0x401046`

Mecanismos de conversão (para comandos do CPU):

- compilador
  - traduz um programa de um nível de abstracção para outro inferior (converte um ficheiro de texto noutro de texto); por ex., de C para *assembly*
  - por vezes inclui mais que um passo de conversão, até chegar à linguagem máquina
- *assembler* ("montador")
  - "monta" os comandos/ instruções em binário (*object*), de acordo com as regras do fabricante do CPU
- interpretador
  - analisa, uma a uma, as instruções de um programa em HLL, e:
    - » **gera código** em linguagem máquina para essa instrução, **e**
    - » **executa** esse código.

Execução de programas num computador (4)

De um programa em HLL até à sua execução:



Execução de instruções (em linguagem máquina) num CPU

Ciclo de execução de instruções:

- Busca da instrução
  ... e incremento do IP
- Descodificação da instrução
- Execução da operação
  - cálculo da localização do(s) operando(s), e ir buscá-lo(s), se necessário
  - execução da operação especificada
  - guardar resultado, se necessário

Modelo de computação de von Neumann (1945)

Análise de um exemplo: `movl Loc,%eax`

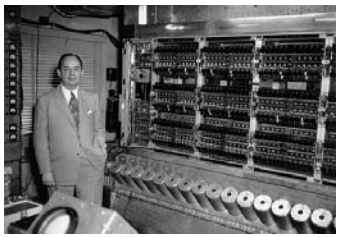
Modelo de computação de von Neumann, 1945/46 (1)

ENIAC (1ª geração, 1945)

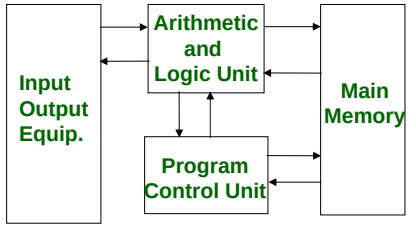
- –objectivo: cálculo tabelas de artilharia
- –máquina decimal
- –18.000 válvulas, 30 ton
- –programação: manual, alterando as conexões (cablagem)

Von Neumann introduz conceito de *stored-program* :

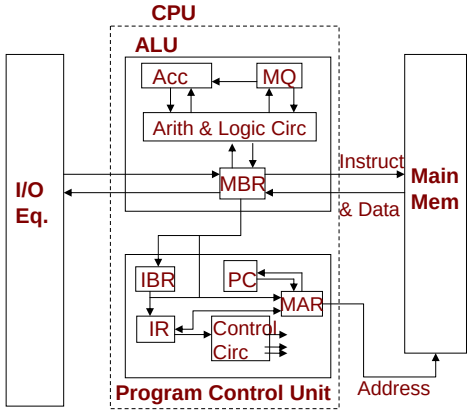
- –dados e instruções em binário, e armazenados numa memória
- –memória acedida pelo endereço da informação
- –execução de instruções de modo sequencial (daí o *Program Counter*, PC), interpretadas pela unid. controlo
- –constrói novo computador, IAS



Modelo de computação de von Neumann, 1945/46 (2)



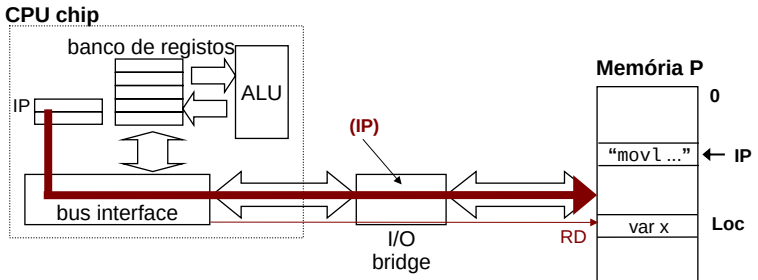
Estrutura básica do IAS
(*Princeton Institute for Advanced Studies*)



Estrutura expandida do IAS

Ex.: `movl Loc,%eax`

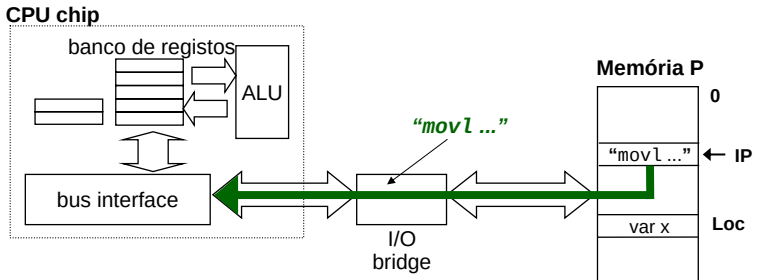
1. Busca da instrução (1)



O CPU coloca o valor do apontador de instruções (IP) no *address bus*, e activa o sinal de controlo RD

Ex.: `movl Loc,%eax`

1. Busca da instrução (2)

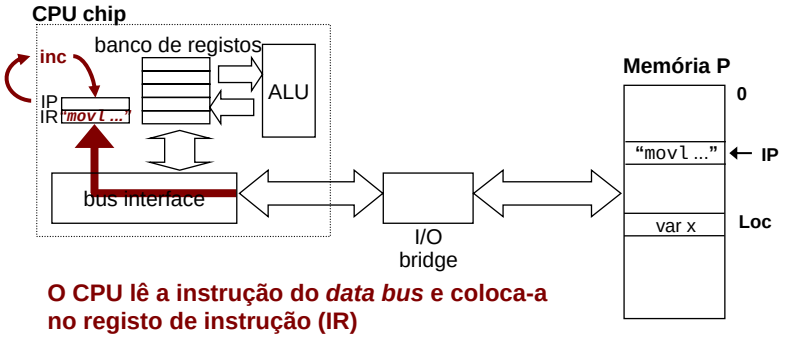


A memória vai buscar a instrução ao endereço definido por IP e coloca-a no *data bus*

Ex.: `movl Loc,%eax`

1. Busca da instrução (3)

... e incremento do IP



O CPU lê a instrução do *data bus* e coloca-a no registo de instrução (IR)

Ex.: `movl Loc,%eax`

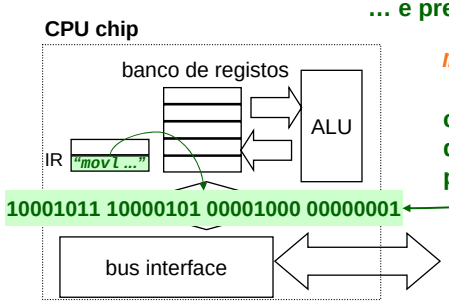
2. Descodificação da instrução

A unidade de controlo do CPU descodifica a instrução…

… e prepara-se para executar a operação:

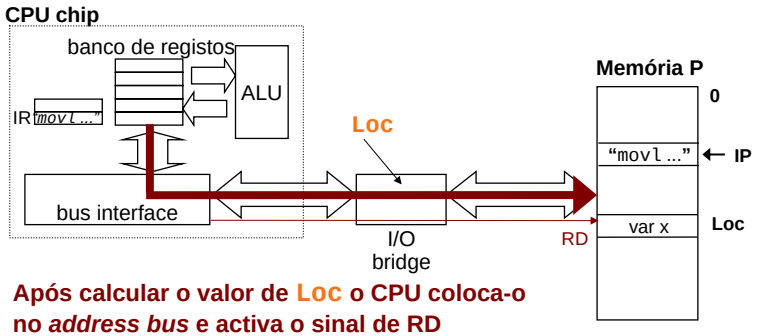
move long

copiar valor com 32 bits da memória, em Loc para o registo %eax



Ex.: `movl Loc,%eax`

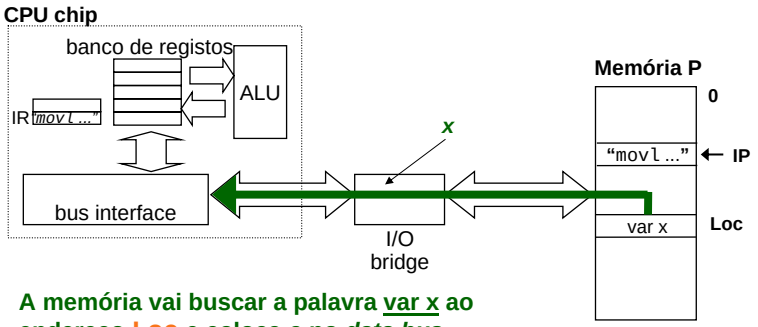
3. Execução da operação (1)



Após calcular o valor de `Loc` o CPU coloca-o no *address bus* e activa o sinal de RD

Ex.: `movl Loc,%eax`

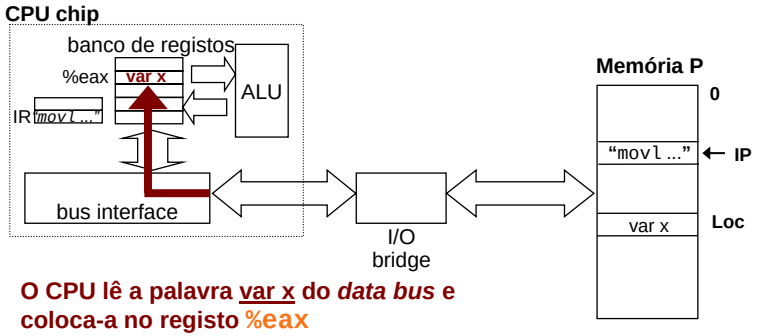
3. Execução da operação (2)



A memória vai buscar a palavra var x ao endereço `Loc` e coloca-a no *data bus*

Ex.: `movl Loc,%eax`

3. Execução da operação (3)



O CPU lê a palavra var x do *data bus* e coloca-a no registo `%eax`

Componentes (físicos) a analisar:

- processador (info adicional):
  - » o nível ISA (*Instruction Set Architecture*): tipos/formatos de instruções, acesso a operandos, ...
  - » paralelismo no CPU: *pipeline*, superescalaridade, ...
  - » CISC *versus* RISC
- hierarquia de memória:
  - *cache*, memória virtual, ...
- periféricos:
  - » interfaces humano-computador (HCI)
  - » arquivo de informação
  - » comunicações